



Adopting IEEE 1451 for the development of a plug&play module for smart sensors in industrial environments

João Maria de Oliveira Torres Pinheiro

Master in Electrical and Computer Engineering

Supervisor: Gil Manuel Magalhães de Andrade Gonçalves

Co-supervisor: Vítor Hugo Machado Oliveira Pinto

October 12, 2022

Abstract

The use of Sensors and Actuators is ubiquitous in an industrial environment. The advent of the Industrial Internet-of-things (IIoT) and the 4.0 industry demands from manufacturers new and more intelligent ways to be able to connect, read and control transducers at the plant floor level.

IEEE 1451 defines a standard for implementing networking modules that allow interaction with smart sensors and actuators. IEEE 1451 also enables smart transducers to supply services such as self-identification, self-calibration, and standardization of data communication.

The adoption of a plug & play module allows the collection of information directly from sensors (temperature from thermometers, the angle from rotary encoders) instead of data (volts, amps, resistance) and, in such a manner, empowers the use of this information to improve process control and the recollection and creation of useful statistics in automation systems.

The literature reviewed in this document presents a lacune in tools and applications of IEEE1451 in the manufacturing industry. It also shows no methodologies for measuring time for plug-and-play integration into IEEE1451-based systems.

The present work consists of the development of a platform for the creation of smart transducer systems associated with a Master's dissertation work. To achieve this objective, it is presented the development of a Transducer Interface Module in line with the IEEE1451 standard. The system developed is characterized by its flexibility, ease of integration, and plug-and-play into a CAN network features.

This document is composed of a background definition, a literature review on the IEEE1451 standard and previous applications of it, an architecture for improved flexibility in transducer-based systems, and the development choices for this particular work as long as the hardware and software decisions made and developed.

Furthermore, this document presents the results of the work, mainly some working examples and possible applications, the performance of the system on some important aspects, and the speed of plug-and-play when connected to a system. It also intends to contribute to the state-of-the-art by proposing a new methodology to measure integration time into a system, along with proposing a way to forecast integration time in IEEE1451 systems.

It is concluded that the presented system, although still not fully compliant with IEEE1451.0 presents advantages for manufacturing system integrators and is set to become a tool on which transducer systems can be based on since it has the objective to be released as an open-source development platform.

Resumo

O uso de Sensores e Atuadores é predominante nos ambientes industriais. O surgimento da Internet das Coisas Industrial (IIot) e da Indústria 4.0 solicita, por parte dos construtores de sistemas, novas e mais inteligentes maneiras de ligar, ler e controlar transdutores ao nível do chão de fábrica.

O standard IEEE1451 parametriza um conjunto de regras e estruturas para implementar módulos com capacidade de comunicação que permitiam a interação com sensores e atuadores inteligentes. O standard IEEE1451 também habilita os transdutores inteligentes para fornecerem serviços como auto-identificação, auto-calibração e standardização da comunicação de dados numa rede.

A utilização de um módulo plug-and-play permite a colheita de informação diretamente dos sensores (temperatura de termómetros, ângulos de encoders rotativos) ao invés dos dados puros (tensões, correntes, resistências) e, dessa forma, permitir o uso desta informação para melhorar o controlo de processos e a colheita e criação de estatísticas úteis em sistemas de automação.

Uma inicial revisão de literatura, apresentada neste documento, demonstra uma lacuna de ferramentas e aplicações do standard IEEE1451 na indústria de manufactura. Também falha em apresentar metodologias para definir o tempo de integração plug-and-play para sistemas baseados no standard IEEE1451.

O trabalho aqui apresentado consiste no desenvolvimento de uma plataforma para a criação de sistemas de transdutores inteligentes e está associado a uma dissertação de grau de Mestre. Para se atingir este objetivo, apresenta-se o desenvolvimento de um Módulo de Interface de Transdutores alinhado com o standard IEEE1451. O sistema desenvolvido caracteriza-se pela sua flexibilidade, facilidade de integração e capacidade de plug-and-play em redes CAN.

O documento presente é composto por uma definição de contexto de trabalho, uma revisão de literatura sobre o standard IEEE1451 e aplicações publicadas do mesmo, uma apresentação de uma arquitetura para flexibilidade aumentada em sistemas baseados em transdutores inteligentes, as escolhas quanto à plataforma de desenvolvimento bem como decisões e trabalhos relativos à implementação quer em hardware quer em software.

Além disso, este documento apresenta os resultados deste trabalho, bem como alguns exemplos funcionais e possíveis aplicações destes, a performance do sistema em aspeto fulcrais e a velocidade de plug-and-play quando conectado a uma rede. Também pretende contribuir para o estado da arte através da proposta de uma nova metodologia para medir o tempo de integração num sistema, propondo uma técnica para prever o tempo de integração em sistemas IEEE1451.

Conclui-se que o sistema apresentado, embora ainda não totalmente em conformidade com o standard IEEE1451.0 apresenta desde já vantagens para os produtores de sistemas de manufactura existindo a possibilidade de se tornar numa ferramenta sobre a qual sistemas de transdutores podem ser desenvolvidos, uma vez se pretende ser publicada sob a forma de uma plataforma open-source.

Nota: Esta dissertação insere-se nos trabalhos do laboratório DIGI2 @ FEUP.

Acknowledgments

(In Portuguese)

Acredito que somos resultado do que nos rodeia e pelos que por nós se cruzam. Por isso, não podia não deixar de agradecer a quem me acompanhou no meu percurso académico, que visou terminar agora.

À Faculdade de Engenharia enquanto conjunto de pessoas que, em conjunto, tornam possível o acesso à informação, o crescimento académico e pessoal e o potenciamento dos talentos individuais de cada aluno. Muito do trabalho é completamente invisível aos olhos de um comum aluno mas não pode nem deve passar incólume de agradecimento.

À comunidade académica docente que me acompanhou nestes cinco anos e me passou conhecimento e método que conto que me sejam muito úteis em anos vindouros. É graças a quem se dedica a ensinar que existe a possibilidade de ver a Engenharia a crescer dia a dia. Nem sempre terá sido a relação mais fácil, mas espero que tenha sido proveitosa para ambas as partes.

Ao meu orientador, o Professor Gil Gonçalves, agradeço pela confiança que depositou em mim tanto na área da investigação como na dissertação que sucede. Agradeço também pela disponibilidade e apoio que me forneceu no desenvolvimento e escrita desta dissertação.

Ao meu co-orientador, o Professor Vítor Hugo Pinto agradeço também a disponibilidade, o apoio e o dinamismo que me transmitiu e que facilitaram o meu trabalho durante esta dissertação. Por alguma razão engenho faz parte da palavra engenharia. Obrigado.

Ao Professor Luís Neto, pelo apoio fundamental que me deu na realização deste trabalho. Tanto na parte prática como na parte teórica este trabalho não teria sido possível sem o apoio do Professor. Também a nível pessoal não podia deixar de agradecer pelas muitas horas de trabalho juntos e que foram, sem dúvida, de alta qualidade, ânimo e diversão. Aproveito e deixo também uma nota de agradecimento à gastronomia portuguesa que, em bons momentos, partilhamos.

Aos membros do DIGI2, pela simpatia e predisposição para ajuda sempre que necessitei.

À minha família, pelo apoio incondicional que me deu durante estes 5 (ou talvez 23) anos. Se somos um produto de alguém, eu sou um produto vosso. Muito obrigado por todo o apoio e por todo o sacrifício que fizeram por mim. Espero deixar-vos orgulhosos.

Aos meus amigos, aqueles que me acompanharam no dia-a-dia durante 5 anos, agradecer por todos os momentos de gargalhada e diversão que tivemos. Uma palavra de apreço pelo tempo que dispenderam comigo e espero que tenha valido a pena para vocês também. Foram 5 anos intensos, cheios de momentos que levo para a vida. Foi, sem dúvida, mais fácil ao vosso lado.

À minha namorada, Carol. Obrigado por todo apoio e por tudo o que me deste durante este percurso. Obrigado também pela motivação que me deste para continuar, mesmo nos momentos mais difíceis deste trabalho.

João Pinheiro

*“ Para ser grande, sê inteiro: nada
Teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.”*

Fernando Pessoa

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Motivation and Proposal	3
1.3	Document Structure	4
2	Background and Bibliographic Review	5
2.1	Introduction	5
2.2	Historical context	6
2.2.1	Smart Sensors	6
2.2.2	The advent of IEEE 1451	7
2.3	Technical Overview of IEEE 1451	9
2.3.1	Introduction	9
2.3.2	NCAP - Network Capable Application Processor	9
2.3.3	TIM - Transducer Interface Module	10
2.3.4	TII - Transducer Independent Interface	10
2.3.5	NI - Network Interface	10
2.3.6	Simplified Architecture	11
2.3.7	Conclusions	12
2.4	Sensorization and Actuation solutions using IEEE 1451	13
2.4.1	Review on the applications of IEEE 1451	13
2.4.2	Presented references analysis	16
2.5	A look into the future for IEEE 1451 - IEEE 21451	18
2.5.1	Perspectives on the future for Smart Sensors and Cyber-Physical Systems	20
2.6	Conclusions on the Bibliographic Review	22
3	Smart Transducers - Requirements, Reference and Proposed Architecture	23
3.1	Introduction	23
3.1.1	Motivation	23
3.1.2	Contextualization	23
3.1.3	Architectural Goals	24
3.2	IEEE1451.0 Reference Architecture	25
3.2.1	Transducer Channels	26
3.2.2	Transducer Electronic Datasheets (TEDS)	26
3.2.3	Commands and Messages	27
3.2.4	Requirements for compliancy	28
3.3	Architecture proposal	29
3.3.1	NCAP	29
3.3.2	TII (Communications Module)	30

3.3.3	TIM Backplane	30
3.3.4	TIM Cards	30
3.3.5	Backplane to Card Interface	31
3.3.6	Advantages from the Transducing System Architecture	31
3.4	Summary and Conclusions	31
4	Implementation platform	33
4.1	Introduction and Contextualization	33
4.2	Hardware	34
4.2.1	Micro-controller: TM4C123XL Platform	34
4.2.2	Development Shield	35
4.3	Soft/Firmware	37
4.3.1	Programming Language Choice	37
4.4	TI-RTOS	37
4.5	Development Tools	38
4.5.1	Code Composer Studio IDE	38
4.5.2	Code Management - Git	38
4.6	Summary and Conclusions	39
5	Implementation - Design and Structure	41
5.1	Introduction and Contextualization	41
5.2	IEEE1451.0 Standard - System specification	42
5.2.1	Args Module	43
5.2.2	Util Module	45
5.2.3	ModuleCommunication Module	46
5.2.4	TIM Services Module	48
5.3	IEEE1451.0 TIM Services	50
5.4	IEEE1451.X Communication Module : CAN	53
5.4.1	ISO-TP - Transport Layer for CAN	53
5.4.2	CAN Layer	56
5.5	Summary and Conclusions	59
6	Sensor and Actuator Integration Results	61
6.1	Introduction and Contextualization	61
6.2	Working Examples	61
6.2.1	One NCAP-to-One TIM-to-One Sensor	62
6.2.2	One NCAP-to-One TIM-to-Many Sensors	63
6.2.3	One NCAP-to-Many TIMs-to-One Sensor	64
6.3	TEDS measurements	65
6.3.1	TransducerChannel Update Time / Sampling Period	66
6.3.2	TransducerChannel read delay time	66
6.3.3	TransducerChannel warm-up time	68
6.4	Handshake Timing Analysis	68
6.4.1	TIM Announcement	68
6.4.2	TIM Discovery	69
6.5	TEDS readout timings	69
6.5.1	Meta-TEDS	69
6.5.2	Transducer Channel TEDS	69
6.6	Integration Timing - Methodology proposal	70

6.7	Summary and Conclusions	72
7	Future work	73
7.1	TIM Cards	73
7.2	TEDS programmer	74
7.3	Implementation of new commands	74
7.4	Improvements on the internal functioning of the TIM	74
8	Conclusions	75
8.1	Scientific Contributions	76
	References	77

List of Figures

1.1	Data vs Information	2
1.2	Edge-oriented automation architecture [1]	2
2.1	Traditional analog sensor vs Smart Sensor [2]	7
2.2	Basic architecture for a distributed control loop based on smart transducers . . .	11
2.3	Wastewater treatment process automation using IEEE 1451[3]	13
2.4	Vehicular Sensor Network Architecture Compliant with IEEE 1451 Standards[4] . . .	15
2.5	Basic smart transducer architecture used in [5]	16
2.6	IEEE 21451 smart transducer architecture[6]	19
2.7	Smart transducer architecture presented by [7]	20
2.8	Example of IEEE 1451-based CPS [8]	21
3.1	IEEE1451.0 reference model [9]	25
3.2	IEEE1451 command structure[9]	28
3.3	IEEE1451 response structure[9]	28
3.4	IEEE1451 TIM-initiated message[9]	28
3.5	Architecture Model Proposal, built around IEEE1451.0's reference model [9] . .	29
3.6	Graphical Representation of an Implementation of the presented Architecture . .	32
4.1	Texas Instruments TM4C123GXL Development Platform	34
4.2	CAN shield	35
4.3	Board Layouts	36
4.4	Final PCBs	36
5.1	TIM architecture implementation UML class diagram	42
5.2	Argument Class	44
5.3	Argument Array (as an object and as a byte stream)	44
5.4	ModuleCommunication classes (implemented classes highlighted)	46
5.5	Message Reception Sequence Diagram	48
5.6	TIM Services Class Diagram (implemented classes in green)	49
5.7	Sequence Diagram Command Handling: Receive	50
5.8	Sequence Diagram Command Handling: Handle	51
5.9	Sequence Diagram Command Handling: Respond	52
5.10	ISO-TP message flow [10]	54
5.11	Handshake Protocol for ID attribution and discovery	56
5.12	Sequence Communication Diagram for the Handshake Protocol	57
6.1	One-to-One-to-One System	62
6.2	CAN Messages Sequence Between TIM and NCAP	63

6.3	CAN Messages for TransducerChannel DataSegment Readout	64
6.4	One-to-One-to-Many System	64
6.5	One-to-Many-to-One System	65
6.6	TransducerChannel Read Delay Time Measurement	67
6.7	Transducer warm-up time as defined per manufacturer [11]	68
6.8	Time for TIM-initiated handshake (accounting for communication)	69
6.9	Time for Meta-TEDS readout(accounting for communication)	70
6.10	Time for TransducerChannel-TEDS readout(accounting for communication) . . .	70
6.11	Integration time of one-to-one-to-one system	71
7.1	Buffering Stage of TIM Cards initial developments	73

List of Tables

2.1	Summary on the usage of IEEE 1451	16
3.1	Meta-TEDS Interpreted	27
4.1	Example of interface functions definition using IDL [9]	37
5.1	IEEE1451 Data Types	43
5.2	Codec functions (simplified)	45
5.3	NetComm Communication Methods	47
5.4	ISO-TP message structures	54
5.5	Can Message Object Structure	57
6.1	Estimation of Integration Time for different scenarios	71

Abbreviations and Symbols

ADC	Analog to Digital Converter
CAN	Controller Area Network
CANOpen	Controller Area Network Open
CPS	Cyber-Physical System
DAC	Digital to Analog Converter
DAQ	Data Acquisition System
ERP	Enterprise Resource Planning
EtherCAT	Ethernet for Control Automation Technology
HVAC	Heating, Venting, and Air Conditioning
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ICPS	Industrial Cyber-Physical System
IIoT	Industrial Internet of Things
IIRA	Industrial Internet Reference Architecture
I/O	Input Output
IoT	Internet of Things
JSON-LD	Javascript Object Notation-Linked Data
KPI	Key Performance Indicator
LoRA	Long Range
LPWAN	Low Power Wide Area Network
MDPI	Multidisciplinary Digital Publishing Institute
MES	Manufacturing execution systems
Modbus	Modicon Bus
MQTT	Message Queuing Telemetry Transport
MTTR	Mean Time To Repair
NB-IoT	Narrow Band Internet of Things
NCAP	Network Capable Application Processor
NI	Network Interface
OPC-UA	Open Platform Communications - Unified Architecture
P2P	Point to Point
PCB	Printed Circuit Board
PoE	Power Over Ethernet
PLC	Programmable Logic Controller
RAMI	Reference Architecture Model for Industry 4.0

SNMP	Simple Network Management Protocol
SPI	Serial Peripheral Interface
(S)TIM	(Smart) Transducer Interface Module
TEDS	Transducer Electronic Datasheet
TII	Transducer Independent Interface
UART	Universal Asynchronous Receiver-Transmitter
VSN	Vehicular Sensing Network
WSN	Wireless Sensor Network
XMPP	eXtensible Messaging and Presence Protocol

Chapter 1

Introduction

1.1 Introduction

Sensing and actuation define the interface between a physical process and control engineering. Being capable of precisely collecting information from a process (Sensing) and precisely inputting physical quantities into a process (Actuation) has a definite impact on the control methods used on manufacturing systems. This allows for higher precision techniques and better control over the variables of a system. Control engineering has evolved over the years providing the industry with more and better solutions to control processes from this standpoint. With every iteration of control systems, the number of sensors and actuators in a loop tends to increase [12]. The more information a controller can collect and directly influence the better and more elaborate its process decision tends to be. Sensors and actuators have become better over the years, many of them already containing microcontrollers inside although the way to insert them into a system is, many times, still antiquated.

A problem then arises: how to easily attach transducers (sensors and actuators) to a control loop while enabling them with smart features, the main question this dissertation work verses on.

Another problem stands in the time it takes for someone upgrading a system to integrate a new transducer into an existing system, another problem this dissertation work pursues.

The early electrical control loops solved this problem by hard-wiring a direct connection between a controller inputs/outputs (I/O) and a transducer. Later, the introduction of fieldbuses and networking in control loops created the possibility of having transducers distributed around a plant floor using communications protocols such as Modicon Bus (Modbus), Controller Area Network Open (CANOpen), or Ethernet for Control Automation Technology (EtherCAT). This approach allowed the reduction of points of failure in a control system. Having a solid networking backbone can provide a stable basis for implementing redundancy over an automation system. From a logical standpoint, each connection between the Programmable Logic Controller (PLC) and a transducer is still implemented as a direct connection, which means raw data is conveyed throughout the network.

Probing an assorted package of data in the network is useless as nothing can be inferred from it. The fourth industrial revolution has brought this problem to a new stage: using information as a basis for control decisions. Information is a concept defined as the bringing of meaning over data [12]. Having an I/O card collecting voltage or current values (data) is useful if it is possible to translate them into physical values (information) (see fig. 1.1).

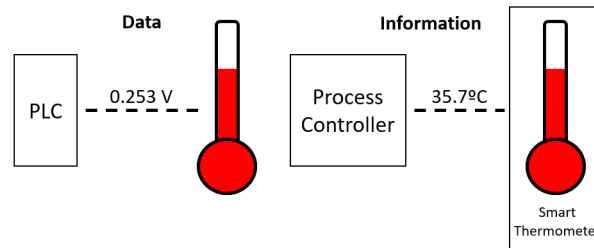


Figure 1.1: Data vs Information

Cyber-Physical Systems (CPS) (also known as the drivers of Industry 4.0) introduce a new concept, the distribution of processing over the network (see fig. 1.2). In the manufacturing industry, the processing over a network requires the utilization of processing-capable components over the shop floor, which are denominated as edge components, as can be seen in the figure. Providing the loop with edge computing components capable of processing data into information and communicating it over the network can be immensely advantageous:

- The control loop can be distributed over edge-oriented nodes
- The sensing and actuation can now be provided as a service from the node which drives the transducer
- Information flows over the network instead of data
- Multiple systems (Manufacturing execution systems (MES), Enterprise Resource Planning (ERP) systems, Condition Monitoring Systems) can obtain and provide information directly to the control loop components

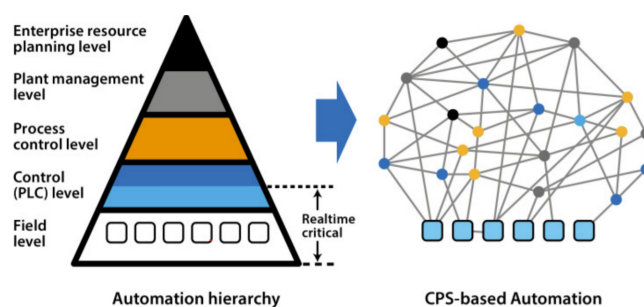


Figure 1.2: Edge-oriented automation architecture [1]

Building upon distributed systems communications and utilizing the ability to convey information over the network, information about the transducer itself can also be carried over. An IEEE 1451 smart transducer should be able to identify itself, transmit what it's capable of (e.g. sense, move, calibrate), its manufacturing characteristics (e.g. precision, accuracy, range of movement), and how it will communicate information. Using self-identification, self-calibration, and standardizing of communications, smart transducers networks can be created. Implementing a control loop as a network of smart transducers largely resembles an industrial internet-of-things network, using multiple nodes cooperating towards a goal: better and more integrated manufacturing.

1.2 Motivation and Proposal

Industry 4.0 systems need ubiquitous usage of data in order to develop what are considered to be intelligent systems. The newest trends in research (machine learning techniques (deep learning, neural networks) and big data techniques) require the capability to sense data from machines, manufacturing systems, tools, and even operators. The use of Industry 4.0 manufacturing architectures, such as the Reference Architecture Model for Industry 4.0 (RAMI 4.0) [13], motivate this newest research trends while aiming to improve and upgrade manufacturing processes through more connected and intelligent shop floors.

At the same time, the current production lines are filled with expensive, precise, perfectly functional sensors and actuators, but which are not equipped with the communication and data-handling capabilities newer technologies need. Throwing these transducers away would not be economical nor ecologically desired.

From other perspective, not considering already existing assets, developing systems based on sensors and actuators is a ubiquitous practice in electrical engineering. Sensors and actuators manufacturing is an enormous industry, providing system developers with immense variety over technologies and techniques for sensing and manipulating processes. A big challenge developers face is how to integrate this sensors into control loops. Smart Transducers based on IEEE1451 help solving this problem, facilitating the process of introduction and upgrade of transducers into applications, through plug-and-play capabilities.

From the research made, there is not, to the authors knowledge, an open source tool which can help develop smart transducers, in order to facilitate such said integration.

The main objective of this dissertation work is to develop an open source platform for converting normal sensors and actuators into Smart Transducers based on IEEE 1451. It shall be capable of interfacing with any transducer to collect and deploy data, convert it to information and communicate it to a network in a real industrial use-case scenario.

This objective is achieved by the development of a Smart Transducer Interface Module responsible for sampling and conditioning signals for both digital and analog sensors. Another objective is to implement the system in a way that makes the introduction of new transducers into a control loop trivial and easily usable in real-life scenarios.

As an example, a scenario is set. Please consider an industrial machine that has an internal system that measures temperatures. Now imagine the temperature sensor broke, and there is no possible way to obtain a similar one. Or imagine there's the need for implementing another sensor to have redundancy, or even imagine there's the need to add another temperature sensor to another point in the system. Usually, any of these problems would present a serious problem to any system developer or maintenance tech. With plug&play capable smart transducers, any of these problems will be solved with relative ease, saving money for companies and time for system integrators and maintenance technicians while developing better and more robust control loops.

1.3 Document Structure

After the introduction chapter, seven other chapters follow.

In chapter 2, an historical background on IEEE 1451 is set. Next, a section is dedicated to explaining the structure of the standard and the main components needed for a correct implementation.

Afterward, still in chapter 2, a literature review is presented, versing on the work already developed around the standard, including previous implementations of the standard in different use-cases. This is followed by a review of the standard update, which is still in the project phase.

Then, in chapter 3, there is the definition of an architecture for the system developed. The motivation behind the design decisions is noted, the goals the architecture intends to meet are defined, and contextualization for work is made. After that, the IEEE1451.0 standard reference model is analyzed, and its main objects of work are explained. Regarding all the points above, a final architecture is proposed and described.

Ensuing, chapter 4 details the implementation platform used, the hardware and software choices are depicted and the tools used are enumerated. Please note the development of a PCB board to meet industrial capability.

Finalizing the development chapters, chapter 5 goes about the software developed for this system. It details the modules, the techniques, and the most important aspects regarding the module's operation and its specificities.

Evaluating the quality of the work, chapter 6 presents use-cases for the system, measurements made to its performance, and helps characterize the system through quantitative tests. Here, some advantages of using this system are evaluated.

As there is never a finite development a system can get, only a limited development time, some future work objectives are drawn in chapter 7.

Finally, in the last chapter 8, conclusions are drawn about the work presented.

