

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Robust Distributed Real-Time Processing Architecture for Man-Machine Cyber-Symbiosis

Luís Miguel Maia Marques Torres E Silva



Mestrado em Engenharia Informática e de Computação

Supervisor: João Paulo Trigueiros da Silva Cunha

October 19, 2023

Robust Distributed Real-Time Processing Architecture for Man-Machine Cyber-Symbiosis

Luís Miguel Maia Marques Torres E Silva

Mestrado em Engenharia Informática e de Computação

October 19, 2023

Abstract

Multiple cultural references both in literature and art exist depicting a world where humans and machines coexist alongside each other, communicating and interacting in real-time. These are often depicted as science-fiction but reveal a long-last dream of this new reality. This work proposes a comprehensive, and flexible reference architecture aiming to seamlessly integrate humans with machine intelligence.

This document starts by presenting a state-of-the-art analysis into Man-Machine Symbiosis, progressing into current technologies that can be leveraged to make this system viable. With this knowledge an architecture that serves as a foundational blueprint for the integration of this concept is presented and detailed. This architecture aims to provide a reliable and hardware-agnostic reference for the deployment of these systems without compromising neither the man's nor the machine's capabilities.

The focus of this work revolves around the smooth integration of human and machine intelligence, and the establishment of a dependable cyber-symbiotic system. To demonstrate a practical application where this interaction is valuable, a test scenario is defined that unveils the value of a true symbiosis. For this scenario, a simulation of a high stakes search and rescue mission in an adversarial scenario, an implementation of the reference architecture is developed. This implementation considers not only the constraints of the environment but also aims at not limiting neither the machines nor the human's actions while relying on consumer-grade hardware in detriment of proprietary or purpose-built alternatives.

The successful implementation of the reference architecture and its successful testing in real conditions not only validates it but also accomplishes the goals of this study. As technology continues to advance, this flexible reference architecture, independent both from software and hardware restrictions, sets an implementation pattern for future technological evolution.

Keywords: IoT, Distributed systems, Edge computing, Containers

Resumo

Múltiplas referências culturais, tanto na literatura como na arte, que imaginam um mundo onde seres humanos e máquinas coexistem lado a lado, comunicando e interagindo em tempo real. Frequentemente, essas representações são retratadas como ficção científica, mas revelam um sonho de longa data dessa nova realidade. Este trabalho propõe uma arquitetura de referência abrangente e flexível com o objetivo de integrar harmoniosamente seres humanos com a inteligência das máquinas.

Este documento começa pela apresentação de uma análise do estado da arte sobre Simbiose Homem-Máquina, avançando para as tecnologias atuais que podem ser aplicadas em tornar esta ideia viável. Com esse conhecimento, é apresentada e detalhada uma arquitetura que serve como um modelo fundamental para a integração desse conceito. Esta arquitetura tem como objetivo fornecer uma referência confiável e independente de hardware para a implantação desses sistemas, sem comprometer as capacidades do homem ou da máquina.

O foco deste trabalho gira em torno da integração da inteligência humana e das máquinas, bem como do estabelecimento de um sistema ciber-simbótico confiável. Para demonstrar uma aplicação prática em que essa interação é valiosa, é definido um cenário de teste que revela o valor de uma verdadeira simbiose. Para esse cenário, é desenvolvida uma simulação de uma missão de busca e resgate de alto risco em um ambiente adverso, com base na arquitetura de referência. Essa implementação leva em consideração não apenas as restrições do ambiente, mas também visa não limitar as ações das máquinas nem dos seres humanos, fazendo uso apenas de hardware comumente disponível, em detrimento de alternativas de hardware proprietárias ou específicas.

A implementação bem-sucedida da arquitetura de referência e do seu teste em condições reais não apenas a validam, mas também alcançam os objetivos deste estudo com sucesso. À medida que a tecnologia continua avançando, esta arquitetura de referência flexível, independente tanto em software quanto em hardware, estabelece um padrão de implementação para o desenvolvimento tecnológico futuro.

Keywords: IoT, Distributed systems, Edge computing, Real-time, Fault-tolerance

Acknowledgments

This document marks the end of a journey. In it is reflected my drive and passion but none of what I lived or the people that supported this journey are in it. Firstly, I would like to acknowledge the BRAIN group that welcomed me and trusted me on a very crucial point of such a revolutionary concept. I would like to thank professor João Paulo Cunha for his guidance and endorsement. I would like to give my appreciation to Vitor Minhoto that not only welcomed my ideas but also helped me mature them and supported me along the way, and without whose support these results would not have been achieved. There are numerous professors, more than I can mention, that impacted me and shaped my perception of the Academia. However, I would like to express my gratitude to Professor Bruno Lima for both his honesty and his playful nature. He was one of the first professors I came across in this last 5 years and one with whom I still keep in touch now in the end. To my friends, the ones outside the academia that led me here and the ones that walked alongside the last few years. They had different views and experiences than me, they wanted to strive in different ways, and I have learned from them and with them. A special acknowledgment to the ones that do not share the same field of studies as they always understood my hunger for knowledge and were willing to show and explain to me their work and never passed the opportunity to join me in any random interdisciplinary project I might come up with. From these a very special thank you to Margarida Santos for her relentless support and insightful points-of-view.

Finally, the most important ones, my family. To my parents that knew me better than I knew myself, that supported me in every circumstance even when they did not understand my reasoning. They were not only parents but my biggest fans and great goalkeepers that defended every curve-ball life thrown at them and at me. To my grandma that showed me the value of work, honesty and faith and to my sister that helps me see how beautiful and surprising life really is.

Luis Marques Torres

“Science investigates; religion interprets. Science gives man knowledge which is power; religion gives man wisdom which is control. Science deals mainly with facts; religion deals mainly with values. The two are not rivals. They are complementary.”

Dr. Martin Luther King

Contents

Abstract	i
Resumo	iii
Acknowledgments	v
1 Introduction	1
1.1 Motivation	2
1.2 Objectives	3
1.3 Contributions	3
1.4 Document Structure	4
2 Background and State of The Art	5
2.1 Distributed computing	5
2.2 Internet of Things	6
2.2.1 Edge computing	7
2.3 Communications for IoT devices	8
2.4 System orchestrator	9
2.5 Containers	10
2.6 Human-Machine symbiosis	10
2.7 Virtual reality with multisensorial feedback for training	11
3 A Conceptual Reference Architecture for a Man-Machine Symbiosis	13
3.1 Containerized services	13
3.2 Node layout	15
3.3 Storage	17
3.4 Monitoring	18
3.5 Cloud	18
3.6 Machine control	18
3.7 Information Flow	19
4 A Reference Architecture Implementation for a Search and Rescue Scenario	23
4.1 Previous Work	23
4.1.1 System restrictions and requirements	23
4.1.2 Containers	26
4.1.3 System Actors	26
4.2 ARTEX23 - ARmy Technological EXperimentation 2023	26
4.3 Equipment	27
4.4 Communications	28

4.5	The UAV	29
4.6	Sensors, the Vital Kit	31
4.7	Experimental Architecture	32
4.8	Execution flow	33
4.9	Results	33
5	Conclusion	41
5.1	Future work	42
5.1.1	Storage and database technologies	42
5.2	Security	43
	References	45
A	Man-Machine Symbiosis UAV Integration for Military Search and Rescue Operations	53

List of Figures

3.1	High-level architecture	14
3.2	Node groups layout	16
3.3	UML Interaction Overview Diagram of Machine control loop interaction	19
3.4	Communication Diagram for the Reference architecture	20
4.1	UML Diagram of Previous work: High Level Architecture based on the work of [1]	24
4.2	Nvidia Jetson TX1	27
4.3	UAV Stork II equipped with first-aid rescue pack	29
4.4	Vital Kit	31
4.5	Node-RED, high-level flow definition	34
4.6	Deployed container organization and communications running	34
4.7	Hardware used and communications	35
4.8	Dashboard showing the ECG signal from the soldier, fall state indicating if the soldier has fallen (red) or not (green), a map with the position from both the UAV and the soldier and three video feeds from the UAV.	38
4.9	The team and the UAV in the ARTEX experiment	39
4.10	Current equipment guided tour	40

Symbols and Abbreviations

ECG	Electrocardiography
EEG	Electroencephalography
GUI	Guest User Interface
HMCS	Human-Machine Cyber-Symbiosis
IoT	Internet of Things
INESC-TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
LUISMMS	Lightweight Unified Integration System for Man-Machine Symbiosis
MIPS	Million Instructions per Second
MQTT	Message Queuing Telemetry Transport
UAV	Unmanned Aerial Vehicles

Chapter 1

Introduction

In the relentless progression of technological advancement, the concept of human-machine symbiosis has emerged as a pivotal paradigm that promises to reshape the dynamics of human interaction with technology. Whilst the concept can be found referenced in papers as back as 1960 their practical applications are yet undocumented.

Unlike the traditional models of command and control, which often relegate machines to passive tools, symbiosis signifies a harmonious cooperation where machines become active collaborators, augmenting human capabilities while possessing their own autonomy. The roots of this revolutionary concept can be traced back to J.C.R. Licklider's visionary article in 1960 [2], where he foresaw the potential for an intimate partnership between humans and machines. Licklider's groundbreaking insights laid the foundation for the development of a symbiotic relationships between individuals and machines, characterized by shared goals, mutual understanding, and the seamless integration of cognitive abilities.

The idea of a symbiotic or cooperative relation, implies the existence of a communication method. This communication would have to be bidirectional between men and machines in order to cooperate and ease each other's tasks and goals. Nowadays, there are multiple ways in which men send commands to machines in order for them to execute men's intentions but in a scenario where machines can make decisions and also call for action by men, it is necessary to establish a communication strategy and a structure that enables this interaction. This bidirectional communication forms the bedrock of symbiosis, fostering a level of understanding that transcends mere programmed responses.

As technology advances, machines are increasingly endowed with the capability to process vast amounts of data, discern patterns, and extrapolate insights that extend beyond human ability. This hails a new era where machines not only understand human needs but also contribute to decision-making processes, offering diverse perspectives and expediting problem-solving. The advances in machine capabilities come with the need for faster data processing. The advent of Internet of Things (IoT) allows both the gathering and real-time communication of more data through a range of sensors but also the leveraging of this devices' processing power and capabilities. The leveraging of these capabilities can increase system functionality without the need for more expen-

sive hardware. Furthermore, the diverse nature of machines poses practical restrictions regarding latency, system failures and limitation of movements. All of this must be considered when developing a solution. This thesis delves into the practical utility of human-machine cyber-symbiosis (HMCS) with cybernetic systems, both on its feasibility and on its architectural implementation. In it a communication strategy is developed and a reference architecture for a system that provides such functionality is proposed and validated. As this work provides a functional structure for human-machine symbiotic programs to be implemented and understand how technological evolution might help evolve cyber-symbiotic relations unveiling unprecedented possibilities for collaborative progress and cognitive augmentation.

1.1 Motivation

There are numerous examples in science-fiction culture of human-machine symbiotic interactions. They are usually represented by the creation of a dialog between them. However, they also provide insights on how vast the application of this concept can be and the new possibilities and capabilities it opens up. The fact is that a real-time interaction with a machine is vital in the execution of some tasks, for instance a military operation, but for others it is not as crucial but can open a new range of possibilities such as in the industry where the collaboration between human workers and robots can be highly beneficial.

In fact, the exploration of this paradigm can influence society as a whole. It is easy to see the potential in the industry, especially when it comes to production processes and supply chains being an area where automation is so prevalent. But being technology so intricately attached to every aspect of everyday life, both at work, in education or socially, this technology as the potential to change the way we interact with the devices we own or use every day, making our interactions streamlined and more natural. Rather than replacing humans, it aims at creating a mutual dependency and beneficial relation with machines in an effective symbiosis.

Moreover, having machines able to make decisions without relying on an external operator avoids the induction of bias on the interpretation of the sensor data or of the surroundings. It also avoids the delay in human reaction and the delay in data/information propagation from the commander to the executor. This is even more relevant in the case of remote-controlled equipment as are Unmanned Aerial Vehicles (UAVs).

This thesis integrates a broader project of creating a cyber-symbiose. This project aims to have a UAV as an auxiliar in adversarial, high-paced and critical missions. The targeted scenario was a search and rescue mission. This high-paced situations can benefit immensely for a real-time interaction between men and machines. Not only can the man ask or send commands directly to the machine, as can the machine, not only make decisions based on its unbiased perception of the situation (making use of far greater computational power to analyse and make decisions than a human could possibly do) but also give suggestions and provide support in a way that a remote operator could never provide since it has only a limited perception of the situation. For all of these

to be possible, it is necessary to have a system that supports it. The developed reference architecture sets the clear path for implementation and evolution of symbiotic systems by providing a system-agnostic robust layer over which future software improvements can build upon reliably.

1.2 Objectives

This work aims to define a reference system architecture for HMCS as defined above. For that mission to be successful we set the following goals:

- The system must be adaptable to multiple scenarios:

In its definition, the system must take into account the different scenarios a machine might be used in and how to not interfere with its functionality.

- The data and rate of data produced must be taken into account.
- There can not be a hindering to the human or the machine's capabilities
- There cannot be strict hardware dependencies:

This work shall develop an hardware-agnostic reference architecture that is reproducible using different systems and hardware

- An implementation of the reference architecture must be tested both virtually and on a field trial

1.3 Contributions

This project delves into the feasibility and affordability of Licklider's proposed symbiosis by developing an infrastructure where it can be implemented and developed upon. In it we introduce a new reference system architecture for HMCS, develop an implementation of it and validates it with a real-world test in cooperation with the Portuguese Army, within the ARTEX (ARmy Technological EXperimentation), the first event of its kind organized by the Portuguese army. We also prove the premise that current consumer-grade technology can support such system. Moreover, we tackle problems present in the state of the art handling reliability in the communications without hindering the freedom of movements of the system actors and without introducing specialized hardware.

This project also studies its various components in terms of reliability and robustness presenting its findings, not only in already existing solutions but as well as of the developed approaches. This is a major contribution since many of the structure makes use of products are either open-source or off-the-shelf solutions that can be useful in other applications and therefore there is a clear pathway for open-source improvement, replication of the findings, discussion and reuse in other breakthrough projects.

This project and its field trial resulted in a paper on the development of a real-time interaction using general-purpose, consumer-grade hardware accepted in the ROBOT2023 conference [3] which is organized by Sociedad Española para la Investigación y Desarrollo en Robótica (SEIDROB) [4] and Sociedade Portuguesa de Robótica (SPR) [5]. Furthermore, a study on consumer-grade hardware capabilities to execute containers and other technologies useful for software development and deployment was also performed and its results presented in this document.

1.4 Document Structure

This document conveys 5 chapters. After this introduction, in chapter 2 an analysis on previous work and on the state of the art is done. The literature both on cyber-symbiotic systems is reviewed as well as technologies that can be used to tackle this challenge. Afterwards, in chapter 3 an analysis of the problem is performed. Its requirements are summarized, and specific goals are set for its success. A general reference architecture is presented as well as the choice of technologies. In this chapter not only are presented some of the technologies used but also explained and shown with experimental results the reasons behind some development decisions. In chapter 4 the experimental set-up is presented. In this section not only, it is presented the experimental scenario, as well as described the available equipment and the deployed architecture and infrastructure. In this section the results are also presented and discussed.

The last chapter discusses summarizes the success of the work and the goals achieved. Moreover, it also holds a future work pathway.

Chapter 2

Background and State of The Art

In order to propose a new system architecture is of the utmost importance to understand the works developed in this regard. It is also important to gather knowledge about the different areas and technologies that take part in the development of the system. This section presents concepts and technologies and displays an overview on the state of the art.

2.1 Distributed computing

The field of distributed computing and distributed systems has been a research topic for many years. Papers going back as far as 1979 can be found on the subject. A definition for distributed computing can be

"Distributed computing is the method of making multiple computers work together to solve a common problem. It makes a computer network appear as a powerful single computer that provides large-scale resources to deal with complex challenges" [6].

Indeed many advantages come from it namely the ability of scaling a system or service performance and availability without hindering the experience of whomever makes use of it, as the interaction with the system is similar to a non-distributed system interaction. A distributed system inserts complexity in the system design since data synchronization and consistency is necessary across nodes, however it also increases the reliability since a failure of a node does not condemn the failure of the system [6, 7].

Distributed computing can be divided in different types with different applications. *Cluster computing* connects machines in a same network in order to achieve higher performance, reliability and resilience. While traditional cluster computing relies on the machines being similar in hardware and software and being located in the same place, *grid computing* aims to tackle the same performance and scalability issues by joining machines that can have different specifications and also be very distant from each other. *Cloud computing* is an extension of this concepts by offloading the computing infrastructure to a provider that would virtualize or allocate resources as demanded, this is *utility computing*:

Utility computing is a service provisioning model that offers computing resources such as hardware, software, and network bandwidth to clients as and when they require them on an on-demand basis. The service provider charges only as per the consumption of the services, rather than a fixed charge or a flat rate [8].

Distributed systems can also be classified in terms of their architectural styles. They can be *Layered architectures* where different parts of the system are in layers, *Object-based architectures* where different objects are in different locations of which is example the Remote Procedure Call protocol is an example, *Resource-based architectures* of which REST is an example interface and *Event-based architectures* of which the *Publisher-Subscriber architecture* is an example. These are not static configurations, and a system might incorporate multiple architectures for distinct parts of it [9].

Decentralised systems can be loosely coupled, when changes in one element do not affect all the others, or tightly coupled. Tightly coupled systems are often used for high-performance systems like processing clusters, sharing memory or other resources, while all machines are assigned the same task, solving it faster (if the task can be parallelized) [10].

Microservices, that have gained popularity in the last few years is one type of distributed systems. The difference between microservices and distributed monolith systems is indeed the difference in coupling. In monolith services are tightly coupled while microservices intents to promote independence and autonomy between services [11].

