FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Leveraging IEC 61499 for Autonomous Maintenance: Digital Twin Integration and 3D Printing in Distributed Control Systems

Pedro de Sousa Mota



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Rui Pedro Ferreira Pinto, PhD

October 31, 2026

© Pedro de Sousa Mota, 2026

Leveraging IEC 61499 for Autonomous Maintenance: Digital Twin Integration and 3D Printing in Distributed Control Systems

Pedro de Sousa Mota

Mestrado em Engenharia Eletrotécnica e de Computadores

Approved in oral examination by the committee:

President: Prof. Mário Jorge Rodrigues de Sousa, PhD

Referee: Prof. Pedro Alexandre Dias Martins, PhD

Referee: Prof. Rui Pedro Ferreira Pinto, PhD

October 31, 2026

Resumo

A transição para a Indústria 5.0 exige sistemas de produção que não sejam apenas eficientes, mas também resilientes, sustentáveis e centrados no ser humano. A Manutenção Preditiva (PdM) antecipa as falhas dos equipamentos e a Manutenção Prescritiva (PsM) determina a ação corretiva, mas o ciclo de manutenção continua em aberto do lado físico: se a peça de substituição não estiver disponível a tempo, a produção para, por mais exata que seja a previsão. Na literatura analisada, as decisões de prognóstico terminam em recomendações ou em ordens de compra, e a produção de peças sobressalentes por Fabrico Aditivo (AdM) é decidida por planeadores.

Para colmatar esta lacuna, propusemos e implementámos um Sistema de Produção Ciber-Físico (CPPS) que liga, de forma autónoma, a previsão de uma falha à produção da peça de substituição. Um Gémeo Digital (DT) ao nível do sistema, realizado como uma instância por ativo monitorizado, aloja um modelo de prognóstico que estima continuamente o Indicador de Saúde (HI) do ativo, a partir do qual se obtém a sua Vida Útil Remanescente (RUL). Uma regra determinística de PsM desencadeia a produção a pedido da peça por AdM quando a RUL estimada desce abaixo de um limite (*threshold*) definido pelo tempo medido para produzir e instalar a peça, acrescido de uma margem calculada a partir do erro do modelo. O sistema foi implementado como uma aplicação de controlo distribuído segundo a norma IEC 61499 da Comissão Eletrotécnica Internacional, cujos *Function Blocks* (FBs), orientados a eventos, correm em nós de *edge* que comunicam através de *Message Queuing Telemetry Transport* (MQTT) e são supervisionados através de *Open Platform Communications Unified Architecture* (OPC UA).

Avaliámos o sistema num banco de ensaio laboratorial composto por três braços robóticos, uma impressora 3D, nove Raspberry Pi e um Raspberry Pi Pico W. Foram treinados três modelos de prognóstico, um por braço e por sensor de vibração, a partir de um ensaio de degradação acelerada das garras dos braços até à falha; os seus erros aumentaram duas a três vezes quando foram deixadas de fora sessões inteiras. Nos ensaios de ponta a ponta (*end-to-end*), realizados sobre um novo ensaio de degradação em cada braço, a impressora começou a imprimir a peça de substituição entre 207 e 215 ms depois de publicada a ordem de impressão, sem intervenção humana na decisão nem na orquestração, e a peça ficou concluída entre 57 e 73 minutos antes da falha completa da garra, a tempo de ser instalada. As margens obtidas ficaram próximas das pretendidas, tendo ficado 5 minutos abaixo num dos braços, e deveram-se em parte a garras que se degradaram mais lentamente do que no ensaio de treino.

Estes resultados mostram que a PdM, a PsM alojada num DT, o controlo distribuído e o AdM podem ser integrados num único ciclo de manutenção em funcionamento, com hardware de baixo custo, em que a intervenção humana fica reservada à instalação da peça, e que o principal obstáculo ao seu funcionamento fiável não está na integração, mas no momento da decisão sob incerteza. Com um ensaio por braço, ficou demonstrada a viabilidade do ciclo, mas ainda não a sua fiabilidade.

Palavras-chave: Indústria 5.0, Sistemas de Produção Ciber-Físicos, Manutenção Preditiva, Gémeo Digital, Manutenção Prescritiva, IEC 61499, Fabrico Aditivo, Controlo Distribuído.

Abstract

The transition towards Industry 5.0 (I5.0) demands manufacturing systems that are not only efficient but also resilient, sustainable, and human-centric. Predictive Maintenance (PdM) anticipates equipment failures, and Prescriptive Maintenance (PsM) determines the corrective action, but the maintenance loop remains open on its physical side: if the replacement part is not available in time, production stops, however accurate the prediction. In the reviewed literature, prognostic decisions end in recommendations or purchase orders, and the production of spare parts by Additive Manufacturing (AdM) is decided by planners.

To address this gap, we proposed and implemented a Cyber-Physical Production System (CPPS) that autonomously links the prediction of a failure to the production of the replacement part. A system-level Digital Twin (DT), realized as one instance per monitored asset, hosts a prognostic model that continuously estimates the Health Index (HI) of the asset, from which its Remaining Useful Life (RUL) is derived. A deterministic PsM rule triggers the on-demand production of the part by AdM when the estimated RUL falls below a threshold set to the measured time needed to produce and install it, plus a margin derived from the error of the model. The system is implemented as an International Electrotechnical Commission (IEC) 61499 distributed control application, whose event-driven Function Blocks (FBs) run on edge nodes that communicate through Message Queuing Telemetry Transport (MQTT) and are supervised through the Open Platform Communications Unified Architecture (OPC UA).

We evaluated the system on a laboratory testbed comprising three robotic arms, a Three-Dimensional (3D) printer, nine Raspberry Pi devices, and a Raspberry Pi Pico W. Three prognostic models, one per arm and vibration sensor, were trained on an accelerated run-to-failure of the grippers of the arms; their errors increased two- to threefold when whole sessions were held out. In end-to-end trials on one new degradation run per arm, the printer started printing the replacement part 207 to 215 ms after the print order was published, with no human intervention in the decision or the orchestration, and the part was completed 57 to 73 minutes before the complete failure of the gripper, in time to be installed. The margins obtained were close to those intended, falling 5 minutes short for one arm, and were partly owed to grippers that degraded more slowly than in the training run.

These results show that PdM, DT-hosted PsM, distributed control, and AdM can be integrated into a single, operating maintenance loop on low-cost hardware, in which human involvement is reserved for the installation of the part, and that the main obstacle to its reliable operation lies not in the integration, but in the timing of the decision under uncertainty. With one run per arm, the feasibility of the loop has been demonstrated, but not yet its reliability.

Keywords: Industry 5.0, Cyber-Physical Production Systems, Predictive Maintenance, Digital Twin, Prescriptive Maintenance, IEC 61499, Additive Manufacturing, Distributed Control.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to address pressing global challenges, including environmental degradation, industrial resilience, and decent work[1].

This dissertation aligns with the Industry 5.0 paradigm by linking the prediction of a failure to the on-demand production of the replacement part by Additive Manufacturing (AdM), within a distributed Cyber-Physical Production System (CPPS). By distributing the control and decision logic across edge devices, and by allowing a replacement part to be kept as a digital file and printed only when a failure is predicted, this work contributes to three goals: **SDG 8** (Decent Work and Economic Growth), **SDG 9** (Industry, Innovation and Infrastructure), and **SDG 12** (Responsible Consumption and Production). Since the system was demonstrated on a laboratory prototype, these contributions are potential, and the table below separates what was shown in this work from what was not measured.

SDG	Target	Contribution	Evidence in this work
8	8.2	Automates the monitoring of equipment health and the decision to produce a replacement part, leaving human operators in supervisory and installation roles.	Print orders issued without human intervention in the decision in all end-to-end trials.
9	9.4	Distributes control and decision across IEC 61499 edge devices, so that the monitoring of each asset does not depend on a central server; the shared printer remains a single resource.	All production cycles completed and no sensor packets lost in one hour of operation.
	9.5	Relies on low-cost hardware (Raspberry Pi) and open-source software, lowering the barriers to adoption for small and medium-sized enterprises.	Open-source runtime, development environment, and print server; hardware cost not assessed.
12	12.5	Allows replacement parts to be kept as digital files and printed on demand, which can reduce physical stock, obsolete parts, and transport.	One part printed per predicted failure; savings in material, storage, and emissions not measured.

[1] https://sdgs.un.org/goals

Acknowledgements

Como não poderia deixar de ser, é precisamente no momento no qual tenho adormecido a pensar, virtualmente todos os dias durante dezoito meses, que me faltam as palavras. E esse momento está longe de ser apenas a entrega desta dissertação. Muito pelo contrário. É, de uma maneira ou de outra, o fim daquilo que foi a minha vida nos últimos seis anos. O fim de uma longa, dura, complicada, mas, sem dúvida, bonita jornada. E se hoje posso apelidar esta jornada de "bonita", é com certeza graças às pessoas que, de mais perto ou mais longe, a fizeram comigo.

Em primeiro lugar, agradeço ao SYSTEC e ao DIGI2. Agradeço ao Professor Doutor Gil Gonçalves, pela oportunidade que me ofereceu e por, pela segunda vez, me ter salvado de uma situação deveras desesperadora. Agradeço aos Engenheiros Luís Portela e ao Diogo Gomes por, reiteradamente, ao longo deste percurso, responderem aos meus pedidos de ajuda o melhor que conseguiam, e pelas ocasionais palavras de encorajamento.

Ao Professor Doutor Rui Pinto, agradeço por um trabalho de orientação exímio. Agradeço por toda a confiança que depositou em mim; por em nenhum momento ter desistido de mim, mesmo quando eu próprio estava quase a fazê-lo; por estar sempre presente quando precisei, pelo seu rigor, por sempre se ter esforçado para me ajudar.

Aos meus pais, Ricardo e Susana, sem os quais eu garantidamente não estaria aqui hoje, obrigado. Por tudo. Por sempre me terem incentivado a seguir aquilo em que acreditava e sonhava; pelo vosso apoio incondicional e inabalável, não importando em que circunstâncias nos encontrássemos; e, mesmo que vos tenha custado, agradeço por me terem deixado voar, sempre sabendo que teria para onde voltar.

À minha irmã, Margarida, agradeço por acreditares em mim seja em que situação for. Verte crescer tem sido um dos maiores orgulhos da minha vida e, quanto mais crescidos ficamos, mais tenho a certeza de que a ligação que temos nunca vai desaparecer. Não importa o tempo que passe, sei que, se te vir ao meu lado naqueles jantares de família críticos, vou ter sempre um ombro amigo, mesmo que às vezes passem dois meses sem sequer nos cruzarmos em casa.

À minha avó Madalena, agradeço por, desde que eu me conheço, seres a minha maior fã. Nunca vi ninguém tão seguro como tu de que eu iria ter sucesso, seja no que fosse que eu quisesse fazer. E se em alguns momentos fraquejei, sabe que, o medo de tu um dia não me poderes ver a completar fases como esta, me deu forças para continuar.

Ao meu avô João, agradeço por ter criado o Engenheiro que existe mim. Se desde sempre tenho o interesse e uma teimosia insuportável por descobrir como tudo funciona, é graças ao homem que metia o Pedro de 5 anos com uma furadora na mão, ou em cima de um martelo pneumático, ou a plantar batatas só de boxers.

À minha restante *Happy Family*: Elisa, Diana, Bruno, Mariana, Eduardo, Sofia e Maria, obrigado por fazerem parte da minha vida, e por terem sido, muitas vezes, um escape num dia aleatório por mês que me fazia melhor do que o que eu se calhar me apercebia.

À Rosa Almeida, minha tiazinha, obrigado por seres um raio de sol em todos os sítios onde estás. Obrigado pelo teu apoio incondicional e por teres feito parte deste percurso e de um outro, bem maior.

À Inês Meireles, não podia ter tido alguém melhor para ter ao meu lado quando viemos os dois aqui parar, apesar de na altura não fazer ideia de tal. Caímos aqui de paraquedas, num ano terrível,

sem sabermos para onde nos virarmos, mas pelo menos tive sempre a garantia que estavamos juntos. Obrigado.

À minha segunda família: Bronquite, Cavalete, Gemas, Marquise, Pisca-Pisca, Poeta Silencioso, Reforço O Soldado de Inverno, Tou Não Tou. Ao eterno Bando de mal-formados, o meu mais profundo obrigado. Olho para trás e poucas são as memórias, desde que entrei nesta Faculade, das quais vocês não fazem parte. Não existe pensar neste percurso sem pensar em vocês. Tiago, não existem memórias onde eu me tenha divertido e rido tanto como tangentes absurdas contigo. Não importa o que aconteça, sei que estarás sempre lá com um sorriso na cara e para me dar um beijinho quando chegas. Eduardo, não existe pensar nestes 6 anos sem andar às turras contigo por tudo e por nada, porque no fundo, eu até gosto um bocadinho de ti. Rita, não há ninguém com quem eu menos me importe de ser cancelado do que quando falo do que quer que seja contigo. Não há um assunto que não seja divertido de ficar a falar contigo durante horas. Pedro, não houve um momento que não tenha sido melhor apenas por tu estares presente. Catarina, não houve uma memória que se tenha tornado icónica que não tenha tido o teu dedo. Mas, acima disso, és mesmo uma amiga de todas as horas, e não podia ter pedido melhor parceira em tantas situações. Gonçalo, não há situação má que a tua leveza com a vida não me tenha arrancado um riso estúpido de dentro. Beatriz, não há nada para que tu me chames que eu não vá querer dizer que sim. E acredita, não há nada absurdo demais que possas dizer que me fará gostar menos de ti. Leo, não há ninguém tão genuinamente amoroso, sem nunca esperar algo em troca, como tu. Obrigado por existirem. Simplesmente. Aos que já são adultos *premium*, espero que nos continuemos a encontrar e a manter a ligação que temos. Aos que ainda ficam por cá, não pensem que se livram de mim assim tão facilmente, que eu só descanso quando tiverem todos o canudo na mão. Ao Poetxinha, é a nossa vez, amigo.

À Marta, meu *mor* primogénito, obrigado pela amizade improvável mais provável de sempre. Agradeço a Oulu—e companhia—por nos ter juntado, mas agradeço acima de tudo a ti, por nunca teres partido. Sabe que tornaste inexistente qualquer migalha de arrependimento sobre ir de ERASMUS que alguma vez possa ter existido em mim.

Ao XauXiXas, agradeço por me teres mostrado que existe sempre um lado positivo em tudo. Por me teres mostrado que há coisas pelas quais ainda vale a pena continuar a tentar. Por me teres aberto sempre as portas de tua casa, conotativa e figurativamente. Obrigado, Tiago. Ao Machachuxas, O Analfabeto, agradeço por teres sido o exemplo que foste, e és, para mim, durante todos estes anos. Por teres sempre uma palavra de apoio, mesmo ao teu jeito desengonçado, porque é isso que te torna especial, Pedro Magano.

Ao José Pedro, agradeço por uma amizade que começou num piscar de olhos e nunca mais se perdeu. Ao José Miguel e à Inês, agradeço por me relembrarem que nada é nunca assim tão sério. À Maria, agradeço por tudo. Por estares sempre lá, seja em que situação for.

Consequentemente, não dá para evitar a brincadeira que me permitiu ter tanta gente especial na minha vida. Por muitas dores de cabeça que me dê, é graças a pelicano-pinguins asiáticos, a uns leõzinhos meio estranhos, a mutações genéticas de bebés com ratos e esquilos, a cães de três cabeças e a personagens ficcionais criadas por aquela que não deve ser mencionada, que estes seis anos foram menos aborrecidos.

Ao Diogo Matos, por me teres mostrado o quão boa a vida pode ser quando gostamos mesmo de alguém.

Por fim, um obrigado a todos por terem feito parte deste percurso, por fazerem parte da minha vida, e por me tornarem numa pessoa melhor.

Pedro de Sousa Mota

"Xanax"

“BASTA!”

Francisco "Kiko is Hot" Soares

Contents

Resumo i

Abstract ii

UN Sustainable Development Goals iii

Acknowledgements iv

List of Acronyms and Symbols xiv

1 Introduction 1
- 1.1 Context and Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Objectives . . . 2
- 1.4 Publication and Code Availability . . . 3
- 1.5 Document Structure . . . 3

2 Background and Fundamental Aspects 5
- 2.1 Industrial Context: From Industry 4.0 to Industry 5.0 . . . 5
- 2.2 Maintenance Strategies and Terminology . . . 6
  - 2.2.1 Maintenance Types . . . 6
  - 2.2.2 Autonomous Maintenance . . . 8
- 2.3 Prognostics and Remaining Useful Life Estimation . . . 8
  - 2.3.1 Prognostics and Health Management . . . 8
  - 2.3.2 Health Indicators and Health Stages . . . 9
  - 2.3.3 Prognostic Approaches . . . 10
  - 2.3.4 Uncertainty and Evaluation of Prognostics . . . 11
  - 2.3.5 From Prediction to Timely Action . . . 11
- 2.4 From Prediction to Decision: Prescriptive Maintenance and Autonomy . . . 12
  - 2.4.1 From Predictive to Prescriptive Analytics . . . 12
  - 2.4.2 Prescriptive Maintenance . . . 13
  - 2.4.3 Automation, Autonomy, and the Role of the Human . . . 14
- 2.5 Cyber-Physical Systems, Cyber-Physical Production Systems and Digital Twins . . . 16
  - 2.5.1 Cyber-Physical Systems and Cyber-Physical Production Systems . . . 16
  - 2.5.2 The Digital Twin: Origins and Definitions . . . 18
  - 2.5.3 Levels of Integration and Direction of Data Flow . . . 19
  - 2.5.4 Scope: What a Digital Twin Represents . . . 20
  - 2.5.5 Digital Twins in Maintenance . . . 21
  - 2.5.6 Definition Adopted in this Dissertation . . . 22
- 2.6 Distributed Control and IEC 61499 . . . 23
  - 2.6.1 Centralized and Distributed Control . . . 23
  - 2.6.2 From IEC 61131-3 to IEC 61499 . . . 24

2.6.3 The Function Block Model . . . 25
2.6.4 Applications, Devices, and Resources . . . 26
2.7 Industrial Communication . . . 27
2.7.1 Communication Paradigms . . . 27
2.7.2 Open Platform Communications Unified Architecture . . . 28
2.7.3 Message Queuing Telemetry Transport . . . 29
2.7.4 Complementary Roles . . . 29
2.8 Additive Manufacturing . . . 29
2.8.1 Additive Manufacturing Processes . . . 30
2.8.2 The Digital Process Chain . . . 30
2.8.3 Build Time and Digital Spare Parts . . . 31
2.9 Summary . . . 32

3 Literature Review **33**
3.1 Review Methodology . . . 33
3.1.1 Search Strategy . . . 34
3.1.2 Screening and Eligibility . . . 34
3.1.3 Limitations of the Review . . . 36
3.2 From Prognostics to Timely Maintenance Decisions . . . 36
3.2.1 Coupling Prognostics with Maintenance and Spare Part Decisions . . . 36
3.2.2 Time to Act and Uncertainty . . . 37
3.2.3 From Decision to Execution . . . 38
3.2.4 Prescriptive Maintenance Frameworks . . . 38
3.3 Digital Twins for Predictive and Prescriptive Maintenance . . . 42
3.3.1 Scope and Updating of the Twin . . . 42
3.3.2 From Twin to Action . . . 43
3.3.3 Autonomy of Digital Twin-Based Maintenance . . . 44
3.4 Distributed Control for Failure Handling and Maintenance . . . 44
3.4.1 Fault Handling, Self-Healing, and Reconfiguration . . . 46
3.4.2 Analytics and Machine Learning in IEC 61499 Applications . . . 47
3.4.3 Agent-Based Approaches to Predictive Maintenance . . . 47
3.5 Additive Manufacturing for On-Demand Spare Parts . . . 50
3.5.1 Selecting Parts and Sourcing Options . . . 50
3.5.2 Digital Inventory and Production Time . . . 50
3.5.3 Linking Additive Production to Equipment Condition . . . 51
3.6 Integrated Approaches and Synthesis . . . 52
3.6.1 Autonomous Maintenance and Integrated Approaches . . . 52
3.6.2 Synthesis . . . 53
3.6.3 Research Gaps and Research Questions . . . 53

4 Methodology **56**
4.1 Proposed System Architecture . . . 56
4.2 Digital Twin and Prescriptive Logic . . . 57
4.3 Distributed Control Implementation . . . 59
4.4 Additive Manufacturing Integration . . . 62
4.5 Degree of Automation and Role of the Human . . . 64

5 Implementation **66**
5.1 Testbed . . . 66
5.2 Control of the Production Line and the Additive Manufacturing Node . . . 67
5.2.1 Production Line . . . 67
5.2.2 Additive Manufacturing Node . . . 68

5.3 Data Collection . . . 69
5.4 Prognostic Model Development . . . 70
5.4.1 Preprocessing and Feature Extraction . . . 70
5.4.2 Health Labels . . . 71
5.4.3 Model Selection and Training . . . 72
5.5 Online Health Estimation . . . 74
5.5.1 Streaming Pipeline . . . 74
5.5.2 Integration with DINASORE . . . 75
5.6 Integration of the Complete System . . . 76
5.7 Implementation Limitations . . . 77

6 Validation and Results **82**
6.1 Validation Strategy . . . 82
6.1.1 Individual Validation . . . 82
6.1.2 Complete Validation . . . 83
6.2 Control and Data Acquisition . . . 84
6.3 Prognostic Models . . . 86
6.4 Triggering of the Replacement Part Production . . . 89
6.5 End-to-End Validation . . . 91
6.6 Summary . . . 92

7 Discussion **93**
7.1 Control and Data Acquisition: A Precondition Confirmed . . . 93
7.2 Prognostic Models: Sensing Modality and Model Behavior . . . 94
7.3 Replacement Part Triggering: Prescriptive Logic and Distributed Routing . . . 96
7.4 Closing the Loop: From Prediction to the Replacement Part . . . 96
7.5 Revisiting the Research Gaps and Questions . . . 98
7.6 Limitations . . . 100

8 Conclusions **102**
8.1 Summary of the Dissertation . . . 102
8.2 Fulfillment of the Objectives . . . 103
8.3 Contributions . . . 104
8.4 Practical Impact . . . 105
8.5 Future Work . . . 106

A Literature Review Research Queries **108**

B Function Block Applications in 4diac IDE **114**

C Disclosure of Generative Artificial Intelligence Usage **118**

Glossary **119**

References **122**

Index **145**

List of Figures

2.1 Hierarchical distribution of maintenance types (adapted from [26]) 7
2.2 Steps of a condition-based maintenance program [33] and the technical processes of a prognostic program [34] 9
2.3 Relation between the evolution of the HI, the prognostic horizon, the RUL at the moment of decision, and the lead time of the resulting action (illustrative) 12
2.4 Four classes of functions to which automation can be applied, each at a different level, illustrated for an example system (adapted from [53]) 15
2.5 Comparison of digital integration levels showing data flow characteristics between the physical entity and its digital representation 20
2.6 Characteristics of an IEC 61499 FB (adapted from [75]) 25
2.7 IEC 61499 models of a distributed system: devices containing resources, and an application whose FBs are distributed across resources and connected through communication SIFBs (illustrative) 27
2.8 Client/server and publish/subscribe interaction paradigms 28
2.9 Digital process chain of AdM, with slicing shown explicitly (based on [93]) . . . 31

3.1 Flow of records through the review 35

4.1 Proposed system architecture 57
4.2 Multiple and independent DT instances sharing AdM resources 58
4.3 Logic pipeline inside the DT 59
4.4 Distributed control implementation of the proposed system architecture 63
4.5 AdM integration in the proposed system architecture as a distributed node 64

5.1 Photographs of the experimental testbed: (a) the three EEZYbotARM Mk2 arms, fitted with the ADXL345, piezoelectric, and MPU9250 sensors on Arms #1 to #3, respectively; (b) the Ender-3 S1 Pro 3D printer and the Raspberry Pi that hosts its OctoPrint and DINASORE instances 79
5.2 Proposed system architecture adapted to the testbed 80
5.3 Gripper parts produced by the AdM node in a single print job 81

6.1 MQTT inter-arrival interval between messages of Arm #1's sensing pipeline . . . 85
6.2 MQTT inter-arrival interval between messages of Arm #3's sensing pipeline . . . 85
6.3 Predicted and labelled HI on the held-out test sequences of the ADXL345 model 87
6.4 Predicted and labelled HI on the held-out test sequences of the piezoelectric model 87
6.5 Predicted and labelled HI on the held-out test sequences of the MPU9250 model . 88
6.6 Predicted and labelled HI with whole sessions held out: (a) ADXL345; (b) piezoelectric; (c) MPU9250 90

B.1 Configuration of the complete system 114
B.2 FB network of the control and sampling devices of Arm #1 115
B.3 FB network of the control device of Arm #2 115

B.4 FB network of the control and sampling devices of Arm #3 116
B.5 FB network of the AdM node 116
B.6 FB network of the DT node of Arm #1 116
B.7 FB network of the DT node of Arm #2 117
B.8 FB network of the DT node of Arm #3 117

List of Tables

2.1 Stages of data analytics and corresponding maintenance strategies (based on [29], [44]) . . . 13
2.2 Levels of automation of decision and action selection (adapted from [53, p. 287]) 15
2.3 The 5C architecture for the implementation of CPSs: focus and attributes of each level (adapted from [32]) . . . 17
2.4 Summary of the main concepts and definitions adopted in this chapter . . . 32

3.1 Review themes and their screening questions . . . 33
3.2 Records retrieved and retained at each stage of the review, per theme . . . 35
3.3 Comparison of representative studies coupling prognostics with timely maintenance decisions . . . 39
3.4 Comparison of representative prescriptive maintenance frameworks . . . 41
3.5 Comparison of representative DT-based maintenance studies . . . 45
3.6 Comparison of representative studies using distributed control for failure handling, analytics, and maintenance . . . 49
3.7 Comparison of representative studies on additive manufacturing for spare parts . 54
3.8 Synthesis of the literature review: state of the art and open issues . . . 54

4.1 Comparison between 4diac FORTE and DINASORE . . . 60
4.2 Degree of automation of the proposed system per class of function (after [53]) . . 64

5.1 Devices of the testbed and their roles . . . 67
5.2 Movements of a production cycle and the MQTT topics that trigger them . . . 68
5.3 Windows, blocks, and sequences of each sensor . . . 73
5.4 Hyperparameter search space and best values per sensor . . . 74
5.5 Custom FBs developed for the system . . . 76
5.6 Calibration of the thresholds of the prescriptive logic, from the RUL error on test sequences with a real RUL between 2.5 and 7 h . . . 77

6.1 Overview of the validation tests . . . 83
6.2 Distributed control and sensor data acquisition results . . . 85
6.3 Health estimation performance on the held-out test sequences . . . 86
6.4 RMSE and MAE of the HI estimated by the proposed models and by simpler regressors, on the same test sequences . . . 88
6.5 HI estimation errors with whole sessions held out (five interleaved groups of sessions) . . . 89
6.6 Replacement part triggering results . . . 89
6.7 End-to-end validation results . . . 91
6.8 Stages of the response time in the end-to-end trials, from the publication of the print order . . . 91

7.1 Summary of the answers to the research questions . . . 99

A.1 Complete list of literature review search queries 108

List of Acronyms and Symbols

Acronyms

3D	Three-Dimensional
5G	Fifth Generation
AAS	Asset Administration Shell
ABS	Acrylonitrile Butadiene Styrene
AdM	Additive Manufacturing
AHP	Analytic Hierarchy Process
AI	Artificial Intelligence
AM	Autonomous Maintenance
AMF	Additive Manufacturing File Format
API	Application Programming Interface
ASTM	American Society for Testing and Materials
CAD	Computer-Aided Design
CAN	Controller Area Network
CBM	Condition Based Monitoring
CM	Corrective Maintenance
CMMS	Computerized Maintenance Management System
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
COVID-19	COronaVIrus Disease 2019
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
DCS	Distributed Control System
DINASORE	Dynamic INtelligent Architecture for Software and MOdular REconfiguration
DL	Deap Learning
DM	Digital Model
DS	Digital Shadow
DT	Digital Twin
ECC	Execution Control Chart
EMA	Exponential Moving Average
EN	European Standard

ERP	Enterprise Resource Planning
FB	Function Block
FDM	Fused Deposition Modeling
FFF	Fused Filament Fabrication
FFT	Fast Fourier Transform
FIFO	First In, First Out
FIPA	Foundation for Intelligent Physical Agents
FPT	First Predicting Threshold
G-code	Geometric Code
GMM	Gaussian Mixture Model
HI	Health Index
HITL	Human-In-The-Loop
HOTL	Human-On-The-Loop
HPC	High-Performance Computing
HS	Health State
HTTP	HyperText Transfer Protocol
HVAC	Heating, Ventilation, and Air Conditioning
I/O	Input/Output
I4.0	Industry 4.0
I5.0	Industry 5.0
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEV	International Electrotechnical Vocabulary
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
KPI	Key Performance Indicator
LLM	Large Language Model
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MC	Monte Carlo
MES	Manufacturing Execution System
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NSF	National Science Foundation
OPC UA	Open Platform Communications Unified Architecture

OS	Operating System
P2V	Physical-To-Virtual
PCA	Principal Component Analysis
PdM	Predictive Maintenance
PH	Prognostic Horizon
PHI	Physics Health Indicator
PHM	Prognostics and Health Management
PID	Proportional-Integral-Derivative
PLA	PolyLactic Acid
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
POMDP	Partially Observable Markov Decision Process
PsM	Prescriptive Maintenance
PvM	Preventive Maintenance
PWM	Pulse Width Modulation
QoS	Quality-of-Service
REST	REpresentational State Transfer
RL	Reinforcement Learning
RMS	Root Mean Square
RMSE	Root Mean Square Error
RQ	Research Question
RTE	RunTime Environment
RUL	Remaining Useful Life
SCADA	Supervisory Control And Data Acquisition
SD	Secure Digital
SIFB	Service Interface Function Block
SPI	Serial Peripheral Interface
STL	STereoLithography
TPM	Total Productive Maintenance
UDP	User Datagram Protocol
USB	Universal Serial Bus
V2P	Virtual-To-Physical
VHI	Virtual Health Indicator
XML	eXtensible Markup Language

Symbols

b	Offset of the exponential degradation function of the health labels (Equation 5.1)
d	Asymptote of the exponential degradation function of the health labels (Equation 5.1)

$HI(t)$	Health index at operating time t
L	Length of the degradation period, $t_{end} - t_{FPT}$
t	Operating time
t'	Operating time normalized between the first predicting time and the end of life (Equation 5.2)
t_{end}	Operating time at the end of life
t_{FPT}	Operating time at the first predicting time
α	Smoothing factor of the exponential moving average
λ	Rate of the exponential degradation function of the health labels (Equation 5.1)
σ	Standard deviation

Units

g	Standard acceleration of gravity, $9.806\,65\,\mathrm{m\,s^{-2}}$
h	Hour
Hz	Hertz
kHz	Kilohertz
mg	Milli-g, one thousandth of g
min	Minute
mm	Millimetre
ms	Millisecond
s	Second

Chapter 1

Introduction

Industry 4.0 (I4.0) promised an era of seamless automation, interoperability, and sustainability [1], [2]. Yet, more than a decade after the term was coined, we find it lacking in three main pillars. First, it lacks human-centricity, as it drove an intentional reduction of human presence on the shop floor. Second, it lacks sustainability, because the computational and energy resources demanded by Artificial Intelligence (AI), big data, and Fifth Generation (5G) networks often offset the efficiency gains they enabled. Third, it lacks resilience, since the highly optimized global supply chains it produced proved vulnerable to disruptions such as the COronaVIrus Disease 2019 (COVID-19) pandemic [2]–[4]. These shortcomings motivated the European Commission and the scientific community to propose Industry 5.0 (I5.0). This new paradigm does not replace I4.0 but rather corrects its course, refocusing industry on human-machine collaboration, environmental stewardship, and the capacity to anticipate, adapt to, and recover from disruption [3], [5].

1.1 Context and Motivation

Industrial maintenance sits at the center of this transition [3]. It heavily affects asset availability, manufacturing quality, and overall productivity, with its impact reaching up to 60 % of a facility's total operating costs [6]. We have seen maintenance evolve considerably over the years. It moved from a purely reactive, run-to-failure basis, through schedule-based preventive strategies, and into Predictive Maintenance (PdM), which continuously monitors equipment condition to anticipate failures ahead of time [6], [7]. Yet, accurately predicting a failure is not the same as resolving it: PdM neither decides what action we should take in response to a prediction, nor guarantees that we actually have the physical means to carry out that action when needed [8]. If a spare part is out of stock, obsolete, or delayed in the supply chain, the machine still experiences downtime, regardless of how accurate the underlying prediction was [9].

We look to two complementary developments to address these remaining gaps. On the decision-making side, Prescriptive Maintenance (PsM), typically hosted by a Digital Twin (DT), goes beyond prediction to determine, recommend, or autonomously trigger the optimal corrective action. When we embed this logic into a suitably autonomous execution architecture, it closes the loop towards a self-healing, Autonomous Maintenance (AM) system [8], [10]. On the physical execution side, Additive Manufacturing (AdM) offers a way to remove the logistical bottleneck altogether. Instead of holding physical spare parts in stock, we maintain a digital inventory of

Computer-Aided Design (CAD) files and raw materials to fabricate replacement parts on demand. This approach reduces lead times by up to 70 % and inventory costs by up to 60 % compared to traditional supply chains [9], [11], while directly contributing to the sustainability and resilience goals of I5.0 [12], [13]. To realize this autonomous, self-healing behavior across a real shop floor, we must distribute the underlying control logic rather than centralize it, avoiding single points of failure and accommodating agile, on-the-fly reconfigurations [14], [15]. Standards such as the International Electrotechnical Commission (IEC) 61499 specifically support this kind of event-driven, distributed orchestration [16].

Taken together, PdM, DT-hosted PsM, distributed control, and AdM outline a maintenance pipeline capable of autonomously predicting a failure, deciding on the correct response, and physically fabricating its solution. This convergence aligns naturally with the human-centric, sustainable, and resilient goals of I5.0. We are motivated by this convergence and the goal of realizing it as a single, physically validated system.

1.2 Problem Statement

Each of the technologies we introduced above (PdM, DT-hosted PsM, distributed Cyber-Physical Production System (CPPS) control, and AdM) is reasonably mature and well established within its own domain. However, we rarely see them brought together. Works centered on prescriptive decision-making rarely address the physical execution layer that would carry out their decisions; works on AdM rarely discuss integration with standards-compliant distributed control; and works on distributed control architectures are rarely framed within a maintenance context at all. In Chapter 3, we examine this fragmentation in detail.

The core problem we address in this dissertation is how to autonomously close the loop between the prediction of an impending failure and its physical mitigation, without relying on centralized control or human intervention in the decision path. Concretely, we must reliably convert an inherently uncertain prognostic estimate into a correctly timed corrective action. We must trigger this action early enough for an on-demand AdM process to reach completion before the predicted failure occurs, and we must coordinate it across distributed nodes rather than a single point of failure. To the extent of the literature we reviewed, no architecture has achieved this end to end, nor validated it as a physically working system.

1.3 Objectives

Our primary objective in this dissertation is to architect, deploy, and validate an integrated CPPS framework in which PsM logic, hosted by a DT, dynamically triggers on-demand AdM processes through a distributed, IEC 61499-compliant control environment. We break this primary objective down into four specific objectives, which guided the design of the proposed solution we present in Chapter 4:

- **O1** — Design a Cyber-Physical System (CPS) architecture that physically integrates prognostics-based PdM, DT-hosted PsM decision logic, distributed control, and AdM into a single maintenance pipeline;

- **O2** — Implement the distributed control layer of this architecture using an IEC 61499-compliant, event-driven RunTime Environment (RTE) capable of hosting Machine Learning (ML)-based prognostic models;
- **O3** — Integrate an AdM device as an autonomous, on-demand production node, triggered directly by the PsM logic without centralized coordination or human decision-making;
- **O4** — Deploy and validate the resulting system end to end, from sensor data acquisition to the autonomous triggering of replacement part production, on a physical, laboratory-scale testbed.

We evaluate these objectives in Chapter 7 against three concrete Research Questions (RQs). We formally grounded these questions in Section 3.6.3 based on the literature gaps we identified throughout the review conducted in Chapter 3:

- **RQ#1** — To what extent can on-demand AdM autonomously bridge the gap between PdM and physical maintenance execution?
- **RQ#2** — To what extent do DT technologies ease the integration of PdM, PsM logic, and the autonomous triggering of decentralized manufacturing commands?
- **RQ#3** — To what extent can a distributed IEC 61499 architecture support AM workflows?

1.4 Publication and Code Availability

Part of the work presented in this dissertation has been accepted for publication in a peer-reviewed journal:

- P. Mota and R. Pinto, "An industrial IoT architecture for autonomous maintenance: Integrating digital twins, distributed control, and on-demand additive manufacturing," *Applied Sciences*, 2026, Accepted for publication

The software developed in this work, including the IEC 61499 applications and custom Function Blocks (FBs), the data processing and model training pipeline, the online estimation scripts, and the trained models, is publicly available at `https://github.com/pedrodsmota/iiot-autonomous-maintenance`.

1.5 Document Structure

The remainder of this dissertation is organized as follows.

Chapter 2 establishes the concepts on which we build our work. We define the terminology we adopt for autonomous maintenance, PsM, and the DT.

Chapter 3 reviews the literature on each link of the maintenance loop. We cover the evolution of industrial maintenance towards PdM and PsM, the role of CPSs and DTs, the architecting of distributed control systems under the IEC 61499 standard, and the use of AdM as an on-demand maintenance service. We then identify the research gaps and formulate the research questions we address.

Chapter 4 proposes our system methodology. We detail the overall CPS architecture, the DT and PsM decision logic, the distributed control implementation based on IEC 61499, and the integration of AdM as an autonomous production node.

Chapter 5 describes our implementation of this solution on a physical testbed. We cover the control of the production line and 3D printer, sensor data collection, the development of the prognostic models, and the orchestration of the complete CPPS.

Chapter 6 presents the validation strategy we adopted, together with the results we obtained. We report on the individual validation of the control and data collection pipelines, the prognostic models, and the triggering of replacement part production, followed by our complete, end-to-end closed-loop trials.

Chapter 7 discusses these results in light of our objectives and research questions. We revisit the literature gaps identified in Chapter 3 and outline the limitations of our implementation and validation strategy.

Finally, Chapter 8 concludes the dissertation. We summarize the developed work, assess the fulfillment of our established objectives, highlight our main scientific contributions and practical impacts, and outline directions for future work.

Additionally, this dissertation includes three appendices. Appendix A provides the complete list of search queries used to conduct the literature review presented in Chapter 3. Appendix B details the IEC 61499 applications of the system described in Chapter 5, as developed in the 4diac Integrated Development Environment (IDE). It illustrates the configuration of the complete system, showing its devices and their resources, alongside the specific function block networks deployed on each device, color-matched to the system configuration diagram for clarity. Finally, Appendix C contains a disclosure regarding the use of generative AI tools during the preparation of this document.

Chapter 2

Background and Fundamental Aspects

In this chapter, we introduce the concepts on which the remainder of this dissertation builds. Its purpose is not to review and compare existing works, which is the subject of Chapter 3, but to define the terminology used throughout and, where the literature is not consistent, to state which definition is adopted. The chapter follows the logic of the system proposed in this dissertation. Section 2.1 presents the industrial context in which it is framed, and Section 2.2 the maintenance strategies on which it builds. Sections 2.3 and 2.4 address how the remaining useful life of equipment is estimated, how such estimates are turned into maintenance decisions, and with which degree of autonomy. Section 2.5 introduces CPSs and DTs, which host these functions, and Sections 2.6 and 2.7 the distributed control and communication technologies used to implement them. Section 2.8 presents AdM, through which replacement parts are produced, and Section 2.9 summarizes the definitions adopted in the chapter.

2.1 Industrial Context: From Industry 4.0 to Industry 5.0

The term I4.0 was first presented at the Hannover Messe in 2011 [18] and later detailed in the recommendations of the German Industrie 4.0 working group, according to which it involves the technical integration of CPSs into manufacturing and logistics and the use of the Internet of Things (IoT) and Services in industrial processes [19]. In the manufacturing environment, these CPSs comprise smart machines, storage systems, and production facilities capable of autonomously exchanging information, triggering actions, and controlling each other, forming smart factories that are better able to manage complexity, less prone to disruption, and able to respond flexibly to disruptions and failures [19]. Four design principles have been identified for implementing I4.0:

- **Interconnectivity** — The interconnection of machines, devices, sensors, and people;
- **Information transparency** — Creating a virtual copy of the physical world created by linking sensor data with digital plant models;
- **Decentralized decisions** — Participants perform their tasks as autonomously as possible;
- **Technical assistance** — Supports humans in increasingly complex production environments [20].

Similar strategic initiatives have been launched in other countries, such as advanced and smart manufacturing in the United States of America, Society 5.0 in Japan, Made in China 2025, and Industry of the Future in France [21].

Like the industrial revolutions that preceded it, I4.0 is considered technology-driven, and it is not regarded as a human-centric initiative [22]. In 2021, the European Commission introduced I5.0, which is value-driven instead [22]. I5.0 is neither a chronological continuation of I4.0 nor an alternative to it, but complements and extends its hallmark features [22], [23], and many of its enabling technologies are expected to help achieve its societal goals [22]. It rests on three core values [23]:

- **Human-centricity** — Places core human needs and interests at the heart of the production process, requiring that "technology serves people, rather than the other way around" [23, p. 15], so that production processes are adapted to the needs of workers rather than the reverse;
- **Resilience** — Aim for a higher degree of robustness in industrial production against disruptions;
- **Sustainability** — Working towards circular processes that reuse and recycle natural resources and reduce waste and environmental impact [23].

These values also reshape the role of automation. Rather than replacing workers, I5.0 is described as a synergy between humans and autonomous machines [24], in which intelligent robots and software take over dull, dirty, and dangerous tasks, while humans concentrate on creativity, decision-making, and flexible problem-solving [25]. The same shift has been proposed for maintenance, where I4.0 technologies such as DTs, ML, and the IoT are mapped onto the three core values of I5.0, and human oversight is kept within increasingly automated maintenance processes [3]. The role of humans in the automated maintenance system we propose in this dissertation is characterized in Section 2.4.3, and its alignment with these values is discussed in Chapter 7.

2.2 Maintenance Strategies and Terminology

Before discussing how maintenance decisions can be automated, it is necessary to establish what the different types of maintenance are. This section presents the maintenance types defined in the European Standard (EN) on maintenance terminology, and clarifies the meaning in which the term AM is used in this dissertation, which differs from its established one.

2.2.1 Maintenance Types

Maintenance terminology is standardized in EN 13306, which defines maintenance as the

> "combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function" [26, pp. cl. 2.1].

Three related terms describe what maintenance responds to:

- **Failure** — The loss of the ability of an item to perform a required function;
- **Fault** — The resulting state of inability to perform it;
- **Degradation** — A detrimental change in the physical condition of the item, whether with time, with use, or due to an external cause [26].

The International Electrotechnical Vocabulary (IEV) provides equivalent definitions [27].

EN 13306 organizes maintenance types hierarchically, as illustrated in Figure 2.1. At the top level, Corrective Maintenance (CM) is carried out after a fault has been recognized, to restore the item to a state in which it can perform its required function, whereas Preventive Maintenance (PvM) is carried out to assess or mitigate degradation and reduce the probability of failure [26]. PvM is in turn divided into predetermined maintenance, carried out at established intervals of time or units of use without previous investigation of the condition of the item, and condition-based maintenance, which includes the assessment of physical conditions, their analysis, and the possible ensuing maintenance actions [26]. Finally, PdM is defined as a form of condition-based maintenance, namely "condition-based maintenance carried out following a forecast derived from repeated analysis or known characteristics and evaluation of the significant parameters of the degradation of the item" [26, pp. cl. 7.4]. What distinguishes PdM from condition-based maintenance in general is therefore the forecast of the future evolution of degradation, as provided by the prognostic methods discussed in Section 2.3.

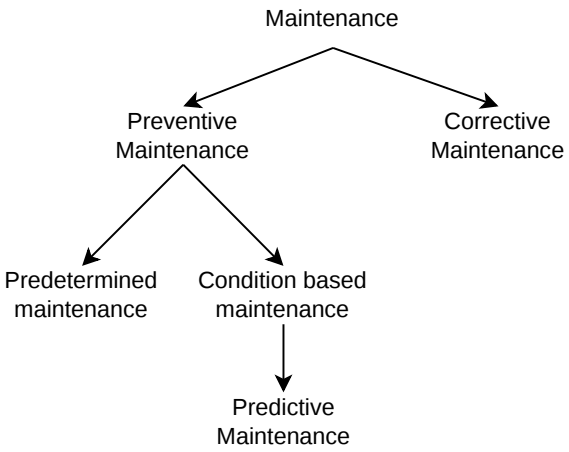


Figure 2.1 – Hierarchical distribution of maintenance types (adapted from [26])

This hierarchy is not always followed in the literature, where the terms are often used loosely. Condition Based Monitoring (CBM) has been described as synonymous with condition-based maintenance and as distinct from PdM [3], PdM has been described as synonymous with condition-based maintenance [28], and the two have also been presented as successive strategies, with condition-based maintenance acting on evidence of degradation and PdM predicting the remaining life of the system [29]. In this dissertation, we follow the terminology of EN 13306, in which PdM is a form of condition-based maintenance and, therefore, of PvM. EN 13306 does not define PsM, which extends PdM with the decision on which action to take, as discussed in Section 2.4.2.

2.2.2 Autonomous Maintenance

The term AM has an established meaning that differs from the one used in this dissertation. It originates in Total Productive Maintenance (TPM), an approach to maintenance that seeks to maximize equipment effectiveness through a maintenance system covering the entire life of the equipment, with the participation of all employees, from top management to shop-floor workers [30]. TPM pursues the goals of zero defects, zero accidents, and zero breakdowns, and is commonly organized into eight pillars, the first of which is AM [28]. In this sense, AM refers to operators taking care of their own machines without being ordered to, performing tasks such as cleaning, lubricating, tightening, adjusting, and inspecting the equipment [28], and it involves housekeeping on the production line, cross-training operators in maintenance tasks, and teams of production and maintenance personnel [30]. EN 13306 reflects this meaning, treating the term as a synonym of operator maintenance: "maintenance carried out by an operator" [26, pp. cl. 7.17].

A different meaning has emerged with the development of intelligent maintenance systems, in which it is the machine, rather than the operator, that becomes self-governing. Lee *et al.* [31] describe self-maintenance as the ability of a machine to carry out regular checks on itself, detect anomalies, and make immediate repairs when needed, using stocked spare parts to avoid potential catastrophic losses, and the 5C architecture presented in Section 2.5.1 likewise envisions production systems with self-maintainability [32]. Both meanings coexist in the recent literature: Khan *et al.* [10] define AM as a PvM strategy focusing on the system being self-governing, while still performing maintenance activities through the cooperation between maintenance personnel and operators.

In this dissertation, AM refers to the second meaning: a maintenance approach in which the monitoring of the equipment, the analysis of its condition, the decision on the required maintenance action, and, as far as possible, its execution are carried out by the production system itself. Since not every step can be automated, the degree of autonomy of such a system is characterized stage by stage, and the role left to humans made explicit, following the framework discussed in Section 2.4.3. The two meanings are not mutually exclusive: in an AM system in which a replacement part is produced automatically but installed by a human, the final step remains, in the sense of EN 13306, maintenance carried out by an operator.

2.3 Prognostics and Remaining Useful Life Estimation

Predicting when a failure will occur is the starting point of any PdM or PsM strategy. This section introduces Prognostics and Health Management (PHM), the construction of health indicators and health stages, the main families of prognostic approaches, and how the uncertainty and performance of prognostic estimates are assessed. It concludes by relating the timing of a prediction to the time needed to act upon it, which is central to the problem addressed in this dissertation.

2.3.1 Prognostics and Health Management

A condition-based maintenance program consists of three main steps:

- **Data acquisition** — Data relevant to the health of the system is collected;

- **Data processing** — The acquired data is handled and analyzed;
- **Maintenance decision-making** — Maintenance policies are recommended [33].

Within this program, prognostics deals with the prediction of faults before they occur, determining whether a fault is impending and how soon and how likely it is to occur [33], by predicting the Remaining Useful Life (RUL) of machinery from its historical and ongoing degradation trends [34]. The RUL is defined as "the length from the current time to the end of the useful life" [34, p. 814], that is, as the time left before a failure is observed, given the current age and condition of the machine and its past operation profile [33].

Lei *et al.* [34] describe a prognostic program as consisting of four technical processes. First, data such as vibration signals are acquired from sensors to monitor the condition of the machine. From these data, Health Indexes (HIs) are constructed to represent its health condition, and the lifetime of the machine is then divided into two or more Health States (HSs) according to the degradation trends of the HIs. Finally, the RUL is predicted [34]. Figure 2.2 relates these four processes to the steps of a condition-based maintenance program. Developing and validating each of these processes requires data covering the complete life of the monitored components. Since run-to-failure data from real industrial equipment is difficult to obtain, most prognostic datasets in the literature are acquired from accelerated degradation test beds, as in the PRONOSTIA [35] platform, where bearings are driven to failure within a few hours by applying radial loads above their maximum dynamic load [34]. Accelerated degradation provides an accessible way of collecting degradation patterns similar to real ones, but since the aging conditions are not real, some information is lost, and the resulting models may not appropriately fit the real degradation phenomenon [36].

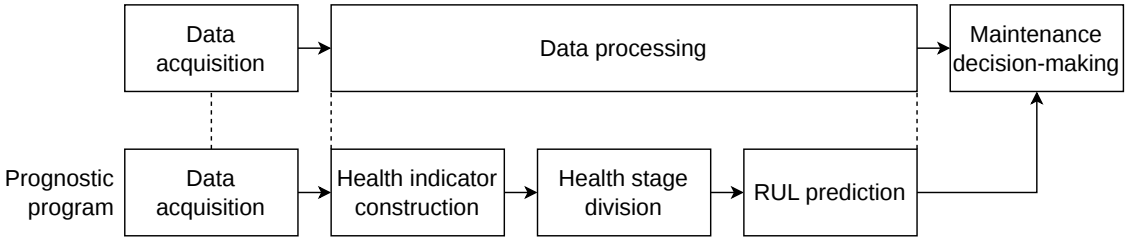


Figure 2.2 – Steps of a condition-based maintenance program [33] and the technical processes of a prognostic program [34]

2.3.2 Health Indicators and Health Stages

An HI is a function or performance indicator that describes the health of the monitored system. It can be designed from the input signals themselves, by fusing information from them, or defined as a linear or exponential degradation function [36]. Lei *et al.* [34] distinguish two classes of HIs. Physics Health Indicators (PHIs) are related to the physics of failure and are extracted from the monitoring signals through statistical or signal processing methods, such as the Root Mean Square (RMS) of a vibration signal, whereas Virtual Health Indicators (VHIs) are constructed by fusing multiple PHIs or signals from multiple sensors. The quality of an HI can be assessed through metrics that depend on the HI alone, on the HI and time, on the HI and the sequence of

HSs, on multiple HIs, or on combinations of these [34]. When labels for the degradation process are not available, the HI can instead be defined by a degradation function fitted to the life of the component. For accelerated life tests, an exponential function is commonly used, since the acceleration of aging drives the component towards failure exponentially [36].

The HS indicates the phase of operation to which the behavior of the system belongs, such as normal operation, degradation, or complete failure [36]. Unlike fault diagnosis, which identifies the pattern and severity of a fault at a single point in time, HS division partitions the continuous degradation process into stages according to the trends of the HIs [34]. One of its main purposes is to determine the First Predicting Threshold (FPT), the start of the unhealthy stage, from which the RUL prediction should be triggered, which requires detecting the incipient degradation of the machine [34], [36]. Since an HI describes the health of the system rather than a time, the RUL corresponds to the time remaining until the HI reaches the level associated with the end of life, which depends on how the HI is expected to evolve.

2.3.3 Prognostic Approaches

Prognostic approaches have been classified in different ways, for instance into statistical, AI, and model-based approaches [33], or into physics model-based, statistical model-based, AI, and hybrid approaches [34]. They are commonly grouped into three families [37], [38]. Physics-based approaches rely on knowledge of the failure mechanisms of the system, such as crack growth, to build a mathematical description of its degradation, whose parameters are identified from measured data [37], [38]. When sufficiently complete, such models tend to outperform other approaches and have a direct physical meaning [38], but they are often unavailable for complex systems, whose failure modes and behavior under varying operating conditions are not fully understood [34], [38]. Data-driven approaches rely only on previously collected data to identify the current damage state and project its future trend [37], [38]. Since degradation data often show distinct characteristics near the end of life, these approaches are particularly effective at predicting near-future behavior [37], but models such as neural networks require many training datasets, which are difficult to obtain in practice [37]. Hybrid approaches combine both families to improve prediction performance [37].

Within data-driven approaches, Deap Learning (DL) models have become widespread. Their performance, however, depends heavily on the scale and quality of the available data [39], and labeled failure data is scarce, since failures are rare [40]. Different architectures capture different aspects of the degradation signal. Convolutional Neural Networks (CNNs) extract local features from their input, but, like other conventional methods, do not account for the temporal relevance of time-series signals that reflect small changes in the state of health [40]. Long Short-Term Memory (LSTM) networks address this by adding a memory control that decides which information to retain for prediction and which to discard, handling long-term dependencies while avoiding the vanishing and exploding gradient problems of conventional recurrent networks [40]. The two can be combined: in such architectures, a CNN extracts robust local features from the sequential input, and a bidirectional LSTM then encodes the temporal information contained in the sequence of those features [39].

2.3.4 Uncertainty and Evaluation of Prognostics

RUL predictions are inherently uncertain. The most important source of this uncertainty is that the future loading, operating, environmental, and usage conditions of the system are not known precisely, to which is added modeling uncertainty, arising from model parameters, model form, and process noise [41]. This uncertainty can be represented through several frameworks, such as probability theory, fuzzy sets, evidence theory, imprecise probabilities, or intervals, and in condition-based prognostics it is interpreted in a subjective, Bayesian sense [41]. Quantifying it is important because RUL predictions are meant to support decisions, such as fault mitigation or replanning, which must take into account the overall uncertainty of the prediction [41].

The performance of prognostic algorithms is evaluated through metrics that depend on ground-truth RUL values, on run-to-failure data, or on the measurements available [34]. Conventional accuracy metrics, such as the Root Mean Square Error (RMSE) of the RUL predictions between the FPT and the end of life [34], together with the Mean Absolute Error (MAE), the Mean Squared Error (MSE), or the coefficient of determination [36], quantify the deviation between predicted and actual values. Noting that prognostic concepts often suffer from ambiguous and inconsistent interpretations, Saxena *et al.* [42] proposed metrics tailored to prognostics. The Prognostic Horizon (PH) is the time between the moment at which the predictions first meet a specified performance criterion and the end of life, indicating whether predictions can be trusted early enough to be useful. The α-λ accuracy evaluates whether the prediction at a given time falls within specified α bounds of the true RUL, while relative accuracy, cumulative relative accuracy, and convergence assess how predictions improve and converge as more data becomes available [42]. These bounds may be asymmetric, since an early prediction is generally preferred to a late one, and, when predictions are expressed as probability distributions, a prediction may be required to have a minimum probability mass within the bounds [42].