Chapter 2

Background and Bibliographic Review

2.1 Introduction

Sensorization and actuation is one of the most studied areas in electrical engineering. Physical technologies can only go so far as the ability to probe data and manipulate data goes. Organizations such as the Institute of Electrical and Electronics Engineers (IEEE), Multidisciplinary Digital Publishing Institute (MDPI), Elsevier, and the International Electrotechnical Commission (IEC) dedicate entire committees and journals to the study and development of sensorization and actuation solutions.

At an industrial level, sensors and actuators are ubiquitous. Process control depends almost entirely on the ability to probe the status of one ongoing process, decide the course of action using the acquired data, and apply the decision to the process in order to steer it accordingly.

Traditionally, industrial controls solutions have been presented as centralized solutions around programmable logic controllers (PLC) [14]. The development of more complex and computationally heavier control techniques moved the control paradigm into distributed systems in order to conjoin efforts between sparsing processing nodes. This transition has led to the advent of Cyber-Physical Systems which are shaping a new Industrial Revolution into the 4.0 Industry [1].

As important as sensing and actuating is the ability to receive the probed data and define the physical quantity something needs to be actuated with. Traditionally, sensors and actuators were considered to be the endpoints of centralized systems. One sensor or one actuator belonged to a control loop and was isolated from the network, i.e, there was no direct access to sensors or actuators. A sensor was seen as one direct wire from which an acquisition card connected to a PLC probed data from. Smart sensors come into play in a conjoined effort to change this paradigm.

In order to fully comprehend the motivation behind the creation of the IEEE 1451 standard, a historical overview of the origins of the standard was made.

2.2 Historical context

2.2.1 Smart Sensors

The term *smart* can be employed in many different ways. In the technology industry, it is normally presented as a marketing term, which makes it vague and unclear. *Smart* tends to mean too many different things, so, to further define what a smart sensor or actuator means in this context it is important to review the history of smart transducers and look for the origins of this term.

2.2.1.1 A broad definition

In 1980, Breckenridge et al. [15] reviewed the meaning of the term smart sensors. At the time, 3 groups of authors [16, 17, 18] presented three interpretations for what a smart sensor should be, all around the same concept, extracting higher quality information from sensors. Tao et al. [16] define a smart sensor as a sensor that contains both sensing and signal processing capabilities. Wiener [17] looks at smart sensors as those which fit the best the application intended, acquiring information from sensor readout and having only as much accuracy as needed (military background with optimization of resources in mind). Breckenridge et al. [18] present the last definition of smart sensors as being capable of doing more than just acquiring data. The concept of a sensor as a service-providing endpoint is introduced here (providing calculations over information acquired from the sensor). Interoperability is also introduced by [18] stating that sensors should exchange data and interact to achieve a goal. One interesting statement in [15] is that the industry will ultimately define what a "smart sensor" is.

2.2.1.2 An industrial look at smart sensing

In 1987, the industrial application of the term 'smart sensor' was again reviewed by Wang et al. in [2]. The field definition referred to by Breckenridge et al. was confirmed. The industry defined the concept of the smart sensor or transducer as one electronic body composed of transducers, signal conditioners, an inner computation capability, a communication interface, and self-identification features. The idea of having one integrated sensing or actuating unit, with communication capabilities behaving like an entity created the possibility to see transducers as services. One can communicate with the sensor to fetch data or one can communicate with the actuator to change a physical quantity.

New advantages for smart sensors are also introduced: maintenance modes for sensors, providing automatic self-check-ups; self-diagnostic communication to a monitoring system; remote configuration of sensors and grouping of multiple sensors in the same board (which is oriented in the same manner as a TIM - section 2.3.3).

A preliminary architecture for smart sensors is introduced by Wang et al. in fig. 2.1.

Continuing on the advantages and possibilities smart sensors have to offer the industry, Smith et al. [19] discuss the integration of smart sensors in 3 industries: process manufacturing, automotive control, and aerospace sensing. Some new aspects are presented such as cabling reduction

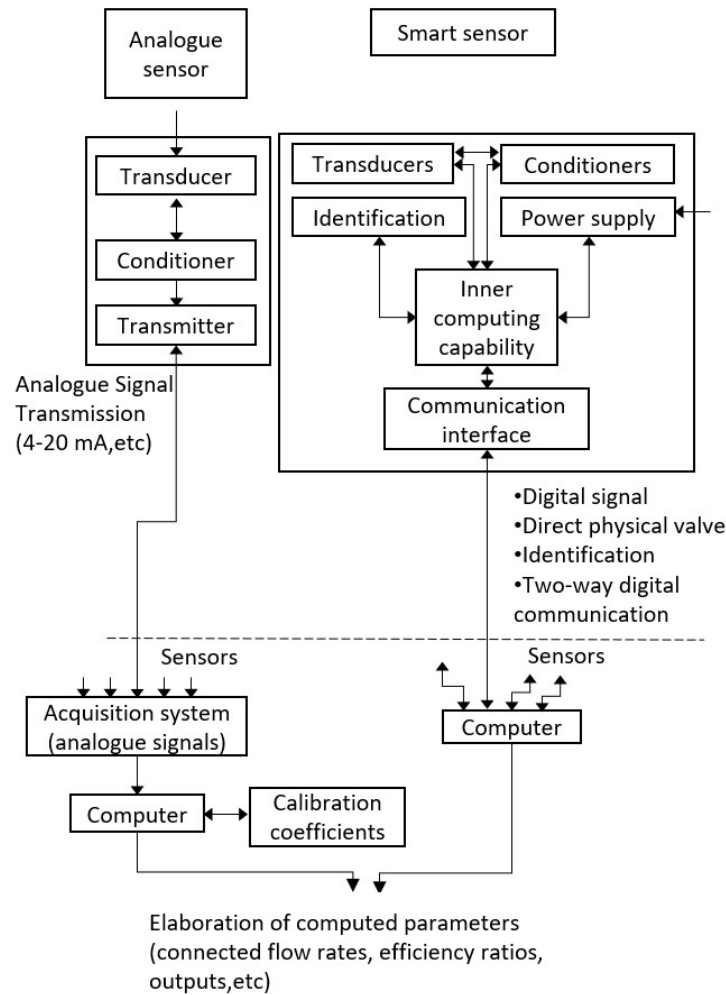


Figure 2.1: Traditional analog sensor vs Smart Sensor [2]

through the use of multiplexed communication protocols (fieldbuses) and reduction of installation costs caused by the automatic configuration these sensors tend to offer. At this stage, smart sensors were still seen as entities dependent on the communication media used and so, Smith et al. present this as one big drawback of smart sensors: deprecation by change of fieldbuses technology.

To mitigate this problem a new group of standards was created: IEEE 1451 [9, 20, 21, 22, 23, 24, 25].

2.2.2 The advent of IEEE 1451

After the introduction of smart transducers¹ the market for these kinds of transducing systems gained traction reaching \$500 million in 1995 [26]. This market explosion led to the development of multiple platforms for deploying transducer networks such as *LonWorks*TM by Echelon[27],

¹Smart sensors and smart transducers merge in literature. In essence, they mean almost the same but smart transducers allow for measurement and actuation whereas smart sensors only account for measurement

Control Area Network (CANTM) by Bosch and Intel[28], and CANTM derivatives *DeviceNet*TM by Allen Bradley[29, 30] and *Smart Distributed System* by Honeywell[31, 30].

Having multiple companies developing standards for transducer communication leads to technology isolation, multiplying costs for development, installation, maintenance, and upgradeability of old systems. Having one standard to rule all transducer to network activity would not only lower the development cost of the transducer networking solutions but also allow for sensor and actuator manufacturers to develop network-universal sensors and actuators diverting the network development funds to R&D efforts to improve them [26].

With this concept in mind, the author of [26], Janusz Bryzek, proposed the development of such a standard to IEEE Sensor Technology committee (TC-09) in 1993 being accepted and formalized as Project P1451 in 1994.[32]

The main objectives of P1451 were set[32]:

- Define a uniform approach for connecting transducers to control or field networks
- Facilitate transducer interoperability across different networks and thus achieve network independence
- Reduce the industry's effort to develop and migrate to networked smart transducers

From 1997 to 2010 IEEE released 7 standards being the most recent versions of them: IEEE 1451.0-2007[9], 1451.1-1999[20], 1451.2-1997[21], 1451.3-2003[22], 1451.4-2004[23], 1451.5-2007[24] and 1451.7-2010[25].

Project P1451.6 started in 2004 but never yielded any standard and has been abandoned.

2.3 Technical Overview of IEEE 1451

2.3.1 Introduction

IEEE 1451 presents the concept of an IEEE 1451 compliant smart transducer as a smart transducer that has more functionalities than those needed to generate a representation of a sensed or controlled quantity. A compliant transducer should provide functions such as self-identification, self-description, self-diagnosis, self-calibration, location-awareness, time-awareness, data processing, reasoning, data fusion, alert notification (report signal), standard-based data formats, and communication protocols [33]. These functions allow the transducer to behave in a plug&play fashion. Once connected to the network, the transducer should be capable of identifying itself and presenting its capabilities to other nodes. This ability is provided via a Transducer Electronic DataSheet (TEDS). The TEDS data structure contains information about a transducer much like an ID card, storing information such as manufacturer data and characteristics, measurement range, accuracy, and calibration data. This information is then provided when the handshake between the transducer and other nodes occurs and communication is set.

IEEE 1451 also defines the architecture for the smart transducer. This architecture is divided into two main modules: the Network Capable Application Processor (NCAP) and the Transducer Interface Module (TIM). The communication between the NCAP and the TIM is made via a Transducer Independent Interface (TII) and between the NCAP and the network via the Network Interface (NI). The specific details and functions of each block are specified below.

2.3.2 NCAP - Network Capable Application Processor

The NCAP is responsible for providing smart transducers with higher-level application processing and general application networking capabilities. It's responsible for the creation and consumption of every piece of information from and to the smart transducer. It behaves like a network node and provides the network with services such as 'read the sensor X's value', 'move the actuator Y 10mm' or 'get TEDS'. To ensure the correct measurement or actuation, the NCAP must be able to communicate with the TIM via read/write operations. An example of an NCAP implementation might be a single board computer running an Open Platform Communications - Unified Architecture (OPC-UA) server providing methods for reading the temperature in an industrial oven. An NCAP can be responsible for only one TIM or multiple TIMs, depending on the implementation of the TII it uses (see 2.3.4). The NCAP can also utilize its processing capabilities to provide more complex methods to other nodes (e.g. value integration, rolling average, threshold surpass counter).

2.3.3 TIM - Transducer Interface Module

The TIM² is responsible for the interface between the sensor/actuator and the NCAP. Its main function is to sample the data sensed or generate the output needed for the actuation. The TIM is also in charge of storing the TEDS and providing it to the NCAP when required. As the TIM interfaces directly with analog signals from and to the transducer, it must contain a way to convert digital to analog values. ADCs, DACs, and Digital I/O are the ways to do this. Being able to create different TIMs custom designed for each application gives the engineering teams a very powerful tool to create smart transducers from scratch. Smart transducers have few limits other than physical ones. If there is a need for a TIM with 3 10bit ADCs, 2 8bit DACs, 1 24bit ADC such TIM can be created using this platform. The only requirement is that the TIM's TEDS reports correctly what is present in the system. Once the TEDS is communicated to the network, the data sent by the smart transducer is automatically contextualized and becomes information ready to be processed by other CPS nodes. TEDS from one TIM can hold information of up to 255 sensors or actuators.

2.3.4 TII - Transducer Independent Interface

The TII defines the interface between the NCAP and the TIM. This interface is responsible for conveying the data collected by the TIM to be processed by the NCAP and the actions received by the NCAP to be carried out by the TIM. The IEEE 1451 standard provides various possibilities for the TII implementation, allowing for vast use-cases of these interfaces. In [33], TII is presented as: a SPI interface (IEEE 1451.2); a multi-drop communication transducer network (IEEE 1451.3); a wireless interface (e.g 802.11 - Wifi, 802.15.1 - Bluetooth, 802.15.4 (Zigbee), 6LowPAN) (IEEE 1451.5); a high-speed CANopen network (P1451.6); a RFID system (IEEE 1451.7). Each of these standards may be applied according to the application needed. Standardization of such an interface is important as it permits NCAPs and TIMs from multiple vendors to be interconnected. This feature allows networking-focused companies to work solely on NCAPs and Signal Acquisition and Conditioning companies to work only on TIMs.

2.3.5 NI - Network Interface

The NI defines the interface between the NCAP and the network holding other nodes. IEEE 1451 does not define any restrictions to this interface and makes it dependent on the control loop network implementation. An example of this implementation may be an OPC-UA communication protocol running over TCP/IP on an Ethernet-based physical network.

²Transducer Interface Module(TIM) and Smart Transducer Interface Module(STIM) are used interchangeably in literature

2.3.6 Simplified Architecture

In figure 2.2 a simplified architecture based on IEEE 1451 is proposed. It presents a simple control loop containing two Smart Transducer Modules and a logical process controller (e.g PLC). The module on the left presents multiple connections to sensors and actuators, making it behave like a custom I/O card. Such a solution could be implemented to install transducers in isolated parts of a machine. Having the signal conditioning and acquisition closer to the source of the signal reduces signal interference caused by electromagnetic noise present on a plant floor. The transport of digital information is also more robust than analog data. Furthermore, by only using one cable to connect the NCAP to the network, the quantity of cabling is reduced, an ecological and economically desired progress.

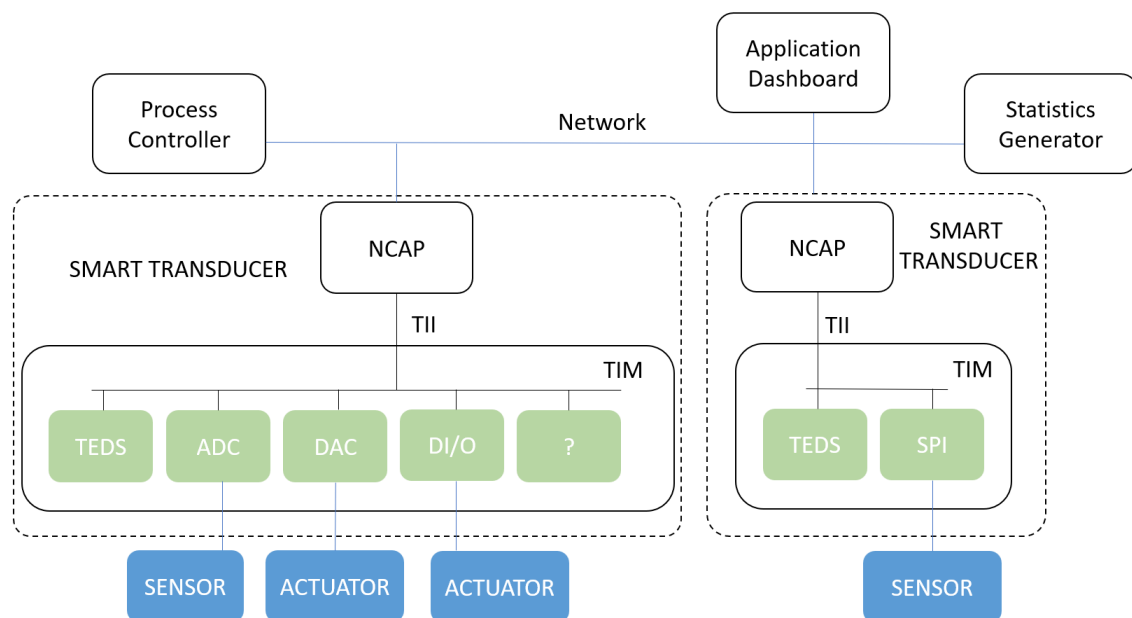


Figure 2.2: Basic architecture for a distributed control loop based on smart transducers

The module on the right presents a situation where a simple transducer is connected to the network. However, including an NCAP responsible for this transducer allows a more complex oversight of this component (e.g. condition monitoring on bearing-based motors). Another possibility not depicted in fig. 2.2 is having multiple TIMs connected to one NCAP as mentioned in subsection 2.3.4. The distribution of smart transducers over an edge-component-based network can turn the sampling of sensors into more elaborate functions, allowing the integration of newer techniques to be used in distributed control loops (e.g. machine learning, data mining, condition monitoring). Such is the example included in the figure, including the introduction of an Application Dashboard and a Statistics generator.

2.3.7 Conclusions

The advantages of using IEEE 1451 are well-cited above: isolation of responsibilities, standardization of interfaces, the modularization of I/O systems, and giving processing power to the edge level of a system. IEEE 1451 also eases the scalability and upgradeability of an automation system by making the nodes easily integrable into the control loop.

Having self-description and self-identification helps the development of control software as it gives the programmer clear and technical instructions about the capabilities of the transducers installed and what they can do. This group of features enables system designers to move one level up the abstraction pyramid. Sensors and actuators no longer need to be thought of as physical components. These are transformed into services that logic controllers and other CPS applications can use.

By abstracting systems developers from the physical implementation of transducers, higher-level concepts can more easily be deployed into networks. Techniques like machine learning or data analysis also largely benefit from having access to endpoints providing structured data (information) [34].

The use of network-capable nodes also brings advantages for the interoperability of systems. An example of this might be a logical process controller and a condition monitor, using the same sensor concurrently to fulfill their objectives. The openness provided by the distributed implementation facilitates resource sharing.

2.4 Sensorization and Actuation solutions using IEEE 1451

IEEE 1451 is more present in literature in an application way, meaning, IEEE 1451 is used more as a means of obtaining robust integrated sensors and actuators other than being studied itself. This fact is relevant and important, as it shows this standard is not dead, as it's being utilized and refined over time.

The current literature review stands on the developments made using the original IEEE 1451 standards, released from 1997 through 2010. It reviews articles in various different applications in order to fetch the most relevant requirements to be implemented in this dissertation work.

2.4.1 Review on the applications of IEEE 1451

IEEE 1451 has been used as a way to implement smart transducers to the most various field of application, extending to all kinds of industries. A literature review was conducted to understand what has been made using the IEEE 1451 standard, the technologies used, and which IEEE 1451 features are more implemented.

In [35], Ulivieri et al. make use of the plug&play and self-configuration capabilities of IEEE 1451.4 to create one artificial "nose" to sense different gases. The self-configuration capabilities allowed Ulivieri et al. to create a sensor array in which different resistive chemical sensing sensors can quickly be arranged to 'smell' different gases.

In [3], Serra et al. use IEEE 1451 node computing capabilities and the reduction of power usage related to the reduction of transmissions needed for data acquisition to create a wastewater management system. NCAP interoperability features are also used by Serra et al. allowing for direct communication between sensors and actuators deployed in different NCAP's. A drawing depicting the system used is present in fig. 2.3.

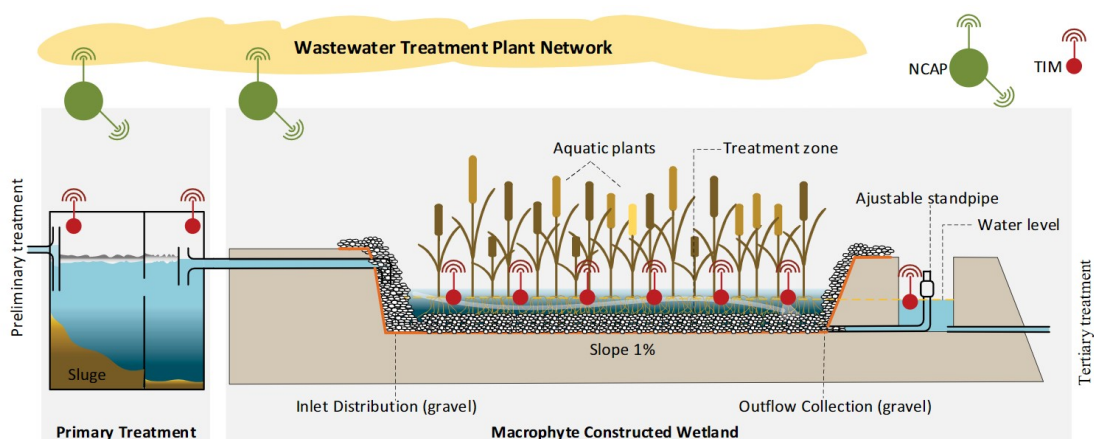


Figure 2.3: Wastewater treatment process automation using IEEE 1451[3]

Wang et al. developed a network-capable smart sensor [36] that captures the audio from a microphone and used a configurable digital signal processor (DSP) filter to clean the signal obtained from the source. It also uses the communication capabilities of IEEE 1451 to configure the sensor

during operation. Furthermore, Wang et al. summarize the IEEE 1451 standards and the specific hardware choices for NCAP, STIM, and TII design.

Making use of the TII interface independence provided by IEEE 1451 Batista et al. implement a smart instrumentation system based on an FPGA system [37]. Batista et al. also utilize the system adaptability provided by the TEDS to create an easily reconfigurable system. The standard is fully detailed and hardware and software implementations are thoroughly explained. Both wired and wireless versions of IEEE 1451 are studied and deployed.

Utilizing IEEE 1451.5 introduction of Zigbee capabilities Higuera et al. propose a one NCAP to many Wireless TIM's (WTIM) architecture [38], natural of Zigbee implementations. The TEDS is thoroughly analyzed and a proposal is made to improve the TEDS definition for Zigbee. Tests are carried out using one NCAP and 3 WTIMs verifying the interoperability and plug&play characteristic IEEE 1451 introduces.

A Smart Comfort Sensing System[39] was created by Kumar et al.. The intent of this system is to estimate the comfort of a building through the sensing of the presence of different gases, such as Carbon Monoxide and Dioxide, the sensing of temperature of the building, and the sensing of the humidity level. Building on top of IEEE 1451 communication and interoperability capabilities Kumar et al. developed a Wireless Sensor Network (WSN) responsible for retrieving data from all sensors and carrying out adjustments to the building's Heating, Ventilation, and Air Conditioning (HVAC) system in order to maximize the air comfort for its occupants.

In [40], Wei et al. utilize IEEE 1451.2 to create a pH measuring system, developing and configuring the TEDS in a way that the calibration adjusts itself based on the temperature of the medium being measured. This capability guarantees the correct pH readout. Wei et al. also utilize an adaptation of IEEE 1451 to allow for the USB connection to the intelligent sensor to carry readout and calibration operations.

Malar et al. [41] devise a smart transducer based on IEEE 1451.4. It uses a Mixed-Mode Interface implemented at the STIM level to capture signals and transmit them to a Digital Acquisition System (DAQ) which functions as the NCAP. The article also presents deficits in the smart transducers industry. It states that, although there are already some manufacturer solutions for smart transducers, there is a lack of development tools for creating such intelligent transducers. Malar et al. propose an architecture to promote the introduction of self-identification features to existing sensors through the addition of a TEDS-holding microcontroller. To test this hypothesis, 2 transducer modules are implemented: a load sensor with a measuring bridge and a thermometer based on a thermocouple sensor. Experiences showed success in this implementation. The work Malar et al. started is highly relevant for the presented dissertation work. Whilst recognizing the relevance of this work, Malar et al. fail to achieve the ease of smart transducer insertion into a system the current dissertation work intends to get. Moreover, the work presented is not public, meaning, the work developed by Malar et al. also fails to provide tools for the scientific community to build upon, which this work intends on.