2.2 Internet of Things

Internet of things, *IoT* is defined by ISO/IEC JTC1 as

“An infrastructure of interconnected objects, people, systems and information resources together with intelligent services to allow them to process the information on the physical and the virtual world and react” [12].

However, this term has been first employed in 1999 by Kevin Ashton when as he states:

"computers were brains without senses—they only knew what we told them" [13].

Indeed, computers were only machines that followed a set of instructions and did not have sensors or other methods to probe the environment in which they were to make decisions or act and therefore could only act on human commands.

However, nowadays any device such as a smartphone is equipped with a range of sensors that can increase its functionality and utility. There is also a range of smaller devices with lower computing capabilities which are an example *Arduino* and *Raspberry Pi*, that can gather data from sensors and communicate with each other or other devices and indeed react to changes either in physical or virtual environments. In fact, IoT projects a vision of a global infrastructure where not only anyone can access the Internet from everywhere, but anything can do it, enabling a connectivity between human-to-human, humans-to-things and things to things [14]. The IoT is particularly

interesting in the way it can leverage different technologies and integrate them into devices in order to produce a data-based outcome, gathering more data points for more accurate results. These can involve any kind of sensors, or communication technologies such as *RFID* or *Low Power, Wide Area (LPWA)*. Nowadays, the global IoT market is estimated to be 190 billion USD. [15] This growth in relevance is accompanied by the belief of many, such as Schahram Dustdar, that this is a new computational paradigm and that old methodologies are inappropriate and will soon be insufficient [16]. However, this author also points out that currently distributed computation in these systems based on client/server architectures has been meeting the requirements in spite of the inherent infrastructure complexity.

2.2.1 Edge computing

Numerous studies and efforts have been undertaken to explore the integration of IoT devices with already existing systems. These works usually do not follow a fully distributed architecture using IoT devices as a method to gather data and many times to do an initial processing, but few take advantage of these devices' capabilities afterwards. However, this has been changing since the nature of IoT applications being scalability and fault tolerance and these are concepts ingrained in the distributed computing area [17]. This need of bring the processing of the information closer to the nodes, has created a computational paradigm called *Edge Computing*, which aims to bring the cloud processing as close as possible to the physical source of data. This is, to avoid dealing with issues such as latency, all the processing that can be done locally is performed before transmitting the data [18].

Some frameworks running machine learning algorithms in IoT devices have proven to be successful, having these algorithms performed as expected [19]. Improvements in the optimization of task scheduling for these devices have also been studied. The fact is that many of these devices have limited power, processing or other resources available. In 2020 an optimization framework was proposed using a "*parallel and scalable neuromorphic computing system*"[20]. This framework is implemented by analysing not only the programming layer for which is an example application partitioning but also optimizing communications. This is not a unique approach as [21] presents a framework for optimization focused on smart cities, not only showing the versatility of usages of IoT equipment but also the need to adapt optimization techniques to their deployment scenarios.

Task partitioning is indeed one of the main concerns in edge computing, especially when dealing with IoT devices that do not have the capabilities to run in a time-sensitive manner monolith applications. Some frameworks have been proposed and in "Efficient Hosting of Robust IoT Applications on Edge Computing Platform" a proposal for a modulation of smaller tasks and a fully distributed approach based on other previous works related to *fog computing* [22] of which is an example "Kubernetes: Towards Deployment of Distributed IoT Applications in Fog Computing" [23]. This shows surprisingly superior results although it does not simulate situations of failure where one device on the network might suddenly become offline.

Such developments inspire the application of already existing frameworks for continuous development. Since Edge computing implies the decentralization of infrastructure, the application of containers is extremely interesting, being this also the subject of recent studies such as [18].

Also, regarding security, many works have been done taking focus on distributed networks of IoT devices. These are networks with many devices, processing data from many origins and communicating with many other devices and several times leveraging hybrid edge and cloud computing resources having, therefore, many attack surfaces [24]. Some works are based on the monitoring of the network [25] whilst others promote frameworks and future study guidelines [24].

Some issues still must be addressed, and Ioannis Psaras addresses them in [26]. There he discusses many issues such as the influence of widely used protocols such as DNS or even the governance and trustworthiness of a network node.

2.3 Communications for IoT devices

Consumer grade UAVs usually provide a long-range radiofrequency-based system. However, this kind of hardware is not common in other devices who usually make use of other communication technologies.

One wireless technology that is present in most consumer hardware is Wi-fi (wireless fidelity) that involves the IEEE 802.11 standards [27]. Moreover, TCP/IP (Transmission Control Protocol/Internet Protocol), a collection of standards for inter-device communication is also widely used.

IOT devices communicate making use of standardized communication protocols. The same happens in distributed systems for nodes to communicate with each other. *MQTT* or Message Queuing Telemetry Transport [28] is a protocol implementing the publisher-subscribe pattern especially well-suited for IoT devices since it was designed for unstable connections and bandwidth saving. Many benchmarks have been conducted on the different existing implementations of the protocol but few of them focus on fully distributed brokers. However, the study "A Comparison of MQTT Brokers for Distributed IoT Edge Computing" shows and compares already existing solutions, namely *EQMX*, *VerneMQ* and *HiveMQ*, showing that all these solutions are reliable and scalable [15]. Moreover, due to the limited wireless spectrum resources in some environments, some research has been made on the optimization of communications. One of these proposed solutions is an anti-interference optimization scheme for the IoT-edge algorithm to circumvent these issues [29]. Other protocols have been proposed for IOT communications, but according to [30] CoAP and MQTT have several advantages for resource-constrained devices since they require low computer power, they use small headers in messages and do not present significant delay in message delivery.

2.4 System orchestrator

A system with several parts or subsystems may require an orchestrator. The orchestrator manages the information flows and set-ups required. Orchestrators are very common in the context of distributed computing and cloud computing due to the existence of several different nodes and moving parts. System orchestration does not require a specific programming language selection or tool. Orchestration of a system can be visualized by flow charts. However, programming languages or descriptor files, of which the YAML format is an example, might not be so effortlessly comprehended.

Visual programming tools, on the other hand, might be easier to use and allowing the visualization of these flows:

- Node-RED

Node-RED is a low-code, flow-based programming tool. Moreover, it presents a GUI that enables the use of drag-and-drop actions to define flows. It is Open-Source and developed in Node.js giving it extreme extendibility and compatibility with other software and runs in a browser window giving it cross-device compatibility. Several works praise its usability and its relevance in the faster development and testing of IOT systems [31, 32].

- Zappier

Zappier is a commercial tool that is focused more on workflows rather than pipelines. While it can be used for these tasks, its free plan lacks the ability to execute multiple tasks with a single trigger. It is also an online tool and cannot, therefore, be applied in many scenarios [33].

- IFTT

IFTT is very similar to Zappier offering integration with different services but also having a web-first approach to automation. The fact that it cannot be easily extended for any use case and requires an internet connection makes it unsuitable for many scenarios [34, 35].

- Hugin

Hugin aims to replicate Zappier and IFTT functionalities while being open-source. Therefore it is easily extendable. It predates Node-RED but did not achieve the same level of popularity. Its main language is ruby [36, 37].

After this analysis, Node-RED was chosen since is the most complete of them. It provides the most functionalities and its Node.js base makes it easy to extend through JavaScript extensions. The existence of many community-built extensions helped solidifying this choice.

2.5 Containers

In the world of software development, containers have emerged as a revolutionary technology. They are a lightweight and efficient way to package, distribute, and run applications and their dependencies [38]. Containers have gained immense popularity due to their flexibility, portability, and ease of management. Contrary to virtual machines, containers do not virtualize a full operating system but rather they virtualize an environment and use kernel calls to the hosts operating system to execute tasks. This means that they are more lightweight and fast since they do not replicate a new kernel [39]. Their popularity in Software Development comes from the fact that by providing a virtualized and lightweight environment, there is a guarantee that the application developed will always run on that environment without the common problems of lacking dependencies or having different or incompatible environment set-ups. Moreover, their use in cloud-computing, serverless computing and microservices comes from the fact that they are easy to orchestrate and replicate since they follow a very well defined definition through a file [23].

2.6 Human-Machine symbiosis

Human-machine systems, human-machine symbiosis and Man-Computer symbiosis were terms first used by J. C. R. Licklider in an article with the same name in 1960 [2]. In it not only is described what would be the difference between a mechanically extended man, a command-and-control operation, and a symbiotic relation with a machine. The core concept lies with the use of Artificial intelligence so that a machine can autonomously run, and its input have as much relevance as the humans to benefit both:

"It is often said that programming for a computing machine forces one to think clearly, that it disciplines the thought process. If the user can think his problem through in advance, symbiotic association with a computing machine is not necessary" [2].

This is the base of computational thinking [40], however, and as the rise of application of artificial intelligence as proven, some problems are not easy or possible to formulate in advance or are faster to solve by trial-and-error approaches with computer guidance or may even be impossible to formulate without computer aid. This is, the machine not only helps to solve the problem but also takes part in figuring it out, therefore there is an intricate relation between man and machine throughout the entire process. In this article, Licklider also details technical pre-requisites for this to be possible from the capacity of information retrieval and shared processing time to memory organization requirements. We can easily look at today's technologies and find his premises possible. However, he also detailed the need to adapt language or create an intermediate layer that allowed man and machine to communicate and the needs for input and output equipment.

The subject of Cyber-Physical and Human Systems has been increasing in relevancy in the last few years [41] and the concept of a human-machine symbiosis has been attracting scientific attention [42]. While the input and output equipment and the creation of a common language is an obvious problem, there is a gap in the literature about this subject [43].

2.7 Virtual reality with multisensorial feedback for training

Several experiments exist in the introduction of multisensorial feedback in virtual reality experiences, especially for training. High-risk jobs such as firefighters are often times the case-study for this experiments due to the dangerous and demanding nature of their work. However, since this work is not focused on these kinds of systems directly, only makes use of a virtual simulator for testing purposes, the goal of this investigation was to gather insight on the kinds of sensor data and at what rates it was being probed from the human. Indeed [44, 45] describe some measures such as heart rate, sweat and ECG scans as sources of this information. Moreover, [46] details the data ratings at which it was possible to infer someone's identification from their ECG data, which opens vast possibilities for the development of this work.

The possibility of Human-Machine symbiosis is believed to be possible with current technology. However, there has not been defined a clear, hardware-agnostic and adaptable architecture that is adaptable and extendable in order to suit the most diverse kinds of machines while not hindering Human's or machine's actions.

Chapter 3

A Conceptual Reference Architecture for a Man-Machine Symbiosis

The following sections expand on the development of this architecture, as depicted in image 3.1 taking into account the requirements defined in 4.1.1. They not only put it in the context of the fully functional project but also dive on the different aspects such as communications and containers, contextualize the options taken and describe their architectural layout. This architecture implements an event-driven pattern and intends to be lightweight, unify and integrate different services in a standard system creating an architecture for **Man-Machine** symbiotic systems.

3.1 Containerized services

With the use of containerized environments we streamlined the development and deployment process. In this project containers helped in the following:

- Keeping the software-agnostic of the system where it is being run
- With its virtualization technology software can be developed agnostically of the system where it will be deployed.
- Software has to be written only once not depending on the device where it will run, making it compatible with any kind of node in the architecture
- Increased reproducibility and easier deployment due to the containerized environment
- Easier testing by running several containers that emulate different nodes in a single host, isolated from each other
- Easier set-up in the virtualized testing environment due to the possibility of setting the infrastructure up in a single machine and using automated scripting
- Better testing due to scripting automation, containerized environments and overall easier setting up process

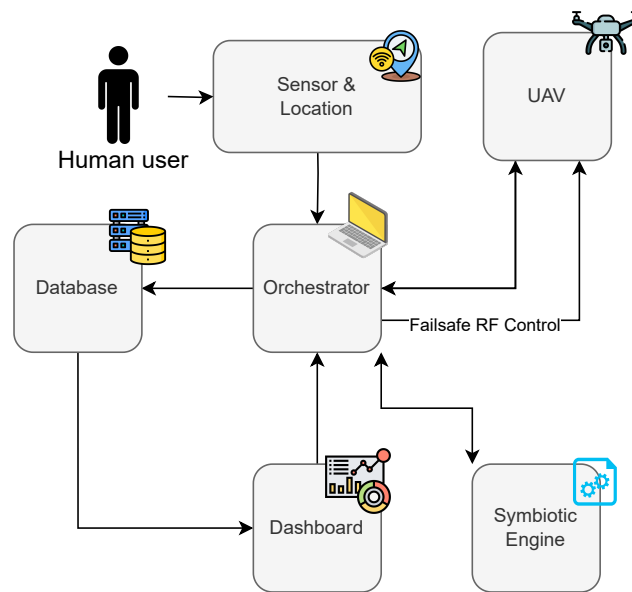


Figure 3.1: High-level architecture

- Easier development and integration with cloud technologies since they use container definitions as base for image deployment

Therefore, the system has 6 different container definitions:

- Machine

The machine is the system that interacts with the Human in a symbiotic manner. It is defined as a container since it is necessary to define an appropriate interface controller to communicate with it transforming the data sent to and received from it

- Message broker

The message broker will control the flow of messages. It will receive them and redirect them to the correct device. It can work as a central broker or it can be replaced by direct peer-to-peer communication.

- Orchestrator

The orchestrator is responsible for maintaining the data flow between each service. It defines and controls the execution flow coordinating the different services.

- Symbiotic Engine

The symbiotic engine is responsible for the machine's actions. It processes sensor data and outputs either commands for the machine to execute or information/requests for the Human, its the core of a machine capability to establish a symbiotic relation.

- Control panel

The control panel is fundamental to keep control of the system and monitor it. Dealing with autonomous machines requires a way to supervise them and debug their actions if needed and also analyse data in real-time. In case of autonomous vehicles its even more important since it allows monitoring them even when they may not be physically reachable.

- Database

The database not only stores the activity and system logs but might also be needed in some systems to fetch necessary information or to elaborate metrics upon which an action might be triggered.

While these containers may run in a single machine or in different machines they may also run in a distributed way across all nodes providing resiliency and availability to the system.

3.2 Node layout

Node groups are sets of nodes of the same type. We define a system comprised by 3 node groups. Figure 3.2, integrates Vitor Minhoto's previous work with the current proposal. Based on this, we can visualize, define and explain 3 node-groups:

- Machine

This node integrates the autonomous machine and its sensors. It receives data from the processing nodes.

- Human-attached devices

They send and receive data from the processing nodes.

- Other nodes for processing and controlling

These nodes include are where the Dashboard, database, processing algorithms and data translation is occurring.

Connecting layer

The connecting layer, or orchestrator, is where the data flow is defined. In this case a centralized connecting layer is defined into which every device connects. This orchestrator is a simple way of easily and rapidly define data flows and data transformations and does not have to be a centralized or independent unit, depending on the implementation. In our specific case we chose to display it as an independent unit since embedding it in other architecture structures could mean the loss of adaptability.

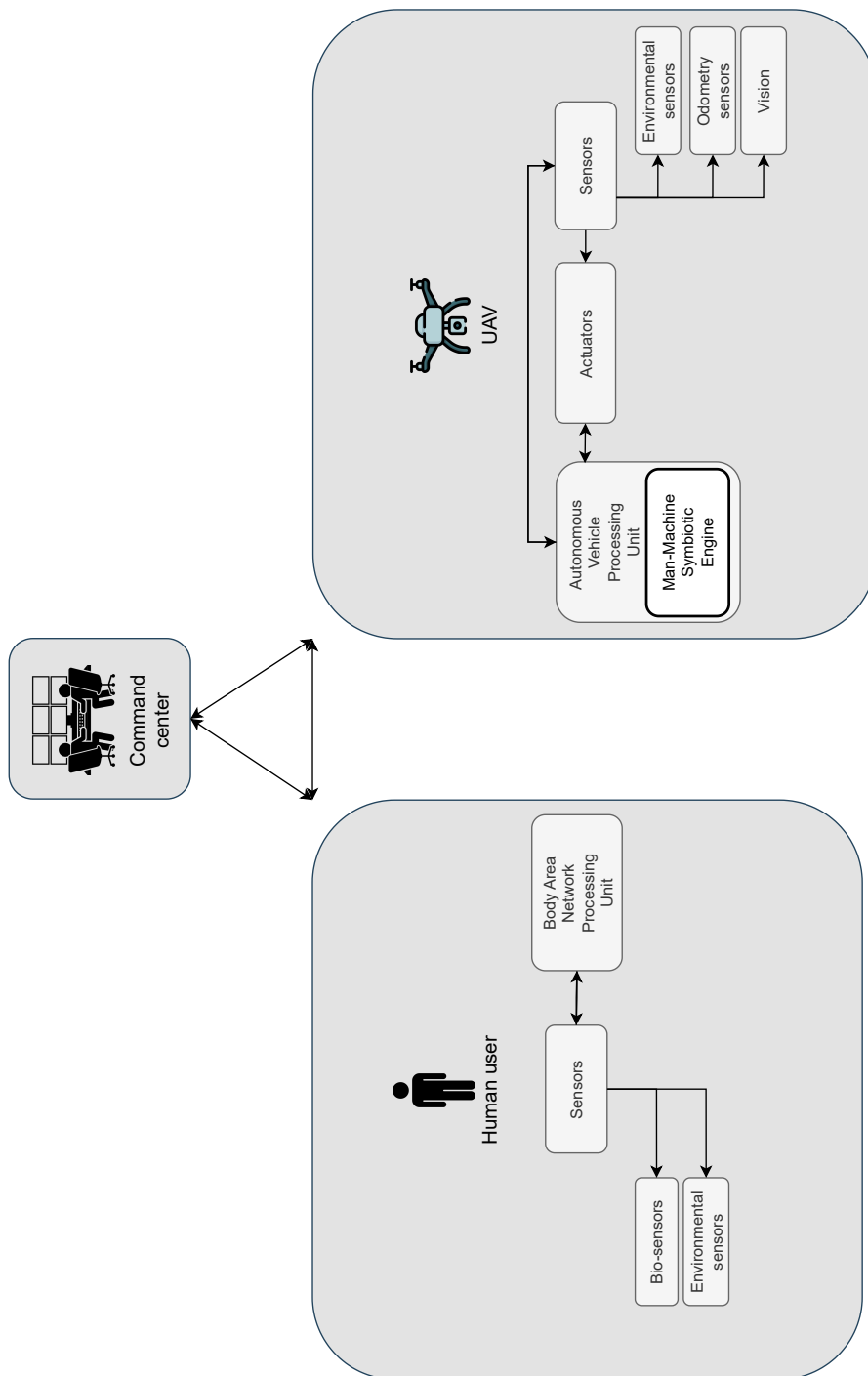


Figure 3.2: Node groups layout

Communications Layout

Beforehand it had already been proven that the communications should be wireless since every actor has free range of mobility. It had also been set up that communication latency had to be considered since we are dealing with real-time interactions. Moreover, the existence of several devices running different operating systems meant that a standard technology should be used. More than that, failures of communication and backup mechanisms had to be present, and the overall infrastructure had to be lightweight in terms of processing power.

