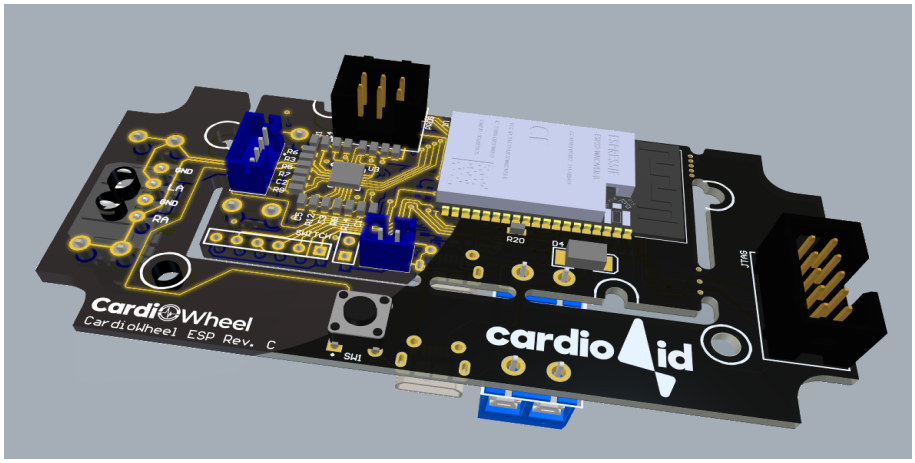
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Electrónica e Telecomunicações e de Computadores



Reliability and Security in Wellbeing Monitoring Embedded Systems

José Filipe Cruz dos Santos

(Licenciado em Engenharia Informática e de Computadores)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadores : Doutor André Ribeiro Lourenço
Doutor Tiago Miguel Braga da Silva Dias

Júri:

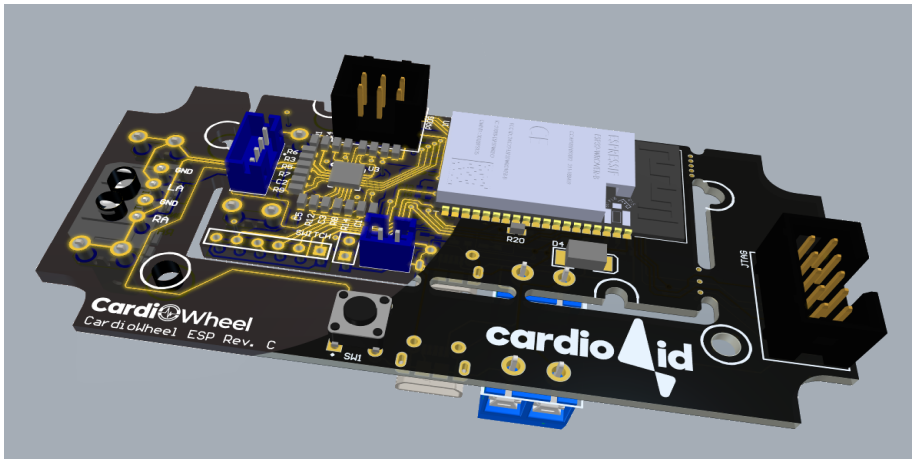
Presidente: Doutor José Manuel de Campos Lages Garcia Simão

Vogais: Mestre Pedro Miguel Fernandes Sampaio
Doutor André Ribeiro Lourenço

Dezembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Electrónica e Telecomunicações e de Computadores



Reliability and Security in Wellbeing Monitoring Embedded Systems

José Filipe Cruz dos Santos

(Licenciado em Engenharia Informática e de Computadores)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadores : Doutor André Ribeiro Lourenço
Doutor Tiago Miguel Braga da Silva Dias

Júri:

Presidente: Doutor José Manuel de Campos Lages Garcia Simão

Vogais: Mestre Pedro Miguel Fernandes Sampaio
Doutor André Ribeiro Lourenço

Dezembro, 2023

Acknowledgments

É com grande apreço que inicio este agradecimento, reconhecendo o apoio inestimável que recebi ao longo da jornada que culminou na conclusão deste trabalho de mestrado. Primeiramente, desejo expressar a minha mais profunda apreciação aos meus orientadores, Professor Doutor Tiago Dias e Professor Doutor André Lourenço. A vossa orientação neste trabalho e em todo o meu percurso académico foi fundamental para o desenvolvimento deste trabalho. Agradeço pelo vosso comprometimento. O vosso apoio foi essencial para o meu crescimento académico e profissional.

Também gostaria de estender os meus agradecimentos ao Instituto Superior de Engenharia de Lisboa (ISEL) por ter sido a instituição onde pude realizar a minha licenciatura e mestrado. Agradeço por proporcionar as condições e recursos necessários para a concretização deste trabalho, bem como pela qualidade do ensino e das oportunidades fornecidas por esta instituição.

De seguida gostaria de agradecer à CardioId e a todos os meus colegas de lá pela oportunidade incrível de poder trabalhar com algo que genuinamente adoro e pelo privilegio de poder ter feito parte da história do CardioWheel.

Na secção seguinte, não posso deixar de mencionar os meus amigos. As palavras não podem expressar adequadamente a minha gratidão por todo o apoio e amor que me proporcionaram ao longo desta jornada e por isso um obrigado a todos. Um apreço especial ao Almeida, Matos, Daniel, Nazariy, João e Vicente por me ajudarem a espairecer e a esquecer um pouco o trabalho; foi crucial para ter 100% energia para toda a duração deste projeto. Agradecer ainda ao João Bonacho e ao Luís Guerra por disporem sempre do tempo deles para me apoiarem em qualquer questão da tese, e por me apoiarem nos trabalhos e em todo o mestrado.

Gostaria ainda de agradecer aos meus pais por todo o apoio emocional e por garantirem que nada me faltava, para que pudesse ser o mais bem sucedido possível. Sem vocês este trabalho não teria acontecido.

Agradecer ao meu cunhado por tentar sempre garantir o meu bem estar e por estar sempre presente e disposto a ouvir-me falar sobre os meus novos interesses.
Não posso deixar de agradecer à minha namorada que foi um grande apoio emocional neste caminho. Sempre foste uma inspiração a nível académico e pessoal e por isso sempre me conseguiste ajudar em todas as minhas questões. Obrigado por não te cansares de me ouvir a reclamar com frustrações minhas.
Por fim agradecer à minha irmã por ser uma inspiração e um modelo de vida para mim. Foste também uma mentora nesta jornada e sempre que precisei estiveste lá para me ajudar. Agradeço-te por acreditares em mim e por me dares força e energia para atingir todos os objetivos a que me proponho.

Abstract

Recently, the reliability and cybersecurity aspects of embedded systems for critical domains, such as health and automotive, has increased interest in the research community and awareness to the general public. This thesis addresses the modelling and the implementation of a reliable and secure software architecture for an embedded system aimed at the acquisition and processing of physiological signals, in particular the electrocardiogram (ECG). The work focused CardioWheel, an ongoing project developed by CardioID Technologies, targeting health and automotive applications. These domains demand special requirements for safety and security, and serve as a showcase for the proposed architecture.

Accordingly, suitable requirements were first established and the architecture state machine was developed using UML and formally validated using Uppaal and Timed Automata modelling. Then, the threat analysis of the architecture was conducted. Finally, the implementation was realized for an ESP32 microcontroller using the FreeRTOS, the ESP-IDF ecosystem, and specially developed hardware independent layers. The communication layer is based on Bluetooth Low Energy (BLE) and allows the transmission of the data from the end-node to a gateway and finally to the cloud. The system has a Over-The-Air (OTA) component that enables the update of the firmware and the security of this operation was also validated.

The proposed architecture was firstly validated using a laboratory prototype. Then, it was deployed to build a small production series of CardioWheel incorporating validation strategies proposed within the context of the ESCEL KDT Valu3s project. Also, a pre-medical trial was conducted at the Hospital de Santa Marta, confirming the reliability and capabilities of our system against a clinical ground-truth.

Keywords: Critical System, Cyber-physical Systems, Reliability, Cybersecurity

Resumo

Ao longo dos últimos anos, a fiabilidade e a segurança dos sistemas embebidos utilizados em áreas críticas, como a saúde e o sector automóvel, têm suscitado um interesse crescente na comunidade científica e ganho maior consciencialização entre o público em geral. Esta tese aborda a modelação e a implementação de uma arquitetura software fiável e segura para um sistema embebido focado na aquisição e processamento de sinais fisiológicos, em particular o eletrocardiograma (ECG). O trabalho realizado visou o CardioWheel, um projeto em curso desenvolvido pela CardioID Technologies, destinado a aplicações nas áreas da saúde e do automóvel. As particularidades destas áreas quanto aos seus requisitos de segurança e proteção dos utilizadores servem de caso de estudo para mostrar as vantagens da arquitetura desenvolvida.

Assim, no estudo realizado foi feito o levantamento dos requisitos do sistema que foram utilizados para projetar a máquina de estados da arquitetura em UML, a qual foi validada formalmente utilizando a ferramenta Uppaal e o modelo de autómatos finitos temporizados. Também foi feita uma análise de ameaças à arquitetura para validar os aspetos relacionados com a segurança. A arquitetura foi desenvolvida para microcontroladores ESP32 usando o ecossistema ESP-IDF e o FreeRTOS, para o que foram consideradas camadas independentes de *hardware*. A camada de comunicação é baseada no protocolo Bluetooth Low Energy (BLE) e permite a transmissão dos dados do nó final para um gateway e, posteriormente, para um servidor na nuvem. A operação de atualização de *firmware* usando o componente Over-The-Air (OTA) foi também implementada e validada quanto à sua segurança.

A arquitetura foi, inicialmente, avaliada e validada usando um protótipo laboratorial. Posteriormente, foi utilizada para realizar uma pequena série de produção do CardioWheel em que se utilizaram as estratégias de validação propostas no contexto do

projeto ESCEL KDT Valu3s. Também foi realizado um ensaio pré-médico no Hospital de Santa Marta usando o CardioWheel com a arquitetura proposta, que permitiu validar a sua fiabilidade e capacidades quando comparado com um eletrocardiógrafo clínico.

Palavras-chave: Sistema crítico, Sistema ciberfísico, Fiabilidade, Cibersegurança

Contents

List of Figures

List of Tables

Acronyms

ADAS Advanced Driver Assistance System. 8

ADC Analog to Digital Converter. xiii, 30, 32, 33, 35, 36, 41, 43, 45, 46, 58

BLE Bluetooth Low Energy. xii, 10, 28, 33, 36, 40, 41, 42, 43, 44

CPS Cyber-Physical System. xi, xiii, xiv, 1, 7, 8, 12, 13, 14, 15, 16, 17, 19, 20, 21, 25, 29, 41, 42, 57

CVD Cardiovascular Disease. 2

DoS Denial of Service. 14

ECG Electrocardiogram. xiv, 2, 3, 28, 36, 44, 45, 52, 57, 58

ECU Engine Control Unit. 8

IC Integrated Circuit. 13

ICD Implantable Cardioverter-Defibrillator. 19

IMU Inertial Measuring Unit. xiii, 28, 33, 34, 36, 42, 44, 45

IoT Internet of Things. xiii, 1, 9, 10, 21

LPWAN Low Power Wide Area Network. 11

LTE Long Term Evolution. 10

MCU Microcontroller. 1, 3, 4, 30, 32, 41, 42, 43, 58

MITM Man-In-The-Middle. 14, 15, 40

MTU Maximum transmission unit. 11

NBIoT Narrowband IoT. 10, 11
NCD NonCommunicable Disease. 2

OSI Open Systems Interconnection. 11
OTA Over-The-Air. 8, 28, 29, 40, 42, 44, 45

PCB Printed Circuit Board. 13

RAM Random Access Memory. 43
ROM Read-only Memory. 43
RTC Real Time Clock. 43
RTOS Real-Time Operating System. 30, 41

SCA Side-channel Attack. 13
SDR Software-Defined Radio. 14
SIG Special Interest Group. 10

TA Timed Automata. xiv, 47, 48, 49, 50

UML Unified Modeling Language. 16, 20
UMTS Universal Mobile Telecommunications System. 10

WHO World Health Organization. 2

1

Introduction

The Embedded Systems market is ever-growing (Figure 1.1), especially since the introduction of smartphones creating a competitive and lively market of powerful microprocessors and Microcontrollers (MCU). These extend to a myriad of industries and also everyday-life use cases, such as home automation, wellbeing and health applications. The conjunction of this and the growth of the Internet has introduced the so-called industry of Internet of Things (IoT). This industry combines the development of Embedded Systems with Internet connectivity, to create smart automations based on sensors and actuators, as well as remote control capabilities. These types of devices are also known as Cyber-Physical Systems (CPS).

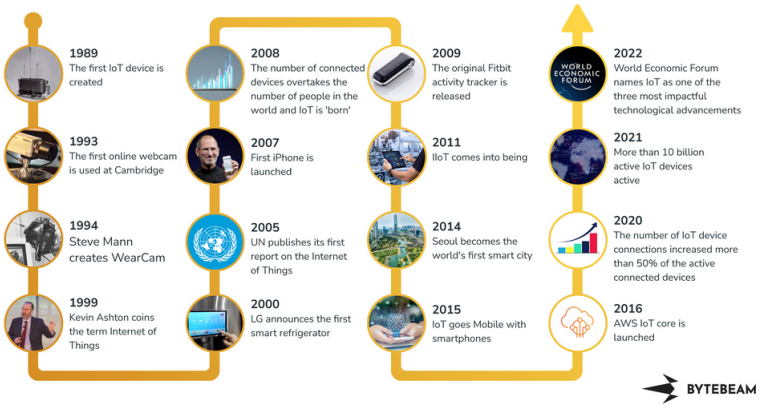


Figure 1.1: Internet of Things History

(extracted from https://bytebeam.io/)

The rise in usage of such systems raises some concerns when it comes to the side of the industry that uses them for crucial operations that have little to no margin for error. On one side Embedded Systems can be great for those applications because of their capability of reproducible and repeatable actions to high degrees of precision, without fatigue but, on the other hand, we need to guarantee that the system is trustable and is fit enough for those tasks because of their criticality, giving a start to another branch of this industry, Critical Embedded Systems. The need for these types of systems stems from the necessity of implementing Embedded Systems in industries like the military, medical, automotive or aeronautics for example, where the processing unit performs critical actions from radiation level control in X-Ray machines, for the medical industry, to missile trajectory planning, for the military industry. When these technologies are correctly applied, they have the potential to enhance these industries by facilitating highly specialized and demanding functions.

In the medical field and according to the World Health Organization, NonCommunicable Diseases (NCD) were the leading causes of death globally in 2019, with Cardiovascular Diseases (CVD) accounting for 38% of these deaths [20]. Early diagnosis of these types of diseases can lead to a better quality of life with, in some cases, preventing early death. In addition to NCDs, road accidents are one of the top 20 leading causes of death in the world as well as a significant cause of non-fatal injuries, many of which result in disability of various degrees of severity.

To address these issues, CardioID Technologies [1] uses Electrocardiogram data obtained from an Embedded System to help doctors diagnose CVD and to measure drowsiness in driving scenarios. The Electrocardiogram data acquired can provide valuable insights to help with both of these problems. Developing a reliable and cyber-secure software system is important in these types of fields, particularly when it comes to handling sensitive data. This is especially true in the healthcare industry, where accurate and secure data is crucial for the diagnosis and treatment of diseases such as NCD and CVD.