Taking IEEE 1451 to a completely different and unexpected application, Pinto et al. create a Vehicular Sensing Network (VSN) [4] based on e-bicycles with the intent to monitor the mobility

platform, i.e. the e-bicycles' battery usage and location, and to measure the air quality throughout a small city. To do this, Pinto et al. defined an infrastructure in which, TIMs are installed in bicycles coupled with Ozone, Humidity, Temperature, and Energy Metering Sensors. NCAPs are implemented into the user's mobile phone which is responsible for monitoring three data transducing systems: the bicycle's TIM, the phone's GPS sensor and a virtual TIM implemented as an Android app featuring a user interface. There are also NCAPs installed in the e-bike photovoltaic charging stations. Communication between NCAPs and TIMs uses Bluetooth (IEEE 1451.5[24]). The TIM, NCAP, and TEDS implementation is explained by Pinto et al.. [4]'s authors conclude on the versatility of the solution implemented thanks to the IEEE 1451 standard being used. This versatility is then again evidenced in the future work proposed already foreseeing the integration of new sensors into the network (e.g. fall detection sensor). The infrastructure devised by Pinto et al. is shown in fig. 2.4

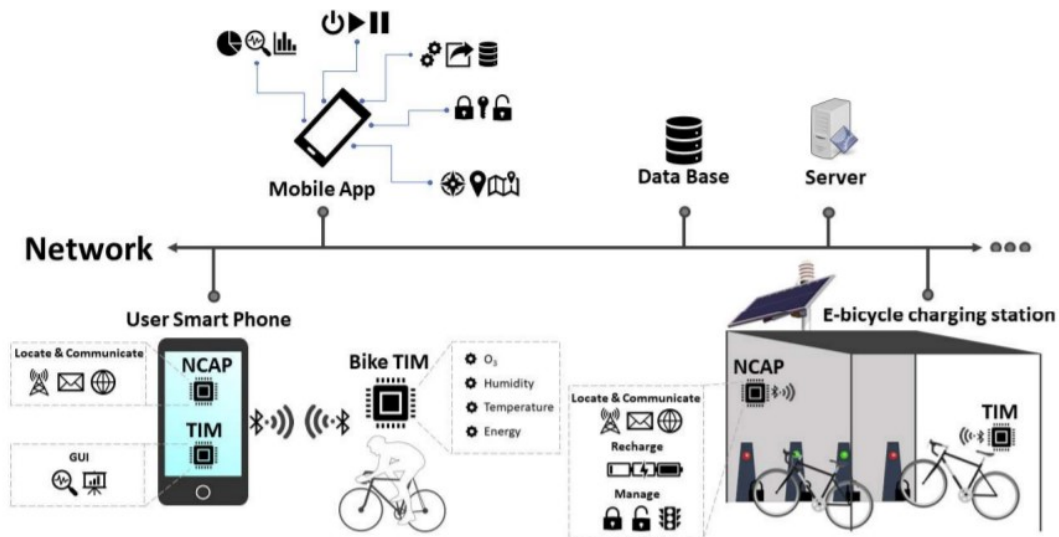


Figure 2.4: Vehicular Sensor Network Architecture Compliant with IEEE 1451 Standards[4]

In a more practical approach, Fan et al. implement a smart transducer[5] based on IEEE 1451.2 [21]. The STIM implementation is made on an FPGA and the NCAP is implemented around an STM32 microcontroller. A detailed report on the work developed is presented, explaining the TIM implementation (10-pin SPI interface as per standard). Tests on the network interface are also made, testing CAN, RS-485, SPI and universal asynchronous receiver-transmitter (UART). This work is concluded by presenting successful reports on the implementation with the sensor communication is presented. Whilst this paper introduces no new scientific knowledge, it presents a hands-on approach on part of the work this dissertation is focused on proving the feasibility of the work. The simple architecture used by Fan et al. is presented in fig. 2.5.

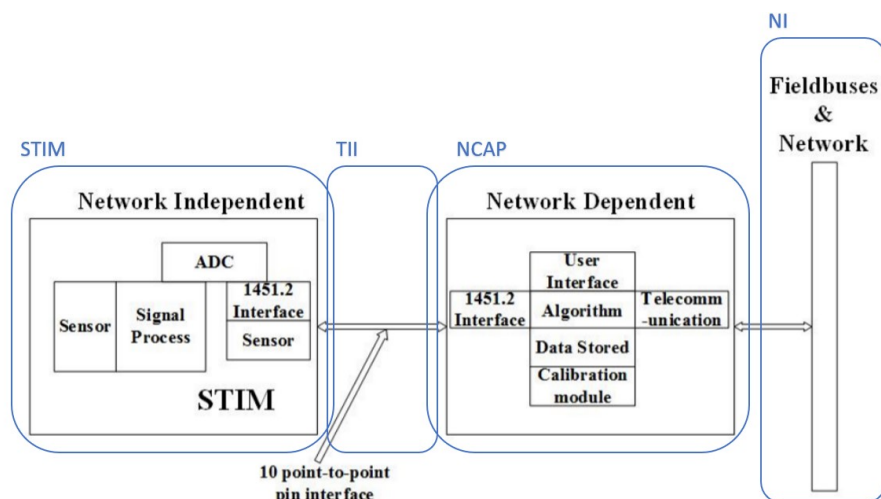


Figure 2.5: Basic smart transducer architecture used in [5]

2.4.2 Presented references analysis

A summary of the works presented above is introduced in this subsection. An analysis is made on table 2.1 about the technology components utilized, the industry it's applied on and the type of network used (either wired or wireless).

Table 2.1: Summary on the usage of IEEE 1451

Reference	Technology components used	Industry	Connectivity
Gas sensor (nose) [35]	Plug&play, self-calibration	Gas Sensing	Wired
Wastewater treatment [3]	edge processing & interoperability	Wastewater	Wireless
Microphone [36]	Edge processing & TEDS to save sensor characteristics	Audio	Wired
FPGA process automation [37]	Interface independence & TEDS reconfigurability	Automation	Wired and Wireless
Zigbee Study [38]	Zigbee TII, Interoperability & TEDS	Academic Research	Wireless
Smart Comfort Sensing System [39]	Interoperability, wireless communication & TEDS	(Building) Automation	Wireless
pH sensor [40]	Self-calibration through Calibration TEDS, Adaptation to USB	(Fish) Farming	Wired
Smart Transducers for Industrial Automation[41]	Mixed Mode Interface, TEDS & plug&play	Industrial automation	Wired
e-Bike Monitoring System[4]	Bluetooth TII, edge computing & TEDS	Vehicular Sensing Networks	Wireless
Smart Transducer[5]	TEDS & 10-line TII	Proof-of-Concept	Wired

Table 2.1 shows the broad applications IEEE 1451 can have. The sheer number of standards published (IEEE 1451.0,1,2,3,4,5,7) can imply this. The necessity of defining multiple possible TII interfaces allows the usage of IEEE 1451 in the most creative and collaborative ways. Everything from a simple microphone filtering system [36] to a Small City e-Bicycle Management System[4] can be achieved through the IEEE 1451 group of standards.

The most utilized feature of IEEE 1451 is TEDS and its ability to standardize the data storage and communication in smart transducers. The plug&play concept is also much utilized as allows for the ease of integration of new transducers into a network.

The literature review conducted showed as result many application papers, utilizing IEEE 1451 to achieve a goal, almost never conducting any analysis on the standard itself. Although there are many application papers on the subject, it was not possible, to the best knowledge of the author, to find tools to develop transducing systems using IEEE1451. The academic community recognizes the importance of the standard and only utilizes it as a tool to develop systems. Search through Google Scholar, Scopus and Web-of-Science also retrieved many book chapters talking about the importance of smart transducers and pointing to IEEE 1451 as almost an absolute way to achieve them. It can be concluded that, in the industrial area, a smart transducer equates to IEEE 1451.

As a final note, the review work conducted above refers to the most relevant references obtained by the author. Academic production regarding IEEE 1451 has been decaying in the last years. Search for the IEEE 1451 keyword yields only 16 results from 2018 onwards. Of those only 4 were considered relevant to the topic being covered. This can mean the industry is moving somewhere else, mainly to the P21451 group proposed standards, which will be reviewed next.

2.5 A look into the future for IEEE 1451 - IEEE 21451

Currently, IEEE 1451 is being updated through IEEE Project P21451, still in a development phase. With the industry moving towards Industry 4.0, Industrial Cyber-Physical Systems(ICPS) are becoming more and more abundant. One of the main components of ICPSs are sensors and actuators, the system interface with the physical world. The growing abundance of data flow in systems and the interoperability, interconnection, and cooperation provided by the growth of IoT have changed the path of IEEE 1451 development. The addition of more and more, smaller nodes, normally with lower computation power requires the alteration of standards to fit into the new paradigm. Newer IoT communication technologies such as Message Queuing Telemetry Transport (MQTT) are being also considered in the upgrade of standards into the NCAP to network communications schema (IEEE P21451.1.6)[42].

The TEDS format description has also been merged and updated being now described by IEEE 21451-4[43] and IEEE 21450³. In [43], Viegas et al. describe and test the TEDS format description while also introducing a new TEDS format request for identifying frequency-time sensors.

Newer low power wide-area network (LPWAN) technologies are also currently being integrated into the standard set, namely, Long Range (Lora)(P1451.5.5), SigFox(P1451.5.6) and Narrow Band IoT (NB-IoT)(P1451.5.10) [42]. Researchers intend to create an effort-efficient design process and Plug&Play property for different LPWANs[44].

Low-power node integration is also being studied and developed under IEEE P21451-002. [42]

One of the characteristics of ICPS is the massive collection of data and information from industrial systems. In order to make use of this data, algorithms are needed to characterize the data collected and infer knowledge from the information contained in this data. With this in mind, IEEE 21451-001[6] comes into play defining signal processing algorithms and data structures at the STIM level. Furthermore, recommended practices are given in order to share and infer signal and state information of an instrumentation or control system. The algorithms proposed by [6] are based on the signal and transducers attached to the system.

A new set of NCAP service features is also defined by P21451-1, which will become addends to the IEEE 21451 standard. IEEE P21451-1-1 defines a TCP/UDP service set and interface, IEEE P21451-1-2 gives the NCAP HTTP service providing capabilities, IEEE P21451-1-3 provides Web Services from the NCAP, IEEE P21451-1-4 does the same for eXtensible Messaging and Presence Protocol (XMPP), and, finally, IEEE P21451-1-5 provides a Simple Network Management Protocol (SNMP) interface to the smart transducer.

These 5 standard subsets will provide the smart transducers with IP, HTTP, Web Services, XMPP, and SNMP services, respectively.[6]

A newer, future version of fig.2.2 is presented in [6] and is provided in fig. 2.6.

³The naming schemes of IEEE 1451 updated standards are changing and differ from author to author, even inside IEEE standard documentation.

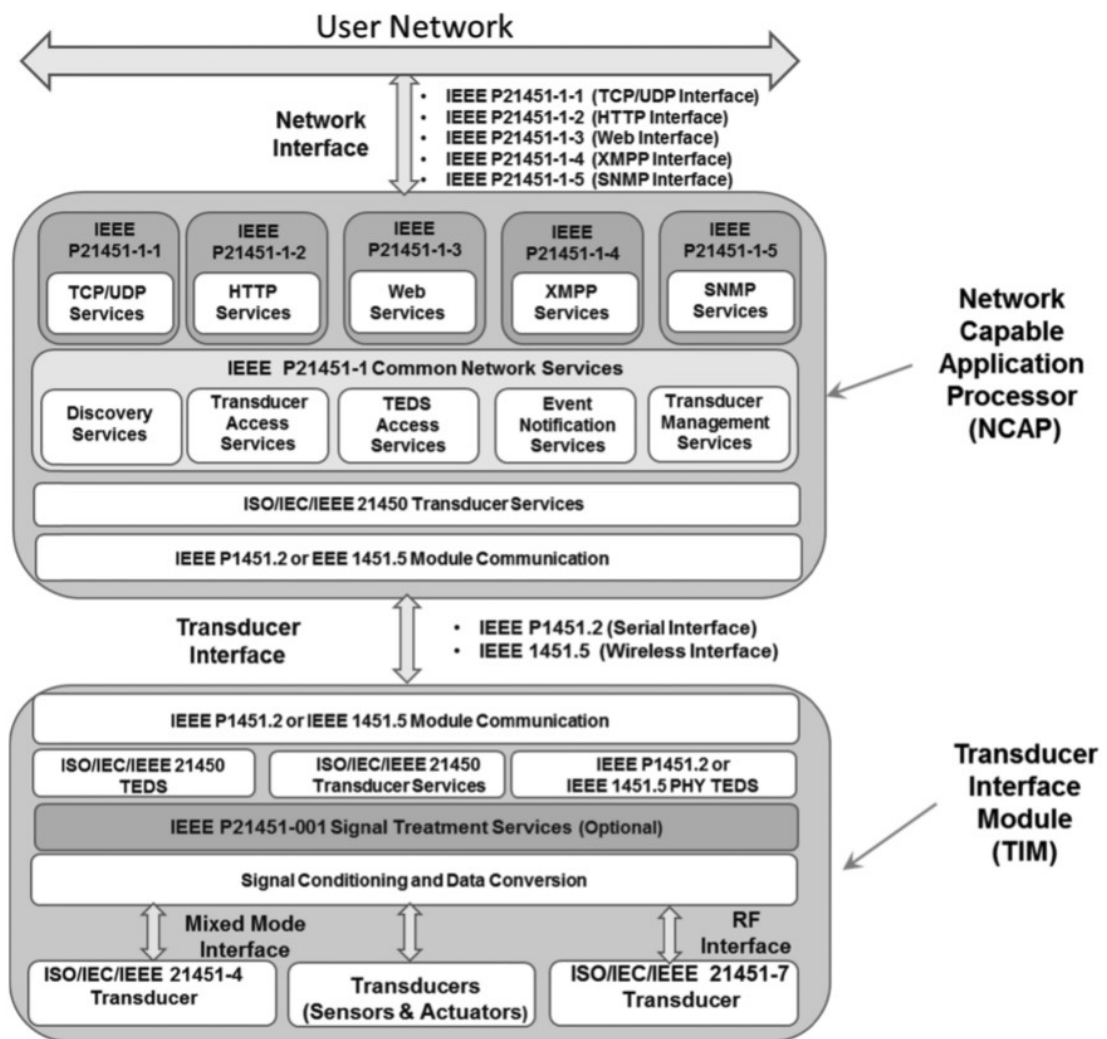


Figure 2.6: IEEE 21451 smart transducer architecture[6]

Applications on IEEE 21451 coincide with those using IEEE 1451 however, IEEE 21451 has more presence in recent literature.

Relating to the topic of this dissertation, Cherian et al. present work on an IoT interface for industrial analog sensor [7]. An NCAP + TIM integrated solution is developed allowing for the input of 4 types of sensors: Voltage-based, Current-based, Pulsed outputs, and NPN/PNP or dry contact sensors. For TEDS communication XMPP is used, as per standard IEEE 21451. The transducing platform presented is also powered through the network using Power Over Ethernet (PoE) which is an interesting approach. The architecture presented is available in fig.2.7. Cherian et al. discuss the importance of upgrading functioning analog sensors while giving them processing capabilities, one important topic this dissertation work fits right in. The upgrading of functioning an high precision manufacturing system is more times than opposite, cheaper than replacing the system with a new one. This work is highly relevant to the present dissertation but, again, there is no access to the tools developed by the authors.

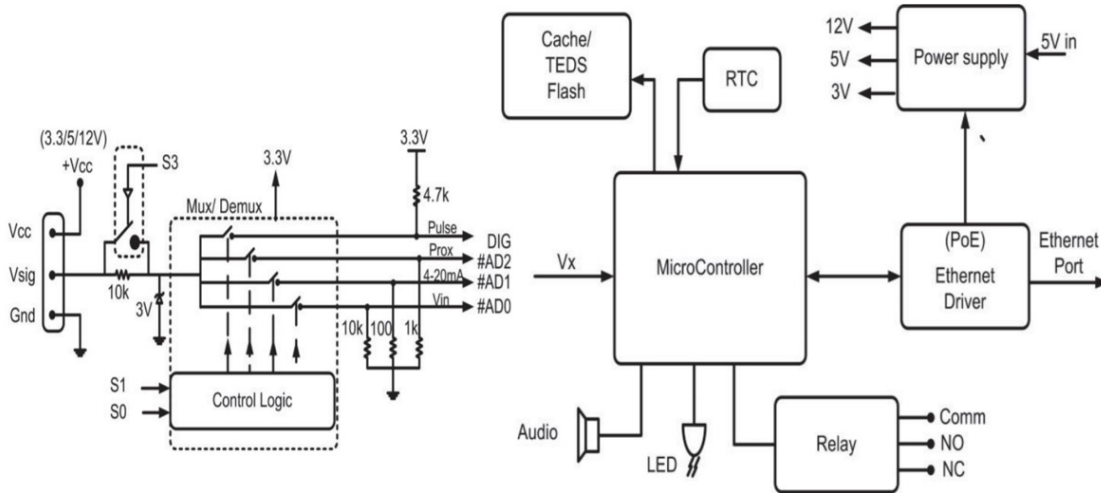


Figure 2.7: Smart transducer architecture presented by [7]

2.5.1 Perspectives on the future for Smart Sensors and Cyber-Physical Systems

As stated in 2.5 the utilization of IEEE (2)1451 fits right into the definition of a CPS component. The plug&play features of TEDS, the interoperability of transducers, and the information-based architecture make IEEE (2)1451 based smart transducers highly attractive to be applied in Industrial CPS. In [45], Rocha et al. analyze the interoperability between a CPS system and an IEEE 1451 smart transducer mainly proposing the implementation of two different types of interoperability: syntactic interoperability and semantic interoperability.

Syntactic interoperability exists when two different systems can use a common format and data structure to exchange data. Whilst data is conveyed correctly, the meaning of data cannot be extracted from the message. Rocha et al. propose using MQTT in a system to guarantee syntactic interoperability.

Semantic interoperability, on the other side, exists when two different systems can use a common format and data structure to exchange data while guaranteeing the meaning is still conveyed. An example of a communication protocol that conveys meaning is OPC-UA, a protocol that conveys the data type along the data value. The use of Industry 4.0 architectures, such as the Reference Architecture Model for Industry 4.0 (RAMI) or Industrial Internet Reference Architecture (IIRA), make semantic interoperability even more important as data referring to the hierarchy levels defined by the architectures must be also conveyed in order for the architectures to be useful. Rocha et al. suggest the use of OPC-UA with Javascript Object Notation-Linked Data (JSON-LD) on top in order to make use of the advantages of semantic interoperability (higher content of information flowing).

In another development perspective, regarding the importance of smart transducers in Cyber-Physical Systems, Song et al. created a Smart Sensor Digital Twin System [8] with the emulation of smart sensors in mind. This tool can be used to emulate the interactions between smart sensors and CPSs, inject faults in the system, and analyze the behavior of the CPS regarding the fault injected. This tool is relevant to the presented work as it can be used to test the developed smart transducing system against a CPS. It can also be used to compare the behavior of the real Smart Transducer to be developed against the emulated Smart Transducer. This comparison can help develop the Smart Transducer Interface Module to standard specification.

Song et al. also present a diagram depicting the interaction between CPSs and IEEE 1451 systems. This diagram is present in fig. 2.8.

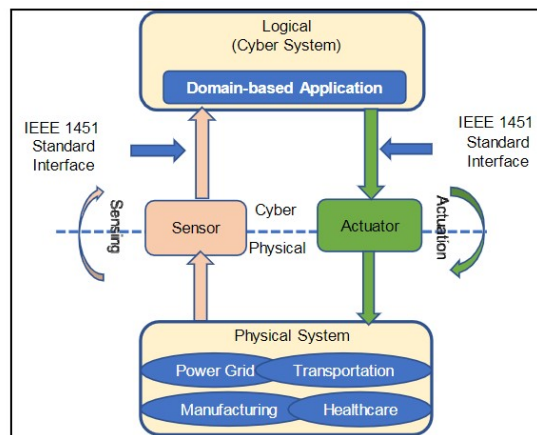


Figure 2.8: Example of IEEE 1451-based CPS [8]

2.6 Conclusions on the Bibliographic Review

The standard upgrading process brings with it naming scheme problems, as visible in section 2.5. The main focus points of this dissertation work bear with the IEEE 1451 parts which have not changed during the transition to IEEE 21451. Having this in mind, and as IEEE 1451 and IEEE 21451 mean the same in the context of this work, the nomenclature IEEE 1451 will be used to refer to the standards in the next section chapter.

From the literature review that has been made so far some conclusions can be drawn:

- IEEE 1451 was born from the industrial market needs as was implanted as the solution to follow to get a smart transducer;
- The IEEE 1451 standard still has space to grow into industrial applications. IEEE 1451 still remains at an academic level of use and, while some real use-case applications exist, the introduction of a Smart Transducer Module would be relevant;
- The developments made using IEEE 1451 are almost always custom-fitted to the application needed. Work towards universal smart transducing modules would be relevant.
- There is the lack of tools to help the development of transducing systems that can adapt to multiple applications
- The integration of IEEE 1451 in networked control loops will help the industry transition into Cyber-Physical Systems;

This dissertation work appears to be relevant in helping the IEEE 1451 standard to be implemented in industrial environments. The development of a configurable system to sample non-smart transducers and tools to create smart transducers, according to the literature review above seems fitting and important. This system can then be used as a tool for testing, deployment and integration of transducing systems. Opening this development back to the scientific community also seems important, to allow for others to develop and build upon the knowledge this work intends to create.

Chapter 3

Smart Transducers - Requirements, Reference and Proposed Architecture

3.1 Introduction

3.1.1 Motivation

From the bibliographic review in the previous chapter, no general and open solution was found to contribute to the usage of IEEE 1451 in industrial environments. Moreover, the transition to Industry 4.0 architectures imposes a high demand for meaningful asset information, which motivates the adoption of IEEE1451. These premises, combined, justify the need to develop a solution that makes the IEEE1451 easier to use in industrial environments. There is a need to create a system that can handle different use cases related to production scenarios and be easily programmed to fit these production scenarios.

Ease of reconfiguration, modularity, and expandability come as goals to try and meet within this dissertation work. This dissertation also lines up with previous and current work being developed by the author, which is detailed next.

3.1.2 Contextualization

Before this dissertation work began, the author was involved in the "EIT-Manufacturing FactorIS" research project. Inserted in this project, a learning factory demonstrator was refurbished to serve as a learning media to introduce students to zero defects manufacturing, condition monitoring, and system integration [46]. In [46], the author et al. detail the process behind the refurbishment and upgrading of an old but functional learning factory. Inserted into these developments, there was the need for adding, upgrading, and integrating sensors and actuators into an already functioning system. Also, by working alongside the DIGI2 laboratory the author contacted with diversified projects regarding Cyber-Physical Systems and Artificial Intelligence integration into systems and their respective developers. A large volume of the work included in these projects is

the integration of sensors and the creation of endpoints to fetch data from in order to be handled or stored for future processing. Moreover, many of the problems mentioned regarding sustainability, reparation and upgrading of systems, described in the introductory chapter were common to many of the projects developed. All these facts motivated the author to research and develop under this engineering area: the integration of transducers into control loops.

This dissertation work is associated with the "Continental FoF" R&D Project. "Continental FoF"'s primary goal is to create technical and academic knowledge to aid in the development of state-of-the-art products that are to be integrated as a way to create a *Factory of the Future*, to be deployed in Continental's Advanced Antenna (AA) factory. This knowledge ought to be used to address the complex challenges arising from the new technological requirements expected for future generation vehicles.

As is well known, control, optimization, and prediction techniques are highly dependent on information.

In an industrial environment, information is mainly generated by sensors and deployed by actuators. When there is enough benefit to applying some of these techniques, the process behind deploying them includes assessing the information available, and, if required, installing new sensors and actuators.

To sustain the development of an Industrial Cyber-Physical System for the *Factory of the Future*, there is a specific need to devise an edge device solution ready for the plug&play of sensors and actuators.

The motivation behind this device is the software development and configuration effort currently required for technicians to integrate new transducers.

Products, processes, and assets will be equipped with transducers, whose information will be used for intelligence purposes. The development of transducer integration techniques constitutes the use case that frames this dissertation, and its results will be validated in the Continental AA factory.

3.1.3 Architectural Goals

As mentioned in the introductory chapter, the main objective of this dissertation work is to develop a platform for converting normal sensors and actuators into Smart Transducers based on IEEE 1451. It shall be capable of interfacing with any transducer to collect and deploy data, convert it to information and communicate it to a network in a real industrial use-case scenario. The development of a Smart Transducer Interface Module responsible for sampling and conditioning signals for both digital and analog sensors will achieve this objective.

Developing this module will not only help in comprehending the IEEE 1451 standards and their intricacies but also serve as a demonstrator of what using the distributed computing power of a Cyber-Physical System and the plug&play functionalities of IEEE 1451 can do.

There is also the need to meet the goals defined in 3.1.1, such as reconfigurability, modularity, and expandability.

With such goals and objectives in mind, the following architecture was developed.

3.2 IEEE1451.0 Reference Architecture

IEEE1451.0 [9] describes a reference model for a smart transducing system, presented in figure 3.1. In this reference model, two main components, the NCAP and the TIM perform complementary actions that communicate via a Communications Module.