Most UAVs already provide a radiofrequency-based communication interface. However, these kinds of interfaces are not common in other consumer hardware. Despite this, we found this method of communication interesting as a safe-guard, this is, as a kill-switch mechanism or takeover in case of failure of the system.

One thing that we were not able to complete was completely decentralize the network or implement a retransmission point, instead relying on a single long-range AP.

Mobile phones were also used to aggregate sensor data querying it via Bluetooth and pre-processing it before transmitting it over MQTT.

Therefore the network layout as seen before in 3.1 can be described as:

- The machine communicates with the control centre unidirectionally and with the orchestrator bidirectionally
 - Sends live video-stream to the control centre
 - Through the orchestrator
 - It sends sensor data and information to be sent to the Human
 - Receives instructions from the symbiotic engine
 - Receives commands from the Human
- The Human
 - Has a phone app that sends commands to the drone
 - Has sensors that send vital information to the phone
 - Has a phone that sends the sensors' data and its location to the orchestrator
- Orchestrator
 - Receives data from the drone and from the Human and transforms it
 - Sends it to the symbiotic engine and returns an action to be performed
 - Sends it to the database

3.3 Storage

The utility of data gathering is priceless specially when dealing with autonomous systems. Furthermore, the development of an in-lab testing simulator benefits immensely of having more data.

Due to the high influx of data from the human-attached sensors, drone sensors, video and devices a storage utility was necessary. One of the most important aspects of this data is to be able to reconstruct the action through time. Therefore, all data is timestamped. For this reason, it made sense to look into a time-series database. InfluxDB is time-series database that claimed to be scalable and have High Write and Query Performance [47, 48, 49] and amongst other advantages it is has an open-source version and diverse integration with other systems. Therefore, InfluxDB was the selected choice. Moreover, as we were proposing an implementation making use of influxDB, an article was made available relating the visualization of ECG data by combining the use of InfluxDB with Grafana [50]. However, that work lacked implementation details and information regarding the produced data and its rate of production and about the visualization plugin and its configurations used in Grafana. Therefore, they were not replicable, but the results presented were encouraging that this was a good choice of technology for our purpose.

3.4 Monitoring

For monitoring the choice relied on Grafana [51]. Grafana was developed for time-series events, and one might view this mission as a time-series event. Moreover, it provides a web interface that can be used cross-device and multiple plugins that extend its capabilities. It is simple and easy to use providing real-time feedback of the events and generating alerts [51].

3.5 Cloud

At this stage, the opportunity and the ability to connect to the cloud exists. The system architecture is scalable to support that and many of the services used have cloud native images. However, not all scenarios guarantee access to the internet and therefore access to cloud resources. Moreover, cloud services could be self-hosted but if they are hosted by third parties there is always the risk of that dependency becoming unusable. During the development of this project, the shutdown of an InfluxDB datacentre without users' prior knowledge [52] set the path we would follow for this first iteration.

3.6 Machine control

This architecture does not aim at replacing the existing functionality of the machine. Instead, it extends it and improves it. Therefore, the control of its actions is not centralized but rather built onto the machine itself. For this to happen, the machine implements a way of receiving commands for execution and inserts them into its execution. As shown in figure 3.3 , a message for an action to be performed is sent, the control loop that extends the machines capabilities receives it and manages its execution on the machine without the need of direct access to machine's hardware control by the message sender. The same does not apply to the kill-switch since the safe-guard mechanism must not be affected by control loop failures. In the case of a drone, a radio-frequency

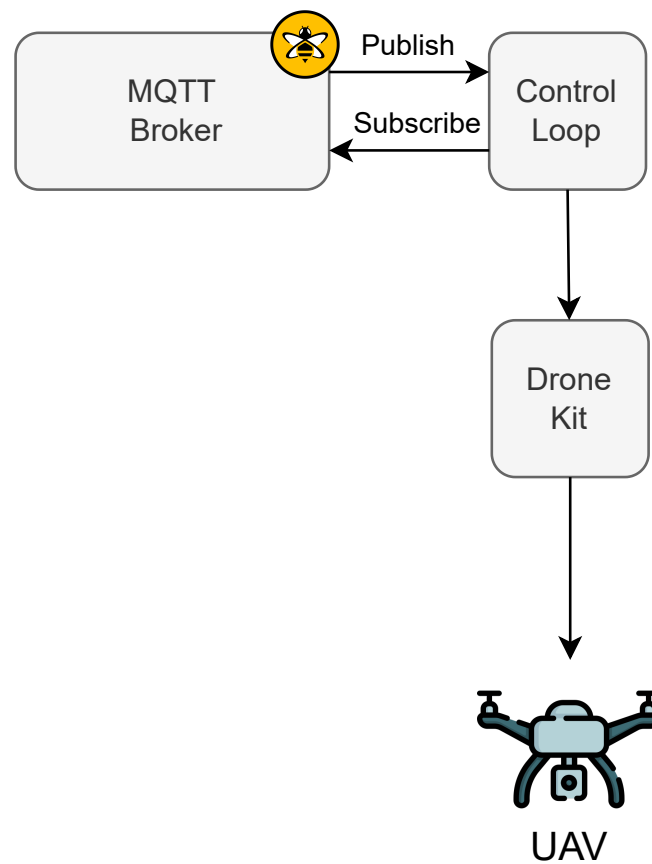


Figure 3.3: UML Interaction Overview Diagram of Machine control loop interaction

remote control circumvents the processing done by the control loop and communicates directly with the UAV core.

3.7 Information Flow

The picture 3.4 details the intricate connections between the different parts of this system. They can be described as the following:

- Data from the drone is transformed in the orchestrator
 - Is sent to the symbiotic engine for retrieval of an action
 - Is sent to the database for logs
- Data from the Human is transformed in the orchestrator
 - Is sent to the symbiotic engine for retrieval of an action

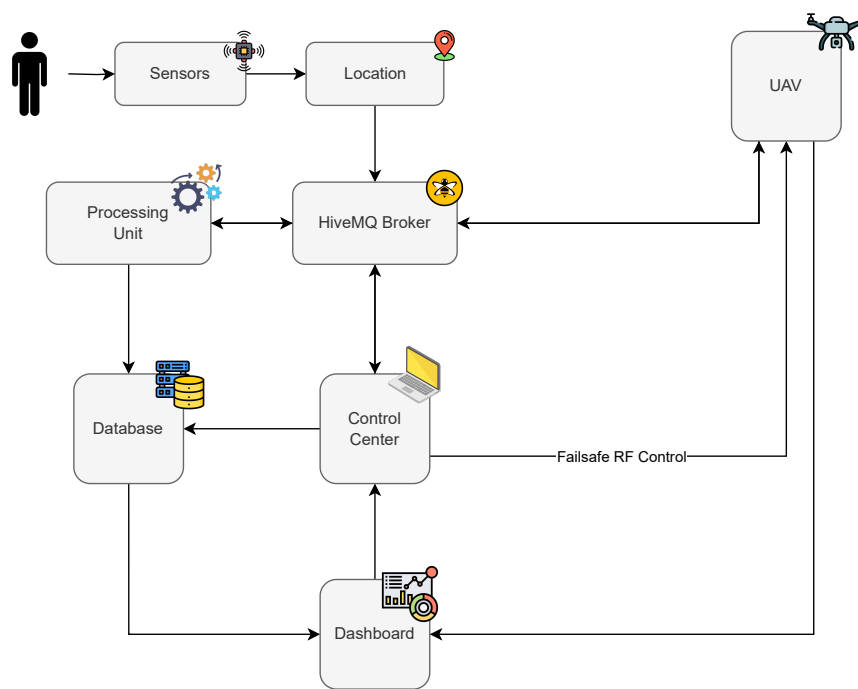


Figure 3.4: Communication Diagram for the Reference architecture

Is sent to the database for logs

Direct commands are relayed directly to the drone

- The control centre

Fetches live data from the drone

Fetches the most recent data from the database for graphical analysis of timeframes

Chapter 4

A Reference Architecture Implementation for a Search and Rescue Scenario

In this chapter its presented the previous work as well as it is described the implementation of the architecture and its deployment in a field-test. This includes the description of the hardware, details on the software deployment, test cases and it results.

4.1 Previous Work

The previous work encloses the overall view of a Human-Machine Symbiotic system previously developed. In it its detailed not only the overall Conceptual view of the system but also the existing work. Moreover, it will detail the system requirements and their impact on the development of a system architecture. This work is part of the research of BRAIN laboratory in INESC-TEC. Previously, Vitor Minhoto had developed a communication pipeline between an EEG wearable, a control unit and an unmanned aerial vehicle. The communication between the UAV and the human wearing the EEG device was made bidirectional by implementing sound generators and flight patterns on the UAV to transmit messages to the human. The high-level UML representation of the architecture can be seen in figure 4.1 A virtual environment for training and testing was also developed and a proof of concept was tested at the US Department of Defence (DoD) & Department of Navy “Joint Interagency Field Experimentation” in Camp Roberts, California [1].

Figure 4.1 shows a UML diagram [53] of the initially proposed design based on the work of Vitor Minhoto. The next sections will elaborate on it.

4.1.1 System restrictions and requirements

In his work Vitor Minhoto defined a very specific case-scenario of having a symbiotic relation between a UAV and a search and rescue officer. This particular scenario is important because it

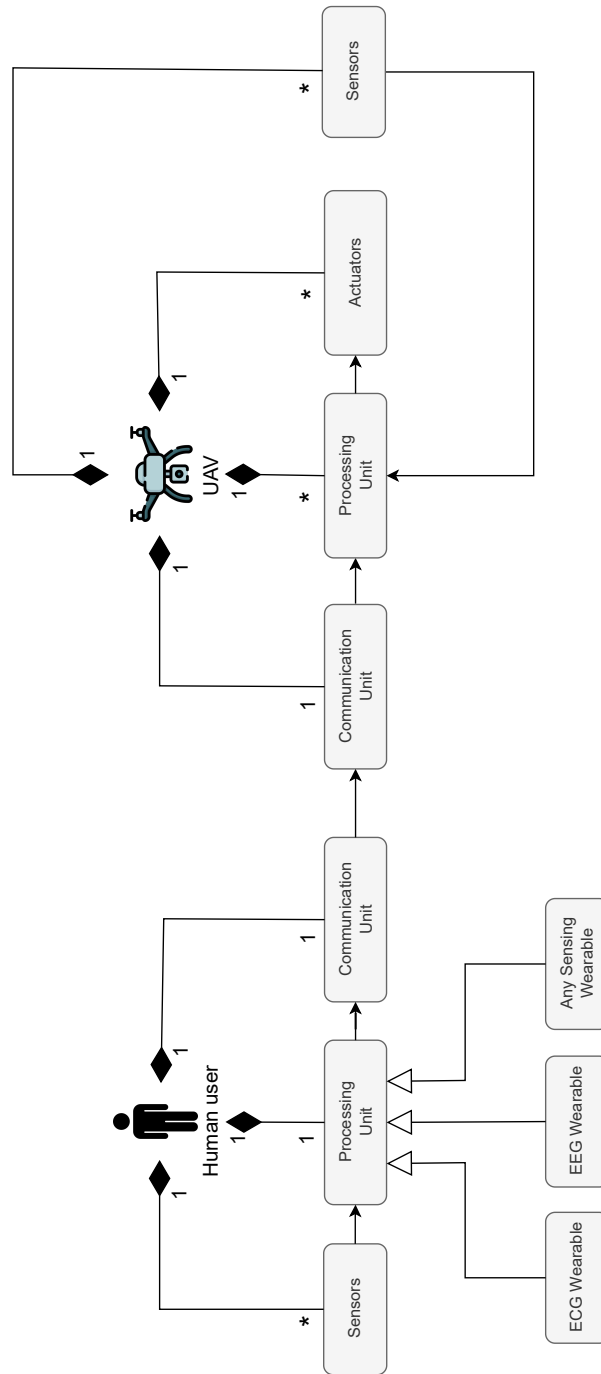


Figure 4.1: UML Diagram of Previous work: High Level Architecture based on the work of [1]

not only displays a real possible use-case of this technology but also introduces several constraints namely:

- The UAV is not-static and requires wireless communications in order to maintain its freedom of movements
- A rescue mission can both happen in a remote location where internet reception is unavailable or in a place where there is access to the fastest internet technologies
- A rescue mission might imply the need to travel long distances outside a control centre range
- A rescue mission might happen in multiple different weather conditions
- A UAV can operate at a fairly big distance from the human since it is flying while having to maintain communication
- In case communications are lost there should be a backup mechanism whether it is by using an alternative communication method, a kill-switch or a predefined-action.
- The communication between Human and Machine must be fast, reliable and not introduce additional complexity for neither of these actors in order to not hinder their mission
- The UAV has limited battery capacity hence it should not spend significant power processing messages, communications and non-flight related data
- The UAV can carry a limited size and weight payload, hence the system should not increase its carry-on hardware

Therefore, developed architecture must fulfil the following requirements:

- System-agnostic
 - Its implementation should be possible when using different devices both sensors or machines without the need for its redefinition
- Must not restrict human or machines movements
- Must be agnostic of the deployment environment
 - It must not make assumptions on the possibility to have external infrastructure or Internet connection, but be able to leverage it if available
- Must take into account possible limitations in hardware and leverage the set of devices in the infrastructure
- Must be extendable
- Must take into account the need for monitoring and logging data
- Must implement a safe-guard feature for machine shutdown in case of system failure.

- Must handle communication between man and machine
- Must take into account the need for low latency in communications with machines
- Must provide reliable communications
- Must take into account the type of data and rate of production and transmission according to the systems' possible needs in order to accommodate these restrictions

4.1.2 Containers

The organization of the system in nodes could imply the existence of multiple machines, that is, a distributed system. Moreover, containers streamline the development, since they provide an isolated and standardized environment where it is possible to deploy each node, being possible to have multiple nodes in the same machine. This is particularly interesting since it reduces our development hardware costs, enabling us to emulate a distributed work with fewer machines. The need to implement containers comes also by the need to develop software in other systems. Since containers provide a standardized environment the use of them allows us to develop a system agnostically of the operating system in which it is running. Finally, using containers gives us flexibility in the evolution of the system, since the use of cloud resources and many distributed technologies such as Kubernetes, leverage the concept of containers.

4.1.3 System Actors

The system actors can clearly be defined as the human and the machine. Both actors have different capabilities which they can leverage in a symbiotic relation. That implies an existing interaction that can be implicit or explicit. An implicit interaction comes from the analysis of data either from visual images or from sensors which generate an action. For instance, the significant change on the human's vital signs might prone the UAV to take action. An explicit interaction is the introduction of direct commands similarly to a command-control situation.

In that way, previous work as introduced the need for the Human to carry a processing module that can aggregate and process any number of time-series data acquired by the sensing units (wearable devices) for implicit communication and introduced set actions for the UAV to signal the need for human action. The introduction of explicit actions might be done through a wearable electroencephalography system or other methods.