The purpose of this work is to develop a software architecture that is implemented on a system that runs on an end node device and measures physiologic signals sending them to be processed by a gateway where they will be processed without compromising on reliability and cyber-security.

This chapter presents the context in which this thesis is placed and the objectives of the proposed work.

[1] www.cardio-id.com

1.1 Context

As previously mentioned, critical embedded systems are key in present and future applications, and ensuring that their integration in real-world applications is reliable, secure and context-aware is crucial. This becomes even more important in critical scenarios, for example when we are monitoring people vital signs and processing them with the intent of evaluating the person's health and wellbeing.

This work is contextualized in the developments of CardioID Technologies embedded systems, in particular in CardioWheel which is a system that acquires the ECG from the driver's hands to continuously detect drowsiness, cardiac health problems, and perform biometric identity recognition. It is composed of an embedded system with a custom analog front-end, which measures the ECG signal, and a computing unit that performs signal processing and sends information to a Gateway/Smartphone using Bluetooth Low Energy (BLE). The system also contains sensors like an accelerometer and a gyroscope that provide crucial information regarding movement and orientation, allowing it to perceive whether the person is in motion or changing position and thus for a better understanding and processing of the biometric data that is obtained, eventually helping detecting disparities in the signal, like for example motion artifacts, that can negatively impact the algorithms that test for drowsiness, heart health diagnosis or biometric recognition.

CardioWheel must also be reliable and secure. We can measure and determine how reliable or secure a given system is, but it can be difficult to define the intended level of each metric without proper methods and adequate tools. This work focuses on ensuring that systems like CardioWheel can effectively and reliably serve their purpose within critical scenarios. To accomplish this, we formally verified the software architecture and also the prototype implementation. The prototype was verified and validated not only through basic unit and integration testing but also as a subject in the project VALU3S [5] and in a pre-medical trial in Hospital de Santa Marta.

1.2 Objectives

The main goal of this thesis was to develop an embedded software architecture that is not only Microcontroller independent but also reliable, and secure. In addition, this thesis aimed at the study and the implementation of encryption techniques that enable the secure exchange of private data between an end-node and a gateway, guaranteeing

identification, authentication, authorization, confidentiality, data integrity, and non-repudiation.

Furthermore, the proposed architecture was designed to comply with the requirements of the medical and automotive industry, since its intended use is in an environment like a car to help with drowsiness detection or in the medical domain to help with the diagnosis of patients.

Finally, the devised architecture and security measures were implemented in an ESP32 Microcontroller using the Espressif Systems Platform specifically to build a prototype of a newer version of CardioWheel.

1.3 Contributions

The work conducted in the scope of this thesis has resulted in some important contributions in several key areas. First and foremost, it enabled the integration of CardioWheel into the VALU3S project, serving as a compelling use case. This integration exemplifies the practical application of CardioWheel in real-world scenarios, validating the system in the process. To do this, a dedicated CardioWheel Validation Station was created. This station is depicted in Figure 5.3 and was designed to facilitate rigorous testing and validation processes, ensuring the robustness and reliability of the CardioWheel system.

Part of this work is discussed in a scientific publication presented at the 43^{rd} IEEE Real-Time Systems Symposium (RTSS 2022) [51], which introduces the MARS tool as a tool that aims to ease the process of safely deploying run-time verification monitors on embedded devices to test the safety and security of devices that manipulate sensitive biometric data

The utilization of CardioWheel in a pre-clinical trial at the Hospital de Santa Marta, with formal approval from the ethics committee, is another important outcome of this thesis, since it represents a crucial step in validating the proposed architecture in a real world use case compliant with ethical standards.

Finally, the publication of another scientific article in the 2023 Portuguese informatics scientific symposium (Inforum 2023) [59], contributes to the scholarly discourse. This dissemination of findings allowed to enhance the visibility and recognition of the research within the academic community.

Collectively, these contributions mark a meaningful advancement in the understanding and practical application of CardioWheel in the health and automotive critical domains.

1.4 Document structure

This document is organized into five chapters containing each of its relevant sections and subsections, the first chapter (1) is the introduction of the problematic with its context, as well as the objectives for this thesis. The second chapter (2) presents the core knowledge inherently necessary to have a frame of reference to the discussion presented forward. The third chapter (3) presents the state of the art surrounding the problem. The fourth chapter (4) aims to discuss what is the proposed architecture and what are the defined requirements both for software and hardware, it presents a prototype created following the architectural needs and the defined requirements. Lastly the fifth chapter (5) enumerates the various methods used to verify and validate the architecture and the created prototype.

2

Background

A CPS is a computational system composed of computational elements that collaboratively work towards managing or controlling a physical system, these types of systems are present in various industries and are the central unit for numerous concepts and applications. As represented by Figure 2.1, to guarantee what is mentioned above we need to create a secure and reliable channel through which the device communicates to the Internet.

This chapter will provide a clear and concise overview of the research topics and the technologies presently available to create a reliable and secure CPS when reading, sending and receiving critical data. These topics will consist of the exploration of available wireless technologies and their characteristics and how fit they are for a critical CPS, the general notion to model a reliable CPS.

Figure 2.1: Reliable and secure Cyber-Physical System

CardioWheel is being designed so that it can be considered a reliable and secure CPS

for the health domain developed for the ecosystem of the CardioID Technology company. This ecosystem comprises three computing elements as shown in Figure 2.2. The end node is a CPS responsible for the acquisition (and preprocessing) of electrocardiographic (ECG) signals. At the fog level, a gateway collects the resulting sensitive data and securely sends it to a cloud server that can process it for several different applications, such as healthcare, automotive, biometrics and cyber security, or human factors engineering. This includes using the data also to re-train Machine Learning models. Furthermore, the end node is managed by a service from the cloud server through the gateway.

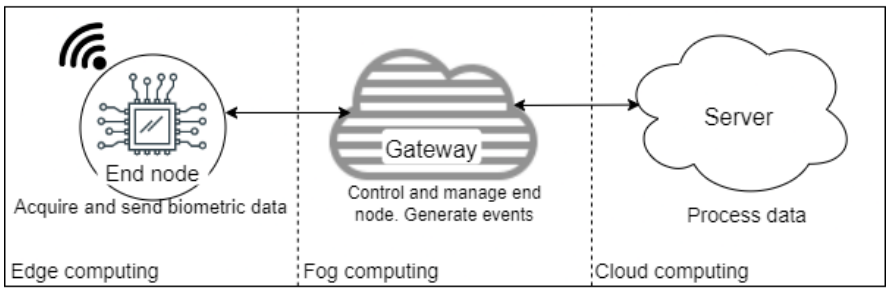


Figure 2.2: CardioID Ecosystem

The end node is intended to respond to two concurrent use cases: the acquisition, processing, and transmission of ECG signal; and the reception and effectivation of Over-The-Air (OTA) firmware updates. These use cases guide the development process to define several tasks dedicated to responding to each of the steps needed to implement the use cases and the definition of a requirements table, listing the system properties that must be observed through the correct interplay between these various tasks.

In this work, the focus is on the end node portion of the CardioID ecosystem, comprising two separate types of CPS - the CardioWheel and Sensitee (also known as Movesense by Suunto™).

The first of these devices is CardioWheel(Figure 2.3a), developed by CardioID is an Advanced Driver Assistance System (ADAS) designed to enhance vehicle and driver safety. It can biometrically identify the driver and produce fatigue alerts [47], [14]. The system is composed of software and hardware in the form of a steering wheel with conductive leather, and it continuously acquires ECG from the driver's hands. It has been integrated with other sensors, telematic solutions and connected to the car's Engine Control Unit (ECU), in an integrated solution entitled CardioID ADAS Services - an all-in-one solution that monitors the driver and driving process in a 360°perspective, making vehicles and drivers more secure.

The second device is Sensitee (Figure 2.3b), which involves a smart t-shirt featuring

conductive textiles as a dry electrode and an embedded sensor. Sensitee is designed for continuous biometric identification and monitoring of individuals. It specifically focuses on tracking fatigue levels and detecting abnormal changes in the individual's heart signal.

(a) CardioWheel

(b) Sensitee

2.1 Wireless Communication

Considering that we are talking about a connected embedded device, there are some aspects to consider when choosing a wireless communication protocol or technology. This starts by defining the systems requirements like power consumption, range, data throughput, and, in some cases, considering local laws.

There are wireless technologies more fit than others for the various types of requirements and in this chapter, we will discuss most of the technologies available that are relevant for IoT devices.

Wireless technologies, in general, are chosen for Embedded Systems because they are often in remote or difficult-to-access locations without significant infrastructure, like roads or forests; they can also be chosen in more urban settings where we are trying to accomplish the same solution for a problem without the need for power or data cables to be installed.

In the table below 2.1 we explore a comparative view of some of the technologies present and relevant to this application. For this comparison the parameters chosen as relevant were range, frequency range, maximum throughput regards the maximum

theoretical speed for data transmission allowed by the technology, maximum transmission unit which is the maximum size of the packets sent, qualitative power consumption and cost. The weight imposed by each of these metrics is relative to the requirements of a specific project.

Table 2.1: Wireless technologies

Technology	**Maximum Range (in meters)**	**Operating Frequency Range**	**Maximum Throughput**	**Maximum Transmission Unit (MTU) in bytes**	Power Consumption	**Cost**
BLE	100	2.402–2.481 GHz	1 Mbps	512	low	low
ZigBee	100	2.4 GHz, 915 MHz, 868 MHz	250 Kbps, 40 Kbps, 20 Kbps (depending on frequency)	128 (28 header, 100 data)	low	low
IEEE 802.11 (Wi-Fi)	50	2.400–2.500 GHz, 4.915–5.825 GHz	300 Mpbs or 1.7 Gbps (depending on frequency)	1500	high	moderate
LoRa	15000	169 MHz, 433 MHz, 868 MHz, 915 MHz	27 Kbps	256	low	low
NBIoT	10000	800-900 MHz	26 Kbps (downlink), 62 Kbps (uplink)	1600	low	high

Bluetooth Low Energy (BLE), one of the prerequisites for the device, as mentioned further in Chapter 4, is that all biometric data must be transmitted via BLE.

BLE, as described by the Bluetooth Special Interest Group [11], is a wireless communication technology that focuses its design for applications with very low power consumption requirements. This makes BLE radio technology suitable for IoT devices powered by a battery. Other than that, the throughput of BLE is high enough for most applications, like audio, but it is not enough for applications with very high throughput needs, like video streaming. Another downside of BLE is its inability to work long distances.

ZigBee or IEEE 802.15.4 from the ZigBee alliance, now called Connectivity Standards Alliance, is a wireless protocol that has low power consumption, with the downside of low throughput [27].

IEEE 802.11 [37] also known as Wi-Fi communication technology (by Wi-Fi Alliance) is one of the biggest wireless communication technologies used in the world, just under telecommunication technologies like LTE and UMTS. This technology is widely used and is in every major city in the world as it provides connection to the Internet. As of generation 5 of Wi-Fi, this technology is usually disregarded for IoT devices that

require low power consumption, this is because Wi-Fi has a very high consumption compared to other technologies, another factor that can deter its use is the fact that by design it is not a long-range technology requiring an access point to be close to enable communication. With that being said Wi-Fi is appropriate for usages where we need to send big payloads having a Maximum transmission unit of 2304B, Wi-Fi only controls the first two layers of the OSI model [1] and allows for payloads to carry a lot of different network, transport and application protocols like, for example, HTTP and MQTT that use TCP/IP as the transport layer. This type of flexibility promotes the usage of more robust protocols to exchange data which can be desirable for the security of the communication.

It's important to mention that there is another standard by the Wi-Fi Alliance the IEEE 802.11ah also known as HaLow that operates in a sub 1 GHz spectrum (850-950 MHz) and is an equally capable communication technology thought specifically for low-power devices providing, at the same time, a longer range.

LoRa (Long Range), similarly to HaLow Lora works on the sub 1GHz spectrum working with frequency ranges of 169 MHz, 433 MHz, 868 MHz and 915 MHz (the usage differs from country to country), this gives it the capability of accomplishing very long ranges [46], being a Low Power Wide Area Network or LPWAN. With LoRa in particular, there is some legislation to follow that differs across continents and countries that limit not only the transmitted power in a given frequency range but also the allowed throughput. Another limiting factor is that LoRaWAN, the version of LoRa that allows a device to be connected to the internet, has a star topology which means that each node needs to communicate through a gateway adding to the cost of the solution, there is also a less used mesh topology that solves this problem with the downside of difficulting the connection of the device to the Internet.

Narrowband IoT (NBIoT) is a LPWAN wireless technology, providing low power, long range and remarkable penetration capabilities. NBIoT is rooted in telecommunication standards which have the advantage of being smoothly integrated with cellular systems, and being easily scalable.

However, there are some disadvantages to take into consideration, being the lower data throughput and higher latency compared to other solutions, it's also more expensive to implement requiring contracts to be made with the mobile network operators that offer such a service.

[1] OSI: is a model that divides a network communication over a network in seven distinct layers

2.2 Security in a Cyber-Physical System

Security in Cyber-Physical Systems is a very broad subject, covering a lot of areas of computer, software and telecommunication engineering, this makes it very difficult to create a truly secure system unless we specify as much detail as possible of the system in the first phase of development and have a multidisciplinary team helping with the development phase. The first phase of development also known as the design phase, requires a type of modelling known as threat modelling. With this type of proactive approach, we ensure that we are always up to date and are able to respond appropriately to cyberattacks. Threat modelling focuses on defining an organization or system architecture, pinpointing entry points, attack vectors and possible mitigation strategies. For this type of approach to work we first need to define what is the asset we are trying to protect, we do this to ensure we don't waste resources on protecting things that don't need protection, after this process we implement the mitigation techniques discovered in the previous step. In the end we should have a system capable of withstanding attacks on the discovered vulnerabilities. It's important to take into consideration that the cybersecurity space is always evolving with new vulnerabilities being discovered every day and because of this the process of creating and managing a system that is cybersecure is continuous, having to take into account that patches, fixes and updates are going to be necessary eventually.

When conducting threat modelling for a CPS we need to take into consideration two critical components: the asset to be protected and the level of protection desired. CPS's can be vulnerable to remote exploits and physical exploits and a common approach to threat modelling such a system is by using the STRIDE method [25]. STRIDE provides a tested and proven methodology to help identify possible threats allowing for ease of design of a system that meets all of the security requirements (Table 2.2).

To secure the CPS some measures will be put in place; firstly we will use the STRIDE method (Table 2.3) to model the threat vulnerabilities of our system and find the appropriate solutions to address those possible threats.

Secondly, taking into consideration the first step, we will characterize the data being outputted by and inputted into the device, with this characterization we will be able to define the best algorithm to encrypt it. Later implementing this into the overall software that will be developed that supports the transmission and receiving of data.