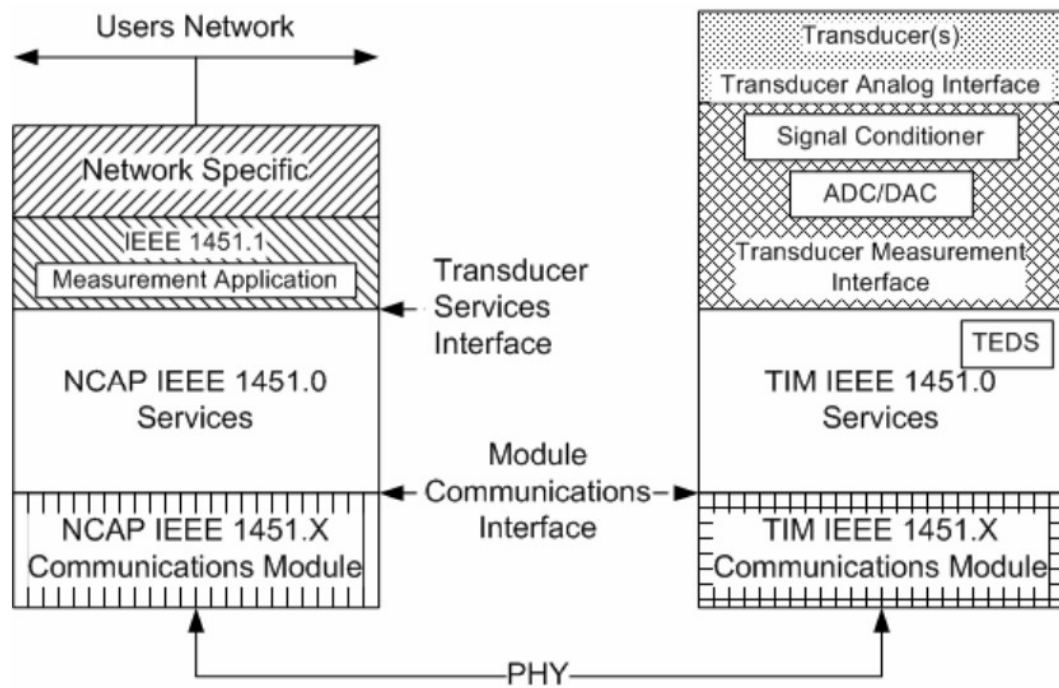


Figure 3.1: IEEE1451.0 reference model [9]

The main goal of this work is to develop a Transducer Interface Module and, as such, regarding this reference architecture, is important to point out that:

- The communications module can be either defined through IEEE1451 standards or implemented through a proprietary module referred to as compliant to fictitious standard IEEE1451.X. Any communication module complies with IEEE1451.X as long as it implements the interfaces defined in IEEE1451.0 referring to the communication module. **This dissertation work will, along with the NCAP team, create an IEEE1451.X module.**
- The transducer measurement interface is left for the developing team to define, with no well-defined interface defined. This interface should be designed to allow the recollection of data that allows the TIM IEEE1451.0 services to be provided. **This design and implementation are an object of this dissertation work.**

An IEEE1451.0 compliant Smart Transducer System must also implement the Transducer Services Interface on the NCAP side.

The main mechanisms the standard defines and utilizes are explained next.

3.2.1 Transducer Channels

The standard defines a transducer channel(TC) as "a transducer and all of the signal conditioning and conversion components associated with that transducer." [9]. This definition encapsulates all the characteristics of a transducer and all the actions that can be executed on it. A TIM can be defined as a module that contains and manages transducer channels. A single TC is always identified by a 16-bit Transducer Channel Number, which belongs to a TIM, ranging from 0x0001 to 0x7FFF for a total of 32767 unique Transducer Channels per TIM. Transducer Channel number 0 is reserved for TIM-related operations.

3.2.2 Transducer Electronic Datasheets (TEDS)

Transducer Electronic Datasheets are binary structure which holds information about the capabilities and configurations of the TIM or Transducer Channels. The TEDS can be categorized in the following manner:

3.2.2.1 Meta TEDS

The meta TEDS is a TEDS related to the TIM which defines timing and control settings and top-level Transducer Channel information (how many transducer channels exist and how they can be controlled and probed). There is one and only one Meta-TEDS per TIM.

3.2.2.2 Transducer Channel TEDS

The Transducer Channel TEDS is a TEDS that carries information about a specific Transducer Channel, the kind of transducer it refers to, how to access it, what information it shall provide or receive as well as attributes regarding buffered storage of data inside the transducer channel for posterior polling. There is one Transducer Channel TEDS per Transducer Channel.

3.2.2.3 PHY TEDS

The Phy TEDS is a TEDS responsible for storing the characterization of the physical communication media being used to communicate with the TIM.

Note: The previous 3 TEDS are required to be implemented.

3.2.2.4 User's Transducer Name TEDS

The User's Transducer Name TEDS is a TEDS responsible for providing each TIM or Transducer Channel with a User Defined Name used for interpretation name. Such TEDS implies a

one to one relationship between a TC or TIM to a Transducer Name. An example of this approach might be a TIM called "Weather Condition Monitor" with Transducer Channels called "Thermometer" and "Humidity Sensor".

There are more TEDS defined in the standard, but were not deemed as relevant for the present document.

3.2.2.5 TEDS: interpretation

To further explain and decrypt how TEDS are implemented and constructed a META-TEDS was generated using a TEDS generating tool developed by the DIGI2 laboratory [47]. Please refer to table 3.1.

The TEDS Data Block is composed of simpler blocks called TLV (Type, Length, Value) Lines. The type determines what the line carries, the length the number of bytes, and the value that carries the data. The TEDS Data Block is preceded by 4 bytes which hold the TEDS Data Block Length and succeeded by a checksum to ensure correct transmission of data.

Table 3.1: Meta-TEDS Interpreted

T	L	V	Value Converted	Description
		00 00 00 2A	2A(16) = 42(2)	TEDS Length
03	04	00 01 01 01	Meta-TEDS version 1	Type of TEDS
04	0A	B3 D1 07 DA 22 A1 4A 2B 98 DC	UUID	Unique ID
0A	04	40 A0 00 00	40A00000(16) = 5.0 (Float32)	Operation Timeout
0B	04	40 40 00 00	40400000(16) = 3.0 (Float32)	Slow-access Timeout
0C	04	3F 80 00 00	3F800000(16) = 1.0 (Float32)	Self-Test Timeout
0D	02	00 06	6(16) = 6(2)	#Transducer Channels
		F8 4B		Checksum

3.2.3 Commands and Messages

Commands are packets of data that convey orders, generated by the NCAP that the TIM should accept and handle through the TIM IEEE1451.0 services. If the command requires a response, i.e., is a request, the TIM shall then respond to the NCAP, which will use this information to finish its service request. There are also commands that do not require a response, typically to set parameters or enable functions. The command structure defined by the standard is present in figure 3.2, the response structure is displayed in figure 3.3.

TIM's are also capable of starting communication and shall do it through the use of TIM-initiated messages, following the structure in figure 3.4

A flow of information based on commands allows for a loose coupling between systems, desirable in such a flexible system. As long as the NCAP and TIM implement the interfaces they are required, the systems should be able to cooperate without each needing to be fitted to each own custom details.

1-Octet							
7	6	5	4	3	2	1	0
Destination TransducerChannel Number (most significant octet)							
Destination TransducerChannel Number (least significant octet)							
Command class							
Command function							
Length (most significant octet)							
Length (least significant octet)							
Command-dependent octets							
.							
.							
.							

Figure 3.2: IEEE1451 command structure[9]

1-Octet							
7	6	5	4	3	2	1	0
Success/Fail Flag							
Length (most significant octet)							
Length (least significant octet)							
Reply-dependent octets							
.							
.							
.							

Figure 3.3: IEEE1451 response structure[9]

1-Octet							
7	6	5	4	3	2	1	0
Source TransducerChannel Number (most significant octet)							
Source TransducerChannel Number (least significant octet)							
Command class							
Command function							
Length (most significant octet)							
Length (least significant octet)							
Command-dependent octets							
.							
.							
.							

Figure 3.4: IEEE1451 TIM-initiated message[9]

3.2.4 Requirements for compliancy

A system is deemed compliant with the IEEE1451.0 standard if, and only if, it follows the following requirements, regarding the TIM (which will be the main focus of this dissertation work):

- The TIM follows and implements the functional specification described in the standard (mainly defined as the transducer channel architecture, the command and message structuring of communication, and the TEDS implementations).
- The TIM implements the commands and messages signaled as required in the standard.
- The TIM supports required TEDS that have the format and content specified in the standard.
- The TIM supports one of the communications protocols and physical media defined by another member of the IEEE 1451 family of standards or implements one according to Communications Module API specification.

3.3 Architecture proposal

Extrapolating from both the IEEE1451 architecture and the objectives retrieved from the Bibliographic Review, some design objectives were set:

- Connecting multiple TIMs to one NCAP, conceding plug and play capabilities to this connection.
- Connecting multiple transducers connected to one TIM
- Making the addition of transducers to the loop control as easy as possible
- Making the TIM independent from the signal conditioning cards

With such objectives in mind, the architecture presented in figure 3.5 was devised.

Following this architecture, the transducing system comprises three main components: the NCAP, the TIM backplane, and the TIM signal acquisition and conditioning cards. As this architecture implements the standard, many of the blocks are similar to the ones found in the reference model. However, this architecture presents some differences, which will be evidenced in this section.

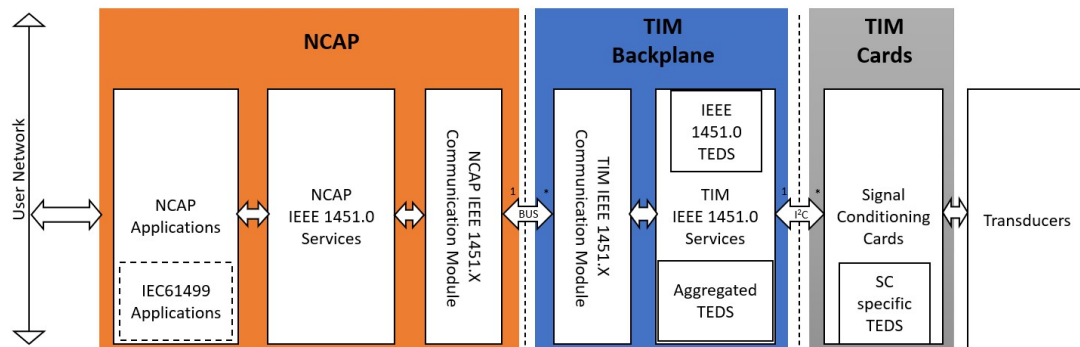


Figure 3.5: Architecture Model Proposal, built around IEEE1451.0's reference model [9]

3.3.1 NCAP

The Network Capable Application Processor is responsible for serving as the processing node which coordinates operations inside the IEEE1451 compliant system. It is responsible for holding applications that make use of the transducer aggregation power provided by the TIM. The NCAP was not the main focus of this dissertation. It was developed in parallel by another student working on a master's degree dissertation in the DIGI2 laboratory. In this particular NCAP implementation, the applications will be deployed using IEC61499 function blocks running on the DIGI2 Lab's DINASORE Runtime Environment [48]. Applications will communicate via the Transducer Services Interface with the NCAP IEEE1451.0 Services module, which will be responsible for coordinating communications with the TIM, using the TEDS structures to poll sensors data and manipulate the actuators.

3.3.2 TIM (Communications Module)

The Communications module serves as the interface between the NCAP and TIM. Through the communications module, commands and messages are passed from the NCAP to the TIM and in reverse. This layer ought to be utterly opaque to the NCAP and TIM. The Communications Module must only implement the Communications Module API functions.

For this specific work, versatility is at a premium, so the ability to connect multiple TIMs to one NCAP is highly relevant. IEEE1451.0 provides two different approaches for communication modules: Point-to-Point Communication and Network Communication. According to the wanted applications, Network Communications were chosen.

Of the physical implementations for networked communication, CAN 2.0 B was chosen. It was the technology that better fitted into the proceedings of the DIGI2 laboratory. Moreover, it presents the following benefits:

- Extended Addressing, allowing both sender and receiver to be identified in the messages being sent
- Low Latency in small messages, such as simple sensor readouts
- Error detection and correction at the data-link level
- Easily deployable peer-to-peer communication
- Easy scalability of a TIM network

3.3.3 TIM Backplane

The TIM Backplane represents a physical hardware module that implements and runs the TIM IEEE1451.0 Services. It will be responsible for receiving, decoding, and interpreting the commands from the NCAP, orchestrating communication with the TIM cards, fetching data from these, and reporting back to the NCAP via a message. Since the TIM Cards also require plug-and-play features, it will also be responsible for deploying registration services for TIM Cards and managing the timings for communication with these. It will also hold the TIM Meta-TEDS and all configurations common to all transducer channels.

3.3.4 TIM Cards

TIM Cards shall serve as the physical representation of Transducer Channels. They shall carry all the signal conditioning circuitry needed to interact with physical transducers. For this reason, all the configurations and information relative to each Transducer Channel shall also be stored inside each TIM Card (encapsulated into each transducer Channel TEDS). It must also contain communication capabilities and be implemented in a way that features plug-and-play capabilities into a I^2C bus. TIM Cards should be hot-swappable, meaning they can be inserted or removed from the system without disrupting its normal functioning. Each TIM Card will necessarily have

to feature a microcontroller to implement and convey all the functions required for the correct functioning. This card might encapsulate multiple sensors and mask them as one or carry out simple operations such as comparative or signal amplification.

3.3.5 Backplane to Card Interface

The interface between the cards and the backplane shall be implemented using I^2C , featuring hot-swappable buffers in order to allow for electric compatibility with plug-and-play behavior. The backplane to card interface must also carry power to the cards to reduce wiring and centralize power distribution. If the card is to need external power, the external power just will not be utilized.

3.3.6 Advantages from the Transducing System Architecture

The architecture hereby presented attains to achieve new degrees of flexibility in transducing systems. By transforming the TIM into a mutable and adaptable component of the system, the objective is to provide ease of upgradeability, expansion, repair, or substitution of components. Also, by containing all the signal acquisition circuitry inside a card, the TIM Backplane can be reutilized between different projects or upgraded inside the same project without the need to replace all the signal acquisition circuitry.

Imagine the following scenario: by the proposed architecture, the TIM backplane can become a bottleneck in the polling frequency of the system (if, for example, there is the need to sample faster for a specific process). To solve this problem a TIM backplane based on a faster processor can be easily swapped in, or the load might be distributed against multiple TIM backplanes, requiring little to any system configurations. With this approach, the TIM backplane becomes a simple component of a broader ecosystem, all interconnected with information being absorbed internally from the TIM cards.

In the same manner, the TIM cards are also expected to bring more flexibility to the system. Imagine having a custom sensor that needs specific hardware to read its measurement (e.g. a load cell measured through a Wheatstone bridge). Once the card is custom fitted around this sensor, it can be deployed into each and any TIM backplane. This can become an advantage if sampling hardware needs to be reused in different scenarios and reutilized throughout time. A temperature sensor removed from a functional asset no longer needs to be discarded as long as it's kept coupled to its card. It will become just another asset waiting to be deployed into a new system. This contributes to reducing electronic waste and adds to the sustainability of the system proposed.

A simplified visual diagram of the hardware interconnections can be seen in figure [3.6](#).

3.4 Summary and Conclusions

This chapter contained the design context and decisions that motivated the development of the TIM-expandability-based architecture. This architecture intends to create a more flexible system

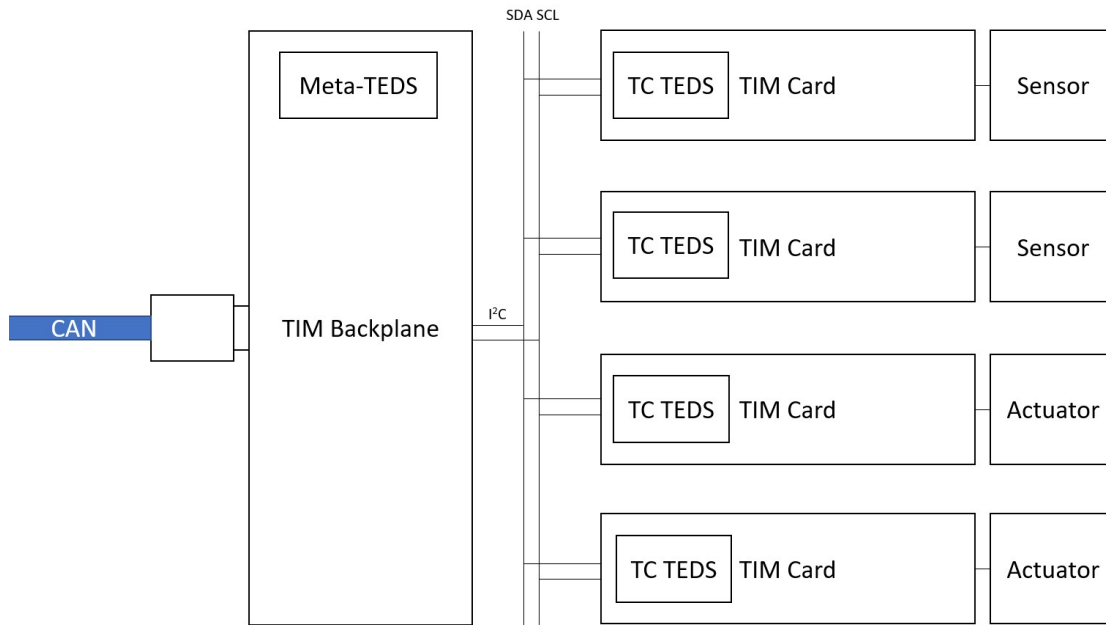


Figure 3.6: Graphical Representation of an Implementation of the presented Architecture

that more easily adapts to a broad set of industrial transducer-based problems.

Moreover, analysis and explanation are made on the IEEE1451, a vital piece of information that inspired and served as a basis for the work presented here. The architecture presented above intends to expand on the features of the standard, which makes little any requirements for the TIM working fashion.

Implementing the TIM along with a robust and well-tested interface such as CAN is expected to provide a strong industrial backbone, allowing for the distribution of TIMs in networks in order to provide a multi-drop transducing system. This approach expects to allow distributed measuring and actuation systems to be more easily deployed on the shop floor.

Introducing the TIM cards also expects to bring novelty and flexibility to the traditional TIM approaches, sometimes seen in literature as a One TIM-One Sensor implementation. This approach, even if theoretical, expects to be a basis for future work.

Chapter 4

Implementation platform

4.1 Introduction and Contextualization

The nature of the problem being solved, developing a transducer interfacing system, carries along the development of software and hardware to fulfill communication-bus-depending, speed, and physical transducer interfacing requirements. The specific case described here fits under the realm of Embedded Systems, with the customization of both hardware and software to fit a clear and well-defined purpose.

This type of system, with such proximity to the hardware level, called for a microcontroller system designed with communication and general-purpose interfacing in mind. The decision process will be detailed in this chapter.

Custom hardware was also needed for implementation. A prototyping shield was developed first to use for tests, being replaced later with a printed circuit board for a more robust solution.

The development process for software and hardware, choices of tools, and development environment choices will be explained in the next subchapters. Given the complexity of the code base, a version control system was also chosen, whose choice will also be explained.

4.2 Hardware

4.2.1 Micro-controller: TM4C123XL Platform

Choosing a microcontroller is an important step in starting an embedded design process. The platform chosen will have a direct impact on the development process to be followed. Some crucial points to help in this decision are as follows:

- The existence of an IDE, tailored to the needs of the specific platform to help in the development and debugging process.
- The availability of a knowledge base about the platform (either in form of human or online databases).
- The support for RTOS and which RTOS are supported.
- The programming languages supported natively by the platform.

The TM4C123GXL complies with all the points: It has a dedicated IDE, Code Composer Studio; Texas Instruments keeps an active online community for developers to ask questions which is much utilized. It supports RTOS's: FreeRTOS and TI's proprietary TI-RTOS. It supports programming natively in C and C++ through the IDE.

The TM4C123GXL (graphically represented in 4.1) is a development platform based on the micro-controller TM4C123GH6PM by Texas Instruments. The TM4C123GH6PM is designed around an ARM Cortex-M Processing core running at 80 MHz. It features a 256kB Flash Memory, 32kB of RAM, and 2kB of EEPROM.

Focusing on the communication capabilities of this micro-controller, it presents the user with 8 Universal Asynchronous Receivers/Transmitters (UART), 4 Synchronous Serial Interfaces (SSI), four high-speed Inter-Integrated Circuit I²C modules, 1 Universal Serial Bus (USB) 2.0 interface, and 2 Controller Area Network (CAN) controllers. For signal acquisition and control, it features 6 General Purpose Input Output (GPIO) blocks, 16 Pulsed Width Modulation (PWM), and two 12-bit Analog to Digital Converters (ADC).

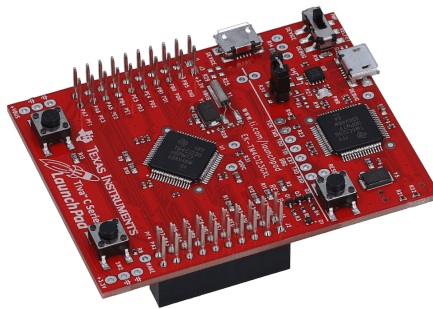
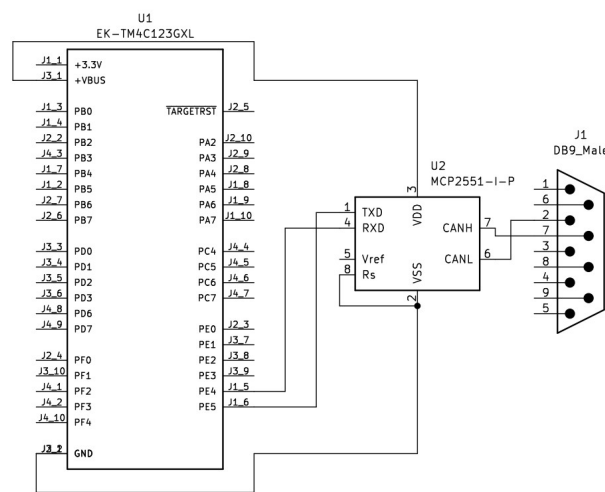


Figure 4.1: Texas Instruments TM4C123GXL Development Platform

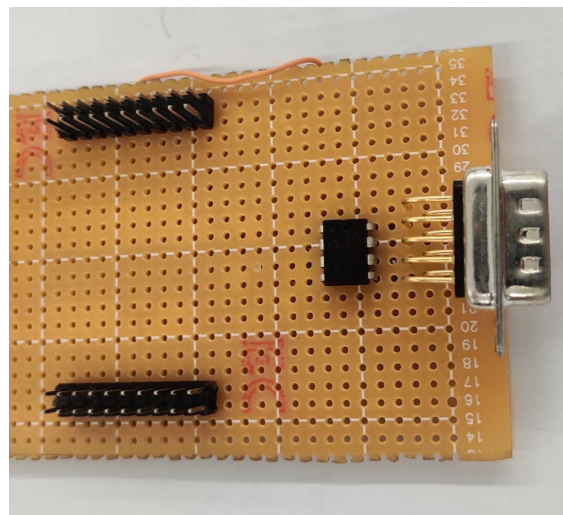
To allow the interface with the CAN bus, a prototype shield was devised.

4.2.2 Development Shield

As referred to in chapter 3, there is the need to communicate with a CAN bus. Although the microcontroller features CAN bus controllers, it requires a CAN bus transceiver to act as the interface between the 12V logic of the CAN bus and the 5V logic from the microcontroller. With such intent, the integrated circuit MCP2551, a high-speed CAN transceiver, was introduced to the circuit. In the beginning, the circuit was assembled using a breadboard, but the circuit was transposed to a prototyping board to give the circuit sturdiness and durability. The CAN bus connection was then transported to D-Sub 9 plugs, which allowed for a more robust connection. A simple circuit schematic is presented in figure 4.2a and a picture of the assembled proto board in figure 4.2b.



(a) CAN shield schematic

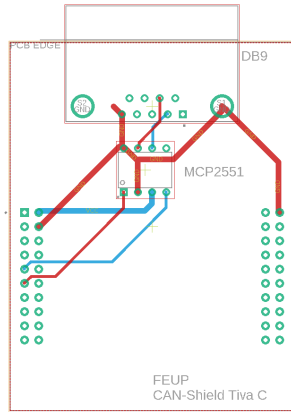


(b) CAN shield in protoboard

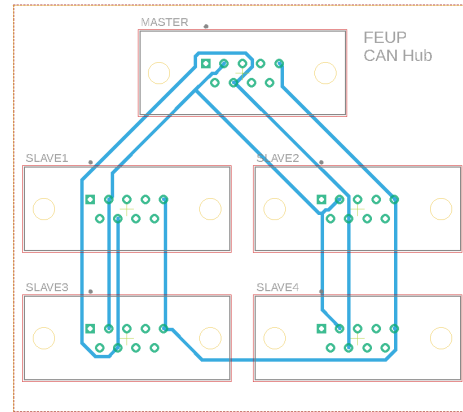
Figure 4.2: CAN shield

After implementing a prototype of the board and verifying all connections and functions of the board were correct, a more definite approach was taken. For this, a Printed Circuit Board was projected in order to have a more sturdy and reliable solution. The board was drawn using Autodesk Eagle.

In parallel with the CAN shield a CAN Hub was also designed in order to facilitate connections from one NCAP to many TIMs all in the same CAN bus. The layouts of both these boards are presented in figures 4.3a and 4.3b.