4.2 ARTEX23 - ARmy Technological EXperimentation 2023

ARmy Technological EXperimentation 2023 or ARTEX23 was an initiative promoted by the Portuguese army, more specifically by CEMTEX-Centre for experimentation and technological modernization of the army that allowed investigation centres and companies to run field-experiments in a military base, the military camp of Santa Margarida in Constancia, Portugal. It was the first of its kind promoted by the Portuguese army. In there it was possible to emulate the test scenario

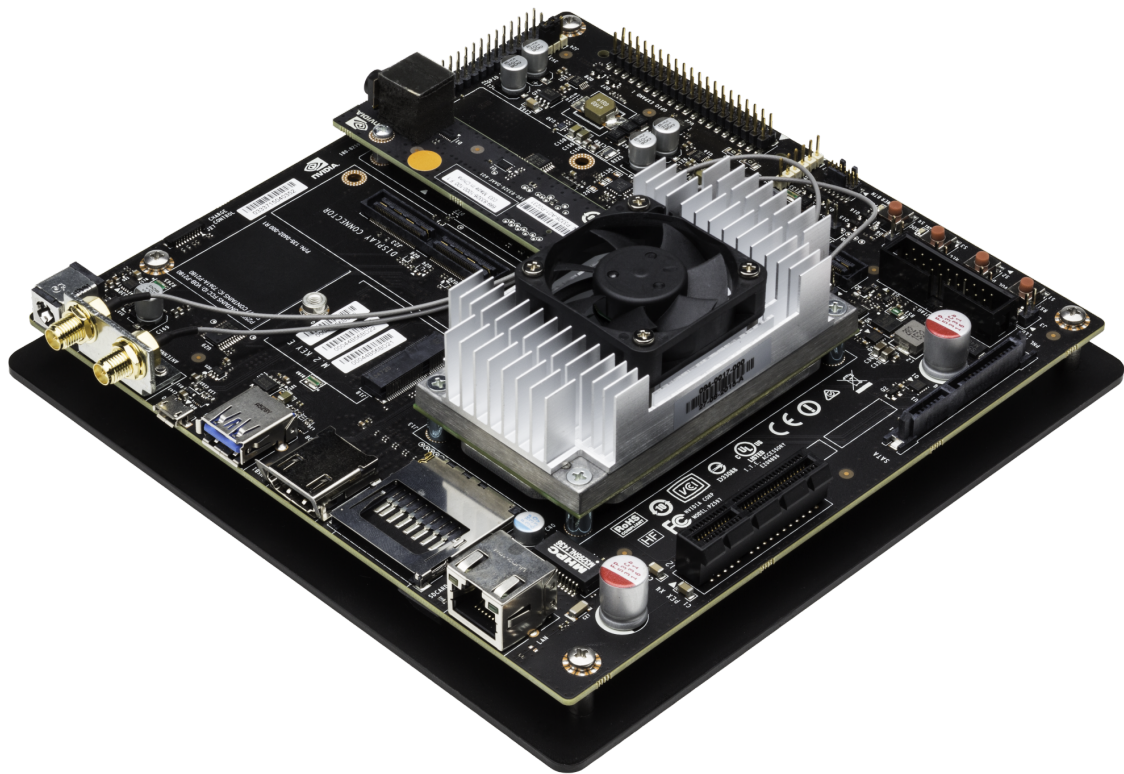


Figure 4.2: Nvidia Jetson TX1

of a rescue mission since there was an unrestricted airfield, difficult terrain and a widespread area. Our team, from INESC-TEC, gathered elements from the Centre for Robotics and Autonomous Systems (CRAS) that lent us the UAV and us from Centre for Biomedical Engineering Research (C-BER) as we can see in figure 4.9. In this experiment it was also possible to see the current equipment used by the army 4.10 and devise improvements and future work. A video detailing our presence in the event can be seen in <https://tinyurl.com/MMCS1>.

4.3 Equipment

The presented solution should be able to run on minimal hardware. So far in previous work the hardware used in field trials was:

- Android device (Moto G31 (w) (XT2173-3)) wit 4+64 GB, not rooted, software version RRWDS31.Q3-26-83-3

Node of the human user, aggregates data from vital sensors and transmits it to the control centre

- Nvidia Jetson TX1 - figure4.2

Used for image offloading and processing, intended to run decision models directly from UAV's data

- ASD+ GoPro Camera

Attached to the human user for recording of the experiment

- VSM Vital Patch and VSM Vital Jacket - figure 4.4

Sensors that gather human user's vital data and transmit it to the Android device

In the previous the Nvidia Jetson TX1 4.2 is an embedded compute unit. It is based on the ARM CPU architecture which makes it substantially different from other devices in the network in the sense that many applications may not be able to run on it without being recompiled. Its use was for expedite the computation of drones actions by offloading image processing tasks into it. in spite that for this experiment this functionality is not required for the validation of our architectural results, it will be used for processing the images from the 3 cameras present in the UAV and run a HiveMQ node in order to prove that the reference architecture is flexible in handling different hardware.

For this experiment a laptop for the containers to run was also used, with the following specifications:

- HP Zbook 17 G

Intel® Core™ i7-8850H 2.60Ghz 6 cores

32Gb Ram

KXG50ZNV512G TOSHIBA SSD 500Gb

NVIDIA Quadro P5200

4.4 Communications

For connectivity a directional Wi-Fi access point was used.

- Interline IP-G23-F5258-HV

Gain: 23dBi

Impedance: 50 ohm

Frequency: 5000 - 5850 MHz (IEEE802.11a/n/ac)

In terms of messaging protocol, as we have discussed in chapter 2 two best suiting proposals arised, MQTT and CoAP. The decision to use MQTT over CoAP (defined by RFC7252)[54] came from the fact that CoAP does not know the connection status to each device, does not have the ability to retain messages, is synchronous, as a bigger message overhead and does not support a publisher/subscriber architecture [55]. Therefore, MQTT seemed a more suitable choice.



Figure 4.3: UAV Stork II equipped with first-aid rescue pack

4.5 The UAV

In order to execute this test a UAV was required. The Robotics and Autonomous Systems Center (CRAS) from INESC-TEC provided a machine for us to use, the STORK II 4.3. The STORK II is a multipurpose vehicle that had previously been used across diverse scenarios, including search and rescue missions, electrical asset inspections, and the meticulous reconstruction of historical landmarks [56, 57].

The UAV's characteristics can be seen in depth in table 4.1 but it is worth mentioning that its operating system is ROS (Robotic Operating System) and the Mavros package to achieve communication between the ROS ambient and the autopilot firmware.

Robotic Operating System

The Robotic Operating System (ROS) is an open source "set of software libraries and tools that help you build robot applications" [58]. ROS is a distributed system [59] which is particularly important since if the nodes are on the same network, they can communicate with each other and execute different tasks. In this way it is possible to communicate with the system running in the UAV by simply having a local node running. This allowed to perform actions such as creating live video streams and implementing a Webserver for MQTT communications easier and without adding extra processing load on the node running on the UAV, since these were only done in other nodes in the network. This also allowed to monitor internal logs in real time and extract further

Table 4.1: UAV features description

Features	STORK II
Size open (DxH)	1060 x 570 mm
Size close (DxH)	670 x 570 mm
Motors	6 x DJI E2000 Pro
Weight (with batteries)	9.3 kg (11 kg)
Batteries	22.2 V - 22000 mAh
Flight time	20 - 25 min
Navigation	
GNSS	RTK GPS simpleRTK3B Heading, dual antenna
Autopilot	INESC TEC autopilot
IMU	STIM300
Perception	
Cameras	Teledyne Dalsa Genie Nano GiE
IR Camera	Teledyne Dalsa Calibir DXM640
LiDAR	Velodyne VLP-16
CPU	i7-6822EQ with 16 GB of RAM DDR4

metrics also in real-time. In the global context of this project, this means that as long as the connecting layer can work in a distributed way, routing and mapping communications becomes easier because there is no need to manually develop a fallback mechanism nor leadership electing algorithms when a node fails since this logic is already handled by both HiveMQ and ROS.

Communication with ROS

ROS natively implements publish/subscribe structures for internal communication. However, these are standardized format messages that require transformations for interpretation by external services and this message queues are not accessible outside of the system environment. After having a local node, it would be possible to query and publish messages using the command-line, which would also be possible through Node-RED. The fact that these processes run in different containers could pose some issues, but the main concern was that it would create yet another layer of complexity if these two parts of the system were running on different nodes. Moreover, this would not guarantee a system-agnostic deployment. Therefore, installing the "rosbridge-suite" that suited our version we were able to set-up a webserver with a JSON API that in turn allowed us to have an endpoint to which we could create and topics as well as publish messages [60]. Although in this streams there were also somewhere sensor data was broadcasted, since we were running a local node and would not add increased load to the UAV we opted to install the extension "web-video-server" in order to have a web-video server that transmit live video streams over HTTP from the UAV's cameras [61]. This approach could be pointless if we wanted to save the video Stream data, case in which subscribing to the video stream messages would be more efficient, but these were only relevant for monitoring and provided no real use for simulations. Moreover, the UAV stored this data itself with more quality (the transmitted data was scaled down for bandwidth efficiency) which we could retrieve afterwards.



Figure 4.4: Vital Kit

On communicating with the UAV its sensor data topics were subscribed and two topics were created: one to publish the action to be performed by the UAV and another to publish the human location every second. A set of 3 actions were pre-defined for test purposes: returning to base, aerial recognition, travelling to the human current coordinates and deploying payload. By using this MQTT approach we could support the system logic and decision making while simply developing an interface to accommodate the Robotic Operating System.

4.6 Sensors, the Vital Kit

Each human in the experiment wore a vital kit, which is composed by a vital patch [62, 63] and an AmbiUnit [64] sensing box. These devices, in figure 4.4, gather vital data which is then sent through Bluetooth to the android device.

The vital patch was attached to the user's chest using two electrodes or a wearable t-shirt [62] and collects the user's medical grade ECG data at 500Hz, body temperature and movement with an IMU at 10HZ [63].

The AmbiUnit sensing box measures the surrounding environment of the soldier. This sensor was positioned at the belt and could collect the surrounding environment's atmospheric pressure, CO and NO₂ concentrations, luminosity, and temperature.

A GoPro camera was used to have a first-person view of the military during the scenario.

The android receives the data by Bluetooth, uses an algorithm to determine if the person is in distress based on the data and if the person has fallen, adds both this information, the data,

the identification of the person wearing the sensors and the current geographical location and publishes this information on the MQTT cluster.

Figure 4.7 depicts the hardware used and the interactions between each component.

4.7 Experimental Architecture

The implemented system can be seen in 4.6. Each part of the system ran on a virtualized container although they all ran in the same machine:

- Orchestrator, processing unit and symbiotic engine

Both the orchestrator the processing unit and a mock symbiotic engine were implemented in Node-RED 4.5.

Node-RED transformed the receiving data, filtered it and dispatched it to the correct endpoints.

- Database

The database used was InfluxDB due to it being a time-series database. It received the data from Node-RED after it being processed.

- Dashboard and control Center

The dashboard was implemented in Grafana due to its ease of use.

It collected a live video feed from the UAV and a 5 second timeframe of data from the database filtering it in order to display relevant graphics such as location variation and ECG frequency.

- UAV

The UAV had ROS as operating system.

A second ROS container ran in order to facilitate the exchange of data and commands with the UAV and execute battery intensive tasks without hindering the UAV's autonomy.

- MQTT broker

The android devices were using a generic MQTT broker.

All other devices were executing HiveMQ, a distributed MQTT broker, making it resilient to failures.

- Human-attached devices

Data from the VitalKit sensors was sent to the android device.

The android device used a custom made application to:

Gather sensor data.

- Filter and process data.
- Add location information.
- Transmit any direct order from the soldier.
- Send an alert in case the soldier falls down.

Besides the UAV and the android devices that communicated with the sensors, all services were executing in containers in the laptop. This, however, was not a problem and due to the container technology proved how efficient this structure can be, running without issues in one computer, and at the same time emulated the existence of multiple nodes on the network.

4.8 Execution flow

As detailed before, Node-RED was used for orchestration, processing unit and as the symbiotic engine mock. In figure 4.5 we can see the high-level flows. These high-level flows show the overall execution path and are implemented with smaller actions, the detailed flow.

When a message is received in the MQTT topic `"/militar_test"` it is known that these are messages containing data from the Human-attached sensors. Therefore this message is processed and is extracted his geographic coordinates and the data from vital sensors is processed (emulating the symbiotic engine) to determine if there is an emergency situation. The geographic coordinates are then published in MQTT topic `"/militar_state"` from where they get pulled by another processing unit, transformed (change in format) and published into InfluxDB (the database) and into ROS Topics in order for the UAV to have access to them. The result from the symbiotic engine is sent to the MQTT topic `"/largar_saco"` and after being transformed to ROS actions published into a ROS Topic for the UAV to execute these computed actions. The MQTT topic `"/instructions_state"` refers to the predefined actions that were set in a mobile phone application and represent direct instructions that the human can give to the UAV, upon the appearance of a new message this is transformed and published to a Ros Topic.

Other flows not represented are the ingestion of sensor data to InfluxDB in batches.

4.9 Results

The experiment proved to be extremely useful to the development of the project. Firstly, we tested the system in real-world and proved that this architecture works as intended. During the trial, some tweaks had to be done to the system and they were easily and rapidly implemented proving the usability of the system and the possibility for it to be extended even for people whose background is not related to information and technologies. The testing conditions were also truly diverse. We had the opportunity of facing several different weathers during the time spent in the military base which were a test to all of the equipment and its adaptability to different conditions. Due to the extended testing area, we could also test the system up to 800 meters of distance. Not only was the performance expected, but we were also able to conduct more than 2 hours of testing without

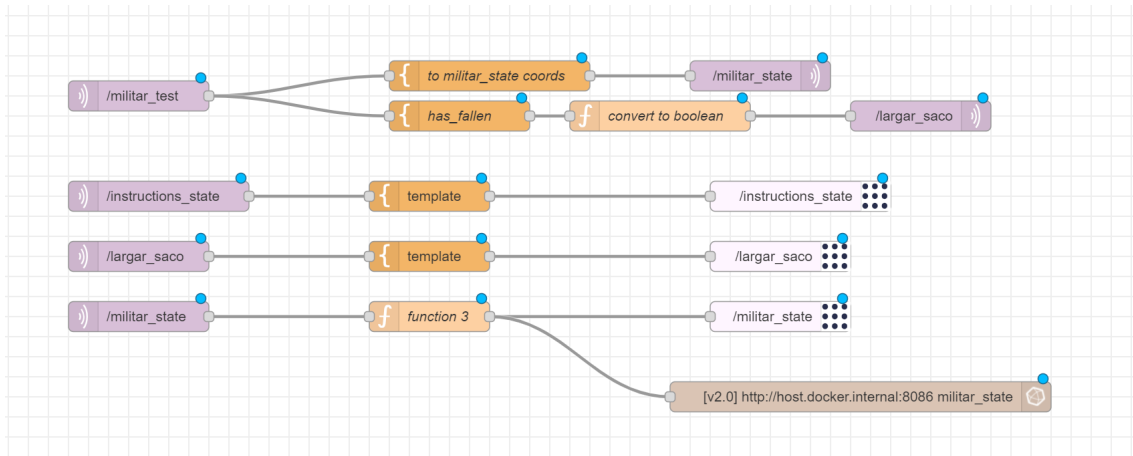


Figure 4.5: Node-RED, high-level flow definition

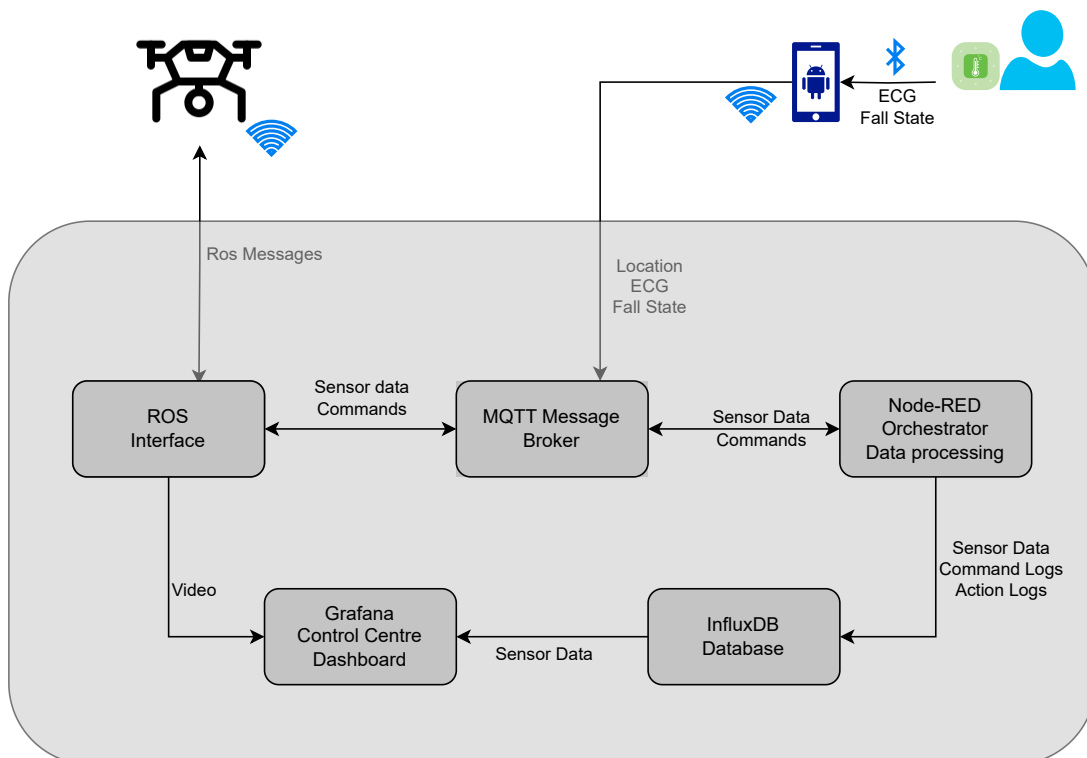


Figure 4.6: Deployed container organization and communications running

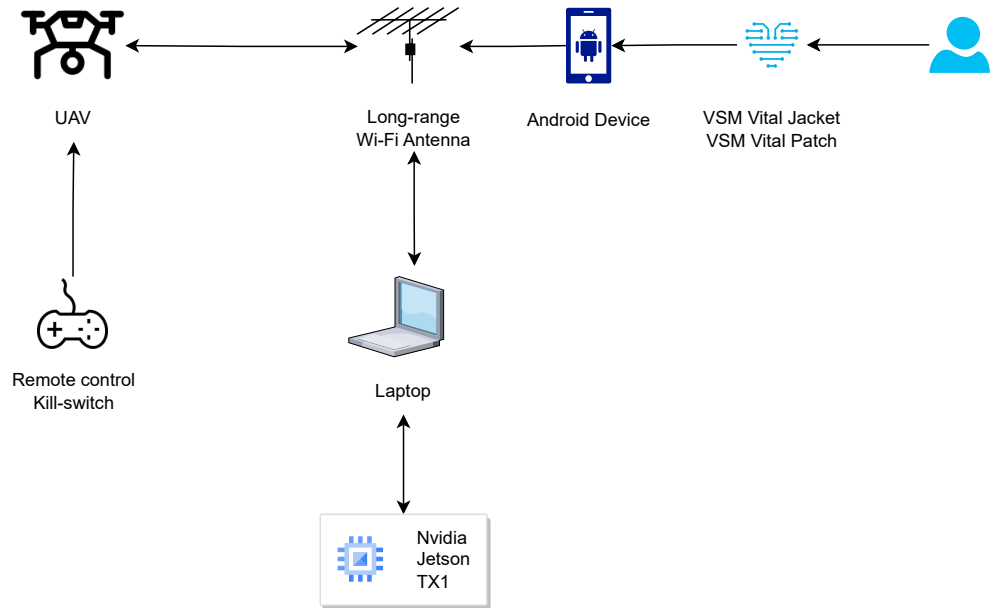


Figure 4.7: Hardware used and communications

requiring external power which was a very good and unexpected result. At the end of the trial not only we gathered truly relevant data that will help in evolving the virtual simulator as we also gathered great interest from both the high-ranking officers present as well as some special-tasks elements present.