Table 2.2: STRIDE method

Threat	**Property Violated**	**Definition**
Spoofing	Authentication	Impersonation of something or someone
Tampering	Integrity	Modifying data or code
Repudiation	Non-repudiation	Claiming to have not performed an action.
Information Disclosure	Confidentiality	Exposing information to someone not authorized
Denial of Service	Availability	Quality of service degradation or service denial
Elevation of Privilege	Authorization	Gain access or elevated capabilities without authorization

A physical attack to a critical CPS can consist of Hardware Trojans which consists of a manufacturer planting a vulnerability or remote access into the hardware, this alteration can be performed at different stages of development and manufacturing including design, fabrication and packaging of the Integrated Circuits (IC) [34], Side-channel Attacks that aim to exploit the information obtained from a system while it performs an algorithm. This can be based on data like execution time (ex. rowhammer attacks[52], timing-based attacks on cryptographic algorithms, etc.), supply current consumed by the device (ex. deducing by the power consumption which algorithm is being used and it's key size, etc.) and electromagnetic leakage (ex. Van Eck phreaking [63], etc.) [60] this types of attacks are becoming very sophisticated [24] and can even be non-profiled attacks meaning that they are performed without the knowledge or details on the implementation of the system, there is also a risk of fault injection attacks (ex power or clock glitches, JTAG fault injections [1], etc.) [7] which induce faults in the execution of an application with the goal of changing memory values or bypassing instructions gaining access to unauthorized to sensitive information.

To protect against hardware attacks, a common approach can be by using a trusted cryptographic computational unit like ARM Trusted Zone or Intel SGX, or even external modules like the Trusted Platform Module. These technologies create a Trusted Execution Environment where sensitive applications can run isolated from potential attacks, another crude way of protecting against most physical attacks can be by not exposing interfaces in the Printed Circuit Board (PCB) or by sealing the access to the

Table 2.3: STRIDE method

Threat	**Property Violated**	**Definition**	**Example**
Spoofing	Authentication	Impersonation of something or someone	Pretend to be the device receiving ECG data
Tampering	Integrity	Modifying data or code	Modify firmware data when downloading or changing ECG data to induce errors in the diagnosis
Repudiation	Non-repudiation	Claiming to have not performed an action	Send false data and pretending to be a device, either ECG from the CPS or firmware from the external device (ex. gateway, phone)
Information Disclosure	Confidentiality	Exposing information to someone not authorized	Gain access to device information like UUIDs and tokens
Denial of Service	Availability	Quality of service degradation or service denial	Induce crashes on the device so it cant be used to acquire ECG data
Elevation of Privilege	Authorization	Gain access or elevated capabilities without authorization	Produce, without permission, firmware updates or start data acquisition

microprocessor altogether with materials like epoxy resin.

A remote exploit is one that explores vulnerabilities in software implementations and/or wireless protocols used, these can usually be Man-In-The-Middle (MITM) attacks which are aimed at exploring techniques that allow to intercept information either through means of physically wiring into the infrastructure or by wirelessly listening to data using, for example, a SDR [40]. MITM attacks can capture valuable information to be used in attacks like relay attacks and replay attacks, where the data captured is manipulated and sent back to the device. Denial of Service attacks purpose is to degrade the quality of service or take down a service this type of attack redirects a lot of traffic to a specific device until it no longer has the capability of withstanding it and either shutdowns or slows down considerably. There are buffer overflow attacks that explore bad verification of input data to write in prohibited memory zones, this type of attacks can

help an attacker remotely execute code or to run arbitrary code that's already on the device, this type of attacks are often also used to help get secret information that is stored on the device. To mitigate this type of threats we need to implement some counter measures like data encryption, and strict input policies, this helps mitigate this type of attacks because with strong data encryption, even if it's possible to perform a MITM attack, all of the data obtained would be unreadable.

The system we are building requires resilience against remote exploits, this means that we will need to find the appropriate cryptographic algorithms for the type of data we are trying to encrypt. When choosing a cryptographic algorithm there are some things to consider, the first one is that the system we are developing is a CPS based on a microcontroller, these often have less computational and memory resources, while also needing to consume less battery. With this we narrow our cryptographic algorithm choice to less computationally intensive algorithms and ones that do not increase a lot our encrypted payload to help reduce memory consumption and transmission time, reducing in turn the power consumption as well.

There are two major types of cryptographic algorithms (Figure 2.4) symmetric key and asymmetric key, the big difference between them stands on the type of key used, the symmetric key is when an algorithm uses both the same key to encrypt and decrypt whereas the asymmetric key has different keys for both purposes[36]. There is also a mode distinction in symmetric algorithms these can be a block cipher or a stream cipher algorithm. Block cipher mode converts a block of data directly, whilst the stream cipher mode converts a block of data by groups of bytes, the last mode is useful for situations where we can't afford to save large blocks of data at a time, so by encrypting smaller blocks we can send them and free the occupied space in memory.

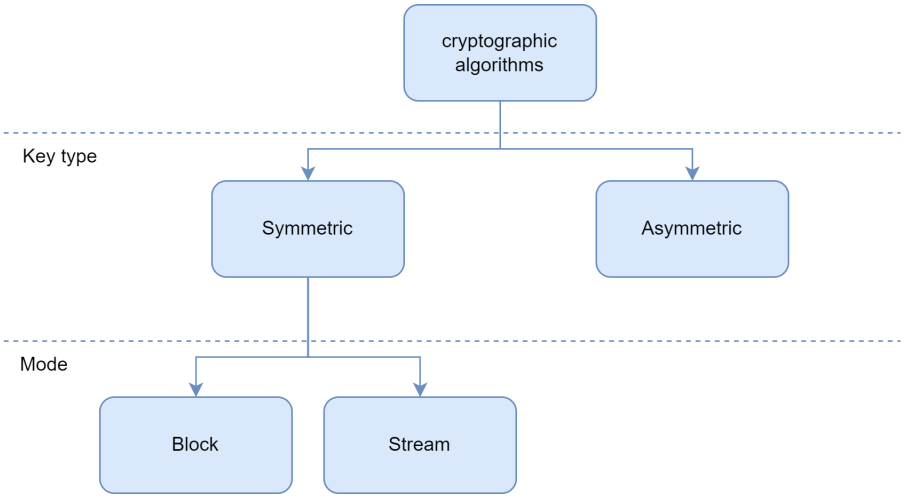


Figure 2.4: Cryptographic Algorithm types

A problem that arises from encryption is the problem of key sharing, to be able to encrypt and decrypt there is some information that needs to be exchanged between the receiver and emitter and to accomplish that there are already some paradigms and algorithms defined.

Another aspects to take into consideration are the data characterization, data type and how repeatable it is, since they are determining factors when choosing an encryption algorithm. This is true especially for data that repeats, because when the output is consistently the same for every input, this may pose a risk by potentially revealing information about the data being transmitted at any given moment. Other aspect to consider is the data packets size if these are characterizable and are different for different types of packets, an attacker is able to distinguish between these packet sizes, revealing information about the data being sent and the system workflow or state, which poses a significant security threat.

2.3 Modelling a reliable Cyber-Physical System

In computer science, the modelling of a system usually follows the following steps, firstly we start by defining the requisites of the system and its functionality, this part of the modelling process is commonly done with the extensive description of the use cases, after that, its defined the logic architecture where we use modelling tools like Unified Modeling Language (UML) to create, for example, the domain model and sequence diagrams needed for the chosen use cases. Finally, the implementation phase comes after defining all the components needed and completing the deployment diagram. In the implementation phase, our focus must be on following the previously defined models and programming best practices, because with that the result will be a maintainable and modular code with an extended lifetime [15].

This model approach is not fit enough to design a reliable Embedded System, for that we need to guarantee that all the following three elements are reliable, the hardware, the firmware and the software of the system. Moreover, we need to use another type of tools and processes to model systems with high-reliability requirements.

As described by Cambridge University [13] reliability is

> the quality of being able to be trusted or believed because of working or behaving well

Concretely this means that reliability is a fitness measure that usually is represented by a percentage value, a higher percentage represents a better suitability of the system to

its requirements. To achieve the high reliability required by the type of system we are going to design, we need more advanced modelling approaches.

This type of advanced approaches consist in the definition of requirements, modelling, implementation and testing. Contrary to what may be inferred at first glance using apparently more steps to accomplish the desired software system, can shorten developing time, improve code reliability and its future maintainability.

The mentioned methods of developing a software system are called formal methods, these types of methods involve using techniques that model software systems as mathematical entities. Adhering to these principles has numerous benefits, including increasing reliability and security, and improving the maintainability and understandability of the code [56].

There are many different formal methods as surveyed in [54] that can be used through modelling tools to model software systems. Tools like Uppaal and Event-B, although different, they can help ensure that the model is correct and accurately reflects the desired behavior of the system. By using this rigorous mathematical modelling approach, it is possible to verify the model's reliability and design a system that is more likely to function as intended without relying on empirical testing which can often be biased, inaccurate and tedious.

3

State Of The Art

When it comes to the medical industry, in the last years there has been a growing increase in the adoption of CPSs to assist in the diagnosis of several illnesses or even proactively saving lives. Implantable devices such as the pacemaker or Implantable Cardioverter-Defibrillator (ICD), or chest compression devices like LUCAS [48] (3.1) are well-known examples of medical embedded systems. These systems requirements include high reliability and security [57], especially the most recent versions that already use wireless technologies [65]. Otherwise, the risk of severely injuring people or losing lives greatly increases. Two very unfortunate examples of such disasters are [49] and [45], which even with a lot of testing, had major failures that resulted in fatalities. Consequently, these types of catastrophes taught a very valuable lesson: testing by itself is not enough to ensure a system is reliable, reliability must be embedded into the model of the system.

In this chapter, we will be firstly exploring modeling approaches and formal methods used in relevant medical and critical applications, and will secondly be talking about the cryptographic algorithms most used for lightweight embedded applications.

The goal is to contextualize the problematic in the present already developed work.

Figure 3.1: Chest compression device

(extracted from [48])

3.1 Modeling Approaches

Regarding the modeling of a critical CPS, significant efforts are being made due to the emerging use of such systems in industries that require a high reliability factor. When it comes to modeling a safety critical system there are proposed works that use Petri Nets [50] as a formal modeling workflow to accomplish a reliable model for such systems, this type of modeling can be used instead of UML Activity diagrams, but both are very similar, with UML Activity diagrams being arguably more fit to model more dynamic systems [28].

There are a large number of reliability models present that are relevant for different applications [41], including medical critical CPS such as pacemakers. The reference [55] introduces a system of this type and the notion of formal methods is mentioned as a key component to model a system with this reliability requirements. In [64] there is a more concrete view on what types of tools exist to implement formal methods, using mCRL2 and Uppaal to model the firmware of a pacemaker. The work executed at [22] explores formal methods and tools applied to medical devices which are critical CPSs. In this paper, the model checker tool used is Uppaal and in contrast to the first mentioned paper uses UML Activity diagram when modeling the workflow. The workflow modeled remains highly pertinent and relevant by current standards as it models a system that contains multiple tasks of processing. Furthermore deeper its also explored the conversion process of UML Activity diagram into the model readable by the Uppaal verification tool. In conclusion, the verification and modeling process was considered vital to prematurely detect and correct errors in the workflow as well as a great method to prevent code bugs to be written into the software. CardioWheel modeling should follow a equivalent methodology to allow for the development of a

reliable system.

3.2 Cryptographic Algorithms

Although there are great efforts being made to create and establish weak points and find vulnerabilities on CPSs there is today still a gap when it comes to studying the art of protecting such systems, this becomes even more critical in an industry that is starting to rely more on these type of technologies like the medical, automotive and aeronautics industries for example.

Even with this gap, there are studies made to tackle the issue of encryption determining what are the more lightweight and suitable algorithms for encrypting data in the context of CPSs, although there is definitely work to be done when it comes to choosing an algorithm based on the type of data being encrypted like the one in this project and it's particular sensitive nature still raises some concerns for the project. With that being said the consensus found in several of the articles read is that there are a group of algorithms that stand out either because they are less computationally extensive or consume less memory or power resources. These algorithms are RSA, elliptic curve, AES, DES, RECTANGLE and some lighter weight and derived variations like PRESENT, DESL/DESLX and CLEFIA for example. In [3][30] the major focus of the study is the algorithms performance when it comes to memory usage, power usage and computational performance. This is great for IoT devices but there is a factor that needs more exploration, which is security performance. Some of these algorithms are also considered outdated like DES which has been decommissioned by NIST, the algorithm in itself isn't insecure but it uses 56 bit key which is considered too short for modern standards. In other resources found, especially on [35], it is studied more in depth the difference between symmetric and asymmetric ciphers, which brings a fundamental and relevant subject to the work. The fundamental difference between symmetric and asymmetric ciphers is the type of key used, the first type of cipher uses the same key in both the emitter and receiver 3.2 and the latter uses a different key on each side of the communication 3.3. Another aspect to explore is the key exchange, which are some proposed ways of making sure that both the emitter and receiver have a way of knowing what is the key so encryption and decryption can occur. The first way is by predetermining the key when programming the devices, this is easier but is also insecure because if one device is compromised the key can't be easily changed and every piece of data can now be decrypted, the other solution is by using key exchange algorithms, which are often more computationally and memory intensive algorithms but

are also fundamental to improve the security of a system, the algorithms use a series of mathematical formulas to exchange keys, these usually are either Diffie-Hellman or RSA. The key aspect of a cipher as described by Kerckhoffs's Principle [1] is the most important one to ensure true security.

As CardioWheel is a system that pretends to acquire biometric data, the cryptographic algorithm chosen will not need to account for that much repeatability in the data given the entropy naturally introduced by the human body. It should also consider power consumption, as the system was conceptualized to work standalone with only a wireless connection to stream the data, and memory and time resources used to allow for the system to, while sending the data, be able to still acquire and process the biometric data.