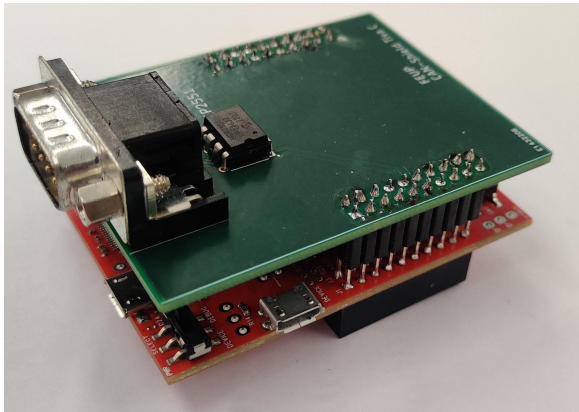
(a) CAN shield board layout



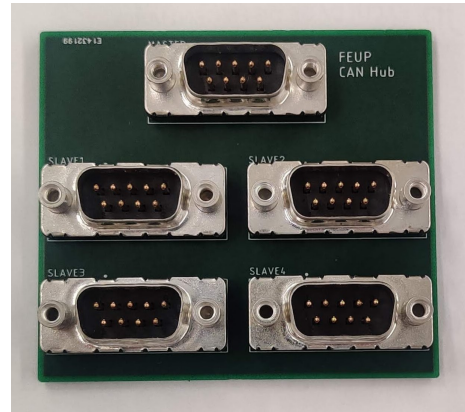
(b) CAN hub board layout

Figure 4.3: Board Layouts

The resultant boards from the layouts above presented can be seen in figures 4.4a and 4.4b.



(a) CAN shield PCB assembled



(b) CAN hub PCB assembled

Figure 4.4: Final PCBs

The PCBs fulfilled their main objective: providing a solid and robust base to work on.

4.3 Soft/Firmware

4.3.1 Programming Language Choice

As previously presented, this dissertation work implements the IEEE1451 standard. This standard is heavily oriented toward modularization, interface definition, inheritance, and data abstraction, all characteristic of Object-Oriented Programming (OOP). The Tiva C platform is oriented towards low-level programming using C or C++. As C is not an OOP language, the natural choice made was C++.

Furthermore, the IEEE1451 standard, with independence from implementation in focus, uses Interface Definition Language (IDL) to describe how to implement the interface functions, i.e., what parameters they should expect as inputs and what parameters they should output as a result. An example of the utilization of IDL in the standard is present in table 4.1.

Table 4.1: Example of interface functions definition using IDL [9]

IEEE1451Dot0::TransducerServices::TedsManager
Args::UInt16 readTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType, out Args::ArgumentArray teds);
Args::UInt16 writeTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType, in Args::ArgumentArray teds);
Args::UInt16 readRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType, out Args::OctetArray rawTeds);
Args::UInt16 writeRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType, in Args::OctetArray rawTeds);
Args::UInt16 updateTedsCache(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType);

Using C++ programming, it was possible to encapsulate all functions into classes representing the modules defined by the standard and guarantee memory isolation between classes using private, protected, and public access specifiers. Through the inheritance of classes, it was possible to generalize code where it was possible and specify it where it was needed, contributing to a more robust and better-organized code base.

4.4 TI-RTOS

As previously stated, the standard defines multiple modules inside the Transducer Interface Module, whose tasks all need to run *at the same time*. As single-core processors can only run a task at any given moment, time-based multiplexing must be implemented in order to divide the time between tasks. Real-time Operating Systems(RTOS) are Operating Systems developed for Embedded Devices whose main goal is to coordinate tasks with time restrictions. TI-RTOS is a proprietary Real-time Operating System developed by Texas Instruments to be used along their micro-controllers. As it's supported by the micro-controller chosen and it supports the libraries written for the Tiva C platform, it was selected to be used in this project. TI-RTOS is built upon Texas Instruments SYS/BIOS real-time multitasking kernel. SYS/BIOS is designed for applications that require real-time scheduling and synchronization or real-time instrumentation [49].

These characteristics fit the work requirements thoroughly. The main modules utilized for development were: Tasks, Hardware (Hwi) and Software(Swi) Interrupts, Timing Services Clock and Timer, Heap allocation capabilities, as well as the Unified Instrumentation Architecture (UIA) debugging features.

TI-RTOS also features driver packages for the Tiva C platform. From these packages were utilized the following drivers: CAN, EEPROM, GPIO, SSI (SPI), Timer, System Tick, and UART.

4.5 Development Tools

Before developing the embedded code for the system, it was necessary to set the development toolset to define a basis for writing code, compile it, and flash it onto the micro-controller to proceed with the tests needed to validate functional requirements. The following tools were chosen: The embedded system was programmed using the C++ Language, utilizing the Code Composer Studio (CCS) Integrated Development Environment (IDE) as a compiler, linker, and flasher. The debugging process also utilized CCS as a tool. Git was used for tracking code changes and managing different versions of code.

4.5.1 Code Composer Studio IDE

The Code Composer Studio IDE is a tool developed by Texas Instruments used to create firmware for TI's micro-controllers. CCS consists of a suite of tools used to build and debug embedded applications, such as the one developed here. It was the primary tool used for this part of the development as it carries mechanisms for code optimization and the automatic inclusion of Texas proprietary libraries. It also allows for the deployment and debugging of the system through the Stellaris In-Circuit Debug Interface (ICDI) present in the micro-controller used for implementation.

4.5.2 Code Management - Git

While developing a code base, it is essential to keep note of the evolution of the code. For such endeavors, there are multiple tools for code versioning, the so-called version control systems (VCS). Git is one of the most famous tools. Its main work principle is creating a history line constituted of commits where changes are recorded in a differential manner. This differential technique means there is no need to save files in every commit entirely, only the changes relative to the previous commit. This VCS technique allows a lightweight approach to keeping code versions. This work used git because of its relative ease of use and previous knowledge of the tool. A git repository was created to manage the project. This repository was then synced with GitHub, an online platform that enables the synchronization of git repositories over multiple locations and multiple developers. GitHub tenses as a powerful tool for both version control and cooperative code development.

4.6 Summary and Conclusions

The chapter here presented explained the details of the implementation stack used to create the TIM described in this dissertation.

The main points referred to were:

- The use of the TIVA C hardware platform
- The development of a CAN shield to allow for the device to communicate through the CAN-Bus
- The language choice for developing the firmware, and the Interface Definition Language that the standard imposes
- The use of the TI-RTOS operating system, in order to create a real-time embedded software solution
- The Code Composer Studio IDE, which is used for programming and debugging the system
- The Git version control system to keep track of changes

The description of the tools used for development is important as some features will be restricted or made possible by the implementation layers onto which the firmware is built to. An example of this is the next chapter, which will contain implementation details specific to the platform utilized, although much of the implementation made intends to be independent of the platform (mainly from the microcontroller used).

Chapter 5

Implementation - Design and Structure

5.1 Introduction and Contextualization

IEEE1451.0 presents well-defined interfaces through the Module Communications API, presented in chapter 9 of [9]. In developing the software platform for the STIM, the module communications API was used as the primary source to define the software architecture. Around it, two main modules were developed: the IEEE1451.0 TIM services and the IEEE1451.X Communication Module, based on CAN. The present chapter explores the software architecture used for both modules. The first section will present general details of the implementation, and the following two sections will detail the implementation of the TIM Services and the implementation of the Communication Module.

Referring back to chapter 3 there is a crucial aspect to be mentioned: the interface between TIM Cards and TIM backplane. Although initially planned and the architecture presented TIM Cards, these are not employed in the current state of the work. Timing and resource limitations led the work to merge TIM Cards and Backplane into a TIM Module. This merge does not mean the architecture is irrelevant; it will only be left as future work as the other blocks of the system are fully ready to accept this change.

5.2 IEEE1451.0 Standard - System specification

Following the architecture proposed in chapter 3, the diagram presented in figure 5.1 depicts a top-level structure of the software produced to implement the standard.

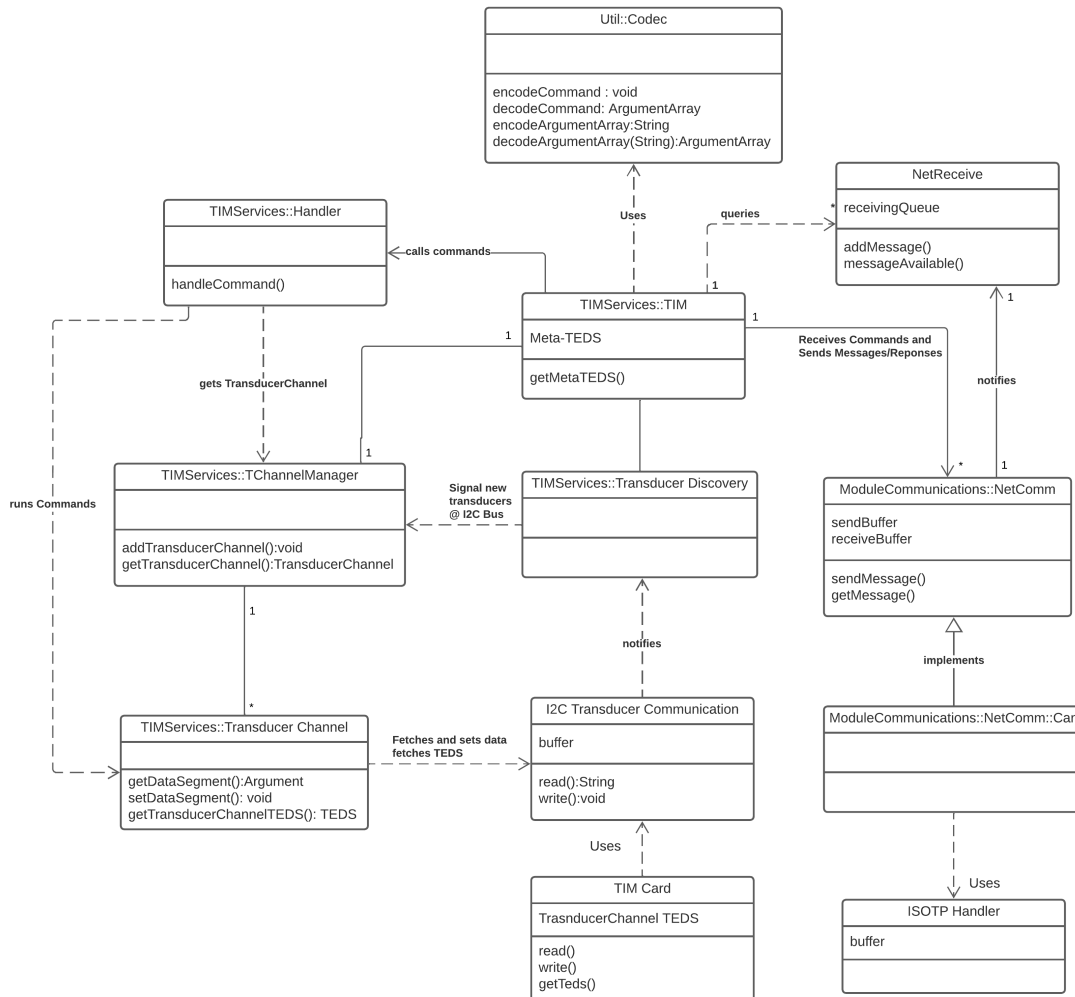


Figure 5.1: TIM architecture implementation UML class diagram

The next subsections will detail the developing blocks present in figure 5.1.

The classes present in the UML diagram are part of modules defined by the IEEE1451.0 standard. These modules are: The Args module, The Util Module, The ModuleCommunication Module, and the TIM Services Module. The interaction between these modules will also be explained alongside the Module description.

5.2.1 Args Module

The Args module defines how to encapsulate data for transportation between different layers and multiple devices. The standard defines the types of data that can be handled, how arguments are constructed with this data and how arguments are encapsulated into argument arrays.

5.2.1.1 Data Types

The IEEE1451.0 standard defines data types to be used in the Args Module. Arguments can hold data of the following types: Signed and Unsigned Integers of 8, 16, and 32 bits (UInt8, Int8, UInt16, Int16, UInt32, and Int32), Floating Point numbers of 32 and 64 bits (Float32 and Float64), Booleans, Octets, Strings, and arrays of all these types. Arguments can still hold representations for time instances (TimeInstance) and intervals of time (TimeDuration). The relation between data types and size is presented in table 5.1.

Table 5.1: IEEE1451 Data Types

		Size				
		8 bits	16 bits	32 bits	64 bits	M bits
Multiplicity	1	Unsigned Int8 (UInt8) Signed Int8 (Int8) Octet Boolean	UInt16 Int16	Int32 Int32 Float32	Float64	String (with M chars) Time_Duration Time_Instance QOS_Params Phy_UNITS
	N	UInt8Array Int8Array OctetArray BooleanArray	UInt16Array Int16Array	UInt32Array Int32Array Float32Array	Float64Array	String Array Time_Duration_Array Time_Instance_Array

5.2.1.2 Argument

The standard defines the Argument class, which encapsulates all these data types. Each argument can hold the data types presented in table 5.1. IEEE1451 presents an argument containing all data types. However, this is inefficient, and a waste of space as the program would have to allocate space for all data types. To solve this problem, in this specific implementation, a C++ union was used. Unions are data structures that can hold different data types using the same memory space. Where primitive data types are used, the variable is directly stored in a single 8 bytes memory space. When there is space for arrays, the union space is used to store the pointer to the array. This poses a significant aspect that allows the TIM to be developed using a micro-controller (due to memory restrictions). The argument class handles all memory allocation and freeing (mainly due to arrays) internally. Figure 5.2 presents a high-level description of the argument class.

The argument class also contains functions implemented to help with the serialization process, which will be presented next.

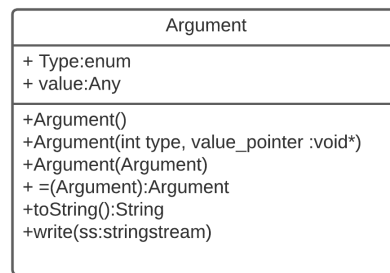


Figure 5.2: Argument Class

5.2.1.3 Argument Array

Argument arrays are objects which function as containers to hold arguments. Arguments can be added to Argument Arrays, where they are identified by position in the array or by a String key. The Argument Array mechanism allows for the insertion and retrieval of arguments. The main advantage of arguments and argument arrays is that the system doesn't need to know the types of the arguments at compile time but will infer them from data in runtime. Argument Arrays can be serialized and deserialized without data loss only with runtime knowledge.

Figure 5.3 presents a graphical representation of an Argument Array. Please note the object form and the serialized form of the Argument Array. The serialized form is used to transport argument arrays over the communication modules.

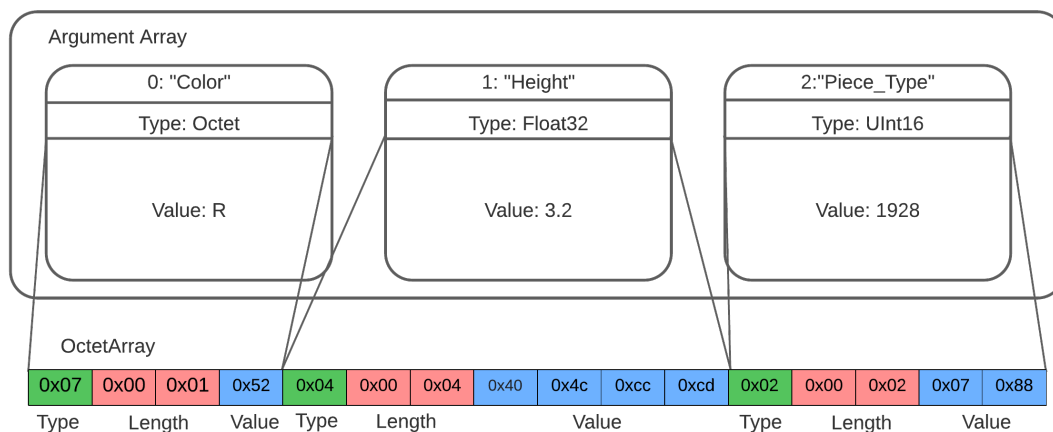


Figure 5.3: Argument Array (as an object and as a byte stream)

5.2.2 Util Module

The Util (Utilities) module is dedicated to providing the system with Codecs, that is, Coding and Decoding features. Each Codec is provided with functions that allow it to encode and decode the outgoing and receiving messages. Such is achieved through the functions presented in table 5.2. The functions presented work, in the same manner, as either a serializer would work, converting structured data and parameters into byte-stream ready to be transmitted, or a deserializer, converting a byte-stream into structured data.

Different communication modules may need different coding and decoding functions. The standard presents mechanisms for associating different codecs to different communication modules but such mechanisms were not implemented as they were deemed not relevant in the scope of this work.

Table 5.2: Codec functions (simplified)

Function	Input	Output
encodeCommand()	Destination Channel ID Command Class ID Command Function ID Command ArgumentArray	OctetArray
decodeCommand()	OctetArray	Destination Channel ID Command Class ID Command Function ID Command ArgumentArray
encodeResponse()	Response ArgumentArray	OctetArray
decodeResponse()	OctetArray	Response ArgumentArray
argumentArray2OctetArray()	ArgumentArray	OctetArray
OctetArray2ArgumentArray()	OctetArray	ArgumentArray

The encode and decode functions normally utilize the argumentArray to OctetArray Serializers and inverse deserializers to perform their tasks. The inputs and outputs of the serializers are represented in figure 5.3.

5.2.3 ModuleCommunication Module

The ModuleCommunication Module presents and defines the interfaces needed for flexible communication using different protocols and physical interfaces.

There are three main classes defined by the module: Comm, Receive, and Registration. How the module is conceptualized allows for independence of implementation for sending or receiving messages through either Point-to-Point (P2P) media or Networked Media. According to the architecture of this module, if there is more than one communication module, they ought to be registered into the Registration Class, and the reception of messages shall be signaled through the Receive Class (where message reception shall be stored).

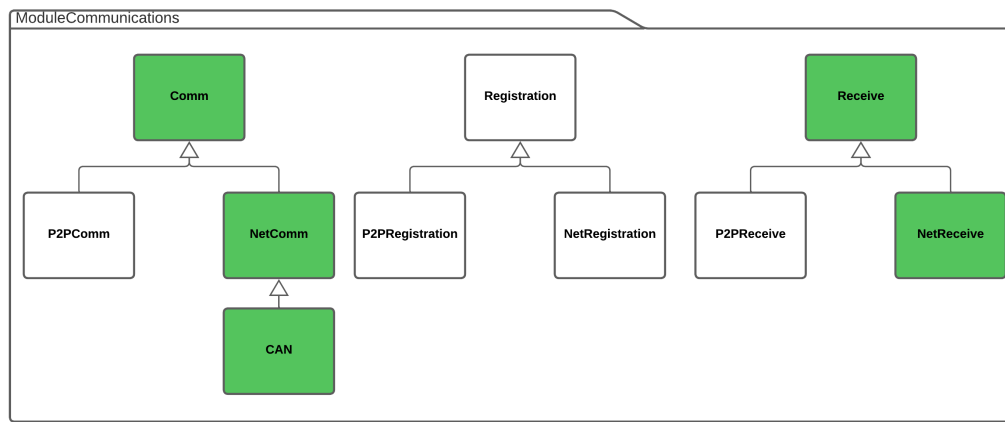


Figure 5.4: ModuleCommunication classes (implemented classes highlighted)

In the next subsections, the implemented classes will be explained in more depth.

5.2.3.1 Comm

Comm refers to the communication interfaces and holds functions common to either P2P Communication or Networked Communication. P2PComm implements the functions relative to simple non-networked communication topologies. NetComm implements the functions relative to networked communication. The implementation of NetComm is more complex as it requires the handling of different communication destinations and different communication sessions.

The main methods defined by the Comm interface are: `init()` for initialization; `shutdown()` for coordinating shutdown; and `sleep()` and `wakeup()` for coordinating transition to and from idle state.

5.2.3.2 NetComm

The NetComm interface class defines the methods for communication across networked communication topologies. Such is the case with this dissertation-specific implementation. The NetComm introduces the concept of communication Id (`commId`) and destination Id (`destId`). The

communication process is ruled by sessions. Each session has a communication Id, which it gets after the session is opened. To open a session, a destId is necessary, which equates to the address of another device in the network (such as an IP or CAN address). After the session is opened, messages can be exchanged between devices. Each time a device sends a command, it might receive a response which will be identified by the messageId (msgId).

This interface also specifies different functions for reading and writing messages and responses. Table 5.3 presents the most important functions of this module and which were utilized the most in the development process.

As the solution presented in this document refers to a TIM-like device, most communications will revolve around receiving commands from the NCAP and, according to the command received, responding or not to the requesting device. This gives singular importance to the functions readMsg() and writeRsp(). Communications will be held using an implementation of the NetComm interface through ISO Transport Layer(ISO 15765-2[10]) over the CAN protocol and physical media.

Table 5.3: NetComm Communication Methods

Function	Input	Output
open()	destID Expects Response:boolean	max Payload Length commId
close()		
readMsg()	commId Timeout	Payload; Payload length lastPayloadSegment:boolean
readRsp()	commId; Timeout msgId; bufferLength	Payload lastPayloadSegment:boolean
writeMsg()	commId; Timeout Payload; lastPayloadSegment	msgId
writeRsp()	commId; Timeout Payload; lastPayloadSegment	

5.2.3.3 NetReceive

The NetReceive class is meant to aggregate all messages received through the NetComm interfaces so that they are polled by the objects responsible for handling them. The solution currently utilizes an implementation of the NetComm interface so that all messages and responses received via the CAN interface generate a call to the NetReceive's notifyMsg() or notifyRsp() methods. Via these methods, the messages of requests or responses msgId's are stored in a buffer. The handlers poll the NetReceive Class to check whether or not a new message has arrived. If it has arrived, the handlers get the messageId correspondent with the message to be handled. The handler then retrieves the message directly from the CAN object buffer and handles it accordingly.

A simple sequence diagram for the reception of a message is presented in figure 5.5.

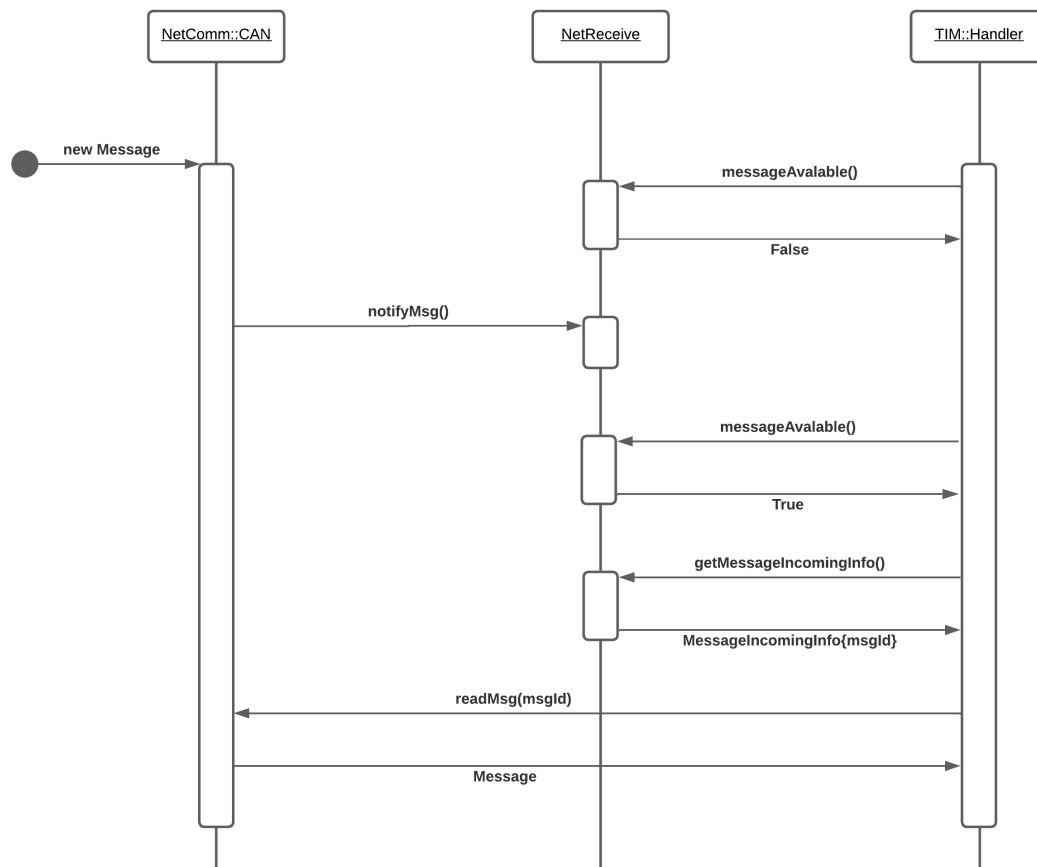


Figure 5.5: Message Reception Sequence Diagram

5.2.4 TIM Services Module

The TIM services module is the module that defines and implements the necessary classes and interfaces to allow for the implementation of the STIM functional model depicted by the IEEE1451.0 standard. The standard does not strictly define the TIM Services Module, so its structural architecture is left for the developing members to define.