On the other hand, this deployment also presented some challenges. During the experiment difficulty in the communication between containers was noticed. Despite having adjusted their configurations so that they would be in working condition this unveiled that improvements must be undertaken so that the deployment is hassle-free. The fact that we had a long-range Wi-fi access point helped to abstract the architecture logic from infrastructure, however this is one point that must be improved and considered even with the implementation of alternative communication technologies that can work as backup. Finally, the database performance degraded substantially through time. While with up to 500 records we would have query times of under 40 milliseconds, at the end of 5 minutes of experiments the database would be taking more than 1 second to satisfy our queries. It was easy to pinpoint that the increase in data stored was the cause of this performance drop. However, it stayed unclear if it was due to some internal operation on rearranging or compressing the data or some other issue. The use of Flux as a query language should help to iron out deficiencies in the query building process however it must be studied. One option we could have tried the running of multiple InfluxDB clusters for a horizontal scaling but due to the great amount of sensor data this database was built to handle, our volume of data should not have caused this kind of behaviour. It should be studied as well if the rate at which we are querying the database to display metrics in Grafana is to be blamed for such behaviour.

		Unmodified Kernel	Modified Kernel	Average Difference
GeekBench CPU	single-core	401	405	0,5%
	multi-core	1325	1343	0,7%
GeekBench GPU	openCL	1108	1109	0,5%
	Vulkan	1080	1053	1,3%
7-Zip	Compression	7020 MIPS	7041 MIPS	0,1%
	Decompression	12077 MIPS	12846 MIPS	3,1%
			Total Average Difference	1%

Table 4.2: Impact of kernel modification on performance

Containers in Android

Running containers in Android was particularly interesting in order to expedite development, since we make use of these devices in our system. Literature on the matter of running containers in Android devices is scarce. In order to evaluate this possibility we used an android device with the following specifications.

- Android device (Moto G31 (w) (XT2173-3)) with 4+64 GB, not rooted, software version RRWDS31.Q3-26-83-3

Many online sources such as [65] and [66] claim this to be impossible due to a lack of kernel features like cgroups and overlays. This issue is not recent and there is even a request for it to be solved since 2016[66]. However, the android system in recent versions can virtualize systems, therefore other sources like [67] or [68] do not seem far-fetched in their claims of success of executing the system.

We followed 2 different approaches. One simply ran virtualized a Linux system while the other one executed changes in the kernel to activate the missing required features and attempted to execute a container.

In order to benchmark the system, we ran Geekbench [69] and 7-zip [70] benchmarks on the device before and after the kernel modifications and also both on the virtualized linux and on a container. These tests were performed 5 times, with the device plugged in, fully charged, in airplane mode after sitting still for 30 minutes at room temperature for each set-up and the results shown compile the average results.

The GeekBench tests do not have a unit of measure however "they are calibrated against a baseline score of 2500 (which is the score of an Intel Core i7-12700). Higher scores are better, with double the score indicating double the performance"[71]. 7-zip benchmarks are measured in Million instructions per second (MIPS).

Table 4.2 show that the modification of the kernel did not account for any significative performance change.

Unfortunately, the execution of containers natively, by using a modified kernel was not successful and due to time constraints, it was chosen not to further investigate it. The execution of

		Device performance	Alpine Term	CHeck vShel
7-Zip	Compression	7030 MIPS	211 MIPS	200 MIPS
	Decompression	12461 MIPS	430 MIPS	411 MIPS

Table 4.3: Virtualized Solutions Benchmark

docker in a virtualized environment was surprising in the fact that some characteristics were natively available. Indeed, it was possible to map the device's filesystem and to keep running the virtualized environment in the background or even with the screen locked. However, the device's hardware was not accessible easily and if another app accessed the same hardware required by a container program, the container would lose its access for the rest of its execution. Moreover, its startup times were noticeably slow, taking up to 3 minutes. We assessed 3 virtualization solutions. All of those required downloading software from non-official sources. Moreover, some of the solutions do not seem to be still supported. The benchmark results on the virtualized environments can be seen in table 4.3, however cannot be recommended at this stage of their development. The now flagship feature of some phones that enable to virtualize environments natively might be a suitable alternative to be evaluated in the future when it is widespread.

The results showed that the modification of the kernel to get root permissions had done no harm when it came to performance. However, the container execution results, even for simple tasks were far from acceptable. In addition to that, the incapability of easily map the phone hardware to containers' virtual interfaces and the unsafe nature of their sources determined that these devices would only be running software specially developed for android and not containerized applications.

HiveMQ

In order to confirm that HiveMQ [72] was the correct choice for us we ran resiliency tests. In fact, the open-source version of HiveMQ was not able to complete any of these tests, however the professional version which was free up until 25 devices met our requirements. In order to confirm its usability, firstly we synced every device's clock. Afterwards, set-up three devices in the same HiveMQ cluster. One of the devices would send messages every second to the cluster with the timestamp when the message was created. The other devices received the message and registered the timestamp when it was received. The messages were set with a quality of service (QoS) 1, ensuring the delivery at least once. Afterwards one of the devices would be shut down and the stability of the cluster would be checked. The device would then be joined again to the cluster and the stability of the cluster tested again. The results showed that the cluster remained stable when a device left it and rejoined. The messages had an average delivery time of 40 milliseconds which would increase when there were cluster modifications up to 125 milliseconds but then returning to the average values. We also observed that all of the messages were received by the node that left the cluster upon rejoining and we confirmed it afterwards by inspecting the log of messages sent and received, which was possible since the timestamp of the creation of the message is a uniquely

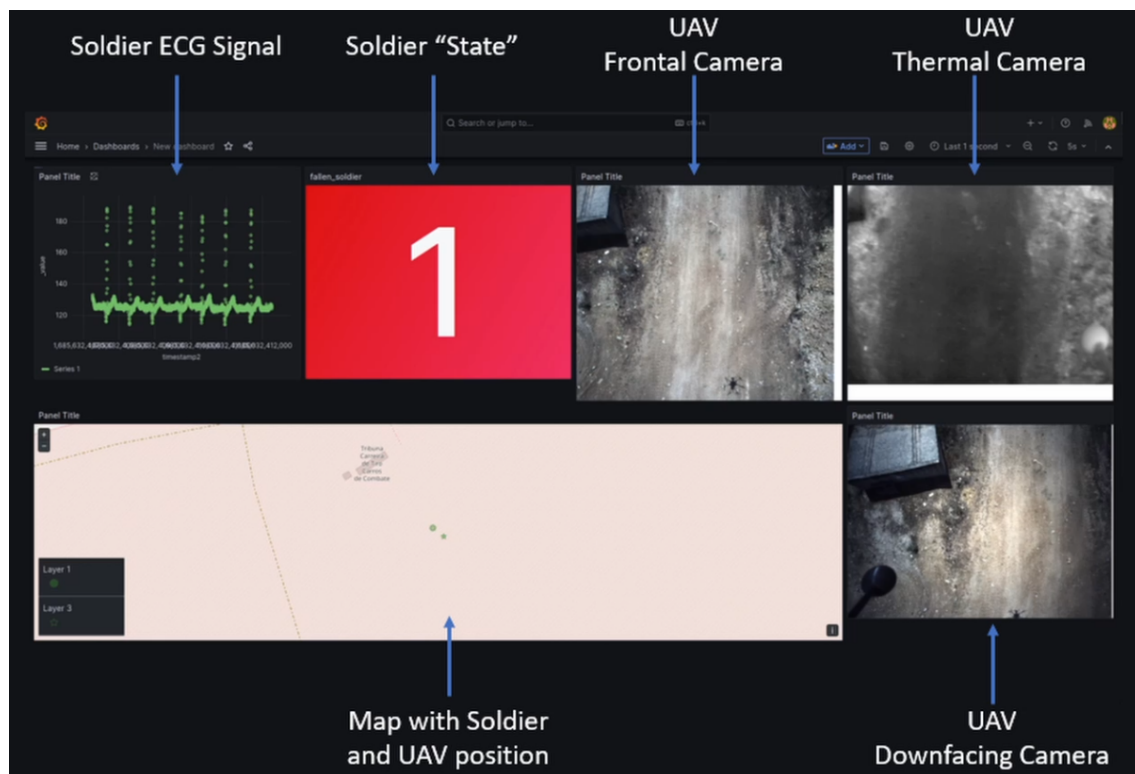


Figure 4.8: Dashboard showing the ECG signal from the soldier, fall state indicating if the soldier has fallen (red) or not (green), a map with the position from both the UAV and the soldier and three video feeds from the UAV.

identifiable key in this situation. Another interesting fact is that the use of this tool allowed the delivery of asynchronous messages, being able to implement non-blocking logic. It also ensures eventual consistency having therefore a loosely coupled system that enforces its distributed capabilities. Due to the nature of the system we can also guarantee total ordering of the received messages.

Therefore, we conclude that HiveMQ fulfilled the desired behaviour as a distributed message broker. In the reference architecture the node for message broker is, in this implementation, HiveMQ, and is in fact a node group, since every device runs a HiveMQ node, being it a distributed message broker.

Control Center dashboard

The figure 4.8 reveals the dashboard implemented with Grafana. It contains a time-series analysis of the soldier ECG signal and a colour changing alert that indicates is "state" or well-being. In the same dashboard it is also possible to see the geographical position of all actors, namely the UAV and the soldier. Moreover, a direct video feed allows the visualization and control of the UAVs cameras, allowing for better control and insight collection of the mission situation. The video in <https://tinyurl.com/MMCS1> show this dashboard and its usage during a real-world experiment.



Figure 4.9: The team and the UAV in the ARTEX experiment



Figure 4.10: Current equipment guided tour

Chapter 5

Conclusion

This work is of great relevance to the evolution of human-machine symbiosis concept. Its results not only corroborate Licklider's predictions but also set the foundations for the evolution of this field. In it we aimed to define a reference architecture to implement MMCS systems. Not only was this goal achieved but it was also verified, and an example implementation was produced and tested in a very constrained environment.

The results showed that the developed reference architecture is viable. Not only we were able to implement it, but it was proven to be applicable in adversarial and highly constrained environments as it is a search and rescue mission without degradation on the performance of both human and the machine. It was also shown that it was possible to develop a system making use of only consumer-grade hardware and that the developed model is flexible and adaptable to other technologies being its modules interchangeable and not constrained by software or operative systems. It was also shown that both the processing power and autonomy of the machine are not negatively impacted in the testing implementation meaning that using this reference architecture the need for purpose-built hardware is not mandatory. The reference architecture considers, as well, the difficulties of network management in adversarial environments, safety constraints and sets way for extending the system. The magnitude of this outcome is further enhanced using either open-source technologies or commercial products that are compliant with standards and interconnecting them. The results presented show a very usable and robust base that is easy to extend setting a clear and usable path for further developments. Not only human-machine symbiosis is possible as it has a solid foundation to be built and improved upon.

Moreover, we achieved a reliable implementation upon which future works can expand. Not only that but we had a successful participation in a Portuguese Army exercise (ARTEX23) and produced a paper for the ROBOT23 conference that can be read in [appendix A](#).

Overall, this work was successful, accomplishing all of its goals. The next section details 5 areas of future expansion and improvements.

5.1 Future work

This project tackles an extremely broad topic. The number of uses of this technology and the number of different interactions to which it can be applied convey the existence of multiple edge cases. Therefore, providing a unified architecture that tackles all of them is difficult. Therefore, and due to the constant evolution of technology, there are a few topics which can be improved upon. Most of them are part of an effort to decentralize the system for it not to be dependent on any node at any time. Although in our experiment the UAV was a main actor which could be working as a central node if it had the autonomy and processing power for it, in other use cases the machine might not be able to assume that role, and thinking in a multi-machine architecture, the idea of a distributed system with agents is more suitable. In summary, this concept is not static but rather an evolving one with the potential for further expansion and development and this work is part of a continuous effort to turn it into a reality.

Distributed Processing

Although the system provides an initial effort to distribute itself more can be done. More specifically, being able to use the devices on the network to enhance the decision processing will help make the system more resilient. This will also help in situations where one device can use cloud computation power to enhance further the results. Moreover, devices with less autonomy or computing power would benefit of offloading these tasks to other more suitable devices. As of now, the orchestration of different technologies is also centralized and should be decentralised to prevent the system from halting if communication with this node is lost.

Network reliability

To satisfy other use cases this architecture should be tested with other networks and including backup mechanisms. This is, the use of mesh networks or other network protocols should be tested as well as the integration of failover to other technologies like 4G. Moreover, the ability to use one device of the network as a gateway for using more advanced computing technologies or data transmission to remote databases should also be evaluated.

Container and cloud technologies

Although this work shows the benefits of container usage, further advances should dive upon the utility of moving to other industry-standard technologies like Kubernetes. Kubernetes would add complexity to the system but would also provide benefits when scaling capabilities would be needed.

5.1.1 Storage and database technologies

This work provides a very simple usage of databases for data storage. This should be studied upon to provide more tailored solutions and with better performance. The study and modulation

of appropriate schemas as well as the introduction of failover and replication nodes cannot be underestimated for the successful progress and evolution of the system.

5.2 Security

Security cannot be underestimated specially when dealing with autonomous machines. In this case it is even more relevant since we have a direct interaction between man and machine and their use in possibly critical situations. Therefore, a thorough security analysis and improvements should be done. This project makes use of node-based technologies and network infrastructure, both are very prone to attacks. The use of open-source technology helps tackling these issues since there is constant feedback and testing, however the integration on our system might also be prone to having flaws that need to be studied.

References

- [1] V. D. V. Minhoto, “Man-Machine Symbiosis - EEG and other biosignal man-machine interaction,” M.S. Thesis, Faculty of Engineering of the University of Porto, 2019.
- [2] J. C. R. Licklider, “Man-Computer Symbiosis,” *IRE transactions on human factors in electronics*, Mar. 1960.
- [3] “Robot 2023,” (Accessed 2023-09-28).
- [4] “SEIDROB – Sociedad Española para la Investigación y Desarrollo en Robótica,” (Accessed 2023-09-28).
- [5] “Sociedade Portuguesa de Robótica,” <http://www.sprobotica.pt/>, (Accessed 2023-09-28).
- [6] M. Steen and A. S. Tanenbaum, *Distributed Systems*, third edition, version 3.01 ed. Erscheinungsort nicht ermittelbar: Maarten van Steen, 2017.
- [7] “What is Distributed Computing? - Distributed Computing - AWS,” <https://aws.amazon.com/what-is/distributed-computing/>, (Accessed 2023-01-27).
- [8] “What Is Utility Computing? Definition, Process, Examples, and Best Practices,” (Accessed 2023-09-02).
- [9] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas behind Reliable, Scalable, and Maintainable Systems*, first edition ed. Boston: O’Reilly Media, 2017.
- [10] I. Toumlilt, P. Sutra, and M. Shapiro, “Highly-available and consistent group collaboration at the edge with colony,” in *Proceedings of the 22nd International Middleware Conference*, ser. Middleware ’21. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 336–351, (Accessed 2023-09-28).
- [11] Atlassian, “What is a distributed system?” <https://www.atlassian.com/microservices/microservices-architecture/distributed-architecture>, (Accessed 2023-09-03).
- [12] International Organization for Standardization, “ISO/IEC JTC 1 WG5 AHG1 (Jul 2014). Internet of Things (IoT).” Jul. 2014, (Accessed 2023-01-30).
- [13] S. Magazine and A. Gabbai, “Kevin Ashton Describes "the Internet of Things",” <https://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/>, (Accessed 2023-01-27).
- [14] S. Madakam, R. Ramaswamy, and S. Tripathi, “Internet of Things (IoT): A Literature Review,” *Journal of Computer and Communications*, vol. 03, no. 05, pp. 164–173, 2015, (Accessed 2023-01-30).