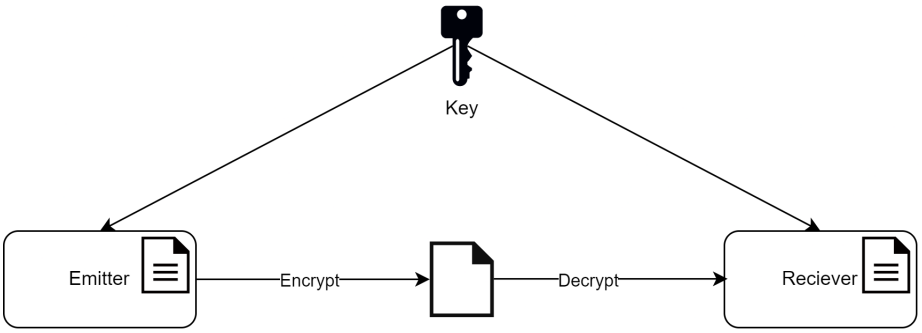


Figure 3.2: Symmetric cipher

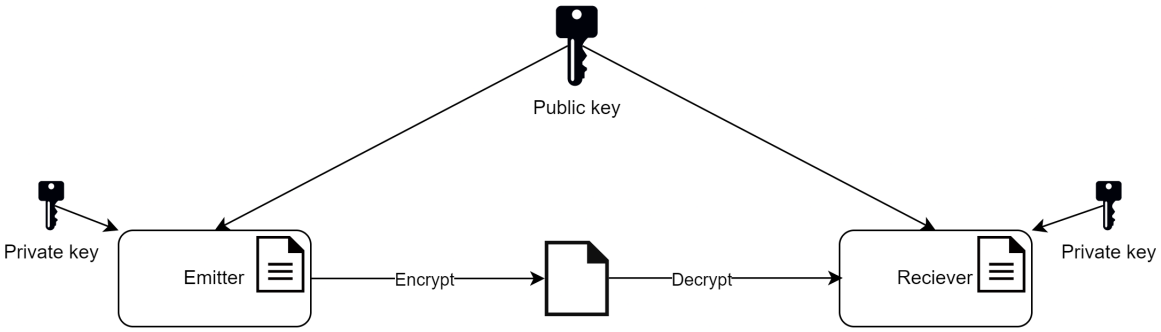


Figure 3.3: Asymmetric cipher

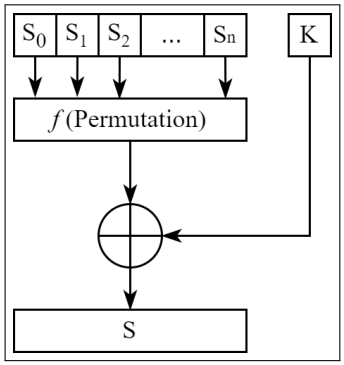
When it comes to describing cryptographic algorithms we need to consider five important aspects, firstly we need to describe it by the cipher type, block cipher or stream cipher, there are also some algorithms that don't fit in either category being that they have characteristics from both. The second characteristic is the structure, cryptographic

[1] Kerckhoffs's Principle: a cryptosystem should be secure even if everything about the system, except the key, is public knowledge

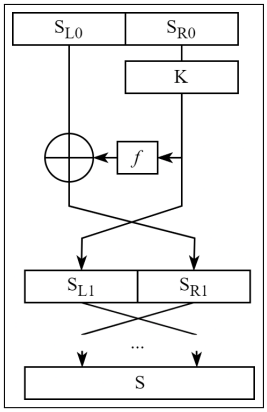
algorithms can be structured in many different ways, the different structures define the general flow of operations applied to the data to encrypt it.

The existing and relevant structures available are the following:

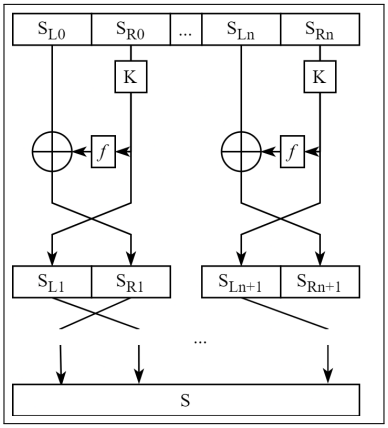
- **Substitution-Permutation network (SPN)** - a series of permutations evolving mathematical operations, these are used for block ciphers (Figure 3.4a).
- **Feistel Network (FN)** - this structure divides the data block in two applying diffusion to one of them and, in the end, XORing both of the blocks together, repeating this process a defined amount of rounds (Figure 3.4b).
- **Generalized Feistel Network (GFN)** - it's an extension of the Feistel Network, where the data is divided into multiple pairs of data instead of only one, the encryption process is the same as for the simpler Feistel Network for each pair (Figure 3.4c).
- **Addition Rotation XOR (ARX)** - this structure is applied to block ciphers and it encrypts data based on add, rotate and XOR operations, it's very fast, compact and easy to implement but it raises some concerns when it comes to its resilience to side channel attacks and simple cryptanalysis attacks (Figure 3.4d).
- **Nonlinear-Feedback Shift Register (NLFSR)** - although mostly used for stream ciphers it can be used on block ciphers, this structure has a nonlinear function that at each step returns a new value computed from the previous input (Figure 3.4e).
- **Hybrid** - a hybrid structure uses multiple structures in one algorithm if well implemented this can be great for adding security to the algorithm.



(a) Substitution-Permutation network (SPN)

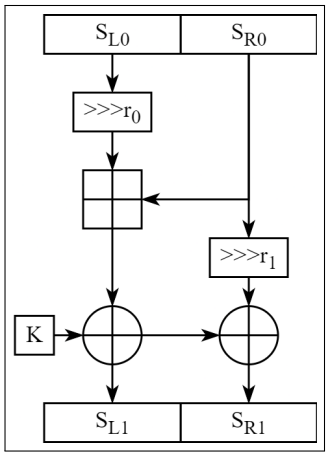


(b) Feistel Network (FN)

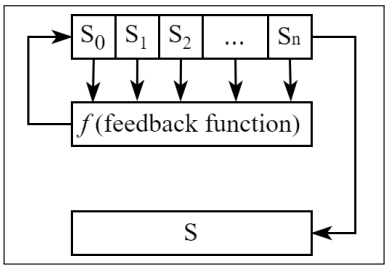


(c) Generalized Feistel Network (GFN)

Figure 3.4: Cryptographic algorithms structure



(d) Addition Rotation XOR (ARX)



(e) Nonlinear-Feedback Shift Register (NLFSR)

Figure 3.4: Cryptographic algorithms structure

Next, we need to characterize the algorithms by three important metrics: key size, block size and the number of rounds. These values give us an insight into the relative level of security each give and a general idea of how much memory they are going to need. The key size is the number of bits used to encrypt and decrypt a given message, the block size applies only to block ciphers and represents the number of bits encrypted at a time, lastly, the number of rounds represents how many times the algorithm is applied until it returns the encrypted message.

Table 3.1 shows a comparison of the previously mentioned metrics for the most relevant lightweight algorithms, providing a comprehensive overview of their characteristics.

Table 3.1: Algorithm software metrics

Algorithm	Type	Structure	Key Size (bits)	Block Size (bits)	Rounds
PRESENT [12]	Block Cipher	SPN	80/128	64	31
NOEKEON [21]	Block Cipher	SPN	128	128	16
GIFT [6]	Block Cipher	SPN	128	64/128	28/40 (depending on block size)
RECTANGLE [17]	Block Cipher	SPN	128	64	25
AES [61]	Block Cipher	SPN	128/192/256	128	10/12/14 (depending on key size)
DESL [43]	Block Cipher	FN	56	64	16
CLEFIA [39]	Block Cipher	GFN	128/192/256	128	18/22/26 (depending on key size)
SPECK [16]	Block Cipher	ARX	64/72/96/128/144/192/256	32/48/4/6/128	22–34 (depending on block and key size)
SPARX [19]	Block Cipher	ARX	64/128	128/256	24/32/40
Halka [18]	Block Cipher	NLFSR	80	64	24
KATAN/KTANTAN [23]	Block Cipher	NLFSR	80	32/48/64	254
Hummingbird-2 [26]	Neither	Hybrid	128	16	4 round start

When it comes to security considerations and as highlighted in [29] Halka, KATAN, KATANTAN and Hummingbird-2 for example have security vulnerabilities that render them unusable for high security demanding purposes, RECTANGLE, GIFT and NOEKEON are alternatives to algorithms like PRESENT with less adoption, reduced security but better overall lightweight application. Finally, algorithms like AES, CLEFIA and PRESENT are considered the more suitable options because of the security level they present and because of their common usage for CPS applications.

4

Proposed Architecture

This chapter approaches the proposed architecture for the generic system and its implementation. The development of an appropriate application architecture significantly impacts performance, scalability, reliability, and usability. Therefore, adhering to established design principles during its development and respective implementation is crucial to accomplish a reliable and secure architecture.

To define the architecture for the modules, a method was used that creates a generic diagram defining the modules needed to accomplish the use cases. Doing so provided a clear and concise view of the functionality needed from each component helping define the concrete functions and properties required.

This process mainly follows the required use cases and the functional and non-functional requirements. Generally speaking and as highlighted in [44] while citing [62], requirements are basically a capability or feature needed to satisfy a contract, standard or specification, in other words, a feature needed to achieve a defined objective. More specifically, that means that the functional requirements consist of the overall functionality performed by the system, whereas the non-functional requirements mainly comprise the performance and reliability requirements of the system. Where performance focuses on the expected yield of the system when it comes to the produced output and reliability refers to its consistency and stability. In that scope, the table below lists the requirements for our project.

In order to understand some of the requirements we need to take into consideration the rationale behind them. Although some of the requirements, like R1.1 and R1.2

Table 4.1: Requirements Table

Ref.	Type	Description	Category
R1	Functional	Device Functionality must be:	
R1.1	Functional	The device must send data through a BLE channel	Mandatory
R1.2	Functional	The device must be able to Update its Firmware OTA	Mandatory
R2	Functional	Device powers-on in an idle state (no data acquisition is in process	Mandatory
R3	Functional	Device must read and send ECG signal	Mandatory
R3.1	Functional	Device only reads data when connected to a receiver device (Gateway, Cellphone App, Etc.)	Mandatory
R3.2	Non-Functional (Reliability)	Device as a consistent Sample Rate	Mandatory
R3.3	Non-Functional (Performance)	Device reaches a maximum Sample Rate of 1000 Hz	Optional
R3.4	Functional	Feature extraction (extract from ECG R-Peak Interval, beats per minute, Etc.)	Optional
R4	Functional	Device must read and send IMU data (adding complementary information for context)	Mandatory

are system requirements proposed by CardioID, the R2 and R3.1 requirement pertains more to the user's privacy only when the user pretends to use the device for acquisition the device starts it. Requirement R3.2 serves the purpose of defining a system that reliably obtains data, whereas requirement R3.3 pertains more to the performance of the device, although not mandatory this requirement is very important because with a higher sample rate we have more resolution and therefore more information to evaluate the acquired signal. Lastly in an effort to acquire more context of the data, for example, movement and position, requirement R4 purposes the mandatory use of a IMU, this requirement is also proposed by CardioID only being mandatory for the prototype created in this thesis.

4.1 Proposed system architecture

The verification process determines the quality of the architecture, including aspects of design verification and software quality analysis. Typically, this process involves using formal methods and tools to verify the abstract architecture and conducting unit and integration testing. In contrast, the validation process establishes if the software meets the requirements, including usability and performance tests.

At the start of an effective verification workflow is the definition of system requirements and intended functionality, which can be expressed in use cases and a requirements table (Table 4.1). This information is then leveraged to generate a state machine using Unified Modelling Language (UML) [38], which aids in the verification process. CardioWheel is intended to respond to two use cases: the acquisition, processing and transmission of ECG signal; and the reception and effectivation of Over-The-Air (OTA) firmware updates.

These use cases guide the development process to define several tasks dedicated to responding to each of the steps needed to implement the use cases and the definition of a requirements table, listing the system properties that must be observed through the correct interplay between these various tasks.

In Figure 4.1, we present a modular view of what is needed to develop a system capable of responding to the two use cases defined. This separation, represented in the diagram, aims to indicate where the implementation of the actual architecture ends and the specific implementation for the specific hardware starts.

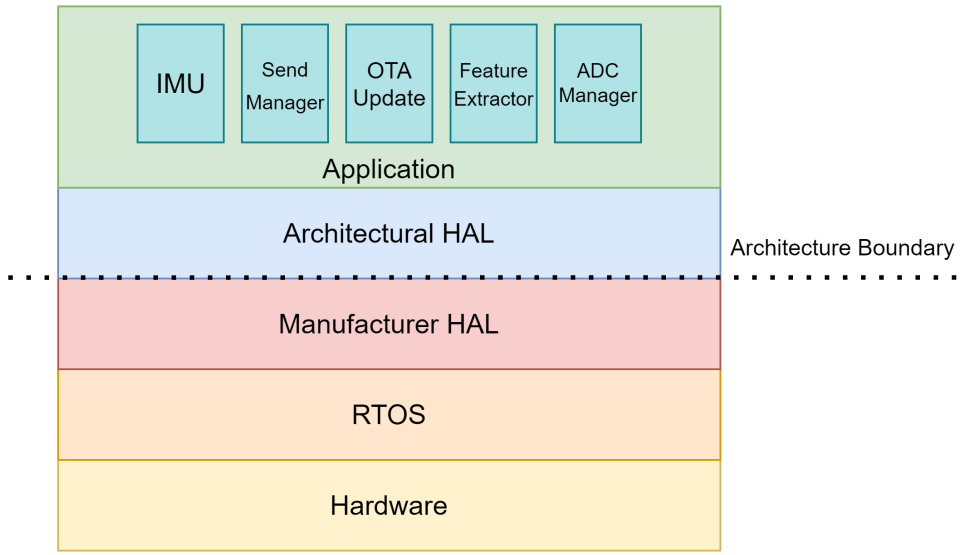


Figure 4.1: Generic Block diagram of the Cyber-Physical System

When choosing the layer after the hardware to manage it, we had two alternatives, bare metal or an Operating System (OS). The bare metal option is great for really resource-constrained MCU, but writing for bare metal creates code that is challenging to maintain, leaving us to the operating system option. This system has real-time requirements (requirement R3.2), which we can meet by taking advantage of the functional aspects of a type of operating system called RTOS, this type of OS allows for splitting of functionality into various tasks, with a kernel responsible for handling the scheduling of tasks greatly simplifying the programming process and allowing for better code maintainability being that each task can be assigned with a different functionality increasing modularity. Another aspect present in an RTOS is the construct of priority level, the priority level of a task is a number assigned to it that defines, in relation to others, the priority of it running, being used later by the RTOS to satisfy processing time according to this priority level, this notion is great if we have a task that needs to be run more often than others, in our case, according to the requirements category, we can assume that the task responsible for managing the ADC will be higher priority than, for example, the task of feature extraction. After the layer of the operating system, we defined a layer for abstraction layer specific to our hardware, this usually pertains to a framework developed to access specific MCU peripherals and can even, in some cases, contain software with protocol implementations that are more efficient for the specific hardware. These can either be developed by the community, by the MCU manufacturer or, in the last case, by the programmer. This layer usually implements most of the software needed to interact directly with the MCU peripherals, but it is often very specific and tied to the hardware architecture, this is a problem because if we are trying to develop a hardware-agnostic architecture and implementation, we cannot be tied to these specific implementations. To overcome this problem we created the Architectural abstraction layer, designed to be generic and abstract the application from the hardware, this layer will include both simple peripheral drivers abstractions and more complex protocol implementations.

The last layer, the application layer, contains the program logic and flow that allows the system to fulfill the required use cases and requirements, this layer is often more complex and it requires defining the flow of the program, and for that, we used state machine diagrams.

The last application layer comprises the operation described in the following state machine Figure 4.2, in it, we define the basic architecture of the CardioWheel, corresponding to a main looping task that coordinates the execution flow of the acquisition and updating related tasks according to the system's state.