A tentative class diagram of the TIM Services Module is presented in figure 5.6.

The main mechanisms behind the functional model are transducer channels. A different transducer channel represents each physical transducer and encapsulates all the functionalities needed for the operation and interaction with the said transducer. Accordingly to the Object Oriented Paradigm, each Transducer Channel behaves like a class and defines the Transducer Channel Class. The TransducerChannelManager class is responsible for storing and modifying a list of all transducer channels attached to the TIM at a given time.

The top class in the TIM Services Module is the TIM class. The TIM class holds all methods and attributes relative to the TIM or common to all Transducer Channels. Such is the case of the Meta-TEDS, which is stored and updated through the TIM class. The TIM class also handles the

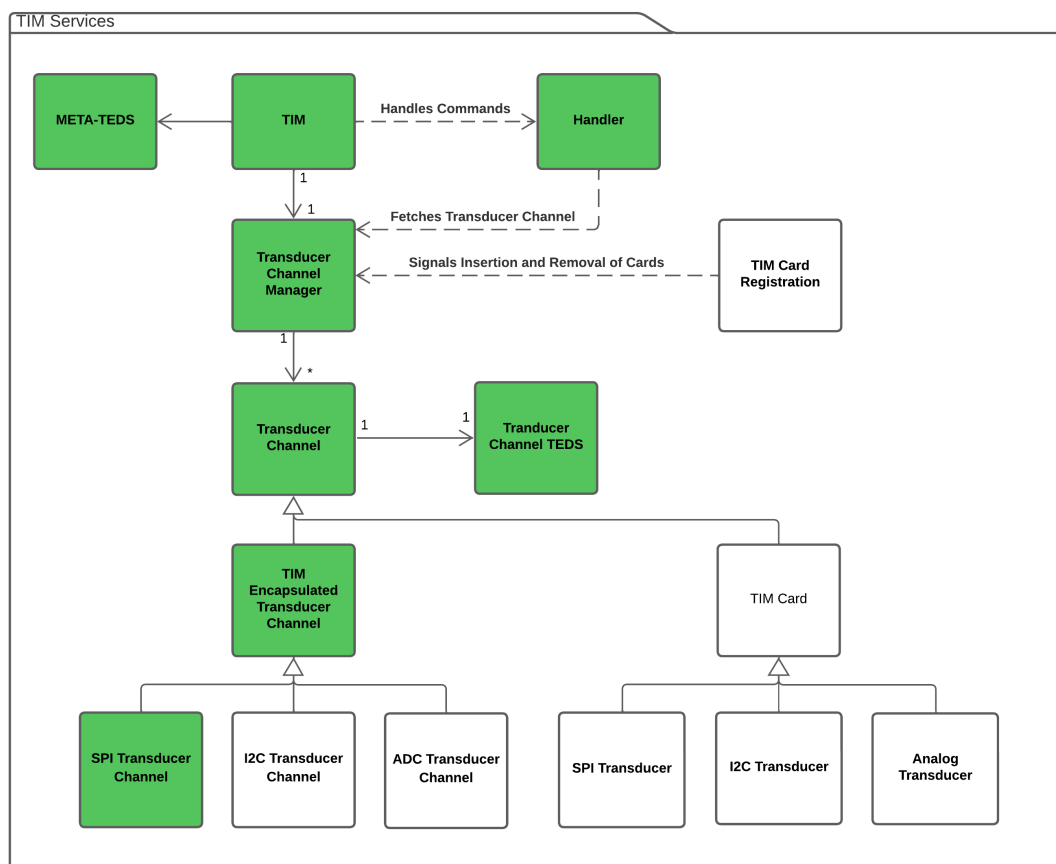


Figure 5.6: TIM Services Class Diagram (implemented classes in green)

commands received by the TIM through the handler class. If the command is related to the TIM itself, the TIM object handles it directly. If the command is tailored towards a transducer channel, the handler fetches the TC from the TransducerChannelManager and performs the actions directly on the TransducerChannel. Actions (commands) over a TransducerChannel include but are not limited to:

- Get and Set the TransducerChannelTEDS
- Get a reading from a sensor
- Configure the TransducerChannel way of polling the sensor
- Trigger a Buffered Read
- Set the output of an actuator
- Configure how an actuator might be triggered
- Get the Status of the Transducer Channel

Each of the actions above is mapped to individual commands which carry along arguments, passed through the Argument Array concept described above. As some commands require the TIM to respond, if that is the case, the TIM will handle the command and generate a response which will, in turn, be encoded and sent, via the communications module, back to the NCAP.

The next sections will expose the direct functioning of the system, mainly through sequence diagrams.

5.3 IEEE1451.0 TIM Services

The main purpose of the TIM Services is to provide a response for the commands requested by the TIM, which define the Status of the TIM, and the interactions the TIM shall make with the transducer channels, TEDS, which configurations the TIM shall accept.

To handle the TIM operations, a Task runs periodically to check the reception of messages through the NetReceive class. The command handling process is initiated if it has received a new command. If there is a new command, there is the need to get the said command from the CAN messages buffer. The command arrives in the form of an OctetArray, a byte-stream, and it needs decoding to be useful. For this, the TIM calls the Util::Codec class, which can decode the byte-stream into the command parameters, namely, the command class and the command function. Internally there is also decoding of the ArgumentArray from the byte-stream.

The reception of a command by the TIM Services module is represented in figure 5.7.

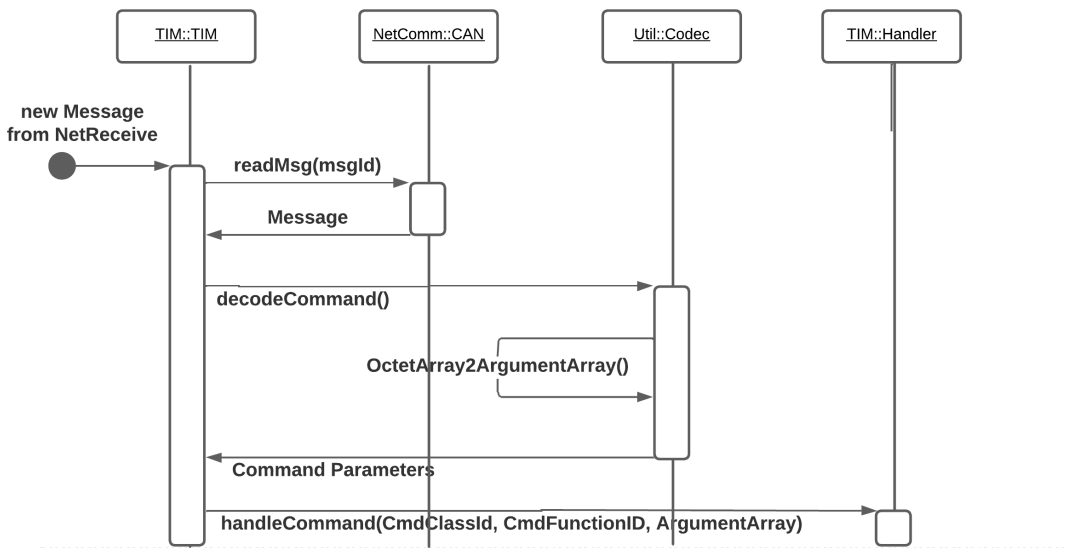


Figure 5.7: Sequence Diagram Command Handling: Receive

After the command is received and decoded, the Handler class moves to execute the function desired by the command. Commands are sorted by command class and command function. If the Command Class is **COMMON_CMD** it means the command intends to execute a function

that is relative to all transducer channels, or, in other terms, relative to the TIM itself. One of the most important concepts relative to all channels is the TIM Meta-TEDS which holds data about the number of transducer channels present in the system, the timing configurations for time-out events, and the unique identifier of the TIM (UUID), amongst others. This makes the retrieval of the Meta-TEDS especially important. For this function to be executed, the command needed is the READ_TEDS_SEGMENT command. The retrieval of the Meta-TEDS via a command is represented in the top part of figure 5.8.

Another essential, if not the primary function of the TIM is to retrieve data from sensors. The bottom part of figure 5.8 represents the fetching of data of a sensor present in the system, which is acting under the Immediate Sampling Mode. To complete this task, the handler must first get the Transducer Channel object from the TransducerChannelManager. After that, it has to request a DataSet from the Transducer Channel, which will, in turn, trigger a readout from the sensor. If the Transducer Channel is directly at the TIM, the TIM will have to wait for the sensor's response as is the case presented. After receiving the sensor readout, the data is encapsulated into an Argument.

In both of the cases presented above, there is the need to respond to the NCAP request for data, whether it be the TEDS or the DataSet.

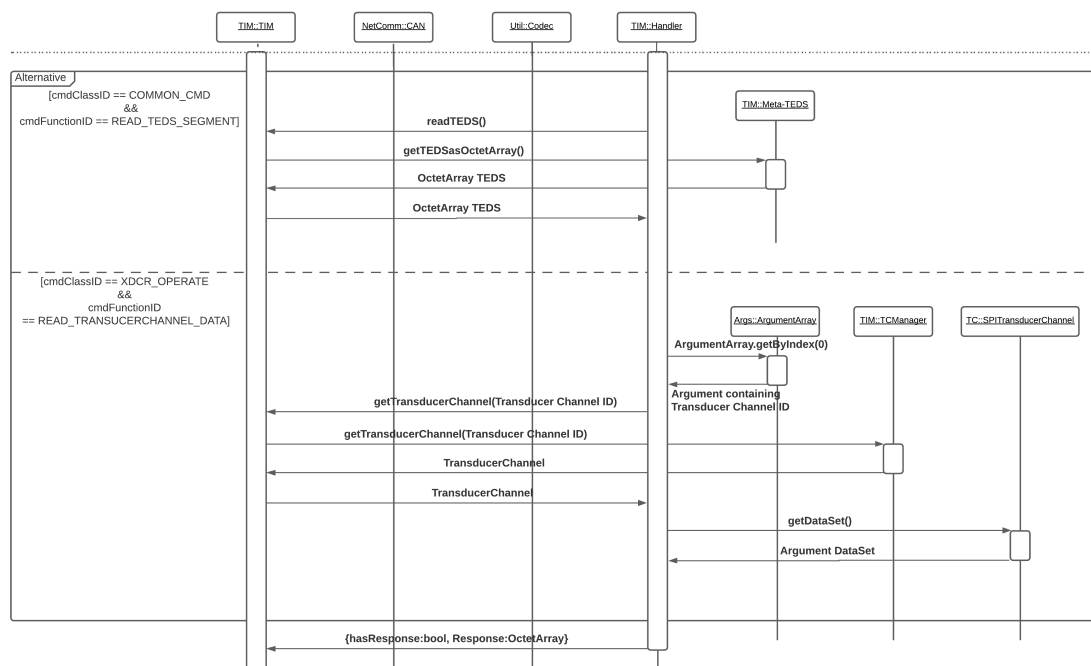


Figure 5.8: Sequence Diagram Command Handling: Handle

The last part of the handling of a trivial command is to respond to the command (if it requires a response). To achieve this first the TIM checks if the handler gave a response to the command via the hasResponse flag. If this is true, the TIM continues by calling the Codec Module to generate a Response byte-stream through the encodeResponse() method. This step also contains converting the response ArgumentArray Object to a byte-stream. This little step confirms the back-and-forth

functionality of the Codec Module. It is capable of dual-way transformation. After encoding the Response, the message is then passed to the CAN module, responsible for conveying this message through the CAN physical media. The operation hereby described can be seen represented in figure 5.9.

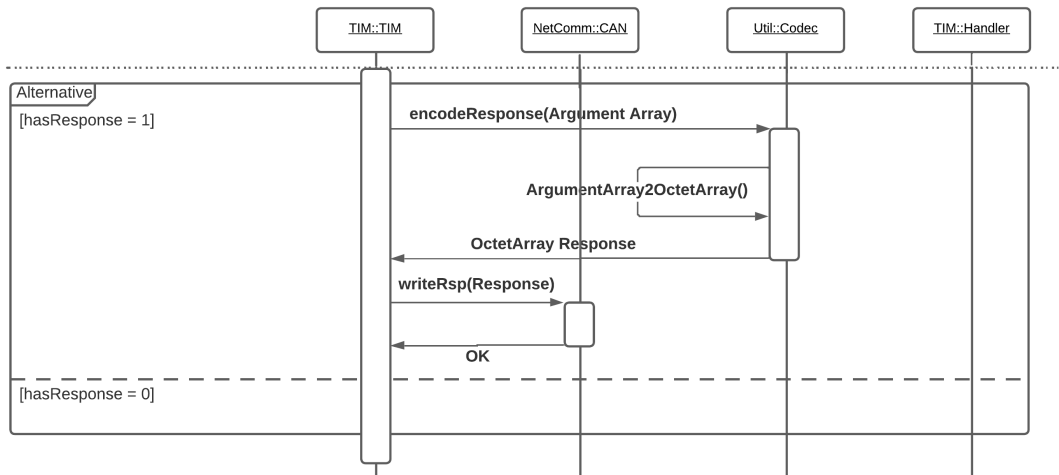


Figure 5.9: Sequence Diagram Command Handling: Respond

This section represents the main work path of the TIM Services Module triggered by the communication provided through the ModuleCommunications module. The inner workings and attributes of that module will be presented in the next section.

5.4 IEEE1451.X Communication Module : CAN

The IEEE1451.X Communication Module used in the present work is based upon a Controller Area Network topology. The development platform chosen supports CAN partially and, after adding the needed hardware to interface with the CAN Network development amongst this technology could be started.

As defined per the CAN specification, the maximum payload size for messages using CAN 2.0B is 8 bytes. This means only 64 bits can be transported at once, which represents sometimes less than a serialized argument length of data. To increase the payload transportation capacity, the ISO 15765-2, also known as ISO-Transport Layer (ISO-TP) was utilized.

5.4.1 ISO-TP - Transport Layer for CAN

ISO-TP is an ISO standard that defines a protocol for transporting packets of data larger than the 8 bytes provided natively by CAN. ISO-TP allows data blocks of up to 4095 bytes to be sent. It achieves this feature by segmenting the data blocks into 8-byte packets carrying flow control bytes which allow for the full reconstitution of the data-block at the receiving end.

The ISO-TP protocol defines 4 types of packets to be sent at low-level CAN:

- Single Frame (SF) - Allows for the transmission of data-blocks of up to 7 bytes in a single CAN packet
- First Frame (FF) - Defines the first frame of a multi-packet transmission. The first frame transmits the size of the data block being transmitted. It also carries 6 bytes of data.
- Consecutive Frame (CF) - Carries the following bytes of the data-block signaling the index of the current message
- Flow Frame (FF) - Carries the control logic that allows for the acknowledgment and flow control messages (timing requirements and maximum number of packets between flow control messages)

An ISO-TP communication with data blocks larger than 7 bytes session behaves accordingly to the figure 5.10^{1 2}.

Internally each ISO-TP message follows the structure presented in table 5.4.

Utilizing the concepts presented above, the system gains the ability to send messages of up to 4kB without needing to implement new flow control methods at the application layer. Furthermore,

¹STmin represents the minimum time interval between consecutive messages

²Block Size(BS) represents the number of consecutive messages between Flow Control Frames

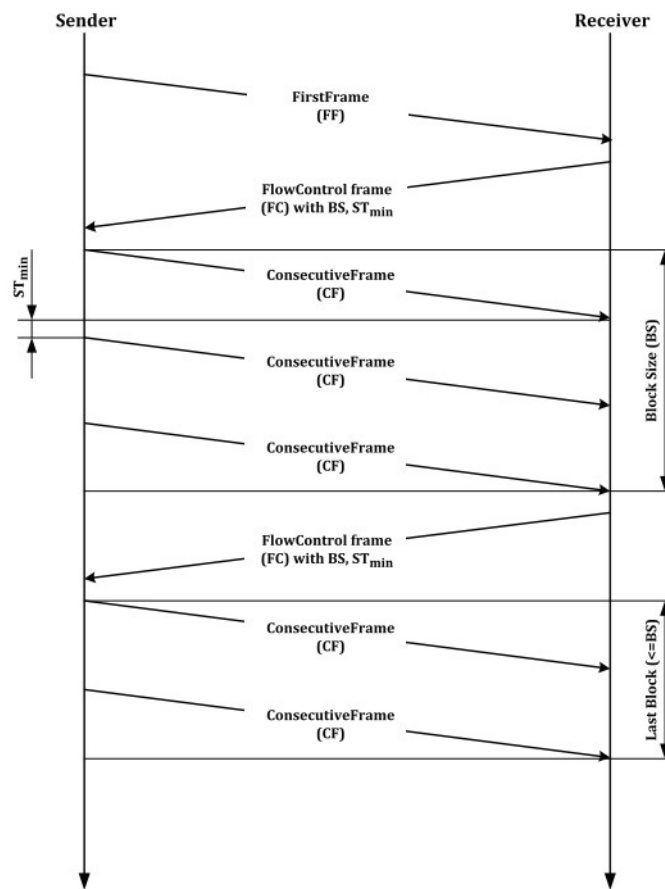


Figure 5.10: ISO-TP message flow [10]

it adds to the system by utilizing a well-known and tried test to implement a transport layer that is custom fitted for the CAN communication protocol.

To implement the ISO-TP transport layer, the isotp-c library by Shen Li³ was utilized. This library was chosen because it was implemented in C, being compatible with the language chosen - C++, and because it used dependency injection, making the ISO-TP layer independent of the CAN implementation utilized.

To utilize the ISO-TP library, there is the need to implement the following methods:

```
uint32 isotp_user_send_can(uint32 arbitration_id,
                          uint8 data[], uint8 size);
```

³Available @ <https://github.com/lishen2/isotp-c>, distributed under MIT Licensing

Table 5.4: ISO-TP message structures

MsgType/Bits	7 .. 4 (Byte 0)	3 .. 0 (Byte 0)	15 .. 8 (Byte 1)	23..16 (Byte 2)	Following Bytes
Single Frame	0	size (0..7)	Data A	Data B	Data C
First Frame	1	size (8..4095)		Data A	Data B
Consecutive	2	index (0..15)	Data A	Data B	Data C
Flow Control (FC)	3	FC flag (0,1,2)	Block size (0..255)	STmin	

a method that is capable of utilizing the CAN layer to send a simple CAN message with the defined arbitration message, and

```
uint32 isotp_user_get_ms();
```

a method to get the system time to calculate the end of the separation time and timeouts.

The system must be capable of forming two different ISO-TP connections, one that listens to directed communication and another that listens to Broadcast Messages. For this, two ISO-TP links must be initiated. An ISO-TP link can be initiated using the following method:

```
void isotp_init_link(IsoTpLink *link, uint32 sendid,
                    uint8 sendbuf[], uint16 sendbufsize,
                    uint8 recvbuf[], uint16 recvbufsize);
```

where *link* points to a struct holding the connection data, *sendid* stores the addressing schema used and the *sendbuf[]* and *recvbuf[]* point to the memory where layer shall store the buffered data, being received or sent.

To facilitate directed communication, the CAN Extended Addressing was utilized, allowing the arbitration ID, with 29bits, to carry information about which device sent the message and the destination it shall hold.

Prior to having communication between two nodes of a can network, it is necessary to know their CAN IDs. For this implementation, it is defined that the NCAP holds the address 0x01, and the address 0x00 is reserved for broadcast messages. Using this info, a simple handshake protocol was devised to provide the TIM with a CAN ID.

5.4.1.1 Handshake and discovery

For communication to happen between two nodes in a network, it is mandatory that they both know which device is communicating at a given time. For this, two approaches can be taken: either they use fixed addresses, which is not an easily scalable approach, or they have a way of agreeing on the ID for communication, the so-called handshake protocol.

The handshaking algorithm drafted is presented in figure 5.11.

It complies with the following structure: After the TIM is powered up, it sends the network a broadcast announcement message - A TIM-initiated message carrying a random 8-bit integer. If the NCAP is present in the network, it shall respond with the random sent by the TIM plus an ID number. If all matches, the TIM will pick this ID and set itself with these IDs for future communication. The ID assigning process is complete.

If the NCAP reboots after the TIM already has a CAN ID, the NCAP will send a broadcast TIM Discovery Command at startup. All TIMs which contain a CAN ID shall respond to this command with a broadcast message carrying the ID they are assigned to, reconstituting the communication in the network.

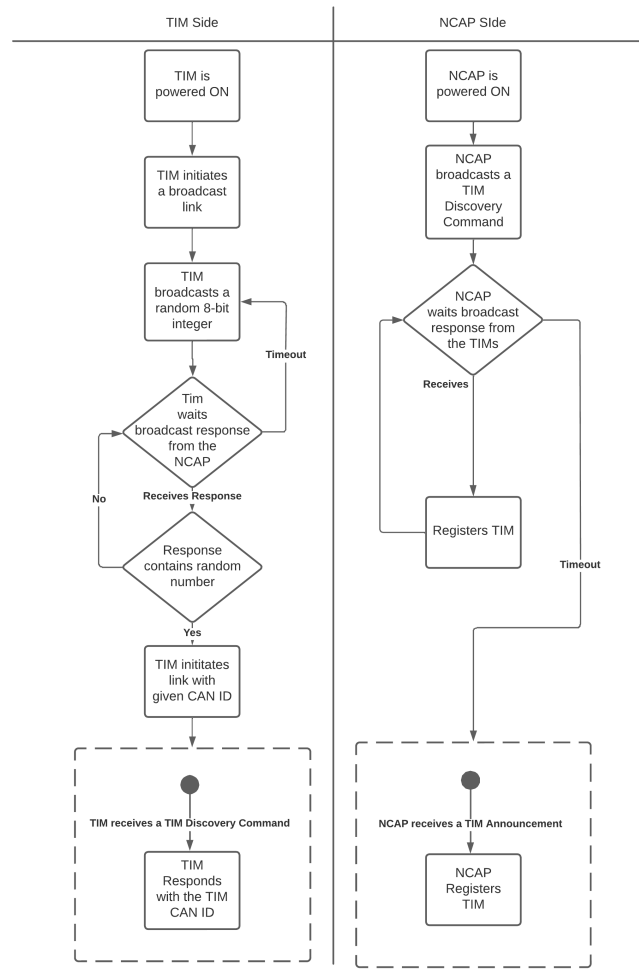


Figure 5.11: Handshake Protocol for ID attribution and discovery

A visual diagram representing a normal handshake process is presented in figure 5.12. It also shows the TIM discovery in case of NCAP reboot or insertion into an already running system.

All the communication made using the ISO-TP Protocol is only possible after the correct implementation of the CAN Layer.

5.4.2 CAN Layer

The CAN implementation sits at the lowest level of programming this dissertation work will meet since it interfaces directly with the CAN hardware module, incorporated into the microcontroller used for the TIM implementation.

Texas Instruments provides a low-level library (which is referred to as a Peripheral Driver) to interface with the hardware module, abstracting the programmer from direct access to registers. The main component used by the CAN module is the CANMsgObject (CAN Message Object). The CAN module supports 32 CANMsgObjects that can be configured individually to send or receive messages.

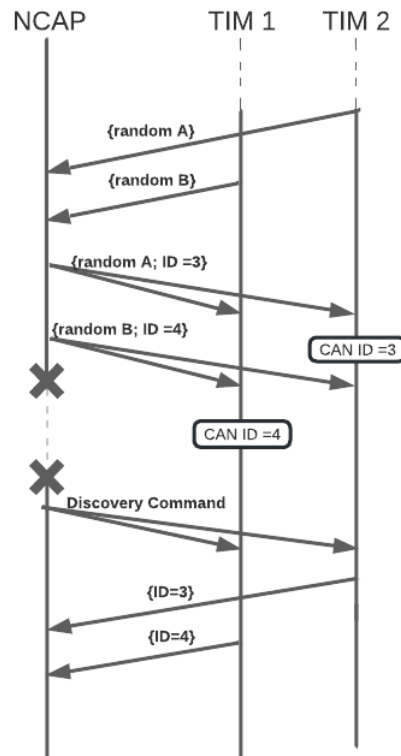


Figure 5.12: Sequence Communication Diagram for the Handshake Protocol

The CAN Message Object is constituted by the fields shown in table 5.5:

Table 5.5: Can Message Object Structure

CANMsgObject		
Type	Name	Description
UInt32	ui32MsgID	Defines the arbitrationID the CAN message will carry or expected from incoming messages
UInt32	ui32MsgIDMask	Defines the mask to filter incoming messages, if filtering is enabled
UInt32	ui32Flags	Defines the flags which control the behaviour of the module and carry status data
UInt32	ui32MsgLen	Defines the length of the message being sent or received
UInt8 *	pui8MsgData	Pointer to the message to be sent or received in memory

When configuring each object, the `ui32Flags` variable is extremely important as it will define the module's behavior. For the desired functioning of the CAN Bus, the reception of a message must cause an interrupt in the system so that the message is handled correctly. On the other hand, when the system wants to send messages, interrupts must be generated, so the system knows when the message is sent. For these interrupts to be raised, the reception object flags must carry the `MSG_OBJ_RX_INT_ENABLE` bit, and the sending object flags must carry the `MSG_OBJ_TX_INT_ENABLE` bit.