- [15] H. Koziolok, S. Grüner, and J. Rückert, “A Comparison of MQTT Brokers for Distributed IoT Edge Computing,” in *Software Architecture*, ser. Lecture Notes in Computer Science, A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya, and O. Zimmermann, Eds. Cham: Springer International Publishing, 2020, pp. 352–368.
- [16] S. Dustdar, “Distributed Computing Continuum Systems,” in *2022 IEEE International Conference on Services Computing (SCC)*, Jul. 2022, pp. 356–356.
- [17] D. Patel, “Distributed Computing for Internet of Things (IoT),” in *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*, Jan. 2021, pp. 1874–1894.
- [18] R. López-Viana, J. Díaz, and J. E. Pérez, “Continuous Deployment in IoT Edge Computing : A GitOps implementation,” in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2022, pp. 1–6.
- [19] T. Shao, D. Chowdhury, S. S. Gill, and R. Buyya, “IoT-Pi: A machine learning-based lightweight framework for cost-effective distributed computing using IoT,” *Internet Technology Letters*, vol. 5, no. 3, p. e355, 2022, (Accessed 2023-01-13).
- [20] B. Shang, S. Liu, S. Lu, Y. Yi, W. Shi, and L. Liu, “A Cross-Layer Optimization Framework for Distributed Computing in IoT Networks,” in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, Nov. 2020, pp. 440–444.
- [21] O. A. Mahmood, A. R. Abdellah, A. Muthanna, and A. Koucheryavy, “Distributed Edge Computing for Resource Allocation in Smart Cities Based on the IoT,” *Information*, vol. 13, no. 7, p. 328, Jul. 2022, (Accessed 2023-01-09).
- [22] C. Avasalcai, B. Zarrin, P. Pop, and S. Dustdar, “Efficient Hosting of Robust IoT Applications on Edge Computing Platform,” in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, May 2020, pp. 1–10.
- [23] P. Kayal, “Kubernetes: Towards Deployment of Distributed IoT Applications in Fog Computing,” in *Companion of the ACM/SPEC International Conference on Performance Engineering*, ser. ICPE ’20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 32–33, (Accessed 2023-01-09).
- [24] G. Ramachandran, L. Garcia, and B. Krishnamachari, “Distributed Computing for Internet of Things Under Adversarial Environments,” in *IoT for Defense and National Security*, Jan. 2023, pp. 285–306.
- [25] S. Pešić, M. Ivanović, M. Radovanović, and C. Bădică, “CAAVI-RICS model for observing the security of distributed IoT and edge computing systems,” *Simulation Modelling Practice and Theory*, vol. 105, p. 102125, Dec. 2020, (Accessed 2023-01-09).
- [26] I. Psaras, “Decentralised Edge-Computing and IoT through Distributed Trust,” in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 505–507, (Accessed 2023-01-09).
- [27] “IEEE 802.11, The Working Group Setting the Standards for Wireless LANs,” <https://www.ieee802.org/11/>, (Accessed 2023-09-04).

- [28] I. O. f. S. , “ISO_IEC_20922:2016,” 2016, (Accessed 2023-01-30).
- [29] Y. Peng, C. Wang, Q. Li, L. Liu, and K. Yu, “Distributed collaboration and anti-interference optimization in edge computing for IoT,” *Journal of Parallel and Distributed Computing*, vol. 163, pp. 156–165, May 2022, (Accessed 2023-01-09).
- [30] J. Wyrębowicz, K. Cabaj, and J. Krawiec, “Messaging Protocols for IoT Systems—A Pragmatic Comparison,” *Sensors*, vol. 21, no. 20, p. 6904, Jan. 2021, (Accessed 2023-09-04).
- [31] “About : Node-RED,” <https://nodered.org/about/>, (Accessed 2023-06-05).
- [32] D. L. R. Torres, “Increasing the Feedback on IoT Development in Node-RED,” Master’s thesis, Faculty of Engineering of the University of Porto, Jul. 2020.
- [33] “Zapier | Automation that moves you forward,” <https://zapier.com/>, (Accessed 2023-09-29).
- [34] N. Bouchard, “IFTTT vs. Zapier in 2023 (Pricing, Features, & Automations),” <https://unito.io/blog/ifttt-vs-zapier/>, Feb. 2022, (Accessed 2023-09-29).
- [35] IFTTT, “IFTTT - Automate business & home,” <https://ifttt.com/>, (Accessed 2023-09-29).
- [36] “Huginn/huginn,” Huginn, Sep. 2023, (Accessed 2023-09-29).
- [37] “Beehive vs huginn vs n8n vs node-red vs st2 | npm trends,” <https://npmtrends.com/beehive-vs-huginn-vs-n8n-vs-node-red-vs-st2>, (Accessed 2023-09-29).
- [38] “O que são contêineres?” <https://cloud.google.com/learn/what-are-containers?hl=pt-br>, (Accessed 2023-06-12).
- [39] “What is a Container? | Docker,” <https://www.docker.com/resources/what-container/>, Nov. 2021, (Accessed 2023-06-12).
- [40] R. Brooks, “What is computational thinking?” <https://online.york.ac.uk/what-is-computational-thinking/>, Sep. 2022, (Accessed 2023-09-03).
- [41] D. J. Matthies, M. Gabrecht, and H. Hellbrück, “Cyber-Physical & Human Systems (CPHS) – A Review and Outlook,” in *Mensch Und Computer 2023*. Rapperswil Switzerland: ACM, Sep. 2023, pp. 364–369, (Accessed 2023-09-03).
- [42] A. Gerber, P. Derckx, D. Döppner, and D. Schoder, “Conceptualization of the Human-Machine Symbiosis – A Literature Review,” in *Hawaii International Conference on System Sciences*, Jan. 2020.
- [43] K. S. Gill, *Human Machine Symbiosis: The Foundations of Human-centred Systems Design*. Springer Science & Business Media, Dec. 2012.
- [44] D. Narciso, M. Melo, S. Rodrigues, J. P. Cunha, J. Vasconcelos-Raposo, and M. E. Bessa, “Using Heart Rate Variability for Comparing the Effectiveness of Virtual vs Real Training Environments for Firefighters,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2022, (Accessed 2022-11-23).
- [45] S. Rodrigues, J. S. Paiva, D. Dias, and J. P. S. Cunha, “Stress among on-duty firefighters: An ambulatory assessment study,” *PeerJ*, vol. 6, p. e5967, Dec. 2018, (Accessed 2022-11-23).

- [46] J. S. Paiva, D. Dias, and J. P. S. Cunha, “Beat-ID: Towards a computationally low-cost single heartbeat biometric identity check system based on electrocardiogram wave morphology,” *PLOS ONE*, vol. 12, no. 7, p. e0180942, Jul. 2017, (Accessed 2022-11-23).
- [47] “InfluxDB outperforms MongoDB with 5x faster query performance,” <https://www.influxdata.com/resources/benchmarking-influxdb-vs-mongodb-for-time-series-data-metrics-and-management/>, Wed, 06 Jul 2022 01:30:44 +0000, (Accessed 2023-06-11).
- [48] P. D. . Jul 19, . . InfluxDB, Community, Developer, and Flux, “Why We’re Building Flux, a New Data Scripting & Query Language,” <https://www.influxdata.com/blog/why-were-building-flux-a-new-data-scripting-and-query-language/>, Thu, 19 Jul 2018 08:54:58 -0700, (Accessed 2023-06-14).
- [49] B. Abareshi, D. Williams, and R. Marshall, “Building a telescope engineering data system with Redis, InfluxDB and Grafana,” in *Software and Cyberinfrastructure for Astronomy V*, Jul. 2018, p. 55.
- [50] A. Aggoune and Z. Benratem, “ECG Data Visualization: Combining the power of Grafana and InfluxDB,” in *2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS)*, Mar. 2023, pp. 1–6, (Accessed 2023-09-29).
- [51] “Grafana: The open observability platform,” <https://grafana.com/>, (Accessed 2023-06-11).
- [52] P. D. . Jul 10 and . . InfluxData, “Message from InfluxData Founder & CTO Paul Dix: Discontinuation of InfluxDB Cloud in AWS Sydney and GCP Belgium,” <https://www.influxdata.com/blog/update-from-influxdata-paul-dix-july-10/>, Mon, 10 Jul 2023 04:00:00 +0000, (Accessed 2023-09-04).
- [53] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Boston: Addison-Wesley, 2004.
- [54] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” Internet Engineering Task Force, Request for Comments RFC 7252, Jun. 2014, (Accessed 2023-09-04).
- [55] I. Craggs, “MQTT vs CoAP for IoT,” <https://www.hivemq.com/article/mqtt-vs-coap-for-iot/>, (Accessed 2023-09-04).
- [56] A. Dias, J. Almeida, A. Oliveira, T. Santos, A. Martins, and E. Silva, “Unmanned Aerial Vehicle for Wind-Turbine Inspection. Next Step: Offshore,” in *OCEANS 2022, Hampton Roads*, Oct. 2022, pp. 1–6.
- [57] F. Azevedo, A. Dias, J. Almeida, A. Oliveira, A. Ferreira, T. Santos, A. Martins, and E. Silva, “Real- Time LiDAR-based Power Lines Detection for Unmanned Aerial Vehicles,” in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Porto, Portugal: IEEE, Apr. 2019, pp. 1–8, (Accessed 2023-09-25).
- [58] “ROS/Introduction - ROS Wiki,” <http://wiki.ros.org/ROS/Introduction>, (Accessed 2023-06-12).
- [59] “ROS/Tutorials/MultipleMachines - ROS Wiki,” <http://wiki.ros.org/ROS/Tutorials/MultipleMachines>, (Accessed 2023-09-04).

- [60] “Rosbridge_suite - ROS Wiki,” http://wiki.ros.org/rosbridge_suite, (Accessed 2023-06-12).
- [61] “Web_video_server - ROS Wiki,” http://wiki.ros.org/web_video_server, (Accessed 2023-06-12).
- [62] J. Guerra, L. Coelho, D. Dias, and J. Cunha, “Human Actigraphy Analysis Through an Inertial Sensor,” in *Congreso Anual de La Sociedad Española de Ingeniería Biomédica*, Bilbao, Nov. 2017.
- [63] J. P. S. Cunha, B. Cunha, A. S. Pereira, W. Xavier, N. Ferreira, and L. Meireles, “Vital-Jacket®: A wearable wireless vital signs monitor for patients’ mobility in cardiology and sports,” in *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, Mar. 2010, pp. 1–2.
- [64] E. Spilmont, D. Dias, and J.P.S. Cunha, “AmbiUnit: A new adaptive wearable environmental monitoring device for hazardous professions,” in *Inforum2018*, Coimbra, 2018.
- [65] “Installation/Running Docker on Android Devices - General Discussions / General,” <https://forums.docker.com/t/installation-running-docker-on-android-devices/130128>, Oct. 2022, (Accessed 2023-09-03).
- [66] “Package request: Docker engine · Issue #60 · termux/termux-root-packages,” <https://github.com/termux/termux-root-packages/issues/60>, (Accessed 2023-09-03).
- [67] F. Oliveira, “This tutorial shows how to run docker natively on Android, without VMs and chroot,” <https://gist.github.com/FreddieOliveira/efe850df7ff3951cb62d74bd770dce27>, Mar. 2021, (Accessed 2023-09-03).
- [68] Jedda, Karim, “Run Docker and Nomad on Android - truly serverless mobile compute,” <https://karimjedda.com/run-docker-on-android/>, Apr. 2022, (Accessed 2023-03-13).
- [69] “Geekbench 6 - Cross-Platform Benchmark,” <https://www.geekbench.com/>, (Accessed 2023-09-29).
- [70] “7-Zip LZMA Benchmark,” <https://www.7-cpu.com/>, (Accessed 2023-09-29).
- [71] “Processor Benchmarks - Geekbench,” <https://browser.geekbench.com/processor-benchmarks>, (Accessed 2023-09-29).
- [72] “About the HiveMQ Company,” <https://www.hivemq.com/company/>, (Accessed 2023-06-05).
- [73] M. Agarwal, “Basics of Distributed System,” <https://interviewnoodle.com/basics-of-distributed-system-ed4df291b38>, Nov. 2021, (Accessed 2023-02-06).
- [74] A. Alnoman, S. K. Sharma, W. Ejaz, and A. Anpalagan, “Emerging Edge Computing Technologies for Distributed IoT Systems,” *IEEE Network*, vol. 33, no. 6, pp. 140–147, Nov. 2019.
- [75] “A Brief History of Drones,” <https://www.iwm.org.uk/history/a-brief-history-of-drones>, (Accessed 2023-01-23).

- [76] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and S. J. Kim, “Challenges and implemented technologies used in autonomous drone racing,” *Intelligent Service Robotics*, vol. 12, no. 2, pp. 137–148, Apr. 2019, (Accessed 2023-01-23).
- [77] W. Chen, L. Xu, G. Li, and Y. Xiang, “A Lightweight Virtualization Solution for Android Devices,” *IEEE Transactions on Computers*, vol. 64, no. 10, pp. 2741–2751, Oct. 2015.
- [78] “Definition of INTERNET OF THINGS,” <https://www.merriam-webster.com/dictionary/Internet+of+Things>, (Accessed 2023-01-27).
- [79] P. Dix, “Why Time Series Matters for Metrics, Real-Time Analytics and Sensor Data,” <https://get.influxdata.com/rs/972-GDU-533/images/why%20time%20series.pdf>, Jul. 2021, (Accessed 2023-06-11).
- [80] S. Dustdar, V. Pujol, and P. K. Donta, “On distributed computing continuum systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, Jan. 2022.
- [81] “Electrocardiogram Sampling Frequency Range Acceptable for Heart Rate Variability Analysis - PMC,” <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6085204/>, (Accessed 2023-07-04).
- [82] K. Ferencz and J. Domokos, “Rapid Prototyping of IoT Applications for the Industry,” in *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, May 2020, p. 6.
- [83] “Flux vs InfluxQL | InfluxDB OSS 1.7 Documentation,” <https://docs.influxdata.com/influxdb/v1.7/flux/flux-vs-influxql/>, (Accessed 2023-06-11).
- [84] A. S. Hassan, “A decentralized MQTT broker for IoT on Web 3.0,” <https://blog.cryptostars.is/a-decentralized-mqtt-broker-for-iot-on-web-3-0-1b45ce191f44>, Jan. 2022, (Accessed 2023-06-12).
- [85] “IBM Documentation,” <https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/txseries/8.2?topic=ov-what-is-distributed-computing>, Apr. 2021, (Accessed 2023-02-06).
- [86] “IEEE 802.11s: The WLAN Mesh Standard | IEEE Journals & Magazine | IEEE Xplore,” <https://ieeexplore.ieee.org/document/5416357>, (Accessed 2023-09-04).
- [87] “On InfluxDB 3.0: The Future of Time Series Analytics. Q&A with Gunnar Aasen – ODBMS.org,” Apr. 2023, (Accessed 2023-06-11).
- [88] “InfluxDB - Moving from InfluxQL to Flux language,” <https://www.sqlpac.com/en/documents/influxdb-moving-from-influxql-language-to-flux-language.html>, (Accessed 2023-06-11).
- [89] “Introduction to Grafana | Grafana documentation,” <https://grafana.com/docs/grafana/latest/introduction/>, (Accessed 2023-06-11).
- [90] “An introduction to time series databases,” <https://aiven.io/blog/an-introduction-to-time-series-databases>, (Accessed 2023-06-12).

- [91] P. Iob, L. Frau, P. Danieli, L. Olivieri, and C. Bettanini, "Avalanche Rescue with Autonomous Drones," in *2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, Jun. 2020, pp. 319–324.
- [92] O. Kwon, J. Jeong, H. B. Kim, I. H. Kwon, S. Y. Park, J. E. Kim, and Y. Choi, "Electrocardiogram Sampling Frequency Range Acceptable for Heart Rate Variability Analysis," *Healthcare Informatics Research*, vol. 24, no. 3, pp. 198–206, Jul. 2018, (Accessed 2023-07-04).
- [93] J. P. Machado, "Near real-time analytics engine for vitals and environmental monitoring of first responders," Master's thesis, Faculty of Engineering of the University of Porto, Jul. 2020.
- [94] "5 Main Node.js Advantages and Disadvantages | EPAM Anywhere Business," <https://anywhere.epam.com/business/node-js-pros-and-cons>, (Accessed 2023-06-05).
- [95] B. Mishra, B. Mishra, and A. Kertesz, "Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements," *Energies*, vol. 14, no. 18, p. 5817, Jan. 2021, (Accessed 2023-06-05).
- [96] J.-M. Sierra-Fernández, O. Florencias-Oliveros, M.-J. Espinosa-Gavira, J.-J. González-de-la-Rosa, A. Agüera-Pérez, and J.-C. Palomares-Salas, "Online System for Power Quality Operational Data Management in Frequency Monitoring Using Python and Grafana," *Energies*, vol. 14, no. 24, p. 8304, Jan. 2021, (Accessed 2023-09-26).
- [97] E. Oyekanlu and K. Scoles, "Real-Time Distributed Computing at Network Edges for Large Scale Industrial IoT Networks," in *2018 Ieee World Congress on Services (Ieee Services 2018)*. New York: Ieee, 2018, pp. 63–64, (Accessed 2023-01-09).
- [98] M.-P. Pacaux-Lemoine and D. Trentesaux, "ETHICAL RISKS OF HUMAN-MACHINE SYMBIOSIS IN INDUSTRY 4.0: INSIGHTS FROM THE HUMAN-MACHINE COOPERATION APPROACH," *IFAC-PapersOnLine*, vol. 52, no. 19, pp. 19–24, 2019, (Accessed 2023-09-03).
- [99] "Package request: Podman · Issue #10188 · termux/termux-packages," <https://github.com/termux/termux-packages/issues/10188>, (Accessed 2023-09-03).
- [100] L. Papageorgiou, P. Eleni, S. Raftopoulou, M. Mantaïou, V. Megalooikonomou, and D. Vlachakis, "Genomic big data hitting the storage bottleneck," *EMBnet.journal*, vol. 24, p. e910, 2018, (Accessed 2023-06-06).
- [101] "Poor performance on last() · Issue #14564 · influxdata/influxdb," <https://github.com/influxdata/influxdb/issues/14564>, (Accessed 2023-06-12).
- [102] K. Rathee, "The Landscape of Timeseries Databases," <https://towardsdatascience.com/the-landscape-of-timeseries-databases-95cd7f7ee64d>, Jan. 2023, (Accessed 2023-06-11).
- [103] R. Chamberlain, "6 reasons your Node.js apps are failing," <https://developer.ibm.com/articles/6-reasons-your-node-js-apps-are-failing/>, Sep. 2018, (Accessed 2023-06-05).
- [104] "ROS: Why ROS?" <https://www.ros.org/blog/why-ros/>, (Accessed 2023-06-12).