2.3.5 From Prediction to Timely Action

The value of a prognostic estimate depends not only on its accuracy but on when it becomes available. As Saxena *et al.* [42] note, "very accurate predictions at a time when no recovery action can be made are not useful" [42, p. 10]: what is sought is sufficient time to plan and carry out an appropriate corrective action. The International Organization for Standardization (ISO) 13381-1 standard captures this through the predictive horizon, defined as the "threshold for prediction of lead time to failure as desired by the user" [43, p. 2]. For a maintenance action that requires a certain lead time, such as producing and installing a replacement part, a prediction is therefore only useful if it is issued while the remaining time before failure still exceeds that lead time. Since the prediction is uncertain, and a late action is more costly than an early one, the decision to act must also include a margin that accounts for this uncertainty. These relations are illustrated in Figure 2.3, in which the action is triggered when the HI crosses a threshold chosen so that the remaining time exceeds the lead time of the action. In mission-critical scenarios, a limited number of early interventions may be tolerable, whereas a missed failure is not [42]. This link between an uncertain prognostic estimate and the timing of the resulting action is the basis of the prescriptive decision discussed in Section 2.4.

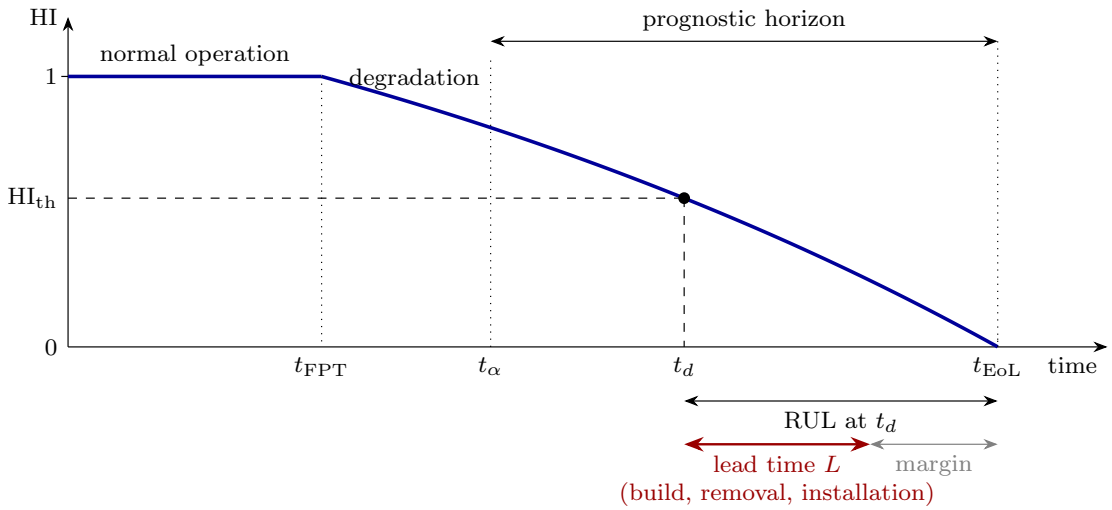


Figure 2.3 – Relation between the evolution of the HI, the prognostic horizon, the RUL at the moment of decision, and the lead time of the resulting action (illustrative)

2.4 From Prediction to Decision: Prescriptive Maintenance and Autonomy

A prediction of remaining useful life does not, by itself, determine what should be done. In this section, we discuss how predictions are turned into decisions, first through the stages of data analytics and then through PsM, before addressing the degree to which such decisions, and the actions that follow them, can be automated, and the role that remains for humans.

2.4.1 From Predictive to Prescriptive Analytics

The transition from predicting a failure to deciding what to do about it is commonly framed in terms of the stages of data analytics, which differ in difficulty, value, and intelligence [44], [45]. Descriptive analytics answers the questions "What has happened?", "Why did it happen?" and "What is happening now?"; predictive analytics answers "What will happen?" and "Why will it happen?"; and prescriptive analytics answers "What should I do?" and "Why should I do it?" [44, p. 58]. Some authors further separate the question of why something happened into a distinct stage, known as diagnostic analytics [44]. Prescriptive analytics thus seeks the best course of action for the future, suggesting the best decision options to take advantage of what has been predicted. To do so, it builds upon the output of predictive analytics and applies methods such as optimization algorithms, expert systems, and AI, which Lepenioti *et al.* [44] classify into probabilistic models, ML and data mining, mathematical programming, evolutionary computation, simulation, and logic-based models. From this perspective, predictive analytics can only be fully exploited in conjunction with prescriptive analytics [44].

The same progression can be mapped onto the maintenance strategies introduced in Section 2.2. Following Gartner's description of the path towards prescriptive analytics, Errandonea *et al.* [29] associate PvM with the question "what happened?", condition-based maintenance with "why did it happen?", PdM with "what will happen?", and PsM with "how can we make it happen?" [29, p. 4], as summarized in Table 2.1. Within PdM itself, Bousdekis *et al.* [45] distinguish

the detection of abnormal behavior (diagnosis), the prediction of future failure modes (prognosis), and the support of decisions ahead of time, describing the latter as the phase in which near real-time predictions trigger proactive recommendations about the maintenance actions and plans that eliminate or mitigate the impact of the predicted failure. They also note that, while data analytics has been extensively used for diagnosis and prognosis, its use in maintenance decision-making is still at an early stage [45]. It is precisely at this step that the RUL estimates we discussed in Section 2.3 become the input of a decision.

Table 2.1 – Stages of data analytics and corresponding maintenance strategies (based on [29], [44])

Analytics stage	Questions answered	Maintenance strategy
Descriptive	What has happened? Why did it happen? What is happening now?	Preventive (what happened?); condition-based (why did it happen?)
Predictive	What will happen? Why will it happen?	Predictive (what will happen?)
Prescriptive	What should I do? Why should I do it?	Prescriptive (how can we make it happen?)

Finally, prescribing an action is not the same as executing it. Following Gartner, Lepenioti *et al.* [44] distinguish two levels of human intervention in prescriptive analytics: decision support, in which recommendations are provided to a human, and decision automation, in which the prescribed action is implemented by the system. Even in the first case, the human may ignore the recommendations, or intervene during their execution if the actions prove unsuitable [45]. This distinction between prescribing and executing underlies the definition of PsM we adopted in Section 2.4.2 and the analysis of automation and human roles in Section 2.4.3.

2.4.2 Prescriptive Maintenance

In the maintenance literature, PsM is commonly framed within knowledge-based maintenance, a term proposed by Ansari *et al.* [46] to denote the range of data and knowledge-driven maintenance capabilities of smart factories, of which PsM represents the highest level of maturity and complexity [46], [47]. Whereas PdM answers the question of what will happen and when, by learning from historical and real-time data to predict future events, PsM answers the question "How can we control the occurrence of a specific event?", by providing actionable recommendations for decision-making and for improving forthcoming maintenance processes [46, p. 484]. The concept therefore extends beyond the prediction of failures: based on historical and real-time data, the required maintenance measures are predicted and a course of action is prescribed, moving from planned PvM to proactive maintenance planning [48]. This addresses the main limitation of PdM, which, while useful to forecast failures, does not determine which actions should be taken, when to take them, and how to optimize these decisions under limited resources and uncertainty [8].

To derive such prescriptions, PsM combines descriptive, diagnostic, and predictive analytics, so as to anticipate both the likelihood of future events and the potential effects of each decision alternative on the machine and on the associated business processes [46]. The PriMa model, for instance, comprises four layers: data management, a predictive data analytics toolbox, a recommender and decision support dashboard, and an overarching layer for semantic learning and reasoning [46]. Similarly, the procedural approach of Matyas *et al.* [48] comprises data acquisition and pre-processing, data analysis and simulation, a reaction model containing the rules from which maintenance measures are derived, and a decision support system. In both cases, the data

used goes beyond condition monitoring, including product quality, process, production planning, and maintenance management data, since maintenance, quality, and production planning jointly determine the performance of a production system [47], [48]. Prescriptions may thus be obtained through optimization, but also through rules and logic-based models, which are one of the categories of methods for prescriptive analytics identified in Section 2.4.1 [44].

The literature diverges, however, on what PsM delivers. In the models of Matyas *et al.* [48] and Nemeth *et al.* [47], the output is a recommendation: the operator or domain expert accepts or rejects the prescribed measures, so as to integrate their implicit knowledge and context sensitivity, and this feedback is used to improve the rules. Likewise, PriMa supports the maintenance planner, who retains the final decision [46], although its authors note that the self-organization capabilities of CPPSs ideally aim at machine self-diagnosis and self-scheduled maintenance. More recent work places PsM closer to execution. Orošnjak *et al.* [8] describe it as embedding decision-making logic into the maintenance execution process, with the prescriptive part aiming "to execute actions that will be self-regulated with minimal human interventions" [8, p. 18], while at the same time noting that the literature overlooks human-in-the-loop feedback and calling for operator-in-the-loop validation in line with I5.0. These positions correspond to the two levels of human intervention in prescriptive analytics discussed in Section 2.4.1: decision support and decision automation.

In this dissertation, we understand PsM as the strategy that uses predictions of future failures, such as RUL estimates, to determine which maintenance action should be taken and when, and prescribes it accordingly, regardless of whether the prescribed action is then carried out by a human or by the system. We further treat the degree to which decisions and actions are automated as a separate dimension in Section 2.4.3. Whatever that degree, a prescription is only useful if it can be carried out in time: an AM system must respond within the time window available for corrective action before the failure becomes critical [10]. For a prescription that involves producing a replacement part, as in this dissertation, this window is bounded by the RUL estimate discussed in Section 2.3, and must accommodate the time needed to manufacture the part.

2.4.3 Automation, Autonomy, and the Role of the Human

Although often used interchangeably, automation and autonomy are distinguished in the literature. Conventional automation systems allow low-level processes to run without human intervention under normal conditions and within a limited system context, whereas autonomous systems can change their behavior in response to unanticipated events without manual intervention; in the words of Gamer *et al.* [49], "Automation is something pre-programmed while Autonomous requires a higher level of reasoning around unexpected events" [49, p. 109]. Other definitions share this emphasis: autonomy has been described as the ability of a system to make its own decisions about its actions without the involvement of an external system or operator [50], as the capability of a machine to perform a task, or part of it, with no or substantially reduced human intervention [51], and as the capability that enables a system to function in the presence of uncertainty [10]. In manufacturing, autonomy allows a production system to respond to unexpected events without reconfiguration at the supervisory level, and the DT discussed in Section 2.5 is

considered a core enabler of it, since autonomous systems require realistic models of the current state of the process and of their own behavior [52].

Autonomy is rarely all-or-nothing, and numerous taxonomies of levels of automation have been proposed, none of which can be considered the best one for all applications [50]. The model of Parasuraman *et al.* [53] is particularly suited to characterizing systems that combine sensing, analysis, and action, since it applies automation to four classes of functions: information acquisition, which covers sensing and the pre-processing of data; information analysis, which covers the processing of that information; decision and action selection; and action implementation. Each class can be automated to a different degree, according to the ten-level scale shown in Table 2.2, so that a given system may, for instance, combine a high level of automation in information acquisition with low levels in the remaining classes [53], as illustrated in Figure 2.4. The authors propose that the level chosen for each class be evaluated primarily by its consequences for human performance, and secondarily by the reliability of the automation and the costs of the consequences of its decisions and actions.

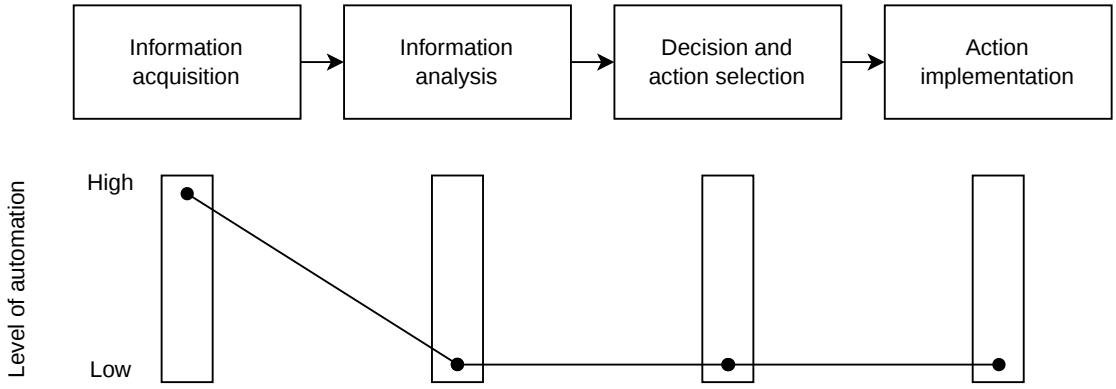


Figure 2.4 – Four classes of functions to which automation can be applied, each at a different level, illustrated for an example system (adapted from [53])

Table 2.2 – Levels of automation of decision and action selection (adapted from [53, p. 287])

Level	Description
10	The computer decides everything, acts autonomously, ignoring the human
9	... informs the human only if it, the computer, decides to
8	... informs the human only if asked, or
7	... executes automatically, then necessarily informs the human, and
6	... allows the human a restricted time to veto before automatic execution, or
5	... executes that suggestion if the human approves, or
4	... suggests one alternative
3	... narrows the selection down to a few, or
2	The computer offers a complete set of decision/action alternatives, or
1	The computer offers no assistance: human must take all decisions and actions

At the level of an entire industrial plant, Gamer *et al.* [49] propose a taxonomy ranging from level 0, with no autonomy and humans in complete control, through operational assistance and decision support (level 1), occasional autonomy in certain situations (level 2), system control in certain situations (level 3), and full control in certain situations combined with learning from past actions (level 4), to fully autonomous operation in all situations (level 5). They draw an important demarcation between levels 0 to 2, in which the human remains in active control, and levels 3 to

5, in which the human plays a passive role at most, for instance by confirming proposed solutions or acting as a fallback. In their analysis, field operations, including maintenance actions such as inspections and repairs, are carried out by staff in the plant itself.

The role left to humans can also be described by their position relative to the control loop [51]. In Human-In-The-Loop (HITL) systems, the machine performs a task for a period of time and then halts, waiting for a human command before continuing. In Human-On-The-Loop (HOTL) systems, the machine performs the task completely on its own, while a human monitors or supervises it, with the ability to intervene and override its decisions in case of failure or error; such systems may become fully autonomous if the supervisor allows them to. Keeping humans in or on the loop is not only a technical limitation. In maintenance, experienced staff are needed to interpret and verify diagnoses and prognoses, a task with limited potential for automation, and human input can help data-driven systems avoid mistakes and learn better decisions over time [54]. The aim is therefore to empower, rather than replace, the human factor [55], in line with the human-centric orientation of I5.0, in which technology is adapted to the needs of the worker rather than the reverse [23]. For these reasons, the autonomy of the maintenance system we propose in this dissertation is not characterized by a single label, but stage by stage, in terms of the level of automation of each class of functions and the position of the human in or on the loop (Chapter 4).

2.5 Cyber-Physical Systems, Cyber-Physical Production Systems and Digital Twins

The prognostic and prescriptive functions we discussed in the previous sections must be hosted by a system that connects them to the physical equipment. In this section, we introduce CPSs and their application to production, followed by the DT, tracing its origins, the ways in which it has been defined, and its use in maintenance, before stating the definition we adopted in this dissertation.

2.5.1 Cyber-Physical Systems and Cyber-Physical Production Systems

The concept of CPS originated in the embedded systems and control communities, where the physical world and the computation that observes and acts upon it can no longer be designed in isolation. Lee defines CPSs as "integrations of computation with physical processes" [56, p. 363], in which embedded computers and networks monitor and control the physical processes, usually through feedback loops where the physical processes affect computations and vice versa. Since time in the physical world progresses inexorably and concurrency is intrinsic, Lee argues that computing abstractions must be rebuilt to embrace physical dynamics and computation in a unified way [56]. Monostori *et al.* [57] offer a networked, service-oriented formulation, describing CPSs as

> "systems of collaborating computational entities which are in intensive connection with the surrounding physical world and its on-going processes, providing and using, at the same time, data-accessing and data-processing services available on the Internet" [57, p. 621].

The two definitions are complementary, since the first describes what makes a system cyber-physical, while the second describes how such systems are organized when many of them interact. The latter idea was summarized in the announcement of the first United States of America's National Science Foundation (NSF) Workshop on CPSs, in 2006, as "Globally Virtual, Locally Physical": sensing and manipulation of the physical world occur locally, while control and observability are enabled across a virtual network [57, p. 623]. In the recent literature, CPSs are commonly structured into a physical layer, comprising the machines on the shop floor, and a cyber layer, comprising the digital and computational space [14], built upon three foundational technologies: computation, communication, and control [58].

When referring to CPSs within manufacturing, the literature shifts to the concept of CPPS [14]. Monostori *et al.* [57] define CPPSs as consisting of

> "autonomous and cooperative elements and sub-systems that are connected based on the context within and across all levels of production, from processes through machines up to production and logistics networks" [57, pp. 623–624],

and underline three main characteristics: intelligence, meaning the elements are able to acquire information from their surroundings and act autonomously; connectedness to the other elements of the system, including human beings, and to the knowledge and services available on the Internet; and responsiveness to internal and external changes. The explicit inclusion of human beings is consistent with the human-centric orientation of I5.0, which favors collaboration between workers and machines over worker displacement [5], [59]. Moreover, for a CPPS to be resilient and flexible, control must be distributed among independent assets capable of self-organization and peer-to-peer collaboration [16], as opposed to traditional hierarchical control, which relies on centralized decision-making and is prone to single points of failure [15]. We address this distribution of control in Section 2.6.

These definitions, however, provide limited guidance on how to implement such systems. To address this, Lee *et al.* [32] proposed the 5C architecture, a step-by-step guideline for implementing CPSs in manufacturing that defines, in five sequential levels, how to construct a CPS from the initial data acquisition, through analytics, to the final value creation, as summarized in Table 2.3.

Table 2.3 – The 5C architecture for the implementation of CPSs: focus and attributes of each level (adapted from [32])

Level	Focus	Attributes
I. Smart Connection	Condition monitoring	Plug & play; tether-free communication; sensor network
II. Data-to-Information Conversion	Self-aware	Smart analytics for component machine health; multi-dimensional data correlation; degradation and performance prediction
III. Cyber	Self-compare	Twin model for components and machines; time machine for variation identification and memory; clustering for similarity in data mining
IV. Cognition	Prioritize and optimize decisions	Integrated simulation and synthesis; remote visualization for human; collaborative diagnostics and decision making
V. Configuration	Actions to avoid	Self-configure for resilience; self-adjust for variation; self-optimize for disturbance

Two features make the 5C architecture a suitable reference for this dissertation. First, unlike generic CPS definitions, it is explicitly oriented towards machine health: at the Conversion level,

prognostics and health management algorithms estimate health values and RUL, and the Cyber level builds a twin of each component and machine [32], anticipating the concept of DT discussed in Section 2.5.2. Second, it separates decision from action. At the Cognition level, the acquired knowledge is presented to expert users to support decisions, such as the prioritization of maintenance tasks, whereas the Configuration level is "the feedback from cyber space to physical space", acting as supervisory control that applies the corrective and preventive decisions made at the Cognition level to the monitored system [32, p. 20]. We later use this distinction to position the system proposed in this dissertation (Chapter 4).

2.5.2 The Digital Twin: Origins and Definitions

The idea of operating a physical system alongside a replica used to reason about it predates the term DT. During the Apollo 13 mission, in 1970, simulators on the ground were used to replicate the conditions aboard the damaged spacecraft and to devise the rescue procedures, which is frequently cited as the first application of the concept [21]. The term itself originates in product lifecycle management: Grieves [60] introduced the concept at an executive course on product lifecycle management at the University of Michigan[1] and attributed the term "digital twin" to John Vickers of National Aeronautics and Space Administration (NASA) [60]. His model comprises three main parts: physical products in real space, virtual products in virtual space, and the connections of data and information that tie the virtual and real products together [60]. Although these connections were conceived in both directions, Grieves [60] himself noted a decade later that the two-way connection between real and virtual space had been lagging behind.

The concept gained prominence in the aerospace field, where it was first formulated in NASA's 2010 technology roadmap [18]. In its most cited form, Glaessgen and Stargel [63] define the DT as

> "an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin" [63, p. 7].

This definition sets a demanding fidelity requirement and ties the twin to one specific, as-built vehicle, rather than to its design. Its purpose is also telling: in its first meaning, the DT emerged as an instrument to support better prediction of failures during the lifecycle of the system, based on sensor data [18], a role closely related to the prognostics discussed in Section 2.3.

From 2013 onwards, the concept moved into manufacturing, where the DT started being regarded as the virtual counterpart of production resources, and not only of products [18]. Negri *et al.* [18] define the DT in this context as a virtual representation of a production system, able to run on different simulation disciplines and characterized by its synchronization with the real system through sensed data, connected smart devices, mathematical models, and real-time data processing [18]. Tao and Zhang [64] extended the scope to the whole shop floor, whose DT comprises the physical shop floor, the virtual shop floor, a shop-floor service system, and the shop-floor digital twin data. Standardization followed with the ISO 23247 series, which defines a DT in manufacturing as "a fit for purpose digital representation of an Observable Manufacturing Element (OME)

[1] Grieves dates this introduction to 2003 [60], as do VanDerHorn and Mahadevan [61], whereas Kritzinger *et al.* [62] place the first presentation of the concept in 2002.

with synchronization between the OME and its digital representation", where an observable manufacturing element may be personnel, equipment, material, a process, a facility, the environment, a product, or a supporting document [65, p. 5].

As the concept spread across domains, its definitions multiplied. Reviews of the literature consistently report that there is no common understanding of the term [18], [62], [66], and VanDerHorn and Mahadevan [61], after reviewing 46 definitions, propose a generalized one, according to which a DT is

> "a virtual representation of a physical system (and its associated environment and processes) that is updated through the exchange of information between the physical and virtual systems" [61, p. 2].

Shao and Helu [67] compare existing perspectives along three dimensions: the viewpoint of the twin, which may be a product, a process, or a system; its fidelity, which may be complete or partial; and its temporal integration, which may be real-time or offline, the latter connecting to the physical system periodically. Based on this comparison, they argue that a DT is a fit-for-purpose representation whose scope depends on its use case, rather than a single complete replica of the physical system [67]. The nature of the underlying model varies accordingly: while the aerospace DT relies on high-fidelity physical simulation, in domains with higher time frequencies, such as condition-based maintenance, it may rely on data-driven approaches instead of time-intensive simulation [62]. A further dimension, the direction and degree of automation of the data flow between the physical and virtual parts, is the most frequently used to distinguish a DT from related concepts, which we discuss in Section 2.5.3.

2.5.3 Levels of Integration and Direction of Data Flow

Since the term DT is often used interchangeably with related concepts, Kritzinger *et al.* [62] proposed to distinguish them by their level of data integration, that is, by whether the data exchanged between a physical object and its digital counterpart flows automatically and in which direction, as depicted in Figure 2.5. A Digital Model (DM) is a digital representation of an existing or planned physical object with no automated data exchange between the two, so that a change in the state of one has no direct effect on the other; a Digital Shadow (DS) adds an automated one-way flow from the physical to the digital object, so that a change in the physical object leads to a change in the digital one, but not vice versa; only when the data flows are fully integrated in both directions is the representation considered a DT, in which case the digital object may also act as a controlling instance of the physical one [62]. Applying this classification to the manufacturing literature, the authors found that most publications described DSs (35%) or DMs (28%): although most papers used the term DT, only 18% described a bidirectional data transfer, and only one case study implemented it, in a laboratory environment [62].

While this classification is useful to separate the three concepts, it treats each direction of data flow as either automated or not, and says little about what flows in each direction. Thelen *et al.* [68] address this by splitting the DT into two connections: a Physical-To-Virtual (P2V) connection, in which an updating engine updates the state of the DM based on sensor data, and a Virtual-To-Physical (V2P) connection, in which a prediction engine uses the updated model to

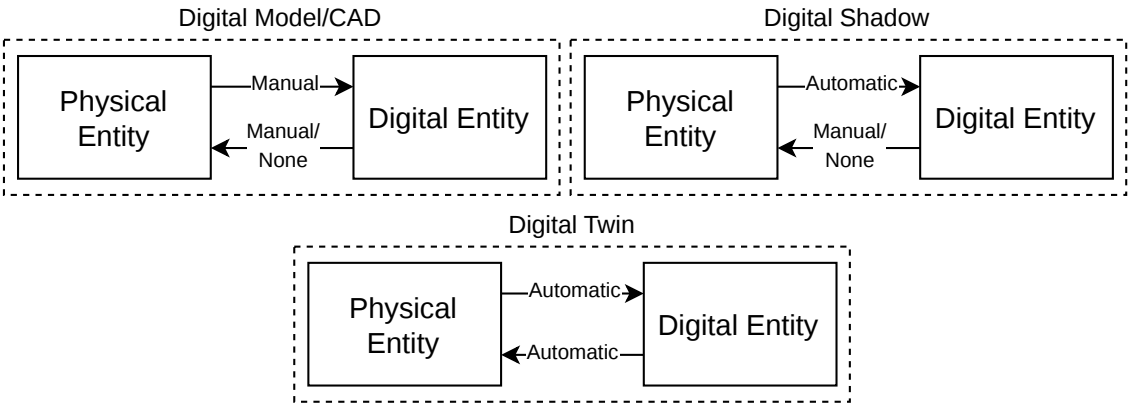


Figure 2.5 – Comparison of digital integration levels showing data flow characteristics between the physical entity and its digital representation

predict the future state of the physical system, enabling predictive decision-making that is fed back to it through control, maintenance, or planning actions. Together with the physical system, the digital system, and an optimization dimension, these connections make up their five-dimensional DT model [68]. Importantly, the authors do not require every connection to be automated. When the DT is used for control, the bidirectional flow needs to be automatic, but when it supports decision-making, as in PdM, the flow from virtual to physical "involves humans in the loop who carry out maintenance actions and, therefore, is not fully automatic but should be handled on time to avoid unexpected breakdown" [68, p. 4]. Jones *et al.* [66] make the same observation: when a predictive model in the virtual twin determines the health of a component and a mechanic is sent to replace it, the mechanic performs the realization of the V2P connection. They also note that the V2P connection is often omitted from descriptions of DTs, even though it is part of the original concept proposed by Grieves [60], and that without it the benefits of a DT are difficult to realize [66].

Finally, the two connections are characterized by how often they occur. Jones *et al.* [66] define twinning as the act of synchronizing the physical and virtual states, and the twinning rate as the frequency with which this synchronization occurs, noting that the literature usually describes it only as "real-time" [66, p. 42]. This complements the temporal integration dimension identified by Shao and Helu [67], which distinguishes real-time from offline twins. Taken together, these works show that the boundary between a DS and a DT depends not only on whether data flows in both directions, but on which parts of each flow are automated, which involve humans, and at what rate the virtual state is kept synchronized with the physical one. We applied these aspects in Section 2.5.6 to state the definition of DT we adopted in this dissertation.

2.5.4 Scope: What a Digital Twin Represents

Besides the direction and automation of its data flows, a DT is characterized by the boundary of the physical system it represents. The early definitions addressed individual objects: the twin proposed by Grieves [60] mirrored a product as it was manufactured, and the aerospace twin mirrored one specific, as-built vehicle [63]. As the concept spread, this boundary widened. VanDerHorn and Mahadevan [61] note that the physical system of interest can range from a single machine part

or component to a complete machine, a machinery system, or all the interconnected systems of an asset, and that its boundary is usually chosen according to its structure or purpose. Moreover, since a physical entity has many aspects that could be modeled, a DT is always constructed for the specific aspects relevant to the problem it is meant to solve, such as the fatigue failure of a shaft, and these must be defined before the twin is built [68].

At the narrowest scope, the DT represents a single asset. A typical example is the machine health DT, which uses process and equipment data to monitor, diagnose, and predict faults and failures of manufacturing equipment, and then either provides actionable recommendations to its users or sends control commands to the machine itself [65], [67]. The 5C architecture follows the same logic at its lower stages, where a cyber-twin is generated for each component and for each machine [32]. In an asset-level twin, the data that updates the virtual state and the actions derived from it concern the same physical object, which is the situation assumed by the DM, DS, and DT distinction discussed in Section 2.5.3.

At wider scopes, the DT represents a process, a production system, or an entire shop floor. Negri *et al.* [18] define the manufacturing DT as the virtual representation of a production system, and the DT Shop-Floor proposed by Tao and Zhang [64] comprises a physical shop floor, made up of humans, machines, and materials, a virtual shop floor that models it and provides control orders to its physical entities, a service system that encapsulates enterprise functions and algorithms, and the data fused from all three. Similarly, a scheduling and routing DT collects data from production equipment and from enterprise systems to optimize the allocation of materials, labor, and equipment [67], and in the 5C architecture the knowledge aggregated from component and machine twins provides the production system with self-maintainability and optimized production and inventory planning [32]. The ISO 23247 framework accommodates all these scopes, since the elements it allows to be twinned include equipment, materials, processes, personnel, and facilities [65].

Rather than one scope being more correct than another, Shao and Helu [67] argue that the viewpoint of the DT, whether product, process, or system, should be determined by the decision or control action it is meant to support, and that there should be no single DT of a physical system, but a variety of twins depending on their use. This choice has a direct consequence for how a DT is classified. In a system-level DT, the data may be collected from one element and the resulting action carried out by another, as when the virtual shop floor issues control orders to several physical entities. Whether such a DT closes the loop between its physical and virtual parts must therefore be assessed at the level of the system it represents, rather than for each of its elements individually. Consequently, we take this consideration into account in the definition adopted in Section 2.5.6.

2.5.5 Digital Twins in Maintenance

Maintenance is one of the most researched applications of the DT, which is consistent with the concept's original vocation for failure prediction discussed in Section 2.5.2. Since creating a DT can be costly, it is mostly applied where the maintenance of the asset itself is expensive, with manufacturing, the energy industry, and aerospace being the sectors where it is most frequently

found [29]. Its motivation in this domain is that PHM has traditionally been driven by the equipment in its physical space, with little connection to a virtual model of it [69].

A representative formulation of a maintenance-oriented DT is the five-dimensional model proposed by Tao *et al.* [69] for the PHM of complex equipment. It comprises the physical entity, with its functional subsystems and sensors; the virtual equipment, modeled at the levels of geometry, physics, behavior, and rules, the latter being what allows the model to judge, evaluate, and predict; a set of services, such as consistency judgment and fault prediction; the DT data, which fuses data from the physical and virtual parts with domain knowledge; and the connections among these elements, each characterized by its own sampling interval. The resulting workflow comprises three stages, observation, analysis, and decision, in which the maintenance strategy selected for a predicted fault is first validated on the virtual equipment and then carried out on the physical one. The authors also note that, given its implementation costs and complexity, such a DT is suited to high-value equipment for which sufficient data is available.

The role of the DT depends on the maintenance strategy it supports, as defined in Section 2.2. Relating the integration levels of Section 2.5.3 to these strategies, Errandonea *et al.* [29] associate CM and PvM with DMs, since their data flows are manual, and condition-based and PdM with DSs, since the digital model automatically obtains information on the state of the asset. The DT level is associated with PsM, in which the final step of proposing a maintenance action is provided. In this case, the DT is complemented with information on the maintenance processes and with optimization models, becoming part of a recommendation system that describes the activity to be carried out by the maintainer [29]. Likewise, the machine health DT described in Section 2.5.4 may either provide actionable recommendations to its users or send control commands directly to the equipment [65].

Finally, when the DT represents a production system rather than a single asset, the response to a failure need not be carried out by the failing asset itself. In the DT-based CPPS proposed by Ding *et al.* [70], disturbances such as machine breakdowns are recognized from the real-time data synchronized to the DT, which then selects a responsive scheduling strategy, decomposes it into commands, and synchronizes them with the physical shop floor, where parts and resources adjust their production plans accordingly. We address how prognostic estimates are produced, and how they are turned into prescriptive decisions in Sections 2.3 and 2.4, while comparing existing maintenance-oriented DT implementations in Chapter 3.

2.5.6 Definition Adopted in this Dissertation

The previous sections show that definitions of the DT differ in what the twin represents, in the nature of its model, and in how data flows between its physical and virtual parts. Since there is no universally accepted definition, we adopted one for the context of this dissertation. It builds on the generalized definition of VanDerHorn and Mahadevan [61], a virtual representation of a physical system and its associated environment and processes that is updated through the exchange of information between the physical and virtual systems, and on the view of the DT as a fit-for-purpose representation whose scope is set by the decision it supports [67]. In this dissertation, a DT is therefore understood as

> a fit-for-purpose virtual representation of the maintenance-relevant state of a production system, comprising the monitored assets and the resources that act upon maintenance decisions, whose state is automatically updated from sensor data (P2V), and which returns prescriptive decisions to the physical system (V2P), automatically when the resulting action can be executed by a machine, and through HITL when it cannot.

Each element of this definition is grounded in the literature reviewed above. The system-level scope follows the observation that a physical system of interest may range from a single component to a system of interconnected assets [61], and that a system-level DT may respond to the failure of one element through the actions of others [70]. The automated P2V connection corresponds to the updating engine of Thelen *et al.* [68], which updates the state of the DM from sensor data, at a twinning rate [66] suited to the dynamics of the degradation being monitored. The acceptance of humans in the V2P connection follows Thelen *et al.* [68] and Jones *et al.* [66], who recognize that in PdM this connection is often realized by the personnel who carry out the maintenance action, and Errandonea *et al.* [29], who associate the DT level of integration with PsM, in which a maintenance action is proposed.

This definition is deliberately less strict than others in two respects, which are acknowledged here. First, under the classification of Kritzinger *et al.* [62], which requires the data flows to be fully integrated in both directions, a twin whose V2P connection to a given asset passes through a human would correspond, for that asset considered in isolation, to a DS. Second, Wright and Davidson [71] consider the third essential part of a DT to be "a means of dynamically updating or adjusting the model in accordance with the data" [71, p. 2], whereas the definition adopted here, following Thelen *et al.* [68], requires only the state of the DM to be updated. Updating the model itself, for instance by recalibrating or retraining it as new data becomes available, is therefore regarded as a desirable extension rather than a defining requirement.

Applied to the system proposed in this dissertation (Chapter 4), the DT represents the monitored asset together with the AdM node. Its P2V connection updates the estimated HI, and the RUL derived from it, from sensor data once per second, and its V2P connection comprises the display of the current health estimate to the user, the automatic dispatch of a print order to the AdM node, and the installation of the replacement part by a human operator. In the prototype developed, the prognostic model is fixed and only fed new inputs, so the stricter criterion of Wright and Davidson [71] is not met, a limitation revisited in Chapter 7.

2.6 Distributed Control and IEC 61499

CPPS are increasingly implemented with distributed control across multiple devices, rather than clustered in a single controller. In this section, we contrast centralized and distributed control, present the IEC 61499 standard as a reference architecture for distributed control, and describe its FB model and its models of applications, devices, and resources.

2.6.1 Centralized and Distributed Control

Manufacturing control architectures are commonly classified according to where decisions are made. A centralized architecture is characterized by a single decision node that concentrates all

planning and information processing functions. It favors global optimization, but performs poorly in terms of speed of response, fault tolerance, and expansibility, especially in large systems [72]. When such centralized control is organized hierarchically, a single failure at one point of the hierarchy may bring the whole system to a halt [72]. In heterarchical architectures, by contrast, decisions are made locally and autonomously, which improves the response to disturbances at the cost of reduced global optimization, since no element has a global view of the system [72]. The same trade-off persists in recent work: centralized orchestrators can solve arbitrarily complex optimization problems but represent single points of failure, whereas fully decentralized approaches struggle to find optimal production plans [15].

Decentralized control has mainly been realized through agents and holons. An agent is an autonomous component that represents a physical or logical object of the system and acts to achieve its goals, interacting with other agents when it cannot reach them alone [72], while a holon is simultaneously a whole and a part of a larger system, such as a robot that is both a component of an assembly line and a system of sensors and actuators [16]. Since no central decision element exists, the loss of one component does not cause the failure of the others [72], and the decentralization of functions provides modularity, robustness, loose coupling, and reconfigurability on the fly [16].

Two related notions must be distinguished here. Decentralization refers to where decisions are made, whereas distribution refers to where the control software is executed. The two are complementary, as expressed by the goal of IEC 61499 of enabling automation in which "the intelligence is genuinely decentralized and embedded into software components, which can be freely distributed across networked devices" [73, p. 768]. In this dissertation, the term Distributed Control System (DCS) refers to a control system whose logic is decomposed into such components and executed across multiple networked devices.[2]

2.6.2 From IEC 61131-3 to IEC 61499

Industrial control has long been programmed according to the IEC 61131-3 standard, whose defining features include the cyclic processing of the application program on a stored image of the input and output values [74]. Each cycle is a single run of the periodically called program, and the time it takes is known as the cycle or scan time. Programs are structured in program organization units of three types, namely functions, which have no memory, FBs, which have their own data and can therefore retain state, and programs, which sit at the top of the user program and access the Programmable Logic Controller's (PLC) Input/Output (I/O), and are written in a set of textual and graphical languages [74]. In this standard, a FB is an encapsulated data structure with a defined algorithm working on that data [74].

IEC 61131-3 focused on standardizing PLC programming for single processors or small configurations of closely coupled processors, and the move towards large-scale distributed functionality created the need for a further standard to harmonize how functionality is defined and distributed [75]. When a DCS is built from several IEC 61131-3 PLCs, conflicts arise over which controller owns the write access to shared outputs, and the software architecture becomes more complex [76]. Its cyclic execution also affects performance: data updated by one PLC can only

[2]The term Distributed Control System is also used in the process industry to designate a particular class of commercial control systems. It is used here in its general sense, independently of that product category.

be processed by another at its next scan, so replacing cyclic scanning with event-driven invocation improves reaction times [76].

IEC 61499 was developed to address these needs. Its central artifact, the FB, was extended from the subroutine-like structure of IEC 61131-3 to a process-like abstraction taken from the theory of distributed computing [73], and the standard addresses requirements such as interoperability, portability, distribution, run-time reconfigurability, availability, and reliability [77]. Its adoption has nevertheless been slow, and it has been the subject of debate. Its critics argue that the standard fails to rigorously define its execution semantics, making the development of RTEs difficult, and that industry has been reluctant to adopt it due to the lack of a mature reference implementation and a steep learning curve [77]. Its proponents acknowledge unresolved semantic issues, the lack of clear development guidelines, and the scarcity of industrial-grade platforms as reasons for the slow adoption, while arguing that these do not determine the failure of the standard [78]. The semantic ambiguities have in fact led to several execution models, such as cyclic and synchronous models that combine the event-driven execution of IEC 61499 with the determinism of the PLC scan cycle [73], and to different behavior of the same application on different devices, owing to the lack of a definition of the execution models of devices, resources, and FBs [79]. IEC 61499 has since been applied in areas such as building management, process control, assembly, food processing, and material handling [73].

2.6.3 The Function Block Model

In IEC 61499, a FB is a functional unit of software with its own data structure, which can be manipulated by one or more algorithms [75]. Its interface separates events from data. The upper part of the FB, its execution control portion, maps incoming events onto the encapsulated functionality, defining which algorithms are triggered by the arrival of each event and when output events are emitted [75]. Events are associated with data through the WITH qualifier: when an input event occurs, only the input variables associated with it are updated, while the others retain their values, and when an output event is triggered, its associated output data are made available to the connected FBs [75]. FBs are therefore passive elements, whose encapsulated functionality is only invoked when triggered by an input event, rather than executed cyclically as in IEC 61131-3. They have no access to global variables, retain all their data between invocations, and rely on their containing resource for scheduling and for access to communication and process interfaces [75], as depicted in Figure 2.6. As a result, each FB can be implemented, tested, and reused independently of the others [78].

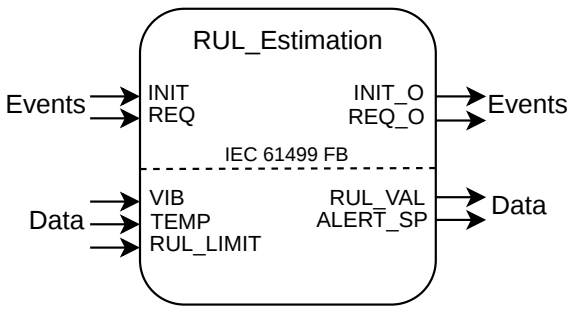


Figure 2.6 – Characteristics of an IEC 61499 FB (adapted from [75])

The standard defines several types of FBs. In a basic FB, the execution control portion is specified by an Execution Control Chart (ECC), a state machine that defines the logic by which algorithms are activated [76], [77], while the algorithms themselves can be written in any programming language, including those of IEC 61131-3 [73]. Several FBs can be combined into a composite FB, which looks like any other FB from the outside but has no execution control of its own, its behavior resulting from that of the FBs it contains [74]. Composite FBs thus support the hierarchical engineering of applications [79]. Finally, Service Interface Function Blocks (SIFBs) form the interfaces between the application and the controlled process, such as the inputs and outputs of the device, and between the application and the communication network [74]. Among them are communication FBs, such as client/server blocks, which formalize the exchange of data between FBs located in different processing resources [75].

2.6.4 Applications, Devices, and Resources

The primary purpose of IEC 61499 is not to define a programming methodology, but an architecture and a set of models for describing DCSs [75]. Beyond individual FBs, these include a system model composed of devices, a device model composed of independent resources, and an application model in the form of a network of FB instances [73]. An application is a network of interconnected FB instances that receives inputs from the controlled system through sensors and sends outputs to it through actuators [77]. A device is any control equipment able to contain resources and execute IEC 61499 applications [79], comprising an interface to the communication network, an interface to the controlled process, and the hardware on which its resources run [74]. A resource is a functional unit within a device that contains applications, or parts of them, and has independent control of its own operation [79], corresponding to a task in the general sense; several resources may run on the same device, executing the same or different applications [74]. The standard thus offers two views of a distributed program: a structural view, formed by the hierarchy of system, devices, and resources, and a user-oriented view, formed by applications and their FBs [74]. These models are illustrated in Figure 2.7.

This separation between applications and the devices that execute them is what allows control to be distributed. FBs are interconnected at the application level, independently of the resources on which they will run, and a single application may be implemented on one or more resources [74]. After an application is designed, a further design phase defines how its functionality is distributed onto the available processing resources [75], with communication SIFBs connecting the parts of the application mapped onto different devices. For this model to work across vendors, the standard includes provisions for portability, through an exchange format that allows tools from multiple vendors to produce and interpret the same FBs, applications, and system configurations, and for interoperability, allowing devices from multiple vendors to interact through communication systems [75]. How these provisions are implemented is specified in compliance profiles, whose rules are defined in Part 4 of the standard [75], [80]. By publishing a compliance profile, a vendor describes the extensions introduced in its implementation, and profiles have been published for aspects such as Ethernet communication and execution models [73].

In practice, an IEC 61499 tool chain comprises two main components. An IDE is used to edit FB designs and translate them into an executable form, and an RTE supports the execution

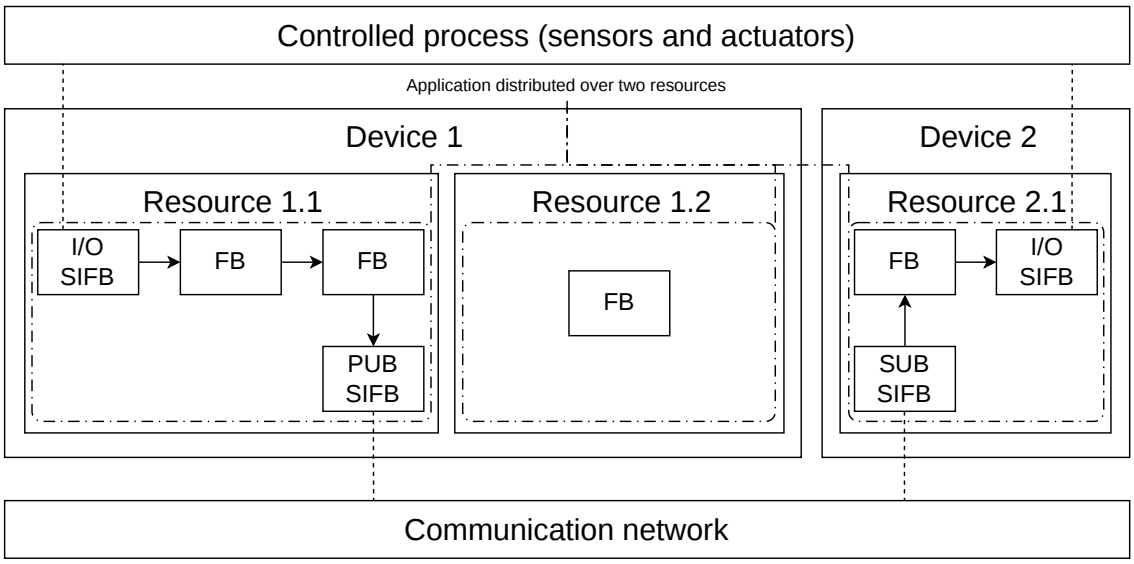


Figure 2.7 – IEC 61499 models of a distributed system: devices containing resources, and an application whose FBs are distributed across resources and connected through communication SIFBs (illustrative)

of that code on each device [73]. The two are connected through a device management protocol, which loads applications onto devices and creates or modifies FB instances [73], making it possible to reconfigure a program while it is controlling the plant [73], [80]. The RTE usually consists of libraries of FBs that implement service functions akin to device drivers, and is responsible for scheduling FB invocations, managing the flow of data and events, and interfacing with the peripherals [73]. The 4diac initiative, for instance, comprises the 4diac IDE [81] for modeling distributed control applications and the FORTE RTE [82] for executing basic, composite, and service interface FBs on embedded devices, with applications downloaded from the former to the latter according to a compliance profile that also specifies Ethernet-based communication between devices [80]. Since the execution semantics are not rigorously defined by the standard, the behavior of an application may depend on the RTE that executes it, as discussed in Section 2.6.2. We compare the specific RTEs considered for this dissertation in Chapter 4.

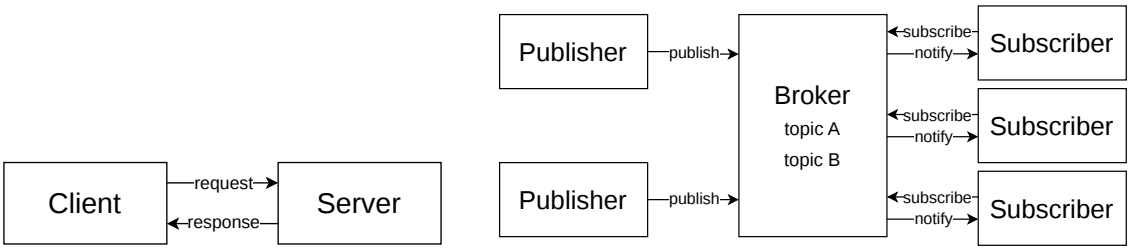
2.7 Industrial Communication

The nodes of a DCS must exchange data and commands reliably. This section presents the two main communication paradigms, client/server and publish/subscribe, and the two protocols used in this dissertation, Open Platform Communications Unified Architecture (OPC UA) [83] and Message Queuing Telemetry Transport (MQTT) [84], focusing on the features that determine their respective roles.

2.7.1 Communication Paradigms

DCSs exchange information according to two main interaction paradigms. In the first, remote invocation, one party requests an operation from another and waits for its result, extending the notion of operation invocation to a distributed context [85]; this is the basis of the client/server model. In the second, publish/subscribe, subscribers express their interest in an event or pattern of

events and are subsequently notified of any matching event generated by a publisher [85]. Publishers and subscribers do not interact directly, but through an event notification service, which stores and manages subscriptions and delivers events, acting as a neutral mediator between them. In the earliest and most widespread variant of this scheme, events are classified into topics identified by keywords, to which participants publish and subscribe [85]. The two paradigms are contrasted in Figure 2.8.



(a) Client/server **(b)** Publish/subscribe

Figure 2.8 – Client/server and publish/subscribe interaction paradigms

The main advantage of publish/subscribe is the decoupling it provides between the interacting parties [85]: in space, since publishers and subscribers do not need to know each other; in time, since they do not need to participate in the interaction at the same time; and in synchronization, since publishers are not blocked while producing events, and subscribers are notified asynchronously while performing other activities. This loose coupling makes the paradigm well suited to event-driven systems with many interacting nodes, such as the distributed control architectures discussed in Section 2.6.

2.7.2 Open Platform Communications Unified Architecture

OPC UA is a platform-independent standard for information exchange in industrial automation, providing a framework for information modeling, secure communication, and interoperability between heterogeneous systems and devices [83]. Its central concept is the address space, the set of objects and related information that a server makes available to its clients, represented as nodes connected by references [83]. These nodes and references form an information model, which can be seen as a graph that links nodes through typed references [86]. Because this model semantically annotates the data it exposes, clients can infer the meaning of data values without previous knowledge of the device [87].

OPC UA was originally built around a client/server model, in which servers provide sets of services through which clients access and discover the information in the address space [83]. It has since been extended with a publish/subscribe model that complements the client/server one, both being based on the same information model [88], and which can deliver messages through intermediary brokers. Security is integrated into the standard, covering the authentication of applications and users and the integrity and confidentiality of their communications [83], and is considered among the most complete of industrial communication protocols [89]. As a result of this device-centric focus on interoperability [87], OPC UA has become the de facto communication standard in industrial automation environments [89].

2.7.3 Message Queuing Telemetry Transport

MQTT is a publish/subscribe protocol designed for remote monitoring, providing low-latency and assured messaging over unreliable networks and the efficient distribution of data to one or many receivers [89], with a focus on a small code footprint and low network bandwidth [87]. It distinguishes clients, the programs or devices that use the protocol, from servers, commonly called brokers, which act as intermediaries between the clients that publish messages and those that have subscribed to them [84]. Each message carries a topic name, which the broker matches against the topic filters of the existing subscriptions, and these filters may express interest in one or more topics [84].

Delivery can be configured through three Quality-of-Service (QoS) levels [84]. At QoS 0, a message is delivered at most once, according to the capabilities of the underlying network, with no acknowledgment or retry. At QoS 1, the message is guaranteed to arrive at least once, but may be duplicated. At QoS 2, the highest level, it is delivered exactly once, for use when neither loss nor duplication is acceptable. The protocol also allows the broker to retain the last message published on a topic for future subscribers, to store a will message that is published if a client disconnects unexpectedly, and to preserve the state of a client's session across reconnections [84].

Since all messages pass through a broker, the way brokers are deployed shapes the robustness of the resulting system. A single central broker concentrates all communication in one element, whose failure interrupts it. The specification, however, allows a server to also act as a client of other servers [84], so that architectures may instead distribute communication across several brokers, for instance one per node or group of nodes, possibly interconnected. We describe the broker topology adopted in this dissertation in Chapter 4.

2.7.4 Complementary Roles

OPC UA and MQTT are therefore often seen as complementary rather than competing. MQTT adds the smallest overhead of the protocols compared by Profanter *et al.* [87], both when establishing a connection and when sending data, but it does not offer additional features such as remote procedure calls or discovery, whereas OPC UA provides a semantically described information model [87] and a more complete security framework [89]. In performance terms, the results are mixed: OPC UA has been found to show less variability but longer completion times than MQTT, whose behavior varies considerably with message size and rate [89], while OPC UA's publish/subscribe implementation was among the fastest of the protocols compared by Profanter *et al.* [87]. Finally, REpresentational State Transfer (REST) Application Programming Interfaces (APIs), typically built on HyperText Transfer Protocol (HTTP) [90], have also been advocated for industrial communication, to the point that extensions have been proposed to make OPC UA itself RESTful [86]. The protocols used for each type of communication in the proposed system are justified in Chapter 4.

2.8 Additive Manufacturing

The last element of the maintenance loop considered in this dissertation is the physical production of replacement parts. In this section, we introduce AdM and its process categories, focusing on

material extrusion, describing the digital process chain from a DM to a finished part, and discuss the factors that determine build time and the concept of digital spare parts.

2.8.1 Additive Manufacturing Processes

ISO/American Society for Testing and Materials (ASTM) 52900 defines AdM as the "process of joining materials to make parts from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing and formative manufacturing methodologies" [91, pp. cl. 3.1.2]. The standard groups AdM processes into seven categories according to how material is added [91]:

- **Binder jetting** — A liquid bonding agent selectively joins powder materials;
- **Directed energy deposition** — Focused thermal energy fuses materials as they are deposited;
- **Material extrusion** — Material is selectively dispensed through a nozzle or orifice;
- **Material jetting** — Droplets of feedstock material are selectively deposited;
- **Powder bed fusion** — Thermal energy selectively fuses regions of a powder bed;
- **Sheet lamination** — Sheets of material are bonded to form a part;
- **Vat photopolymerization** — A liquid photopolymer is selectively cured by light.

Material extrusion, also known as Fused Deposition Modeling (FDM) or Fused Filament Fabrication (FFF), is the category we used in this dissertation. In its most common form, a continuous filament of thermoplastic is pushed by pinch rollers into a heated liquefier, where it melts, while its solid portion acts as a piston that forces the melt through the nozzle [92], [93]. The nozzle scans a pattern that produces each cross-section of the part [93], depositing the semi-liquid material on the build platform or on top of the previously printed layers [94]. Typical materials include thermoplastics such as Acrylonitrile Butadiene Styrene (ABS), polyamide, polycarbonate, and PolyLactic Acid (PLA) [94]. The mechanical properties of the resulting parts depend mainly on the layer thickness, the width and orientation of the deposited filaments, and the gaps between them [94]. Parts produced by material extrusion have a characteristic ridged surface [92] and anisotropic mechanical properties, being stronger along the direction of deposition than across the bonds between adjacent filaments [92], partly due to the voids formed between them [94]. Weak mechanical properties, poor surface quality, and a limited range of materials are therefore considered its main drawbacks [94].

2.8.2 The Digital Process Chain

Regardless of the category, the production of a part by AdM follows a sequence of steps that Gibson *et al.* [93] divide into eight: conceptualization and CAD modeling, conversion of the model to a tessellated format, transfer and manipulation of that file on the AdM machine, machine setup, build, part removal and cleanup, post-processing, and application. The tessellated formats most commonly used are the STereoLithography (STL) format, which describes the surface geometry of an object as a set of triangles, and the Additive Manufacturing File Format (AMF) format,

which additionally supports colors, materials, lattices, textures, and metadata [91]. Since most AdM processes build parts layer by layer, the tessellated model must then be sliced, extracting the profile of each layer [93]. Slicing software, whether embedded in the machine or external to it, converts the DM into toolpath instructions that the printer can execute [93], commonly in the form of Geometric Code (G-code) for material extrusion printers.