To send a message using this mechanism, the message object must be flagged as ready to be sent.

Message masking is also of importance because the system shall not be interrupted when random messages cross the CAN network and only when the messages are directed towards the TIM in question.

After enabling the interrupts for the messages, these interrupts must be handled by an Interrupt Handler, which is called when interrupts are raised. The interrupt handler commented with a behavioral explanation as follows:

```
//Interrupt handler for CAN communication
void CANIntHandler(void)
{
    uint32_t ui32Status;
    uint32_t origin;
    ui32Status = CANIntStatus(CAN0_BASE, CAN_INT_STS_CAUSE);
    //If the interrupt was caused by a STATUS Interrupt
    if(ui32Status == CAN_INT_INTID_STATUS)
    {
        //Save Control Flags Status
        ui32Status = CANStatusGet(CAN0_BASE, CAN_STS_CONTROL);
        //Save which Object raised the interrupt
        origin = CANIntStatus(CAN0_BASE, CAN_INT_STS_CAUSE);
        if(origin == 1 && (ui32Status & CAN_STS_RXOK)){
            g_ui32MsgCount++; //Direct message received
        }
        if(origin == 2 && (ui32Status & CAN_STS_TXOK)){
            g_clrToSend = 1; //Message has been transmitted
        }
        if(origin == 3 && (ui32Status & CAN_STS_RXOK)){
            g_ui32MsgBroadcastCount++; //Broadcast Message received
        }
    }
    (...)
}
```

Different message counters are updated every time a CAN frame is correctly transmitted or received, depending on the origin.

To handle these updates, there is a cyclic task, which runs periodically, preempting the task that oversees the TIM operations. This task is responsible for fetching the CAN frames from the hardware registers, buffering them, and passing them to the Iso-TP layer. Depending on the origin of the messages, the frames will be handled by different Iso-TP links, one for direct messages and another for broadcasted messages. This task is named `commTask()` and is responsible for triggering the `NetReceive` Functions for messages and responses.

5.5 Summary and Conclusions

This chapter versed on some implementation features deemed important for comprehending some of the main flows the system uses. It does not intend to be a comprehensive report of the software functioning but only show the most relevant parts.

Both the TIM Services Module and the ModuleCommunications Module are defining for the system architecture and play vital roles in the correct functioning of the system.

There are some missing features to make the TIM fully IEEE1451.0 compliant, but the features implemented at the time of writing this document follow the standard. This makes the TIM IEEE1451.0 compliant to some extent. There is also the TIM Cards feature which is not implemented but will be in future work.

All the software developed here is intended to be released as open-source to allow for modification, application, and improvement. The way the firmware is built intends to facilitate the introduction of new implementations of the interfaces using dependency injection. Using the developed and presented architecture it is also expected to help further developments at a future time.

Chapter 6

Sensor and Actuator Integration Results

6.1 Introduction and Contextualization

The Smart Transducing System intends to be integrated into industrial environments. Test scenarios were devised to show the different approaches that can be taken while using the developed platform.

This chapter intends to present realistic scenarios and measure and quantify the system's capabilities, from its maximum sampling speed to the time it takes to connect and handshake, among others.

Furthermore, it intends to display the advantages this system can bring to a shop floor in terms of the time it takes to deploy a simple sensing system using the proposed work.

6.2 Working Examples

Some simple scenarios were set to verify the system's correct functioning. These scenarios pretend to illustrate the plug-and-play capability of the system recognizing the TIMs installed in the network and the sensors and actuators that are Transducer Channels. For this example, temperature sensors were used as the transducer to be read.

There are three scenarios presented:

- One NCAP, One TIM, One Sensor
- One NCAP, One TIM, Four Sensors
- One NCAP, Four TIMs, One Sensor per TIM

Each of these approaches will be described and characterized in the following subsections.

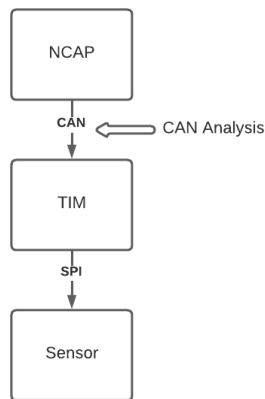
6.2.1 One NCAP-to-One TIM-to-One Sensor

The scenario hereby presented is the most simple one. There is only TIM in the system, which only contains one sensor. This replicates a typical networked sensor but is implemented using IEEE1451.0. For example, one could argue that this setup resembles the IEEE1451.2 TII definition, only exchanging the 10-wire interface for a CAN-based communication system.

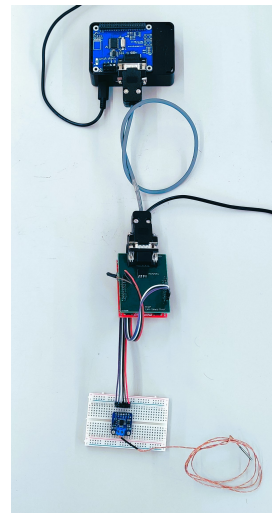
For this and the following examples, a simple SPI temperature sensor based around the MAX31855 Thermocouple-to-Digital converter was used. It communicates via an SPI interface that outputs the temperature readout along with some flags once triggered. The transducer channel is capable of decoding the temperature from the readout and outputting the temperature in Celsius.

It could, for example, be used in a simple temperature monitoring system in a greenhouse.

A basic diagram and setup pictures are provided in figure 6.1. Please note this setup does not make use of the CAN Hub.



(a) Diagram representing one-to-one-to-one system connection



(b) Physical representation of the connected one-to-one-to-one system

Figure 6.1: One-to-One-to-One System

6.2.1.1 Connection

When the system starts up, it begins by executing the handshake protocol with the NCAP. During the handshake, the NCAP provides the TIM with a communication ID. This is followed by a Meta-TEDS request by the NCAP. The TIM responds with the Meta-TEDS, which carries the information that the TIM currently holds one Transducer Channel. With this information, the NCAP requests the Transducer Channel TEDS to know what can be sampled from the Transducer Channel - the TIM responds. In this case, the temperature can be sampled. The system is now fully described and ready to start sampling. In order to fully comprehend the communication process that goes behind the connection and registration process, the specific CAN frames carried over the CAN BUS are presented and explained in fig 6.2.

1659707140.728992	18DA0102	08 07 10 01 00 00 80 01	3F	Random
1659707140.795567	18DA0201	06 05 10 01 01	3F 04	Handshake Protocol
1659707140.822713	18DA0401	08 10 13 00 04 00 00 01 02		
1659707140.823969	18DA0104	08 30 01 14 00 00 00 00 00		CAN ID
1659707140.875514	18DA0401	08 21 00 0B 01 00 01 01 03		
1659707140.875927	18DA0104	08 30 01 14 00 00 00 00 00		
1659707140.923630	18DA0401	07 22 00 04 00 00 00 00 00		Meta-TEDS Request Command
1659707140.926228	18DA0104	08 10 37 00 04 01 00 32 03		
1659707140.944737	18DA0401	03 30 08 00		Flow Control – ISO-TP
1659707140.945944	18DA0104	08 21 00 04 00 00 00 00 12		
1659707140.946929	18DA0104	08 22 00 28 00 00 00 24 03		
1659707140.947936	18DA0104	08 23 04 00 01 01 01 04 0A		
1659707140.948923	18DA0104	08 24 81 C0 F9 74 48 81 F5		
1659707140.949924	18DA0104	08 25 62 2E 78 0A 04 3F 00		
1659707140.950955	18DA0104	08 26 00 0C 04 C0 A0 00		Meta-TEDS Response from TIM
1659707140.951946	18DA0104	08 27 00 0D 02 00 01 F8 82		#Transducer Channels
1659707141.016848	18DA0401	08 10 13 00 05 00 01 01 02		
1659707141.017945	18DA0104	08 30 01 14 00 00 00 00 00		
1659707141.079875	18DA0401	08 21 00 0B 01 00 01 03 03		
1659707141.080954	18DA0104	08 30 01 14 00 00 00 00 00		TranducerChannelTEDS Request
1659707141.147487	18DA0401	07 22 00 04 00 00 00 00 00		Command
1659707141.150409	18DA0104	08 10 72 00 05 01 00 6D 03		
1659707141.163670	18DA0401	03 30 08 00		
1659707141.164971	18DA0104	08 21 00 04 00 00 00 00 12		
1659707141.165965	18DA0104	08 22 00 63 00 00 00 5F 03		
1659707141.166978	18DA0104	08 23 04 00 03 01 01 0A 01		
1659707141.167949	18DA0104	08 24 01 0B 01 00 0C 06 32		
1659707141.168956	18DA0104	08 25 01 00 39 01 82 0D 04		
1659707141.169934	18DA0104	08 26 43 69 00 00 0E 04 43		
1659707141.170950	18DA0104	08 27 B0 80 00 0F 04 40 00		
1659707141.171951	18DA0104	08 28 00 00 10 01 01 12 09		
1659707141.227922	18DA0401	03 30 08 00		
1659707141.228967	18DA0104	08 29 28 01 00 29 01 02 2A		
1659707141.229953	18DA0104	08 2A 01 0C 14 04 3D CC CC		
1659707141.230953	18DA0104	08 2B CD 16 04 37 D1 B7 17		
1659707141.231946	18DA0104	08 2C 17 04 3D CC CC CD 18		
1659707141.232957	18DA0104	08 2D 04 41 F0 00 00 19 04		
1659707141.233948	18DA0104	08 2E 37 D1 B7 17 1A 04 40		
1659707141.234991	18DA0104	08 2F A0 00 00 1F 03 30 01		TransducerChannelTEDS
1659707141.235961	18DA0104	08 20 02 EF 2C 00 00 00 00		Response from TIM

Figure 6.2: CAN Messages Sequence Between TIM and NCAP

6.2.1.2 Sampling

The Transducer Channel inserted into the presented system is configured into immediate sampling mode. This means when the NCAP sends a command requesting the TransducerChannel Data Segment, the TIM will generate a request for the Transducer Channel to go ahead and sample the sensor. After the sensor is sampled, the TIM generates the response and sends it to the NCAP, carrying with it a success flag. If the operation could not be carried out, the success flag would not be set.

Again, the specific CAN frames for the operation are presented under figure 6.3.

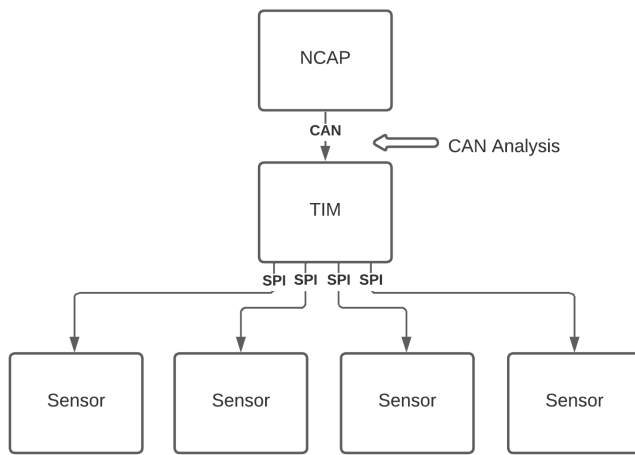
6.2.2 One NCAP-to-One TIM-to-Many Sensors

This scenario intends to demonstrate multiple transducer channels inserted into one system. As the example still uses temperature sensors, it could equate, for example, to a temperature readout in a robotic arm, sampling the temperature of different motors. The primary behavioral difference,

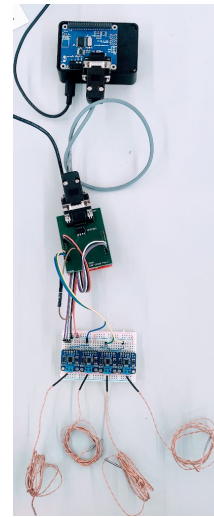
1659713175.671488	18DA0401	08	10	0F	00	0A	00	01	03	01	msgID	channelID
1659713175.672100	18DA0104	08	30	01	14	00	00	00	00	00	cmdClassID	cmdFunctionID
1659713175.734548	18DA0401	08	21	00	07	03	00	04	00	00		
1659713175.735102	18DA0104	08	30	01	14	00	00	00	00	00	Transducer Channel Read	
1659713175.788783	18DA0401	03	22	00	00						Read DataSegment Command	
1659713175.791926	18DA0104	08	10	11	00	0A	01	00	0C	03	successFlag	
1659713175.810002	18DA0401	03	30	08	00							
1659713175.811087	18DA0104	08	21	00	04	00	00	00	00	02		
1659713175.812068	18DA0104	08	22	00	02	00	18	00	00	00	Response from TIM	

Figure 6.3: CAN Messages for TransducerChannel DataSegment Readout

in this case, is the sampling of different transducer channels. This will equate to the change in the Meta-TEDS number of transducer channels and will enable the TIM to sample multiple transducer channels.



(a) Diagram representing one-to-one-to-many system connection



(b) Physical representation of the connected one-to-one-to-many system

Figure 6.4: One-to-One-to-Many System

Please note that whilst the system configuration changed quite a lot, the architecture allows the system to operate with little updates to the TEDS. At this moment, the SPI channels must be manually introduced into the software, and, still, the implementation effort is minimal. This is all thanks to the flexibility allowed by the software architecture and implementation.

6.2.3 One NCAP-to-Many TIMs-to-One Sensor

The last scenario to be presented here is one where the temperature sensors are distributed amongst different TIMs in the network. This scenario intends to demonstrate the ease of deployability of the system in a distributed scenario. It is as simple as connecting the TIMs to the network. No further configuration is required. The CAN Hub is required but only because of how the CAN bus is physically implemented.

A great example of a real use-case under this scenario would be a drying paint oven in a car manufacturing factory. The temperature sensors could be deployed over great distances (the current spec allows for 100m according to table 1 of [50]). This would allow for the sampling of the temperature gradient over distances, along with providing data for data-analysis processes and allowing for the data that is retrieved to be fed into the control loop of the process.

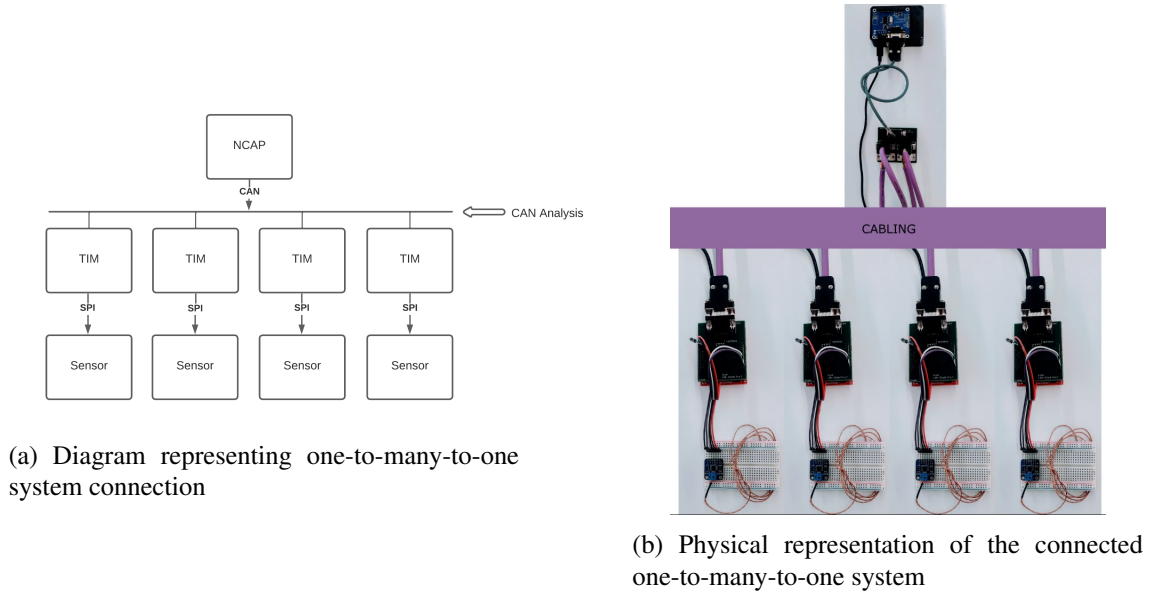


Figure 6.5: One-to-Many-to-One System

6.3 TEDS measurements

Physical and implementation-related limitations bound the operation of the TIMs. Such is the case for the timings for interacting with transducer channels. These timing-related limitations are described using the Transducer Channels. In order to facilitate comprehension, consolidate the concepts utilized and measure the system performance, timing measurements were taken in order to define the following TEDS fields:

- TransducerChannel sampling period
- TransducerChannel update time
- TransducerChannel read delay time
- TransducerChannel warm-up time

As the fields are directly defined by the IEEE1451.0 standard, each subsection contains the definition per-standard [9]. The sensor characterized is the same used in the system described in the previous section.

6.3.1 TransducerChannel Update Time / Sampling Period

TransducerChannel Update Time: “This field is required for all TransducerChannel TEDS. For TransducerChannels being operated in the free-running without pre-trigger sampling mode or the free-running with pre-trigger sampling mode, this number should be equal to the sample interval.”

According to the implementation of the TransducerChannel, the sample time is the same as the TransducerChannel Sampling Period.

TransducerChannel Sampling Period: “This field is required for all TransducerChannels.”“The TransducerChannel sampling period (tsp) shall be the minimum sampling period expressed in seconds of the TransducerChannel unencumbered by read or write considerations (since there is no requirement to read or write with each trigger)”“. If reads or writes are involved, the actual sampling rates are further limited by setup and data transfer times depending on the TransducerChannel type. ”

In light of the implementation that is currently being tested, the author is presented with an SPI-dependent communication sensor readout. As that is the case, the sampling time is defined as the timing difference between the data request through SPI and the full reception of data.

```
//Test for defining TC Sampling Period
for(int i=0;i<100;i++){
    time=micros;
    tchannel->getDataSet(argforread);
    time = micros-time;
    fprintf(stdout,"%lu\n",time);
}
}
```

The data retrieved from these measurements contained a really consistent timing measurement with 96 measurements of 56 microseconds and four measures of 57 microseconds. Considering the worst measurement, this equates to a maximum frequency for polling of around 17 kHz.

These parameters should be set as **0.000057** seconds.

6.3.2 TransducerChannel read delay time

TransducerChannel read delay time: “This field contains the maximum delay time (tch), expressed in seconds, between the receipt of a Read TransducerChannel data-set segment command and the beginning of the transmission of the data frame”

This time accounts for the overhead between the reception of a command, the decoding of the command, the command handling, transducer channel command execution, response generation, and response encoding. As some of these points are divided into different tasks (with one of them only being executed every millisecond), it is expected for it to take upwards of 2ms.

For this measurement, the following test was implemented:

1. The NCAP sends a read command

2. The TIM saves the time the command has fully been received
3. The TIM executes the command
4. The TIM saves the time it sends the response

The code for this test is as follows:

```
//Save time at the reception of the isotp message
extern uint32_t time_u;
extern uint32_t micros;
if(receiving_ret==ISOTP_RET_OK){
    time_u=micros;
    addMessage(0x01,(const char*)receiveBuffer,(size_t) receiveSize);
}

/*****/

//Save time difference right before sending the iso-tp message
time_u= micros-time_u;
fprintf(stdout,"%lu\n",time_u);
fflush(stdout);
UInt32 ret =
    isotp_send(&g_link,(uint8_t*)payload.c_str(),payload.size());
```

After 100 measurements were taken, the following boxplot was generated in figure 6.6.

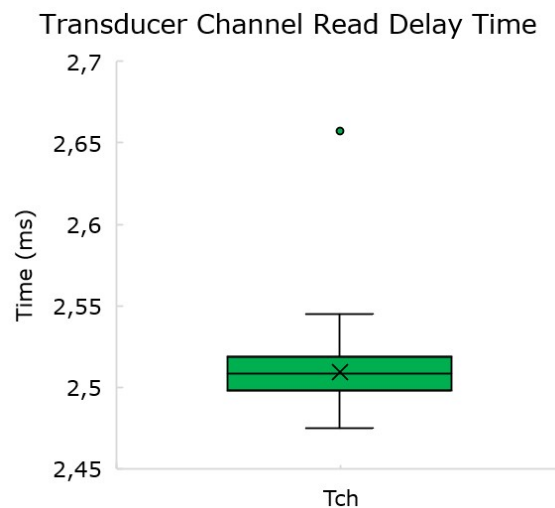


Figure 6.6: TransducerChannel Read Delay Time Measurement

The average time for the read delay was 2,51 ms, with the maximum being 2,66 seconds. This information tells that, considering the worst case, the system (without communication) can be probed about 250 times per second. This time could be improved, but it would limit the future work on buffered reads, i.e., the capability of the system reading without a trigger.

This field would be set with a value never inferior to **0.0027** seconds.

6.3.3 TransducerChannel warm-up time

TransducerChannel warm-up time: “This field contains the period of time, expressed in seconds, in which the TransducerChannel stabilizes its performance to predefined tolerances, as specified in Uncertainty under worst-case conditions (see 8.5.2.7), after the application of power to the TransducerChannel.”

This TEDS field is an excellent example of the parallelism drawn between the Transducer Electronic Datasheet and the real Datasheet of a transducer. In this case, the sensor manufacturer provides the information that the sensor needs 200 ms to stabilize and give accurate readings, as seen in figure 6.7.

Thermocouple Conversion Power-Up Time	t _{CONV_PU} (Note 6)	.	200	ms
--	-------------------------------	---	-----	----

Figure 6.7: Transducer warm-up time as defined per manufacturer [11]

This field should be set as **0.2** seconds.

Note: The previous measurements were taken considering one TIM with only one sensor. Adding more transducer channels will imply a greater load in the system, implying, naturally, worse timing conditions

6.4 Handshake Timing Analysis

One of the biggest features of the work here presented is the capability of self-registration and self-identification in the CAN network. These capabilities are even more relevant if they are fast enough that they go unnoticed. The time between **plug** and **play** defines the relevance of this term.

There are 2 ways of handshake proposed by this work. The TIM can announce itself with a random number and receive an ID from the NCAP or, if the TIM was already given an ID previously, the NCAP can send a discovery command to which the TIM shall respond.

6.4.1 TIM Announcement

Following the protocol explained in chapter 5, the time for the handshake will be dependent on how many TIMs are present in the network. However, this is not a factor that the TIM is not responsible for as the TIM is the initiator of the request to identify. Still, the measurement of a simple handshake in a one-to-one configuration seems relevant.

The time measurement is made from before sending the request for ID to the moment the TIM receives an ID. A boxplot, presented in figure 6.8 was generated from 100 measures. The average TIM announcement protocol execution took 33,76 ms.

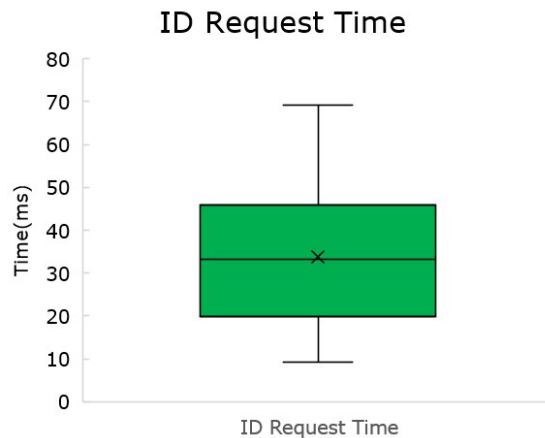


Figure 6.8: Time for TIM-initiated handshake (accounting for communication)

6.4.2 TIM Discovery

The TIM discovery process functions as a response to a command by the NCAP. While measuring the time it takes for the system to be discovered it is not dependent on the TIM implementation, but on the whole system instead. It is noted, however, that a response to the NCAP discovery command shall take no longer than the TIM Announcement protocol.

6.5 TEDS readout timings

The capability of fetching the TEDS is vital in the functioning of the system. The time this process takes directly influences the time for integration of a new TIM into a network. To discover how much time a TEDS readout takes, two different scenarios were set: the readout of a Meta-TEDS and the readout of a Transducer Channel TEDS.