- [105] E. Shahri, P. Pedreiras, and L. Almeida, “Extending MQTT with Real-Time Communication Services Based on SDN,” *Sensors*, vol. 22, no. 9, p. 3162, Apr. 2022, (Accessed 2023-06-14).
- [106] A. Solnichkin, “SQL is Dead, Hail to Flux,” Dec. 2018, (Accessed 2023-06-11).
- [107] “SQL vs. Flux: Choose the right query language for time-series data,” <https://www.timescale.com/blog/sql-vs-flux-influxdb-query-language-time-series-database-290977a01a8a/>, Aug. 2018, (Accessed 2023-06-11).
- [108] Z. Telnarova, “Using relational databases for time series data,” in *INTERNATIONAL CONFERENCE OF COMPUTATIONAL METHODS IN SCIENCES AND ENGINEERING 2018 (ICCMSE 2018)*, Thessaloniki, Greece, 2018, p. 030003, (Accessed 2023-06-12).
- [109] “Time Series Database (How It Works & Use Cases),” (Accessed 2023-06-12).
- [110] Y. Hao, X. Qin, Y. Chen, Y. Li, X. Sun, Y. Tao, X. Zhang, and X. Du, “TS-Benchmark: A Benchmark for Time Series Databases,” in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, Apr. 2021, pp. 588–599, (Accessed 2023-09-26).
- [111] K. Vyas, “A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles (UAVs),” <https://interestingengineering.com/innovation/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs>, Jun. 2020, (Accessed 2023-01-23).
- [112] M. Wolf, “Machine Learning + Distributed IoT = Edge Intelligence,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2019, pp. 1715–1719.
- [113] H.-J. Yoon, “A Study on the Performance of Android Platform,” *International Journal on Computer Science and Engineering*, vol. 4, Apr. 2012.

Appendix A

Man-Machine Symbiosis UAV Integration for Military Search and Rescue Operations

Man-Machine Symbiosis UAV Integration for Military Search and Rescue Operations

V. Minhoto^{1,2}, T. Santos¹, L. Torres E Silva^{1,2}, P. Rodrigues¹, A. Arrais¹, A. Amaral¹, A. Dias^{1,3}, J. Almeida^{1,3}, and J.P.S. Cunha^{1,2}

¹ INESC TEC, Campus da Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, Porto, Portugal {vitor.minhoto, tiago.a.santos, luis.m.silva, paulo.m.rodrigues, adriana.g.arrais, alexandre.a.oliveira, andre.dias, jose.m.almeida, joao.p.cunha}@inesctec.pt

² Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, Porto, Portugal

³ Instituto Superior de Engenharia do Porto, Rua Dr. António Bernardino de Almeida, 431, Porto, Portugal

Abstract. Over the last few years, Man-Machine collaborative systems have been increasingly present in daily routines. In these systems, one operator usually controls the machine through explicit commands and assesses the information through a graphical user interface. Direct & implicit interaction between the machine and the user does not exist.

This work presents a man-machine symbiotic concept & system where such implicit interaction is possible targeting search and rescue scenarios. Based on measuring physiological variables (e.g. body movement or electrocardiogram) through wearable devices, this system is capable of computing the psycho-physiological state of the human and autonomously identify abnormal situations (e.g. fall or stress). This information is injected into the control loop of the machine that can alter its behavior according to it, enabling an implicit man-machine communication mechanism.

A proof of concept of this system was tested at the ARTEX (ARmy Technological EXperimentation) exercise organized by the Portuguese Army involving a military agent and drone. During this event the soldier was equipped with a kit of wearables that could automatically monitor several physiological variables and detect a fall during a mission. This information was continuously sent to the drone that successfully identified this abnormal situation triggering the take-off and a situation awareness fly-by flight pattern, delivering a first-aid kit to the soldier in case he did not recover after a pre-determined time period.

The results were very positive, proving the possibility and feasibility of a symbiotic system between humans and machines.

Keywords: Man-Machine Symbiosis · Search and Rescue · Unmanned Aerial Vehicle

1 Introduction

Many types of natural and human-made disasters, such as earthquakes, floods, wildfires, hurricanes among others, can cause different disaster outcomes. In these situations, search and rescue operations are often characterized by a similar set of constraints: complex search and rescue (SAR) decisions to be made (e.g. where to search first, victims rescue priority, etc.), lack of information on the critical situations and time criticality as any delay can result in dramatic consequences [1]. A recent report from the World Health Organization [2] states that during the period 2011–2016, an average of 700 disasters were reported annually. These disasters affected around 200 million people.

Although Unmanned Aerial Vehicles (UAVs) – commonly referred as “drones” – have been around since the 1950s for military purposes [3], reports on their use on supporting SAR missions only emerge in the literature by the 2010’s [4,5]. Due to drone’s low price in the last decade, several studies have been developed to make these machines as autonomous as possible using computer vision [6,7,8], environmental sensors [9,10] or behaving like a swarm [11]. UAVs have been also used in SAR scenarios [12] such as to locate humans in the water [1] or helping the operation official with path planning and scouting [13]. Furthermore, in 2022, we have been watching the war in Ukraine with great concern, where off-the-shelf commercial technologies such as UAVs have been used recurrently as military operations support/observation vehicles and even as weapons.

In these examples, one user usually can control the drone through explicit commands and assess the information through a graphical user interface. Direct & implicit interaction between the UAV and the user (e.g. drone detects his operator has fallen and performs, autonomously, a situation evaluation flyby, sending a help call to the rescue command center) on the field does not exist. Thus, our group has been developing a man-machine symbiotic concept & system between a SAR agent and a “companion” UAV that can significantly improve the agent’s efficiency & his own security during SAR missions. Searching SCOPUS and Web of Science for man-machine symbiosis in SAR scenarios shows no results assuring the novelty of this work.

Man-Machine Symbiosis was first introduced in a paper by J.C.R Licklider [14] in 1960. This work defended that (intelligent) technology should be developed to augment human intelligence, not replace it, turning the typical user-centered application into a distributed system. It involves developing systems that enable machines to think and perform tasks autonomously while being aware of their human counterparts. The goal is to create a system that promotes cooperation and collaboration between man and machine, improving efficiency, effectiveness, and autonomy in fulfilling a common objective.

SAR response has been an area the Portuguese military has been trying to improve over the last few years. To do so they created the ARTEX (ARmy Technological EXperimentation) exercise opening a military reserved area to both industry and academia to experiment with emerging technologies. In this context, this work addresses the development of a man-machine symbiotic solution, where a soldier equipped with wearable technology that monitors his

condition can trigger, in an implicit & autonomous manner the intervention of UAVs. Our team participated in the first ARTEX week-long exercise with a proof of concept of our Man-Machine symbiotic system tackling search and rescue problems. Our use case was to have the drone detect autonomously his military companion fall and, after a flyby evaluation procedure drop an emergency kit box, if the soldier did not recover from the fall after a certain amount of seconds.

For the rest of the document, the concept of Man-Machine Symbiosis and the results taken on ARTEX are presented. In Section 2, the proposed Man-Machine Symbiotic system is described together with a conceptual architecture. In Section 3, the implementation of this system for ARTEX is described in detail and the results are presented in Section 4. Finally, this work is concluded by discussing the challenges and opportunities associated with the development and deployment of Man-Machine Symbiosis systems and potential future directions for research and development in this area.

2 Man-Machine Symbiosis Concept and Architecture

2.1 Concept

According to the Macmillan dictionary [15], symbiosis is a "close relationship between two different things or people from which both get benefits". In this symbiotic relationship, neither the machine will be just a tool for the Human nor the Human will be an aid and/or supervisor of the machine's automated processes. In our concept of symbiosis, the possible interactions can either be explicit or implicit.

Explicit interactions are the most common way a human communicates with a machine. They are usually aided by visual interfaces where the machine can send information to the Human and controllers where humans send commands to the machine.

Implicit interaction, similar to the one described by Han and Kim [16], implies that both entities should perceive the state and intention of the other without issuing an explicit command. In our novel approach, for the drone to be aware of its companion Human, a continuous psychophysiological evaluation is performed by measuring its vitals using wearable devices. This information is then injected into the machine control loop through what we call the Man-Machine Symbiotic Engine, making it possible to adjust the drone behaviour based on changes in the human state, putting the Human in the Loop.

Using a combination of these two interactions, a human and a machine can then work together towards the same goal while being aware of each other tasks and state.

2.2 Conceptual Man-Machine Symbiosis Architecture

In Figure 1 a general architecture of the proposed Man-Machine Symbiosis system is presented. This system has 2 actors involved: a human and a machine. Over the following subsections, the characteristics and devices associated with to each will be detailed and discussed.

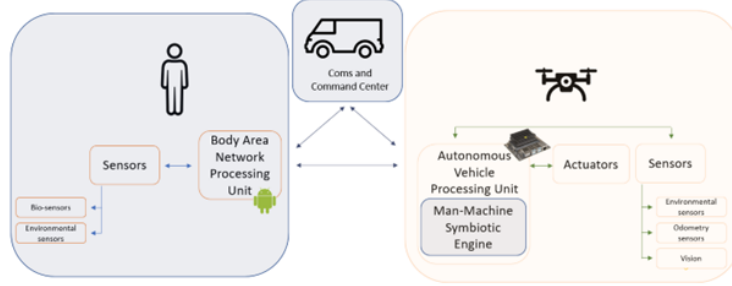


Fig. 1: Proposed Man-Machine Symbiotic System

Human and associated devices

On the human side of the symbiotic system, there are two main components: The Sensing devices and the processing unit. The sensing devices are wearable sensors that measure human physiological data and environmental characteristics. The Body Area Network Processing unit aggregates and processes the data measured from all sensors. It synchronizes the streams from all sources and computes any relevant features needed by the symbiotic system.

Machine and associated devices

On the machine side of the system, similarly to the human, there are also sensing devices and a processing unit, plus the actuators needed to move the robot. When operating, the machine usually has a fixed goal that must be accomplished before moving to the next one. The control loop reads the sensor information and controls the actuators so the robot can finish its task. In parallel to the control loop, there is a module that receives information from the human and can change the robot's goal accordingly. We call this Man-Machine Symbiotic Engine. This Engine obtains the psychophysiological data from the user processing unit and implicitly perceives the psychophysiological state and intention. This contributes to the concept of Human in the Loop, where the machine is always aware of its companion without explicitly being informed.

3 Implementation

For the ARTEX23 event, we built a proof-of-concept man-machine symbiotic system to aid the military during search and rescue activities. To do so, together with the ARTEX organizers, a target use case was created to test the feasibility of this technology. This use case consisted of a sensorized soldier and a companion UAV. The soldiers would perform a scouting round looking for victims, and one of the soldiers would simulate a fall. The system would automatically detect the fall and launch a UAV to assess the situation through a fly-by flight pattern and,

if no recovery is detected, drop a first aid kit. In this section, a description of the developed system is presented.

3.1 Vital Kit

To collect and process the data from the humans, each participant in the experiment wore a vital kit, Figure 2, containing the following sensors: a vital patch [17,18], an AmbiUnit [19] sensing box, a smartphone with the aggregator app and a GoPro camera to be attached to the user's chest.



Fig. 2: Vital Kit

The vital patch was attached to the user's chest using two electrodes or a wearable t-shirt [18]. The patch can collect the user's medical grade ECG data at 500Hz, body temperature and movement with an IMU at 10HZ [17]. This information was then sent to a mobile application via Bluetooth.

The AmbiUnit sensing box measures the surrounding environment of the soldier. This sensor was positioned at the belt and could collect the surrounding environment's atmospheric pressure, CO and NO2 concentrations, luminosity, and temperature. Similarly to the patch, it sent the information to a mobile application via Bluetooth.

The smartphone was responsible for reading and processing the data from the sensors. It also had a simple user interface if the military wanted to send a command to the UAV explicitly. The developed fall detection algorithm was also implemented within the app.

The GoPro camera was used to have a first-person view of the military during the scenario. The resulting video was synchronized to the data acquired by filming the moment the soldier pressed the application "Start" button.

3.2 Android App and Fall Detection Algorithm

For this proof of concept, a simple two-screen application was developed. The first screen contains a start button the soldier has to press to start the mission.

When the user pressed start, it would connect to the predefined wearable devices. Each wearable device sends an array of synchronized values using the timestamp taken when the Android Device reads them. The second screen was built as a debugging and fail safe tool consisting in a simple interface composed of five buttons that, additionally to the implicit actuation, allowed the user to explicitly order the drone to: go to his position, drop the first aid kit, return home and perform the area recognition instruction.

Once the device started receiving a stream from the vital sticker, the fall detector algorithm would run. A window analysis approach was implemented to update the user's fall state continuously. The inertial data was processed in consecutive 2 seconds windows, allowing for real-time analysis and updating of the fall status.

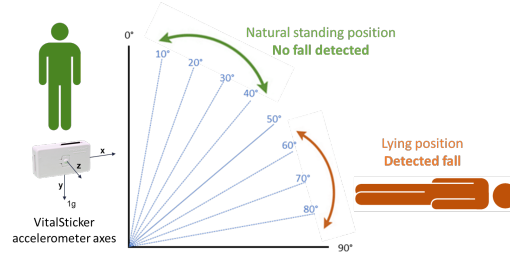


Fig. 3: Fall detector threshold.

The fall status was defined by evaluating the inclination of the soldier's upper body, based on a study by Li *et al* [20]. In detecting fall events, a threshold value of 50° was established. Two critical points were identified: the inclination start and finish points. The inclination start points were determined by detecting instances where there was a decrease in inclination to a value inferior to 50° ; in turn, the inclination finish points were identified when there is an increase in inclination to values greater than 50° , Figure 3.

All inclination windows were examined for the presence of fall start or finish points and the fall status was updated at each iteration. The fall detection logic was structured around two distinct states relative to the "fall_indice" variable. When the analysis started in the application, the "fall_indice" was initialized to -1. As the data was processed, if a fall was identified without a subsequent return to an upright posture, the "fall_indice" was updated to the index of the detected fall's starting point. This transitioned into a positive integer value indicating the change to the second state, where the application recognized that a fall event had been detected. Once the application identifies the end of a fall event, the "fall_indice" is reset to -1 again, returning to the default state, where the focus shifts back to detecting the initiation of fall occurrences. This two-state logic, enabled real-time fall status monitoring and updates through the "FALL" variable, which was set to either true or false.

This variable was then inserted in an object, representing the state of the human, together with the raw data collected from the wearable devices. This object was broadcast every second to a specific MQTT topic.

3.3 Network Architecture

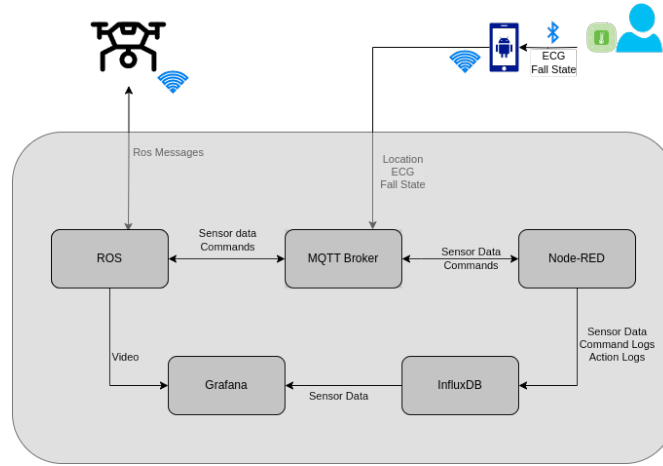


Fig. 4: Container organization and communications

A system based on docker containers was built to support, process, store and show all information being broadcast during the mission. A docker container provides a lightweight, standardized environment with software and its dependencies. By adopting this approach, it is possible to package an application, test it, and deploy it in several systems independently of their configuration and built-in packages. The deployment of software is, therefore, standardized, avoiding many of the usual problems when deploying it as if it was being deployed to a virtual, freshly installed virtual machine[21,22]. The machine that hosted the containers was an HP Zbook 17 G5 equipped with an Intel® Core™ i7-8850H, 32GB RAM and an NVIDIA Quadro P5200.

This system was required to host the MQTT broker, store all the data in a Database and present vital information from the soldier and data from the drone in a dashboard, providing a full monitoring unit in a single machine. To communicate with the UAV, a 5Ghz Wi-Fi antenna (IEEE 802.11a) was used.

MQTT Broker For each node to communicate, the publish/subscribe protocol MQTT, Message Queuing Telemetry Transport, was used. The MQTT broker used was HiveMQ MQTT broker. It was selected not only for its ability to deal with vast amounts of messages in usable time but mainly for its ability to work

distributed and in clusters [24]. Each device can then execute an instance of the broker while being part of the cluster. This guarantees redundancy since all other communications remain working in case of a device disconnection.

Database As in many applications and experiments, data collection is fundamental for logging information and gathering metrics and working on improvements. The database is an essential part of the system, as seen in figure 4 since our monitoring solution relies on it. A time-series database was desired since all data is timestamped. Therefore, InfluxDB was selected since it is a popular open-source time-series database designed to handle high volumes of timestamped data efficiently [25].

Monitoring solution As with every robotic system experiment, there should be a monitoring technique. Because our investigation relied on the travelling of certain distances by the robot, and it was impossible and unsafe to walk beside it, monitoring and data visualization were crucial. Grafana is an open-source data visualization tool built for time-series data [26]. Due to its open-source nature, it has many community-built extensions that increase its capabilities, such as setting up alerts or even creating action buttons, making it complete as a command-and-control centre, and therefore it allows to get alerts when the soldier falls to the ground, four live camera feeds from the UAV, live updates on the drone's and soldier's location in a map and live representation of the soldier's ECG.