Each step of this chain adds information to the part's digital description. For repeatable production, a spare part cannot be stored only as a geometric model, but must be complemented with the raw material specifications and the production parameters to be used, such as the layer thickness [95]. In practice, however, companies usually store the native CAD models, while the files converted for printing are not usually kept in their information systems [96]. In the digital spare part network described by Chekurov *et al.* [97], for instance, the models are only converted into files fit for manufacturing once a part has been requested. The stage at which a part is stored therefore determines how many steps remain before it can be printed: a CAD or tessellated model is independent of the machine but still requires slicing and setup, whereas a sliced file is ready to be executed, but specific to the machine, material, and parameters for which it was prepared. Figure 2.9 summarizes this chain and the two stages at which a part can be stored.

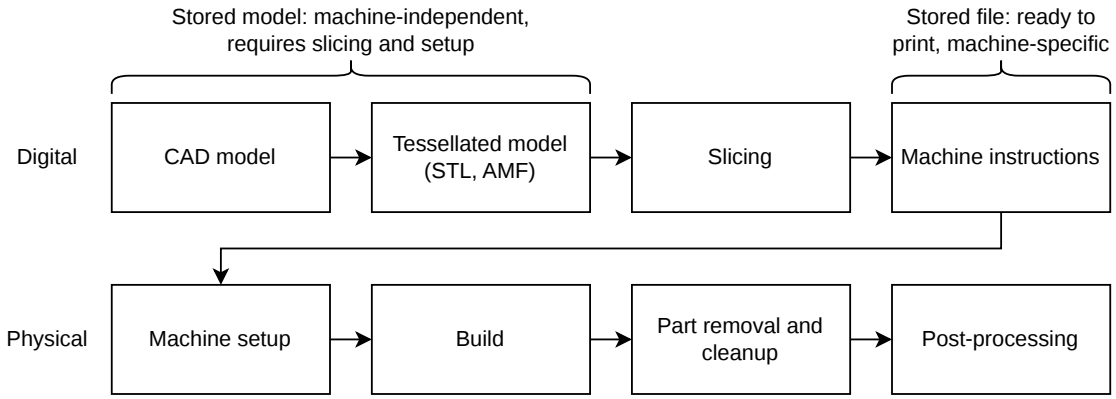


Figure 2.9 – Digital process chain of AdM, with slicing shown explicitly (based on [93])

2.8.3 Build Time and Digital Spare Parts

The time needed to produce a part by material extrusion depends on three groups of factors: the volume and mass of the part, the configuration of the printer, and its kinematic characteristics, such as the printing speed, the acceleration and deceleration of the print head, and the speed of rapid movements [98]. The printing speed is in turn limited by the rate at which the liquefier can melt the material and feed it through the nozzle [92], [93] and by the stiffness of the gantry that moves the print head [92], and smaller nozzles, while more precise, lead to much longer build times [93]. Since the toolpath is fully defined after slicing, the build time can be estimated in advance, although the estimates provided by AdM machines have been found to deviate from measured build times by around 5 to 10% [98]. To the build time must be added the time for part removal and cleanup and for any post-processing required [93].

These characteristics underlie the concept of digital spare parts, in which spare parts and their related manufacturing data are stored and transferred in digital form [96], so that defective components can be replaced by parts manufactured close to where they are needed, reducing repair

and delivery times and costs, emissions, and inventory [97]. This requires a digital warehouse to store and manage the digital parts and decide how and where each should be produced [95]. The concept is considered particularly suited to low-volume, complex parts [95], such as small and complex components, long-tail parts of spare part catalogs [97], and parts of old equipment [96]. However, industry experts have only partially confirmed the benefits reported in the literature, pointing to limitations such as the excessive need for post-processing, the difficulty of matching the quality of conventional suppliers, and inadequate information technology infrastructures and file formats [97], alongside the limited range of materials, unmet tolerances, and the limited size of build chambers [95], [96]. The lead time of an additively manufactured replacement part thus comprises its build time, which can be estimated from the sliced file or, when the same file is printed repeatedly, measured beforehand, together with the time required for removal, post-processing, and installation, and constitutes the lead time against which the prognostic horizon must be compared, as discussed in Section 2.3.5.

2.9 Summary

In this chapter, we introduced the concepts on which the remainder of this dissertation builds. Where the literature does not offer a single, consistent definition, a specific meaning was adopted, and we use these meanings throughout the following chapters. They are summarized in Table 2.4.

Table 2.4 – Summary of the main concepts and definitions adopted in this chapter

Concept	Meaning adopted	Section
PdM	Condition-based, and therefore preventive, maintenance carried out following a forecast of the degradation of the item	2.2.1
AM	Maintenance in which monitoring, analysis, decision, and, as far as possible, execution are carried out by the production system itself, as opposed to the operator-based meaning of TPM	2.2.2
Health indicator	Indicator of the health of the monitored item, which may be defined by a degradation function	2.3.2
RUL	Time from the current instant to the end of the useful life; when health is expressed through an HI, the time until the HI reaches its end-of-life level	2.3.1, 2.3.2
Timely action	A maintenance action is only useful if triggered while the remaining time still exceeds its lead time, plus a margin for uncertainty	2.3.5
PsM	Combination of failure prediction with knowledge of the available actions to determine which action to take and when, delivered as a recommendation or executed automatically	2.4.2
Degree of autonomy	Characterized per class of function (acquisition, analysis, decision, action) and by the position of the human in or on the loop	2.4.3
DT	Fit-for-purpose, system-level representation with automated P2V updates and a V2P connection that may involve humans	2.5.6
DCS	Control system whose logic is decomposed into components executed across multiple networked devices	2.6.1
Communication	Publish/subscribe for event-driven exchange between nodes; OPC UA for semantically described, supervisory access	2.7
Lead time of an additively manufactured part	Build time, estimated from the sliced file or measured in trial prints, plus removal, post-processing, and installation	2.8.3

Chapter 3

Literature Review

The use of prognostic estimates to take timely maintenance decisions, DTs supporting PdM and PsM, distributed control architectures handling failures and hosting analytics, and AdM for on-demand spare parts are the elements that the maintenance loop we propose in this dissertation brings together. In this chapter, we perform a literature review targeted at that interaction. While Chapter 2 defined the concepts involved, this chapter compares how existing works have combined them, identifies what remains open, and shows how the open issues motivate the research questions. Section 3.1 describes the review methodology, Sections 3.2 to 3.5 review each theme, and Section 3.6 reviews the studies that integrate several of them, synthesizes the findings into the research gaps, and derives from them the RQs of this dissertation.

3.1 Review Methodology

A preliminary search for studies combining the four elements of the proposed system, namely DTs, maintenance, IEC 61499, and AdM, returned no records in any of the databases used when restricted to titles, abstracts, and keywords. Since a review built on that intersection alone would be empty, the literature was reviewed as a set of themes, each corresponding to one link of the maintenance loop, followed by a theme dedicated to studies combining several links (Table 3.1).

Table 3.1 – Review themes and their screening questions

Theme	Topic	Screening question
T1	Prognostics and timely maintenance decisions (Section 3.2)	Does a prognostic estimate drive a decision that accounts for the time and means to act, or a prescribed maintenance action?
T2	DTs for predictive and prescriptive maintenance (Section 3.3)	Is a DT used for maintenance, and does its prediction lead to a decision or action, or connect to the systems that execute maintenance?
T3	Distributed control for failure handling and maintenance (Section 3.4)	Is distributed or decentralized control used to detect, predict, decide, or act on failures or maintenance, to reconfigure after them, or to host analytics in the control layer?
T4	AdM for on-demand spare parts (Section 3.5)	Does the study address the production of spare parts by AdM for the maintenance of other equipment, or link it to condition information?
T5	Autonomous maintenance and integrated approaches (Section 3.6)	Does the study combine at least three of the four elements, or claim machine-level autonomous maintenance?

3.1.1 Search Strategy

The searches were performed in Scopus, IEEE Xplore, and ScienceDirect in September 2026, restricted to titles, abstracts, and keywords (index terms, in IEEE Xplore). Each theme was covered by one to three queries combining two or three blocks of synonyms; T1, for instance, combined prognostic terms (*remaining useful life*, *prognos**, *predictive maintenance*) with terms related to the means of acting (*spare part**, *lead time*), and a separate query addressed prescriptive maintenance. The queries were adapted to the syntax of each database; since ScienceDirect does not support wildcards and limits the number of Boolean operators per field, its queries were expanded with explicit word variants and, where necessary, split into several searches. The complete search strings are listed in Appendix A. Results were limited to journal articles and conference papers written in English and published from 2015 onwards and, in Scopus and ScienceDirect, to the Engineering, Computer Science, and Decision Sciences subject areas, with Business added for the AdM theme, where much of the spare parts literature is published. Book chapters were accepted only in T5, given the scarcity of integrated approaches.

3.1.2 Screening and Eligibility

For each theme, the records retrieved from the three databases were merged and deduplicated, then screened by title and abstract and, for the remaining records, by full text. From T2 onwards, the records of the queries belonging to the same theme were screened together, since they shared the same screening question. A record was included if it met all of the following inclusion criteria:

- **IC1** — It is a primary study that proposed, implemented, or evaluated a method, model, architecture, or system, whether through simulation, a laboratory prototype, or an industrial case;
- **IC2** — It answers the screening question of its theme;
- **IC3** — It is written in English and published within the search period;
- **IC4** — It addresses equipment in manufacturing, process, energy, or transport contexts, public benchmark datasets being accepted.

A record was excluded if it met any of the following exclusion criteria:

- **EC1** — It is a review or survey, which was used only for context;
- **EC2** — It lacks a decision or action component where the theme required one, as in studies limited to prediction accuracy;
- **EC3** — It replicates an approach already represented by another included study;
- **EC4** — It addresses other domains, such as civil structures, healthcare, information technology, or finance;
- **EC5** — Its full text was not accessible. Records that were not open access and whose full text could not be retrieved through the institutional access of the University of Porto were also excluded;

- **EC6** — It uses a search term in a different sense, such as *lead time* meaning the warning time of a fault prediction, *autonomous maintenance* in the sense of TPM, *digital twin* abbreviated as DT for other concepts, or the maintenance of 3D printers rather than the production of spare parts.

Studies excluded under EC3, or included only for a specific point, were retained as citations in passing. For T5, conceptual architectures were accepted, and are identified as such in the comparison.

The number of records at each stage is given in Table 3.2 and summarized in Figure 3.1. The ambiguity of some search terms explains a large share of the exclusions: in T1, for instance, most records retrieved by the term *lead time* used it to describe how far in advance a failure was predicted, rather than the time needed to obtain a replacement part.

Table 3.2 – Records retrieved and retained at each stage of the review, per theme

	Retrieved					
Theme	**Scopus**	**IEEE**	**ScienceDirect**	**Unique**	**Full text**	**Retained**
T1a (decisions)	238	279	44	447	33	33
T1b (prescriptive)	82	29	12	90	32	21
T2 (DTs)	382	210	53	473	51	36
T3a (IEC 61499, faults)	163	140	13	191	31	16
T3b (IEC 61499, analytics)	18	21	3	30	13	6
T3c (agents)	79	106	13	161	45	9
T4 (additive spare parts)	428	171	103	549	35	28
T5 (integrated)	243	228	35	258	17	11
Total	1633	1184	276	2199	257	160
Four-concept query	0	0	0	–	–	–

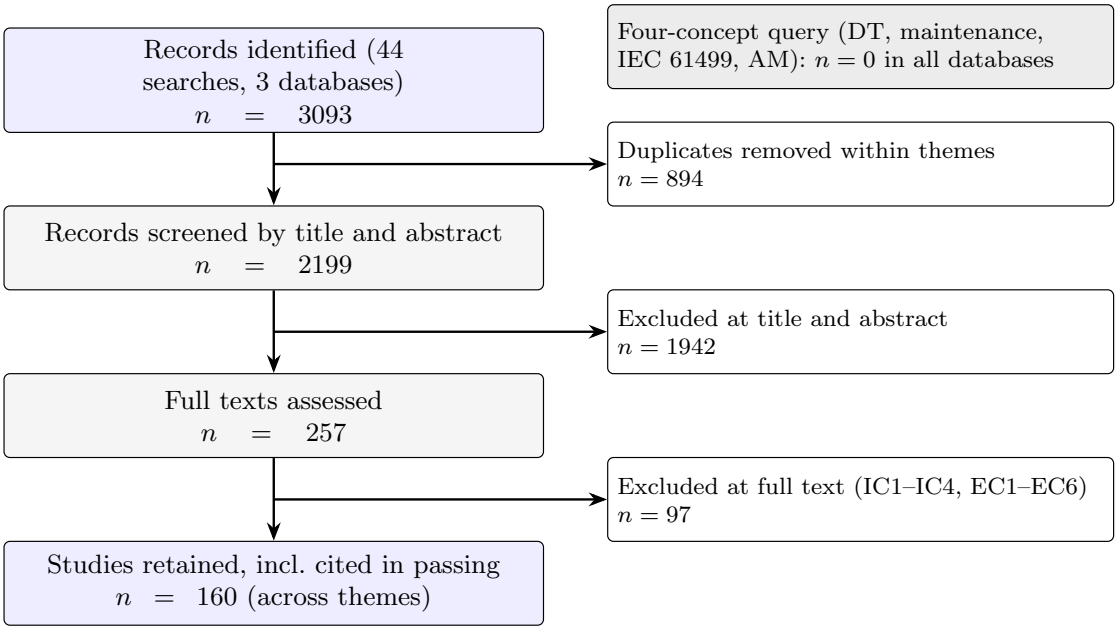


Figure 3.1 – Flow of records through the review

The 160 retained records include studies cited in passing, and some studies were retained in more than one theme. For each retained study, a structured extraction form was completed, recording the problem and application domain, the prognostic output and how it triggers a decision, whether uncertainty or the time to act are considered, what the DT represents and which data flows are automated, whether decisions are recommended or executed and the role of humans, the control architecture and communication protocols, the integration of AdM, the validation, and the limitations stated by the authors, complemented by items specific to each theme. These items correspond to the criteria defined in Chapter 2 and are the basis of the comparison tables in the following sections.

3.1.3 Limitations of the Review

The review is limited to three databases and to publications in English, and the screening was performed by a single reviewer. No backward or forward snowballing was performed; to mitigate this, the queries were designed with broad sets of synonyms and run in three complementary databases, and the results were checked against studies known a priori to be relevant, all of which were retrieved. The screening of the largest themes was precision-oriented, and studies addressing a theme with terminology not covered by the queries may have been missed.

3.2 From Prognostics to Timely Maintenance Decisions

This section reviews how RUL estimates have been turned into maintenance decisions that account for the time needed to act upon them, the first link of the maintenance loop considered in this dissertation. The reviewed studies are discussed below according to three aspects: how the prognostic estimate is coupled with the decision (Section 3.2.1), how the time available to act and the uncertainty of the estimate are taken into account (Section 3.2.2), and how, and by whom, the resulting decision is executed (Section 3.2.3). Table 3.3 compares the most representative studies according to these aspects. A complementary search on prescriptive maintenance frameworks, in which the prediction is turned into a prescribed action rather than an ordering or replacement time, is discussed in Section 3.2.4 and summarized in Table 3.4.

3.2.1 Coupling Prognostics with Maintenance and Spare Part Decisions

A recurring motivation in this literature is that RUL prediction and maintenance decision-making have mostly been studied separately, so that prognostic information is either produced without being used, or maintenance plans are optimized from assumed rather than predicted degradation [99], [100]. The dominant response has been the joint optimization of maintenance and spare part ordering, in which the predicted degradation of a component determines both when it should be replaced and when its replacement should be ordered. Wang *et al.* [101] model the degradation of a deteriorating system with a Wiener process, update its lifetime distribution from condition monitoring data, and use it to optimize the ordering and replacement times, re-evaluating the decision at each monitoring instant. The same structure appears in multi-component settings, where ordering and maintenance thresholds are derived from a prognostic condition index [102] or from RUL predictions under imperfect maintenance [103], and it has been progressively combined with deep

learning prognostics, such as Bayesian neural networks [99], Transformers with kernel density estimation [100], convolutional and Informer-based models [104], and ensembles [105]. Models close to the one used in this dissertation are also found: Chen *et al.* [106] combine a deep autoencoder with a bidirectional LSTM to estimate the RUL, and Wang *et al.* [107] use a CNN-bidirectional LSTM with quantile regression to time the replacement and ordering of cutting tools.

More recently, the decision itself has been learned rather than optimized. Wesendrup and Hellingrath [108] train a reinforcement learning agent that jointly plans spare part sourcing, production, and maintenance for a single machine, and Nie *et al.* [109] use two agents, trained in a fully decentralized manner, to coordinate maintenance scheduling and spare part inventory from probabilistic RUL estimates. These approaches remove the need for an explicit cost model, but, like their optimization-based counterparts, they are evaluated in simulation, mostly on the C-MAPSS turbofan dataset [99], [104], [106], [109].

3.2.2 Time to Act and Uncertainty

Most of the reviewed decision models recognize that a replacement cannot be carried out instantly after it is decided. Verbert *et al.* [110] make this explicit: to prevent a failure, the decision must be taken at least a certain time before it, corresponding to the time needed to bring personnel and material to the asset and to perform the maintenance, which introduces a trade-off between waiting for more accurate predictions and acting in time. Several studies translate this into thresholds. Wang *et al.* [111] introduce a scheduling threshold, at which maintenance resources are arranged, ahead of the maintenance threshold, at which the intervention takes place, with the distance between them determined by the lead time of the service providers. Similar logic underlies the maintenance preparation threshold of Cai *et al.* [103], the ordering rule based on the difference between the predicted RUL and the lead time of Zhao *et al.* [112], the division of the degradation process into decision intervals according to the inspection interval and the spare part lead time in Xia *et al.* [113], and the additional alarm thresholds that reserve time for ordering and transport before the failure threshold is reached in Lyu *et al.* [114].

The lead time itself is almost always modeled as a parameter of the procurement process: constant [99], [100], [102], deterministic and known [115], random [101], or split into regular and expedited options [105], [116]. Some studies simplify it further, assuming that no lead time exists [108] or that spare parts are always available [117], an assumption that Zhao *et al.* [112] identify as common in earlier maintenance models.

The uncertainty of the prognostic estimate is handled in increasingly explicit ways. Most recent studies predict a distribution of the RUL rather than a point value [99], [100], [109], and use it to decide conservatively: Wang *et al.* [118] base decisions on the lower bound of an uncertainty-aware estimate obtained with Monte Carlo dropout, Zhu *et al.* [119] penalize optimistic errors through an asymmetric loss and include the uncertainty of the maintenance task duration, and Rozas *et al.* [115] protect the decision against inaccurate RUL distributions through distributionally robust chance constraints. Simpler rules achieve a similar effect with a margin, as in Chen *et al.* [106], where the RUL threshold is adjusted by the mean absolute error of the predictor, or in Almatared *et al.* [120], where the RUL is calibrated to a lower quantile and a trigger is only issued when it remains below the threshold over a sustained period. The latter also evaluate the

resulting triggers with a lead-time accuracy metric, defined as the fraction of triggers for which the conservative RUL still exceeds the planning threshold.

3.2.3 From Decision to Execution

In most of the reviewed studies, the output of the decision model is a recommendation: a recommended maintenance or ordering time for managers or engineers to act upon [99]–[101], [107]. Only a few describe how the decision is connected to the systems that execute it. Bousdekis *et al.* [121] embed the decision model in an event-driven architecture, in which prediction events processed by a complex event processing engine trigger an online recommendation on when to maintain and when to order, validated in an automotive lighting plant. Chau *et al.* [122] integrate RUL prediction into a maintenance management system connected to the plant's Supervisory Control And Data Acquisition (SCADA), in which a rule-based engine compares the predicted life with the delivery lead time of components and creates work orders automatically. Almatared *et al.* [120] issue maintenance orders to a work-order system through a REST interface, and Park *et al.* [123] expose health index alarms to cell controllers and line dispatchers through OPC UA, MQTT, or REST adapters, triggering an automatic re-inspection. In the agent-based simulation of Basingab [117], maintenance actions are triggered automatically when the projected health of a machine drops below a preconfigured level, with sensor data streamed through an MQTT broker. The remaining industrial validations concern planning or prescription tools whose outputs are acted upon by maintenance staff [124], [125].

The automation of the ordering step itself has been proposed conceptually, with sensors triggering a replacement process that is either executed automatically or subject to the customer's approval [126], and, in the military domain, in the form of autonomic logistics systems in which health management information drives support decisions [127]. Practitioners, however, remain cautious: in interviews with manufacturing firms, an autonomous spare part ordering system was considered acceptable only if accurate and properly monitored, with people kept in control of the process [128]. Notably, in the only reviewed study that mentions AdM, Three-Dimensional (3D) printing appears in a company's current process map as a manual fallback, used when a spare part is not available [129].

3.2.4 Prescriptive Maintenance Frameworks

Studies explicitly framed as PsM extend the decision beyond the timing of replacement and ordering, prescribing which action to take among a set of alternatives. The prescriptions are generated in a variety of ways: through multi-criteria decision analysis, as in the fuzzy analytic hierarchy process used by Wittbrodt [130], which weighs failure risk against repair lead times, resources, and costs; through mathematical programming under uncertainty, as in the chance-constrained scheduling of Mao *et al.* [131] and Cho *et al.* [132]; through simulation, as in the discrete-event framework of Meissner *et al.* [133], where a predicted failure within the prognostic horizon triggers the evaluation of the available maintenance opportunities against logistic lead times and resources; or, increasingly, through deep reinforcement learning, fed by RUL estimates [134], [135], health indicators and health states [136], or directly by sensor observations [137]. Simple rule-based mappings are also found, translating health indicator zones and RUL estimates into actions

Table 3.3 – Comparison of representative studies coupling prognostics with timely maintenance decisions

Study	Decision driven by the prognosis	Time to act	Uncertainty	Execution and architecture	Validation
[101]	Ordering and replacement times	Random procurement lead time	RUL distribution	Recommendation	Aircraft system case
[102]	Maintenance and ordering thresholds	Fixed lead time	Stochastic degradation	–	Numerical
[111]	Scheduling and maintenance thresholds	Service provider lead time	–	–	Numerical
[110]	Timing of maintenance decision	Time for personnel, material, intervention	Accuracy vs. timeliness	–	Railway case
[103]	Preparation and ordering thresholds	Preparation time	RUL error	–	Simulation
[106]	RUL and degradation-state rules	–	Margin from predictor error	Recommendation	C-MAPSS
[99]	Dynamic maintenance and ordering	Constant lead time	RUL distribution	Recommendation	C-MAPSS
[113]	Order or maintain, per interval	Inspection interval and lead time	Rolling updates	–	C-MAPSS
[118]	Maintenance and ordering times	Ordering in advance	Lower bound (MC dropout)	Decision-maker selects	Public datasets
[119]	Conservative RUL-based decision	Uncertain task duration	Asymmetric loss	–	C-MAPSS
[115]	Joint maintenance and provisioning plan	Fixed, known	Distributionally robust	Rolling horizon	Wind turbine case
[109]	Learned maintenance and inventory actions	Delivery time	RUL distribution	Decentralized agents (learning)	Numerical
[121]	Maintenance and ordering times	Lead time	Stochastic	Online recommendation; event-driven architecture	Automotive plant
[122]	RUL vs. delivery and overhaul time	Delivery lead time	–	Automatic work orders; CMMS–SCADA, PLCs	Pilot (pump bearings)
[120]	Sustained RUL-threshold trigger	Planning threshold	Calibrated lower quantile	Orders issued via REST	Building HVAC data
[123]	Health index dual thresholds	Warning lead time	–	Alarms to cell controllers (OPC UA, MQTT, REST)	Synthetic timeline
[117]	Projected-health trigger	Unlimited spares assumed	–	Automatic; agents, MQTT	Simulation
[125]	Multi-objective prescriptions	Reorder points from lead times	Point estimates	Dashboard recommendations	Textile plant
[126]	Sensor-triggered replacement	Considered	–	Automatic or approved; cloud	Conceptual
This dissertation	Health indicator threshold	Build and installation time	Margin	Automatic print order; IEC 61499, MQTT	Laboratory testbed

ranging from continued monitoring to scheduled inspection [138]. The prescribed actions are not limited to maintenance: some approaches also adjust production sequences according to the health indicator [139], the use of the equipment [140], [141], or its operating point, in order to control the rate of degradation until the planned intervention [142].

As in the decision models of Section 3.2.3, the prescription is delivered almost always as a recommendation, whether as ranked scenarios for decision-makers [130], reports with ranked alternatives for authorization [143], advice transcribed into natural language [144], or guidance for the operators who execute it [145], and human experts remain in the loop, either authorizing the prescription or, in reinforcement learning approaches, shaping it through their feedback [134]. Automatic execution is found in only a few cases, and never for the physical replacement of a component. N. Björsell and B. -O. Gradén [142] transfer the prescribed pressure setpoint directly to the cascade controller of a pumping system, while pump replacements remain with technicians; Choubey *et al.* [146] execute remote software fixes automatically, but issue work orders to service personnel whenever a physical repair or downtime is required, explicitly accounting for the time needed to order parts and deploy personnel; Tham *et al.* [137] execute binary maintenance decisions on edge computing nodes in simulation; and Alonso-González *et al.* [147] propose an agentic framework in which prescriptions are executed through the enterprise systems, such as the Manufacturing Execution System (MES), Enterprise Resource Planning (ERP), and maintenance management systems, without direct human intervention, although only in a conceptual case study. DTs appear in several of these frameworks, mainly as environments in which candidate actions are simulated or evaluated before being released [143], [144], [147], or as the virtual counterpart hosting the prediction and decision models [131]. The architectures range from centralized platforms, such as the cloud-based workflow of Ordieres-Meré *et al.* [148], which also monitors and retrains the predictive models, to distributed ones based on edge nodes [137], federated learning [134], or multi-agent systems [143], [149], with OPC UA and MQTT as the communication protocols where they are specified [137], [143], [144]. Validation is dominated by public benchmark datasets and simulation, with industrial validation reported by Wittbrodt [130], over twelve months in a metalworking company, and by Ordieres-Meré *et al.* [148], on data from a petrochemical plant.

Synthesis

The coupling of prognostics with maintenance decisions is well established. The reviewed studies show how predicted degradation can determine both when to act and when to prepare the action, how the time needed to act can be encoded in thresholds or decision windows, how the uncertainty of the prediction can be absorbed through distributions, conservative estimates, robust formulations, or simple margins, and how prescriptive frameworks can select among alternative actions, including adjustments to production and operation. Three aspects, however, remain largely open. First, the time to act is always the lead time of an external procurement process, either fixed, random, or expedited, and in no reviewed study is the replacement part produced within the system on demand; AdM appears only as a manual fallback. Second, decisions and prescriptions are mostly evaluated on benchmark datasets or in simulation and are delivered as recommendations. Automatic execution is limited to operating setpoints, software fixes, or simulated decisions, never

Table 3.4 – Comparison of representative prescriptive maintenance frameworks

Study	Prognostic input	Prescription method	Execution and human role	Architecture	Validation
[133]	RUL within prognostic horizon	Discrete-event simulation of maintenance opportunities	Recommendation; mechanics execute	Simulation framework	Aircraft fleet simulation
[130]	Fuzzy risk levels, condition classes	Fuzzy AHP with lead times and resources	Ranked scenarios; managers select	–	Metalworking company, 12 months
[131]	RUL (Transformer)	Chance-constrained scheduling with production planning	Recommendation via service terminal	DT	C-MAPSS
[143]	RUL distribution, HI, fault evidence	Multi-agent alignment with cost–risk optimization	Reports for authorization; CMMS work orders	Multi-agent system, DT; OPC UA or MQTT	Public datasets, simulator
[148]	Temperature forecasts	Reinforcement learning	Recommendation; thresholds set by humans	Cloud workflow with model retraining	Petrochemical plant data
[144]	Production KPIs	Reinforcement learning with DT	LLM-transcribed advice; team executes	DT; OPC UA	Laboratory assembly line
[134]	RUL (Transformer)	Reinforcement learning with human feedback	Recommendation	Federated learning	C-MAPSS
[136]	HI, health states, RUL	Reinforcement learning	Recommendation; engineers decide	Centralized	Truck run-to-failure data
[137]	Sensor-based belief states	POMDP or reinforcement learning	Automatic binary decision	Edge nodes; MQTT, REST	Simulated cycles on edge hardware
[142]	HI (soft sensor)	Analytical setpoint optimization	Setpoint to controller automatically; replacement by technicians	Cascade control loop	Simulation; industrial soft sensor
[146]	Time to failure	Case-based reasoning and rules	Automatic software fixes; work orders otherwise	Local and global decision support	Healthcare equipment case
[147]	Failure probability, time to failure	Agentic AI, reinforcement learning, DT simulation	Automatic via MES, ERP, CMMS; human oversight	Agentic AI with DT	Conceptual case

extends to the physical replacement of a component, and, when the decision is connected to its execution, this is done through centralized information systems, such as maintenance management systems, work-order systems, event-processing platforms, or enterprise systems, rather than within the control architecture of the production system. Third, when the automation of the replacement process is proposed, it remains conceptual, and practitioners stress the need for humans to remain in control. For this dissertation, these observations suggest investigating whether the external procurement step can be replaced by on-demand production triggered directly by the prognostic decision, which would turn the lead time into a build time that can be measured in advance.

3.3 Digital Twins for Predictive and Prescriptive Maintenance

This section reviews how DTs have been used to support PdM and PsM, and how their outputs reach the systems and people that execute maintenance. Reviews of the field describe the typical maintenance DT as one that predicts potential failures and alerts the maintenance scheduler [150], delivering its insights through dashboards, maintenance tools, and enterprise systems [151], and point to the lack of shared platforms through which production and maintenance decisions can be coordinated [152]. Against this background, the reviewed studies are discussed according to what the DT represents and how it is updated (Section 3.3.1), how its outputs are turned into actions (Section 3.3.2), and which degree of autonomy is attributed to it (Section 3.3.3), and are compared in Table 3.5 according to the criteria introduced in Section 2.5.

3.3.1 Scope and Updating of the Twin

Most of the reviewed DTs represent a single asset: an industrial pump [153], a processing machine [154], a machining unit [155], a robot arm [156], a battery storage system [157], or an additive manufacturing machine, whose maintenance is itself optimized through a DT [158]. System-level DTs are less common. Ren *et al.* [159] propose a three-level DT for process manufacturing systems, with unit-level DTs of each device deployed on edge devices for real-time monitoring and analysis, station-level DTs on fog servers that optimize and control process parameters, and a shop-level DT in the cloud that supports production planning and sends decision instructions back to the fog servers. Other system-level examples represent a six-asset manufacturing cell [160], the interdependencies between machines as a graph [161], or the observable manufacturing elements of ISO 23247, from machines to production lines [162]. In industrial deployments, the DT is often defined by its data infrastructure rather than by a simulation model: Hamzah *et al.* [163] integrate around ten thousand live tags of an offshore facility through an asset model aligned with ISO 14224 and governed by the Asset Administration Shell, and Embarki *et al.* [164] combine vibration sensing, a historian, an asset repository, and analytics services around a reverse osmosis unit.

All the reviewed DTs update their state from sensor data, at rates ranging from about 10 Hz in a laboratory sorting machine [165] to every few seconds in a machining unit [155] or once every set number of charge and discharge cycles for a battery [157]. Updating the model itself, which several authors consider an essential feature of a DT [166], [167], is less frequent and takes different forms: online re-estimation of the model parameters over a sliding window [155],

retraining when the divergence between DT and asset exceeds a threshold [160], or feedback of maintenance outcomes into the models [153], [159], while mechanisms to keep the models of complex DTs consistent as they evolve are still being developed [168], [169].

3.3.2 From Twin to Action

The V2P connection of the reviewed DTs follows three main paths. In the first, and most common, the DT supports a human decision. Embarki *et al.* [164], in the most complete industrial deployment reviewed, link confirmed alerts to the generation of work orders in the maintenance management system and to execution feedback, but explicitly describe the result as a decision-support system that improves the timeliness and traceability of human intervention, rather than full autonomy, which they consider inappropriate in a safety- and quality-critical environment. Similar HITL arrangements are found in the recommendations of advisory layers [153], [157], in predictions reviewed by engineers [154], and in DTs supporting technicians or operators [170]–[173]. Reporting on eighteen months of use of system-level DTs in industrial processes, Hassan *et al.* [174] stress that close communication between the DT and the maintenance management system is needed to reduce false alarms.

In the second path, the DT's output reaches the enterprise systems that manage maintenance. Nozari and Szmelter-Jarosz [175] transmit failure probabilities and maintenance recommendations from the DT to an ERP system through a REST interface, where they trigger automated maintenance actions and schedule updates, validated in an industrial composting facility. Hamzah *et al.* [163] connect alerts to the maintenance management system through an execution loop gated by the value of information, in which operators raise pre-populated work orders directly from alert cards, evaluated over twelve weeks of live operation. Ariansyah and Pardamean [176] use ontology rules so that a failure identified by the DT triggers an inventory check and, if the stock of the spare part is below its minimum, a purchase work order, and Wang *et al.* [177] build DTs of the equipment of an enterprise to forecast its demand for spare parts. Planning and execution systems are also reached through the MES, which treats the DT's maintenance recommendations as earliest feasible times [158], integrates a reinforcement learning planner that includes the RUL of machines in its state [178], or reschedules production after a fault diagnosed by the DT [179], and ERP integration has been proposed as the condition for maintenance decisions to be executed at all [180].

In the third path, the DT acts directly on the controlled process. Xu and Wang [155] use predicted deviations of a machining unit to adjust the gains of a Proportional-Integral-Derivative (PID) controller proactively, sending the updated control signals back to the physical controller through OPC UA, and B. Soykan *et al.* [165] publish control commands from a discrete-event DT to Raspberry Pi controllers through MQTT, adjusting the speed and sorting mechanism of a laboratory machine. Similarly, Myakala *et al.* [181] wrap the output of a reinforcement learning scheduler into commands, such as scheduling maintenance or reducing load, communicated through MQTT or OPC UA, although only evaluated on benchmark datasets, and Badoi *et al.* [182] plan a laboratory architecture in which the results of anomaly detection trigger automated maintenance workflows through bidirectional control flows.

3.3.3 Autonomy of Digital Twin-Based Maintenance

The degree of autonomy granted to these DTs is addressed explicitly by only a few studies. Ma *et al.* [183] adapt the levels of autonomy of autonomous vehicles to predictive maintenance, defining six levels, from no automation to full autonomy, and the functional requirements a DT must meet to reach each of them, illustrated conceptually on a bridge. Kamoise *et al.* [184] treat the DT and the human as cooperating agents and allocate decisions between them according to a level of automation, discussing an intermediate level at which the human keeps a veto over the DT's decisions. When actions are executed automatically, their authority is bounded: R. Sakha *et al.* [160] intercept every action proposed by the DT's agents before it reaches the equipment, executing safe actions, blocking unsafe ones, and escalating uncertain ones to a human, and M. M. R. Shamim *et al.* [161] let predictions with low uncertainty be executed without oversight while submitting uncertain ones to human review. At the level of the whole system, L. Mendonça *et al.* [162] propose an ISO 23247-aligned architecture in which an agentic module communicates action plans to operators, who validate them and feed their knowledge back, although it is validated only conceptually.

Synthesis

DTs have become a common vehicle for predictive and, increasingly, prescriptive maintenance. The reviewed studies show DTs that host prognostic models and decision logic, that are integrated with maintenance management, ERP, and MES systems, in some cases in industrial deployments, that act automatically on control parameters or machine commands, and that bound the authority of automated actions through safety checks, uncertainty thresholds, or human vetoes. Three aspects, however, remain open. First, most DTs represent a single asset, and the few system-level DTs act on production planning, process control, or maintenance scheduling rather than on the resources needed to carry out a maintenance action. Second, the virtual-to-physical connection ends at a work order, a purchase order, a schedule, or a control parameter; the physical maintenance action and the provision of the replacement part remain with humans or external suppliers, and no reviewed DT triggers the production of the part it has predicted will be needed. Third, the architectures that connect the DT to execution are centralized in cloud platforms, historians, enterprise systems, or microservices, with edge devices used for acquisition and analytics, and none uses a distributed control architecture such as IEC 61499 as the layer through which the DT's decisions are executed. For this dissertation, these observations suggest extending the scope of the DT to the system that carries out the maintenance action, so that its virtual-to-physical connection can reach a decentralized manufacturing resource, such as a printer producing the replacement part.

3.4 Distributed Control for Failure Handling and Maintenance

This section reviews how distributed control architectures, and IEC 61499 in particular, have been used to handle equipment failures and to support maintenance. Section 3.4.1 addresses the use of IEC 61499 to detect faults, reconfigure control applications, and recover from failures; Section 3.4.2 addresses the execution of analytics and machine learning models within IEC 61499

Table 3.5 – Comparison of representative DT-based maintenance studies

Study	Twin scope	Model updating	Virtual-to-physical path	Human role and automation	Validation
[164]	Single unit	–	Alerts to CMMS work orders, execution feedback	Decision support; human intervention by design	Industrial deployment
[163]	Offshore facility (AAS)	–	Gated loop to CMMS	Operators raise work orders	12 weeks live
[175]	Critical machines of a plant	–	Recommendations to ERP via REST	Automated maintenance actions and schedules	Industrial facility
[176]	Asset hierarchy	–	Ontology rules to CMMS and ERP	Automatic purchase work orders	Ontology model
[159]	Unit, station, shop levels	Feedback adjustment	Cloud decisions to fog servers	Hierarchical decisions	6-month test
[158]	AdM machine	Continuous data	Recommendations to MES as earliest times	Risk-checked scheduling	Simulation
[178]	Production system	–	RL planner integrated with MES	Closed-loop planning	Industrial pilot
[155]	Machining unit	Online, sliding window	PID gains to controller via OPC UA	Automatic	Machining unit
[165]	Sorting machine	–	Commands to Raspberry Pi via MQTT	Automatic	Laboratory
[160]	Six-asset cell	Retraining on drift	Agent actions after safety test	Uncertain actions escalated	Manufacturing cell
[161]	Machine graph	–	Maintenance priorities	Uncertain predictions reviewed	Dataset
[184]	Manufacturing line	–	Cooperative decisions	Human veto	Learning platform
[162]	ISO 23247 elements	Operator feedback	Action plans to operators	Human validation	Conceptual
This dissertation	Asset and AdM node	None (fixed model)	Print order to AdM node via MQTT	Automatic print; human installation	Laboratory testbed

applications; and Section 3.4.3 addresses agent-based approaches to predictive maintenance. The representative studies are compared in Table 3.6.

3.4.1 Fault Handling, Self-Healing, and Reconfiguration

The reconfiguration capabilities of IEC 61499 have mostly been exploited to make control systems resilient to faults. At the level of the control logic, Leitão *et al.* [185] combine fault diagnosis in discrete event systems with supervisory control to compute fault-tolerant control strategies, and use the dynamic reconfiguration of IEC 61499 to switch automatically to the control logic suited to the diagnosed fault. At the level of the devices, Atmojo *et al.* [186] distribute the control of a flexible assembly line across ten controllers, so that production continues when one of them fails, with another controller taking over its responsibilities, and parts of the system can be updated while it runs. Zhou *et al.* [187] go further, combining model-driven engineering with an ontological knowledge base whose rules automatically initiate the reconfiguration of a machine control system in response to external or internal events, executed on an IEC 61499-compliant runtime that provides the management functions required for dynamic reconfiguration. In the power domain, where IEC 61499 has been combined with IEC 61850, the same principle underlies self-healing distribution grids, in which switching devices act as agents that locate and isolate faults and restore supply among themselves [188], [189].

Several studies place these capabilities within a self-management loop. W. Dai *et al.* [190] propose an autonomic service management layer, built on a service-oriented IEC 61499 runtime in which each function block is mapped to a service, whose aim is to allow industrial cyber-physical systems to manage themselves with minimal human intervention; self-healing is achieved by controllers that discover faults in themselves or in other devices and relocate control logic accordingly. In a subsequent work, Dai *et al.* [191] move fault identification to a cloud-based decision support system, which runs parallel simulations of the control software ahead of real time, analyzes their outputs with fault trees modeled as function blocks, and, when a fault is found, overrides the outputs of the controllers and reconfigures the control software through IEC 61499 management commands. A similar division of labor is advocated by Prenzel and Steinhorst [192], who note that dynamic reconfiguration is rarely used in practice because the standard does not explain how to achieve it safely and in real time. They propose a hybrid self-adaptation architecture in which monitoring and execution are decentralized, while the computationally heavy analysis and planning are centralized, and measure the adaptation times achievable on a Raspberry Pi running 4diac FORTE. In agent-based variants, software agents supervise machining stations controlled by embedded IEC 61499 processors and reconfigure part routings when equipment fails randomly [193], or diagnose faults in the function block applications themselves, presenting their diagnosis to engineers [194].

Only one of the reviewed studies uses a prediction, rather than a detected fault, as the trigger. Kang *et al.* [195] propose an automatic deployment process for reconfigurable edge applications, in which a fault prediction module signals the unhealthy state of a device and an optimization algorithm redeploys its function blocks to suitable devices before the system fails, evaluated in simulation for a nuclear power control system. In all cases, the response to the failure is a reconfiguration of the control software, whether by switching logic, relocating function blocks, or

rerouting production, rather than a maintenance action on the failing equipment.

3.4.2 Analytics and Machine Learning in IEC 61499 Applications

IEC 61499 applications have also been used to host data analytics close to the equipment. In its simplest form, the analysis is implemented directly as function blocks, as in the online diagnosis of poorly tuned controllers proposed by Nindyasari *et al.* [196], whose function blocks exchange data through MQTT and were tested with data from a fluid catalytic cracking plant. Python-based runtimes have made this considerably easier. DINASORE executes function blocks written in Python, giving direct access to scientific computing and machine learning libraries at the edge [197], and Silva *et al.* [198] use it on a Raspberry Pi connected to the Controller Area Network (CAN) network of heavy-duty machines to acquire signals and compare them with reference behavior, producing similarity metrics that technicians interpret remotely to find faults. Cruz Salazar *et al.* [199] embed predictive agents, developed in a Python agent framework and using supervised learning, within PLC-based control environments that follow IEC 61131-3 and IEC 61499, with FORTE as the runtime, and connect them to DTs through the Asset Administration Shell, in a laboratory setup. When the model is not hosted in the control application, it is connected to it through standard interfaces, such as an OPC UA server that bridges a Python-based vision function into an IEC 61499 application [200], or gateways that stream data from IEC 61499 applications to cloud platforms running artificial intelligence services [201]. Dobrescu *et al.* [202] envisage both options, with reusable algorithms that can either be downloaded to the controllers or executed remotely, returning only their results to the process.

Running analytics alongside control raises computational and timing concerns. Low-cost platforms such as the Raspberry Pi can host IEC 61499 runtimes and act as industrial controllers [203], but the devices of a heterogeneous system differ widely in computing performance, and Lyu *et al.* [204] propose an optimizer that decides on which device each component should be deployed according to its computational complexity, across platforms ranging from low-cost controllers to edge nodes and cloud servers. Gao *et al.* [205] propose prioritized, deterministic execution semantics that allow real-time control to coexist with computational tasks, such as a convolutional neural network for defect detection deployed within the same function block network on an edge node, while treating tasks without hard real-time constraints, such as sending operational data to the cloud for fault prediction, as lower-priority tasks that give way to control when resources are limited. At the level of a whole workshop, Wang *et al.* [206] model a DT with IEC 61499 components that evaluates the health status of stations and lines and dispatches commands to the production system through a control middleware.

3.4.3 Agent-Based Approaches to Predictive Maintenance

Multi-agent and holonic systems address a complementary question: how the decisions derived from health information are coordinated across the resources of a production system. Several studies use agents to integrate predicted maintenance needs into production control. Foerster *et al.* [207] integrate condition-based maintenance orders into autonomous production control through a reversed contract net protocol, in which machines act as managers and both production and maintenance orders bid for machine time, treating predictively generated maintenance orders as

uncertainties of the production system; the approach was validated in a workshop with managers from industry. Bencheikh *et al.* [208] propose a multi-agent framework in which supervisor, customer, maintainer, and producer agents interact through a blackboard to schedule production and predictive maintenance jointly, using the probability density function of the RUL of each device provided by a prognostic module, and operating without human intervention in computational experiments. Babu *et al.* [209] use multi-agent reinforcement learning, with centralized training and decentralized execution, to decide jointly on maintenance interventions and production rescheduling, with a safety layer that checks production feasibility, maintenance resource availability, the projected machine condition, and active interlocks before any proposed action is applied, evaluated in simulation.

Agents have also been used to structure the analysis itself. In the framework of Peres *et al.* [210], monitoring agents acquire and preprocess data at each node, machine learning models run within a distributed stream-processing cluster, and their predictions are sent to the runtime decision module of the production system, which may trigger its self-reconfiguration or issue maintenance recommendations to operators; the models can be updated at runtime without stopping the system. Sekar and Nezamoddini [211] orchestrate pipelines of anomaly detection and failure prediction agents on edge nodes, reconfiguring them according to local conditions, and T. Borangiu *et al.* [212] combine a holonic control architecture with cloud-embedded DTs of resources, orders, and the system, in which machine learning models predicting resource performance run centrally in high-performance computing clusters, while agents reallocate operations when a resource breaks down, demonstrated in a four-robot manufacturing cell. In the most recent works, agents are complemented by large language models, but authority remains with deterministic components or humans: Guha and Datta [213] commit alerts only through a deterministic decision agent, with a language model providing explanations without influencing the decision; González-Potes *et al.* [214] confine the language model to an advisory role, keeping safety-critical logic in deterministic agents and in the PLC and SCADA systems of an industrial beverage plant; and Farahani *et al.* [215] present the recommendations of a centrally orchestrated agent system to human experts for final verification.

Synthesis

Distributed control architectures based on IEC 61499 have been shown to support fault detection, fault-tolerant control, self-healing, and the dynamic reconfiguration of control applications; runtimes such as DINASORE, together with agent frameworks embedded in controllers, make it possible to execute data-driven analytics within the control layer at the edge; and multi-agent systems have been used to coordinate predicted maintenance needs with production, in some cases without human intervention. Their use for maintenance, however, remains limited in four respects. First, in IEC 61499 applications the triggers are almost always faults that have already been detected, and only one reviewed study uses a failure prediction, in simulation, to reconfigure the system before it fails. Second, the response to a failure or prediction is a reconfiguration of the control software, a rescheduling of production, or a recommendation, and never the execution of a maintenance action, let alone the production of a replacement part. Third, the analytics hosted in the control layer are mainly diagnostic, while prognostic tasks, which lack hard real-time constraints and are

Table 3.6 – Comparison of representative studies using distributed control for failure handling, analytics, and maintenance

Study	Trigger	Response	Where analysis runs	Platform and communication	Validation
[185]	Diagnosed fault	Switch to fault-tolerant control logic	Control layer	IEC 61499	Fluid mixing case
[186]	Controller failure	Another controller takes over; online updates	Distributed controllers	IEC 61499 controllers; UDP	Assembly line
[187]	External or internal events	Rule-based self-reconfiguration	Knowledge base with runtime management	IEC 61499 runtime; OPC UA	Motion control prototype
[190]	Faults in devices	Relocation of control logic	Autonomic service layer	Service-oriented IEC 61499 runtime	Simulated baggage handling
[191]	Faults found in simulation ahead of time	Output override; reconfiguration by management commands	Cloud	IEC 61499 on controllers and cloud; industrial Ethernet	Water heating case
[192]	Unanticipated failures, attacks	Self-adaptation	Decentralized monitoring and execution; centralized planning	4diac FORTE, Raspberry Pi	Measured adaptation times
[193]	Random equipment failures	Rerouting of parts by agents	Agents over IEC 61499 stations	FIPA agent messages	Agent-based simulation
[195]	Fault prediction	Redeployment of function blocks	Edge	IEC 61499	Simulation
[198]	Remote fault finding	Similarity metrics for technicians	Edge (Raspberry Pi)	DINASORE; CAN	Heavy-duty machine
[199]	Supervised-learning predictions	Adaptive decisions by predictive agents	Agents embedded in PLC-based control	Python agents, FORTE; AAS	Laboratory tank, simulated DTs
[210]	Failure likelihood from stream analytics	Self-reconfiguration or recommendations	Agents and distributed stream processing; runtime model updates	JADE, Kafka, Storm; OPC UA	Cluster testbed, simulated data
[208]	RUL probability density	Joint production and maintenance schedule	Agents on a blackboard	C++ implementation	Computational experiments
[209]	Machine health estimates	Maintenance and rescheduling actions after safety checks	Edge; decentralized execution	Multi-agent RL	Simulation
[212]	Predicted resource performance	Reallocation of operations by agents	Central HPC cluster; holonic control	Cloud-embedded DTs	Four-robot cell
This dissertation	Health indicator threshold	Automatic print order for replacement part	Edge nodes	DINASORE; MQTT, OPC UA	Laboratory testbed

computationally more demanding, tend to be treated as lower-priority tasks or moved to cloud or central services, and the agent platforms used for maintenance decisions are generally separate from the control layer, with only a few studies integrating them with IEC 61499 or PLC-based control. Fourth, where learning-based decisions are allowed to act, their authority is bounded by deterministic rules, safety checks, or human verification. For this dissertation, these observations suggest using the distributed control layer not only for the control of the production system, but also to host the prognostic model and the prescriptive decision, and to carry the resulting action, within explicit bounds, through to the physical maintenance workflow.

3.5 Additive Manufacturing for On-Demand Spare Parts

This section reviews how AdM has been used to supply spare parts for maintenance, and how its production has been connected to maintenance decisions. The reviewed studies are discussed according to how parts suitable for AdM are selected and when on-demand production pays off (Section 3.5.1), how parts are stored and prepared for production, and how long that production takes (Section 3.5.2), and how the production of a part has been linked to the condition of the equipment (Section 3.5.3). The representative studies are compared in Table 3.7.

3.5.1 Selecting Parts and Sourcing Options

A large share of the literature addresses which spare parts should be produced by AdM. Applied to a manufacturer's catalog of more than fifty thousand parts over nine years, the assessment of Heinen and Hoberg [216] found that up to 8% of the stock keeping units, and 2% of the units supplied, could be produced more economically by AdM. Structured selection methods narrow large catalogs down to a few candidates through technical and economic criteria [217]–[219], and decision models classify parts between AdM and conventional manufacturing according to demand, costs, lead times, and the uncertainty of failure rates [220], [221]. Peron *et al.* [220], for instance, found AdM to be the most cost-effective option in 76% of the scenarios analyzed, but in fewer scenarios once uncertainty in the failure rates of the printed parts was considered. At the level of the supply chain, decision support systems and optimization models address where printers should be located [222], how the supply chain should be configured [223], whether an AdM machine should be purchased rather than holding stock or buying urgently [224], and how AdM can serve as a second sourcing option combined with conventional supply [225]. The conclusions are nuanced: in many cases, on-demand AdM is not cost-competitive with a conventional warehouse [226], and it is best suited to parts with low demand or high demand variability [227], or to parts that are unavailable, cannot be bought separately from their assembly, or are expensive, as reported in a case in which a metal producer began printing polymer spare parts in-house with two material extrusion printers [228].

3.5.2 Digital Inventory and Production Time

The digital inventory is usually understood as a library of geometric models. Parts are exchanged as STL files, whether between the sites of a distributed manufacturing scenario through a product lifecycle management system [229] or as the output of reverse engineering [230], and must still be

sliced before printing [219]. Only Montero *et al.* [231] deliver a machine-ready build file, containing the pre-processed data and machine parameters, to the facilities where the part is produced, so that the deployed team does not need expertise in design for AdM. Building this inventory is itself a major effort: the digital model is often assumed to exist [216], [226], but in practice only 54 of the 100 technical drawings requested by Chaudhuri *et al.* [217] could be obtained, and the costs of digitalizing parts are part of the decision [218], [223]. When parts are no longer available, they are reverse engineered from the worn component and reproduced, as in the case of turbine blades [230] and pump impellers [232], [233].

The production time of a printed part is rarely short. Printing, heat treatment, support removal, and post-processing of a single metal bushing were reported to take about 12 hours [227], a batch of 121 small parts required between about 24 and 47 hours of build time depending on the material [234], and post-processing was found to be the most time-consuming phase of the whole chain [232]. Nevertheless, compared with conventional procurement, AdM reduced the delivery time of a replacement part by 86% in a pilot study [235], and a digital warehouse combined with on-demand production was modeled with delivery times of five to fourteen days [236]. Where production time enters the models, it is either estimated from the volume of the part [226] or treated as a lead time comparable to that of a supplier [216], [220].

3.5.3 Linking Additive Production to Equipment Condition

In most of the reviewed studies, the production of a part is triggered by a demand, whether a failure, an order, or a planned stock level, and decided by maintenance planners, procurement teams, or a central factory [218], [229]. Li *et al.* [227] go further, letting a maintenance operator who owns dedicated AdM equipment produce a number of parts in advance of each maintenance task, but base this decision on the distribution of maintenance demands rather than on the condition of the equipment. Only one reviewed study uses condition information: Cubillo *et al.* [237] combine the health-monitoring information of an aircraft component with a cost and downtime model to select among repair options that include AdM, and present the recommended option to the person responsible, who may accept or override it.

DTs appear in this literature mainly in repair scenarios. Bender *et al.* [238] describe the flow of information in a DT that links on-site optical inspection, in which defects are identified automatically, to the slicing and directed energy deposition repair of a large component, demonstrated in an industrial case, and Nouri Rahmat Abadi *et al.* [239] use a DT to generate the toolpaths of a robotic repair of wind turbine blades, while humans still identify the defects. Similar digital models support the additive restoration of railway wheelsets [240], and assistance systems recommend additive repair processes for damaged parts [241]. For the supply of spare parts, Peron [242] argue, in a position paper, that a DT is needed to achieve a fully digital spare parts supply chain, in which parts are stored digitally and produced only when needed, close to the point of use.

Synthesis

The use of AdM for spare parts is well understood at the strategic and tactical levels. The reviewed studies show which parts are suitable for printing, under which demand, cost, and uncertainty conditions on-demand production pays off, how supply chains with printers can be configured,

how digital spare parts can be created by redesign or reverse engineering, and how long their production realistically takes once post-processing is included. Three aspects, however, remain open. First, the production of a part is triggered by a demand, an order, or a stocking policy, and decided by planners or procurement teams; only one study uses condition information, as a recommendation to a human, and none triggers the production of a replacement part automatically from a prognostic estimate. Second, the time needed to print a part is treated as a lead time or estimated from its volume, and it is never used, together with a prediction of the remaining life, to decide when printing should start. Third, the printers are treated as separate production resources or facilities, connected to maintenance through planning and information systems, and, apart from repair scenarios and a position paper, DTs have not been used to connect the condition of the equipment to the production of its spare parts. For this dissertation, these observations suggest bringing the printer into the maintenance loop, so that the production of the part is triggered by the prognostic decision and timed according to its build time, measured in advance.

3.6 Integrated Approaches and Synthesis

The previous sections reviewed each link of the maintenance loop considered in this dissertation separately. This section first reviews the studies that combine several of these links, or that claim some form of autonomous maintenance in the sense adopted in Section 2.2.2 (Section 3.6.1), then synthesizes the findings of the whole chapter (Section 3.6.2), and finally derives from them the research gaps and research questions of this dissertation (Section 3.6.3).