The state machine is composed of one idle state that evolves into all other important

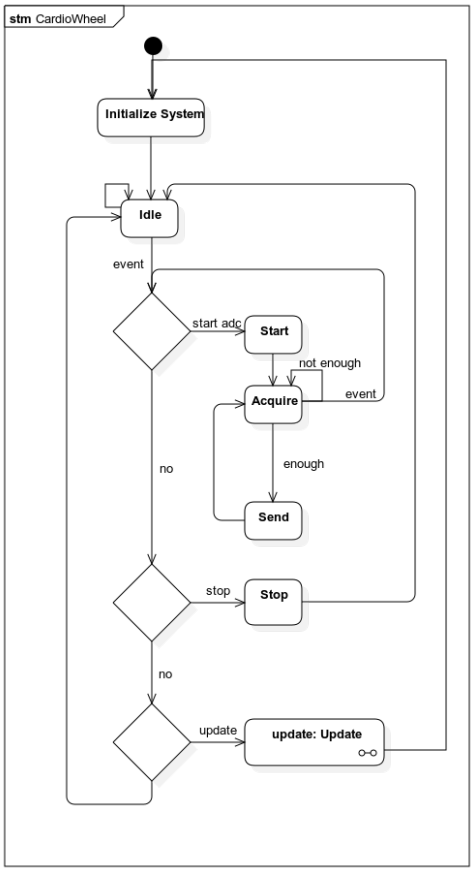


Figure 4.2: CardioWheel state machine

functionality states based on generated events. When in idle, the state machine waits for an event to start either use case and after an event is triggered the state machine evolves into either starting an acquisition or updating the firmware, when starting an acquisition, this state machine proposes an acquisition cycle that collects data samples and sends them after a defined number of samples (N Samples) are acquired. In order to meet the requirements, we must assume that the sending state is non-blocking and returns on time for consistent acquisition. Our goal is to create a reliable and consistent sampling rate. If an event is generated while acquiring, the process will stop either gracefully or not. A graceful stop occurs if the event generated is a stop event. A non-graceful stop occurs if another start event is generated or an update system event occurs. With this process, we ensure that even if the acquisition is stopped, we only miss a maximum of N minus one samples only on the non-graceful stop, provided that stop state implements a send of the remaining data.

In an effort to simplify the reading of the state machine, we created the state machine in Figure 4.3 that represents the flow for updating. After starting the update, the flow comprises three phases, the first phase of the update consists of downloading the

firmware file, after the download, the file needs to be verified, not only the firmware file signature but also the files integrate. Finally, to end the installation process, the last phase is the installation of the firmware, which consists of changing the previous firmware in Flash to the most recent one and restarting the device. This process will vary depending on the specific MCU, with some architectures even allowing rollback and factory resets functionality, and this can interfere with the complexity level of the implementation, possibly affecting the flow presented by this state machine.

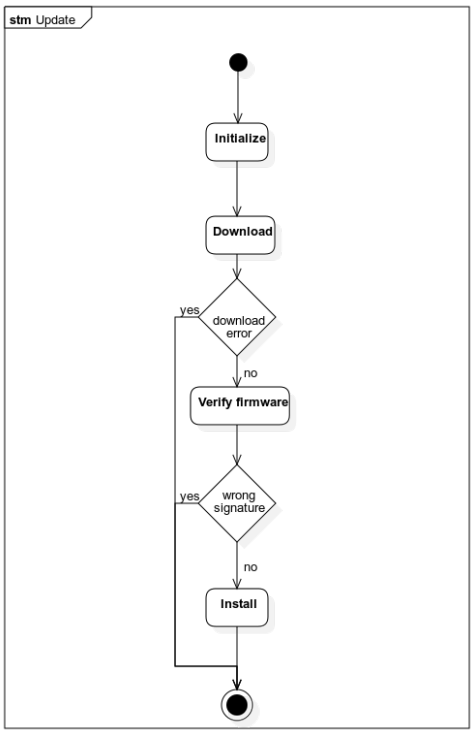


Figure 4.3: Update State Machine

After concluding the definition of the general flow of the state machine, we need to explore better the specific sub-tasks presented. And that can be done by creating sequence diagrams that can later be used to specify better and define the overall system architecture.

In our study, and considering the requirements, we decomposed our state machines into three sequence diagrams comprising the logic architecture of our system. The first one, shown in Figure 4.7, is dedicated to the process of reading the data from the ADC peripheral (related to requirements R3.1, R3.2 and R3.3). This diagram also includes the feature extraction process of the electrocardiogram signal, named post processing. As a result of these feature extraction, we can later obtain values like R-Peak Interval and beats per minute (requirement R3.4).

The second sequence diagram created was associated with sending the data to a receiver in a protocol-agnostic manner, represented in Figure 4.4. On this diagram and to be protocol-agnostic, we propose a single data-sending function, this function is fully responsible not only for encapsulating and composing the message if it is required depending on the protocol but also responsible for sending the data in chunks in the case that it is needed.

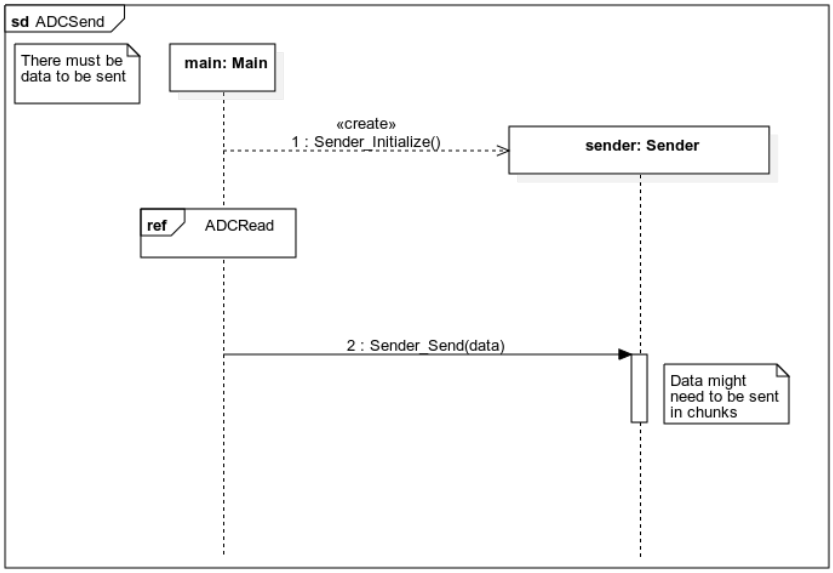


Figure 4.4: Protocol-agnostic Send Data Sequence Diagram

Finally, to guarantee the requirement R4, we proposed the sequence diagram presented in Figure 4.5, this diagram proposes the initialization and configurations of a IMU peripheral, and after reading raw accelerometer and gyroscope data.

There is also a more hardware and protocol-specific sequence diagram for all of these diagrams, in the case of ADC (Figure 4.6) composed by not only the peripheral initialization and configuration but also the read and conversion process based on the voltage reference.

To fulfil the requirement R1.1, the sequence diagram on Figure 4.8 contains BLE specific configuration parameters; like advertising name and GATT Characteristics data (service and attributes identifiers). The diagram was also created with consideration for the necessary callbacks required for BLE reading and related operations.

Finally, we produced a diagram for our implementation's IMU, the BHI160d. To create this diagram, we consulted the information provided in [10]. This allowed us to obtain the necessary operational information for building a complete sequence of actions, including initializing the sensor, uploading the sensor firmware after each reset, and the