6.5.1 Meta-TEDS

The time for reading a Meta-TEDS was measured from the CAN network analyzer and is set from the first frame of the TEDS request command to the last frame of the response.

100 measures were taken, from which a boxplot was graphed, which can be seen in figure 6.9. The average value of a Meta-TEDS request was measured at **88,5** ms, for a Meta-TEDS of 55 bytes.

6.5.2 Transducer Channel TEDS

The test protocol follows the Meta-TEDS readout time protocol presented above.

100 measures were taken, from which a boxplot was graphed, which can be seen in figure 6.10. The average value of a Transducer Channel-TEDS request was measured at **205,6** ms, for a Transducer Channel-TEDS of 114 bytes.

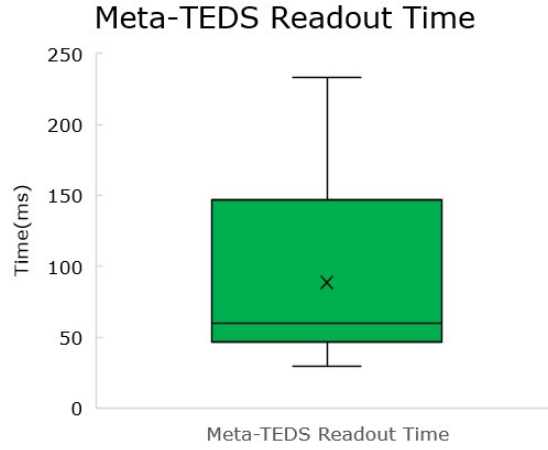


Figure 6.9: Time for Meta-TEDS readout(accounting for communication)

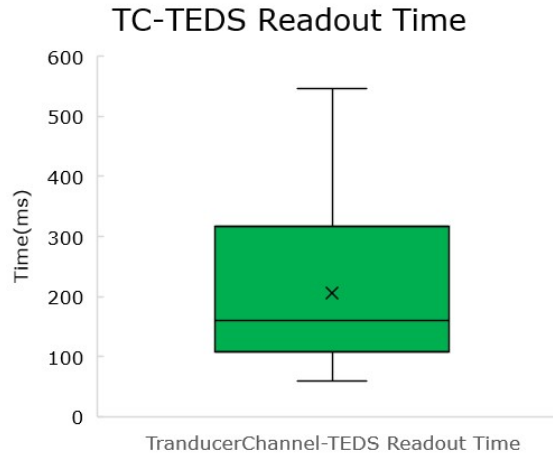


Figure 6.10: Time for TransducerChannel-TEDS readout(accounting for communication)

6.6 Integration Timing - Methodology proposal

The integration timing is considered to be measured from the moment the TIM is connected to the network (or is powered up) to the moment the NCAP receives the last Transducer Channel TEDS.

This integration timing is mainly affected by 5 parameters: the number of TIMs in a network, the handshake time, the time for a META-TEDS request command, the number of transducer channels in a TIM, and the time each TransducerChannelTEDS request command would take to be handled.

Considering all these factors, a formula was drafted to estimate the time for integration of a Smart Transducer System, presented below, in equation 6.1.

$$t_{integration} = \#TIMS \times (t_{handshake} + t_{META_TEDS}) + \sum_{i=0}^{\#TIMS} (\#_{TransducerChannels_i} \times t_{TC_TEDS}) \quad (6.1)$$

Considering the cases presented above, estimations were made for the time for integration of the three different cases.

Please take as reference the average captured times of $t_{handshake} = 33,76$ ms, $t_{META_TEDS} = 88,5$ ms and $t_{TC_TEDS} = 205,6$ ms.

The results for the different cases were computed and are presented in table 6.1.

Table 6.1: Estimation of Integration Time for different scenarios

	Case One-to-one-to-one	Case One-to-one-to-many	Case One-to-many-to-one
#TIMs	1	1	4
#TC/TIM	1	4	1
$t_{integration}$	327,8 ms	944,6 ms	1311,2 ms

To verify the results in the table above, a verification test was made using the one-to-one-to-one system. It resulted in the times presented in figure 6.11.

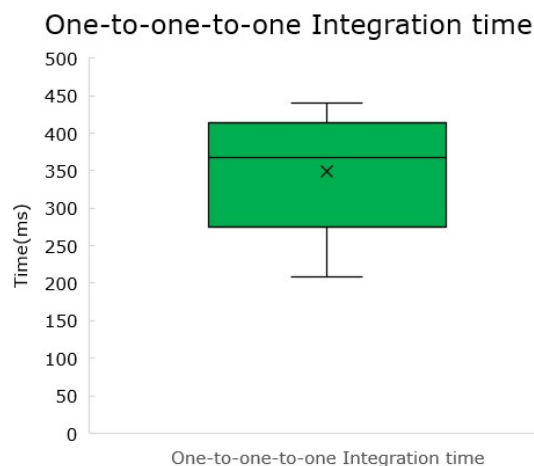


Figure 6.11: Integration time of one-to-one-to-one system

The results average at an integration time of **348,8** ms, against the 327,8 ms suggested by the estimation formula. The difference between expected and measured is about **6%** and is attributed to the variation of the communication timings, which reflect in a large variance from sample to sample.

The formula, while only standing for a loose estimation is capable of giving the system integrator a tool to figure out how much time the integration of the system will take.

Please note that the integration times account for the hot insertion of the TIM in the system, i.e., without turning the system on and off, which is typical of connecting a sensor or actuator to the system. Being able to introduce a transducer in a system and have it ready to read in a matter of seconds poses a big advantage for system integrators, upgraders, or maintainers.

6.7 Summary and Conclusions

This chapter presents the results achieved in this dissertation work.

It starts by presenting working examples of transducer systems based on the TIM developed. The presented examples illustrate a large variety of applications a transducer system might be used to. It also presents explanations of the CAN frames that allow the system to function.

Next, some TEDS fields are measured/calculated. These measurements are a process that is important for the definition and description of the system. These measures also detail the capabilities of the system. A sample frequency of 17kHz for a single sensor was achieved through an SPI transducer channel. The system is also capable of being requested readings at a frequency of 250Hz.

Following, the time for the handshake was analyzed from the TIM point of view. The system is capable of announcing itself and receiving an ID in under 34 ms. This is, without doubt, faster than having a human introduce an ID for the TIM. This is another great advantage of this system.

TEDS readout times are also measured, to understand how much time the system takes to be integrated into an IEEE1451 system. Measurements came to the order of magnitude of 100ms.

Finally, a methodology is proposed for measuring integration timing. This work considers the integration timing to be measured from the insertion of the system to the full description of the system being known by the NCAP. An estimation formula is proposed and estimations are provided for the examples provided at the beginning. To verify the accuracy of the estimation formula, one of the systems is tested, proving the formula to be a reasonable estimation.

Chapter 7

Future work

No work can be concluded without analyzing how it could be further developed. The next chapter presents the work's limitations, and how those limitations could be improved with future work.

One of the implemented system's main purposes was to pose as a proof concept to advocate for the advantages of using the IEEE1451 standard. That objective was successfully achieved. However, there still is space for improvement.

7.1 TIM Cards

One of the most interesting parts that were left unexplored was the concept of the TIM cards. This concept still poses as highly relevant and was only not developed due to lack of time. The process of design and parts selection was started, namely the I^2C buffering stage, developed around the PCA9513A Hot-swappable I^2C buffer, responsible for the seamless plug-and-play feature. Initial hardware development can be seen in figure 7.1.

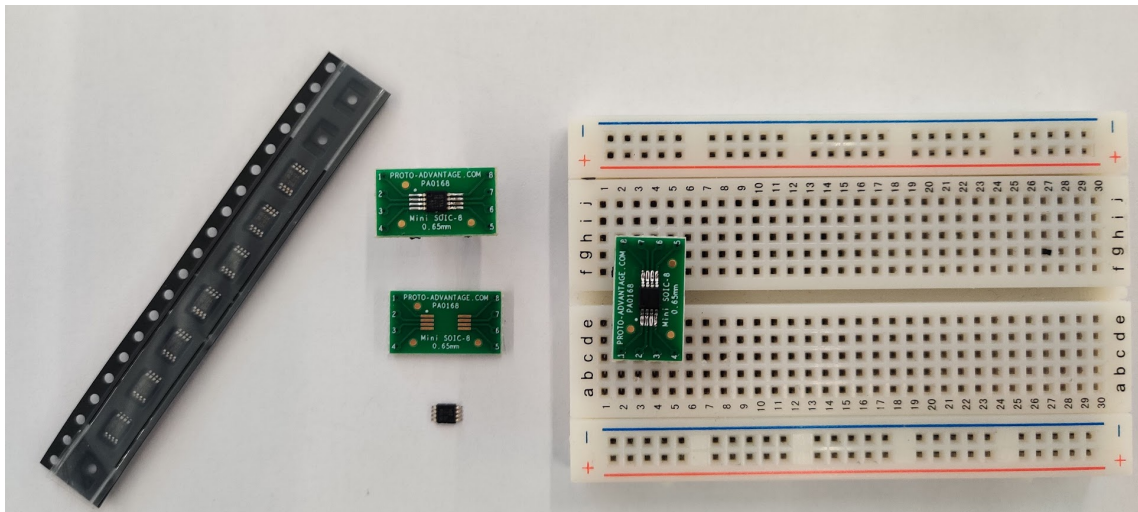


Figure 7.1: Buffering Stage of TIM Cards initial developments

Future work will have to reside in this concept in order to expand even further the plug-and-play capabilities of the transducing system.

7.2 TEDS programmer

At the moment, TEDS changes have to be made through the NCAP or through the change of firmware. While this is a completely feasible approach, which does not require much effort, another interesting approach would be to have dedicated software with the sole intention of programming offline TIMs, that would be inserted later into the system already fully described.

Such a programming software would also be useful to program TransducerChannel TEDS into TIM Cards later since the TIM Cards will not be running the TIM-command-handling-capable software, which allows the previously mentioned way of programming.

An interesting approach would be to expand the software already developed by the lab for TEDS generation [47] to support flashing the TEDS via UART communication.

7.3 Implementation of new commands

The main mechanisms for compliance with the IEEE1451.0 standard already were developed under the work presented. However, to deem a system fully IEEE1451.0 compliant, work would have to be done to implement all commands required by the standard, which would introduce some new and interesting features, but which were not an objective of this work.

Transducer Channel Groups, Vector Channel Groups, and Buffered Reading would be some features that would be very interesting to work with, once implemented.

7.4 Improvements on the internal functioning of the TIM

The implementation of the TIM follows strictly the architecture presented in the previous chapters. This is a very important feature this work has. However, the implementation of the apparent concurrent processing which is made by the TIM could use some work on the way the tasks are handled, and improvements could be made on the way tasks are generated.

Some further developments focus on memory management, garbage collection, and the interaction between tasks and task scheduling would also be beneficial to the project.

Chapter 8

Conclusions

The present dissertation work presents the development of a plug-and-play module for smart sensors/transducers in industrial environments. This module was developed in line with the IEEE1451 standard, which is thoroughly explained in this document.

Before the development work, a literature review was conducted in order to understand the origin of the term Smart Sensors (Transducers), the origin of IEEE1451, its core features, and some applications implemented using this standard. An update of the standard (IEC21451) was also overviewed and main changes were presented. From the literature review, it was concluded that there was no general and open solution for the usage of IEEE 1451 in industrial environments. This conclusion justified the efforts put into this dissertation work.

Aligning the motivation with the literature review, an IEEE1451-based architecture was presented. This architecture aims to increase flexibility through the encapsulation of hardware, software, and plug-and-play features. It presents a tripartition of the TIM into TIM Backplane, TIM Cards, and Backplane-to-Cards Interface. This architecture is also based on an NCAP-to-TIM Interface based on CAN. While this architecture was not fully implemented, concerning time and resource limitations, its design decisions are still deemed relevant and motivate future work in this implementation.

As this work configures an embedded system development, the design and development platform is described and characterized, along with the reasons why each of the components was chosen. This section is expected to provide others, who face similar implementation challenges, to have a solid development environment, to more easily achieve their targets. Much of these implementation decisions forced the author to contact with new technologies, which had an impact on the speed of development. This factor is relevant although it did not restrict the coming to results.

After this, the implementation of a compressed version of the architecture presented is described. Some of the main features of the system are described in this chapter, along with implementation details that were considered relevant. Work on two main modules is described, the Communications Module based on CAN (which was not found in the literature) and the TIM Services Module (also developed from the ground up). This development work tried, at its best effort,

to follow the architecture presented, to both facilitate comprehension of the codebase and allow for other applications to reuse these same developments. This platform intends to be released as an open-source solution, to contribute back to the scientific community, especially those who want to develop in the realm of IEEE1451 systems.

The results chapter presents examples and metrics obtained from utilizing the system. 3 use cases were defined and characterized to showcase the flexibility of the applications of the system. The TEDS field-filling process was also showcased, whilst serving also as a performance metering process. The handshake process was also analyzed, since it is the basis for the plug-and-play functionality. Lastly, an integration timing metric is defined and an estimation formula is set to help others compare its work with the one here presented. This expects to help in the scientific process behind IEEE1451 systems development. The formula was applied to the 3 use cases presented and compared with a test for one of the cases, achieving satisfying results.

At last, a large variety of concepts to be developed in future work, either by the author or other developers. This broad definition demonstrates the validity of the work and that the system is relevant to the point that it deserves further development.

Concluding, the work here presented achieved, to a large extent, the objectives initially set. A platform for developing smart transducers was devised, achieving interesting results. The flexibility the system demonstrated (which only partially implemented the original architecture), met the objective of developing a smart and flexible transducer system. The encapsulation, standardization of data, loose coupling, information flow, and plug-and-play features of this system make it relevant and applicable under the new Industry 4.0 techniques. It hopes to serve as a tool for introducing cutting-edge technologies into the shop floor.

8.1 Scientific Contributions

The work here presented is not an isolated work, as it fetches from the literature presented in chapter 2 and intends to produce some new scientific knowledge. Examples of that would be the implementation of the NCAP-to-TIM using a CAN-based network. To the best of the knowledge of the author, this is novel and can pave way for more authors to create IEEE1451.X interfaces between NCAPs and TIMs. An interesting aspect can also be the use of ISO-TP protocol layer, the handshake protocol using CAN, and the physical implementation of the CAN interface (using the TIVA C platform).

This dissertation work culminated on the release of a journal article [51] by the author et al.. This article reports on implementation specificities, the way the TIM was developed, the architecture behind the system presented, the TIM Cards concept, and the results obtained through this dissertation work was released, in order to add to the scientific knowledge community on the development process behind the implementation of IEEE1451.0 solutions.

The resources developed under this dissertation work can be found in the repository:

https://github.com/jmotp/STIM_1451 (accessed on 12 October 2022).

References

- [1] László Monostori. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 17:9–13, 2014.
- [2] Qianwen Wang, Shen Wang, Pan Zhang, al, Susan M Scott, David E McClelland, Teiko Heinosaari, Oskari Kerppo, and Leevi Leppäjärvi. Smart sensors in industry. *Journal of Physics E: Scientific Instruments*, 20:1087, 9 1987.
- [3] P M D Serra, A Espírito-Santo, F S Relvas, and J Bonifácio. Capacitive level smart sensors in the management of wastewater treatment processes. 2019.
- [4] Rita Pinto, Marcelo Moreira, Joel Bonifacio, and Antonio Espirito Santo. An iee 1451 vehicular sensor network development. *I2MTC 2020 - International Instrumentation and Measurement Technology Conference, Proceedings*, 5 2020.
- [5] Hua Fan, Jiangmin Mao, Quanyuan Feng, Xu Qi, Dagang Li, Lang Fen, Daqian Hu, Xi-aopeng Diao, Rami Ghannam, and Hadi Heidari. The design of intelligent sensor interface circuit based on 1451.2. *2019 IEEE 2nd British and Irish Conference on Optics and Photonics, BICOP 2019*, 12 2019.
- [6] IEEE Industrial Electronics Society, Industrial Electronics Society Standards Committee, IEEE Instrumentation, Measurement Society, Sensor Technology Committee, Signals, and Systems in Measurement Committee. 21451-001-2017 - iee recommended practice for signal treatment applied to smart transducers. 2017.
- [7] Avarachan Cherian, Darold Wobschall, and Mehrdad Sheikholeslami. An iot interface for industrial analog sensor with iee 21451 protocol. *SAS 2017 - 2017 IEEE Sensors Applications Symposium, Proceedings*, 4 2017.
- [8] Eugene Y. Song, Martin Burns, Abhinav Pandey, and Thomas Roth. Ieee 1451 smart sensor digital twin federation for iot/cps research. *SAS 2019 - 2019 IEEE Sensors Applications Symposium, Conference Proceedings*, 5 2019.
- [9] Ieee standard for a smart transducer interface for sensors and actuators : common functions, communication protocols, and transducer electronic data sheet (teds) formats. *IEEE Std 1451.0-2007*, 2007.
- [10] International Organization for Standardization. Iso 15765-2:2016 road vehicles — diagnostic communication over controller area network (docan) — part 2: Transport protocol and network layer services. 2016.
- [11] MAXIM Integrated. Max31855 datasheet. available at <https://datasheets.maximintegrated.com/en/ds/MAX31855.pdf> (accessed on 8 August 2022).

- [12] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.
- [13] Martin Hankel and Bosch Rexroth. The reference architectural model industrie 4.0 (rami 4.0). *ZVEI*, 2(2):4–9, 2015.
- [14] Kelvin T. Erickson. Programmable logic controllers. *IEEE Potentials*, 15:14–17, 2 1996.
- [15] Roger A. Breckenridge and Stephen J. Katzberg. Smart sensors for the 80's: The status of smart sensors. *AIAA Paper*, pages 41–46, 1980.
- [16] T. F. Tao, D. Hilmers, B. Evenor, and D. Bar Yehoshua. Focal plane processing techniques for background clutter suppression and target detection. *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, 0178:2–12, 8 1979.
- [17] Thomas F. Wiener. Problems with imaging sensor systems used for terminal homing. *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, 0178:124–126, 8 1979.
- [18] R. Breckenridge and C. Husson. Smart sensors in spacecraft - the impact and trends. 11 1978.
- [19] Gerry Smith and Mark Bowen. Considerations for the utilization of smart sensors. *Sensors and Actuators A: Physical*, 47:521–524, 3 1995.
- [20] Ieee standard for a smart transducer interface for sensors and actuators - network capable application processor information model. *IEEE Std 1451.1-1999*, 1999.
- [21] Ieee standard for a smart transducer interface for sensors and actuators - transducer to microprocessor communication protocols and transducer electronic data sheet (teds) formats. *IEEE Std 1451.2-1997*, 1997.
- [22] Ieee standard for a smart transducer interface for sensors and actuators : digital communication and transducer electronic data (teds) formats for distributed multidrop systems. *IEEE Std 1451.3-2004*, 2004.
- [23] Ieee standard for a smart transducer interface for sensors and actuators - mixed-mode communication protocols and transducer electronic data sheet (teds) formats. *IEEE Std 1451.4-2004*, 2004.
- [24] Ieee standard for a smart transducer interface for sensors and actuators : wireless communication protocols and transducer electronic data sheet (teds) formats. *IEEE Std 1451.5-2007*, 2007.
- [25] Ieee standard for smart transducer interface for sensors and actuators–transducers to radio frequency identification (rfid) systems communication protocols and transducer electronic data sheet formats. *IEEE Std 1451.7-2010.*, 2010.
- [26] Janusz Bryzek. Introduction to ieee-p1451, the emerging hardware-independent communication standard for smart transducers. *Sensors and Actuators A: Physical*, 62:711–723, 7 1997.
- [27] Byoung Hee Kim, Kwang Hyun Cho, and Kyoung Sup Park. Towards lonworks technology and its applications to automation. *Proceedings - KORUS 2000: 4th Korea-Russia International Symposium on Science and Technology*, 2:197–202, 2000.

- [28] S. B. Khoh and R. T. McLaughlin. Autos carry the can: Controller area network (can) for industrial applications. *Assembly Automation*, 14:17–19, 3 1994.
- [29] Dave van Gompel. Devicenet offers a new option for device networking. *Control engineering*, 42:63, 3 1995.
- [30] Joe Brown. Open networks link factory floor devices. *Power transmission design*, 36:19–23, 7 1994.
- [31] Douglas Norris. Honeywell gets smart. *Sensor Review*, 15:15–17, 6 1995.
- [32] Kang Lee. Proposed smart transducer interface standard. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 1:129–133, 1998.
- [33] Eugene Y. Song and Kang Lee. Understanding iee 1451-networked smart transducer interface standard - what is a smart transducer? *IEEE Instrumentation Measurement Magazine*, 11(2):11–17, 2008.
- [34] Oliver Niggemann, Gautam Biswas, John S Kinnebrew, Hamed Khorasgani, Sören Volgmann, and Andreas Bunte. Data-driven monitoring of cyber-physical systems leveraging on big data and the internet-of-things for diagnosis and control. In *DX@ Safeprocess*, pages 185–192, 2015.
- [35] Nicola Ulivieri, Cosimo Distante, Tondi Luca, Santina Rocchi, and Pietro Siciliano. Ieee1451.4: A way to standardize gas sensor. *Sensors and Actuators B: Chemical*, 114:141–151, 3 2006.
- [36] Hong Wang and Yue Zhou. Research and implementation on networked smart sensor based on iee1451 standard. *Open Electrical and Electronic Engineering Journal*, 8:492–500, 12 2014.
- [37] E. A. Batista, L. Gonda, A. C.R. Da Silva, S. R. Rossi, M. C. Pereira, A. A. De Carvalho, and C. E. Cugnasca. Hw/sw for an intelligent transducer network based on iee 1451 standard. *Computer Standards & Interfaces*, 34:1–13, 1 2012.
- [38] Jorge Higuera, Jose Polo, and Manel Gasulla. A zigbee wireless sensor network compliant with the iee1451 standard. *SAS 2009 - IEEE Sensors Applications Symposium Proceedings*, pages 309–313, 2009.
- [39] Anuj Kumar and Gerhard P. Hancke. An energy-efficient smart comfort sensing system based on the iee 1451 standard for green buildings. *IEEE Sensors Journal*, 14:4245–4252, 12 2014.
- [40] Yaoguang Wei, Xuelun Hu, and Dong An. Design of an intelligent ph sensor based on iee1451.2. *IFAC-PapersOnLine*, 51:191–198, 1 2018.
- [41] K Malar and N Kamaraj. Development of smart transducers with iee 1451.4 standard for industrial automation. *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014.
- [42] Victor Huang, Hiroaki Nishi, Antonio Espirito-Santo, Allen Chen, and Dietmar Bruckner. Standards and interoperability in industrial electronics – a trending view. pages 1–6, 10 2021.

- [43] Vítor Viegas, Octavian Postolache, and J. M. Dias Pereira. Transducer electronic data sheets: Anywhere, anytime, anyway. *Electronics (Switzerland)*, 8, 11 2019.
- [44] Yang Wei, Yucheng Liu, Kim Fung Tsang, and Hao Wang. Modeling of iee1451-standardized low power wide area networks. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–4, 2021.
- [45] Helbert Da Rocha, Antonio Espirito-Santo, and Reza Abrishambaf. Semantic interoperability in the industry 4.0 using the iee 1451 standard. *IECON Proceedings (Industrial Electronics Conference)*, 2020-October:5243–5248, 10 2020.
- [46] João Pinheiro, Diogo Oliveira, Luís Neto, and Gil Gonçalves. Introducing students to zero defects, condition monitoring and system integration using a refurbished learning factory. *12th Conference on Learning Factories (CLF), CLF 2022*, 4 2022.
- [47] Luís Neto @ DIGI2-FEUP. Iee1451.0 teds editor tool. 2022. available at https://github.com/DIGI2-FEUP/IEEE1451.0_TEDS_Editor (accessed on 8 August 2022).
- [48] Eliseu Pereira, Joao Reis, and Gil 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)*, pages 63–71, 2020.
- [49] Texas Instruments. Sys/bios (ti-rtos kernel) user’s guide (rev. v). 2020. available at <https://www.ti.com/lit/ug/spruex3v/spruex3v.pdf>(accessed on 8 August 2022).
- [50] Steve Corrigan. Controller area network physical layer requirements-texas instruments. 2008. available at <https://www.ti.com/lit/an/slla270/slla270.pdf> (accessed on 8 August 2022).
- [51] João Pinheiro, Diogo Oliveira, Luís Neto, Vítor H. Pinto, and Gil Gonçalves. Development of an iee 1451 plug-and-play module for smart transducers in industrial environments. *Sensors*, 22(20), 2022.