3.6.1 Autonomous Maintenance and Integrated Approaches

Studies that describe machines or production systems as self-maintaining mostly implement self-diagnosis and self-prediction, followed by a recommendation. Kangju *et al.* [243] propose a self-maintenance strategy for Computer Numerical Control (CNC) machine tools that combines online monitoring, fault tree analysis, and case-based reasoning to issue early warnings and troubleshooting advice, applied industrially, while the physical maintenance remains with maintenance personnel. Similarly, a DT of a CNC machine tool designed for self-sensing, self-prediction, and self-maintenance estimates the remaining life of a ball screw and raises alarms accompanied by suggested solutions, in simulation [244]; a virtual model of a screw turbine computes the remaining lifetime of its components from their nominal life and operating time and sends automated alerts to technicians through MQTT [245]; and the self-analysis of degradation proposed by Szpytko and Duarte [246] for overhead cranes provides risk profiles on which the asset manager decides. At a larger scale, the PREMIuM platform supports self-healing manufacturing through a distributed edge–cloud architecture that detects early failure indicators and issues maintenance recommendations, validated on offshore drilling equipment [247]. Automatic corrective action is found only in a simulated, domain-independent framework in which agents guided by a DT select and apply repair plans to cyber-physical infrastructures [248].

The combination of DTs with AdM is found mainly in two settings, neither of which concerns the supply of replacement parts. In the first, the DT represents the additive process or machine itself. Rodriguez and Alvares [249] propose a digital ecosystem for AdM aligned with ISO 23247,

in which a digital thread connects design, process planning, execution, and inspection DTs, with OPC UA, MQTT, and MTConnect interfaces, predictive analytics on robot telemetry, and closed-loop adaptive control of the process, implemented on a material extrusion printer and a robotic deposition cell in the laboratory. Alshaheen *et al.* [250] connect a material extrusion printer to an MQTT broker to monitor it remotely through a DT that replicates its movements, while print jobs are still sent manually and predictive maintenance is left for future work. In the second setting, the DT supports the additive repair of a component: the DT of a turbine blade drives the path planning of laser deposition welding in a proposed autonomous repair process chain [251], the DT of a railway wheelset supports the automated control of its laser cladding restoration [240], and, as discussed in Section 3.5.3, DTs link inspection to repair by directed energy deposition [238] or robotic printing [239]. The vision of a DT connecting the condition of equipment to the on-demand production of its spare parts has so far been formulated only as a position paper [242]. No study was found that combines a DT, maintenance, IEC 61499, and AdM: as reported in Section 3.1, the query combining the four concepts returned no records in any of the databases searched.

3.6.2 Synthesis

Table 3.8 summarizes, for each aspect of the maintenance loop considered in this dissertation, what the reviewed literature has established and what remains open, and relates each open issue to the research gaps formulated in Section 3.6.3.

Taken together, the reviewed literature has developed each element of an autonomous maintenance loop in considerable depth, but largely in isolation. Prognostic estimates are routinely turned into decisions on when to maintain and when to order a part, but the part is always obtained from an external supplier or a stock, and the decision is delivered as a recommendation or a work order. DTs host these decisions and connect them to maintenance and enterprise systems, but their virtual-to-physical connection ends at a work order, a schedule, or a control parameter. Distributed control architectures based on IEC 61499 can detect faults, reconfigure themselves, and host analytics at the edge, but they respond to faults that have already occurred by reconfiguring software, rather than to predicted failures by triggering maintenance actions. AdM makes on-demand production of spare parts technically and, for suitable parts, economically viable, but its production is triggered by demand and decided by planners, outside the control loop of the production system. Across all these streams, authority over physical maintenance remains with humans, and automation, where it exists, is bounded by rules and human oversight.

3.6.3 Research Gaps and Research Questions

The synthesis above reveals three research gaps, each corresponding to a link of the maintenance loop that the reviewed literature has not yet closed.

- **Gap G1: from prediction to the physical provision of the replacement part** — Prognostic decisions end in recommendations, work orders, or purchase orders, and the time needed to act is always the lead time of an external procurement process (Section 3.2). Additive manufacturing makes the on-demand production of spare parts viable, but its production is

Table 3.7 – Comparison of representative studies on additive manufacturing for spare parts

Study	Focus	Trigger of production	Stored part	Production time	Validation
[216]	AdM suitability of a parts catalog	Inventory policy	Model assumed	Same lead time as supplier	Industrial data
[217]	Part selection	–	CAD drawings (partly missing)	–	Company case
[220]	AdM vs. conventional under failure-rate uncertainty	Demand	–	Procurement lead time	Parametric analysis
[225]	AdM as dual sourcing	Sourcing and maintenance policy	–	Pre- and post-processing	Aviation case
[226]	On-demand supply performance	Demand	STL library assumed	Estimated from volume	Simulation
[227]	In-house production in advance	Maintenance demand distribution	–	About 12 h incl. post-processing	Numerical, field data
[229]	Distributed production	Central factory	STL via PLM	–	Demonstrator
[231]	Decentralized digital spares	Deployed team	Machine-ready build file	28 h build (proofing)	Four redesigns
[228]	In-house polymer spare parts	Suggestions from staff	–	–	Industrial case (FDM, PLA)
[237]	Repair option selection	Health monitoring, human approval	–	Downtime model	Aerospace case
[238]	DT from inspection to repair	Detected defect	STL, sliced	–	Industrial case
[242]	DT for digital spare parts supply	On demand	Digital parts	–	Position paper
This dissertation	On-demand replacement in maintenance loop	Health indicator threshold, automatic	Sliced machine file (G-code)	Slicer estimate	Laboratory testbed

Table 3.8 – Synthesis of the literature review: state of the art and open issues

Aspect	Established in the literature	Open issue	Gap
From prognosis to decision (3.2)	Joint maintenance and ordering decisions; prescriptive frameworks selecting among actions	Decisions end in recommendations, work orders, or purchase orders	G1
Time to act (3.2.2, 3.5.2)	Lead time encoded in thresholds and decision windows; realistic AdM production times	Lead time is always external procurement; build time never combined with the RUL to time the action	G1
Uncertainty (3.2.2)	RUL distributions, conservative estimates, robust formulations, margins	– (point estimate with margin used here; see Chapter 7)	–
Twin scope and virtual-to-physical flow (3.3)	Twins hosting prognostics and decisions; integration with CMMS, ERP, MES; automated control actions	Mostly single-asset DTs; no DT triggers production of the part it predicts will be needed	G2
Integration architecture (3.2.3, 3.3.2)	Event-driven platforms, enterprise systems, cloud and edge services	Decisions reach execution through centralized IT, not through the control architecture	G2, G3
Distributed control (3.4)	IEC 61499 fault handling, self-healing, reconfiguration; analytics in function blocks; agents coordinating maintenance and production	Triggers are detected faults; responses are reconfigurations or schedules; prognostics moved out of the control layer	G3
Additive spare parts (3.5)	Part selection, sourcing decisions, digital inventories, decentralized production	Production triggered by demand and planners; not linked to prognostics; printers outside the control loop	G1
Autonomy (3.2.3, 3.3.3, 3.4.3, 3.6.1)	Recommendations dominate; automated actions bounded by rules, safety checks, or humans; self-maintenance as self-diagnosis	Automation of the physical maintenance step, even partially, is not addressed	G1–G3
Integration of all elements (3.6.1)	Pairwise combinations only	No study combines prognostics-driven decisions, a DT, distributed control, and AdM	G1–G3

triggered by demand and decided by planners, and the build time of a part has never been combined with a prediction of the remaining life to decide when to act (Section 3.5).

- **Gap G2: from the DT's prescription to a decentralized manufacturing command** — DTs host prognostic and prescriptive logic and are integrated with maintenance and enterprise systems, but they mostly represent a single asset, and their virtual-to-physical connection ends at a work order, a schedule, or a control parameter, reached through centralized information systems. No reviewed DT triggers the production of the part it predicts will be needed (Section 3.3).
- **Gap G3: from distributed fault handling to distributed maintenance workflows** — IEC 61499 architectures detect faults, reconfigure control applications, and host diagnostic analytics at the edge, but respond to detected faults by reconfiguring software, while prognostic tasks are moved out of the control layer and maintenance decisions are coordinated by platforms outside it (Section 3.4).

Underlying these three gaps is a common problem: no reviewed study integrates a prognostics-driven decision, a DT, distributed control, and AdM (Section 3.6.1), and, across all streams, the physical maintenance step remains outside any automated loop. Closing this loop requires converting an inherently uncertain RUL estimate into a correctly timed action, triggered early enough for the replacement part to be produced before the predicted failure, and coordinated across distributed resources, while keeping explicit the steps that remain with humans, such as the installation of the part. From these gaps, the following research questions are formulated:

- **RQ#1** (G1) — To what extent can on-demand AdM autonomously bridge the gap between PdM and physical maintenance execution?
- **RQ#2** (G2) — To what extent do DT technologies ease the integration of PdM, PsM logic and the autonomous triggering of decentralized manufacturing commands?
- **RQ#3** (G3) — To what extent can a distributed IEC 61499 architecture support AM workflows?

The following chapters address these questions by proposing a system that closes this loop (Chapter 4), implementing it on a laboratory testbed (Chapter 5), and validating it (Chapter 6), before discussing the answers to each question in Chapter 7.

Chapter 4

Methodology

The review in Chapter 3 concluded that the elements of an autonomous maintenance loop have been developed in depth but largely in isolation, and that the step from a prognostic estimate to the physical provision of the replacement part remains open (Section 3.6.3). This chapter presents the methodology followed to close that loop: a CPPS in which PsM logic, hosted by a DT, triggers the on-demand production of replacement parts by AdM through a distributed, IEC 61499-compliant control environment. The design follows the specific objectives established in Section 1.3.

Section 4.1 presents the overall system architecture and the DT it realizes (**O1**). Section 4.2 describes the prognostic and prescriptive logic hosted by the DT, and Section 4.3 details the distributed control implementation (**O2**). Section 4.4 describes the integration of AdM as an on-demand production node (**O3**). Finally, Section 4.5 characterizes the degree of automation of the resulting system and the role left to humans, and relates the design to the conclusions of the literature review.

4.1 Proposed System Architecture

The proposed solution is a CPS and, as such, is composed of two layers: a physical layer, formed by the machines on the shop floor, and a cyber layer, formed by the digital and computational space that monitors and acts upon them (Section 2.5.1). The architecture of the solution is depicted in Figure 4.1.

The physical layer comprises the industrial equipment to be maintained, the sensors that monitor it, and the AdM device that produces replacement parts. The cyber layer comprises the DT, which ingests the data collected by the sensors, estimates the health of the monitored components, decides, through PsM logic, whether a replacement part is needed, and sends the corresponding production command to the AdM device.

Following the definition adopted in Section 2.5.6, the DT is scoped at the level of the part of the production system that is relevant to maintenance, rather than of a single asset: it represents the monitored equipment together with the AdM node that acts upon its maintenance decisions. Architecturally, this twin is realized as multiple instances, one per monitored asset, each hosting its own prognostic and prescriptive logic, which share the AdM resources, as shown in Figure 4.2. The decision is therefore not concentrated in a single node, in line with the decentralized control architectures discussed in Section 2.6.1. This distribution was preferred to a single, central twin

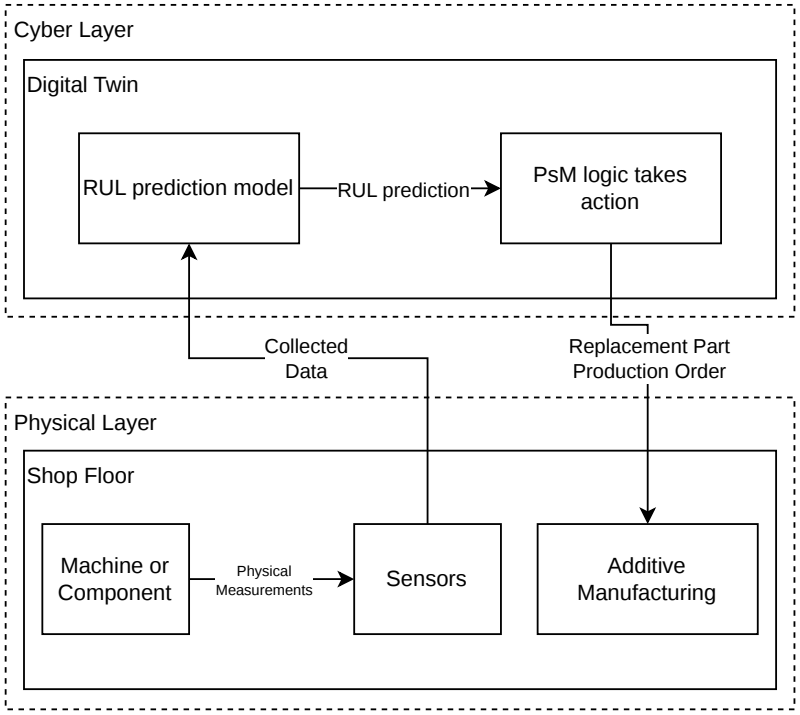


Figure 4.1 – Proposed system architecture

for three reasons. First, it scales with the number of assets: monitoring a new asset amounts to deploying a new instance, without modifying the existing ones. Second, it isolates failures: the failure of one instance affects the monitoring of its own asset, but not the others. Third, it keeps the high-rate sensor data local: the raw sensor stream of each asset travels only between the devices that sample and process it, and only the resulting health estimate and, when needed, the print order are exchanged beyond them. The system-level twin is thus composed of asset-level instances, an arrangement that echoes the hierarchical twins reviewed in Section 3.3.1. The price of this arrangement is that the instances do not coordinate their decisions directly, while sharing a common AdM resource; how concurrent print orders are handled is described in Section 4.4.

In terms of the data flows that characterize a DT (Section 2.5.3), the P2V connection is fully automated: sensor data are streamed to the twin, which updates its estimate of the health of each monitored component, and of the RUL derived from it, once per second. The V2P connection combines automated and human steps: the current health estimate is exposed as an output of the twin, which the user can supervise from the 4diac IDE through OPC UA (Section 4.3), the print order is dispatched automatically to the AdM node when a replacement part is needed, and the installation of the printed part is carried out by a human operator. In the prototype developed, the prognostic model is fixed and only fed new inputs, so the twin updates its state but not its model, a limitation discussed in Chapter 7.

4.2 Digital Twin and Prescriptive Logic

Within each DT instance, sensor data are processed and fed to a trained prognostic model, which outputs a HI of the monitored component. Following the prognostic pipeline described in Section 2.3.1, the HI describes the health of the component rather than a time, and it is from its

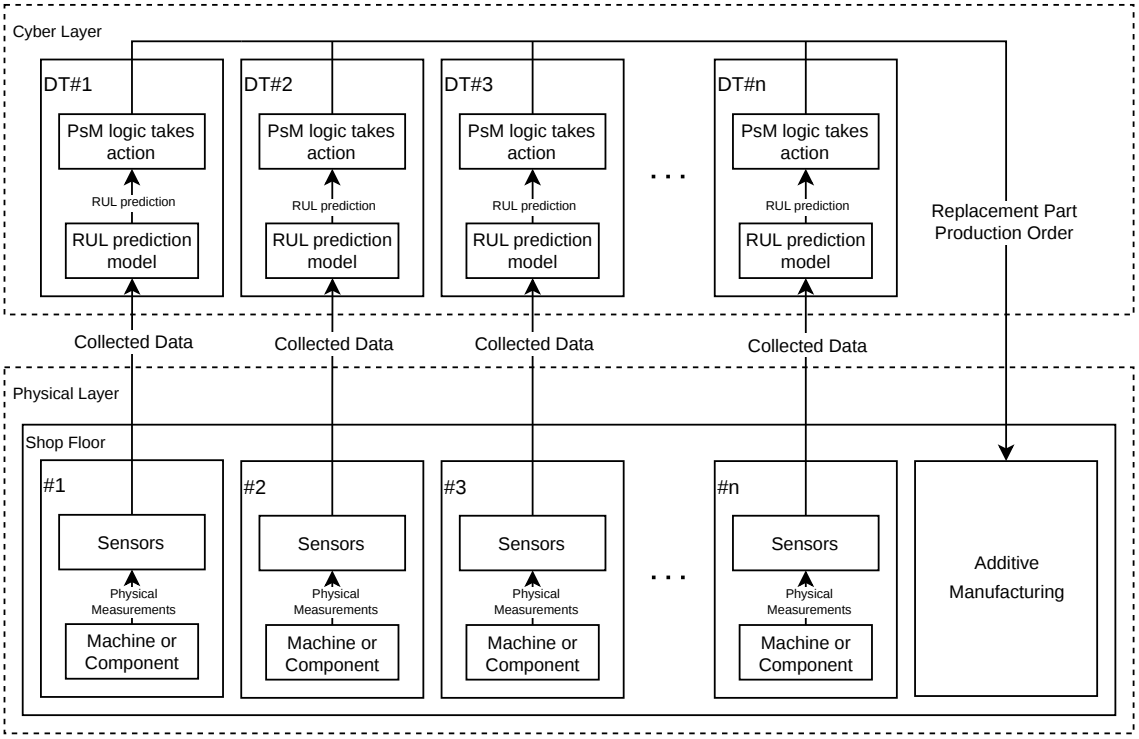


Figure 4.2 – Multiple and independent DT instances sharing AdM resources

evolution that the time remaining until the end of life is inferred (Section 2.3.2). Since the HI labels used to train the model follow a known function of time between the FPT and the end of life, each HI value corresponds to a RUL, so that the estimate can be read in either form. The prognostic model, its training, and this correspondence are described in Chapter 5.

The PsM logic evaluates each new estimate against a simple premise: the replacement part must be available, and installed, before the monitored component fails (Section 2.3.5). The time needed to act is therefore the time needed to produce the part plus the time needed to remove it from the printer and install it on the equipment. Since the part is stored as a sliced file, its production is repeatable, and its duration can be measured beforehand in trial prints; the estimate provided by the slicer alone was found to underestimate it considerably (Section 5.2), so the measured duration is used. This premise is implemented as a threshold on the RUL derived from the estimated HI: when the estimated RUL falls below the threshold, the logic issues a print order to the AdM node.

The threshold is therefore defined in time. A target RUL is set as the time needed to act plus a safety margin, and the estimated RUL, obtained from the estimated HI under the definition of the labels, is compared with it. The margin accounts for the error of the prognostic model where it matters, which is not the error over the whole life cycle, but the risk of the estimate being optimistic, that is, of the estimated RUL exceeding the real one, near the moment of the decision: an optimistic estimate delays the print order, while a pessimistic one only anticipates it. The margin is therefore set as an upper quantile of the difference between the estimated and the real RUL, observed on test data that the model had not seen, in the region of the life cycle in which the decision is taken, with a minimum value that applies even when the model is, on average, pessimistic in that region. The calibration of the thresholds of the prototype is described

in Section 5.6. This rule still assumes that, in operation, the component degrades at a pace similar to the one observed in the training data, an assumption discussed in Chapter 7.

A fixed, deterministic threshold, calibrated in this way, was preferred to the adaptive or probabilistic decision rules reviewed in Section 3.2.2 for two reasons. First, the prognostic model outputs a point estimate of the HI, without a quantification of its uncertainty, so decision rules based on distributions or confidence bounds of the estimate were not available; the margin derived from the observed error is the simplest substitute, similar to the margin based on the error of the predictor used by Chen *et al.* [106] (Section 3.2.2). Second, a deterministic rule is interpretable and keeps the authority to trigger a physical action in explicit, verifiable logic, which is the pattern followed by recent works that allow learned models to inform actions, as discussed in Sections 3.4.3 and 3.3.3. Thresholds based on a quantified uncertainty of each estimate are left as future work.

In terms of the definition adopted in Section 2.4.2, this logic is a deliberately simple form of PsM: it combines the prediction of the component's failure with knowledge of the available maintenance action, the on-demand production of its replacement, to decide whether and when to act, and it delivers the prescription as an automatically executed action rather than as a recommendation. Among the families of prescriptive methods, it corresponds to a logic-based rule (Section 2.4.1). More elaborate prescriptions, weighing several alternative actions or quantifying the uncertainty of the estimate, are outside the scope of this dissertation, whose focus is the integration of the complete loop rather than the sophistication of the prescriptive algorithm. The flowchart in Figure 4.3 shows this pipeline.

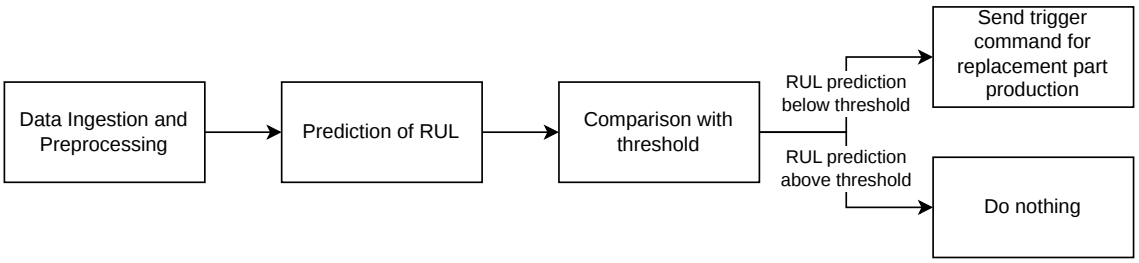


Figure 4.3 – Logic pipeline inside the DT

4.3 Distributed Control Implementation

The proposed solution is implemented using distributed control compliant with the IEC 61499 standard. Together with the cyber-physical architecture discussed so far and the manufacturing context of the framework, this makes the system a CPPS (Section 2.5.1). Implementing it requires the choice of an RTE and of the hardware hosting it.

Although the IEC 61499 standard has been implemented by numerous RTEs since its introduction, a comparison of nine established implementations [252] (including 4diac FORTE, FBDK, FBBeam, Archimedes, ISaGRAF, nxtIECRT, ICARUFB, Fuber, and RTFM-RT) shows that none natively supports FBs written in Python. Since the prognostic models of Section 4.2 were to be developed in Python, native support for Python FBs was adopted as a selection criterion. Among

the implementations that do not meet it, further practical constraints apply: ISaGRAF and nxtIECRT are closed-source commercial solutions, unsuitable for an academic prototype, the latter being itself built upon 4diac FORTE [252], while Fuber and RTFM-RT are research vehicles oriented towards the study of execution semantics rather than deployment, and ICARUFB targets embedded contexts of a considerably smaller scale than the one required here. The comparison was therefore narrowed to the only RTE offering native Python FB execution, Dynamic INtelligent Architecture for Software and MOdular REconfiguration (DINASORE) [253], and the RTE it is most directly positioned against in the literature and with which it shares its development tooling, 4diac FORTE. It should be noted that this criterion favors DINASORE by construction: it reflects the convenience of developing the prognostic models and deploying them in the same language, rather than a requirement of the IEC 61499 standard, and FORTE could also host ML models through the alternatives discussed below.

Table 4.1 summarizes the comparison between the two RTEs. Both implement the event-driven execution model of the IEC 61499 standard and integrate with the 4diac IDE, ensuring a consistent development environment across the two options. DINASORE implements FBs in Python, granting direct access to libraries such as TensorFlow, PyTorch, and Keras, and embeds an OPC UA server automatically in every FB, without requiring a dedicated SIFB [253]. FORTE, in contrast, is a small, portable C++ implementation targeting embedded control devices, which executes basic, composite, and service interface FBs [80] with hard real-time performance [252], restricts FB algorithms to compiled C/C++ code, and has been repeatedly deployed on low-cost platforms such as Raspberry Pi and BeagleBone Black boards [254], [255]. Given that the prognostic models were developed in Python, and that the priority of this prototype is the integration of the complete maintenance loop, DINASORE was selected as the RTE for the proposed architecture.

Table 4.1 – Comparison between 4diac FORTE and DINASORE

Criterion	FORTE [82]	DINASORE [253]
FB implementation language	C++ (compiled)	Python (interpreted)
Real-time guarantees	Hard real-time	Not designed for hard real-time
Target hardware	Small embedded control devices	Devices able to run a Python interpreter
ML library access	None natively; requires C/C++ inference code or an external service	TensorFlow, PyTorch, Keras, within the supported Python version
OPC UA integration	Requires dedicated SIFB	Embedded automatically per FB
Maturity and deployment precedent	Reference runtime of Eclipse 4diac; extensive low-cost hardware precedent	Research runtime; limited deployment precedent

FORTE therefore remains the better choice in several scenarios. For control loops with strict timing requirements, such as motion or safety-related control, its compiled, deterministic execution provides guarantees that an interpreted Python runtime does not [252], [253]. For devices too constrained to run a Python interpreter, such as microcontroller-based controllers, it is the only one of the two options that can be deployed. And for industrial applications, its maturity, its role as the reference runtime of Eclipse 4diac, and its record of deployments on low-cost hardware reduce the risk of adopting it, whereas DINASORE remains a research runtime with a considerably smaller body of reported deployments. Nor is the absence of native Python support an insurmountable obstacle: a trained model could be exported to a C/C++ inference library and called from a FORTE

FB, or served by an external process communicating with the IEC 61499 application, as in the approaches reviewed in Section 3.4.2, at the cost of additional engineering effort and of splitting the prognostic pipeline across two execution environments.

The choice of DINASORE has implications for this dissertation. First, since every node of the prototype, including those that command the servo motors of the monitored equipment, runs DINASORE, the prototype provides no real-time guarantees: the evaluation in Chapter 6 measures the latencies and jitter observed in operation, but cannot establish bounds on them. This is acceptable for the tasks at the core of this work, namely data ingestion, health estimation, and threshold evaluation, which are decision-support processes rather than millisecond-sensitive control loops, but an industrial deployment would more likely combine both runtimes, assigning time-critical control to FORTE and prognostics to DINASORE. Since both implement IEC 61499 and share the same development environment, such a combination is possible in principle, although the differences in execution semantics between runtimes, discussed in Section 2.6.2, mean that the behavior of an application is not guaranteed to be identical across them. Second, the prognostic models become dependent on the Python environment supported by the runtime, including its interpreter version and the libraries available for it. Third, the results obtained with DINASORE rely on a runtime with limited deployment precedent, which constrains their generalization to industrial settings. These implications are revisited in Chapter 7.

The event-driven execution model of IEC 61499 also suits the nature of the prescriptive logic. Unlike the cyclic, scan-based execution of IEC 61131-3 programs, IEC 61499 FBs execute only in response to incoming events (Section 2.6.2), which matches the threshold-triggered logic of Section 4.2: a print order is an event, emitted once, when the threshold is crossed. In contrast with the IEC 61499 applications reviewed in Section 3.4, which mostly respond to detected faults by reconfiguring the control software, and in which prognostic tasks tend to be moved out of the control layer, the prognostic model and the prescriptive decision are here executed on the edge nodes of the control architecture itself, under the control of its function blocks, and their outcome is a maintenance action carried out through it. How the model is executed alongside the runtime is described in Section 5.5.

Communication between the distributed nodes relies on two complementary protocols, whose roles follow from the characteristics discussed in Section 2.7:

- **MQTT** — Transmits operational messages, such as sensor data, health estimates, and print orders, owing to its lightweight publish/subscribe model, well suited to the event-driven nature of the system. This choice is further supported by an independent evaluation of MQTT and HTTP [90] on a Raspberry Pi running FORTE, which found MQTT to offer lower latency, smaller packets, and lower bandwidth consumption [256]. Rather than relying on a single central broker, each node that needs to receive messages hosts its own broker, running on the same device, to which the publishing nodes connect, so that the failure of one broker only affects the messages addressed to that node. Messages are published with the default QoS level of the client library, QoS 0, under which a message is delivered at most once (Section 2.7.3). On the network of the prototype this proved sufficient, but a deployment in which print orders could be lost would require a higher QoS level, as discussed in Chapter 7.

- **OPC UA** — Used for supervisory communication, to monitor the state of the distributed runtime nodes from the 4diac IDE, since it is automatically embedded in every DINASORE FB [253].

A single protocol could, in principle, have served both purposes, since OPC UA also supports a publish/subscribe model (Section 2.7.2). Two protocols were nonetheless used because their strengths are complementary: MQTT adds less overhead than OPC UA, both when establishing connections and when sending data (Section 2.7.4), and its decoupling of publishers and subscribers suits the high-rate sensor streams and the one-off print orders exchanged between nodes, whereas the information model of OPC UA suits the supervision of the nodes' state, and is provided by DINASORE without additional development.

Figure 4.4 depicts the distributed implementation in detail. Every node in the architecture, including those interfacing directly with sensors and actuators, is hosted on a Raspberry Pi running DINASORE, ensuring a homogeneous hardware and software base across the system, consistent with prior work deploying IEC 61499-compliant control on similar low-cost, single-board platforms [254]–[256]. By distributing the control logic across multiple independent nodes, rather than relying on a central controller or a central broker, the implementation avoids the single point of failure of a central coordinator inherent to centralized architectures (Sections 2.6.1 and 2.7.3), although the shared AdM node, as a single manufacturing resource, remains one for the production of the parts. Within each asset, the functions of control, sensor sampling, and prognostic inference are, moreover, assigned to separate devices, each corresponding to one IEC 61499 device. This separation keeps the high-rate sampling of the sensors, which relies on dedicated sampling utilities, from competing for processing resources with the inference of the prognostic model or with the control of the equipment, and ensures that an overload or failure in one function does not stall the others. It also mirrors the separation usually found in industry between field devices, controllers, and edge computing resources. The cost of this choice is a larger number of devices to deploy and maintain, and a greater dependence of the system on the network that connects them.

4.4 Additive Manufacturing Integration

The distributed control implementation described in Section 4.3 transports the command issued by the prescriptive logic of Section 4.2. This section describes what happens once that command reaches its destination. When the estimated RUL falls below the defined threshold, the resulting print order is published over MQTT by the same FB pipeline responsible for the prognostic and prescriptive logic, and received, in a single hop, by the AdM node.

The AdM node is hosted on a Raspberry Pi running the OctoPi [257] image, a Raspberry Pi Operating System (OS) distribution bundling the OctoPrint [258] 3D printer management server, alongside a DINASORE instance. The incoming MQTT event triggers a dedicated FB within this instance, which issues a REST API call to OctoPrint to start printing the replacement part. This architecture is depicted in Figure 4.5.

The replacement parts are stored on the AdM node as sliced machine instructions (G-code), prepared in advance for the printer, material, and parameters used. In terms of the digital process chain described in Section 2.8.2, the digital inventory is therefore kept at the machine-ready stage:

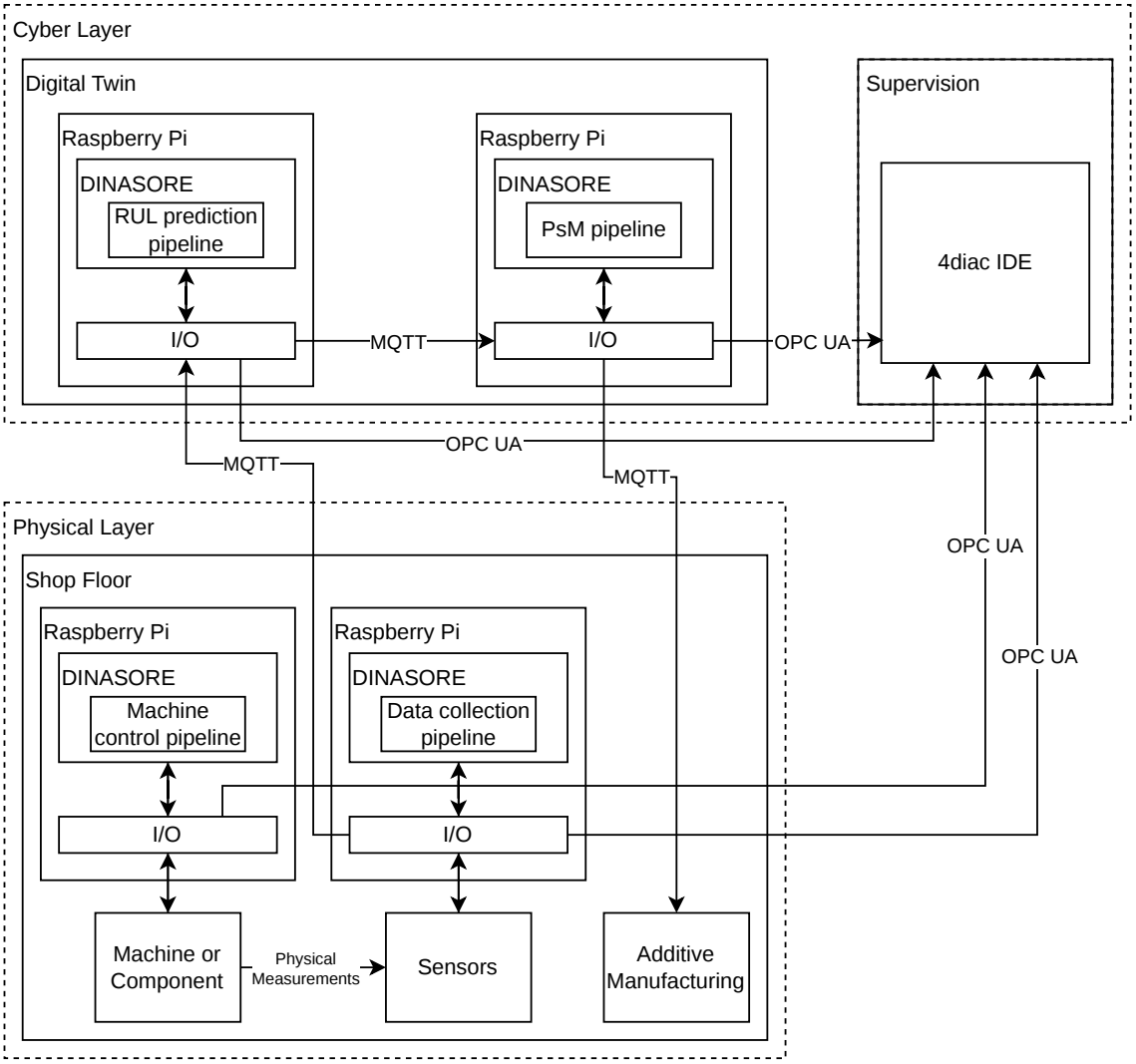


Figure 4.4 – Distributed control implementation of the proposed system architecture

no slicing or setup decisions remain to be taken when a print order arrives, and, since the same file is printed every time, the build time of each part can be measured in advance, which is what allows the lead time of the replacement to be taken into account by the prescriptive logic, although it may still vary from one print to the next (Section 7.4). The price of this choice is that each stored file is specific to the printer and material for which it was prepared, and that the design and slicing of the parts are carried out beforehand by a human. The print command specifies the location of the file, either the local storage of OctoPrint or the printer's Secure Digital (SD) card, and its name. In the prototype, in which all the monitored components are identical, a single file serves all DT instances. A second FB on the AdM node monitors the status of the print, so that both the response to the print order and the progress of the print can be supervised from the 4diac IDE. Since the AdM node is shared by all the DT instances, incoming print orders pass through a queuing FB before reaching the one that starts the print. If the printer is idle, the order is forwarded immediately; if a print job is already under way, or the printer is not operational, the order is held until the operator releases it through an input event. Concurrent print orders are therefore serialized, and their release in such cases is left to a human rather than coordinated between instances.

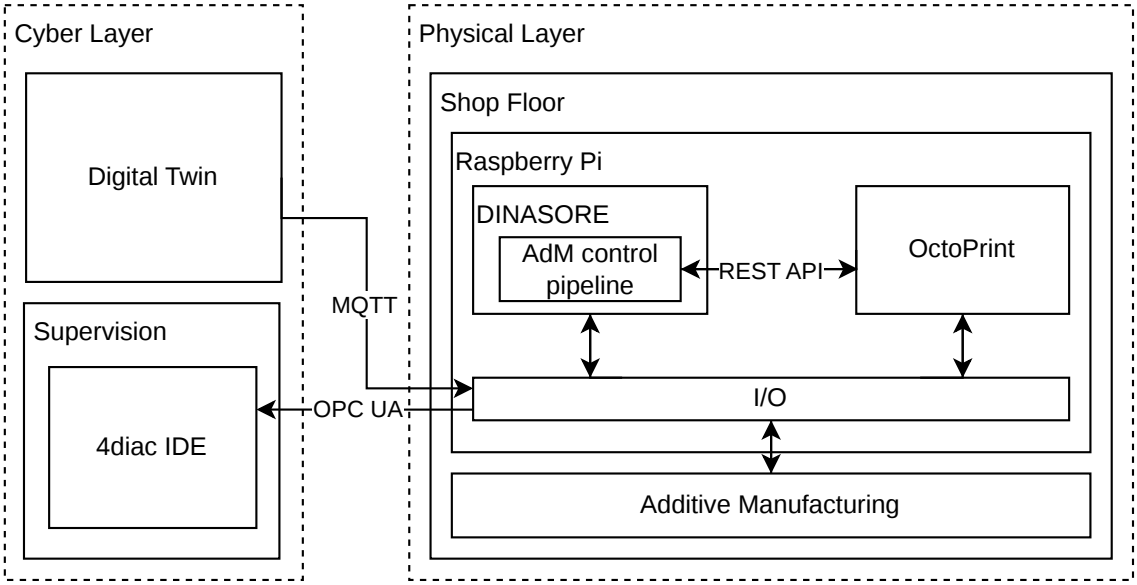


Figure 4.5 – AdM integration in the proposed system architecture as a distributed node

This mechanism realizes the digital spare parts concept described in Section 2.8.3: replacement parts are not held in stock, but stored digitally and produced only once a specific, predicted need has been identified. Unlike the approaches reviewed in Section 3.5, in which the production of spare parts is triggered by a demand and decided by planners, here it is triggered directly by the prognostic decision, within the control loop of the production system.

4.5 Degree of Automation and Role of the Human

Following the framework discussed in Section 2.4.3, the degree of automation of the proposed system is characterized stage by stage rather than by a single label, as summarized in Table 4.2.

Table 4.2 – Degree of automation of the proposed system per class of function (after [53])

Class of function	**Implementation**	**Performed by**
Information acquisition	Vibration sampling and streaming by the sensor nodes	System
Information analysis	Health estimation by the prognostic model, once per second	System
Decision and action selection	Threshold rule issuing the print order, supervisable and cancellable by the operator	System
Action implementation	Production of the replacement part	System (printer); operator releases orders held while the printer is busy or not operational
	Removal of the printed part and installation on the equipment	Human operator
Preparation (offline)	Design and slicing of the parts; preparation of the printer (empty bed, filament loaded, calibration)	Human

Information acquisition, information analysis, and decision and action selection are fully automated, and so is the part of the action implementation that involves producing the replacement part. The decision is executed automatically, and the operator is informed of it only when consulting the supervisory interface, which corresponds to level 8 of the scale in Table 2.2. This holds whenever the printer is available; when it is busy or not operational, the print order is held until

the operator releases it, and the automation of the action falls, in that case, to the level at which the system executes only with human approval. The operator can nevertheless cancel the print order through an event of the corresponding FB, and is therefore on the loop for the decision, in the sense of Section 2.4.3. The operator also retrieves the printed part and installs it on the equipment, and is therefore in the loop for the physical execution of the maintenance action, while taking no part in the decision itself: the operator does not evaluate the health estimate, nor decide whether or when to print, nor select the part to be produced. Outside the loop, the design and slicing of the parts and the preparation of the printer, which must have an empty bed, be loaded with filament, and be calibrated, are also human tasks, performed before operation. In the sense adopted in Section 2.2.2, the system is therefore an AM system in which monitoring, analysis, decision, and the production of the replacement part are carried out by the production system itself, while its installation remains, in the sense of EN 13306, maintenance carried out by an operator. Since the decision follows a pre-programmed rule, the system automates the maintenance decision rather than reasoning about unexpected events, and it does not claim the higher levels of autonomy discussed in Section 2.4.3. Keeping the physical intervention with the operator is also consistent with the human-centric orientation of I5.0 (Section 2.1), in which automation takes over repetitive tasks while humans retain a meaningful role.

Taken together, the design choices described in this chapter respond to the conclusions of the literature review (Section 3.6.3). First, the replacement part is produced within the system, on demand, triggered by the prognostic decision and timed according to the measured time needed to produce and install it, with a margin for the error of the estimate, instead of being ordered from an external supplier after a recommendation. Second, the DT encompasses the AdM node, so that its virtual-to-physical connection reaches a decentralized manufacturing resource, rather than ending at a work order or a control parameter. Third, the prognostic inference and the prescriptive decision run on the edge nodes of an IEC 61499 architecture, which carries the resulting action through to the printer, rather than responding to detected faults by reconfiguring software. The implementation of this architecture on a laboratory testbed is described in Chapter 5.

Chapter 5

Implementation

This chapter describes the physical implementation of the system proposed in Chapter 4, in direct pursuit of the deployment aspect of objective **O4** (Section 1.3). A laboratory-scale testbed was built to implement the complete maintenance loop, from the acquisition of sensor data to the production of the replacement part. Section 5.1 describes the testbed, and Section 5.2 the control of the production line and of the AdM node. Sections 5.3 and 5.4 describe the collection of run-to-failure data and the development of the prognostic models, and Section 5.5 their adaptation to online health estimation. Section 5.6 describes the integration of the complete system, and Section 5.7 summarizes the limitations of the implementation. The screenshots of the FB applications developed in the 4diac IDE are provided in Appendix B, and the code, models, and scripts described in this chapter are available in the repository of this work (Section 1.4).

5.1 Testbed

The testbed, shown in Figure 5.1, uses equipment already available in the laboratory from previous projects: three EEZYbotARM Mk2 [259] robotic arms, that had their gripper modified for the one designed by sthone [260], an Ender-3 S1 Pro 3D printer, and several Raspberry Pi 3 Model B V1.2 boards. The arms were mounted on a wooden base to form a pseudo-production line that moves a single piece back and forth in sequence, constituting the industrial process of the CPPS, while the printer constitutes its AdM node. The arms, including their grippers, are themselves largely made of 3D-printed PLA parts, which is what makes the printer able to produce their replacement parts.

The architecture of Figure 4.1, adapted to this testbed, is shown in Figure 5.2. It is deployed across nine Raspberry Pi devices and one Raspberry Pi Pico W, whose roles are listed in Table 5.1. Following the separation of functions described in Section 4.3, each monitored arm has a dedicated device for control, one for sensor sampling, and one for prognostic inference and prescriptive decision, referred to hereafter as its PdM node. Arm #2 is the exception: its piezoelectric sensor is sampled by a Pico W connected by a serial link directly to the arm's PdM node, so no separate sampling device is needed, for the reasons explained in Section 5.3. All the Raspberry Pi devices run their own DINASORE instance and are identified by static Internet Protocol (IP) addresses. They communicate over Wi-Fi, except for the links that carry the high-rate vibration streams: on Arms #1 and #3, the sampling device and the PdM node are connected by an Ethernet cable, and, on Arm #2, the Pico W is connected to the PdM node by a serial Universal Serial Bus (USB) link.

The arms and their devices are installed in one laboratory, connected to its router, which is in turn connected to the campus network, while the printer and the Raspberry Pi of the AdM node are installed in another laboratory, connected over Wi-Fi to the university's eduroam network. Print orders therefore travel from the network of the testbed's laboratory, through the campus network, to the AdM node.

Table 5.1 – Devices of the testbed and their roles

Asset	Device	Role
Arm #1	Raspberry Pi 3 (control)	Drives the arm's servo motors and coordinates with the other arms
	Raspberry Pi 3 (sampling)	Samples the ADXL345 sensor and streams its data
	Raspberry Pi 3 (PdM node)	Estimates the HI and RUL and decides whether to print
Arm #2	Raspberry Pi 3 (control)	Drives the arm's servo motors and coordinates with the other arms
	Raspberry Pi Pico W	Samples the piezoelectric sensor
	Raspberry Pi 3 (PdM node)	Receives the piezoelectric signal over serial; estimates the HI and RUL and decides whether to print
Arm #3	Raspberry Pi 3 (control)	Drives the arm's servo motors and coordinates with the other arms
	Raspberry Pi 3 (sampling)	Samples the MPU9250 sensor and streams its data
	Raspberry Pi 3 (PdM node)	Estimates the HI and RUL and decides whether to print
AdM	Raspberry Pi 3 (AdM node)	Runs OctoPrint and DINASORE; queues, starts, and monitors print jobs

5.2 Control of the Production Line and the Additive Manufacturing Node

This section describes how the two kinds of equipment of the testbed are controlled: the three arms, which form the production line and coordinate their movements among themselves, without a central controller, and the printer, which is driven by the AdM node in response to the print orders issued by the PdM nodes. In both cases, the control is implemented with custom DINASORE FBs, which are listed, together with those of the rest of the system, in Section 5.6.

5.2.1 Production Line

Each arm is driven by servo motors connected to a Waveshare Servo Driver HAT mounted on its control device, a 16-channel, 12-bit Pulse Width Modulation (PWM) driver based on the PCA9685 [261] controller, which is commanded through the Adafruit ServoKit library. A custom `ARM_CONTROL` FB receives, as an input, an eXtensible Markup Language (XML) file describing the positions of the arm's joints and the sequence in which they should be reached, parses it, and executes each movement when triggered. Separating the description of the movements from the FB allows the same block to be reused on the three arms, each with its own file.

A production cycle consists of six movements, which carry the piece from one end of the line to the other and back: Arm #1 moves it from the left position to the center-left, Arm #2 from the center-left to the center-right, and Arm #3 from the center-right to the right, after which the sequence is reversed. The arms coordinate through MQTT, following the topology described in

Section 4.3: when an arm completes a movement, it publishes a message to the broker of the arm that must perform the next one, on a topic that identifies that movement, as listed in Table 5.2. At the ends of the line, the arm that completes a movement publishes to its own broker, triggering its return movement. No device orchestrates the cycle: each arm only knows which movement to perform when a message arrives, and whom to notify when it finishes.

Table 5.2 – Movements of a production cycle and the MQTT topics that trigger them

Step	Arm	Movement	Triggered by topic
1	#1	Left → center-left	`/ARM1_12/RUN`
2	#2	Center-left → center-right	`/ARM2_23/RUN`
3	#3	Center-right → right	`/ARM3_23/RUN`
4	#3	Right → center-right	`/ARM3_32/RUN`
5	#2	Center-right → center-left	`/ARM2_21/RUN`
6	#1	Center-left → left	`/ARM1_21/RUN`

All MQTT communication uses the paho-mqtt client library, through the MQTT FBs of the DINASORE FB repository [262], and each device that needs to receive messages runs its own Mosquitto broker.

5.2.2 Additive Manufacturing Node

The AdM node runs the OctoPi [257] image, which bundles the OctoPrint [258] server for the remote control and monitoring of 3D printers, alongside a DINASORE instance hosting three custom FBs, which interact with the printer through the REST API of OctoPrint:

- `PRINT_QUEUE` receives the print orders arriving from the PdM nodes. If the printer is idle, it forwards the order to `PRINT_START`; if a job is already under way, or the printer is not operational, it holds the order until the operator releases it through an input event, as described in Section 4.4;
- `PRINT_START` starts the print. It receives, as inputs, the location of the file, either the local storage of OctoPrint or the printer's SD card, and its name, and sends a request to the files endpoint of the API that selects the file and starts printing it in a single call. It reports whether the request succeeded, and an input event allows the operator to cancel the print order;
- `PRINT_CHECK` monitors the job, querying the job endpoint of the API once per second and exposing the state of the printer, the progress of the current job, and any error, which `PRINT_QUEUE` uses to decide whether a new order can be forwarded.

The replacement part is stored on the AdM node as a single file of machine instructions (G-code), from which all the pieces of the gripper are printed in one job, as shown in Figure 5.3. The file was sliced in advance for the printer, its 0.6 mm nozzle, and the PLA filament used, so no slicing or setup decisions remain to be taken when an order arrives (Section 4.4). The slicer estimated a build time of about 2 hours and 34 minutes, with the quality setting `FAST` and infill of 20%, whereas the build time measured in a trial print was 3 hours and 26 minutes. This deviation,

of about a third of the estimate (34%), is well above the 5 to 10% reported in the literature for the estimates of AdM machines (Section 2.8.3), so the slicer's estimate was not used: the time needed to act was set from the measured duration (Section 5.6).

5.3 Data Collection

Prior use of these arms in previous projects indicated that the gripper is the component most subject to wear, and therefore the one most in need of maintenance. Since the arms are made of 3D-printed PLA, the loosening and wear of their parts manifest as changes in vibration, which was therefore chosen as the monitored quantity. Monitoring only the gripper also matched the number of available sensors, since estimating the health of every component of the line was not necessary to demonstrate the maintenance loop.

Three different vibration sensors were available, and each gripper was fitted with one of them: an ADXL345 [263] on Arm #1, a piezoelectric vibration sensor [264] on Arm #2, and an MPU9250 [265] on Arm #3. The ADXL345 and MPU9250 are digital accelerometers providing three-axis acceleration, from which vibration can be inferred, while the piezoelectric sensor is an analog transducer that outputs a raw vibration signal. The ADXL345 and MPU9250 were configured through the Serial Peripheral Interface (SPI) bus, as specified in their datasheets, to sample at their maximum rates of 3200 Hz and 4000 Hz, respectively, each connected to its own Raspberry Pi. Their sampling is performed by C-based utilities, chosen over Python for their ability to sustain these rates; the utility used for the ADXL345 is based on the open-source adxl345spi tool [266], and the one used for the MPU9250 was developed for this work. The latter configures the accelerometer for a full scale of ± 16 g, with its digital low-pass filter bypassed so that it outputs data at 4 kHz, and stores the samples in the sensor's internal First In, First Out (FIFO) buffer, which is emptied by burst reads over SPI. To avoid losing samples while the data is written to disk, the samples are accumulated in two alternating buffers of about one second each, and written by a separate thread pinned to a different processor core, so that sampling is never blocked by storage.

Since the Raspberry Pi 3 has no built-in analog-to-digital converter, the piezoelectric sensor was connected to a Raspberry Pi Pico W, whose 12-bit analog-to-digital converter reads the signal in a continuous loop, without pauses, and sends each value over its USB serial port, yielding an effective sampling rate of about 2800 Hz, limited by the throughput of the serial link. Because the memory of the Pico W precludes storing the data onboard, the stream is captured by a Raspberry Pi, which assigns a timestamp to each value on reception, and which during data collection was a separate storage device and, in the deployed system, is the PdM node of Arm #2. Occasionally, two lines of the serial stream are merged into a single value outside the range of the converter; these values are discarded.

To capture the complete life cycle of the grippers within the time available, an accelerated degradation process was adopted, in which faults were deliberately introduced into the grippers: the holes of their screws were progressively loosened, and holes were made in their parts. Accelerated degradation is the usual way of obtaining run-to-failure data in prognostics, as it collects degradation patterns similar to real ones within a practical time frame, although, since the aging conditions are not real, some information about the real degradation process is lost [36]. The arms

were operated in sessions of about one hour, each recorded as one file per sensor; the ninth session was interrupted after 34 minutes. The first sixteen sessions were recorded in normal operation, without any introduced fault. From the seventeenth to the twenty-first session, faults were introduced in five successive stages, one at the start of each session and simultaneously in the three grippers, at logged timestamps. Before that, the notes taken during the collection had recorded a slight wear of the grippers from the tenth session onwards, which, since no fault had yet been introduced, was attributed to their natural wear under continuous operation. At the start of the twenty-second session, the grippers no longer worked; this final session, of 30 minutes, recorded their complete failure. In total, about 21 hours of operation were recorded, of which 20 hours and 34 minutes elapsed until the complete failure of the grippers, taken as their end of life.

Two events during the collection affected the recorded data. First, the piezoelectric sensor was replaced after the fourteenth session. To keep the amplitude of its signal consistent across the life cycle, the recordings of sessions 15 to 22 were multiplied by the ratio between the standard deviation of the signal in the fourteenth session and that in the fifteenth, a factor of 1.917, which is also applied to the live signal of the new sensor in operation (Section 5.5). Second, the ADXL345 was disconnected during part of the collection: no data was recorded in the eighth session, and only part of sessions 6, 7, 9, 10, and 11, so that about four and a half hours of its life cycle are missing.

5.4 Prognostic Model Development

The development of the prognostic models followed the systematic guide proposed by Berghout and Benbouzid [36], and comprised the construction of the health labels, the selection of the model, and its training.

5.4.1 Preprocessing and Feature Extraction

The initial plan was to fuse the information of the three sensors into a single model, which required the three datasets to share a common sampling rate, that of the piezoelectric sensor, 2800 Hz. The fusion plan was later abandoned, as explained in Section 5.4.3, but, since no relevant vibration content was found above 1400 Hz, the Nyquist frequency of the common rate, this rate was kept. The ADXL345 and MPU9250 recordings were resampled with a polyphase filter, with ratios of $7/8$ and $7/10$, respectively, whose anti-aliasing filter prevents the content between 1400 Hz and the original Nyquist frequency from folding into the retained band. The piezoelectric values, whose effective sampling rate is not uniform, were linearly interpolated, from their reception timestamps, onto a uniform grid at 2800 Hz.

A fourth-order high-pass Butterworth filter with a cutoff of 1 Hz was then applied to all the signals, to remove baseline drift and the components introduced by the movement of the arms and by gravity. The filter is applied causally, in second-order sections, which avoid the numerical ill-conditioning of a very low cutoff relative to the sampling rate, and starting from the steady state corresponding to the first sample, which avoids a transient at the start of each recording. This is exactly how the filter is applied to the live stream in operation (Section 5.5), so that the features of training and operation are computed from identical signals.

The signals were then segmented into one-second windows, and eleven features were extracted from each window and each channel: six in the time domain (root mean square, variance, peak, crest factor, skewness, and kurtosis) and five in the frequency domain, computed with a real Fast Fourier Transform (FFT) (the total spectral energy and the energy in four bands of 350 Hz up to the Nyquist frequency of 1400 Hz). This yields 33 features per window for the three-axis ADXL345 and MPU9250, and 11 for the single-channel piezoelectric sensor. Windows with a variance below 10^{-6} were discarded as stationary, which removes the periods in which a sensor recorded no signal; the same criterion is applied in operation.

5.4.2 Health Labels

In the terms of the guide, the collected data was incomplete: no labels existed, so HS and HI labels had to be constructed, as in the case of the PRONOSTIA bearing dataset, which was also obtained through accelerated degradation [36]. To this end, the features of each sensor were standardized and reduced by Principal Component Analysis (PCA), retaining 95 % of the variance, which resulted in 10 principal components for the ADXL345, 12 for the MPU9250, and 5 for the piezoelectric sensor. These components were used only to construct the labels; the prognostic models were trained on the extracted features themselves, as described in Section 5.4.3.

The windows of the three sensors were placed on a common time line, the operating time elapsed since the start of the collection, computed from the real duration of each session. Placing the windows by operating time, rather than by their position in the dataset, prevents the windows missing from the ADXL345 recording, or discarded as stationary, from compressing the time line of a sensor.