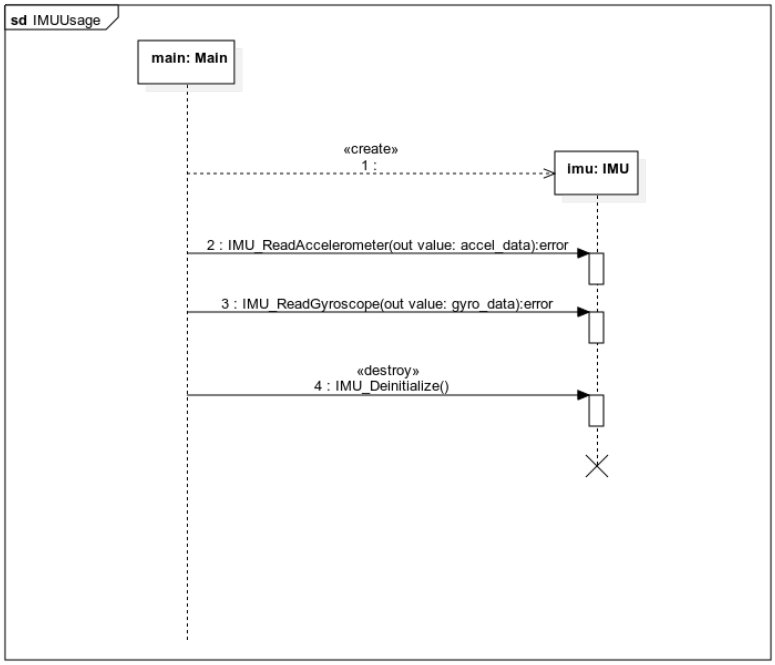


Figure 4.5: IMU Sequence Diagram

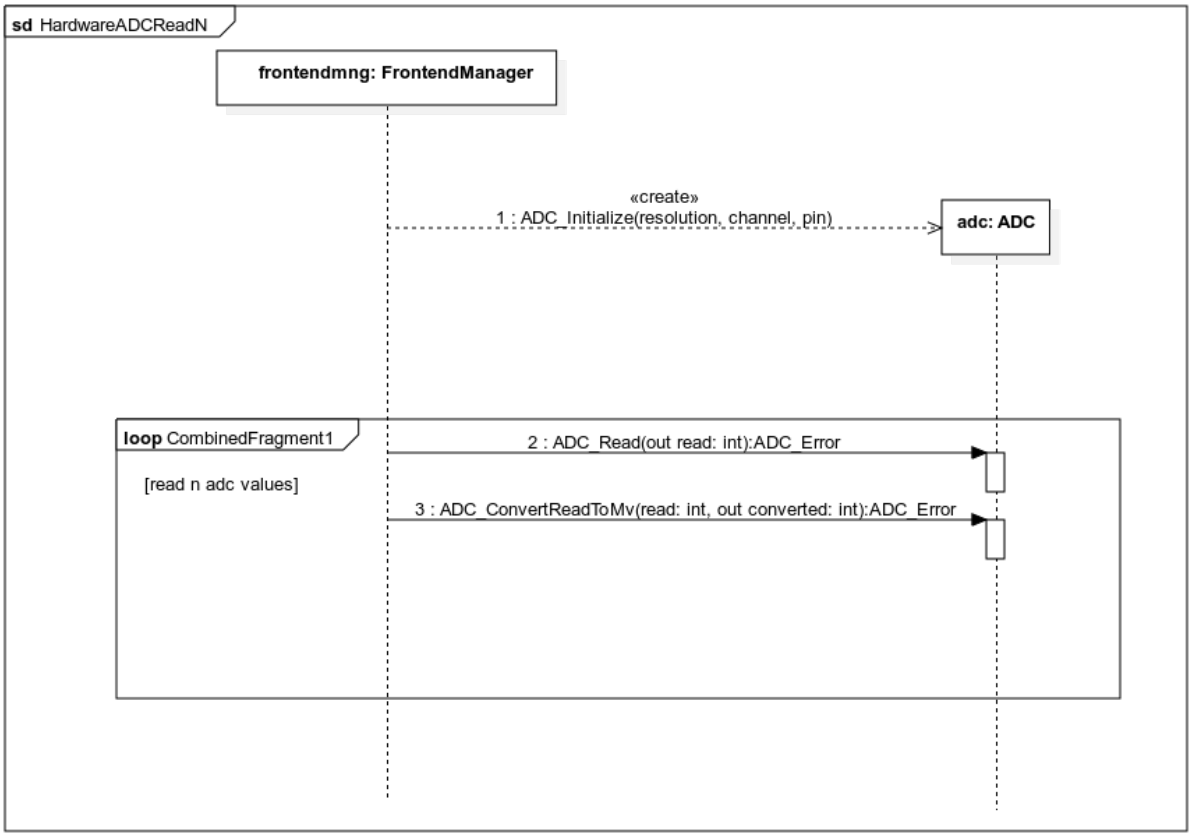


Figure 4.6: Hardware Read Sequence Diagram

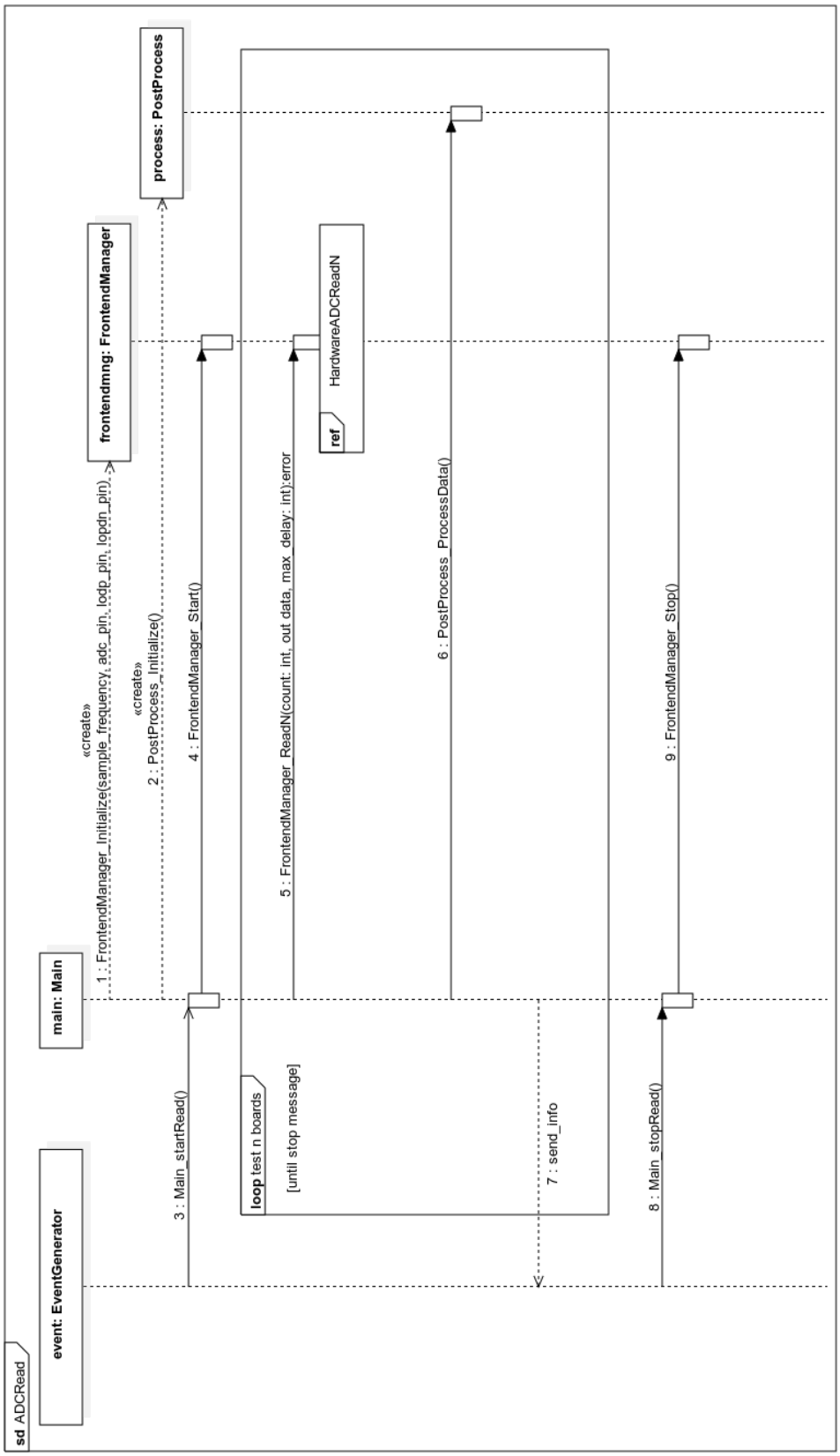


Figure 4.7: ADC Read Sequence Diagram

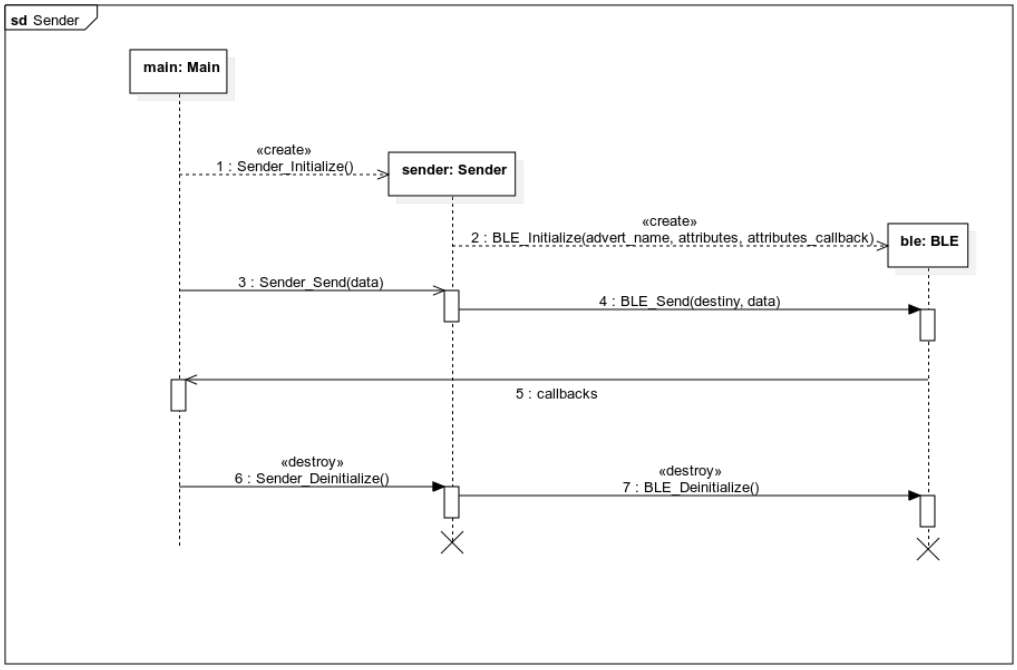


Figure 4.8: Send Data Sequence Diagram

reading process involving a FIFO stack. These details are specific to the sensor and can be found in Figure 4.9.

Compiling the present on all sequence diagrams results in the creation of detailed architecture (Figure 4.10). There is a distinction between all the necessary modules and classes following some methodologies presented in [42]. The detailed architecture and sequence diagrams serve as a blueprint for the implementation process.

In this detailed architecture, we propose four modules: a module containing the software for reading the ECG data, a second module for configuring and reading the IMU data, a module that handles the BLE communication and finally a module containing utility software, like the post-processing of data and logic that implements stable sampling of the ADC as well as the data sending logic.

4.2 Threat Modelling

To deal with the security dimension, a Threat Modelling process was used following the methodology outlined in [33]. As a result of this, an Attack Sequence Table (Table 4.2) is created. This table information will later be used to help choose the security measures necessary to develop a secure device.

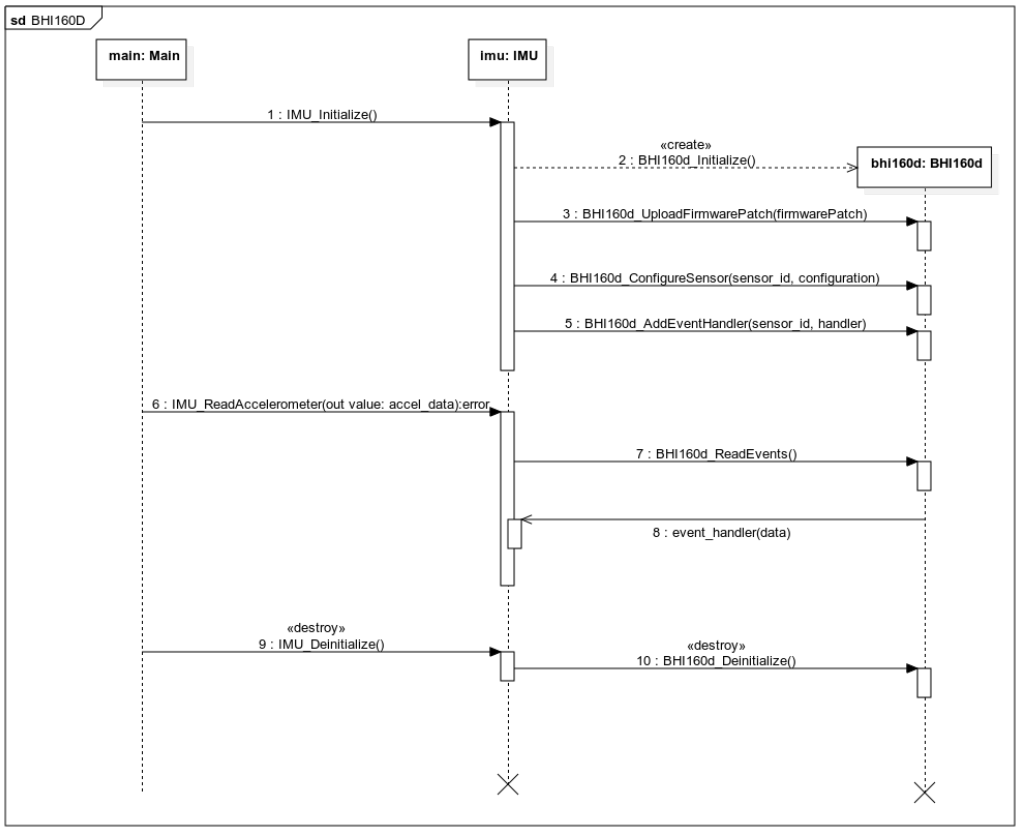


Figure 4.9: BHI160d IMU Sequence Diagram

Table 4.2: Attack Sequence Table

S/N.	**Point of Entry**	**Threat Actor(s)**	**Sequence of Attack**
1.1 - Reconnaissance	Device	APT Group, Cybercriminals	1,3,5
1.2 - Reconnaissance	Operating System	APT Group, Cybercriminals	1,2
2.1 - DoS	Application	Cybercriminals	1
3.1 - Application vulnerabilities	Application	APT Group, Cybercriminals	1,2,4,5,6 ✰
3.2 - Application vulnerabilities	Application	APT Group, Cyber criminals	1,3,5,6 ✰
4.1 - MITM	Application	Cyber criminals	1,7 ✰
4.2 - MITM	Application	Cyber criminals	1,7 ✰

Security aspects outside of the scope of this threat modelling scenario won't be taken into account, but it is important to note that, all of the network and application scenarios related to the gateway or external device, need to be secure in order to have a secure system overall. This concerns especially the firmware update of the device and

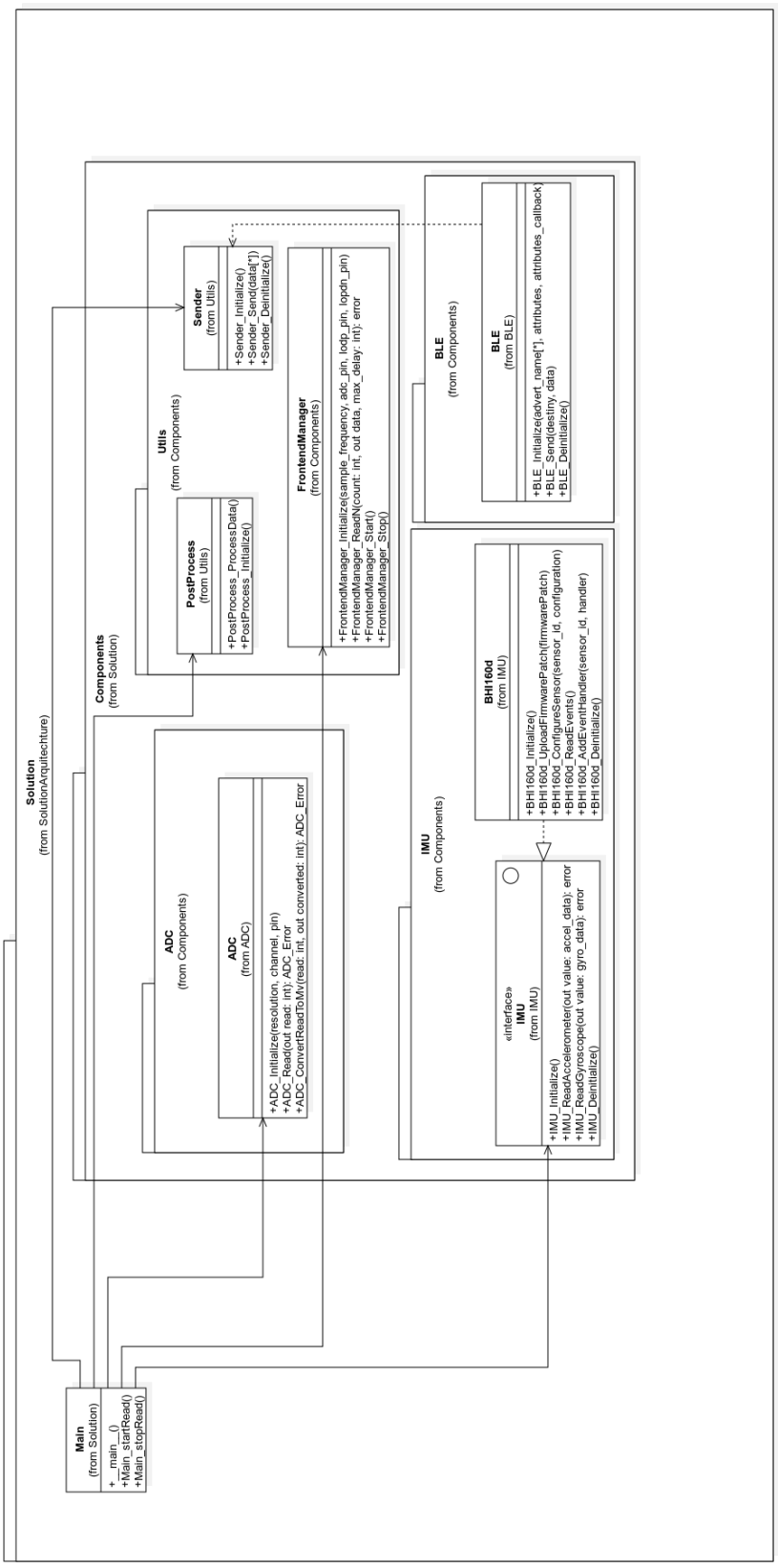


Figure 4.10: Detailed Architecture

Table 4.3: Attack Sequence Description

S/N.	Threat Event	Examples
1.1	With scans of MAC addresses and information published by the device on BLE advertisement or even the device Bluetooth class, an attacker can plan an attack to a known vulnerability on that hardware.	Using scanning tools like btscanner, hcitool, etc. we can find the device class, as well as its MAC address. With the MAC address using external tools, websites (e.g. https://macvendors.com/) or btscanner itself, we can determine the OUI owner which is the company owner of that MAC address range.
1.2	Preform scans to discover properties like MAC addresses, OS specific measurements. Such information can be used to exploit known vulnerabilities of the operating system	Exploring tools like Nmap, if the device uses wifi interfaces and exposes TCP/IP or UDP ports, another device can scan and identify the operating system based on fingerprinting those open ports. After that, this information can be used to explore OS-related vulnerabilities.
2.1	Floding the device application with Bluetooth requests.	Using Bluetooth DOS attack scripts to flood the stack with requests, crashing the device.
3.1	Using memory leaks the attackers can exploit vulnerabilities that allow sensitive data to be extracted from the device's memory.	Using techniques like buffer overflows, to inject code or induce code failure.
3.2	Exploring buffer overflows or bad driver implementations to gain access to sensitive data.	Using techniques like buffer overflows to inject code or induce code failure.
4.1	MITM Attacks gaining access to data that is being sent from the application to the authenticated Gateway or external device.	Using BLE MITM tools like GATTACKER.

the sharing of biometric data to the cloud servers. Although this information is encrypted and signed (only the firmware contains a signature) it is essential to guarantee that the data is securely transported to and from the respective devices and is safely stored and backed up on the databases.

When it comes to the attack vectors defined and the likelihood and impact analysis of the threats is required to create a secure system. This step is done after identifying possible threats and results in a risk matrix (Figure 4.11) containing the impact of a successful attack on the system for the overall system security and the organization. The risk matrix also contains the likelihood of a specific threat resulting in a successful attack.

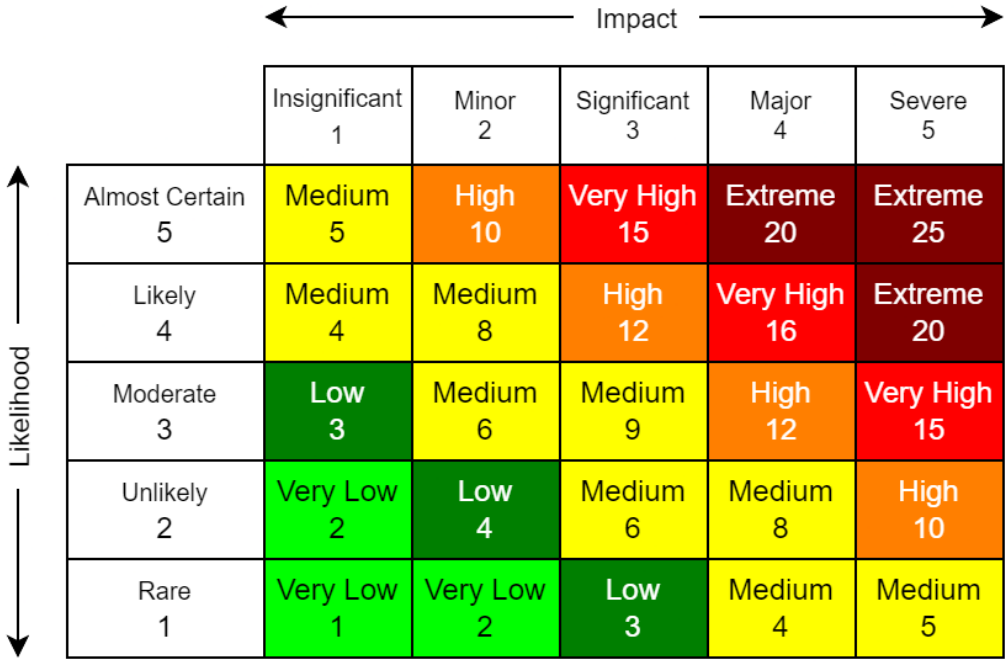


Figure 4.11: Risk Matrix

(based on [58])

4.3 Security Solutions

Although the data sent is encrypted, ensuring that the transmission tunnel is also secure is an excellent way of guaranteeing multiple layers of security, in the case of this work, with the usage of security techniques provided by the communication standard chosen (BLE) this is accomplished and also helps further mitigating threats like MITM attacks and impersonation attacks. Impersonation attacks are still possible as alighted by [2].