Robotic Operating System The UAV used on this mission is based on ROS (Robotic operative system). This way, to get the video feed from the aircraft, a ROS container was created to subscribe to the topics where the frames were being published. This local node was used to create a live video broadcast of the cameras on the mission, a more CPU-heavy task, without compromising the UAV's autonomy. This live broadcast was then sent to Grafana for real-time display.

Man-Machine Symbiotic Engine This module inputs the state of the human and information from the drone and computes what course of action the machine should take to aid the soldier during this mission better. Due to the simplicity of the task performed in this proof of concept, this Engine is just a simple state machine that reads the Human fall state and orders the drone to lift off and fly to the human position if the human is down for some seconds and come back to home once everything is ok. Once the following command to the UAV is decided, this information is published on a vehicle-specific MQTT topic to be redirected to the corresponding vehicle ROS topic. Despite the man-machine symbiotic engine being included on the machine part of the system when represented in Figure 1 this module can run anywhere within the network. In this scenario, it was run as a docker container in the HP Zbook machine.

Node-Red Node-RED is an open-source visual programming tool for connecting hardware devices, APIs, and online services[27]. It provides a browser-based flow editor allowing users to create and deploy applications by clicking pre-built nodes[28].

Node-Red was used in this system to subscribe to each MQTT topic and transform the data to publish to InfluxDB. Furthermore, it was also used to transform MQTT messages to ROS messages to send to the drone information like the position of the soldier and the command it has to perform. Node-RED orchestrated and defined the system flow.

3.4 UAV STORK for Search and Rescue Military Operations

The UAV STORK II, Figure 5a, was already available for field tests in other application scenarios such as search and rescue operations, inspection of electrical assets, and 3D reconstruction of monuments [29,30,31,32]. It consists of a customized heptagon geometry integrating six rotors on the back and a pan&tilt gimbal on the front with 120° of field-of-view. The gimbal sensor payload comprises two cameras, a thermographic and a high-resolution visual camera. The UAV payload was designed to be modular and allow for different configurations of sensors and modules, such as a LiDAR system, for 3D reconstruction, obstacle detection, and avoidance. In this case, a payload release module was added, described in the following section.

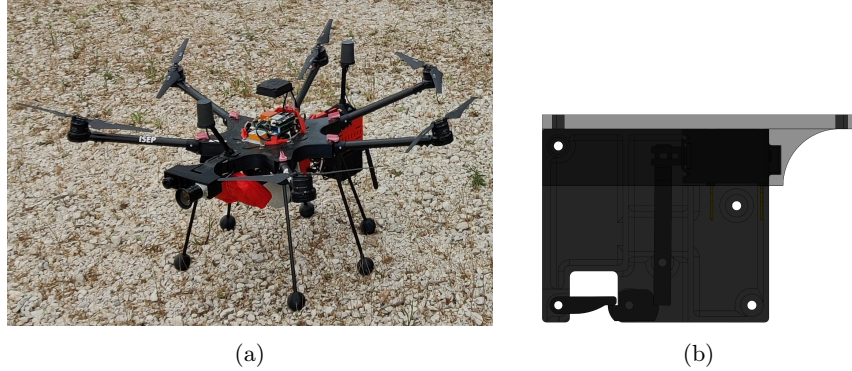


Fig. 5: a) UAV Stork II equipped with first-aid rescue pack b) Payload Release module

Payload Release System

To deploy the first-aid kit, a release system was designed. This system is composed of an electrical solenoid that, through a mechanical method of levers,

unlocks the trapdoor that holds the bag. Figure 5b shows the 3D model of the module, with transparency, showing the internal mechanism. The lever system design aims to reduce the force needed to unlock the trapdoor since the solenoid directly wouldn't be strong enough due to the friction between the bag and the part. The solenoid is directly controlled by a driver, composed of a MOSFET and an optocoupler connected to the onboard PC. Despite the small size of the system (80 x 67 x 27 mm), it allows the release of a payload of up to 5 Kg.

Stork and Android app instructions integration

The Stork CPU runs ROS and uses the Mavros package to achieve communication between the ROS ambient and the autopilot firmware. Using the setpoints instructions, the position of the UAV was controlled by the instructions received in a ROS topic by the Android app. The following instructions were used:

- ID: 0 - Land at takeoff position;
- ID: 1 - Goes to soldier position;
- ID: 2 - Area recognition;

The area recognition instruction was a pre-determined manoeuvre to fly forward to the two sides relative to the UAV. After that, it goes to the soldier position again.

In addition to these instructions, there is another ROS topic with the soldier's information (as his GPS position) and a ROS service responsible for activating the release system.

4 Results

The ARTEX exercise made available a scenery emulating a real-life example of soldiers inside a trench.

At ARTEX three missions were performed in preparation for the demonstration day, and all the data was saved for future use. A final mission was performed as a demonstration to the ARTEX organizers.

These missions consisted of a pair of soldiers patrolling the area. Suddenly one of the soldiers would simulate a fall. The system was supposed to detect the fall and autonomously deploy the UAV, carrying a first-aid kit, to aid the soldiers if needed.

In total during the event we collected about 2h of synchronized soldier ECG, position, body temperature, fall state, GoPro video, UAV position, LiDAR scan, RGB and IR video.

In Figure 6 one can see the dashboard showing synchronized data from both the Human and the UAV. The environmental variables from AmbiUnit were not used during this exercise, so they were excluded from being shown in this iteration of the dashboard. During the test and demonstration flights, the fall detection algorithm was able to correctly identify that the soldier has fallen and

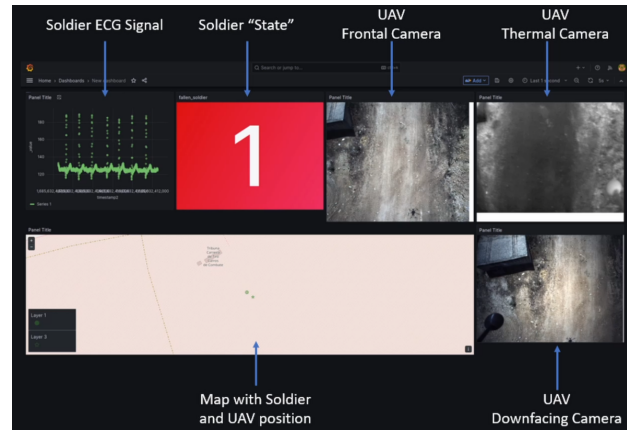


Fig. 6: Dashboard showing the ECG signal from the soldier, fall state indicating if the soldier has fallen (red) or not (green), a map with the position from both the UAV and the soldier and three video feeds from the UAV.

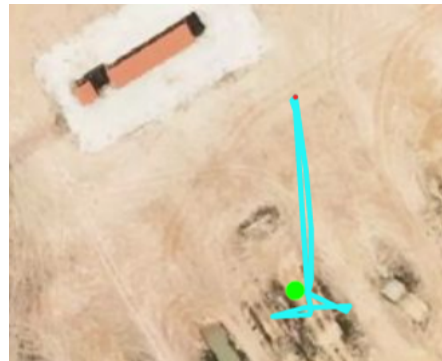


Fig. 7: UAV trajectory. The red dot represents the takeoff and landing position, and the green dot the release position.

the man-machine symbiotic engine was able to automatically send the command to the UAV to lift off and carry the first-aid kit to the soldier's position.

Using the satellite view of Google Maps, figure 7 was made, where it shows the trajectory of the UAV (blue line) and the position of the injured soldier (green circle). It can be seen that after the takeoff, the UAV flew to the soldier's position, followed by the area fly-by recognition manoeuvre. After that, it came back to the takeoff position and landed. The sensors of the UAV, cameras and LiDAR, also captured the moment, as is shown in figure 8.

A short video of the experiment can be found in <https://drive.inesctec.pt/s/Mf8dnzaX9eEazie>.

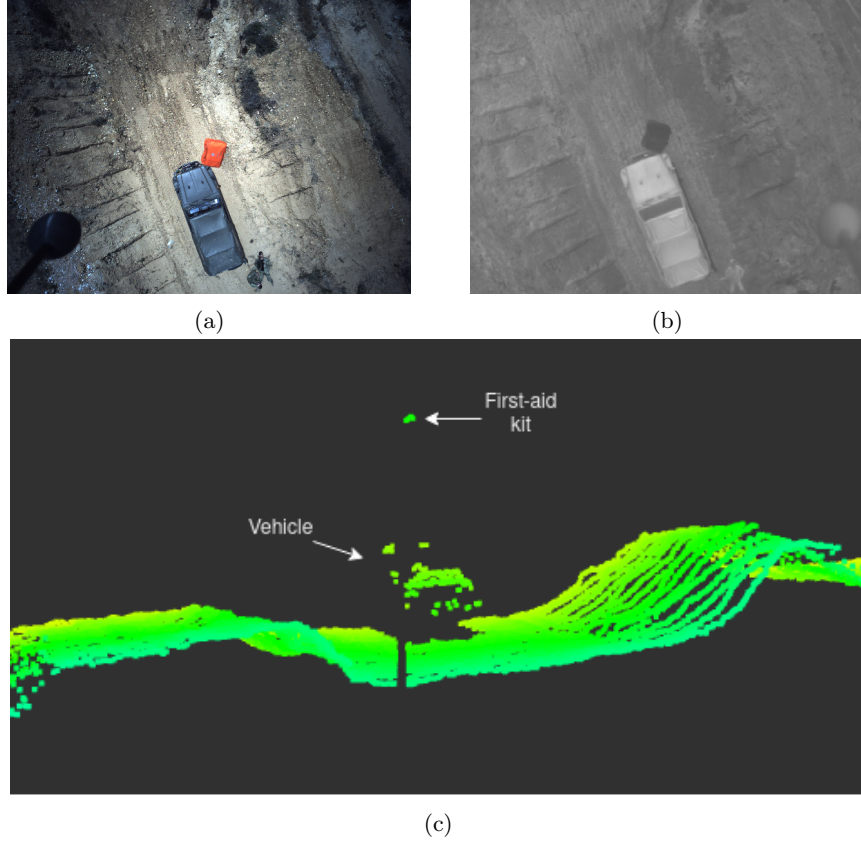


Fig.8: Cameras and Velodyne data at the moment of the first-aid kit release. (a) RGB camera (b) Thermographic camera. (c) Pointcloud.

5 Conclusions

In this work, we present a man-machine symbiotic system conceptual architecture and a use case implementation for a SAR scenario. To the best of our knowledge, this type of approach to these SAR scenarios where implicit interaction between man and UAV occurs automatically in a symbiotic way is not found in the literature. A system was developed to aid soldiers during SAR missions that monitors their psychophysiological state and injects this information into the UAV control loop, enabling implicit interactions between the man and the machine. The proof-of-concept use case was tested with success at the ARTEX military event at the Santa Margarida Army Base facilities in Portugal. The ARTEX army staff's positive feedback and invite to further explore this concept in their use cases in the near future reinforces the project's relevance and the

need for a tighter coupling between man and machines in SAR scenarios, setting the stage for further research into this kind of novel technology approaches.

Acknowledgements This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020. The group also thanks the Portuguese Army and the CEMTEX group for organizing the ARTEX event and all the support during the experiment days. The authors would also like to acknowledge and thank the contribution of Prof. Bob Iannucci (Carnegie-Mellon University and Google inc.) for his seminal contributions to the early stages of the Man-Machine Symbiosis Concept.

References

1. Waharte, S. & Trigoni, N. Supporting Search and Rescue Operations with UAVs. *2010 International Conference On Emerging Security Technologies*. pp. 142-147 (2010)
2. Emergency response framework (ERF), 2nd edition, <https://www.who.int/publications/i/item/9789241512299>; Last accessed 20 July 2023.
3. Keane, J. & Carr, S. A Brief History of Early Unmanned Aircraft. *JOHNS HOPKINS APL TECHNICAL DIGEST*. **32** (2013)
4. Sambolek, S. & Ivasic-Kos, M. Automatic Person Detection in Search and Rescue Operations Using Deep CNN Detectors. *IEEE Access*. **9** pp. 37905-37922 (2021)
5. S. Sambolek and M. Ivasic-Kos, "Automatic Person Detection in Search and Rescue Operations Using Deep CNN Detectors," in *IEEE Access*, vol. 9, pp. 37905-37922, 2021, doi: 10.1109/ACCESS.2021.3063681.
6. Dandois, J., Olano, M. & Ellis, E. Optimal Altitude, Overlap, and Weather Conditions for Computer Vision UAV Estimates of Forest Structure. *Remote Sensing*. **7**, 13895-13920 (2015,10),
7. Xu, G., Zhang, Y., Ji, S., Cheng, Y. & Tian, Y. Research on computer vision-based for UAV autonomous landing on a ship. *Pattern Recognition Letters*. **30**, 600-605 (2009,4),
8. Mejias, L., Correa, J., Mondragon, I. & Campoy, P. COLIBRI: A vision-Guided UAV for Surveillance and Visual Inspection. *Proceedings 2007 IEEE International Conference On Robotics And Automation*. pp. 2760-2761 (2007)
9. Liu, A., Mo, W. & Ling, Y. Multi-sensor environmental mapping. (2016,3), <https://patents.google.com/patent/US20160070265A1/en>
10. Bürkle, A., Segor, F. & Kollmann, M. Towards Autonomous Micro UAV Swarms. *Journal Of Intelligent & Robotic Systems*. **61**, 339-353 (2011,1), <https://doi.org/10.1007/s10846-010-9492-x>
11. Lamont, G., Slear, J. & Melendez, K. UAV Swarm Mission Planning and Routing using Multi-Objective Evolutionary Algorithms. *2007 IEEE Symposium On Computational Intelligence In Multi-Criteria Decision-Making*. pp. 10-20 (2007)
12. Doherty, P. & Rudol, P. A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization. *AI 2007: Advances In Artificial Intelligence*. pp. 1-13 (2007)
13. Ryan, A. & Hedrick, J. A mode-switching path planner for UAV-assisted search and rescue. *Proceedings Of The 44th IEEE Conference On Decision And Control*. pp. 1471-1476 (2005)

14. Licklider, J. Man-Computer Symbiosis. *IRE Transactions On Human Factors In Electronics*. **HFE-1**, 4-11 (1960)
15. SYMBIOSIS (noun) definition and synonyms | Macmillan Dictionary., <https://www.macmillandictionary.com/dictionary/british/symbiosis>; Last accessed 20 July 2023.
16. Han, J. & Kim, J. Human-robot interaction by reading human intention based on mirror-neuron system. *2010 IEEE International Conference On Robotics And Biomimetics*. pp. 561-566 (2010)
17. Guerra, J., D. Dias, J. P. S. Cunha, and L. COELHO. "Human Actigraphy's Analysis Through an Inertial Sensor." In XXXV Congreso anual de la Sociedad Española de Ingeniería Biomédica: Libro de actas, pp. 323-326. Servicio Editorial= Argitalpen Zerbitzua, 2017.
18. Cunha, J., Cunha, B., Pereira, A., Xavier, W., Ferreira, N. & Meireles, L. Vital-Jacket®: A wearable wireless vital signs monitor for patients' mobility in cardiology and sports. *2010 4th International Conference On Pervasive Computing Technologies For Healthcare*. pp. 1-2 (2010)
19. E. Spilmont, D. Dias, and J.P.S. Cunha, "AmbiUnit: A new adaptive wearable environmental monitoring device for hazardous professions", in Inforum2018, Coimbra, 2018.
20. Li, Q.; Stankovic, J.; Hanson, M.; Barth, A.; Lach, J. and Zhou, G. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. *Proceedings - 2009 6th International Workshop on Wearable and Implantable Body Sensor Networks, BSN 2009*, pp. 138-143 (2009).
21. O que são contêineres? | Google Cloud <https://cloud.google.com/learn/what-are-containers?hl=pt-br> Last accessed 12 June 2023
22. What Is a Container? | Docker <https://www.docker.com/resources/what-container/>; Last accessed 12 June 2023
23. Koziolok, H., Grüner, S. & Rückert, J. A Comparison of MQTT Brokers for Distributed IoT Edge Computing. *Software Architecture*. pp. 352-368 (2020)
24. Hassan, A. A Decentralized MQTT Broker for IoT on Web 3.0. *Medium*. (2022,1)
25. Dix, P. Why Time Series Matters for Metrics, Real-Time Analytics and Sensor Data. (https://get.influxdata.com/rs/972-GDU-533/images/why_time_series.pdf,2021,7)
26. , Grafana: The open observability platform, <https://grafana.com/>; Last accessed 12 June 2023
27. About : Node-RED, <https://nodered.org/about/>, Last accessed 12 June 2023
28. Ferencz, K. & Domokos, J. Rapid Prototyping of IoT Applications for the Industry. (2020,5), Pages: 6
29. A. Dias, J. Almeida, A. Oliveira, T. Santos, A. Martins and E. Silva, "Unmanned Aerial Vehicle for Wind-Turbine Inspection. Next Step: Offshore," OCEANS 2022, Hampton Roads, Hampton Roads, VA, USA, 2022, pp. 1-6, doi: 10.1109/OCEANS47191.2022.9977308.
30. Silva, P., A. Dias, A. Pires, T. Santos, A. Amaral, P. Rodrigues, J. Almeida, and E. Silva. "3D Reconstruction of historical sites using an UAV." In *Proceedings of the CLAWAR*. 2020
31. A. Dias et al., "ROSM - Robotic Oil Spill Mitigations," OCEANS 2019 - Marseille, Marseille, France, 2019, pp. 1-7, doi: 10.1109/OCEANSE.2019.8867213.
32. Azevedo, F.; Dias, A.; Almeida, J.; Oliveira, A.; Ferreira, A.; Santos, T.; Martins, A.; Silva, E. LiDAR-Based Real-Time Detection and Modeling of Power Lines for Unmanned Aerial Vehicles. *Sensors* 2019, 19, 1812. <https://doi.org/10.3390/s19081812>