The HS was obtained by clustering the principal components with a Gaussian Mixture Model (GMM) into four clusters, corresponding to the four operational states [36]. To adapt the clustering to the physical nature of the degradation process, three measures were taken: outliers were attenuated with a rolling median filter over a 15-minute window, followed by an Exponential Moving Average (EMA); the normalized operating time, weighted by a factor of 15, was added to the clustering input, forcing the GMM to respect chronological distance, so that similar amplitudes recorded at unrelated stages of the life cycle were not assigned to the same state; and the resulting clusters were ordered chronologically and passed through a cumulative maximum, enforcing the monotonicity expected of degradation.

The HI was then derived from the HS. The FPT was defined as the moment at which the GMM marks the transition from normal operation to early degradation, and the end of life as the moment of complete failure of the grippers, at the start of the final session. The FPT occurred after 8 h 08 min, 5 h 22 min, and 3 h 02 min of operation for the ADXL345, MPU9250, and piezoelectric sensors, respectively, that is, at about 40%, 26%, and 15% of the life cycle. From the FPT onwards, the HI was modeled with the exponential degradation function recommended by the guide for accelerated life tests [36], given by Equation 5.1:

$$HI(t) = d - e^{\lambda t + b}, \tag{5.1}$$

where d, λ, and b are derived from the boundary conditions $HI(t_{FPT}) = 1$ and $HI(t_{end}) = 0$. Using the normalized time of Equation 5.2 and setting $b = 0$, these conditions yield $d = 2$ and $\lambda = \ln 2$,

as shown by Equation 5.3. Before the FPT, the HI is equal to 1, and, during the final session, after the end of life, it is equal to 0.

$$t' = \frac{t - t_{FPT}}{t_{end} - t_{FPT}} \in [0,1] \tag{5.2}$$

$$HI(t') = 2 - e^{\ln(2)\cdot t'}, \quad t' \in [0,1] \tag{5.3}$$

Since the labels are a known function of time, each HI value after the FPT corresponds to a RUL. Inverting Equation 5.3 gives Equation 5.4, where $L = t_{end} - t_{FPT}$ is the length of the degradation period, which is 12 h 26 min for the ADXL345, 15 h 11 min for the MPU9250, and 17 h 32 min for the piezoelectric sensor. Before the FPT, the HI is 1 and Equation 5.4 gives L, which is then a lower bound of the RUL, in the manner of a piecewise linear RUL. This conversion is used to express the estimates of the models as a RUL, both when evaluating them (Chapter 6) and when setting the thresholds of the prescriptive logic (Section 5.6).

$$RUL = t_{end} - t = (1 - \log_2(2 - HI)) \cdot L, \quad 0 \leq HI < 1 \tag{5.4}$$

Two consequences of this construction should be noted. First, the exponential shape was justified in the guide for bearings, whose degradation accelerates towards failure [36], and its adoption here is an assumption carried over to the PLA grippers. Second, because of the weight given to the time feature and of the cumulative maximum, the FPT is largely a chronological boundary rather than a change detected in the signal alone, and, after it, the HI is a fixed function of the position of the window in the life cycle. The labels therefore encode elapsed time as much as condition. This is apparent when the FPT is compared with the data collection protocol: the FPT falls between the fourth and the ninth session, well before the first fault was introduced, in the seventeenth. The early degradation it marks may therefore reflect the natural wear of the PLA parts under continuous operation, which the notes taken during the collection recorded from the tenth session onwards (Section 5.3), or be partly an effect of the chronological weighting of the clustering, the more so as the FPT of the three sensors precedes even these observations; the data collected does not allow the two to be fully distinguished. In the case of the ADXL345, the FPT also falls within the part of the life cycle in which its recording is incomplete, so its position is less reliable than that of the other two sensors.

5.4.3 Model Selection and Training

The model was selected following the flowchart of the guide [36], which considers the availability, complexity, and drift of the data, and the complexity of the model. Given the volume and velocity of the data, about 2800 samples per second over more than 20 hours, and the nonlinearity of the degradation of 3D-printed parts, the data was classified as complex, which points towards DL models. The continuous change of the state of health throughout a run-to-failure life cycle introduces drift in the data [36]; however, since the dataset covers the complete life cycle, the states the model encounters in operation are the pre-characterized ones on which it was trained, and an offline learning architecture was considered sufficient, without the additional cost of online adaptation.

The sensor fusion plan was abandoned in favor of three independent models, one per sensor and, therefore, one per arm, each trained on its own sensor's features against its own HI labels. This choice follows from the distribution of the DT described in Section 4.1: with independent models, the PdM node of each arm estimates the health of its own gripper without depending on data from the other arms, or on their synchronization.

Each model combines a one-dimensional convolutional layer with a bidirectional LSTM network, an architecture in which the convolutional layer extracts local features from the input sequence and the bidirectional LSTM encodes its temporal evolution (Section 2.3.3). The input is a sequence of 60 consecutive windows, that is, the last minute of operation, each described by its 33 or 11 features. It passes through a convolutional layer with the same padding, a ReLU activation, and a max-pooling layer that halves the length of the sequence, followed by the bidirectional LSTM, whose output at the last time step goes through dropout and a linear layer that produces the HI of the last window of the sequence. Since the whole sequence lies in the past, the bidirectional processing within it does not use any future information in operation.

The data of each sensor, ordered by operating time, was divided into consecutive blocks of 60 windows, which were randomly assigned, with a fixed seed, to training (80 %) and test (20 %) sets. A sequence is only formed by contiguous windows, that is, consecutive windows of the same session with no window discarded between them. The training sequences are all the sequences of 60 contiguous windows lying within training blocks, including those that span two adjacent training blocks. Each test block yields a single sequence, formed by its own 60 windows, provided they are contiguous, so that no test sequence contains a window of the training set; test blocks interrupted by a gap were excluded. In addition, 20 % of the training blocks were set aside as a validation set for the optimization of the hyperparameters, so that the test set was used neither in the optimization nor in the training. The resulting numbers of windows and sequences are given in Table 5.3.

Table 5.3 – Windows, blocks, and sequences of each sensor

	ADXL345	**MPU9250**	**Piezo**
Windows	58 871	74 548	75 291
Training / test blocks	784 / 197	993 / 249	1003 / 251
Training sequences	36 896	46 487	47 305
of which used in the optimization	24 110	29 997	29 714
Validation sequences (optimization)	1689	1783	1902
Test sequences	192	246	248

The hyperparameters were optimized separately for each sensor with Optuna [267], using its default tree-structured Parzen estimator sampler with a fixed seed, over 30 trials of 15 epochs each, without pruning, minimizing the MSE on the validation set, with the features standardized on the remaining training blocks. The length of the sequence was fixed at 60 windows, the value selected for the three sensors in a preliminary search over 30, 60, and 90 windows, since the size of the blocks of the split depends on it. The search space and the best values found are given in Table 5.4. The final models were then trained on all the training sequences for 50 epochs with the AdamW optimizer [268], an MSE loss, and a batch size of 256, without early stopping or learning rate scheduling, with the random seeds of Python, NumPy, and PyTorch fixed, on an Apple MacBook

Air, using the Metal Performance Shaders backend of PyTorch. The features were standardized with a scaler fitted on the training blocks, which is stored with the model and used in operation.

Table 5.4 – Hyperparameter search space and best values per sensor

Hyperparameter	**Search space**	**ADXL345**	**MPU9250**	**Piezo**
Sequence length (windows)	fixed	60	60	60
Convolutional filters	{16, 32, 64}	64	32	32
Kernel size	{3, 5}	3	3	5
LSTM hidden units	{32, 64, 128}	128	128	64
LSTM layers	1–2	1	2	1
Dropout	0.1–0.5	0.395	0.427	0.405
Learning rate	10^{-4}–10^{-2} (log)	2.08×10^{-3}	2.68×10^{-4}	6.59×10^{-3}
Weight decay	10^{-6}–10^{-3} (log)	2.25×10^{-6}	1.25×10^{-4}	1.94×10^{-4}
Trainable parameters	–	205 313	564 609	52 097

The performance of the resulting models is evaluated in Chapter 6.

5.5 Online Health Estimation

Moving from offline training to online estimation required adapting the processing pipeline to a continuous stream of sensor data, while reproducing the preprocessing used in training exactly.

5.5.1 Streaming Pipeline

The vibration stream arrives at the PdM node over MQTT, for the ADXL345 and MPU9250, or over the serial link, for the piezoelectric sensor. The ADXL345 and MPU9250 streams are sent at the native rates of the sensors, 3200 Hz and 4000 Hz, and resampled to 2800 Hz on the PdM node by a streaming implementation of the same polyphase filter used in training, whose output is identical to that of the filter applied to a complete recording. The piezoelectric values are checked against the range of the converter, multiplied by the calibration factor of the new sensor (Section 5.3), and linearly interpolated onto a continuous grid at 2800 Hz from their reception timestamps, as in training.

Filtering a stream in one-second pieces raises a practical problem: applying the high-pass filter to each window independently would introduce artifacts at the boundaries between windows, and would compute the features on a signal different from the one used in training. The Butterworth filter is therefore applied causally, keeping its internal state from one window to the next and starting from the same initial state as in training, so that the filtered signal is continuous across windows and identical to the one from which the training features were computed. The stream is then cut into windows of exactly 2800 samples. Windows without signal are discarded, and the sequence is restarted, as in training; for the other windows, the same features are computed, in the same order, which is checked against the order stored with the model, scaled with the scaler fitted during training, and appended to a rolling sequence buffer. Once the buffer holds a complete sequence, it is fed to the model, which outputs an estimate of the HI every second. To verify that the online pipeline reproduces the preprocessing of training, recorded sessions were replayed through the online scripts, and the features they computed were compared, window by

window, with the training features of the same sessions. One session per sensor was replayed, the twelfth for the ADXL345 and the MPU9250 and the eighteenth for the piezoelectric sensor. The median relative difference between the online and the training features was 4.3×10^{-7}, 2.5×10^{-8}, and 3.5×10^{-8}, respectively, at the level of floating-point precision, and no window had a median relative difference above 10^{-3}; the largest differences in individual windows, of up to 2%, occurred in the skewness and kurtosis, which are sensitive to small numerical changes when close to zero.

The raw estimates are noisy. Transient operational events, such as the abrupt start of a movement of the arm, can momentarily resemble a deterioration of the gripper, and would, without further processing, risk triggering premature print orders. The estimates of the three models are therefore smoothed in the same way, with a trailing rolling median over 60 windows, which is largely immune to isolated spikes, followed by an EMA with a span of 60 windows ($\alpha = 2/61 \approx 0.033$), seeded with the first median value. Since each window lasts one second, the smoothed estimate reflects the trend of roughly the last minute of operation, at the cost of delaying its response by a similar amount. The smoothed HI is then converted into a RUL with Equation 5.4.

When the operation of an arm has to be interrupted, as when the gripper is handled to introduce a fault (Chapter 6), its stream can be paused without restarting the inference: the sampling of Arms #1 and #3 is stopped through the `STOP` event of `IMU_STREAM` (Section 5.6), and, for Arm #2, whose serial link is read continuously, the incoming values are discarded while a pause flag is set on the PdM node. In both cases, the state of the estimator, that is, its sequence buffer and smoothing, is kept, and, for the piezoelectric stream, the paused interval is removed from the time scale, so that the stream continues as if uninterrupted, in the same way as the sessions of the training data were concatenated.

The models are saved as checkpoints containing their trained weights, their hyperparameters, the dimensions of their input, the ordered list of their features, and the definition of their labels (t_{FPT}, t_{end}, and L), and are reconstructed and loaded on the PdM node, where inference runs on the processor of the Raspberry Pi, in plain PyTorch. The fitted scaler is stored separately. In operation, the PdM nodes produced an estimate every second without falling behind the incoming data, although their memory was nearly fully used; when replaying recorded data instead of live measurements, they reached about five estimates per second. These observations were not the object of a systematic measurement.

5.5.2 Integration with DINASORE

The original intention, following the design of Section 4.3, was to execute the prognostic model directly within a DINASORE FB. During implementation, however, it became apparent that this was not possible: although its documentation states support for Python 3.6 or later, DINASORE did not run on interpreters more recent than Python 3.8, whereas the version of PyTorch used by the models requires a more recent interpreter, the models having been developed with Python 3.13. The native access to ML libraries that motivated the choice of DINASORE is thus bounded, in practice, by the interpreter versions the runtime supports.

This limitation was circumvented by separating the inference from the FB runtime. A custom `RUL_ESTIMATION` FB acts as an orchestrator: upon a `START` event, it launches the inference script of its sensor as a background process, in a dedicated Python virtual environment with the

required interpreter and libraries, and terminates it safely upon a `STOP` event. After each prediction, the script writes the latest smoothed HI, a status flag, and the corresponding RUL to a file, replacing it atomically, so that a partially written value is never read. The FB reads this file upon a `POLL` event, at most twice per second, and fires a `DATA` event that exposes the HI and the RUL, in hours, as output variables; since the `DATA` event is also connected back to the `POLL` input, the block keeps reading the file in a continuous loop. Until the first complete sequence has been processed, the block exposes a RUL of -1, meaning that no estimate is yet available. It also records every new estimate, with its timestamp, in a log file on the PdM node, and has the inference script record the features of every window, so that a live run can be analyzed afterwards without storing the raw signal. This file-based bridge avoids blocking the runtime while the model computes, at the cost of an additional process on each PdM node and of an indirect coupling between the model and the FB application, whose consequences are discussed in Section 5.7.

5.6 Integration of the Complete System

The complete system integrates the control of Section 5.2, the data acquisition of Section 5.3, and the online estimation of Section 5.5 into a running CPPS. Besides the MQTT FBs of the DINASORE repository [262], it relies on the custom FBs listed in Table 5.5.

Table 5.5 – Custom FBs developed for the system

FB	Device	Function
`ARM_CONTROL`	Control device	Executes the movements described in an XML file when triggered
`IMU_STREAM`	Sampling device	Runs the C sampling utility and streams the vibration data over MQTT
`RUL_ESTIMATION`	PdM node	Runs the inference script in a separate environment and exposes the smoothed HI and the corresponding RUL
`RUL_THRESHOLD`	PdM node	Compares the estimated RUL with the threshold and emits the print order once
`PRINT_QUEUE`	AdM node	Forwards print orders when the printer is idle, or holds them for release by the operator
`PRINT_START`	AdM node	Selects and starts the print of the stored file through the OctoPrint API; can be cancelled
`PRINT_CHECK`	AdM node	Queries the state and progress of the print job once per second

For Arms #1 and #3, `IMU_STREAM` runs on the sampling device of the arm, invoking the same C sampling utility used during data collection as a subprocess, at the same sampling rate, and publishes the resulting stream over MQTT to the broker of the arm's PdM node. For Arm #2, the Pico W is treated as a peripheral directly attached to the PdM node, consistent with the device model of IEC 61499, in which the local process interfaces of a device are distinct from its communication interfaces [269]; its `RUL_ESTIMATION` instance reads the serial link directly, without a sampling device or `IMU_STREAM`.

On each PdM node, `RUL_THRESHOLD` implements the prescriptive logic of Section 4.2: whenever a new estimate is exposed by `RUL_ESTIMATION`, it compares the estimated RUL with a configured threshold, expressed in hours, and, when the estimate falls below it, emits an event that is

published over MQTT to the broker of the AdM node, where PRINT_QUEUE receives it. The order is emitted only once: the block is then disarmed, and is only rearmed when the estimated RUL rises again above the threshold plus 30 minutes, as when the gripper has been replaced and the estimate returns to that of a healthy one, so that the fluctuations of the estimate around the threshold cannot issue repeated orders. Negative values, which signal that no estimate is yet available, are ignored.

The thresholds were calibrated as described in Section 4.2. The time needed to act was set to 3.6 h, that is, 3 h 36 min: the duration of the print of the replacement part, measured at 3 h 26 min in a trial print, plus about 8 minutes for its post-processing and the replacement of the gripper on the arm, rounded up to 10 minutes. The error of each model was evaluated on its test sequences whose real RUL lay between 2.5 and 7 hours, the region of the life cycle around the decision, after converting the estimated HI into a RUL with Equation 5.4. The margin was set as the 95th percentile of the difference between the estimated and the real RUL, positive when the estimate is optimistic, with a minimum of 15 minutes, which was not reached for any sensor, and the threshold as the resulting target RUL, with which RUL_THRESHOLD compares the estimated RUL directly. Table 5.6 summarizes the calibration. The three thresholds lie between 3 h 56 min and 4 h 38 min. For reference, the table also gives the HI that corresponds to each threshold under Equation 5.4; the differences between these values reflect, above all, the different length of the degradation period of each sensor's labels, rather than a different time to act.

Table 5.6 – Calibration of the thresholds of the prescriptive logic, from the RUL error on test sequences with a real RUL between 2.5 and 7 h

	ADXL345	**MPU9250**	**Piezo**
Degradation period L	12 h 26 min	15 h 11 min	17 h 32 min
Test sequences in the region	51	57	53
Mean error (estimated − real)	+4 min	−1 min	−10 min
Mean absolute error	20 min	28 min	15 min
Margin (95th percentile)	52 min	1 h 02 min	20 min
Threshold: target RUL (3 h 36 min + margin)	4 h 28 min	4 h 38 min	3 h 56 min
as configured in RUL_THRESHOLD	4.47 h	4.63 h	3.93 h
Equivalent HI	0.441	0.381	0.288

The applications of all the devices are developed and deployed from the 4diac IDE, which is also used to supervise the running system through the OPC UA server embedded in each DINASORE FB: the current HI and RUL of each arm, the response to each print order, and the progress of the print can all be followed from it. The resulting applications are shown in Appendix B.

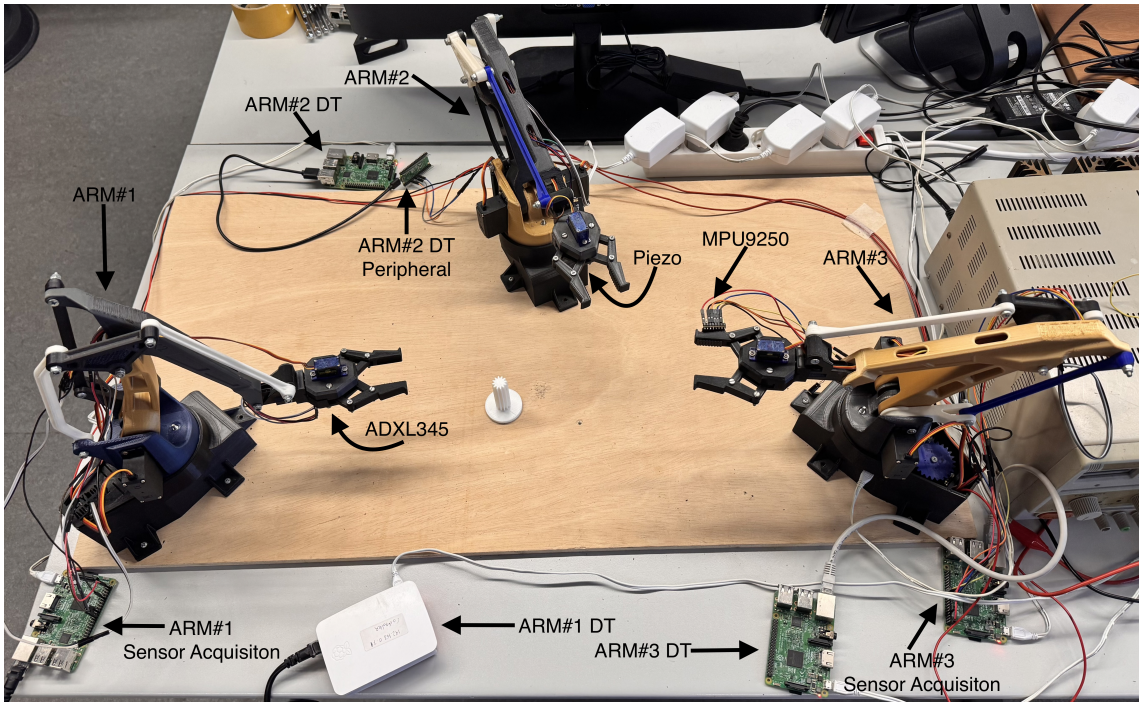
5.7 Implementation Limitations

The implementation entails limitations that bound the scope of the prototype and must be considered when interpreting the results of Chapter 6:

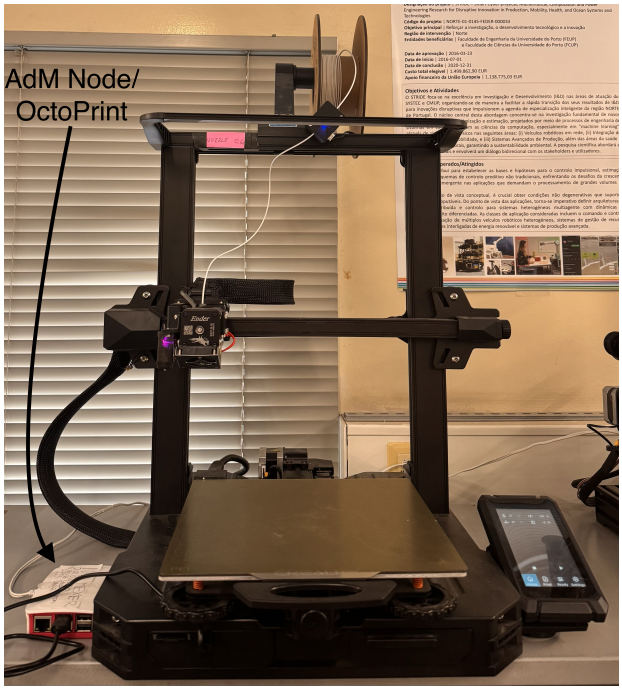
- **Fidelity of the equipment** — The testbed relies on hobbyist-grade arms made largely of PLA, rather than industrial machinery. Their vibration signatures and failure mechanisms,

such as a loosened joint, differ from those typical of industrial equipment, such as bearing wear or structural fatigue, and the faults were introduced artificially to accelerate degradation;

- **A single life cycle per sensor** — Each model was trained and tested on data from one run-to-failure life cycle, split into blocks. Since the test blocks come from the same life cycle as the training blocks, the test error measures how well the model reproduces the labels of that life cycle, and is likely to be optimistic with respect to an unseen one; the same applies to the margins of the thresholds, which were derived from it;
- **Labels that encode time** — The HI labels were constructed from an assumed exponential degradation and, after the FPT, depend on the position of each window in the life cycle (Section 5.4.2). The thresholds, set from the same life cycle, assume that the grippers degrade in operation at a pace similar to the one observed in the data;
- **Incomplete and corrected recordings** — About four and a half hours of the ADXL345 recording are missing, around the FPT of its labels, and the piezoelectric sensor was replaced during the collection, its later recordings being rescaled by a single factor, which also removes any genuine change in the amplitude of the vibration between the two sessions used to compute it;
- **Point estimates** — The models output a single HI value, without a quantification of its uncertainty; the margin of the thresholds accounts for the error observed on test data, not for the uncertainty of each estimate, and the smoothing that makes the estimate robust also delays its response by about a minute;
- **Heterogeneous sensing** — The three arms use different sensors, sampling rates, and acquisition paths, so the three models are not directly comparable;
- **Runtime constraints** — As anticipated in Section 4.3, DINASORE provides no hard real-time guarantees. In addition, the interpreter versions it supports prevented the models from running within its FBs, which required running them in separate processes bridged by files, adding load to the PdM nodes, whose memory was nearly exhausted;
- **Message delivery** — Print orders are published with QoS 0 over a network that is mostly wireless, so their delivery is not guaranteed by the protocol.



(a)



(b)

Figure 5.1 – Photographs of the experimental testbed: (a) the three EEZYbotARM Mk2 arms, fitted with the ADXL345, piezoelectric, and MPU9250 sensors on Arms #1 to #3, respectively; (b) the Ender-3 S1 Pro 3D printer and the Raspberry Pi that hosts its OctoPrint and DINASORE instances

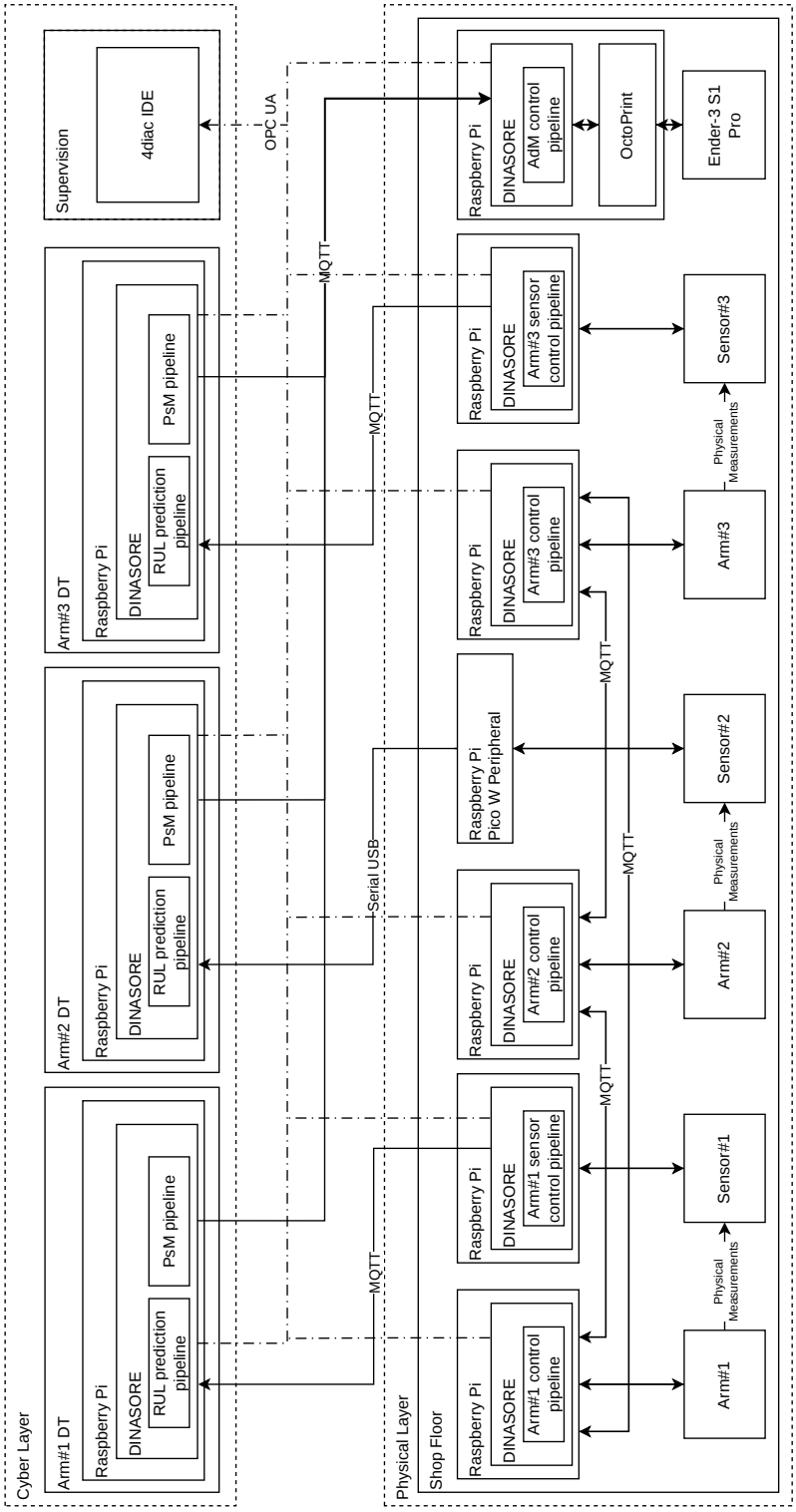


Figure 5.2 – Proposed system architecture adapted to the testbed

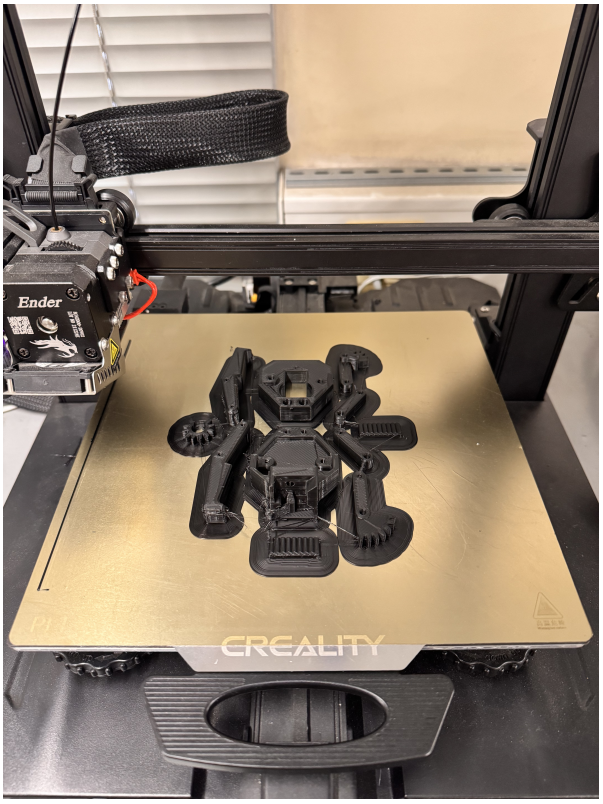


Figure 5.3 – Gripper parts produced by the AdM node in a single print job

Chapter 6

Validation and Results

This chapter presents the validation of the system implemented in Chapter 5 and its results, completing objective **O4** (Section 1.3). Section 6.1 describes the validation strategy, organized in two phases, the individual validation of each part of the system and the validation of the complete system, together with the procedure of each test and the reference against which its results are assessed. Sections 6.2 to 6.4 report the individual validation of, respectively, the control and data acquisition, the prognostic models, and the triggering of the replacement part, and Section 6.5 reports the end-to-end validation of the integrated system. Section 6.6 summarizes the results, which are discussed in Chapter 7.

6.1 Validation Strategy

The validation was organized in two phases. The individual validation evaluated each part of the system in isolation, so that potential sources of failure could be identified before the system was evaluated as a whole. The complete validation then evaluated the integrated system end to end, assessing whether it can close the loop between the estimation of the health of a component and the physical production of its replacement. Table 6.1 summarizes the tests of both phases, the metrics they report, and the reference against which their results are assessed.

6.1.1 Individual Validation

Three parts of the system were validated individually:

- **Control and data acquisition** — The coordination of the arms over MQTT and the streaming of sensor data to the PdM nodes are preconditions for everything else. They were validated by operating the line and streaming the sensor data for one hour, and measuring the regularity of the production cycles, the latency and jitter of the coordination messages, and the continuity of the sensor streams;
- **Prognostic models** — The three models were evaluated on their held-out test sequences, against the HI labels constructed in Section 5.4.2, using the RMSE and the MAE (Section 2.3.4), complemented, after the FPT, by the error of the RUL derived from the estimated HI with Equation 5.4. To put these errors in perspective, the models were compared

Table 6.1 – Overview of the validation tests

Test	Procedure	Metrics	Assessed against
Control and data acquisition (Section 6.2)	One hour of operation of the line, with the sensor data streamed to the PdM nodes	Cycles completed and their duration; latency and jitter of the coordination messages; packets and samples lost; longest gap in the streams	All cycles completed; gaps in the streams shorter than the one-second analysis window
Prognostic models (Section 6.3)	Evaluation on the held-out test sequences, comparison with simpler regressors, and evaluation with whole sessions held out	RMSE and MAE of the HI; error of the derived RUL after the FPT	Errors of the simpler regressors; errors with whole sessions held out
Triggering of the replacement part (Section 6.4)	30 print orders sent from the PdM node of each arm by a test script	Orders received and prints started; round-trip latency	Every order executed; latency negligible compared with the duration of the print
End-to-end validation (Section 6.5)	One new degradation run per arm, with the complete system in operation	Time of the crossing; response time; completion of the part; complete failure; advance notice; margin	Part completed at least the installation time before the failure; target RULs and intended margins

with three simpler regressors trained on the same sequences. Since the test blocks are drawn from the same life cycle as the training blocks, the models were also evaluated with whole sessions held out, which is closer to the conditions of a new degradation run;

- **Triggering of the replacement part** — The routing and execution of the print order were validated by running a test script on the PdM node of each arm, which published print orders to the broker of the AdM node in place of the prescriptive logic, and verifying that each order starts the print, while measuring the round-trip latency of this chain. The emission of the order by the prescriptive logic, from a real estimate, is exercised by the complete validation.

6.1.2 Complete Validation

The complete validation consisted of closed-loop, end-to-end trials of the integrated system on grippers degraded anew, rather than on the data used to train the models. One new accelerated degradation run was carried out on each of the three arms, following the procedure of Section 5.3, with one difference in the schedule: the healthy phase was shortened from sixteen sessions to three, that is, to about three hours of operation. As during the data collection, faults were then introduced in five successive stages, one at the start of each session from the fourth, at 3 h 00 min of operation, to the eighth, and the run was continued until the complete failure of the gripper. A shorter healthy phase reduced the duration of each run and, by placing the faults at a moment of operation very different from the one in the training data, made the trials sensitive to whether the models respond to the condition of the gripper or to the elapsed operating time (Section 7.4). The three trials were run in sequence, one arm at a time, so that the printer was always idle when a print order arrived and `PRINT_QUEUE` forwarded each order immediately. The thresholds were those of Table 5.6, and the complete failure of a gripper was defined as the moment from which it, and therefore its arm, could no longer pick up the piece and place it where intended.

In these trials, the RUL estimated online is left to decrease until it falls below the configured threshold, and the system is expected to trigger the print order and start printing the replacement part with no human involvement between the acquisition of the sensor data and the start of the print, other than the physical introduction of the faults. For each trial, the time at which the threshold was crossed, the response time between the publication of the print order and the start of the print, the duration of the print, and the time of the complete failure of the gripper were recorded, the times of operation being measured from the start of the run. Two quantities were derived from them: the advance notice, the time between the crossing of the threshold and the complete failure, which is compared with the target RUL of the threshold; and the margin, the time between the completion of the part and the complete failure, which is compared with the margin intended when the threshold was set, plus about 10 minutes for installation (Section 5.6). A trial is considered to have closed the loop in time if the part is completed at least the time needed to install it, about 10 minutes, before the failure. The part was not installed before the failure: each trial was continued until the complete failure of the gripper, so that the margin could be measured, and the part was installed afterwards, to confirm that the arm resumed its normal operation.

The response time was measured from the publication of the print order by the PdM node to the transition of the printer to the printing state. Since the two nodes are in different laboratories, and their applications could not be modified or interrupted during the trials, this was done with observers external to the applications, on nodes whose clocks were synchronized with chrony, the PdM node of the arm under test acting as the time server of the AdM node. On the PdM node, the publication of the order was timestamped by the kernel, in a capture of the outgoing MQTT packets; on the AdM node, its arrival was timestamped by a separate client subscribed to the same topic, and the transitions of the state of the printer were taken from the log of OctoPrint, with a resolution of one millisecond. The uncertainty of the synchronization was estimated, every two seconds, as the offset of the clock of the AdM node plus half the delay and the dispersion of its measurements of the time server, and the worst value during each trial is reported. As a cross-check, the observer also polled the state of the print job through the OctoPrint API every 20 ms after the arrival of the order, so that the transition taken from the log could be checked against the interval bounded by the last poll before it and the first poll after it.

6.2 Control and Data Acquisition

The control and data acquisition were validated over one hour of continuous operation of the line, with the arms executing the production cycle described in Section 5.2 while their sensor data was streamed to the corresponding PdM nodes. This test was carried out before the online pipeline was revised, when the ADXL345 stream was still sent at 2800 Hz, rather than at the native rate of 3200 Hz described in Section 5.5; it was not repeated, so the stream of Arm 1 at its current rate was only exercised in the end-to-end trials of Section 6.5. The results are given in Table 6.2. For the evaluated metrics, coordination latency is defined as the time between the publication of the message by one arm and its reception by the next; MQTT jitter and sensor jitter represent the variation of the interval between consecutive coordination and sensor messages, respectively; and samples lost indicates the fraction of expected samples missing from the received stream.

Table 6.2 – Distributed control and sensor data acquisition results

Metric	Arm 1	Arm 2	Arm 3	Overall
Control hand-off cycles	–	–	–	84
Successful control cycles	–	–	–	100 %
Mean cycle duration (s)	–	–	–	43.65 ± 0.02
Max cycle jitter (s)	–	–	–	0.14
Mean coordination latency (ms)	–	–	–	12.4 ± 3.8
Max coordination delay (ms)	–	–	–	114
Mean MQTT jitter (ms)	5.70 ± 1.70	5.81 ± 2.23	6.08 ± 4.57	–
Max MQTT jitter (ms)	24.21	28.57	97.74	–
Sensor link	MQTT	Serial	MQTT	–
Mean sensor jitter (ms)	3.96	n/a	13.41	–
Packet loss (%)	0.00	n/a	0.00	–
Samples lost (%)	<0.54	0.00	0.00	–
Longest observed gap (s)	0.246	0.695	0.910	–

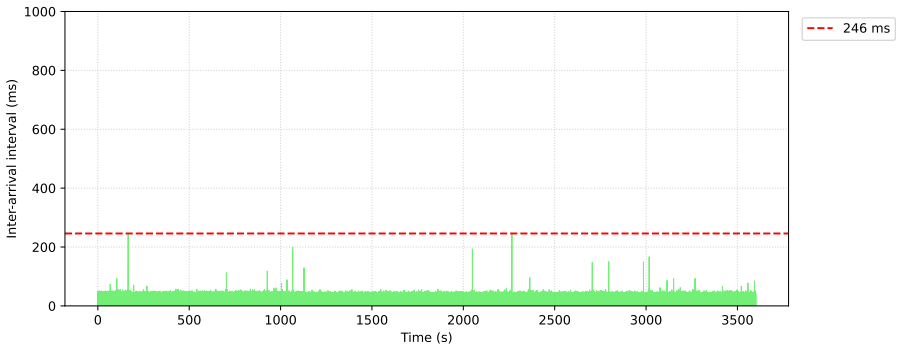


Figure 6.1 – MQTT inter-arrival interval between messages of Arm #1's sensing pipeline

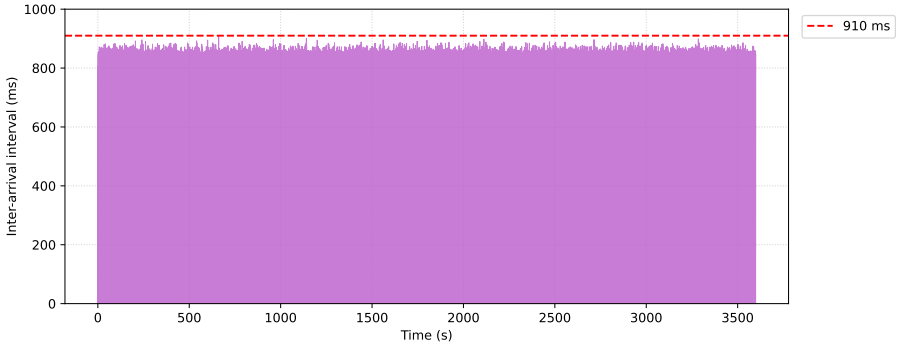


Figure 6.2 – MQTT inter-arrival interval between messages of Arm #3's sensing pipeline

All 84 production cycles completed during the hour were executed successfully, with a mean duration of 43.65 s and a standard deviation of 0.02 s, the longest deviation from the mean being 0.14 s. The coordination messages between the arms took 12.4 ms on average to be delivered, with a maximum of 114 ms, and their jitter was similar for the three arms, around 6 ms on average, although Arm 3 showed occasional peaks of up to 98 ms. Since a movement of an arm lasts several seconds, delays of this magnitude have no visible effect on the operation of the line, whose cycle duration remained practically constant.

No MQTT packets of sensor data were lost on the links of Arms 1 and 3, and no samples were lost on Arms 2 and 3, while less than 0.54 % of the samples of Arm 1 were missing. The streams were not, however, free of interruptions: the longest gaps observed were 0.246 s, 0.695 s, and 0.910 s for Arms 1, 2, and 3, respectively, the latter two approaching the duration of a one-second analysis window. The mean jitter of the sensor messages was also higher for the MPU9250 of Arm 3, at 13.41 ms, than for the ADXL345 of Arm 1, at 3.96 ms. These interruptions do not, by themselves, affect the analysis windows. For the ADXL345 and MPU9250, the online pipeline forms its windows from the received samples, in order, rather than from their times of arrival (Section 5.5), so a delay in the delivery of a message does not alter them, and only missing samples, such as those of Arm 1, make a window span slightly more than one second of operation. For the piezoelectric sensor, whose values are interpolated from their times of reception, a gap is bridged by interpolation, as it was in the training data. Analysis of the inter-arrival times revealed that these longest gaps are not anomalous spikes, but rather variations very close to the median interval, as is shown by Figures 6.1 and 6.2, reflecting normal network and buffering behavior.

6.3 Prognostic Models

Each of the three models described in Section 5.4.3 was evaluated on its held-out test set, formed by one sequence per test block, none of which was used for training or for the optimization of the hyperparameters. Table 6.3 reports the resulting metrics, computed against the constructed HI labels, together with the error of the RUL obtained from the estimated HI through Equation 5.4, for the test sequences after the FPT. Figures 6.3 to 6.5 show the predicted HI of each model against its labels over the operating time.

Table 6.3 – Health estimation performance on the held-out test sequences

		HI, all test sequences		**RUL, after the FPT (h)**	
Arm	**Sensor**	**RMSE**	**MAE**	**RMSE**	**MAE**
1	ADXL345	0.0492	0.0292	0.72	0.49
2	Piezoelectric	0.0412	0.0294	0.83	0.57
3	MPU9250	0.0840	0.0472	1.55	0.93

The piezoelectric and ADXL345 models achieved similar errors, with an RMSE of 0.0412 and 0.0492, respectively, while the MPU9250 model showed the highest, with an RMSE of 0.0840, about twice that of the other two. Since the HI ranges from 0 to 1, errors of this magnitude correspond to a few percent of the full scale of the indicator. Expressed as a RUL, the mean absolute error after the FPT ranged from about half an hour, for the ADXL345 and piezoelectric models, to about one hour, for the MPU9250 model, over degradation periods of 12 to 18 hours.

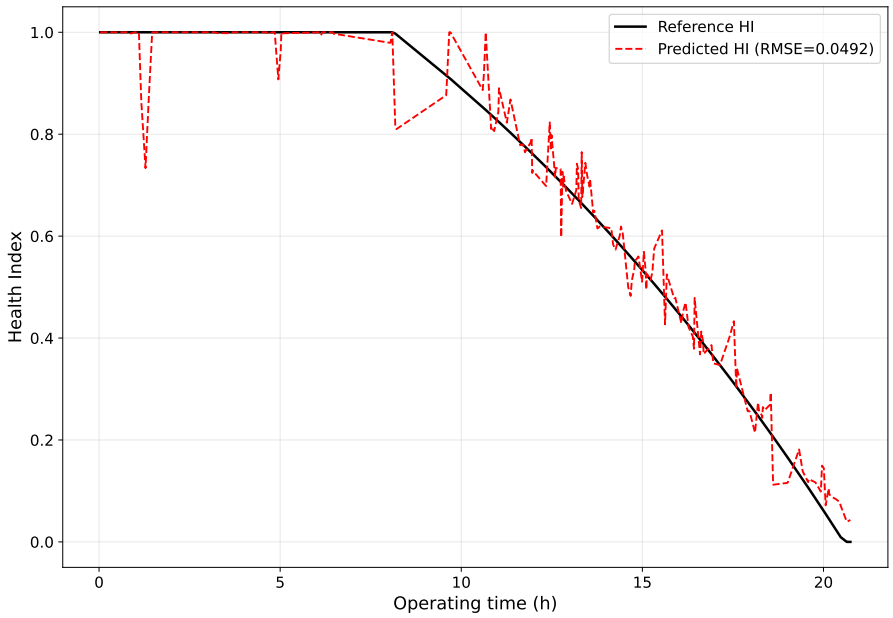


Figure 6.3 – Predicted and labelled HI on the held-out test sequences of the ADXL345 model

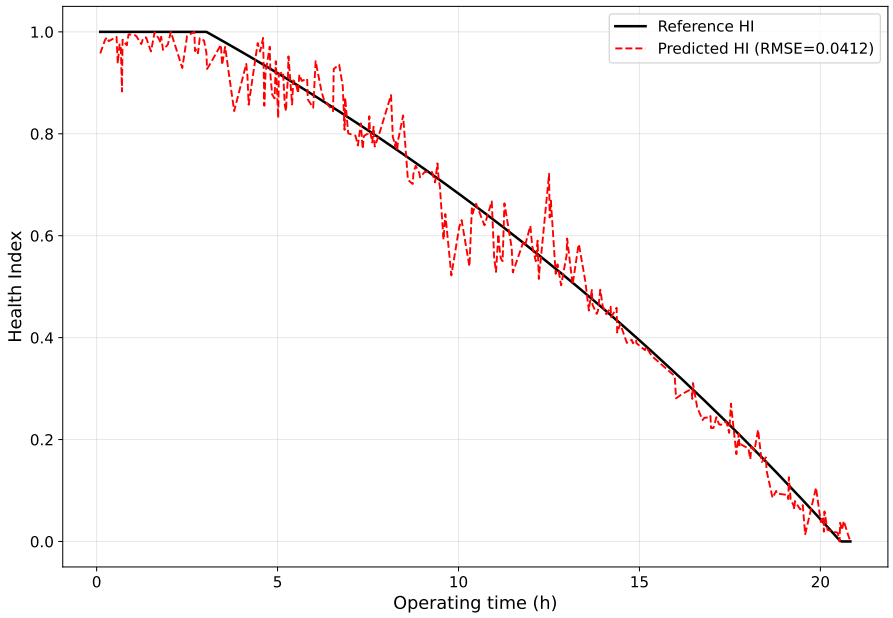


Figure 6.4 – Predicted and labelled HI on the held-out test sequences of the piezoelectric model

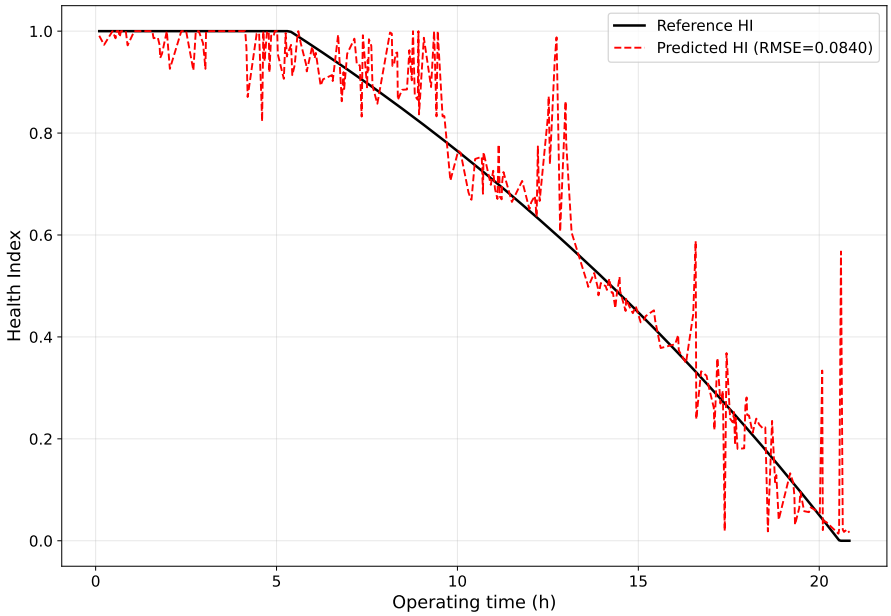


Figure 6.5 – Predicted and labelled HI on the held-out test sequences of the MPU9250 model

To put these errors in perspective, the models were compared with three simpler regressors, trained on the same training sequences and evaluated on the same test sequences: a random forest, with 100 trees and at least five samples per leaf, and a ridge regression, both fed with the mean of the features over the sequence and the features of its last window; and a linear regression on the mean root mean square of each axis over the sequence, which is equivalent to a threshold on the vibration amplitude. The results are given in Table 6.4. The CNN-bidirectional LSTM models achieved the lowest errors for the three sensors. The margin was large for the ADXL345 and MPU9250, whose random forest reached an RMSE about 2.7 and 1.4 times higher, respectively, but small for the piezoelectric sensor, for which the random forest came within 0.008 of the RMSE of the deep model. The linear regression on the amplitude alone performed worst in all cases, with errors of about a quarter of the scale of the HI.

Table 6.4 – RMSE and MAE of the HI estimated by the proposed models and by simpler regressors, on the same test sequences

	ADXL345		Piezoelectric		MPU9250	
Model	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
CNN-bidirectional LSTM	0.0492	0.0292	0.0412	0.0294	0.0840	0.0472
Random forest	0.1324	0.0650	0.0486	0.0307	0.1197	0.0719
Ridge regression	0.1750	0.1379	0.1303	0.1024	0.1766	0.1372
Linear regression on amplitude	0.2726	0.2256	0.2231	0.1849	0.2322	0.1775

The random split into blocks places windows of every session in both the training and the test sets, so the errors above measure how well a model reproduces the labels of a life cycle from which it has already seen neighboring windows. To approximate the conditions of a new degradation run, the models were also evaluated with whole sessions held out. The sessions of each sensor were divided into five interleaved groups (the first containing sessions 1, 6, 11, 16, and 21, the second sessions 2, 7, 12, 17, and 22, and so on), so that each group covers the whole life cycle; for the ADXL345, which has no data from the eighth session, the groups were formed in the same way

over its 21 sessions. For each group, a model was trained from scratch on the remaining sessions, with the same architecture, hyperparameters, and training procedure, and with its own scaler, and evaluated on every tenth sequence of the held-out sessions. In the end, every session was estimated by a model that had never seen it. The results are given in Table 6.5 and shown in Figure 6.6.

Table 6.5 – HI estimation errors with whole sessions held out (five interleaved groups of sessions)

	ADXL345	Piezoelectric	MPU9250
Sequences evaluated	5735	7402	7310
RMSE, all sequences	0.1197	0.1072	0.1777
MAE, all sequences	0.0765	0.0807	0.1304
RMSE, before the FPT	0.0440	0.0568	0.1420
RMSE, after the FPT	0.1453	0.1135	0.1886
RMSE, block split (Table 6.3)	0.0492	0.0412	0.0840

With whole sessions held out, the errors were between 2.1 and 2.6 times higher than on the test sequences of the block split, with an RMSE of 0.120, 0.107, and 0.178 for the ADXL345, piezoelectric, and MPU9250 models, respectively. The models remained close to the labels before the FPT, where the HI is constant, except for the MPU9250, and the larger part of the error arose after it, during degradation. The ranking of the three sensors was preserved.

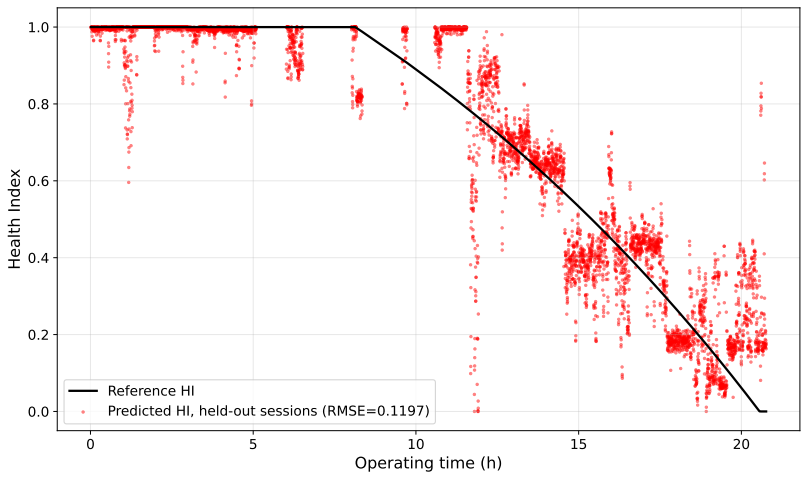
6.4 Triggering of the Replacement Part Production

The routing and execution of the print order were validated with a test script, run on the PdM node of each arm in place of its prescriptive logic. In each trial, the script cancelled any active print and waited for the printer to become idle, published a print order on the topic to which the AdM node subscribes, and measured the time until the AdM node published the acknowledgment that the start of the print had been requested from OctoPrint. The test was repeated 30 times on each of the three PdM nodes, for a total of 90 trials. As the AdM node is located in a different laboratory from the arms (Section 5.1), the orders travelled from the network of the testbed's laboratory, through the campus network, to the AdM node, connected over Wi-Fi to eduroam; once received by the broker of the AdM node, the remaining steps, up to the request to OctoPrint, took place locally on that node. For these trials, the `PRINT_QUEUE` block was removed from the chain, to isolate the direct communication delay. The results are given in Table 6.6.

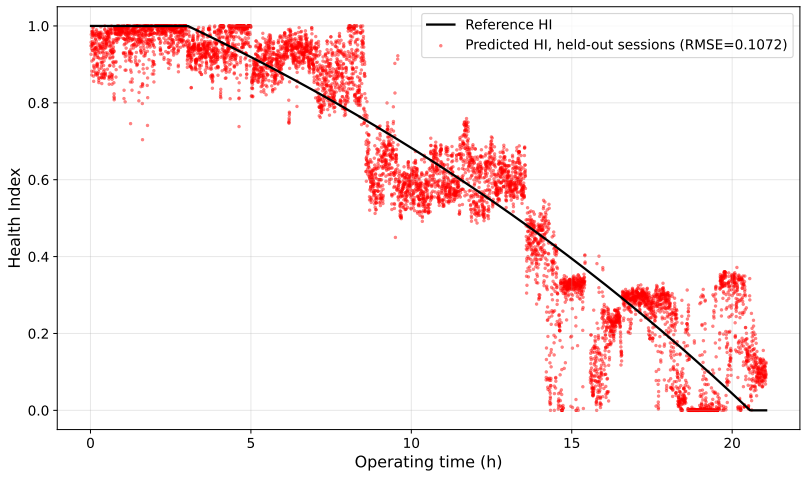
Table 6.6 – Replacement part triggering results

Arm	Trials	Order sent	Order received	Print started	Mean latency (ms)	σ (ms)
1	30	30/30	30/30	30/30	350	98
2	30	30/30	30/30	30/30	339	83
3	30	30/30	30/30	30/30	388	109

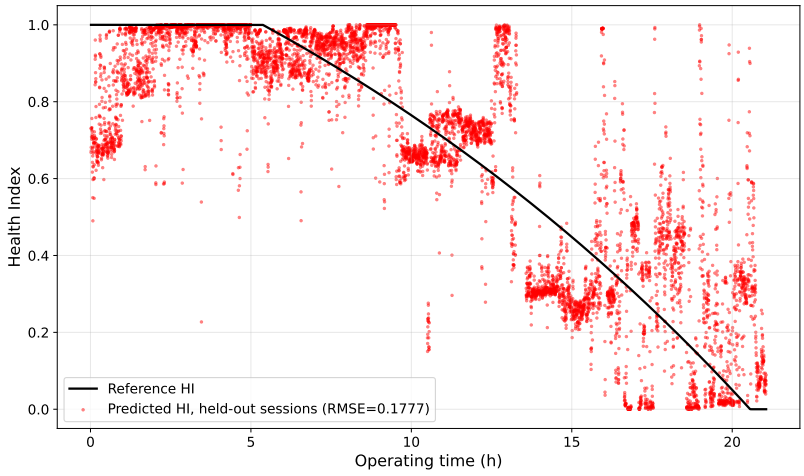
In all 90 trials, the order was received by the AdM node and the print was started. The mean round-trip latency, between the publication of the order and the reception of the acknowledgment, ranged from 339 ms, for Arm 2, to 388 ms, for Arm 3, with standard deviations between 83 and 109 ms, the variability being attributable mainly to the transmission over the shared wireless and campus networks.