Another encrypted tunnel was used to implement the OTA update, to add to the security of the system, every new firmware contains an encryption key that should be used on the next firmware update increasing the security level, this is because it simplifies the process of changing the encryption key for specific devices and in the case of the key being cracked, it also makes it possible for the keys to be unique to the devices. With this, something that needs to be considered is the database that relates the devices to each of their encryption keys, which needs to be secured, backed up and reliable.

4.4 Implementation Prototype

In this chapter, you will find a detailed explanation of how the modelled architecture was implemented. It covers important aspects like the technology stack and framework utilized and specific details pertaining to challenges related to the hardware used.

To implement, first we need to define what modules are required to accomplish the solution, considering the requirements for CardioWheel, the proposed architecture needs and the specific Hardware present in CardioWheel (e.g. sensors). This step consists of creating a hardware-specific block diagram (Figure 4.12). This block diagram comprises five essential layers; Firstly, the **Hardware** layer constitutes the physical hardware, in this case, the CardioWheel CPS, presented in Sub-section 4.4.1. Next, the **FreeRTOS** layer contains the RTOS, that powers the system, this layer contains all of the operating system components for task creation and management as well as other operating system functionality like Queues, Semaphores and in this case, as the hardware allows for it, multi-processing handling. The third layer, **ESP-IDF**, comprises ESP32 MCU-specific programming, including peripheral drivers, protocol implementations, and connection handling software, it's also important to note that the framework and RTOS layers are coupled because the framework is distributed with FreeRTOS inside. Next, the **Abstraction** layer aims to create a layer of software that separates the application code from the specific libraries found on the ESP-IDF, creating a generic Application that improves code portability. Lastly, the **Application** layer contains all the functionality and software that uses the Abstraction and FreeRTOS layers to implement all the tasks needed to satisfy all the requirements defined.

Considering the application needs and the diagrams described previously in this chapter, the abstraction layer is created. This layer contains the following components:

- ADC - Component that configures and reads from the ADC
- Frontend Manager - Component taking advantage of ADC to acquire ECG with the defined configuration
- GPIO - Component that configures, reads and sets the value of IO on the MCU
- I2C - Abstraction layer for interacting with devices through the I2C protocol
- SPI - Abstraction layer for interacting with devices through the SPI protocol
- BLE - Component that handles BLE connection

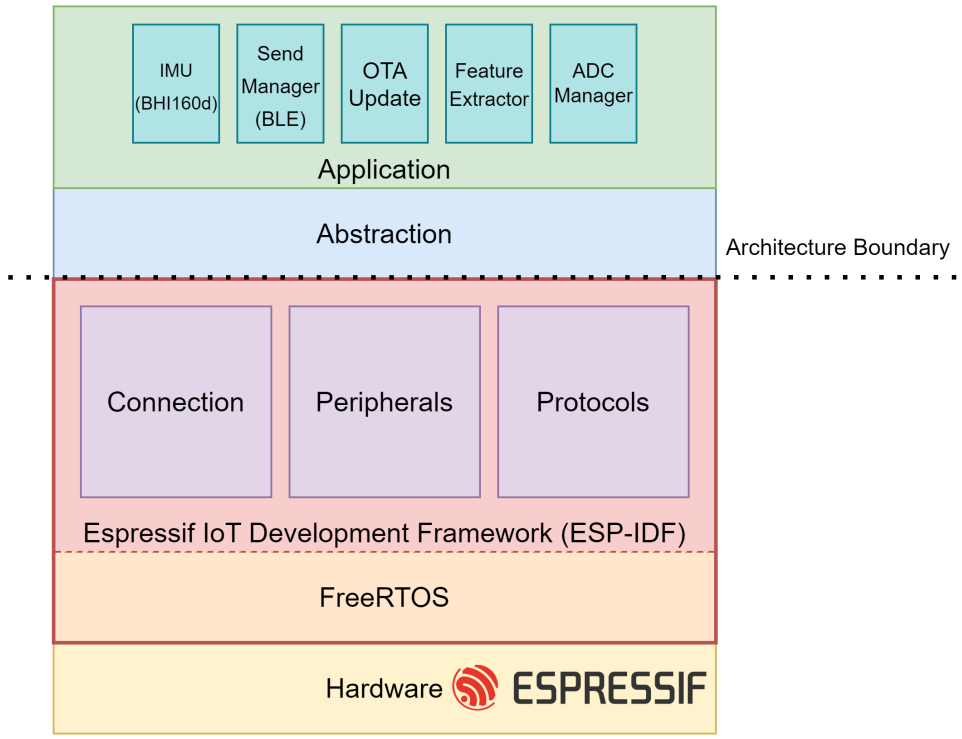


Figure 4.12: Block diagram of the Cyber-Physical System

- BLE Manager - Abstraction layer to manage BLE input an output
- OTA - Component that updates the firmware on the device
- Updater - Abstraction layer that uses the OTA component to update the device
- BHI160D - Component that configures and reads data from the BHI160D Inertial Measuring Unit (IMU)
- BQ27441G1 - Component that reads information from the fuel gauge
- Battery Manager - Battery Manager abstraction layer to allow for different types of fuel gauges

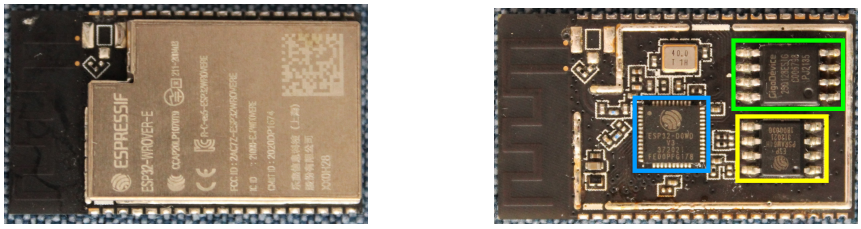
In the following subsections, modules that are more important to this work will be described as well as the hardware present on the board. The doxygen-generated documentation can also be found in Appendix B.

4.4.1 Hardware

The hardware part of the system is based around the Microcontroller (MCU) ESP32 inside the ESP32-WROVER-E-N16R8 module package, this MCU contains 520 KB of

internal static Random Access Memory (also known as SRAM) of which 320 KB of DRAM (Data RAM) and 200 KB of IRAM (Instruction RAM), there is also 16 KB of SRAM dedicated for the Real Time Clock (RTC) half called slow memory and the other half called fast memory, 1 Kbit of eFuse to store application specific configurations and 448 KB of internal Read-only Memory or ROM. Added to the internal memory there's 8 MB of external SRAM and 16 MB of external ROM located inside the module package as illustrated by Figure 4.13b.

This MCU includes a BLE and WiFi radio, this MCU also goes through the following hardware reliability testing with High-temperature Operating Life (HTOL), High-Temperature Storage Life (HTSL), Highly Accelerated Temperature and Humidity Pressure Test (uHAST), Temperature Cycling Test (TCT) and Electrostatic Discharge (ESD). It has a configurable clock frequency from 80 MHz to 240 MHz, which can be imperative for guaranteeing a high and constant sampling rate while sending the data simultaneously, it is also dual-core, helping with this problem. It includes some peripherals, like I2C, SPI and a 12-bit ADC that are needed for this implementation.

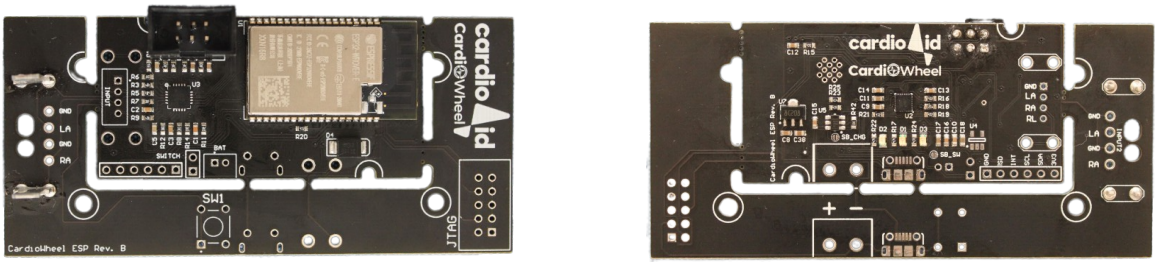


(a) Module with cover

(b) Module without cover (■MCU, ■RAM, ■ROM)

Figure 4.13: ESP32-WROVER-E-N16R8 Module

The complete system, observable by the Figure 4.14, is composed, at the front, by the MCU and a front-end integrated circuit (AD8232 [4]), at the back by power regulators and a sensor hub (BHI160d [10]) with an accelerometer and a gyroscope.



(a) CardioWheel Front

(b) CardioWheel Back

Figure 4.14: CardioWheel

4.4.2 BLE Manager

This module contains the universal unique identifiers for the services and characteristics. It also implements the functionality required to start the BLE advertisement using the BLE module, add and read handler for specific characteristics and a handler for things like writes and notifies. This functionality is fundamental for the application because every data exchanged to and from the device is done using BLE.
To implement the BLE Manager a series of specification documents were created for each service and characteristic, in this subsection, we give an overview of the present services and characteristics, it will be a surface-level description as the more in-depth description will be located in the BLE specification confidential documents.

Table 4.4: GATT Services

Service name	**UUID**	**16-bit Identifier**	**Description**
ECG	f334ccc0-55fe-01bf-e611-b9e4166680f9	0xccc0	The ECG Acquisition procedure is supported this BLE GATT Service, with one Characteristic. The CCCx values are placeholders for the 16-bit identifiers of the service and each characteristic.
OTA	6f746100-55fe-01bf-e611-b9e4166680f9	0x6100	This service supports the firmware update procedure, it contains three Characteristics. The 610x values are placeholders for the 16-bit identifiers of the service and each characteristic.
IMU	696d7500-55fe-01bf-e611-b9e4166680f9	0x7500	The IMU Acquisition procedure is supported by one custom BLE GATT Service, with four Characteristics. The 75xx values are placeholders for the 16-bit identifiers of the service and each characteristic.

Table 4.5: GATT Characteristics

Service name	**16-bit Identifier**	**Description**
ECG	0xccc4	Raw ECG Value and Lead-On Detection
OTA	0x6101	Firmware Information Characteristic
OTA	0x6102	Firmware Data Characteristic
OTA	0x6103	Firmware Execute Characteristic
IMU	0x7501	Accelerometer Value
IMU	0x7502	Gyroscope Value
IMU	0x7503	Accelerometer Config
IMU	0x7504	Gyroscope Config

Besides these custom characteristics, the system uses some standard services as specified in [31]. These are the standard Battery Service, Generic Access Service with Device Name Characteristic and Device Information with Manufacturer Name String, Firmware Revision String, PnP ID and Serial Number String characteristics.

4.4.3 Frontend Manager and Processing

Frontend Manager, contains all the needed software to setup an acquisition system with a specific and exact sampling frequency. To accomplish this in the present system we require two types of peripherals, a hardware timer and a ADC. The first possible approach could be by just using the ADC with a loop and a specific delay. This is problematic for two main reasons, the minimum delay possible in FreeRTOS is 1ms which could be a problem because reading and processing the ADC data takes some time, making it impossible to have exactly a 1ms delay between reads what will happen is the time between read will be 1ms added to the time it takes to read an process the data, another problem is the fact that some tasks do not run in concurrency can result in the task running the sampling of the ADC not be called n the correct timing resulting in inconsistent sampling timings. This brings us to the second possible approach, the fact that the system has the capability of running two cores simultaneously provides the possibility to reserve a single core to the reading of the ADC, this option eliminates the problem with inconsistent, but while consuming a whole core for that task does

not fix the problem with the added time of reading and processing of data. To fix all of these problems, there is a third solution, the one that was implemented. This solution consists of the configuration of a hardware timer to deploy an interrupt, while handling this interrupt the ADC is read and sent to a Queue to be processed and sent to a client device, this method of reading provides a reliable and correct sampling rate. It is important to note that a relatively high-priority task is tasked with consuming the data from the Queue so that no data loss occurs. Figure 4.7 describes this process.

This last solution also allows for simultaneously process the data, the processing of the data is crucial for this application as it is composed by not only digital filtering of the signal but also for feature extraction. The feature extraction process calculates the biometric features that can directly be used to identify a persons drowsiness level, possible heart health and even biometric identify them. If it was not possible to do it on the device we would be required to send constantly a stream of raw data, processing it gives the possibility of sending only the important features obtaining the same result with less throughput.

5

Evaluation

In this chapter, we will discuss the outcomes of our research. We conducted a series of rigorous experiments to test our proposed model and implemented solutions. Ultimately, we will showcase the results obtained and draw conclusions on the strengths and limitations of our model. These insights can be used to advance future work. While the diagrams created in the previous phase of the project help provide an abstract definition of the system, their real potential is unlocked when translated to be used with formal verification tools that evaluate if the system's design is compatible with the established requirements. One example of such a tool is Uppaal, a tool jointly developed by the universities of Uppsala (Sweden) and Aalborg (Denmark)[9].

5.1 System architecture verification

Uppaal uses Timed Automata (TA) models to represent the system's tasks. TAs are directed graphs representing system states in their nodes and conditional transitions in their edges. Uppaal exploits this structure to include real-time and concurrency properties in the system's representation. Furthermore, the definition of several of these TAs, one for each task, connected through trigger nodes, allows the tool to explore timing-related requirement coverage and concurrency-related faults before any line of code is written.

To use Uppaal, we converted the State Machine UML models into a TA model, expanding the architecture possibilities as TA can model real-time systems with concurrency.

In a TA, each state is finite and can be terminated when a condition of an outgoing transition is met. This transition will occur once activated according to a random choice or a predetermined priority. To accomplish this conversion, we need to determine the timing of our state machine and its operations. To do this, we resorted to the sequence diagrams.

To support the complete system verification, further specification of data acquisition, data transmission and update management was conducted. Additionally, to emulate external interaction and restrict the acquisition task to follow a specific sampling frequency, the concepts of interrupt/event generators and timers were created in Uppaal. External interaction emulation is important as acquisition start and stop signals, as well as the update request, are received as external inputs that change the system's internal state. The timer has the important role of ensuring the readings are performed in fixed time intervals to guarantee the precision of the sampling rate. These two components were modelled into TAs represented in Figure 5.1a, and Figure 5.1b that generate events and interact with the overall state machine.

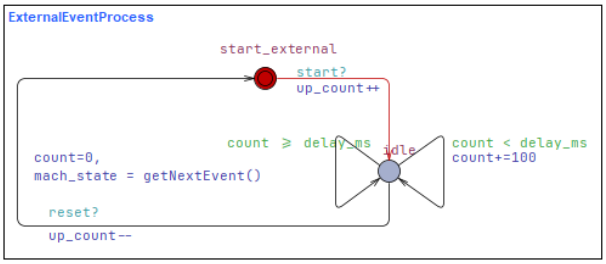
The timer TA can be started by generating a start event. When started, this TA increases a value and generates an event transition. The location that perpetuates this transition (timer) is defined as urgent, meaning that it will not induce the passage of time. This is important because its purpose is only to introduce a timed event in another TA and should not consume any time. The channel used (clk) is also urgent, this is essential because, when triggering a synchronization through it, it will have priority over transitions that require time. To stop the timer, we can induce a stop timer or system reset through adequate channels.