(a)



(b)



(c)

Figure 6.6 – Predicted and labelled HI with whole sessions held out: (a) ADXL345; (b) piezoelectric; (c) MPU9250

6.5 End-to-End Validation

The three end-to-end trials were carried out as described in Section 6.1.2, with the live data of each degrading gripper fed into the system, and without any of the manual overrides used in Section 6.4. In each trial, the `RUL_ESTIMATION` block of the respective arm produced a decreasing estimate of the RUL which, after 4 h 07 min to 4 h 16 min of operation, fell below the configured threshold. `RUL_THRESHOLD` then emitted the print order, which was published over MQTT, and the printer started printing the replacement part 207 to 215 ms later, without any manual intervention between the start of the trial and the start of the print, other than the physical introduction of the faults. The grippers failed completely during the ninth session, after 8 h 43 min to 8 h 55 min of operation. The results of the three trials are given in Table 6.7, in which all times are measured from the start of each trial.

Table 6.7 – End-to-end validation results

Arm	Threshold crossed	Response time	Print completed	Complete failure	Advance notice	Margin
1 (ADXL345)	4 h 09 min	215 ms	7 h 46 min	8 h 43 min	4 h 34 min	57 min
2 (Piezo)	4 h 16 min	209 ms	7 h 51 min	8 h 48 min	4 h 32 min	57 min
3 (MPU9250)	4 h 07 min	207 ms	7 h 42 min	8 h 55 min	4 h 48 min	73 min

Response time: from the publication of the print order to the start of the print (Table 6.8). Print completed: crossing of the threshold, plus the response time, plus the duration of the print, of 3 h 37 min, 3 h 35 min, and 3 h 35 min for Arms 1, 2, and 3, respectively. Advance notice: time between the crossing of the threshold and the complete failure, to be compared with the target RUL of Table 5.6. Margin: time between the completion of the part and the complete failure.

Table 6.8 breaks the response time down into its stages. In the three trials, the transition of the printer to the printing state taken from the log of OctoPrint fell within the interval bounded by the polls of the observer, which confirms the consistency of the synchronized timestamps.

Table 6.8 – Stages of the response time in the end-to-end trials, from the publication of the print order

Stage	Arm 1 (ms)	Arm 2 (ms)	Arm 3 (ms)
Transmission of the order over MQTT	11.3	6.0	3.5
Processing on the AdM node, until OctoPrint accepts the order	151.6	135.5	164.0
Start of the print, until the printer is printing	52.0	67.0	39.0
Response time	214.9	208.5	206.5
Uncertainty of the synchronization	±0.7	±2.6	±1.0

The response time, of 207 to 215 ms, was dominated by the processing on the AdM node, between the arrival of the order and its acceptance by OctoPrint, which comprises the FBs of the node and the handling of the REST request, while the transmission of the order between the two laboratories took 3.5 to 11.3 ms. It does not include the time between the estimate falling below the threshold and the publication of the order, which involves only the propagation of events between the FBs of the PdM node and was not measured.

The response time of the system was negligible compared with the duration of the print, of 3 h 35 min to 3 h 37 min, which was, however, 9 to 11 minutes longer than the 3 h 26 min assumed for the print itself when setting the time to act. The thresholds were crossed 4 h 34 min,

4 h 32 min, and 4 h 48 min before the complete failure of the grippers of Arms #1, #2, and #3, respectively, against target RULs of 4 h 28 min, 3 h 56 min, and 4 h 38 min, that is, 6 to 36 minutes earlier than intended. In all three trials, the replacement part was completed before the failure of the gripper, with margins of 57, 57, and 73 minutes, against the margins of 52, 20, and 62 minutes, plus about 10 minutes for installation, intended when the thresholds were set. The margin obtained exceeded the intended one, with installation, by 27 minutes for Arm #2 and by 1 minute for Arm #3, but fell 5 minutes short of it for Arm #1, whose print took 11 minutes longer than assumed, while its threshold was crossed only 6 minutes earlier than intended. Even so, in the three arms, the part was completed with 47 to 63 minutes to spare after the time allowed for its installation, so all three trials met the criterion of Section 6.1.2. The part was installed only after the failure, once the margin had been measured; in the three arms, the installed part restored the normal operation of the line.

6.6 Summary

The individual validation showed that the arms coordinate reliably over MQTT, with all production cycles completed and coordination delays in the order of milliseconds; that the sensor data reaches the PdM nodes without packet loss, although with occasional interruptions of up to about one second; that the three prognostic models reproduce their HI labels with errors of a few percent of the scale of the indicator, lower than those of simpler regressors, but between two and three times higher when whole sessions are held out; and that a print order is transmitted to the AdM node and executed in every trial, with a round trip of less than half a second. The end-to-end validation showed that, on newly degraded grippers with a much shorter healthy phase than the training run, the system crossed the threshold, issued the print order, and started the production of the replacement part with no manual intervention, within about 210 ms of the publication of the order, and that the part was completed 57 to 73 minutes before the failure of the grippers, in time to be installed, with margins close to or above those intended when the thresholds were set, except for Arm #1, whose margin fell 5 minutes short of the intended one because of a longer print. These results, and the limitations that bound them, are discussed in Chapter 7.

Chapter 7

Discussion

This chapter discusses the results reported in Chapter 6 in light of the literature reviewed in Chapter 3. Following the structure of the validation, Section 7.1 discusses the control and data acquisition results, Section 7.2 the prognostic models, and Section 7.3 the triggering of the replacement part. Section 7.4 draws on all three, together with the end-to-end validation, to discuss the closing of the loop between prediction and the provision of the replacement part. Section 7.5 revisits the research gaps identified in Section 3.6.3 and answers the research questions formulated there. Finally, Section 7.6 discusses the limitations of the proposed solution and of the strategy used to validate it.

7.1 Control and Data Acquisition: A Precondition Confirmed

Section 6.2 reported the results of the control and sensor data acquisition, which underlie every other part of the system. All 84 production cycles executed during one hour of operation were completed, with a mean duration of 43.65 s, a standard deviation of 0.02 s, and a mean coordination latency of 12.4 ms between the arms, with a maximum of 114 ms. No MQTT packets of sensor data were lost, and the only samples missing, on Arm 1, amounted to less than 0.54 % of the stream. The longest intervals between sensor messages, which could at first sight be read as interruptions, lie close to the median interval between messages, and therefore reflect the publication of the samples in batches rather than gaps in the acquisition (Figures 6.1 and 6.2).

These results bear on the choice of MQTT for operational messages, which Section 4.3 supported with an evaluation reporting lower latency, smaller packets, and lower bandwidth consumption than HTTP on a Raspberry Pi running FORTE [256]. The results obtained here extend that conclusion to DINASORE, whose Python-based execution provides no real-time guarantees. For the tasks this architecture assigns to it, the trade-off accepted in Section 4.3, giving up hard real-time guarantees in exchange for access to Python ML libraries, held in practice: coordination delays of the order of ten milliseconds are negligible for movements lasting several seconds, as shown by the practically constant duration of the cycles. It should be stressed, however, that these figures describe the behavior observed over one hour, on a lightly loaded network, and do not constitute bounds; an application with strict timing requirements would still call for a runtime that provides them, as discussed in Section 4.3.

The different treatment of the piezoelectric sensor of Arm 2, attached to its PdM node by a serial link rather than streamed over MQTT, was justified in Section 5.6 by the device model of IEC 61499, which distinguishes the local process interfaces of a device from its communication interfaces [269]. The serial link of Arm 2 lost no samples, as did the MQTT link of Arm 3, while less than 0.54 % of the samples of Arm 1 were missing, which indicates that this architectural difference did not translate into a loss of completeness of the acquired data, at least over the monitored hour.

More broadly, the coordination of the arms emerges from direct message exchanges between them, without any device orchestrating the cycle (Section 5.2). The absence of failed cycles is therefore consistent with the argument of Section 2.6.1 that decentralized control avoids the single point of failure of a central coordinator, although this test did not inject failures to verify how the line behaves when one of its devices or brokers stops. As in Section 6.1.1, this reliability is a necessary, but not sufficient, condition for the rest of the loop.

7.2 Prognostic Models: Sensing Modality and Model Behavior

Section 6.3 reported that the piezoelectric and ADXL345 models achieved similar errors against their HI labels on the test sequences, with an RMSE of 0.0412 and 0.0492, respectively, while the MPU9250 model showed the highest, with 0.0840. Several explanations are compatible with this ordering. The three datasets were brought to the same sampling rate before feature extraction (Section 5.4), so the difference is unlikely to stem from bandwidth. Nor can it be attributed to the resolution of the MPU9250: although it was set to its widest range, $\pm 16\,\mathrm{g}$, its resolution in that range, of about 0.49 mg per least significant bit, is still finer than that of the ADXL345, of about 3.9 mg. The difference may instead reflect the noise of the sensors, the way they were mounted on the grippers, the transduction principle, since the accelerometers measure acceleration while the piezoelectric sensor outputs a vibration signal directly, or differences between the three grippers, which, although nominally identical, are individually printed parts, so that part of the gap may reflect genuinely different degradation trajectories rather than sensing performance. The recordings themselves also differ in quality: the ADXL345 lacks about four and a half hours of its life cycle, and the piezoelectric sensor was replaced and its later recordings rescaled (Section 5.3). With a single life cycle per arm, these explanations cannot be separated.

The comparison with simpler regressors (Table 6.4) shows that the CNN-bidirectional LSTM models are justified for two of the three sensors: for the ADXL345 and the MPU9250, a random forest fed with summaries of the same sequences reached errors 2.7 and 1.4 times higher, whereas, for the piezoelectric sensor, it came close to the deep model. The poor performance of the linear regression on the vibration amplitude alone also shows that the HI labels are not a simple function of the level of vibration, so that a monitoring rule based on amplitude would not reproduce them. This comparison, however, is made against the constructed labels, and carries the same caveat as the errors themselves.

As discussed in Section 5.4.2, the HI labels are not an observed ground truth, but a construction: the health stages were obtained by a clustering weighted towards chronological order, and, after the FPT, the HI is a fixed function of the position of each window in the life cycle. The

FPT itself fell between the fourth and the ninth session, well before the first fault was introduced. Placing the windows on the real operating time line made the FPT differ between the sensors, at 15%, 26%, and 40% of the life cycle, so the clustering does not simply divide the life cycle into equal parts; but the shape of the HI after the FPT remains determined by time. Low errors against these labels therefore show that the models reproduce the constructed trajectory closely, but do not, on their own, show whether they learned the condition of the gripper or, at least in part, the elapsed operating time. The end-to-end trials were designed to separate the two, by shortening the healthy phase of the new runs, and are discussed in this respect in Section 7.4.

The evaluation with whole sessions held out quantifies how much of the accuracy of the block split depended on neighboring windows of the same sessions being available for training. Without them, the errors increased by a factor between 2.1 and 2.6, mostly after the FPT, confirming that the test error of the block split is optimistic with respect to a new life cycle, as anticipated in Section 5.7. Even this evaluation is favorable to the models, since the held-out sessions are surrounded by training sessions of the same life cycle, at stages of degradation immediately before and after them. Its consequence for the decision is direct. Expressed as a RUL, in the region of the life cycle in which the decision is taken, with a real RUL between 2.5 and 7 hours, the mean absolute error with whole sessions held out was 1.0, 1.4, and 2.5 hours for the ADXL345, piezoelectric, and MPU9250 models, against 0.3, 0.3, and 0.5 hours on the test sequences from which the margins of the thresholds were derived (Table 5.6). On average, the estimates in this region were pessimistic, which anticipates the print order, but the 95th percentile of their optimistic error, the quantile used to set the margins, of about 1.4, 1.2, and 4.4 hours, largely exceeds the margins of 52, 20, and 62 minutes adopted. The margins were therefore calibrated on an error that underestimates the one to be expected on a new run, and the end-to-end trials, discussed in Section 7.4, are the test of whether they suffice.

The same reading applies to what the metrics measure. For the purpose of this dissertation, the most consequential question is not how closely a model tracks the whole trajectory, but whether it crosses the threshold at the right moment, early enough for the replacement part to be ready and installed before failure. Aggregate errors of a few hundredths of the scale of the HI may hide systematic deviations precisely in the region of the threshold, which is why the error of the RUL in that region was used to calibrate the thresholds, and why the timing of the end-to-end trials is the more direct evidence of the adequacy of the models for their role. Metrics that evaluate predictions against the time available to act, such as the prognostic horizon discussed in Section 2.3.4 or the lead-time accuracy used by Almatared *et al.* [120], would capture this more directly still.

Finally, the three independent models are exactly the alternative that the abandoned sensor fusion strategy would have replaced. Fusion was abandoned on deployment grounds, so that each DT instance can estimate the health of its own asset without depending on the others (Section 5.4.3), rather than on a demonstrated difference in accuracy, and whether independent models are more or less accurate than a fused one was not tested.

7.3 Replacement Part Triggering: Prescriptive Logic and Distributed Routing

Section 6.4 validated the reception and execution of print orders by the AdM node: in all 90 trials, the order was received and the print started, with mean round-trip latencies between 339 and 388 ms, including the transmission of the order, its processing by the FBs of the AdM node, the start of the print through the OctoPrint API, and the return of the acknowledgment. Since the orders were sent from the PdM nodes by a test script, rather than by their prescriptive logic, this test covers the path of the order from each PdM node to the printer, but not its emission; the emission of the order by `RUL_THRESHOLD`, from a real estimate, is demonstrated instead by the end-to-end trials of Section 6.5. In both cases, the results bear on two distinct claims: that the prescriptive logic fires when it should, and that its command is reliably carried to, and executed by, the manufacturing resource.

The first claim concerns a deliberately simple instance of PsM: a single threshold comparison, delivered as an automatically executed action (Section 4.2), rather than the selection among several alternative actions, or the explicit reasoning about costs and uncertainty, that characterize the prescriptive frameworks reviewed in Section 3.2.4 [8]. Its correct operation validates the prescriptive capability at the scope set for this dissertation, a trigger that reliably fires on a predicted health condition, and says nothing about how a richer prescriptive policy would perform within the same architecture.

The second claim concerns the path from the decision to the manufacturing resource. Whereas the coordination of the arms, discussed in Section 7.1, exchanges messages between peers performing the same role, this path carries a command from a decision-making node to an actuating one, in another laboratory, reached through the campus network; the results show that the same MQTT-based, event-driven infrastructure supports both patterns without a dedicated mechanism for each. The latencies obtained, below half a second for a round trip that includes starting the print, show that the orchestration layer contributes a negligible delay to the replacement process, whose duration is dominated by the print itself. Two aspects of this test limit its scope, however: `PRINT_QUEUE` was bypassed, so the handling of concurrent orders was not exercised, and the orders were published with QoS 0, so the reliability observed rests on the network rather than on a delivery guarantee of the protocol.

7.4 Closing the Loop: From Prediction to the Replacement Part

Section 6.5 reported the end-to-end validation, in which a new accelerated degradation run was carried out on each arm, independently of the data used to train and test the models, and the system was left to operate without any manual intervention in the decision until the complete failure of the grippers. This is the only test capable of demonstrating the complete chain: the results discussed in Sections 7.1 to 7.3 show that each link is reliable in isolation, but only their combination, exercised on new data, shows that the chain closes. Using new runs, rather than replaying held-out data from the original life cycle, is what allows this test to speak to both the reliability of the pipeline and the behavior of the models on unseen degradation.

The shortened healthy phase of the new runs also bears on the question left open in Section 7.2, whether the models respond to the condition of the gripper or to the elapsed operating time. In the training run, the faults were introduced after 15 h 34 min of operation, and the labels reach the thresholds of the three sensors between 22 and 64 minutes later, at 15.9 to 16.6 h of operation. A model that had learned the elapsed operating time would therefore cross its threshold at about the same time in the new runs, regardless of when the faults were introduced, that is, long after the failure of grippers whose faults started in the fourth session. A model that responds to the condition of the gripper would instead cross it according to the progression of the faults. In the three trials, the thresholds were crossed after 4 h 07 min to 4 h 16 min of operation, early in the fifth session, 1 h 07 min to 1 h 16 min after the first fault was introduced, in the pause between the third and the fourth sessions, at 3 h 00 min of operation, and not near 16 h. This indicates that the models respond to the condition of the grippers, or at least to something that follows the progression of the faults, rather than to the elapsed operating time. This evidence is limited to one run per arm, and the faults were of the same kind, and introduced in the same way, as in the training run, so the models may still rely on features of that particular kind of intervention.

In the three trials, the printer started printing 207 to 215 ms after the print order was published, and the print of the replacement parts took between 3 h 35 min and 3 h 37 min. The response time of the system is therefore negligible compared with the duration of the print: in the terms of Section 2.3.5, the time to act is determined by the production of the part and its installation. This is the counterpart, for on-demand production, of the observation made in Section 3.5.2 that, although AdM shortens the delivery of spare parts compared with procurement, its production times are measured in hours. Here, those hours could not be taken from the estimate of the slicer, which the measured build time exceeded by about a third (Section 5.2), but, since the same sliced file is printed every time, they could be measured in advance, which is what allows them to be taken into account by the decision. Even so, the prints of the trials took 9 to 11 minutes longer than the print measured in advance, which shows that the build time of the same file also varies from one print to the next, and that the time to act should allow for this variation.

The central result of the validation is the timing. The thresholds were set so that the estimated RUL at the moment of the decision would cover the production and installation of the part, 3 h 36 min, plus a margin derived from the error of the models on test data, of 20 to 62 minutes (Section 5.6). In the trials, the thresholds were crossed 4 h 32 min to 4 h 48 min before the failure, 6 to 36 minutes earlier than the target RULs, and the part was completed 57 to 73 minutes before it, against intended margins of 20 to 62 minutes, plus about 10 minutes for installation. In all three trials, the part was therefore available in time to be installed before the failure, even though the evaluation with whole sessions held out suggested that the error on a new run could exceed the margins (Section 7.2). Three qualifications apply. First, the intended margins were not met in every case: for Arm #1, the margin fell 5 minutes short of the intended one, as its print took 11 minutes longer than assumed, while its threshold was crossed only 6 minutes earlier than the target, and, for Arm #3, it exceeded it by only 1 minute. Second, the margins did not come from conservative estimates: the thresholds were crossed 1 h 07 min to 1 h 16 min after the first fault, later than the labels of the training run reach them, 22 to 64 minutes after it, but the new grippers lasted 5 h 43 min to 5 h 55 min after the first fault, against five hours in the training run.

Had they failed as soon after the first fault as in the training run, with the thresholds crossed at the same moment, the parts would have been completed only 9 to 18 minutes before the failure, about the time needed to install them, and, for Arm #2, less than it. Third, with one run per arm, this does not establish that the calibration would be sufficient on other runs, whose variability was not characterized. Even so, the result illustrates, on a working system, the problem discussed in Section 2.3.5: a prediction is only useful if it arrives while the remaining time still exceeds the time needed to act, with a margin for the uncertainty of the prediction. It also shows why the decision rules reviewed in Section 3.2.2, which base the decision on the lower bound of the remaining life [118], penalize optimistic estimates [119], or calibrate the trigger to a conservative quantile [120], are a requirement for such a system to operate reliably. The rule adopted here, a margin given by a quantile of the observed error, is the simplest member of that family; replacing it with a quantified uncertainty of each estimate would require no change to the architecture, since the threshold is a parameter of the `RUL_THRESHOLD` block.

Throughout the trials, no human evaluated the health estimate, decided whether or when to print, or selected the part to be produced. In the terms of Section 2.4.3, information acquisition, information analysis, decision, and the production of the part were fully automated, with the operator on the loop for the decision, able to cancel the print order, and in the loop only for the installation of the part (Section 4.5). The trials therefore demonstrate the notion of AM adopted in Section 2.2.2, in which the production system itself carries out monitoring, analysis, decision, and the production of the replacement part, while the physical intervention on the equipment remains with the operator, consistent with the human-centric orientation of I5.0 [23]. In the trials, the installation was deliberately deferred until after the failure, so that the margin could be measured; in all three trials, the part was completed early enough for the operator to have installed it before the failure, with 47 to 63 minutes to spare after the time allowed for installation.

7.5 Revisiting the Research Gaps and Questions

Section 3.6.3 identified three research gaps, from which the research questions of this dissertation were formulated. Table 7.1 summarizes the answer to each question, the evidence on which it rests, and its main qualification; the remainder of this section considers each gap and question in turn.

The first gap (G1) concerned the step from the prediction of a failure to the physical provision of the replacement part: in the reviewed literature, prognostic decisions end in recommendations or purchase orders, the time to act is the lead time of an external supplier, and the production of spare parts by AdM is triggered by demand and decided by planners. The system developed in this dissertation replaces the external procurement step by on-demand production triggered directly by the prognostic decision, with a lead time measured in advance. The end-to-end trials show that this step can be closed autonomously: in three new degradation runs, the print was triggered autonomously and the part was completed 57 to 73 minutes before the failure of the gripper, with no human decision. They also show that closing it in time depends on calibrating the decision to the full time to act, including installation, and to the error of the prediction on unseen degradation: the margins obtained were close to those intended, falling short of them for one arm, and owed

Table 7.1 – Summary of the answers to the research questions

Question	Answer	Main evidence	Main qualification
RQ#1	Up to the provision of the part, delivered in time in all three trials	Print started about 210 ms after the order; part completed 57 to 73 min before failure on new degradation runs	One run per arm; margin 5 min below the intended one for Arm #1 and partly owed to slower degradation than in training; installation remains human
RQ#2	To a large extent, when the twin is scoped at the level of the system	Prediction, decision, and print command integrated in one loop, reaching the printer in a fraction of a second without centralized information systems	Model not updated; model bridged to the runtime through a separate process; shared printer coordinated by the operator
RQ#3	For decision-support workflows at prototype scale	Coordination, acquisition, estimation, decision, and printing supported without failures on nine devices	Runtime limited by its Python versions and memory; no real-time guarantees; a single runtime tested

part of their size to grippers that degraded more slowly than in the training run, at whose pace they would barely have covered the installation of the part.

- **RQ#1** — *To what extent can on-demand AdM autonomously bridge the gap between PdM and physical maintenance execution?* **Up to the provision of the replacement part, which it delivered in time in all three trials.** On-demand AdM bridged the gap up to the provision of the replacement part: in all three end-to-end trials, the part was produced autonomously and completed 57 to 73 minutes before the failure of the gripper, early enough to have been installed. The installation of the part remains a human task. The extent of the bridge is determined less by the AdM integration, whose orchestration delay is negligible, than by the calibration of the decision to the lead time and to the error of the prediction, which, with thresholds calibrated on a single life cycle and point estimates without uncertainty, was sufficient on these runs, but has not been shown to be reliable across runs.

The second gap (G2) concerned the connection between the prescription of a DT and a decentralized manufacturing command: the reviewed twins mostly represent a single asset, and their virtual-to-physical connection ends at a work order, a schedule, or a control parameter, through centralized information systems. The twin realized in this dissertation is scoped at the level of the system, encompassing the monitored assets and the AdM node, and its virtual-to-physical connection reaches the manufacturing resource directly, through the distributed control layer.

- **RQ#2** — *To what extent do DT technologies ease the integration of PdM, PsM logic, and the autonomous triggering of decentralized manufacturing commands?* **To a large extent, provided that the twin is scoped at the level of the system rather than of a single asset.** Structuring the system as a system-level twin, composed of one instance per asset sharing the AdM node, made it possible to integrate the prognostic model, the prescriptive logic, and the manufacturing command in a single loop, in which the command is issued automatically and reaches the printer in a fraction of a second, without passing through centralized information systems. The ease of this integration is qualified in three respects. The twin updates its state but not its model, so it does not meet the stricter definitions that require

the model itself to evolve with the data (Section 2.5.6); the prognostic model could not be executed within the runtime hosting the twin, and had to be bridged to it through a separate process; and the coordination of the shared manufacturing resource between instances falls back on the operator when the printer is busy.

The third gap (G3) concerned the use of distributed control for maintenance workflows: in the reviewed literature, IEC 61499 applications respond to detected faults by reconfiguring the control software, prognostic tasks are moved out of the control layer, and maintenance decisions are coordinated by platforms outside it. In this dissertation, the prognostic estimate and the prescriptive decision are produced on the edge nodes of an IEC 61499 architecture, and the resulting action is carried through it to the manufacturing resource.

- **RQ#3** — *To what extent can a distributed IEC 61499 architecture support AM workflows?* **It can support them for decision-support tasks at prototype scale, within the constraints of the runtime used.** The IEC 61499 architecture, implemented with DINASORE on nine Raspberry Pi devices, supported the complete workflow: the decentralized coordination of the line, the streaming of sensor data, the estimation of health, the prescriptive decision, and the execution of print orders, all without failures in the trials conducted. The support is qualified, however, by the runtime. The native access to Python ML libraries that motivated the choice of DINASORE was bounded in practice by the interpreter versions it supports, so the models ran in separate processes on the same nodes, whose memory was nearly exhausted; the runtime provides no real-time guarantees; and the results were obtained with a single runtime, on a laboratory testbed. An IEC 61499 architecture can therefore support such workflows for decision-support tasks at prototype scale, while their industrial deployment would depend on runtimes combining access to ML models with the maturity and timing guarantees discussed in Section 4.3.

Taken together, these answers indicate that the three links reviewed separately in Chapter 3, and never found combined in the literature searched (Section 3.6.1), can be integrated into a single, operating maintenance loop on commodity hardware, and that the main obstacle to its reliable operation lies not in the integration itself, but in the timing of the decision under uncertainty.

7.6 Limitations

The technical limitations of the implementation, such as those of the runtime, the sensors, and the communication, are listed in Section 5.7. This section addresses the limitations of the methodology followed to develop, implement, and validate the system, which bound the generality of the conclusions drawn above:

- **Development** — The system was developed and demonstrated on a single case study: one laboratory testbed, with one printer and one type of replacement part, the gripper, on hobbyist-grade equipment whose failure mechanisms differ from those of industrial machinery. Scenarios with several components, failure modes, or printers were not addressed. The prescriptive logic was deliberately limited to fixed thresholds, one per sensor, without the

evaluation of alternative actions or their costs, and the prognostic model is not updated with new data. The orchestration of the AdM node was likewise kept simple: the print queue is not optimized, and multiple simultaneous production requests are not planned, but serialized and released by the operator;

- **Implementation** — The prognostic models were developed from a single run-to-failure life cycle per sensor, obtained through artificially accelerated degradation, with health labels constructed under an assumed exponential shape and a chronologically weighted clustering, whose first predicting time preceded the first introduced fault. From the training data alone, whether the models track the condition of the grippers or the elapsed operating time cannot therefore be established; the end-to-end trials indicate the former, but with one run per arm (Section 7.4). The thresholds were calibrated on the same life cycle: they account for the full time to act, including installation, and for the error observed on test data, but not for the uncertainty of each estimate, and their margins were derived from an error that the evaluation with whole sessions held out showed to be optimistic;

- **Validation** — The end-to-end validation comprised one new degradation run per arm, which demonstrates the complete loop but does not allow the variability of the timing of the decision to be characterized statistically. The evaluation with whole sessions held out is closer to a new run than the block split, but its held-out sessions remain surrounded by training sessions of the same life cycle. The triggering test sent the orders from a test script on the PdM nodes, rather than from their prescriptive logic, and bypassed the print queue, the monitoring of control and acquisition lasted one hour without injected failures, and the system was not compared with a centralized or recommendation-only baseline, so the benefits of the distributed design are argued rather than demonstrated comparatively. Finally, the printed parts were not evaluated for dimensional accuracy, their fit being verified only by installing them after the failure and operating the arms; the time allowed for their installation was taken from a preliminary test rather than timed in the trials; and the sustainability benefits associated with on-demand production were not measured.

The comparisons drawn in this chapter with the reviewed literature also inherit the limitations of the review described in Section 3.1.3, which was conducted by a single reviewer, in three databases, without snowballing.

Chapter 8

Conclusions

I5.0 calls for industrial maintenance that is not only efficient, but also human-centric, sustainable, and resilient (Section 2.1). While PdM answers the question of when a failure will occur, and PsM the question of what to do about it, neither closes the physical side of the maintenance loop when the replacement part must first be obtained. This dissertation set out to close that loop autonomously, by connecting the prognostic and prescriptive logic of a DT to a distributed, IEC 61499-compliant CPPS that produces the replacement part on demand by AdM. Chapter 4 proposed such a system, Chapter 5 implemented it on a laboratory testbed, Chapter 6 validated it, first part by part and then end to end, and Chapter 7 discussed the results against the gaps identified in Chapter 3.

This final chapter closes the dissertation. Section 8.1 summarizes the work, Section 8.2 assesses the extent to which its objectives were met, Section 8.3 states its contributions, Section 8.4 discusses its practical impact, and Section 8.5 outlines the future work it opens.

8.1 Summary of the Dissertation

Chapter 2 established the concepts on which the work builds, and the definitions adopted for autonomous maintenance, PsM, and the DT. Chapter 3 reviewed the literature on each link of the maintenance loop and found that, although each had been developed in depth, they had been developed largely in isolation: prognostic decisions end in recommendations or purchase orders, DTs connect to maintenance through centralized information systems, distributed control architectures respond to detected faults by reconfiguring software, and the production of spare parts by AdM is triggered by demand and decided by planners. No study combining a DT, maintenance, IEC 61499, and AdM was found.

Chapter 4 proposed a system in which a DT, scoped at the level of the system and realized as one instance per monitored asset, estimates the health of each asset and, when the RUL derived from its HI falls below a threshold set according to the time needed to produce and install the replacement part, triggers its production on a shared AdM node, all implemented as IEC 61499 FBs running on DINASORE and communicating over MQTT. Chapter 5 implemented this system on a testbed of three robotic arms, a 3D printer, nine Raspberry Pi devices, and a Raspberry Pi Pico W, developing three prognostic models from an accelerated degradation run for which health labels had to be constructed, and calibrating the thresholds of the decision to the measured time

needed to produce and install the part, with a margin for the error of the models. Chapter 6 validated the system: the arms completed all production cycles with coordination delays of the order of ten milliseconds, the sensor data reached the prognostic nodes without packet loss, the models reproduced their labels with errors of a few percent of the scale of the HI, which increased two- to threefold when whole sessions were held out, the AdM node executed all 90 print orders sent to it, and, in three end-to-end trials on new degradation runs, the system issued the print order on its own, and the printer started printing about 210 ms after it was published, completing the part 57 to 73 minutes before the failure of the grippers. Chapter 7 concluded that the loop between prediction and the provision of the replacement part can be closed autonomously and with negligible orchestration delay, and that the main obstacle to its reliable operation is the timing of the decision under uncertainty, rather than the integration itself.

8.2 Fulfillment of the Objectives

The primary objective, established in Section 1.3, was to architect, deploy, and validate an integrated CPPS framework in which DT-hosted PsM logic triggers on-demand AdM through distributed control. This objective was met at prototype scale, as follows:

- **O1** was met in Chapter 4, with the design of an architecture that integrates health estimation, prescriptive logic, distributed control, and AdM into a single maintenance loop, and whose design choices are justified against the conclusions of the literature review;
- **O2** was met in Chapters 4 and 5, with the implementation of the distributed control layer on DINASORE. It was met with a qualification: because of the interpreter versions supported by the runtime, the prognostic models could not run within its FBs, and were executed in separate processes on the same nodes, bridged to the FBs through files;
- **O3** was met in Sections 4.4 and 5.2, with the integration of a 3D printer as an on-demand production node, triggered directly by the prescriptive logic, and validated in Sections 6.4 and 6.5;
- **O4** was met in Chapters 5 and 6, with the deployment and end-to-end validation of the complete system, in which the decision to produce the replacement part was taken and executed without centralized coordination or human decision. Human intervention remains in the installation of the part, and in the release of print orders when the printer is busy.

The objectives were therefore fulfilled within the scope set for this dissertation: a validated demonstration of feasibility at prototype scale, rather than an industrial deployment or a comparative demonstration against centralized alternatives. In the end-to-end trials, the parts were completed in time to be installed before the failure; however, with one run per arm, the variability of the timing of the decision was not characterized, and part of the margin came from grippers that degraded more slowly than in the training run, so the feasibility of the loop has been demonstrated, but not yet its reliability. This point shapes the future work of Section 8.5.

8.3 Contributions

The contribution of this dissertation is not the testbed itself, but the knowledge obtained by building and validating it, most of which does not depend on the specific hardware used. Part of it has been accepted for publication in a peer-reviewed journal [17].

- **A working integration of four previously separate research streams** — To the extent of the literature reviewed, no study had combined a prognostics-driven decision, a DT, distributed control, and AdM (Section 3.6.1). This dissertation instantiates that combination as a single, operating system on commodity hardware, and validates it end to end on new degradation runs, showing that the integration is not only conceptually coherent, but operable;

- **A quantified account of the digital-to-physical trigger** — None of the reviewed studies reported the time between a prescriptive decision and the start of the physical production of a replacement part. This dissertation measured it in the end-to-end trials, at 207 to 215 ms from the publication of the order to the start of the print, broken down into the transmission of the order, its processing by the AdM node, and the start of the print, complemented by a round trip of less than 400 ms, in the triggering tests, between an order and the confirmation of the start of the print. It also showed that this delay is negligible compared with the build time of the part, which therefore determines the time needed to act;

- **Evidence on the timing of prognostics-driven production** — This dissertation set the trigger in time rather than in health, as an estimated RUL that covers the production and installation of the part plus a margin taken from the error of the model, and tested it on new degradation runs. In the three trials, the thresholds were crossed 4 h 32 min to 4 h 48 min before the failure, against target RULs of 3 h 56 min to 4 h 38 min, and the part was completed 57 to 73 minutes before it, against intended margins of 20 to 62 minutes plus the installation, falling short of the intended margin by 5 minutes for one arm, and owing part of its size to grippers that degraded more slowly than in the training run, at whose pace it would barely have covered the installation. Together with the evaluation with whole sessions held out, which showed the error on unseen data to be several times larger than on the life cycle used for training, this provides empirical support, from a closed-loop system, for the argument made in the literature that prognostic decisions must account for the time to act and for the uncertainty of the prediction, and identifies the calibration of the decision, rather than the integration, as the critical factor for such systems;

- **An account of the use of an IEC 61499 runtime for ML-based maintenance** — The comparison of Section 4.3 and the implementation of Chapter 5 document the trade-offs faced when an IEC 61499 system must host ML models: the absence of native Python support in established runtimes, the loss of real-time guarantees in the Python-based runtime that provides it, and, in practice, the constraint imposed by the interpreter versions that runtime supports. This account is relevant to any CPPS facing the same choice, independently of the maintenance context in which it was obtained;

- **A documented procedure for prognostics without failure labels, and its pitfalls** — The construction of health stages and indicators described in Section 5.4.2 applies the guidance of Berghout and Benbouzid [36] to a new context, the vibration of 3D-printed robotic grippers, and documents not only the procedure but also its main pitfalls: labels built with strong chronological constraints may encode elapsed time as much as condition, and a random split into blocks of the same life cycle yields errors two to three times lower than those obtained with whole sessions held out, both of which must be taken into account when interpreting the resulting models;

- **A low-cost, open reference implementation** — The system relies on open-source software, from the runtime to the printer management server, and on commodity devices, showing that an autonomous, closed-loop maintenance system of this kind can be built and validated without a proprietary industrial platform, which makes it replicable by other research groups and by companies with limited resources. Its code, models, and scripts are kept in the repository of this work (Section 1.4).

8.4 Practical Impact

The practical impact of this dissertation is that of a validated laboratory prototype, not of a deployed industrial system. The testbed relies on hobbyist-grade equipment, a single replacement part, and accelerated degradation, none of which is representative, on its own, of a production shop floor, and the timing of the decision has been tested on a single run per arm, whose margins, although enough to install the part in time, were partly owed to grippers that degraded more slowly than in the training run. Its impact today is therefore that of a demonstrated capability.

That capability addresses the concerns that motivated the work. By producing the replacement part within the system, triggered by the prognostic decision, it points towards the reduction of repair times and of the inventory associated with spare parts that motivates the concept of digital spare parts (Section 2.8.3), although these benefits were not measured here. By keeping the human operator in a supervisory role, able to cancel print orders, and in the physical installation of the part, while automating the monitoring, the analysis, the decision, and the production of the part, it illustrates the human-centric allocation of tasks advocated by I5.0 [23]. By distributing control across independent devices, without a central coordinator or a central broker, it avoids the single point of failure of a central coordinator, although the shared printer remains a single resource and the behavior of the system under device failures was not tested.

Its most immediate impact is as a starting point for further work. Because the system is built from open-source software and low-cost devices, it can be replicated, adapted, or extended at comparatively little cost, including by small and medium-sized enterprises that cannot afford a dedicated industrial pilot. Translating it into measured impact on a production line depends on the future work outlined below.

8.5 Future Work

The limitations discussed in Section 7.6 define the boundary between what this dissertation validated and what remains open. The following directions address them, the first being the most pressing:

- **Uncertainty-aware decisions** — The thresholds already account for the full time needed to act, including the installation of the part, and for the error observed on test data. Deriving the margin from the error expected on unseen degradation, as estimated with whole sessions or life cycles held out, or replacing it with a quantified uncertainty of each estimate, for instance through Monte Carlo dropout, ensembles, or quantile regression, would turn the availability of the part before failure into a reliable margin for its installation. Since the threshold is a parameter of the architecture, this requires no change to the rest of the system;
- **More and more varied life cycles** — Collecting several run-to-failure life cycles per component, with different durations and with faults introduced at different moments, would allow the models to be evaluated on unseen life cycles, the variability of the timing of the decision to be characterized statistically, and the question of whether the models track condition or elapsed time to be settled. Updating the models with new data, rather than keeping them fixed, would also bring the twin closer to the stricter definitions discussed in Section 2.5.6;
- **Broader coverage and resource planning** — Extending the system to several components, failure modes, part types, and printers would require optimizing the print queue and planning simultaneous production requests, which are currently serialized and released by the operator;
- **Richer prescriptive logic** — Replacing the single threshold with a prescriptive layer that weighs several alternative actions, such as printing, ordering, or adjusting the operation of the equipment, and their costs, would test whether the architecture continues to support PsM beyond the deliberately narrow scope adopted here;
- **Comparative and industrial validation** — Comparing the system with a centralized baseline and with a recommendation-only system, and repeating the validation on industrial equipment under naturally occurring degradation, would move the case for the proposed architecture from argument to demonstration;
- **Physical and sustainability outcomes** — Evaluating the dimensional accuracy, fitness, and installation of the printed parts, and measuring the energy, material, and lead-time outcomes of on-demand production, would close the physical side of the loop as rigorously as its digital side;
- **Runtime and communication** — Deploying time-critical control on FORTE and prognostics on DINASORE, executing the models within the runtime once it supports the required interpreter versions, and publishing print orders with a delivery guarantee would address the constraints identified in the implementation;

- **Human-centric evaluation** — Studying how operators experience, supervise, and trust a system that automates the maintenance decision while leaving them the installation of the part would complement the technical validation reported here.

Taken together, these directions do not question the central result of this dissertation, that a DT-hosted prescriptive logic can autonomously trigger the on-demand production of a replacement part across a distributed, IEC 61499-compliant CPPS, and have it ready before the component fails, as it did in all three end-to-end trials. They chart the path from this result towards a system that does so with a reliable margin, on industrial equipment, and in comparison with the alternatives. In closing, even partially, the gap between digital prediction and the physical provision of the replacement part, this dissertation offers a concrete, working answer to a problem that the reviewed literature had, until now, addressed only in separate pieces.

Appendix A

Literature Review Research Queries

This appendix presents the full list of research queries, shown in Table A.1 executed when performing the literature review of this dissertation, reported on Chapter 3.

Table A.1 – Complete list of literature review search queries

Theme	Database	Query
T0	IEEE Xplore	(("Document Title":"digital twin" OR "Abstract":"digital twin" OR "Index Terms":"digital twin") AND ("Document Title":maintenance OR "Abstract":maintenance OR "Index Terms":maintenance) AND ("Document Title":"IEC 61499" OR "Abstract":"IEC 61499" OR "Index Terms":"IEC 61499") AND ("Document Title":"additive manufactur*" OR "Document Title":"3D print*" OR "Abstract":"additive manufactur*" OR "Abstract":"3D print*" OR "Index Terms":"additive manufactur*" OR "Index Terms":"3D print*"))
T0	ScienceDirect	Title abstract keywords "digital twin" AND maintenance AND "IEC 61499" AND ("additive manufacturing" OR "3d printing")
T0	Scopus	TITLE-ABS-KEY ("digital twin*" AND maintenance AND "IEC 61499" AND ("additive manufactur*" OR "3D print*"))
T1A	IEEE Xplore	((("Document Title":"remaining useful life" OR "Document Title":prognos* OR "Document Title":"predictive maintenance" OR "Abstract":"remaining useful life" OR "Abstract":prognos* OR "Abstract":"predictive maintenance" OR "Index Terms":"remaining useful life" OR "Index Terms":prognos* OR "Index Terms":"predictive maintenance") AND ("Document Title":"spare part*" OR "Document Title":"lead time" OR "Abstract":"spare part*" OR "Abstract":"lead time" OR "Index Terms":"spare part*" OR "Index Terms":"lead time")))
T1A	ScienceDirect	Title, abstract, keywords: ("remaining useful life" OR prognosis OR prognostics OR "predictive maintenance") AND ("spare part" OR "lead time")
T1A	Scopus	TITLE-ABS-KEY (("remaining useful life" OR prognos* OR "predictive maintenance") AND ("spare part*" OR "lead time")) AND LANGUAGE (english) AND SUBJAREA (ENGI OR DECI) AND DOCTYPE (ar OR cp) AND PUBYEAR > 2014 AND PUBYEAR < 2028
T1B	IEEE Xplore	(("Document Title":"prescriptive maintenance" OR "Abstract":"prescriptive maintenance" OR "Index Terms":"prescriptive maintenance") AND ("Document Title":"remaining useful life" OR "Document Title":prognos* OR "Document Title":"predictive maintenance" OR "Abstract":"remaining useful life" OR "Abstract":prognos* OR "Abstract":"predictive maintenance" OR "Index Terms":"remaining useful life" OR "Index Terms":prognos* OR "Index Terms":"predictive maintenance"))

Continued on next page

Table A.1 – continued from previous page

Theme	Database	Query
T1B	ScienceDirect	Title, abstract, keywords: "prescriptive maintenance" AND ("remaining useful life" OR prognosis OR prognostics OR "predictive maintenance")
T1B	Scopus	TITLE-ABS-KEY("prescriptive maintenance" AND ("remaining useful life" OR prognos* OR "predictive maintenance")) AND PUBYEAR > 2014 AND PUBYEAR < 2027 AND PUBYEAR > 2016 AND PUBYEAR < 2027 AND (LIMIT-TO (SUBJAREA,"ENGI") OR LIMIT-TO (SUBJAREA,"DECI") OR LIMIT-TO (SUBJAREA,"COMP")) AND (LIMIT-TO (DOCTYPE,"ar") OR LIMIT-TO (DOCTYPE,"cp")) AND (LIMIT-TO (LANGUAGE,"English"))
T2A	IEEE Xplore	("Document Title":"digital twin" AND (("Document Title":"predictive maintenance" OR "Document Title":"prescriptive maintenance" OR "Document Title":"remaining useful life" OR "Document Title":prognos* OR "Abstract":"predictive maintenance" OR "Abstract":"prescriptive maintenance" OR "Abstract":"remaining useful life" OR "Abstract":prognos* OR "Index Terms":"predictive maintenance" OR "Index Terms":"prescriptive maintenance" OR "Index Terms":"remaining useful life" OR "Index Terms":prognos*) AND ("Document Title":decision OR "Document Title":"closed loop" OR "Document Title":autonomous OR "Document Title":prescriptive OR "Abstract":decision OR "Abstract":"closed loop" OR "Abstract":autonomous OR "Abstract":prescriptive OR "Index Terms":decision OR "Index Terms":"closed loop" OR "Index Terms":autonomous OR "Index Terms":prescriptive) AND ("Document Title":manufacturing OR "Document Title":"production system" OR "Document Title":"shop floor" OR "Document Title":industr* OR "Abstract":manufacturing OR "Abstract":"production system" OR "Abstract":"shop floor" OR "Abstract":industr* OR "Index Terms":manufacturing OR "Index Terms":"production system" OR "Index Terms":"shop floor" OR "Index Terms":industr*)))
T2A	ScienceDirect	Title "digital twin", title-abstract-keywords ("predictive maintenance" OR "prescriptive maintenance") AND (decision OR "closed loop" OR autonomous) AND (manufacturing OR "production system" OR industrial)
T2A	ScienceDirect	Title "digital twin" title-abstract-keywords ("remaining useful life" OR prognostics) AND (decision OR "closed loop" OR autonomous) AND (manufacturing OR "production system" OR industrial)
T2A	Scopus	TITLE ("digital twin*") AND TITLE-ABS-KEY (("predictive maintenance" OR "prescriptive maintenance" OR "remaining useful life" OR prognos*) AND (decision* OR "closed loop" OR "closed-loop" OR autonomous OR prescriptive) AND (manufactur* OR "production system*" OR "shop floor" OR industr*)) AND PUBYEAR > 2017 AND PUBYEAR < 2028 AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (LANGUAGE , "english"))
T2B	IEEE Xplore	("Document Title":"digital twin*" AND "All Metadata":maintenance AND ("All Metadata":mes OR "All Metadata":erp OR "All Metadata":cmms OR "All Metadata":"work order*" OR "All Metadata":"spare part*" OR "All Metadata":"production planning"))
T2B	ScienceDirect	Title "digital twin", Title, abstract, keywords: maintenance AND (mes OR erp OR cmms OR "work order" OR "spare part" OR "production planning")

Continued on next page

Table A.1 – continued from previous page

Theme	Database	Query
T2B	Scopus	TITLE-ABS-KEY ("digital twin*" AND maintenance AND (mes OR erp OR cmms OR "work order*" OR "spare part*" OR "production planning")) AND PUBYEAR > 2017 AND PUBYEAR < 2028 AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (LANGUAGE , "english"))
T3A	IEEE Xplore	((("Document Title":"IEC 61499" OR "Abstract":"IEC 61499" OR "Index Terms":"IEC 61499") AND ("Document Title":maintenance OR "Document Title":fault OR "Document Title":failure OR "Document Title":reconfigur* OR "Document Title":"digital twin" OR "Document Title":prognos* OR "Abstract":maintenance OR "Abstract":fault OR "Abstract":failure OR "Abstract":reconfigur* OR "Abstract":"digital twin" OR "Abstract":prognos* OR "Index Terms":maintenance OR "Index Terms":fault OR "Index Terms":failure OR "Index Terms":reconfigur* OR "Index Terms":"digital twin" OR "Index Terms":prognos*)))
T3A	ScienceDirect	Title, abstract, keywords: "IEC 61499" AND (maintenance OR fault OR failure OR reconfiguration OR reconfigurable OR "digital twin" OR prognosis OR prognostics)
T3A	Scopus	TITLE-ABS-KEY ("IEC 61499" AND (maintenance OR fault* OR failure* OR reconfigur* OR "digital twin*" OR prognos*)) AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "DECI")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English")) AND PUBYEAR > 2014 AND PUBYEAR < 2027
T3B	IEEE Xplore	(("Document Title":"IEC 61499" OR "Abstract":"IEC 61499" OR "Index Terms":"IEC 61499") AND ("Document Title":"machine learning" OR "Document Title":"artificial intelligence" OR "Document Title":"neural network*" OR "Document Title":"deep learning" OR "Document Title":"data analytics" OR "Abstract":"machine learning" OR "Abstract":"artificial intelligence" OR "Abstract":"neural network*" OR "Abstract":"deep learning" OR "Abstract":"data analytics" OR "Index Terms":"machine learning" OR "Index Terms":"artificial intelligence" OR "Index Terms":"neural network*" OR "Index Terms":"deep learning" OR "Index Terms":"data analytics"))
T3B	ScienceDirect	Title, abstract, keywords: "IEC 61499" AND ("machine learning" OR "artificial intelligence" OR "neural network" OR "deep learning" OR "data analytics")
T3B	Scopus	TITLE-ABS-KEY ("IEC 61499" AND ("machine learning" OR "artificial intelligence" OR "neural network*" OR "deep learning" OR "data analytics")) AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English")) AND PUBYEAR > 2014 AND PUBYEAR < 2026

Continued on next page

Table A.1 – continued from previous page

Theme	Database	Query
T3C	IEEE Xplore	((("Document Title":"multi-agent" OR "Document Title":multiagent OR "Document Title":holonic OR "Abstract":"multi-agent" OR "Abstract":multiagent OR "Abstract":holonic OR "Index Terms":"multi-agent" OR "Index Terms":multiagent OR "Index Terms":holonic) AND ("Document Title":"predictive maintenance" OR "Document Title":"prescriptive maintenance" OR "Document Title":prognos* OR "Document Title":"remaining useful life" OR "Abstract":"predictive maintenance" OR "Abstract":"prescriptive maintenance" OR "Abstract":prognos* OR "Abstract":"remaining useful life" OR "Index Terms":"predictive maintenance" OR "Index Terms":"prescriptive maintenance" OR "Index Terms":prognos* OR "Index Terms":"remaining useful life") AND ("Document Title":manufacturing OR "Document Title":"production system" OR "Document Title":industr* OR "Abstract":manufacturing OR "Abstract":"production system" OR "Abstract":industr* OR "Index Terms":manufacturing OR "Index Terms":"production system" OR "Index Terms":industr*)))
T3C	ScienceDirect	Title, abstract, keywords: ("multi-agent" OR multiagent OR holonic) AND ("predictive maintenance" OR "prescriptive maintenance" OR prognostics OR "remaining useful life") AND (manufacturing OR industrial)
T3C	Scopus	TITLE-ABS-KEY (("multi-agent" OR multiagent OR holonic) AND ("predictive maintenance" OR "prescriptive maintenance" OR prognos* OR "remaining useful life") AND (manufactur* OR "production system*" OR industr*)) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "DECI")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English")) AND PUBYEAR > 2014 AND PUBYEAR < 2028
T4A	IEEE Xplore	((("Document Title":"additive manufacturing" OR "Document Title":"3D print" OR "Document Title":"digital spare part" OR "Document Title":"digital inventory" OR "Document Title":"digital warehouse" OR "Abstract":"additive manufacturing" OR "Abstract":"3D print" OR "Abstract":"digital spare part" OR "Abstract":"digital inventory" OR "Abstract":"digital warehouse" OR "Index Terms":"additive manufacturing" OR "Index Terms":"3D print" OR "Index Terms":"digital spare part" OR "Index Terms":"digital inventory" OR "Index Terms":"digital warehouse") AND ("Document Title":"spare part" OR "Document Title":"replacement part" OR "Abstract":"spare part" OR "Abstract":"replacement part" OR "Index Terms":"spare part" OR "Index Terms":"replacement part") AND ("Document Title":maintenance OR "Document Title":"on-demand" OR "Document Title":"on demand" OR "Document Title":"lead time" OR "Document Title":inventory OR "Document Title":"supply chain" OR "Abstract":maintenance OR "Abstract":"on-demand" OR "Abstract":"on demand" OR "Abstract":"lead time" OR "Abstract":inventory OR "Abstract":"supply chain" OR "Index Terms":maintenance OR "Index Terms":"on-demand" OR "Index Terms":"on demand" OR "Index Terms":"lead time" OR "Index Terms":inventory OR "Index Terms":"supply chain")))
T4A	ScienceDirect	Title, abstract, keywords: ("additive manufacturing" OR "3d printing") AND ("spare part" OR "replacement part") AND (maintenance OR "on demand" OR "lead time" OR inventory OR "supply chain")
T4A	ScienceDirect	Title, abstract, keywords: ("digital spare part" OR "digital inventory" OR "digital warehouse") AND ("spare part" OR "replacement part") AND (maintenance OR "on demand" OR "lead time")
T4A	ScienceDirect	Title, abstract, keywords: ("digital spare part" OR "digital inventory" OR "digital warehouse") AND ("spare part" OR "replacement part") AND (inventory OR "supply chain")

Continued on next page

Table A.1 – continued from previous page

Theme	Database	Query
T4A	Scopus	TITLE-ABS-KEY (("additive manufactur*" OR "3D print*" OR "digital spare part*" OR "digital inventory" OR "digital warehouse") AND ("spare part*" OR "replacement part*") AND (maintenance OR "on-demand" OR "on demand" OR "lead time" OR inventory OR "supply chain")) AND PUBYEAR > 2014 AND PUBYEAR < 2027 AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "DECI") OR LIMIT-TO (SUBJAREA , "BUSI")) AND (LIMIT-TO (LANGUAGE , "English"))
T4B	IEEE Xplore	(("All Metadata":"additive manufactur*" OR "All Metadata":"3d print*") AND ("All Metadata":"digital twin*" OR "All Metadata":"predictive maintenance" OR "All Metadata":"prescriptive maintenance" OR "All Metadata":prognos* OR "All Metadata":"remaining useful life" OR "All Metadata":"condition monitoring") AND ("All Metadata":"spare part*" OR "All Metadata":"replacement part*" OR "All Metadata":repair* OR "All Metadata":maintenance))
T4B	ScienceDirect	Title, abstract, keywords: ("additive manufacturing" OR "3d printing") AND ("predictive maintenance" OR "prescriptive maintenance" OR "condition monitoring") AND ("spare part" OR "replacement part" OR repair OR maintenance)
T4B	ScienceDirect	Title, abstract, keywords: ("additive manufacturing" OR "3d printing") AND ("digital twin" OR "remaining useful life" OR prognostics) AND ("spare part" OR "replacement part" OR repair OR maintenance)
T4B	Scopus	TITLE-ABS-KEY (("additive manufactur*" OR "3d print*") AND ("digital twin*" OR "predictive maintenance" OR "prescriptive maintenance" OR prognos* OR "remaining useful life" OR "condition monitoring") AND ("spare part*" OR "replacement part*" OR repair* OR maintenance)) AND PUBYEAR > 2014 AND PUBYEAR < 2027 AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci") OR LIMIT-TO (subjarea , "busi")) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (LANGUAGE , "english"))
T5A	IEEE Xplore	("self-maintenance" OR "self-maintaining" OR "autonomous maintenance" OR "autonomic maintenance") AND (manufactur* OR "cyber-physical" OR "production system" OR industr*) NOT ("total productive maintenance" OR tpm OR "jishu hozen")
T5A	IEEE Xplore	("self-healing" OR "self-repair*") AND ("predictive maintenance" OR prognos* OR "remaining useful life" OR "condition monitoring") AND (manufactur* OR "production system" OR "cyber-physical")
T5A	ScienceDirect	Title, abstract, keywords: ("self-maintenance" OR "self-maintaining") AND (manufacturing OR "cyber-physical" OR "production system" OR industrial) NOT ("total productive maintenance" OR tpm OR "jishu hozen")
T5A	ScienceDirect	Title, abstract, keywords: ("self-healing" OR "self-repair") AND ("predictive maintenance" OR prognostics OR "remaining useful life" OR "condition monitoring") AND (manufacturing OR "production system" OR "cyber-physical")
T5A	Scopus	TITLE-ABS-KEY (("self-maintenance" OR "self-maintaining" OR "autonomous maintenance" OR "autonomic maintenance") AND (manufactur* OR "cyber-physical" OR "production system*" OR industr*) AND NOT ("total productive maintenance" OR tpm OR "jishu hozen")) AND PUBYEAR > 2014 AND PUBYEAR < 2028 AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (LANGUAGE , "english"))

Continued on next page

Table A.1 – continued from previous page

Theme	Database	Query
T5A	Scopus	TITLE-ABS-KEY (("self-healing" OR "self-repair*") AND ("predictive maintenance" OR prognos* OR "remaining useful life" OR "condition monitoring") AND (manufactur* OR "production system*" OR "cyber-physical")) AND PUBYEAR > 2014 AND PUBYEAR < 2028 AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (LANGUAGE , "english"))
T5B	IEEE Xplore	"digital twin" AND ("additive manufactur*" OR "3d print*") AND (maintenance OR "spare part" OR prognos* OR "remaining useful life")
T5B	ScienceDirect	Title, abstract, keywords: "digital twin" AND ("additive manufacturing" OR "3d printing") AND (maintenance OR "spare part" OR prognostics OR "remaining useful life")
T5B	Scopus	TITLE-ABS-KEY ("digital twin*" AND ("additive manufactur*" OR "3d print*") AND (maintenance OR "spare part*" OR prognos* OR "remaining useful life")) AND PUBYEAR > 2014 AND PUBYEAR < 2028 AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (LANGUAGE , "english"))
T5C	IEEE Xplore	("iec 61499" OR "multi-agent" OR "distributed control" OR "function block") AND ("additive manufactur*" OR "3d print*")
T5C	ScienceDirect	Title, abstract, keywords: ("iec 61499" OR "multi-agent" OR "distributed control" OR "function block") AND ("additive manufacturing" OR "3d printing")
T5C	Scopus	TITLE-ABS-KEY (("iec 61499" OR "multi-agent" OR "distributed control" OR "function block*") AND ("additive manufactur*" OR "3d print*")) AND PUBYEAR > 2014 AND PUBYEAR < 2028 AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (subjarea , "engi") OR LIMIT-TO (subjarea , "comp") OR LIMIT-TO (subjarea , "deci")) AND (LIMIT-TO (LANGUAGE , "english"))

Appendix B

Function Block Applications in 4diac IDE

This appendix presents the IEC 61499 applications of the system described in Chapter 5, as developed in the 4diac IDE. Figure B.1 shows the configuration of the complete system, with its devices and their resources. Furthermore, Figures B.2, B.3, B.4, B.5, B.6, B.7 and B.8 show the FB networks deployed on each device, color-matched to the devices shown in Figure B.1.

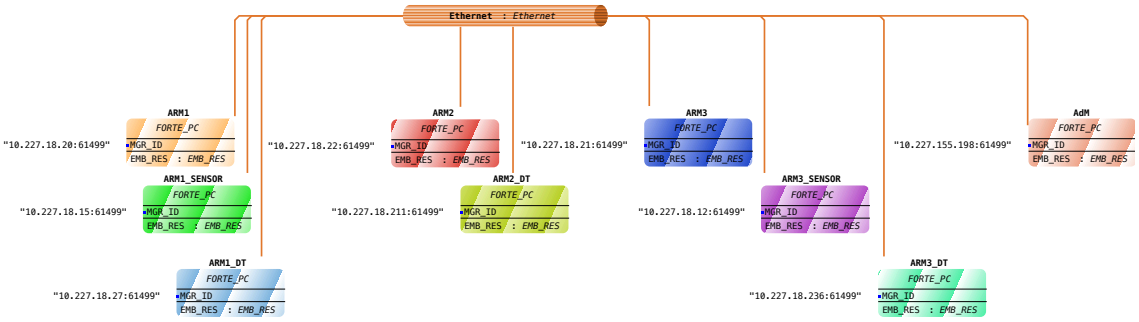


Figure B.1 – Configuration of the complete system

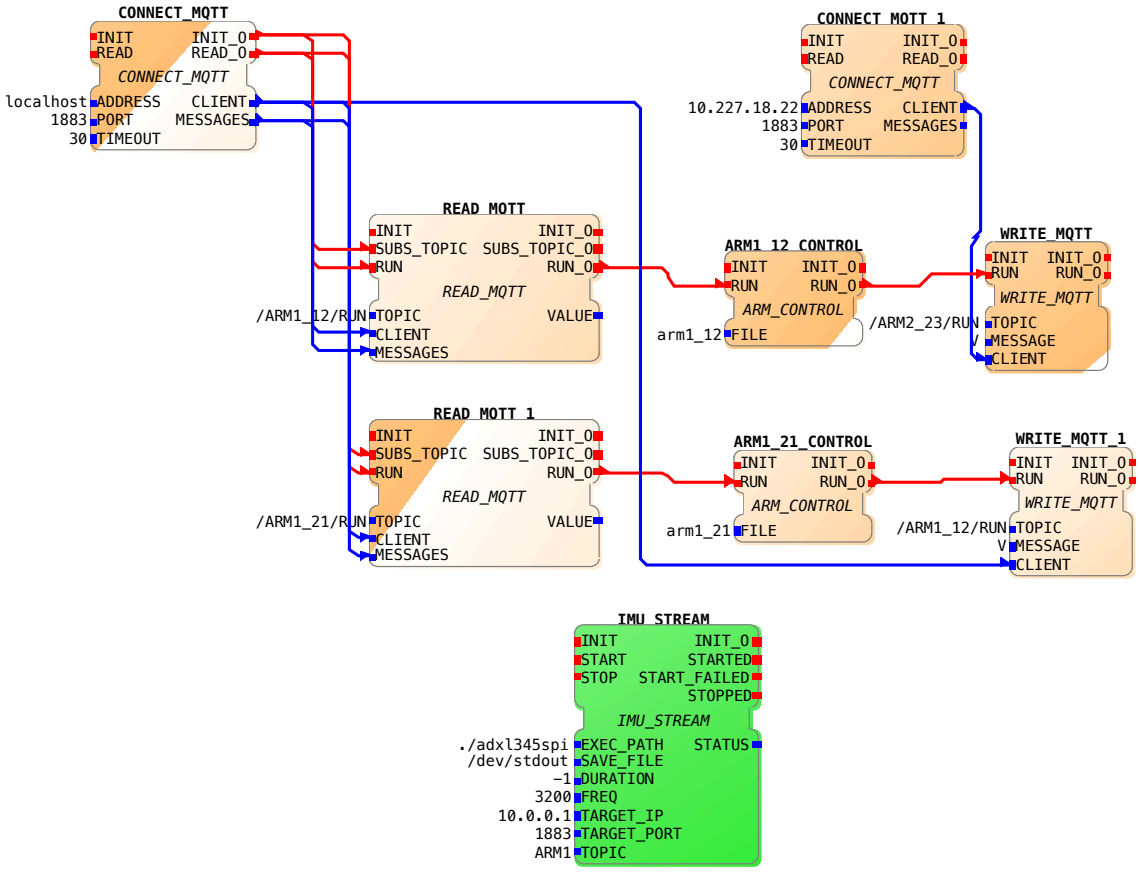


Figure B.2 – FB network of the control and sampling devices of Arm #1

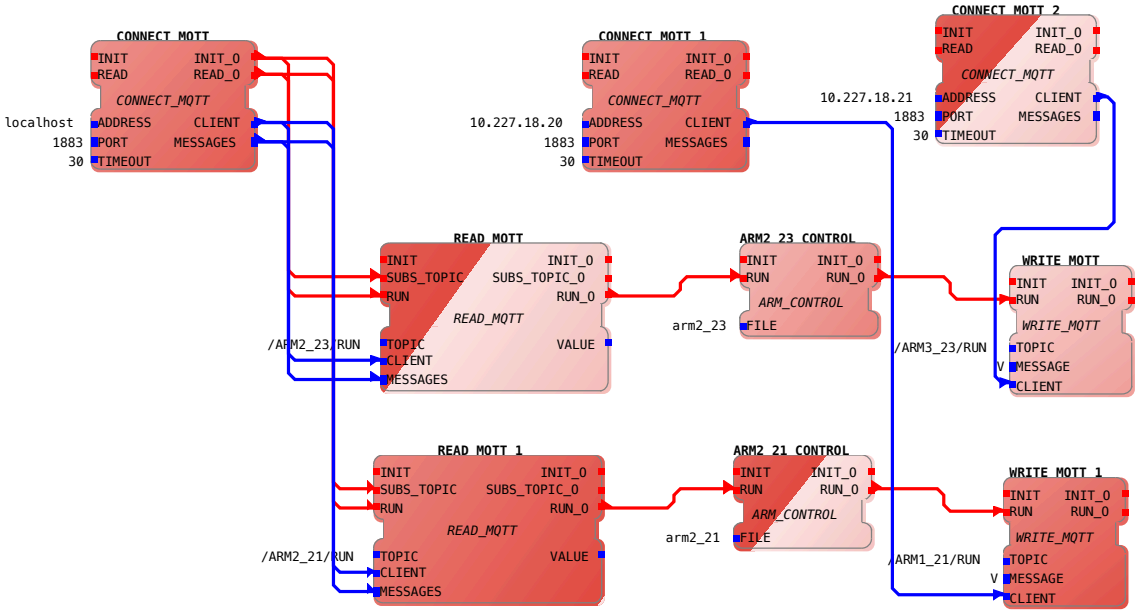


Figure B.3 – FB network of the control device of Arm #2

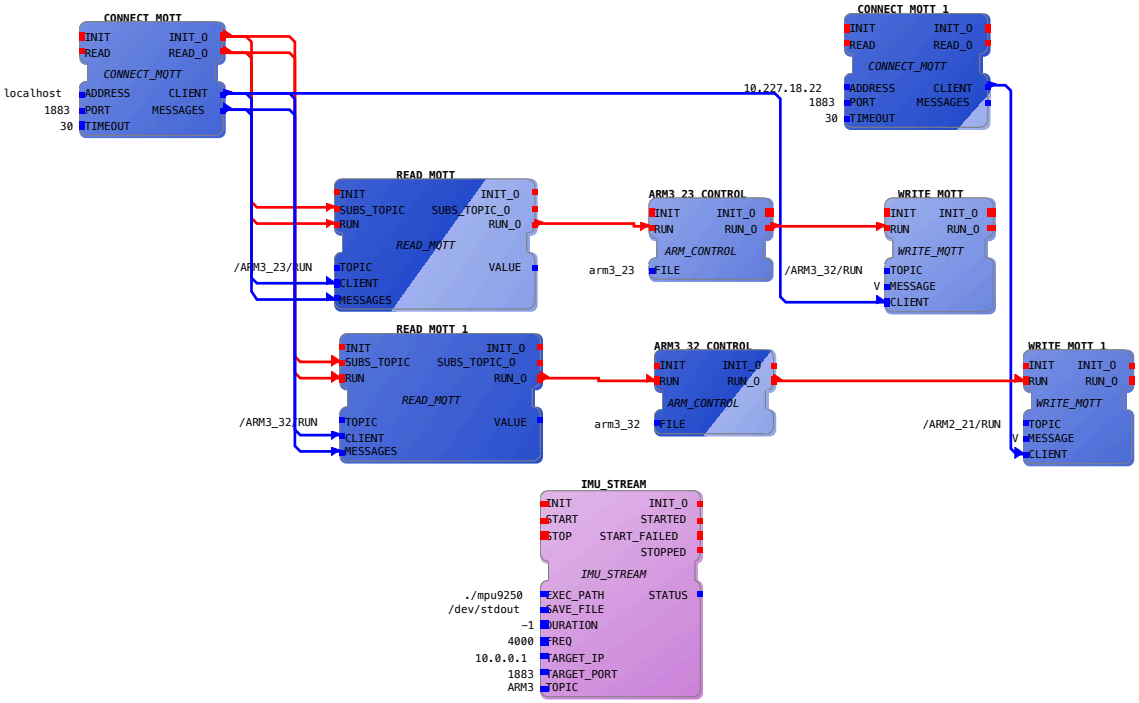


Figure B.4 – FB network of the control and sampling devices of Arm #3

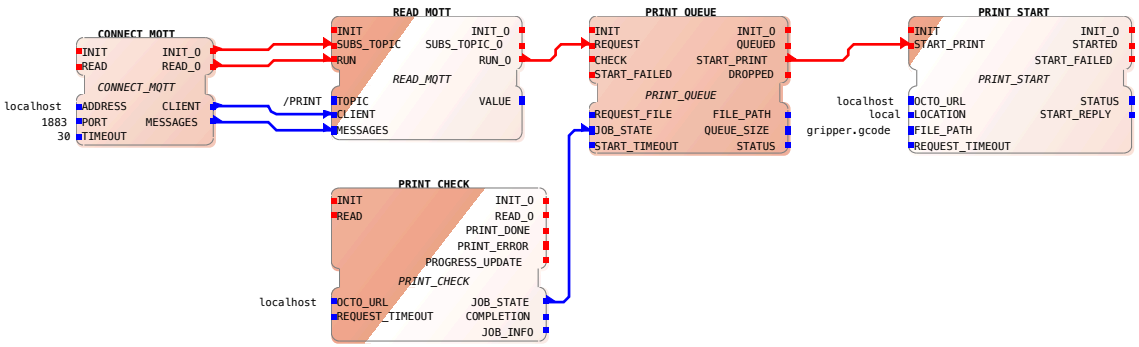


Figure B.5 – FB network of the AdM node

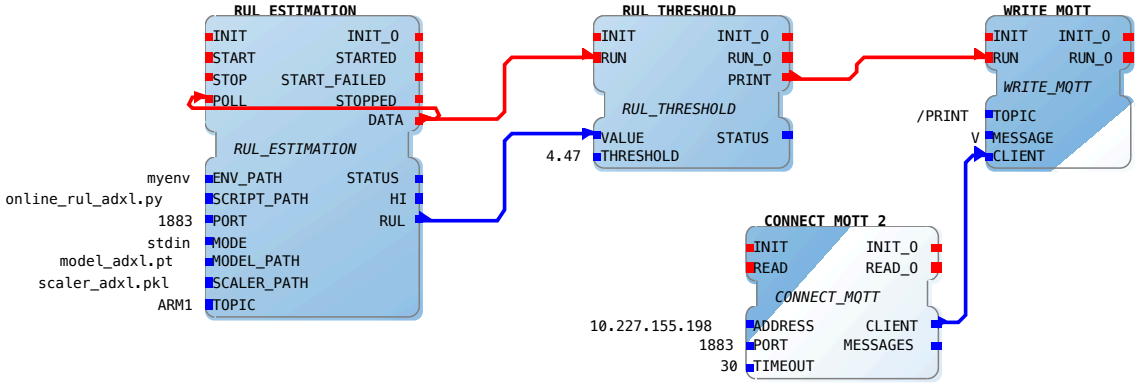


Figure B.6 – FB network of the DT node of Arm #1

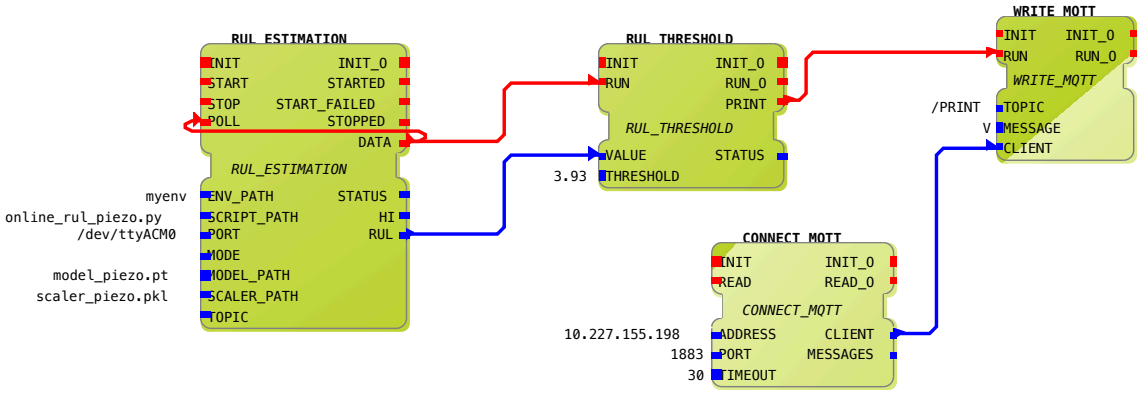


Figure B.7 – FB network of the DT node of Arm #2

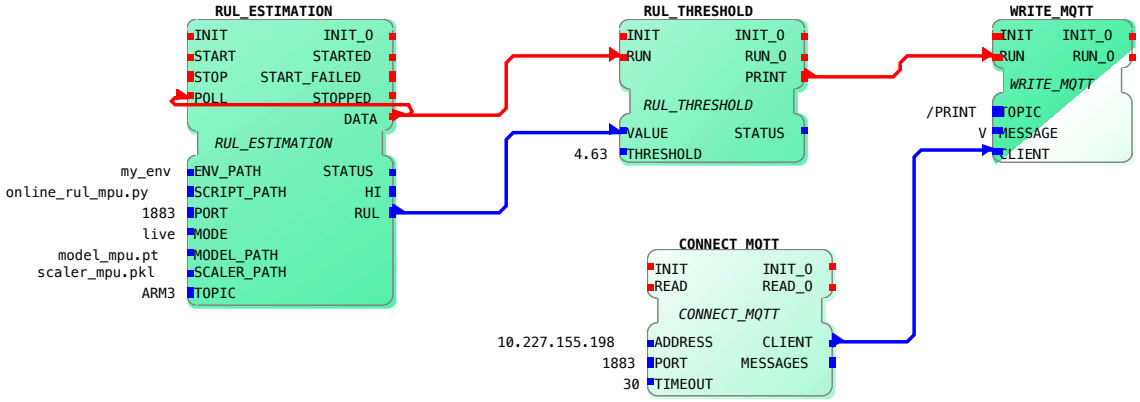


Figure B.8 – FB network of the DT node of Arm #3

Appendix C

Disclosure of Generative Artificial Intelligence Usage

In accordance with FEUP's publishing guidelines regarding the ethical use of AI, the author discloses the use of generative AI tools during the preparation of this dissertation.

Google's Gemini Notebook[1] (powered by Google's Gemini 3.5[2]) was utilized during the literature review process (Chapter 3) as an assistive tool to help extract and synthesize relevant content from the full texts of the screened articles. Additionally, Anthropic's Claude Opus 5.5[3] was used as an assistive tool to help structure the literature review, to provide conceptual explanations of machine learning principles required for this work, and for general copy-editing, including spelling and grammar refinement throughout the drafting of the manuscript.

No AI system is credited as a co-author. All AI-assisted summaries, conceptual explanations, and edits were thoroughly reviewed, verified, and revised by the human author, who assumes full responsibility and accountability for the final content, originality, and integrity of this dissertation.

[1] https://notebook.google

[2] https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-5/

[3] https://claude.com

Glossary

This glossary lists the main terms used in this dissertation, in the meaning in which they are used, together with the sections in which they are introduced or applied.

Accelerated degradation
: Run-to-failure process in which faults are deliberately introduced into a component, so that its complete life cycle can be observed within a practical time (Section 5.3).

Additive manufacturing
: Process of joining materials to make parts from 3D model data, usually layer upon layer; in this work, material extrusion on a desktop printer (Section 2.8).

Advance notice
: In the end-to-end trials, the time between the crossing of the threshold and the complete failure of the gripper, compared with the target RUL (Section 6.1.2).

Autonomous maintenance
: Maintenance in which monitoring, analysis, decision, and, as far as possible, execution are carried out by the production system itself, as opposed to the operator-based meaning of the term in total productive maintenance (Section 2.2.2).

Build time
: Time taken by the printer to produce a part from its sliced file, which can be estimated by the slicer or measured in trial prints (Sections 2.8.3 and 5.2).

Complete failure
: In the testbed, the moment from which a gripper, and therefore its arm, can no longer pick up the piece and place it where intended; taken as the end of life (Sections 5.3 and 6.1.2).

Degree of autonomy
: Characterization of an automated system per class of function (acquisition, analysis, decision, and action) and by the position of the human in or on the loop, rather than by a single label (Sections 2.4.3 and 4.5).

Digital spare part
: Spare part kept, together with its manufacturing data, in digital form, and produced only when it is needed (Section 2.8.3).

Digital twin
: Fit-for-purpose, system-level representation of the maintenance-relevant state of a production

system, with an automated physical-to-virtual connection and a virtual-to-physical connection that may involve humans; in this work, realized as one instance per monitored asset, sharing the additive manufacturing node (Sections 2.5.6 and 4.1).

Distributed control system
Control system whose logic is decomposed into components executed across multiple networked devices (Section 2.6.1).

End of life
Moment at which the monitored item can no longer perform its function, at which the health index reaches 0 (Sections 2.3.1 and 5.4.2).

First predicting time
Start of the unhealthy stage, from which the prediction of the RUL is triggered; in the health labels of this work, the moment at which the health index starts to decrease from 1 (Sections 2.3.2 and 5.4.2).

Function block
In IEC 61499, an encapsulated data structure with a defined algorithm working on that data, executed in response to events (Section 2.6.2).

Health index
Indicator of the health of the monitored item, which may be defined by a degradation function; in this work, a value between 1, healthy, and 0, at the end of life (Sections 2.3.2 and 5.4.2).

Health stage
Stage of the degradation process, obtained by partitioning it according to the trends of the health indicators (Sections 2.3.2 and 5.4.2).

Human in the loop, on the loop
A human in the loop takes part in carrying out a function; a human on the loop supervises it and can intervene, as the operator who can cancel a print order (Sections 2.4.3 and 4.5).

Lead time
Time needed to carry out a maintenance action once it has been decided; for an additively manufactured part, its build time plus its removal, post-processing, and installation (Sections 2.3.5 and 2.8.3).

Margin
Either the safety margin added to the time to act when setting a threshold, derived from the optimistic error of the model (Section 4.2), or, in the end-to-end trials, the time between the completion of the replacement part and the complete failure (Section 6.1.2).

Physical-to-virtual connection
Flow of data by which the state of a digital twin is updated from sensor data (Section 2.5.3).

Predictive maintenance
Condition-based, and therefore preventive, maintenance carried out following a forecast of the degradation of the item (Section 2.2.1).

Prescriptive maintenance
Combination of failure prediction with knowledge of the available actions to determine which action to take and when, delivered as a recommendation or executed automatically (Section 2.4.2).

Publish/subscribe
Communication paradigm in which publishers send messages on topics through a broker, which delivers them to the subscribers of those topics, without publishers and subscribers knowing each other (Section 2.7).

Remaining useful life
Time from the current instant to the end of the useful life; when health is expressed through a health index, the time until it reaches its end-of-life level (Sections 2.3.1 and 2.3.2).

Response time
In the end-to-end trials, the time from the publication of the print order to the transition of the printer to the printing state (Section 6.1.2).

Runtime environment
Software that executes IEC 61499 applications on a device, scheduling the execution of function blocks and managing the flow of data and events (Sections 2.6.2 and 4.3).

Target RUL
Time to act plus the safety margin; the threshold below which the estimated RUL triggers the print order (Sections 4.2 and 5.6).

Time to act
Time needed to produce and install the replacement part, set to 3 h 36 min in the prototype (Sections 4.2 and 5.6).

Timely action
A maintenance action is only useful if triggered while the remaining time still exceeds its lead time, plus a margin for uncertainty (Section 2.3.5).

Virtual-to-physical connection
Flow by which the decisions of a digital twin are returned to the physical system, automatically or through humans (Section 2.5.3).

Whole sessions held out
Evaluation in which models are trained without entire recording sessions and tested on them, closer to the conditions of a new degradation run than the split into blocks (Section 6.3).

References

[1] S. Sahoo and C. Y. Lo, "Smart manufacturing powered by recent technological advancements: A review," *Journal of Manufacturing Systems*, vol. 64, pp. 236–250, Jul. 2022, ISSN: 02786125. DOI: 10.1016/J.JMSY.2022.06.008 (Cited on page 1).

[2] M. I. Khan *et al.*, "Integrating industry 4.0 for enhanced sustainability: Pathways and prospects," *Sustainable Production and Consumption*, vol. 54, pp. 149–189, Mar. 2025, ISSN: 2352-5509. DOI: 10.1016/j.spc.2024.12.012 (Cited on page 1).

[3] A. Ahmed Murtaza, A. Saher, M. Hamza Zafar, S. Kumayl Raza Moosavi, M. Faisal Aftab, and F. Sanfilippo, "Paradigm shift for predictive maintenance and condition monitoring from Industry 4.0 to Industry 5.0: A systematic review, challenges and case study," *Results in Engineering*, vol. 24, p. 102 935, Dec. 2024, ISSN: 2590-1230. DOI: 10.1016/j.rineng.2024.102935 (Cited on pages 1, 6, 7).

[4] T. Bitam, A. Yahiaoui, D. E. Boubiche, R. Martínez-Peláez, H. Toral-Cruz, and P. Velarde-Alvarado, "Artificial Intelligence of Things for Next-Generation Predictive Maintenance," *Sensors*, vol. 25, no. 24, Dec. 2025, ISSN: 14248220. DOI: 10.3390/S25247636 (Cited on page 1).

[5] B. F. Mon, M. Hayajneh, N. A. Ali, F. Ullah, H. Ullah, and S. Alkobaisi, "Digital Twins in the IIoT: Current Practices and Future Directions Toward Industry 5.0," *Computers, Materials and Continua*, vol. 83, no. 3, pp. 3675–3712, 2025, ISSN: 15462226. DOI: 10.32604/CMC.2025.061411 (Cited on pages 1, 17).

[6] A. Benhanifia, Z. B. Cheikh, P. M. Oliveira, A. Valente, and J. Lima, "Systematic review of predictive maintenance practices in the manufacturing sector," *Intelligent Systems with Applications*, vol. 26, p. 200 501, Jun. 2025, ISSN: 2667-3053. DOI: 10.1016/j.iswa.2025.200501 (Cited on page 1).

[7] M. Khan, M. A. Khan, B. Moser, W. Rafique, X. Xiaolong, and D. Wanchun, "AI-Driven Predictive Maintenance in Industrial IoTs: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 13, no. 10, pp. 20 275–20 298, May 2026, ISSN: 2327-4662. DOI: 10.1109/JIOT.2026.3660780 (Cited on page 1).

[8] M. Orošnjak, F. Saretzky, and S. Kedziora, "Prescriptive Maintenance: A Systematic Literature Review and Exploratory Meta-Synthesis," *Applied Sciences (Switzerland)*, vol. 15, no. 15, Aug. 2025, ISSN: 20763417. DOI: 10.3390/APP15158507 (Cited on pages 1, 13, 14, 96).

[9] J. Azeta *et al.*, "Artificial intelligence and robotics in predictive maintenance: A comprehensive review," *Frontiers in Mechanical Engineering*, vol. 11, Jan. 2026, ISSN: 22973079. DOI: 10.3389/FMECH.2025.1722114 (Cited on pages 1, 2).

[10] S. Khan, M. Farnsworth, R. McWilliam, and J. Erkoyuncu, "On the requirements of digital twin-driven autonomous maintenance," *Annual Reviews in Control*, vol. 50, pp. 13–28, Jan. 2020, ISSN: 13675788. DOI: 10.1016/J.ARCONTROL.2020.08.003 (Cited on pages 1, 8, 14).

[11] E. M. Bouyahrouzi, R. Benmimoun, Y. El Kihel, and M. S. Bajjou, "Integrating industry 4.0 technologies and maintenance 4.0 for sustainable manufacturing: A systematic literature review," *International Journal of Advanced Manufacturing Technology*, vol. 140, no. 1-2, pp. 35–61, Sep. 2025, ISSN: 14333015. DOI: 10.1007/S00170-025-16194-3 (Cited on page 2).

[12] V. A. Wankhede and R. Agrawal, "Unveiling the nexus of Industry 4.0 and Lean Six Sigma for sustainable development: Insights from bibliometric and structural topic modeling analysis," en-US, *International Journal of Lean Six Sigma*, vol. 16, no. 6, pp. 1337–1365, Mar. 2025, ISSN: 2040-4166. DOI: 10.1108/IJLSS-06-2024-0128 (Cited on page 2).

[13] M. S. Ahmmed, S. P. Isanaka, and F. Liou, "Promoting Synergies to Improve Manufacturing Efficiency in Industrial Material Processing: A Systematic Review of Industry 4.0 and AI," *Machines*, vol. 12, no. 10, Oct. 2024, ISSN: 20751702. DOI: 10.3390/MACHINES12100681 (Cited on page 2).

[14] H. A. Barrero-Arciniegas, A. A. Bataleblu, E. Rauch, and D. T. Matt, "Sustainable and resilient cyber-physical production systems: A systematic literature review," *Journal of Computational Design and Engineering*, vol. 13, no. 5, pp. 215–249, May 2026, ISSN: 22885048. DOI: 10.1093/JCDE/QWAG043 (Cited on pages 2, 17).

[15] L. Prenzel and S. Steinhorst, "Decentralized Autonomous Architecture for Resilient Cyber-Physical Production Systems," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ISSN: 1558-1101, Feb. 2021, pp. 1300–1303. DOI: 10.23919/DATE51398.2021.9473954 (Cited on pages 2, 17, 24).

[16] P. Leitão, J. Queiroz, and L. Sakurada, "Collective Intelligence in Self-Organized Industrial Cyber-Physical Systems," en, *Electronics*, vol. 11, no. 19, p. 3213, Jan. 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11193213 (Cited on pages 2, 17, 24).

[17] P. Mota and R. Pinto, "An industrial IoT architecture for autonomous maintenance: Integrating digital twins, distributed control, and on-demand additive manufacturing," *Applied Sciences*, 2026, Accepted for publication (Cited on pages 3, 104).

[18] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017, ISSN: 2351-9789. DOI: 10.1016/j.promfg.2017.07.198 (Cited on pages 5, 18, 19, 21).

[19] H. Kagermann, J. Helbig, and W. Wahlster, *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry ; Final Report of the Industrie 4.0 Working Group*. Forschungsunion, 2013, 112 pp., Google-Books-ID: AsfOoAEACAAJ (Cited on page 5).

[20] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, Journal Abbreviation: Proc. Annu. Hawaii Int. Conf. Syst. Sci., vol. 2016-March, IEEE Computer Society, 2016, pp. 3928–3937, ISBN: 978-0-7695-5670-3. DOI: 10.1109/HICSS.2016.488 (Cited on page 5).

[21] C. Bae, E. Choi, and S. Lee, "Technologies, Applications, and Challenges of Digital Twin Across Industries: A Systematic Review of the State-of-the-Art Literature," *IEEE Access*, vol. 13, pp. 152 843–152 869, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3601615 (Cited on pages 6, 18).

[22] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2021.10.006 (Cited on page 6).

[23] Directorate-General for Research and Innovation (European Commission), Breque, Maija, De Nul, Lars, and Petridis, Athanasios, *Industry 5.0: towards a sustainable, human centric and resilient European industry*. Publications Office of the European Union, 2021, ISBN: 978-92-76-25308-2. Accessed: Sep. 27, 2026. [Online]. Available: https://data.europa.eu/doi/10.2777/308407 (Cited on pages 6, 16, 98, 105).

[24] S. Nahavandi, "Industry 5.0-a human-centric solution," *Sustainability (Switzerland)*, vol. 11, no. 16, 2019, ISSN: 2071-1050. DOI: 10.3390/su11164371 (Cited on page 6).

[25] Y. Lu *et al.*, "Outlook on human-centric manufacturing towards industry 5.0," *Journal of Manufacturing Systems*, vol. 62, pp. 612–627, 2022, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2022.02.001 (Cited on page 6).

[26] "Maintenance — maintenance terminology," Brussels, Belgium, Tech. Rep. EN 13306:2017, 2017 (Cited on pages 6, 7, 8).

[27] "International electrotechnical vocabulary (IEV) — part 192: Dependability," Geneva, Switzerland, Tech. Rep. IEC 60050-192:2015, 2015 (Cited on page 7).

[28] I. Ahuja and J. Khamba, "Total productive maintenance: Literature review and directions," *International Journal of Quality and Reliability Management*, vol. 25, no. 7, pp. 709–756, 2008, ISSN: 0265-671X. DOI: 10.1108/02656710810890890 (Cited on pages 7, 8).

[29] I. Errandonea, S. Beltrán, and S. Arrizabalaga, "Digital twin for maintenance: A literature review," *Computers in Industry*, vol. 123, 2020, ISSN: 0166-3615. DOI: 10.1016/j.compind.2020.103316 (Cited on pages 7, 12, 13, 22, 23).

[30] K. E. McKone, R. G. Schroeder, and K. O. Cua, "The impact of total productive maintenance practices on manufacturing performance," *Journal of Operations Management*, vol. 19, no. 1, pp. 39–58, Jan. 1, 2001, ISSN: 0272-6963. DOI: 10.1016/S0272-6963(00)00030-9 (Cited on page 8).

[31] J. Lee, M. Ghaffari, and S. Elmeligy, "Self-maintenance and engineering immune systems: Towards smarter machines and manufacturing systems," *Annual Reviews in Control*, vol. 35, no. 1, pp. 111–122, 2011, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2011.03.007 (Cited on page 8).

[32] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, Jan. 2015, ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001 (Cited on pages 8, 17, 18, 21).

[33] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006, ISSN: 1096-1216. DOI: 10.1016/j.ymssp.2005.09.012 (Cited on pages 9, 10).

[34] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2017.11.016 (Cited on pages 9, 10, 11).

[35] P. Nectoux *et al.*, "PRONOSTIA : An experimental platform for bearings accelerated degradation tests.," in *Conference on Prognostics and Health Management.*, vol. sur CD ROM, Denver, Colorado, United States: IEEE Catalog Number : CPF12PHM-CDR, Jun. 2012, pp. 1–8. Accessed: Sep. 27, 2026. [Online]. Available: https://hal.science/hal-00719503 (Cited on page 9).

[36] T. Berghout and M. Benbouzid, "A systematic guide for predicting remaining useful life with machine learning," *Electronics*, vol. 11, no. 7, p. 1125, Jan. 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11071125 (Cited on pages 9, 10, 11, 69, 70, 71, 72, 105).

[37] D. An, N. H. Kim, and J.-H. Choi, "Practical options for selecting data-driven or physics-based prognostics algorithms with reviews," *Reliability Engineering and System Safety*, vol. 133, pp. 223–236, 2015, ISSN: 0951-8320. DOI: 10.1016/j.ress.2014.09.014 (Cited on page 10).

[38] L. Liao and F. Köttig, "Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 191–207, 2014, ISSN: 0018-9529. DOI: 10.1109/TR.2014.2299152 (Cited on page 10).

[39] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2018.05.050 (Cited on page 10).

[40] C. Ferreira and G. Gonçalves, "Remaining Useful Life prediction and challenges: A literature review on the use of Machine Learning Methods," *Journal of Manufacturing Systems*, vol. 63, pp. 550–562, Apr. 2022, ISSN: 02786125. DOI: 10.1016/J.JMSY.2022.05.010 (Cited on page 10).

[41] S. Sankararaman, "Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction," *Mechanical Systems and Signal Processing*, vol. 52-53, no. 1, pp. 228–247, 2015, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2014.05.029 (Cited on page 11).

[42] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel, "Metrics for offline evaluation of prognostic performance," *International Journal of Prognostics and Health Management*, vol. 1, no. 1, 2010, ISSN: 2153-2648. DOI: 10.36001/ijphm.2010.v1i1.1336 (Cited on page 11).

[43] International Organization for Standardization, "Condition monitoring and diagnostics of machines — prognostics — part 1: General guidelines," International Organization for Standardization, Geneva, CH, Standard ISO 13381-1:2015, 2015. [Online]. Available: https://www.iso.org/standard/51436.html (Cited on page 11).

[44] K. Lepenioti, A. Bousdekis, D. Apostolou, and G. Mentzas, "Prescriptive analytics: Literature review and research challenges," *International Journal of Information Management*, vol. 50, pp. 57–70, 2020, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2019.04.003 (Cited on pages 12, 13, 14).

[45] A. Bousdekis, K. Lepenioti, D. Apostolou, and G. Mentzas, "A review of data-driven decision-making methods for industry 4.0 maintenance applications," *Electronics (Switzerland)*, vol. 10, no. 7, Apr. 2021, ISSN: 20799292. DOI: 10.3390/ELECTRONICS10070828 (Cited on pages 12, 13).

[46] F. Ansari, R. Glawar, and T. Nemeth, "PriMa: A prescriptive maintenance model for cyber-physical production systems," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 4, pp. 482–503, 2019, ISSN: 0951-192X. DOI: 10.1080/0951192X.2019.1571236 (Cited on pages 13, 14).

[47] T. Nemeth, F. Ansari, W. Sihn, B. Haslhofer, and A. Schindler, "PriMa-x: A reference model for realizing prescriptive maintenance and assessing its maturity enhanced by machine learning," in *Procedia CIRP*, Journal Abbreviation: Procedia CIRP, vol. 72, Elsevier B.V., 2018, pp. 1039–1044. DOI: 10.1016/j.procir.2018.03.280 (Cited on pages 13, 14).

[48] K. Matyas, T. Nemeth, K. Kovacs, and R. Glawar, "A procedural approach for realizing prescriptive maintenance planning in manufacturing industries," *CIRP Annals - Manufacturing Technology*, vol. 66, no. 1, pp. 461–464, 2017, ISSN: 0007-8506. DOI: 10.1016/j.cirp.2017.04.007 (Cited on pages 13, 14).

[49] T. Gamer, M. Hoernicke, B. Kloepper, R. Bauer, and A. J. Isaksson, "The autonomous industrial plant – future of process engineering, operations and maintenance," *Journal of Process Control*, vol. 88, pp. 101–110, Apr. 1, 2020, ISSN: 0959-1524. DOI: 10.1016/j.jprocont.2020.01.012 (Cited on pages 14, 15).

[50] M. Vagia, A. A. Transeth, and S. A. Fjerdingen, "A literature review on the levels of automation during the years. what are the different taxonomies that have been proposed?" *Applied Ergonomics*, vol. 53, pp. 190–202, 2016, ISSN: 0003-6870. DOI: 10.1016/j.apergo.2015.09.013 (Cited on pages 14, 15).

[51] S. Nahavandi, "Trusted autonomy between humans and robots: Toward human-on-the-loop in robotics and autonomous systems," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 3, no. 1, pp. 10–17, Jan. 2017, ISSN: 2333-942X. DOI: 10.1109/MSMC.2016.2623867 (Cited on pages 14, 16).

[52] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," in *IFAC-PapersOnLine*, Journal Abbreviation: IFAC-PapersOnLine, vol. 28, 2015, pp. 567–572. DOI: 10.1016/j.ifacol.2015.06.141 (Cited on page 15).

[53] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 30, no. 3, pp. 286–297, 2000, ISSN: 1083-4427. DOI: 10.1109/3468.844354 (Cited on pages 15, 64).

[54] S. Wellsandt *et al.*, "Towards using digital intelligent assistants to put humans in the loop of predictive maintenance systems," in *IFAC-PapersOnLine*, Journal Abbreviation: IFAC-PapersOnLine, vol. 54, Elsevier B.V., 2021, pp. 49–54. DOI: 10.1016/j.ifacol.2021.08.005 (Cited on page 16).

[55] C. Emmanouilidis *et al.*, "Enabling the human in the loop: Linked data and knowledge in industrial cyber-physical systems," *Annual Reviews in Control*, vol. 47, pp. 249–265, 2019, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2019.03.004 (Cited on page 16).

[56] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, Orlando, FL, USA: IEEE, May 2008, pp. 363–369, ISBN: 978-0-7695-3132-8. DOI: 10.1109/ISORC.2008.25 (Cited on page 16).

[57] L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621–641, 2016, ISSN: 00078506. DOI: 10.1016/j.cirp.2016.06.005 (Cited on pages 16, 17).

[58] P. Sajjadi, F. Dinmohammadi, and M. Shafiee, "Machine Learning in Prognostics and Health Management of Cyber-Physical Systems: A Review," *IEEE Access*, vol. 13, pp. 162 320–162 354, 2025, ISSN: 21693536. DOI: 10.1109/ACCESS.2025.3605553 (Cited on page 17).

[59] N. R. Amaliah, B. Tjahjono, and V. Palade, "Human-in-the-Loop XAI for Predictive Maintenance: A Systematic Review of Interactive Systems and Their Effectiveness in Maintenance Decision-Making," *Electronics (Switzerland)*, vol. 14, no. 17, Sep. 2025, ISSN: 20799292. DOI: 10.3390/ELECTRONICS14173384 (Cited on page 17).

[60] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," Mar. 1, 2015 (Cited on pages 18, 20).

[61] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision Support Systems*, vol. 145, 2021, ISSN: 0167-9236. DOI: 10.1016/j.dss.2021.113524 (Cited on pages 18, 19, 20, 22, 23).

[62] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.08.474 (Cited on pages 18, 19, 23).

[63] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and u.s. air force vehicles," in *Collect Tech Pap AIAA ASME ASCE AHS Struct Struct Dyn Mater*, Journal Abbreviation: Collect Tech Pap AIAA ASME ASCE AHS Struct Struct Dyn Mater, 2012, ISBN: 978-1-60086-937-2 (Cited on pages 18, 20).

[64] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2756069 (Cited on pages 18, 21).

[65] G. Shao, "Use case scenarios for digital twin implementation based on ISO 23247," National Institute of Standards and Technology, Gaithersburg, MD, Advanced Manufacturing Series (NIST AMS) 400-2, May 4, 2021, Last Modified: 2022-11-29T09:11-05:00. DOI: 10.6028/NIST.AMS.400-2 (Cited on pages 19, 21, 22).

[66] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, May 2020, ISSN: 17555817. DOI: 10.1016/j.cirpj.2020.02.002 (Cited on pages 19, 20, 23).

[67] G. Shao and M. Helu, "Framework for a digital twin in manufacturing: Scope and requirements," *Manufacturing Letters*, vol. 24, pp. 105–107, Apr. 1, 2020, ISSN: 2213-8463. DOI: 10.1016/j.mfglet.2020.04.004 (Cited on pages 19, 20, 21, 22).

[68] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: Modeling and twinning enabling technologies," *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 354, Dec. 2022, ISSN: 1615-147X, 1615-1488. DOI: 10.1007/s00158-022-03425-4 (Cited on pages 19, 20, 21, 23).

[69] F. Tao, M. Zhang, Y. Liu, and A. Nee, "Digital twin driven prognostics and health management for complex equipment," *CIRP Annals*, vol. 67, no. 1, pp. 169–172, 2018, ISSN: 0007-8506. DOI: 10.1016/j.cirp.2018.04.055 (Cited on page 22).

[70] K. Ding, F. T. Chan, X. Zhang, G. Zhou, and F. Zhang, "Defining a digital twin-based cyber-physical production system for autonomous manufacturing in smart shop floors," *International Journal of Production Research*, vol. 57, no. 20, pp. 6315–6334, Oct. 18, 2019, ISSN: 0020-7543. DOI: 10.1080/00207543.2019.1566661 (Cited on pages 22, 23).

[71] L. Wright and S. Davidson, "How to tell the difference between a model and a digital twin," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, no. 1, 2020, ISSN: 2213-7467. DOI: 10.1186/s40323-020-00147-4 (Cited on page 23).

[72] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2008.09.005 (Cited on page 24).

[73] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 768–781, 2011, ISSN: 1551-3203. DOI: 10.1109/TII.2011.2166785 (Cited on pages 24, 25, 26, 27).

[74] K.-H. John and M. Tiegelkamp, *IEC 61131-3: Programming industrial automation systems: Concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Berlin Heidelberg, 2010, 390 pp., ISBN: 978-3-642-12014-5. DOI: 10.1007/978-3-642-12015-2 (Cited on pages 24, 26).

[75] A. Zoitl and R. Lewis, *Modelling control systems using IEC 61499, 2nd edition*. Institution of Engineering and Technology, 2014, 231 pp., ISBN: 978-1-84919-761-8 (Cited on pages 24, 25, 26).

[76] W. Dai and V. Vyatkin, "Redesign distributed PLC control systems using IEC 61499 function blocks," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 390–401, Apr. 2012, ISSN: 1558-3783. DOI: 10.1109/TASE.2012.2188794 (Cited on pages 24, 25, 26).

[77] K. Thramboulidis, "IEC 61499 function block model: Facts and fallacies," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 7–23, 2009, ISSN: 1932-4529. DOI: 10.1109/MIE.2009.934788 (Cited on pages 25, 26).

[78] A. Zoitl and V. Vyatkin, "IEC 61499 architecture for distributed automation: The "glass half full" view," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 7–22, 2009, ISSN: 1932-4529. DOI: 10.1109/MIE.2009.934789 (Cited on page 25).

[79] T. Strasser, A. Zoitl, J. H. Christensen, and C. Sünder, "Design and execution issues in IEC 61499 distributed automation and control systems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 41, no. 1, pp. 41–51, 2011, ISSN: 1094-6977. DOI: 10.1109/TSMCC.2010.2067210 (Cited on pages 25, 26).

[80] T. Strasser *et al.*, "Framework for distributed industrial automation and control (4diac)," in *IEEE Int. Conf. Ind. Informatics (INDIN)*, Journal Abbreviation: IEEE Int. Conf. Ind. Informatics (INDIN), 2008, pp. 283–288, ISBN: 978-1-4244-2171-8. DOI: 10.1109/INDIN.2008.4618110 (Cited on pages 26, 27, 60).

[81] Eclipse Foundation. "Eclipse 4diac," Accessed: Jul. 22, 2026. [Online]. Available: https://eclipse.dev/4diac/ (Cited on page 27).

[82] Eclipse Foundation, *4diac FORTE*, https://eclipse.dev/4diac/4diac_forte/, Accessed: 2026-07-22, 2026 (Cited on pages 27, 60).

[83] "IEC 62541-1:2025 — OPC Unified Architecture — Part 1: Overview and Concepts," International Electrotechnical Commission, International Standard, 2025 (Cited on pages 27, 28).

[84] OASIS, *Message queuing telemetry transport (mqtt) version 3.1.1*, OASIS Standard, Oct. 2014. Accessed: Sep. 28, 2026. [Online]. Available: http://oasis-open.org (Cited on pages 27, 29).

[85] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003, ISSN: 0360-0300. DOI: 10.1145/857076.857078 (Cited on pages 27, 28).

[86] S. Grüner, J. Pfrommer, and F. Palm, "RESTful industrial communication with OPC UA," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1832–1841, 2016, ISSN: 1551-3203. DOI: 10.1109/TII.2016.2530404 (Cited on pages 28, 29).

[87] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance evaluation of industry 4.0 protocols," in *Proc IEEE Int Conf Ind Technol*, Journal Abbreviation: Proc IEEE Int Conf Ind Technol, vol. 2019-February, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 955–962, ISBN: 978-1-5386-6376-9. DOI: 10.1109/ICIT.2019.8755050 (Cited on pages 28, 29).

[88] International Electrotechnical Commission. "OPC Unified Architecture - Part 14: PubSub," Accessed: Sep. 28, 2026. [Online]. Available: https://iec.ch (Cited on page 28).

[89] D. Silva, L. I. Carvalho, J. Soares, and R. C. Sofia, "A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA," *Applied Sciences (Switzerland)*, vol. 11, no. 11, 2021, ISSN: 2076-3417. DOI: 10.3390/app11114879 (Cited on pages 28, 29).

[90] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext transfer protocol – HTTP/1.0," Internet Engineering Task Force (IETF), RFC 1945, May 1996. Accessed: Jul. 27, 2026. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1945 (Cited on pages 29, 61).

[91] ISO/ASTM International, "Additive manufacturing — general principles — fundamentals and vocabulary," International Organization for Standardization, Geneva, Switzerland, Tech. Rep. ISO/ASTM 52900:2021, Nov. 2021 (Cited on pages 30, 31).

[92] B. N. Turner, R. Strong, and S. A. Gold, "A review of melt extrusion additive manufacturing processes: I. process design and modeling," *Rapid Prototyping Journal*, vol. 20, no. 3, pp. 192–204, 2014, ISSN: 1355-2546. DOI: 10.1108/RPJ-01-2013-0012 (Cited on pages 30, 31).

[93] I. Gibson, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*. Springer International Publishing, 2020, 675 pp., ISBN: 978-3-030-56127-7. DOI: 10.1007/978-3-030-56127-7 (Cited on pages 30, 31).

[94] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Nguyen, and D. Hui, "Additive manufacturing (3d printing): A review of materials, methods, applications and challenges," *Composites Part B: Engineering*, vol. 143, pp. 172–196, 2018, ISSN: 1359-8368. DOI: 10.1016/j.compositesb.2018.02.012 (Cited on page 30).

[95] N. Kretzschmar, S. Chekurov, M. Salmi, and J. Tuomi, "Evaluating the readiness level of additively manufactured digital spare parts: An industrial perspective," *Applied Sciences (Switzerland)*, vol. 8, no. 10, 2018, ISSN: 2076-3417. DOI: 10.3390/app8101837 (Cited on pages 31, 32).

[96] M. Salmi *et al.*, *Digital Spare Parts*. Mar. 1, 2018 (Cited on pages 31, 32).

[97] S. Chekurov, S. Metsä-Kortelainen, M. Salmi, I. Roda, and A. Jussila, "The perceived value of additively manufactured digital spare parts in industry: An empirical investigation," *International Journal of Production Economics*, vol. 205, pp. 87–97, 2018, ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2018.09.008 (Cited on pages 31, 32).

[98] G. Komineas, P. Foteinopoulos, A. Papacharalampopoulos, and P. Stavropoulos, "Build time estimation models in thermal extrusion additive manufacturing processes," in *Procedia Manuf.*, Journal Abbreviation: Procedia Manuf., vol. 21, Elsevier B.V., 2018, pp. 647–654. DOI: 10.1016/j.promfg.2018.02.167 (Cited on page 31).

[99] L. Zhuang, A. Xu, and X.-L. Wang, "A prognostic driven predictive maintenance framework based on bayesian deep learning," *Reliability Engineering and System Safety*, vol. 234, 2023. DOI: 10.1016/j.ress.2023.109181 (Cited on pages 36, 37, 38, 39).

[100] E. Dong *et al.*, "A data-driven intelligent predictive maintenance decision framework for mechanical systems integrating transformer and kernel density estimation," *Computers and Industrial Engineering*, vol. 201, 2025. DOI: 10.1016/j.cie.2025.110868 (Cited on pages 36, 37, 38).

[101] Z. Wang, C. Hu, W. Wang, X. Kong, and W. Zhang, "A prognostics-based spare part ordering and system replacement policy for a deteriorating system subjected to a random lead time," *International Journal of Production Research*, vol. 53, no. 15, pp. 4511–4527, 2015. DOI: 10.1080/00207543.2014.988892 (Cited on pages 36, 37, 38, 39).

[102] K.-A. Nguyen, P. Do, and A. Grall, "Joint predictive maintenance and inventory strategy for multi-component systems using birnbaum's structural importance," *Reliability Engineering and System Safety*, vol. 168, pp. 249–261, 2017. DOI: 10.1016/j.ress.2017.05.034 (Cited on pages 36, 37, 39).

[103] B. Cai *et al.*, "Condition-based maintenance method for multi-component system based on RUL prediction: Subsea tree system as a case study," *Computers and Industrial Engineering*, vol. 173, 2022. DOI: 10.1016/j.cie.2022.108650 (Cited on pages 36, 37, 39).

[104] J. Zhu, B. Li, Z. Zhu, and X. Zhao, "Dynamic predictive maintenance strategy for the multi-state system based on remaining life prediction," *Reliability Engineering and System Safety*, vol. 264, 2025. DOI: 10.1016/j.ress.2025.111289 (Cited on page 37).

[105] T. Yan, Y. Lei, N. Li, X. Si, L. Pintelon, and R. Dewil, "Online joint replacement-order optimization driven by a nonlinear ensemble remaining useful life prediction method," *Mechanical Systems and Signal Processing*, vol. 173, 2022. DOI: 10.1016/j.ymssp.2022.109053 (Cited on page 37).

[106] C. Chen, Z. Zhu, J. Shi, N. Lu, and B. Jiang, "Dynamic predictive maintenance scheduling using deep learning ensemble for system health prognostics," *IEEE Sensors Journal*, vol. 21, no. 23, pp. 26 878–26 891, 2021. DOI: 10.1109/JSEN.2021.3119553 (Cited on pages 37, 39, 59).

[107] Y. Wang, G. Wang, Y. Wu, G. Zhang, and M. Wu, "An uncertainty-aware deep learning ensemble approach for effective cutting tool predictive maintenance decision-making," *Measurement Science and Technology*, vol. 36, no. 2, 2025. DOI: 10.1088/1361-6501/ada462 (Cited on pages 37, 38).

[108] K. Wesendrup and B. Hellingrath, "Post-prognostics demand management, production, spare parts and maintenance planning for a single-machine system using reinforcement learning," *Computers and Industrial Engineering*, vol. 179, 2023. DOI: 10.1016/j.cie.2023.109216 (Cited on page 37).

[109] Z. Nie, Z. Zhu, W. Cheng, X. Chen, S. Zhang, and W. Wang, "Joint optimization for predictive maintenance and spare parts management via multiagent reinforcement learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, 2025. DOI: 10.1109/TIM.2025.3618740 (Cited on pages 37, 39).

[110] K. Verbert, B. De Schutter, and R. Babuška, "Timely condition-based maintenance planning for multi-component systems," *Reliability Engineering & System Safety*, vol. 159, pp. 310–321, Mar. 1, 2017, ISSN: 0951-8320. DOI: 10.1016/j.ress.2016.10.032 (Cited on pages 37, 39).

[111] H.-K. Wang, H.-Z. Huang, Y.-F. Li, and Y.-J. Yang, "Condition-based maintenance with scheduling threshold and maintenance threshold," *IEEE Transactions on Reliability*, vol. 65, no. 2, pp. 513–524, 2016. DOI: 10.1109/TR.2015.2487578 (Cited on pages 37, 39).

[112] F. Zhao, Y. Zhang, and X. Liu, "Joint optimization of spare ordering and preventive replacement policy based on RUL," presented at the 2019 Prognostics and System Health Management Conference, PHM-Qingdao 2019, 2019. DOI: 10.1109/PHM-Qingdao46334.2019.8942897 (Cited on page 37).

[113] J. Xia, Y. Feng, C. Lu, Z. Song, and Q. Du, "Degradation prediction and rolling predictive maintenance policy for multi-sensor systems based on two-dimensional self-attention," *Advanced Engineering Informatics*, vol. 54, p. 101 772, Oct. 1, 2022, ISSN: 1474-0346. DOI: 10.1016/j.aei.2022.101772 (Cited on pages 37, 39).

[114] D. Lyu, E. Liu, B. Zhang, E. Zio, T. Yang, and J. Xiang, "Threshold-varying assessment for prognostics and health management," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 1, pp. 685–698, 2025. DOI: 10.1109/TSMC.2024.3489879 (Cited on page 37).

[115] H. Rozas, W. Xie, N. Gebraeel, and S. Robinson, "Data-driven joint optimization of maintenance and spare parts provisioning: A distributionally robust approach," *European Journal of Operational Research*, vol. 328, no. 1, pp. 122–136, 2026. DOI: 10.1016/j.ejor.2025.06.025 (Cited on pages 37, 39).

[116] C. Wang, J. Xu, H. Wang, and Z. Zhang, "Condition-based predictive order model for a mechanical component following inverse gaussian degradation process," *Mathematical Problems in Engineering*, vol. 2018, 2018. DOI: 10.1155/2018/9734189 (Cited on page 37).

[117] M. Basingab, "Agent-based predictive maintenance and throughput optimization for sustainable industrial environments," *Journal of Industrial Information Integration*, vol. 53, 2026. DOI: 10.1016/j.jii.2026.101187 (Cited on pages 37, 38, 39).

[118] L. Wang, Y. Chen, Z. Zhu, and X. Zhao, "Dynamic predictive maintenance framework for mechanical systems via uncertainty-aware RUL estimation," *Mechanical Systems and Signal Processing*, vol. 247, 2026. DOI: 10.1016/j.ymssp.2026.113977 (Cited on pages 37, 39, 98).

[119] J. Zhu, B. Li, and X. Zhao, "A predictive maintenance framework based on conservative remaining useful life method," *Annals of Operations Research*, 2026. DOI: 10.1007/s10479-026-07419-z (Cited on pages 37, 39, 98).

[120] M. Almatared, M. Sulaiman, A. Alghamdi, and E. Nasrallah, "An artificial-intelligence-based predictive maintenance strategy using long short-term memory networks for optimizing HVAC system performance in commercial buildings," *Buildings*, vol. 15, no. 22, 2025. DOI: 10.3390/buildings15224129 (Cited on pages 37, 38, 39, 95, 98).

[121] A. Bousdekis, N. Papageorgiou, B. Magoutas, D. Apostolou, and G. Mentzas, "A proactive event-driven decision model for joint equipment predictive maintenance and spare parts inventory optimization," presented at the Procedia CIRP, vol. 59, 2017, pp. 184–189. DOI: 10.1016/j.procir.2016.09.015 (Cited on pages 38, 39).

[122] H. Chau, R. Li, T. Lee, B. Cheung, and T. Loy, "Implementation of computerized maintenance management system in upgraded pillar point sewage treatment works," *Lecture Notes in Mechanical Engineering*, vol. 19, pp. 889–900, 2015. DOI: 10.1007/978-3-319-09507-3_77 (Cited on pages 38, 39).

[123] H.-M. Park *et al.*, "An end-to-end predictive maintenance system for autonomous inspection based on health index," presented at the International Conference on Advanced Communication Technology, ICACT, 2026, pp. 141–149. DOI: 10.23919/ICACT68090.2026.11431557 (Cited on pages 38, 39).

[124] B. Einabadi, M. Mahmoodjanloo, A. Baboli, and E. Rother, "Dynamic predictive and preventive maintenance planning with failure risk and opportunistic grouping considerations: A case study in the automotive industry," *Journal of Manufacturing Systems*, vol. 69, pp. 292–310, 2023. DOI: 10.1016/j.jmsy.2023.06.012 (Cited on page 38).

[125] C. Sanga, V. Prado, P. Sanga, A. Sanga, and N. Chambi, "A prescriptive maintenance framework for textile machinery enabled by hybrid machine learning and multi-objective optimization," *Eng*, vol. 7, no. 5, 2026. DOI: 10.3390/eng7050210 (Cited on pages 38, 39).

[126] M. Khan, G. Rozati, and T. Wuest, "Sensor triggered replacement of spare parts: Customer service process innovation," presented at the IFIP Advances in Information and Communication Technology, vol. 488, 2016, pp. 343–350. DOI: 10.1007/978-3-319-51133-7_41 (Cited on pages 38, 39).

[127] P. Zhao, X. Yi, Q. Wang, S. Liu, and Y. Liu, "Research on armored equipment autonomic logistics mode based on PHM technology," in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, Journal Abbreviation: 2018 Prognostics and System Health Management Conference (PHM-Chongqing), Oct. 26, 2018, pp. 559–566, ISBN: 2166-5656. DOI: 10.1109/PHM-Chongqing.2018.00101 (Cited on page 38).

[128] D. Skoumpopoulou, S. Toliyat, A. Ojra, A. Shokri, and S. Hu, "Challenges of achieving digital transformation in manufacturing firms: The case of predictive maintenance and spare part inventory management," *Journal of Manufacturing Technology Management*, vol. 36, no. 1, pp. 159–178, 2025. DOI: 10.1108/JMTM-04-2024-0211 (Cited on page 38).

[129] A. Shokri, S. Toliyat, S. Hu, and D. Skoumpopoulou, "Integrating spare part inventory management and predictive maintenance as a digital supply chain solution," *Journal of Modelling in Management*, vol. 20, no. 3, pp. 1003–1029, 2025. DOI: 10.1108/JM2-05-2024-0131 (Cited on page 38).

[130] P. Wittbrodt, "Algorithmic approach to prescriptive maintenance in industry 4.0," *Production Engineering Archives*, vol. 32, no. 1, pp. 123–135, 2026, Type: Article. DOI: 10.30657/pea.2026.32.11 (Cited on pages 38, 40, 41).

[131] H. Mao, Z. Liu, C. Qiu, Y. Huang, and J. Tan, "Prescriptive maintenance for complex products with digital twin considering production planning and resource constraints," *Measurement Science and Technology*, vol. 34, no. 12, 2023, Type: Article. DOI: 10.1088/1361-6501/aced5f (Cited on pages 38, 40, 41).

[132] A. D. Cho, R. A. Carrasco, and G. A. Ruz, "Improving prescriptive maintenance by incorporating post-prognostic information through chance constraints," *IEEE Access*, vol. 10, pp. 55 924–55 932, 2022, Type: Article. DOI: 10.1109/ACCESS.2022.3177537 (Cited on page 38).

[133] R. Meissner, A. Rahn, and K. Wicke, "Developing prescriptive maintenance strategies in the aviation industry based on a discrete-event simulation framework for post-prognostics decision making," *Reliability Engineering and System Safety*, vol. 214, 2021, Type: Article. DOI: 10.1016/j.ress.2021.107812 (Cited on pages 38, 41).

[134] Y. Zhao, J. Yang, W. Wang, H. Yang, and D. Niyato, "TranDRL: A transformer-driven deep reinforcement learning enabled prescriptive maintenance framework," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 35 432–35 444, 2024, Type: Article. DOI: 10.1109/JIOT.2024.3436110 (Cited on pages 38, 40, 41).

[135] Z. Hao, F. Di Maio, L. Pinciroli, and E. Zio, "Optimal prescriptive maintenance of nuclear power plants by deep reinforcement learning," in *Proceedings of the 32nd European Safety and Reliability Conference, ESREL 2022 - Understanding and Managing Risk and Reliability for a Sustainable Future*, Type: Conference paper, 2022, pp. 2812–2819. DOI: 10.3850/978-981-18-5183-4_S22-02-466-cd (Cited on page 38).

[136] Z. Shao and M. Kumral, "A decision support framework based on deep reinforcement learning for equipment health diagnostics toward prescriptive maintenance," *Mining, Metallurgy and Exploration*, vol. 43, no. 1, pp. 119–150, 2026, Type: Article. DOI: 10.1007/s42461-025-01441-8 (Cited on pages 38, 41).

[137] C.-K. Tham, N. Sharma, and J. Hu, "Model-based and model-free prescriptive maintenance on edge computing nodes," in *IEEE Vehicular Technology Conference*, Type: Conference paper, vol. 2023-June, 2023. DOI: 10.1109/VTC2023-Spring57618.2023.10200886 (Cited on pages 38, 40, 41).

[138] S. Joshi, S. Patil, R. Shriram, and N. Daimiwal, "Data-driven wind turbine fault prediction," in *2026 IEEE 1st International Conference on Instrumentation (INSTCon)*, Journal Abbreviation: 2026 IEEE 1st International Conference on Instrumentation (INSTCon), Jul. 24, 2026, pp. 1–7. DOI: 10.1109/INSTCon69741.2026.11691885 (Cited on page 40).

[139] S. Todoskoff, N. Björsell, S. Verron, and B. Castanier, "Integrating health indicators into production planning via prescriptive maintenance," in *IEEE PES Innovative Smart Grid Technologies Conference Europe*, Type: Conference paper, 2025. DOI: 10.1109/ISGTEurope64741.2025.11305653 (Cited on page 40).

[140] P. D. Longhitano, K. Tidriri, C. Bérenguer, and B. Echard, "A closed-loop prescriptive maintenance approach for an usage dependent deteriorating item – application to a critical vehicle component," in *Proceedings of the 31st European Safety and Reliability Conference, ESREL 2021*, Type: Conference paper, 2021, pp. 2465–2472. DOI: 10.3850/978-981-18-2016-8_545-cd (Cited on page 40).

[141] N. Esposito, B. Castanier, and M. Giorgio, "A prescriptive maintenance policy for a gamma deteriorating unit," in *Proceedings of the 32nd European Safety and Reliability Conference, ESREL 2022 - Understanding and Managing Risk and Reliability for a Sustainable Future*, Type: Conference paper, 2022, pp. 635–641. DOI: 10.3850/978-981-18-5183-4_R14-05-139-cd (Cited on page 40).

[142] N. Björsell and B. -O. Gradén, "Integrating machine conditioning monitoring into prescriptive maintenance for operational optimization," in *2026 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Journal Abbreviation: 2026 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), May 25, 2026, pp. 1–6, ISBN: 2642-2077. DOI: 10.1109/I2MTC66907.2026.11695012 (Cited on pages 40, 41).

[143] X. Tang, C. Wang, Y. Gao, C. Wu, and Z. Fang, "Rec-PdM: Decision-calibrated recursive multi-agent systems for uncertainty-aware prescriptive maintenance in smart manufacturing," *Journal of Manufacturing Systems*, vol. 89, pp. 260–279, 2026, Type: Article. DOI: 10.1016/j.jmsy.2026.08.012 (Cited on pages 40, 41).

[144] S. Chen *et al.*, "ENHANCING DIGITAL TWINS WITH DEEP REINFORCEMENT LEARNING: A USE CASE IN MAINTENANCE PRIORITIZATION," in *Proceedings - Winter Simulation Conference*, Type: Conference paper, 2024, pp. 1611–1622. DOI: 10.1109/WSC63780.2024.10838867 (Cited on pages 40, 41).

[145] O. Abdelkader, S. Scheffer, and F. Ansari, "Integrating augmented reality in knowledge-based maintenance: A layered architecture for informed decision support in metro systems," in *Procedia CIRP*, Type: Conference paper, vol. 142, 2026, pp. 896–901. DOI: 10.1016/j.procir.2026.05.365 (Cited on page 40).

[146] S. Choubey, R. Benton, and T. Johnsten, "Prescriptive equipment maintenance: A framework," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, Type: Conference paper, 2019, pp. 4366–4374. DOI: 10.1109/BigData47090.2019.9006213 (Cited on pages 40, 41).

[147] M. Alonso-González, V. García-Díaz, and B. López Pérez, "Designing an agentic AI–driven framework for decision-making in industrial maintenance," *Communications in Computer and Information Science*, vol. 2729 CCIS, pp. 118–130, 2026, Type: Conference paper. DOI: 10.1007/978-3-032-09945-7_11 (Cited on pages 40, 41).

[148] J. Ordieres-Meré, A. Sánchez-Herguedas, and Á. Mena-Nieto, "A data-driven monitoring system for a prescriptive maintenance approach: Supporting reinforcement learning strategies," *Applied Sciences (Switzerland)*, vol. 15, no. 12, 2025, Type: Article. DOI: 10.3390/app15126917 (Cited on pages 40, 41).

[149] F. Mahr, C. Kratzke, J. Franke, and M. Ockel, "Towards a multi-agent-based diagnostic system for prescriptive maintenance applications in industrial environments," in *2026 International Conference on Electronics Packaging and Hybrid Bonding Symposium, ICEP-HBS 2026*, Type: Conference paper, 2026, pp. 359–360. DOI: 10.23919/ICEP-HBS69241.2026.11550601 (Cited on page 40).

[150] K. Shahin and S. Lazarova-Molnar, "Digital twins in prognostics and health management: A review of diagnostic, remaining useful life, and predictive maintenance applications," presented at the 2024 8th International Conference on System Reliability and Safety, ICSRS 2024, 2024, pp. 720–729. DOI: 10.1109/ICSRS63046.2024.10927469 (Cited on page 42).

[151] E. Muka, G. Marinova, and E. Mankolli, "A review of digital twin architectures in IIoT-driven manufacturing," presented at the Communications in Computer and Information Science, vol. 2900 CCIS, 2026, pp. 303–316. DOI: 10.1007/978-3-032-24513-7_26 (Cited on page 42).

[152] G. Fede, F. Sgarbossa, and N. Paltrinieri, "Integrating production and maintenance planning in process industries using digital twin: A literature review," presented at the IFAC-PapersOnLine, 19, vol. 58, 2024, pp. 151–156. DOI: 10.1016/j.ifacol.2024.09.124 (Cited on page 42).

[153] M. Manzelati, M. Hashemi, and A. Karimiafshar, "Optimizing predictive maintenance based on digital twins using artificial intelligence tools," presented at the Proceedings of 2025 9th International Conference on Internet of Things and Applications, IoT 2025, 2025. DOI: 10.1109/IoT69654.2025.11297603 (Cited on pages 42, 43).

[154] Q. Nguyen, T. Luong, P. Vu, H. Luong, P. Vo, and N. Pham, "Digital twin–driven predictive maintenance in resource-constrained manufacturing: Comparative models and operational impact from a vietnam case," presented at the Lecture Notes in Networks and Systems, vol. 1963 LNNS, 2026, pp. 229–240. DOI: 10.1007/978-3-032-25631-7_19 (Cited on pages 42, 43).

[155] Q. Xu and S. Wang, "Digital twin-driven real-time prediction and fault-tolerant control for a CNC machining unit in an intelligent manufacturing workshop," *International Journal on Interactive Design and Manufacturing*, 2026. DOI: 10.1007/s12008-026-02688-1 (Cited on pages 42, 43, 45).

[156] D. Kibira, G. Shao, and B. Weiss, "Buiding a digital twin for robot workcell prognostics and health management," presented at the Proceedings - Winter Simulation Conference, vol. 2021-December, 2021. DOI: 10.1109/WSC52266.2021.9715501 (Cited on page 42).

[157] K. Zhao, Y. Liu, W. Ming, Y. Zhou, and J. Wu, "Digital twin-supported BESS decision support of RUL-based maintenance," presented at the Proceedings of the 31st ICE IEEE/ITMC Conference on Engineering, Technology, and Innovation: AI-Driven Industrial Transformation: Digital Leadership in Technology, Engineering, Innovation and Entrepreneurship, ICE 2025, 2025. DOI: 10.1109/ICE/ITMC65658.2025.11106621 (Cited on pages 42, 43).

[158] K. Cheikh, E. Boudi, R. Rabi, and H. Mokhliss, "Stochastic maintenance optimization in additive manufacturing with digital twin integration," *International Journal of Advanced Manufacturing Technology*, vol. 143, no. 1, pp. 971–1005, 2026. DOI: 10.1007/s00170-026-17400-6 (Cited on pages 42, 43, 45).

[159] J. Ren, C. Xu, J. Wang, J. Zhang, X. Mao, and W. Shen, "An edge-fog-cloud computing-based digital TwinModel for prognostics HealthManagement of ProcessManufacturing systems," *CMES - Computer Modeling in Engineering and Sciences*, vol. 135, no. 1, pp. 599–618, 2023. DOI: 10.32604/cmes.2022.022415 (Cited on pages 42, 43, 45).

[160] R. Sakha, S. Venkat, R. Sekaran, R. Vaithiyanathan, J. S, and G. G S, "A safety-constrained agentic digital twin framework for predictive maintenance in smart manufacturing systems," in *2026 International Conference on Data Science and Business Systems (ICDSBS)*, Journal Abbreviation: 2026 International Conference on Data Science and Business Systems (ICDSBS), vol. 3, May 5, 2026, pp. 1–5. DOI: 10.1109/ICDSBS69077.2026.11634862 (Cited on pages 42, 43, 44, 45).

[161] M. M. R. Shamim, M. Nuruzzaman, M. M. R. Opu, M. M. Rahman, M. R. Ahmed, and S. K. Ghosh, "Physics-informed graph digital twin with reinforcement learning and sensor data fusion for predictive maintenance optimization in smart manufacturing," in *2026 5th International Conference on Communication, Computing and Electronics Systems (ICCCES)*, Journal Abbreviation: 2026 5th International Conference on Communication, Computing and Electronics Systems (ICCCES), Jan. 21, 2026, pp. 18–24. DOI: 10.1109/ICCCES62661.2026.11437105 (Cited on pages 42, 44, 45).

[162] L. Mendonça, J. Barbosa, and P. Leitao, "A human-centred architecture integrating digital twin and agentic AI for predictive maintenance," in *2026 IEEE International Conference on Industrial Technology (ICIT)*, Journal Abbreviation: 2026 IEEE International Conference on Industrial Technology (ICIT), Mar. 4, 2026, pp. 1–6, ISBN: 2643-2978. DOI: 10.1109/ICIT64854.2026.11491044 (Cited on pages 42, 44, 45).

[163] N. Hamzah, T. Mutallimov, C.-T. Panaitescu, M. Grant, G. Cristofoli, and H. Tan, "A modular digital twin architecture for offshore safety and decision-making: A north sea platform case study," *Reliability Engineering and System Safety*, vol. 274, 2026. DOI: 10.1016/j.ress.2026.112428 (Cited on pages 42, 43, 45).

[164] S. Embarki, Y. El Kihel, and B. El Kihel, "Industrial deployment of a decision-oriented digital twin for predictive maintenance of a reverse osmosis system: An end-to-end case study in a beverage filling plant," *Journal Europeen des Systemes Automatises*, vol. 59, no. 3, pp. 785–796, 2026. DOI: 10.18280/jesa.590321 (Cited on pages 42, 43, 45).

[165] B. Soykan, G. Blanc, and G. Rabadi, "A proof-of-concept digital twin for real-time simulation: Leveraging a model-based systems engineering approach," *IEEE Access*, vol. 13, pp. 58 899–58 912, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3557367 (Cited on pages 42, 43, 45).

[166] Z. Hu, C. Hu, and W. Hu, "16 - a tutorial on digital twins for predictive maintenance," in *Structural Health Monitoring/ Management (SHM) in Aerospace Structures*, F.-G. Yuan, Ed., Woodhead Publishing, Jan. 1, 2024, pp. 453–501, ISBN: 978-0-443-15476-8. DOI: 10.1016/B978-0-443-15476-8.00005-8 (Cited on page 42).

[167] Z. Wu and J. Li, "A framework of dynamic data driven digital twin for complex engineering products: The example of aircraft engine health management," presented at the Procedia Manufacturing, vol. 55, 2021, pp. 139–146. DOI: 10.1016/j.promfg.2021.10.020 (Cited on page 42).

[168] L. Dong, R. F. Guo, and Z. Zhigang, "Update mechanism and consistency control and management methods for complex system digital twin models," in *2025 IEEE 23rd International Conference on Industrial Informatics (INDIN)*, Journal Abbreviation: 2025 IEEE 23rd International Conference on Industrial Informatics (INDIN), Jul. 12, 2025, pp. 1–8, ISBN: 2378-363X. DOI: 10.1109/INDIN64977.2025.11278954 (Cited on page 43).

[169] J. Murray *et al.*, "Reconceptualizing the prognostics digital twin for smart manufacturing with data-driven evolutionary models and adaptive uncertainty quantification," presented at the Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM, 1, vol. 15, 2023 (Cited on page 43).

[170] M. Vathoopan, M. Johny, A. Zoitl, and A. Knoll, "Modular fault ascription and corrective maintenance using a digital twin," 11, vol. 51, 2018, pp. 1041–1046. DOI: 10.1016/j.ifacol.2018.08.470 (Cited on page 43).

[171] B. Deon *et al.*, "Digital twin and machine learning for decision support in thermal power plant with combustion engines," *Knowledge-Based Systems*, vol. 253, 2022. DOI: 10.1016/j.knosys.2022.109578 (Cited on page 43).

[172] W. Hu, Z. Cao, A. E. Messaoudi, P. Li, and Z. Zeng, "A desktop learning factory for smart and resilient manufacturing based on digital twin and AI," in *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*, Journal Abbreviation: 2025 11th International Conference on Control, Decision and Information Technologies (CoDIT), vol. 1, Jul. 15, 2025, pp. 739–744, ISBN: 2576-3555. DOI: 10.1109/CoDIT66093.2025.11321707 (Cited on page 43).

[173] S. Essalih, Z. Haouat, M. Ramadany, F. Bennouna, and D. Amegouz, "Application of an industry 4.0 implementation model: Digital twin-based predictive maintenance of a polish filter in palm oil refining," *International Journal of Advanced Manufacturing Technology*, vol. 142, no. 7, pp. 3719–3733, 2026. DOI: 10.1007/s00170-025-17342-5 (Cited on page 43).

[174] M. Hassan, M. Svadling, and N. Björsell, "Experience from implementing digital twins for maintenance in industrial processes," *Journal of Intelligent Manufacturing*, vol. 35, no. 2, pp. 875–884, 2024. DOI: 10.1007/s10845-023-02078-4 (Cited on page 43).

[175] H. Nozari and A. Szmelter-Jarosz, "A predictive maintenance approach for composting plants based on ERP and digital twin integration," *Machines*, vol. 13, no. 12, 2025. DOI: 10.3390/machines13121123 (Cited on pages 43, 45).

[176] D. Ariansyah and B. Pardamean, "Enhancing interoperability of digital twin in the maintenance phase of lifecycle," presented at the Proceeding - 6th International Conference on Information Technology, Information Systems and Electrical Engineering: Applying Data Sciences and Artificial Intelligence Technologies for Environmental Sustainability, ICITISEE 2022, 2022, pp. 514–518. DOI: 10.1109/ICITISEE57756.2022.10057666 (Cited on pages 43, 45).

[177] S. Wang, Y. Wang, and J. Wang, "Spare parts demand forecasting method of modern enterprises based on digital twin model," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 13, no. 6, 2022. DOI: 10.1142/S1793962322500453 (Cited on page 43).

[178] M. Li, C.-M. Yang, W. Lo, and Y.-W. Kao, "A digital-twin-enabled AI-driven adaptive planning platform for sustainable and reliable manufacturing," *Machines*, vol. 14, no. 2, 2026. DOI: 10.3390/machines14020197 (Cited on pages 43, 45).

[179] J. Liu, D. Yu, Y. Hu, H. Yu, W. He, and L. Zhang, "CNC machine tool fault diagnosis integrated rescheduling approach supported by digital twin-driven interaction and cooperation framework," *IEEE Access*, vol. 9, pp. 118 801–118 814, 2021. DOI: 10.1109/ACCESS.2021.3106797 (Cited on page 43).

[180] L. Damant *et al.*, "Exploring the transition from preventive maintenance to predictive maintenance within ERP systems by utilising digital twins," presented at the Advances in Transdisciplinary Engineering, vol. 16, 2021, pp. 171–180. DOI: 10.3233/ATDE210095 (Cited on page 43).

[181] R. Myakala, S. Pedhapally, V. Podduturi, A. Jagirapu, P. Nagata, and S. Lavudya, "Unified predictive maintenance with generative anomaly injection, RL scheduling in data-driven digital twins," presented at the Proceedings - 2025 IEEE International Conference on Future Machine Learning and Data Science, FMLDS 2025, 2025, pp. 176–183. DOI: 10.1109/FMLDS67896.2025.00039 (Cited on page 43).

[182] C. Badoi, B. Cetin, K. Cetin, C. Karatas, M. Ozbek, and S. Sahin, "Architecting the future factory: Industrial PLC-IoT, digital twins and predictive AI based maintenance," presented at the International Symposium on Wireless Personal Multimedia Communications, WPMC, 2025. DOI: 10.1109/WPMC67460.2025.11351140 (Cited on page 43).

[183] S. Ma, K. Flanigan, and M. Berges, "Levels of autonomy for predictive maintenance: A structured approach with digital twin integration," presented at the 2025 IEEE 5th International Conference on Digital Twins and Parallel Intelligence, DTPI 2025, 2025. DOI: 10.1109/DTPI65196.2025.11088866 (Cited on page 44).

[184] N. Kamoise, C. Guérin, M. Hamzaoui, and N. Julien, "Using cognitive work analysis to deploy collaborative digital twins: Application to predictive maintenance," presented at the Proceedings of the 32nd European Safety and Reliability Conference, ESREL 2022 - Understanding and Managing Risk and Reliability for a Sustainable Future, 2022, pp. 2336–2343. DOI: 10.3850/978-981-18-5183-4_S10-10-646-cd (Cited on pages 44, 45).

[185] H. A. S. Leitão, R. S. U. Rosso, A. B. Leal, and A. Zoitl, "Fault handling in discrete event systems applied to IEC 61499," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Type: Conference paper, vol. 2020-September, 2020, pp. 1039–1042. DOI: 10.1109/ETFA46521.2020.9212177 (Cited on pages 46, 49).

[186] U. D. Atmojo *et al.*, "Distributed control architecture for dynamic reconfiguration: Flexible assembly line case study," in *Proceedings - 2018 IEEE Industrial Cyber-Physical Systems, ICPS 2018*, Type: Conference paper, 2018, pp. 690–695. DOI: 10.1109/ICPHYS.2018.8390791 (Cited on pages 46, 49).

[187] N. Zhou, D. Li, S. Li, S. Wang, and C. Liu, "Model-based development of knowledge-driven self-reconfigurable machine control systems," *IEEE Access*, vol. 5, pp. 19 909–19 919, 2017, Type: Article. DOI: 10.1109/ACCESS.2017.2754507 (Cited on pages 46, 49).

[188] G. Zhabelova, V. Vyatkin, and V. N. Dubinin, "Toward industrially usable agent technology for smart grid automation," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2629–2641, Apr. 2015, ISSN: 1557-9948. DOI: 10.1109/TIE.2014.2371777 (Cited on page 46).

[189] N. Kashyap, C.-W. Yang, S. Sierla, and P. G. Flikkema, "Automated fault location and isolation in distribution grids with distributed control and unreliable communication," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2612–2619, 2015, Type: Article. DOI: 10.1109/TIE.2014.2387093 (Cited on page 46).

[190] W. Dai, V. N. Dubinin, J. H. Christensen, V. Vyatkin, and X. Guan, "Toward self-manageable and adaptive industrial cyber-physical systems with knowledge-driven autonomic service management," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 725–736, Apr. 2017, ISSN: 1941-0050. DOI: 10.1109/TII.2016.2595401 (Cited on pages 46, 49).

[191] W. Dai, L. Riliskis, P. Wang, V. Vyatkin, and X. Guan, "A cloud-based decision support system for self-healing in distributed automation systems using fault tree analysis," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 989–1000, 2018, Type: Article. DOI: 10.1109/TII.2018.2791503 (Cited on pages 46, 49).

[192] L. Prenzel and S. Steinhorst, "Towards resilience by self-adaptation of industrial control systems," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Type: Conference paper, vol. 2022-September, 2022. DOI: 10.1109/ETFA52439.2022.9921597 (Cited on pages 46, 49).

[193] R. W. Brennan and G. Lyu, "Towards sustainable cyber-physical manufacturing systems," in *IFAC-PapersOnLine*, Type: Conference paper, vol. 58, 2024, pp. 345–350. DOI: 10.1016/j.ifacol.2024.07.175 (Cited on pages 46, 49).

[194] B. Dowdeswell, R. Sinha, and S. G. Macdonell, "Architecting an agent-based fault diagnosis engine for IEC 61499 industrial cyber-physical systems," *Future Internet*, vol. 13, no. 8, 2021, Type: Article. DOI: 10.3390/fi13080190 (Cited on page 46).

[195] J. Kang, Y. Zhang, X. Wen, and W. Dai, "Towards automatic deployment process for reconfigurable industrial edge applications," in *ICIEA 2022 - Proceedings of the 17th IEEE Conference on Industrial Electronics and Applications*, Type: Conference paper, 2022, pp. 1228–1233. DOI: 10.1109/ICIEA54703.2022.10005974 (Cited on pages 46, 49).

[196] F. V. Nindyasari, A. N. I. Wardana, and A. Arif, "Online tuning diagnosis of proportional integral derivative controller based on IEC 61499 function blocks," *Jurnal Nasional Teknik Elektro*, vol. 10, no. 3, pp. 160–165, 2021, Type: Article. DOI: 10.25077/jnte.v10n3.940.2021 (Cited on page 47).

[197] V. Barbosa, R. Pinto, J. Pinheiro, G. Gonçalves, and A. Ribeiro, "Towards a digital twin simulation for cycle times analysis in a cyber-physical production system," in *Proceedings of the International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, Type: Conference paper, vol. 1, 2023, pp. 369–376. DOI: 10.5220/0012123100003546 (Cited on page 47).

[198] R. Silva, L. Neto, G. Gonçalves, F. Mateus, and R. Pires, "Digital twin for remote fault detection of heavy-duty machinery," in *IEEE International Conference on Industrial Informatics (INDIN)*, Type: Conference paper, 2024. DOI: 10.1109/INDIN58382.2024.10774207 (Cited on pages 47, 49).

[199] L. A. Cruz Salazar, H. S. Tovar, E. A. Prada Salamanca, J. C. Márquez Juárez, and E. M. Cruz, "Adaptive learning and interoperability in i4.0: A case study of agent-based digital twin with asset administration shell," in *IFAC-PapersOnLine*, Type: Conference paper, vol. 59, 2025, pp. 202–207. DOI: 10.1016/j.ifacol.2025.12.150 (Cited on pages 47, 49).

[200] T. Ladune, M. Colas, A. Philippot, B. Riera, F. Gellot, and P. Hampikian, "Analysis of event-based camera signals for safety integration to IEC61499," in *12th 2026 International Conference on Control, Decision and Information Technologies, CoDIT 2026*, Type: Conference paper, 2026, pp. 3444–3449. DOI: 10.1109/CoDIT70676.2026.11631249 (Cited on page 47).

[201] T. Lyu, N. Galkin, T. Liakh, C. -W. Yang, and V. Vyatkin, "Methods of data streaming from IEC 61499 applications to cloud storages," in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, Journal Abbreviation: 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), Jun. 19, 2023, pp. 1–6, ISBN: 2163-5145. DOI: 10.1109/ISIE51358.2023.10228163 (Cited on page 47).

[202] S. Dobrescu, O. Chenaru, N. Matei, L. Ichim, and D. Popescu, "A service oriented system of reusable algorithms for distributed control of petroleum facilities in onshore oilfields," presented at the Proceedings of the 8th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2016, 2017. DOI: 10.1109/ECAI.2016.7861115 (Cited on page 47).

[203] G. Vieira, J. Barbosa, P. Leitao, and L. Sakurada, "Low-cost industrial controller based on the raspberry pi platform," presented at the Proceedings of the IEEE International Conference on Industrial Technology, vol. 2020-February, 2020, pp. 292–297. DOI: 10.1109/ICIT45562.2020.9067148 (Cited on page 47).

[204] T. Lyu, U. Atmojo, and V. Vyatkin, "Complexity-aware distribution optimization for computing-intensive distributed automation systems with IEC 61499," presented at the IEEE International Conference on Industrial Informatics (INDIN), 2025. DOI: 10.1109/INDIN64977.2025.11279613 (Cited on page 47).

[205] S. Gao, K. Qin, X. Zhang, and W. Dai, "Prioritized deterministic real-time execution semantics for industrial edge applications based on IEC 61499 event types," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 11, pp. 8550–8561, 2025, Type: Article. DOI: 10.1109/TII.2025.3584428 (Cited on page 47).

[206] Y. Wang *et al.*, "Digital workshop health status evaluation method based on DTVSM," *Journal of Manufacturing Systems*, vol. 88, pp. 108–125, 2026, Type: Article. DOI: 10.1016/j.jmsy.2026.05.011 (Cited on page 47).

[207] F. Foerster, D. Mueller, D. Scholz, A. Michalik, and L. Kiebler, "Integration of condition based maintenance orders into the decision-making of autonomous control methods," presented at the Procedia CIRP, vol. 81, 2019, pp. 216–221. DOI: 10.1016/j.procir.2019.03.038 (Cited on page 47).

[208] G. Bencheikh, A. Letouzey, and X. Desforges, "An approach for joint scheduling of production and predictive maintenance activities," *Journal of Manufacturing Systems*, vol. 64, pp. 546–560, 2022. DOI: 10.1016/j.jmsy.2022.08.005 (Cited on pages 48, 49).

[209] M. Babu, A. AlArjani, and M. Lahby, "MIRA: Safety-constrained multi-agent reinforcement learning for joint prescriptive maintenance and production rescheduling in industrial IoT," *Future Internet*, vol. 18, no. 8, 2026. DOI: 10.3390/fi18080430 (Cited on pages 48, 49).

[210] R. Peres, A. Dionisio Rocha, P. Leitao, and J. Barata, "IDARTS – towards intelligent data analysis and real-time supervision for industry 4.0," *Computers in Industry*, vol. 101, pp. 138–146, 2018. DOI: 10.1016/j.compind.2018.07.004 (Cited on pages 48, 49).

[211] M. Sekar and N. Nezamoddini, "Real-time predictive maintenance with edge-orchestrated multi-agent pipelines," presented at the IISE Annual Conference and Expo 2025, Conference Proceedings, 2025, pp. 1739–1744. DOI: 10.21872/2025IISE_8830 (Cited on page 48).

[212] T. Borangiu, S. Răileanu, A. Silişteanu, S. Anton, and F. Anton, "Smart manufacturing control with cloud-embedded digital twins," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, Journal Abbreviation: 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), Oct. 8, 2020, pp. 915–920, ISBN: 2372-1618. DOI: 10.1109/ICSTCC50638.2020.9259684 (Cited on pages 48, 49).

[213] S. Guha and A. Datta, "ASPIRE: An agentic decision system for early equipment failure prediction and intervention in industrial IIoT," *IEEE Access*, vol. 14, pp. 105 006–105 031, 2026. DOI: 10.1109/ACCESS.2026.3711010 (Cited on page 48).

[214] A. González-Potes *et al.*, "Hybrid AI and LLM-enabled agent-based real-time decision support architecture for industrial batch processes: A clean-in-place case study," *AI (Switzerland)*, vol. 7, no. 2, 2026. DOI: 10.3390/ai7020051 (Cited on page 48).

[215] M. Farahani, M. Khan, and T. Wuest, "Hybrid agentic AI and multi-agent systems in smart manufacturing," *Journal of Manufacturing Systems*, vol. 86, pp. 612–623, 2026. DOI: 10.1016/j.jmsy.2026.04.002 (Cited on page 48).

[216] J. Heinen and K. Hoberg, "Assessing the potential of additive manufacturing for the provision of spare parts," *Journal of Operations Management*, vol. 65, no. 8, pp. 810–826, 2019. DOI: 10.1002/joom.1054 (Cited on pages 50, 51, 54).

[217] A. Chaudhuri *et al.*, "Selecting spare parts suitable for additive manufacturing: A design science approach," *Production Planning and Control*, vol. 32, no. 8, pp. 670–687, 2021. DOI: 10.1080/09537287.2020.1751890 (Cited on pages 50, 51, 54).

[218] G. Cardeal, M. Leite, and I. Ribeiro, "Decision-support model to select spare parts suitable for additive manufacturing," *Computers in Industry*, vol. 144, 2023. DOI: 10.1016/j.compind.2022.103798 (Cited on pages 50, 51).

[219] R. Lastra, A. Pereira, M. Díaz-Cacho, J. Acevedo, and A. Collazo, "Spare parts made by additive manufacturing to improve preventive maintenance," *Applied Sciences (Switzerland)*, vol. 12, no. 20, 2022. DOI: 10.3390/app122010564 (Cited on pages 50, 51).

[220] M. Peron, A. Coruzzolo, R. Basten, N. Knofius, F. Lolli, and F. Sgarbossa, "Choosing between additive and conventional manufacturing of spare parts: On the impact of failure rate uncertainties and the tools to reduce them," *International Journal of Production Economics*, vol. 278, 2024. DOI: 10.1016/j.ijpe.2024.109438 (Cited on pages 50, 51, 54).

[221] Q. Zhao, A. Coruzzolo, E. Balugani, R. Gamberini, and F. Lolli, "A novel three-way decision framework for classifying spare parts between additive and conventional manufacturing," presented at the IFAC-PapersOnLine, 10, vol. 59, 2025, pp. 1892–1897. DOI: 10.1016/j.ifacol.2025.09.318 (Cited on page 50).

[222] F. De Brito, G. Da Cruz Junior, E. Frazzon, J. Basto, and S. Alcala, "An optimization model for the design of additive manufacturing supply chains," presented at the IEEE International Conference on Industrial Informatics (INDIN), vol. 2019-July, 2019, pp. 881–885. DOI: 10.1109/INDIN41052.2019.8972028 (Cited on page 50).

[223] A. Cantini, M. Peron, F. De Carlo, and F. Sgarbossa, "A decision support system for configuring spare parts supply chains considering different manufacturing technologies," *International Journal of Production Research*, vol. 62, no. 8, pp. 3023–3043, 2024. DOI: 10.1080/00207543.2022.2041757 (Cited on pages 50, 51).

[224] T. Cokyasar, H. Kose, W. Dong, X. Li, M. Jin, and W. Li, "An agent-based simulation model for optimization of spare parts sourcing with additive manufacturing," presented at the IISE Annual Conference and Expo 2018, 2018, pp. 1902–1907 (Cited on page 50).

[225] N. Knofius, M. van der Heijden, A. Sleptchenko, and W. Zijm, "Improving effectiveness of spare parts supply by additive manufacturing as dual sourcing option," *OR Spectrum*, vol. 43, no. 1, pp. 189–221, 2021. DOI: 10.1007/s00291-020-00608-7 (Cited on pages 50, 54).

[226] Y. Zhang, S. Jedeck, L. Yang, and L. Bai, "Modeling and analysis of the on-demand spare parts supply using additive manufacturing," *Rapid Prototyping Journal*, vol. 25, no. 3, pp. 473–487, 2019. DOI: 10.1108/RPJ-01-2018-0027 (Cited on pages 50, 51, 54).

[227] A. Li, H. Lan, J. Guo, Z. Lyu, and Y. Jiang, "A hybrid additive manufacturing production strategy for spare parts inventory under irregular maintenance schedules and capacity constraints," *Computers and Industrial Engineering*, vol. 221, 2026. DOI: 10.1016/j.cie.2026.112310 (Cited on pages 50, 51, 54).

[228] B. Jæger, F. Wiklund, and L. Halse, "Additive manufacturing: A case study of introducing additive manufacturing of spare parts," presented at the IFIP Advances in Information and Communication Technology, vol. 690 AICT, 2023, pp. 605–616. DOI: 10.1007/978-3-031-43666-6_41 (Cited on pages 50, 54).

[229] L. Durão, A. Christ, R. Anderl, K. Schützer, and E. Zancul, "Distributed manufacturing of spare parts based on additive manufacturing: Use cases and technical aspects," presented at the Procedia CIRP, vol. 57, 2016, pp. 704–709. DOI: 10.1016/j.procir.2016.11.122 (Cited on pages 50, 51, 54).

[230] F. Bauer, M. Schrapp, and J. Szijarto, "Accuracy analysis of a piece-to-piece reverse engineering workflow for a turbine foil based on multi-modal computed tomography and additive manufacturing," *Precision Engineering*, vol. 60, pp. 63–75, 2019. DOI: 10.1016/j.precisioneng.2019.07.008 (Cited on pages 50, 51).

[231] J. Montero, S. Weber, M. Bleckmann, and K. Paetzold, "A methodology for the decentralised design and production of additive manufactured spare parts," *Production and Manufacturing Research*, vol. 8, no. 1, pp. 313–334, 2020. DOI: 10.1080/21693277.2020.1790437 (Cited on pages 51, 54).

[232] M. Stadler, K. Schatz, S. Rehfeldt, and H. Klein, "Reverse engineering and performance evaluation of a pump impeller fabricated with additive manufacturing," *Chemie-Ingenieur-Technik*, 2026. DOI: 10.1002/cite.70123 (Cited on page 51).

[233] P. Avdovic, M. Galijasevic, V. Navrotsky, and A. Graichen, ""ICDBY3d" intelligent component development for gas turbine by using 3d printing at siemens energy AB sweden," presented at the Lecture Notes in Networks and Systems, vol. 233, 2021, pp. 286–300. DOI: 10.1007/978-3-030-75275-0_33 (Cited on page 51).

[234] C. Rockinger, J. Tomas, M. Rupp, M. Sedlmajer, J. Kortsch-Banzhaf, and M. Merkel, "An analysis of technical and economic aspects of digital spare parts on variating materials in powder bed fusion of metals using a laser beam (PBF-LB/m)," *Materialwissenschaft und Werkstofftechnik*, vol. 56, no. 6, pp. 884–892, 2025. DOI: 10.1002/mawe.1745 (Cited on page 51).

[235] R. Novaki, A. De M. Leite, A. Sejka, and O. Junior, "An approach based on advanced manufacturing for replacement parts in equipment maintenance context," presented at the Advances in Transdisciplinary Engineering, vol. 16, 2021, pp. 415–424. DOI: 10.3233/ATDE210121 (Cited on page 51).

[236] I. El-Thalji, "Green spare parts evaluation for hybrid warehousing and on-demand manufacturing," *Applied System Innovation*, vol. 8, no. 1, 2025. DOI: 10.3390/asi8010008 (Cited on page 51).

[237] A. Cubillo, S. Perinpanayagam, M. Miguez, and P. John, "Real-time maintenance optimization considering health monitoring and additive manufacturing," *International Journal of Prognostics and Health Management*, vol. 8, no. 2, 2017. DOI: 10.36001/ijphm.2017.v8i2.2632 (Cited on pages 51, 54).

[238] D. Bender, J. Anderson, M. Gilbert, and A. Barari, "Information flow in digital twin for "detection to repair" of defects using additive manufacturing," presented at the IFAC-PapersOnLine, 19, vol. 58, 2024, pp. 736–741. DOI: 10.1016/j.ifacol.2024.09.215 (Cited on pages 51, 53, 54).

[239] B. Nouri Rahmat Abadi, K. Rashvand, and A. Sarhadi, "Towards digital twin-based robotic additive manufacturing for wind turbine blade repair," *Frontiers in Mechanical Engineering*, vol. 12, 2026. DOI: 10.3389/fmech.2026.1847187 (Cited on pages 51, 53).

[240] I. A. Vishnyakov, A. S. Shinkaruk, E. Y. Narusova, R. H. Rafikov, V. G. Struchalin, and A. A. Rogov, "Development of digital models for the application of additive technologies in the repair processes of rolling stock," in *2026 ElCon Conference of Young Researchers in Electrical Engineering, Automation & Control Systems (ElCon-EE)*, Journal Abbreviation: 2026 ElCon Conference of Young Researchers in Electrical Engineering, Automation & Control Systems (ElCon-EE), Feb. 3, 2026, pp. 208–212. DOI: 10.1109/ElCon-EE69794.2026.11453043 (Cited on pages 51, 53).

[241] N. Ganter, S. Plappert, P. Gembarski, and R. Lachmayer, "Assessment of repairability and process chain configuration for additive repair," presented at the Lecture Notes in Mechanical Engineering, 2022, pp. 261–268. DOI: 10.1007/978-3-030-90700-6_29 (Cited on page 51).

[242] M. Peron, "A digital twin-enabled digital spare parts supply chain," *International Journal of Production Research*, 2024. DOI: 10.1080/00207543.2024.2338878 (Cited on pages 51, 53, 54).

[243] L. Kangju, S. Weitang, L. Yefeng, and Z. Yuan, "Research on self-maintenance strategy of CNC machine tools based on case-based reasoning," presented at the 3rd International Conference on Industrial Artificial Intelligence, IAI 2021, 2021. DOI: 10.1109/IAI53119.2021.9619222 (Cited on page 52).

[244] W. Luo, T. Hu, W. Zhu, and F. Tao, "Digital twin modeling method for CNC machine tool," presented at the ICNSC 2018 - 15th IEEE International Conference on Networking, Sensing and Control, 2018, pp. 1–4. DOI: 10.1109/ICNSC.2018.8361285 (Cited on page 52).

[245] H. Harja, H. Setiawan, Y. Erdani, and M. Febriansyah, "Development of virtual model for cyber-physical screw turbine," presented at the Proceedings of the 2022 IEEE International Conference on Internet of Things and Intelligence Systems, IoTaIS 2022, 2022, pp. 38–42. DOI: 10.1109/IoTaIS56727.2022.9975960 (Cited on page 52).

[246] J. Szpytko and Y. Duarte, "Technical devices degradation self-analysis for self-maintenance strategy: Crane case study," presented at the IFAC-PapersOnLine, 1, vol. 54, 2021, pp. 729–736. DOI: 10.1016/j.ifacol.2021.08.085 (Cited on page 52).

[247] L. Stojanovic and N. Stojanovic, "PREMIuM: Big data platform for enabling self-healing manufacturing," in *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, Journal Abbreviation: 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Jun. 27, 2017, pp. 1501–1508. DOI: 10.1109/ICE.2017.8280060 (Cited on page 52).

[248] K. Sandhya, "Digital twin guided cognitive swarm agents for explainable predictive self-healing in cyber-physical infrastructure," presented at the 2026 11th International Conference on Applying New Technology in Green Buildings, ATiGB 2026, 2026, pp. 193–198. DOI: 10.1109/ATiGB70203.2026.11628273 (Cited on page 52).

[249] E. Rodriguez and A. Alvares, "Toward a digital ecosystem for additive manufacturing driven by standards-based digital thread and digital twins," *IEEE Access*, vol. 13, pp. 207 776–207 807, 2025. DOI: 10.1109/ACCESS.2025.3641205 (Cited on page 52).

[250] O. Alshaheen, Z. Khan, and S. Mekid, "RDM-3d: A remote dashboard for 3d printer performance monitoring," presented at the 1st International Conference on Innovative Engineering Sciences and Technological Research, ICIESTR 2024 - Proceedings, 2024. DOI: 10.1109/ICIESTR60916.2024.10798381 (Cited on page 53).

[251] B. Denkena, P. Nyhuis, B. Bergmann, N. Nübel, and T. Lucht, "Towards an autonomous maintenance, repair and overhaul process: Exemplary holistic data management approach for the regeneration of aero-engine blades," presented at the Procedia Manufacturing, vol. 40, 2019, pp. 77–82. DOI: 10.1016/j.promfg.2020.02.014 (Cited on page 53).

[252] L. Prenzel, A. Zoitl, and J. Provost, "Iec 61499 runtime environments: A state of the art comparison," English, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12014 LNCS, M.-D. R., Q.-A. A., and P. F., Eds., pp. 453–460, 2020, Cited by: 16; Conference name: 17th International Conference on Computer Aided Systems Theory, EUROCAST 2019; Conference date: 17 February 2019 through 22 February 2019; Conference code: 239119; All Open Access, Green Open Access, ISSN: 03029743. DOI: 10.1007/978-3-030-45096-0_55 (Cited on pages 59, 60).

[253] E. Pereira, J. Reis, and G. Gonçalves, "Dinasore: A dynamic intelligent reconfiguration tool for cyber-physical production systems," in *Eclipse Conference on Security, Artificial Intelligence, and Modeling for the Next Generation Internet of Things (Eclipse SAM IoT)*, 2020, pp. 63–71. [Online]. Available: http://ceur-ws.org/Vol-2739/#paper_9 (Cited on pages 60, 62).

[254] C. A. García, E. X. Castellanos, and M. V. García, "Uml-based cyber-physical production systems on low-cost devices under iec-61499," English, *Machines*, vol. 6, no. 2, 2018, Cited by: 23; All Open Access, Gold Open Access, Green Open Access, ISSN: 20751702. DOI: 10.3390/MACHINES6020022 (Cited on pages 60, 62).

[255] S. Bustos-Pulluquitin, G. Caiza, M. Llumitasig-Galarza, M. Castro-Mayorga, C. Sánchez-Benítez, and M. V. Garcia, "Constrained dynamic matrix control under international electrotechnical commission standard 61499 and the open platform communications unified architecture," English, *Sensors*, vol. 23, no. 15, 2023, Cited by: 18; All Open Access, Gold Open Access, Green Open Access, ISSN: 14248220. DOI: 10.3390/s23156919 (Cited on pages 60, 62).

[256] H. da Rocha, R. Abrishambaf, J. Pereira, and A. E. Santo, "Integrating the ieee 1451 and iec 61499 standards with the industrial internet reference architecture," English, *Sensors*, vol. 22, no. 4, 2022, Cited by: 34; All Open Access, Gold Open Access, Green Open Access, ISSN: 14248220. DOI: 10.3390/s22041495 (Cited on pages 61, 62, 93).

[257] OctoPrint Community. "OctoPi: Raspberry Pi image with OctoPrint preinstalled," Accessed: Jul. 27, 2026. [Online]. Available: https://github.com/guysoft/OctoPi (Cited on pages 62, 68).

[258] G. Häußge. "OctoPrint," Accessed: Jul. 27, 2026. [Online]. Available: https://octoprint.org/ (Cited on pages 62, 68).

[259] daGHIZmo. "EEZYbotARM Mk2 – 3D printed robot," Accessed: Jul. 27, 2026. [Online]. Available: https://www.instructables.com/EEZYbotARM-Mk2-3D-Printed-Robot/ (Cited on page 66).

[260] sthone. "Parallel gripper for eezybotarm mk2," Accessed: May 23, 2026. [Online]. Available: https://www.thingiverse.com/thing:4362304 (Cited on page 66).

[261] *PCA9685: 16-channel, 12-bit PWM Fm+ I2C-bus LED controller*, Rev. 4, Data Sheet, NXP Semiconductors, Apr. 2015. [Online]. Available: https://nxp.com (Cited on page 67).

[262] DIGI2-FEUP. "DINASORE function blocks," Accessed: Jul. 27, 2026. [Online]. Available: https://github.com/DIGI2-FEUP/dinasore_function_blocks (Cited on pages 68, 76).

[263] Analog Devices, *Data Sheet ADXL345*, https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf, Revision G, May 2022 (Cited on page 69).

[264] DFRobot, *Analog Piezo Disk Vibration Sensor (SKU:DFR0052)*, https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2096/DFR0052_Web.pdf, 2017 (Cited on page 69).

[265] InvenSense, *PS-MPU-9250A-01: MPU-9250 Product Specification*, https://product.tdk.com/system/files/dam/doc/product/sensor/mortion-inertial/imu/data_sheet/ps-mpu-9250a-01-v1.1.pdf, Revision 1.1, Jun. 2016 (Cited on page 69).

[266] K. Nagimov. "Adxl345spi: Read data from adxl345 accelerometer through spi interface on raspberry pi." version 1.0, Accessed: Sep. 30, 2026. [Online]. Available: https://github.com/nagimov/adxl345spi (Cited on page 69).

[267] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701 (Cited on page 73).

[268] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017 (Cited on page 73).

[269] "IEC 61499-1:2012 — Function Blocks — Part 1: Architecture," International Electrotechnical Commission, International Standard, 2012 (Cited on pages 76, 94).

Index

Numbers
4diac IDE, 4, 27, 57, 66

A
Accelerated degradation, 9, 69, 83, 96, 102
Additive manufacturing, 1, 5, 33, 56, 66, 84, 96, 102

B
Build time, 30, 42, 63, 68, 97

C
Communication
  MQTT, 27, 38, 62, 67, 84, 93, 102
  OPC UA, 27, 38, 57, 77
  publish/subscribe, 27, 62
  quality of service, 29, 96
Cyber-physical production system, 2, 14, 56, 66, 102
Cyber-physical system, 3, 5, 56

D
Digital spare parts, 30, 51, 64, 105
Digital twin, 1, 5, 33, 56, 73, 95, 102
  physical-to-virtual connection, 19, 57
  virtual-to-physical connection, 19, 43, 57
Distributed control system, 2, 5, 33, 56, 99, 102

E
Evaluation
  baselines, 88, 94
  whole sessions held out, 88, 95
Exponential moving average, 71

F
First predicting time, 10, 58, 71, 86, 94
Function block, 3, 23, 59, 66, 91, 96, 102

G
G-code, 62, 68
Gaussian mixture model, 71

H
Health index, 9, 57, 71, 86, 94, 102
Health stage, 9, 71
Human in the loop, 40, 65, 98
Human on the loop, 16, 65, 98
Hyperparameter optimization, 73

I
IEC 61131-3, 24, 47, 61
IEC 61499, 2, 23, 33, 56, 76, 94, 102
Industry 4.0, 1, 5
Industry 5.0, 1, 6, 65, 98, 102

L
Lead time, 2, 11, 37, 63, 98
Level of automation, 15, 44
Literature review, 4, 23, 33, 56, 93

M
Maintenance
  autonomous, 1, 6, 65, 98
  predictive, 1, 7, 33, 66, 84, 94, 102
  prescriptive, 1, 7, 33, 56, 96, 102
  preventive, 7
  total productive, 8

N
Neural networks
  convolutional, 10, 37
  long short-term memory, 10, 37, 73, 88, 94

O
OctoPrint, 62, 68, 84, 96

P
Principal component analysis, 71
Prognostic horizon, 32, 38, 95
Prognostics and health management, 8

R
Raspberry Pi, 43, 60, 66, 93, 102
Raspberry Pi Pico W, 66, 102
Remaining useful life, 9, 36, 57, 72, 84, 95, 102
Research questions, 3, 33
Run-to-failure, 1, 9, 66
Runtime environment, 25, 59
  4diac FORTE, 27, 46, 59, 93
  DINASORE, 60, 66, 93, 102

S
Safety margin, 58
Sensors
  ADXL345, 69, 84, 94
  MPU9250, 69, 86, 94
  piezoelectric, 66, 86, 94
Single point of failure, 2, 17, 62, 94, 105
Slicer, 58, 68, 97

T
Threshold, 11, 36, 58, 72, 83, 95, 102
Time to act, 8, 36, 58, 69, 91, 97

U
Uncertainty, 8, 36, 59, 84, 96, 103