The event generator intends to simulate user events, which takes a timeout and a list of states, and for every timeout number of times, a new event will be induced.

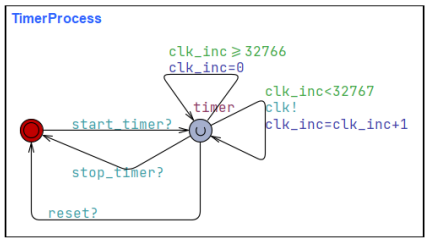
The actual functionality of the state machine is modelled in four other separate TAs. The first is the main process TA (Figure 5.1c) that effectively controls the flow of the machine, which is composed of all of the channel synchronization needed to start the execution of the current state.

The next TA is the ADC TA (Figure 5.1d) which, when started, takes the responsibility of starting a timer and following its synchronization signals to save the data read on the ADC peripheral at the specified sampling rate. After a predetermined number of samples is acquired, the samples are sent and the sender TA (Figure 5.1e) is activated.

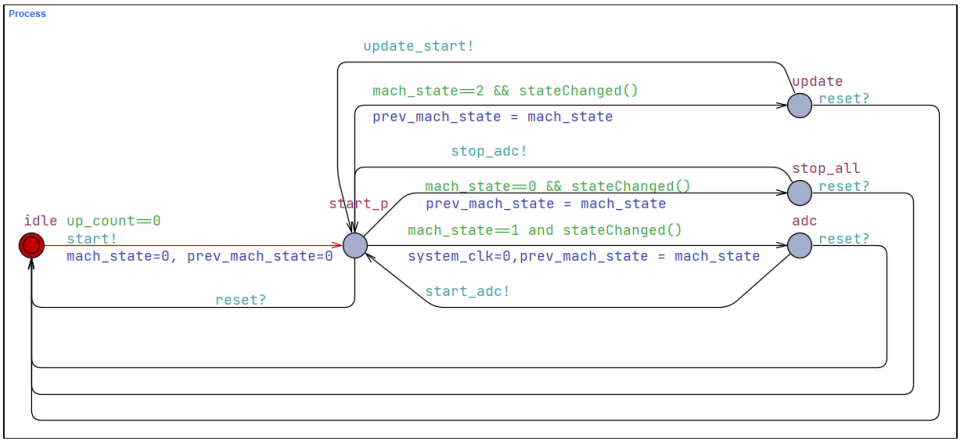
The last TA (Figure 5.1f) is the update TA, its purpose is to simulate an update on the device and, at the end, prompt a reset of the system. The reset needs to be propagated through the whole state machine, and for that we use a synchronization channel of the



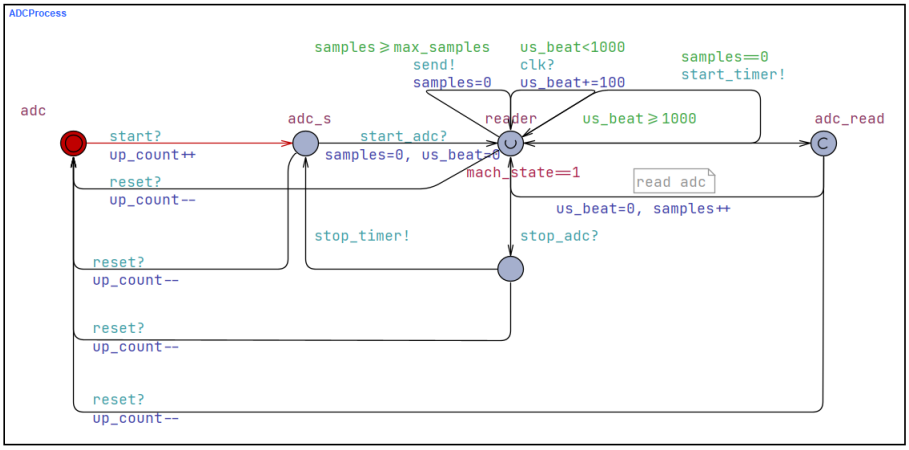
(a) External Event TA



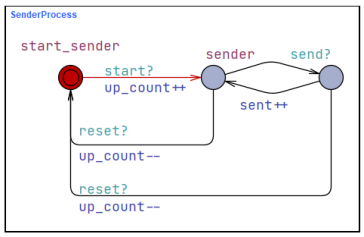
(b) Timer TA



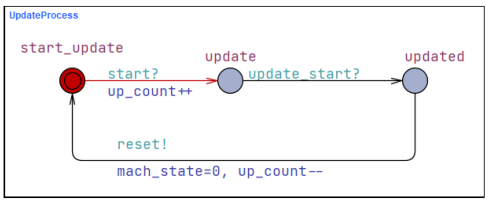
(c) Main TA



(d) ADC TA



(e) Sender TA



(f) Update TA

Figure 5.1: TA Processes

broadcast type.

With this definition, the Uppaal tool can explore the TA graphs starting from different trigger nodes, *i.e.* starting from the various possible system states, and verify that specific system properties are kept throughout those runs. By defining heuristic rules that represent those properties, it is possible to have guarantees that the system will not enter any deadlock states or that the signal acquisition and transmission tasks are properly cancelled when starting an OTA update. Furthermore, it is also possible to identify dead branches in the system design, *i.e.* execution steps that cannot be reached through the system's expected behaviour.

As a part of the validation process, we created heuristics that test this architecture. To test the stability of the system we tested whether there were any possible deadlocks in the designed flow, we also tested if some of the states were eventually met while also testing if the timing requirements for the acquisition were met. All of this was done considering the heuristics language used by Uppaal [8].

The first heuristic ($E <>$ *not deadlock*) checked if there would be a deadlock in the system eventually. After that, we tested if the state changes over time ($E <>$ *mach_state* $==$ 1) to assess if the state machine was evolving as expected. This step involved two heuristics ($E <>$ *Process.adc and mach_state* $==$ 1, $E <>$ *mach_state* $==$ 1 *and ADCProcess.reader*) to check if the required processes were running correctly in that state. Similarly, next, we tested the following state, testing next the other two heuristics created to evaluate the system's flow ($E <>$ *mach_state* $==$ 2 *and not ADCProcess.reader*, $E <>$ *mach_state* $==$ 2 *and UpdateProcess.updated*). As presented in Figure 5.2, all of these heuristics were tested and passed, as represented by the green colour.

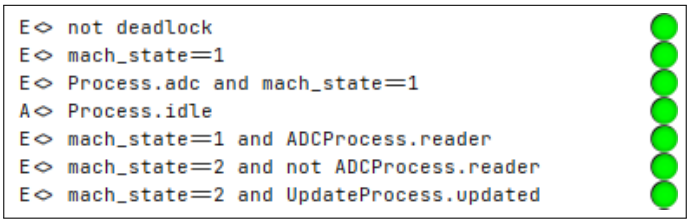


Figure 5.2: Uppaal Results

5.2 Experimental Evaluation

Validating this system requires the development of tools capable of testing the major aspects of the system like the microprocessor, the analog front-end and the wireless capabilities.

To do this, an automatic validation station was developed in the scope of the VALU3S

project. It comprises a raspberry-pi-enabled central unit and a simple touch interface (see Figure 5.3). The central unit manages the installation of test firmware and provides a local network and a BLE master to test the communication capabilities of the CardioWheel. It also coordinates the injection of a synthetic ECG signal that allows the validation of hardware properties of the analogue front-end.

To validate timing-related properties robustly, the validation station also manages the creation and installation of monitors that perform run-time verification based on formal requirements [32]. This formal validation method observes expected system properties as it runs in real-time.

Encapsulated in a compact format and with a simple interface, this validation system provides two major advantages to the V&V process of CardioWheel's production:

- The automation of the whole process ensures that a thought-out V&V process is planned and its application is made wholly and systematically.
- The expertise requirements to perform V&V of such a complex system becomes concentrated on the design phase of the V&V routines. Still, it can be applied by any operator, significantly reducing the costs associated with system validation.

Figure 5.3: Validation System: The touch interface allows any operator to select the appropriate hardware and firmware version, and the station manages the test procedure and production firmware installation.

Besides using the Validation System, the CardioWheel system was tested in a clinical environment against a clinical-grade Electrocardiogram system in Hospital de Santa

Marta. This test was approved by the Clinical Trial Unit of the Centro Hospitalar Universitário de Lisboa Central, and aimed to collect data with real patients, demonstrating the real accuracy of the system when compared with an already certified acquisition system.

The data was extracted from the medical digital record, an example of a medical ECG is in 5.4, using an automatic parser developed in CardioID. An example of signals obtained with CardioWheel is in 5.5. The two signals don't have the same sampling frequency, CardioWheel samples at 1000 Hz and the Clinical ECG samples at 500 Hz. To compare these two signals, we needed to extract and resample them to be in the same sample frequency. We chose to resample the medical ECG to 1000 Hz and to do that we used interpolation [53].

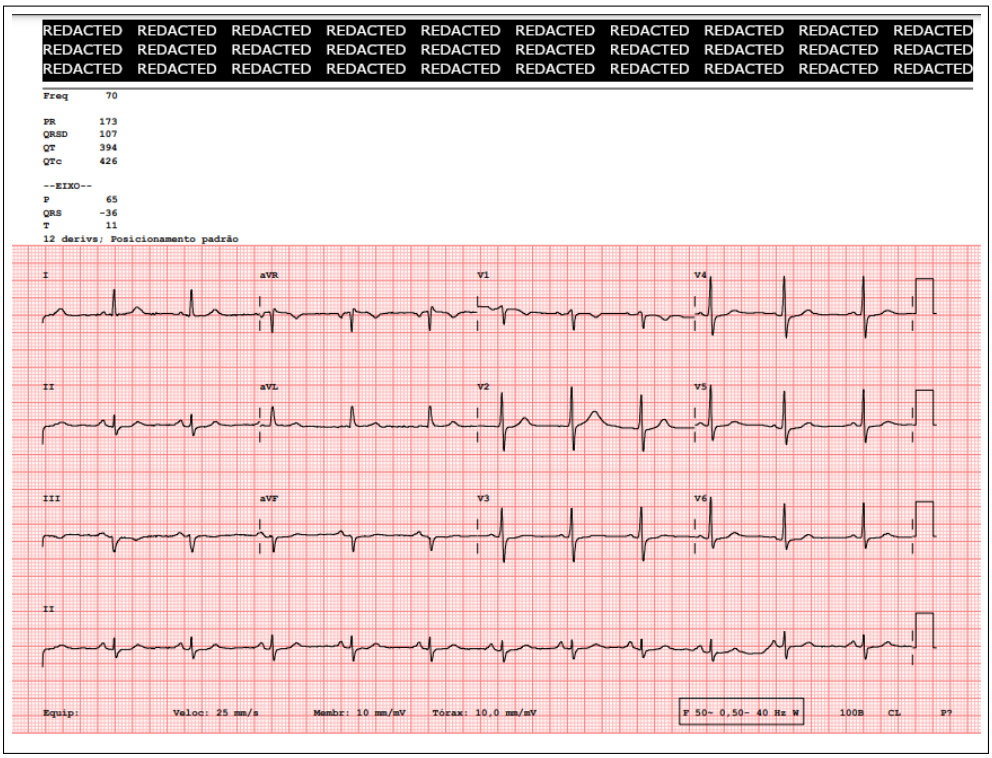


Figure 5.4: Medical ECG example

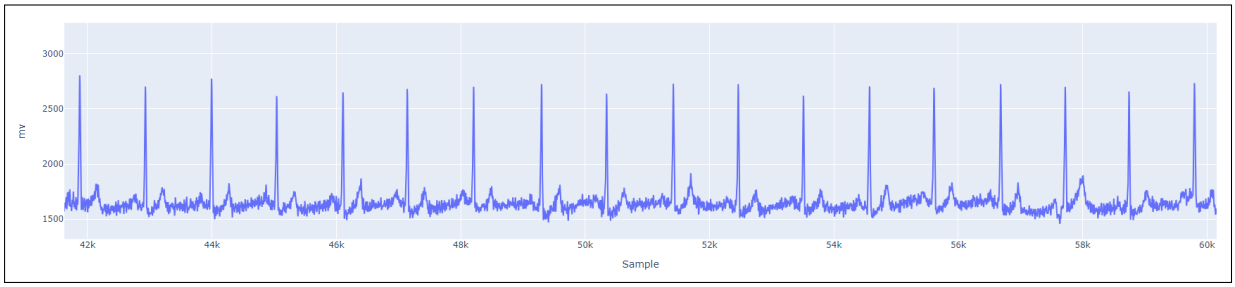


Figure 5.5: CardioWheel ECG example (equivalent to Lead I).

Both acquisitions were performed simultaneously, but the start time may differ. To find the set of beats that are common between the acquisitions, a correlation between the two signals was performed, resulting in the best sobreposition between the two signals, as exemplified in 5.6 (with Lead I), and in figure 5.7 (with Lead II). In both pictures, the alignment was very good.

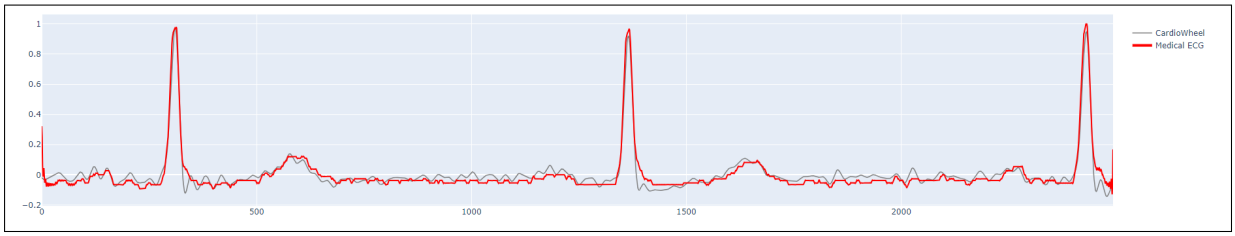


Figure 5.6: Correlation Process (Lead I)

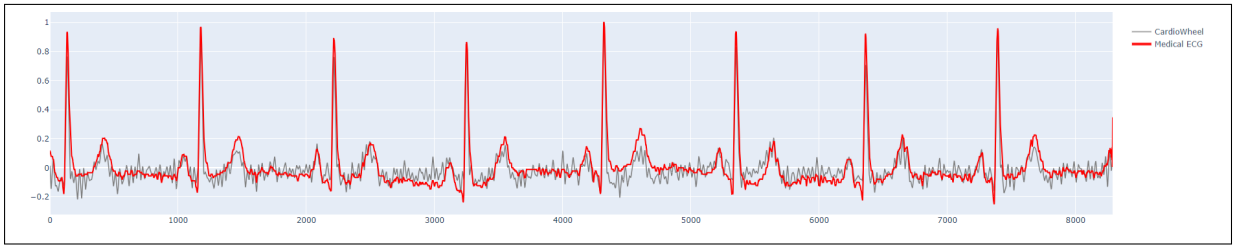


Figure 5.7: Correlation Process (Lead II)

In order not to introduce pathology-related variability, a set of non-pathological signals was selected from the obtained database to quantify the systematic alignment between the obtained signals. In tables 5.1 and 5.2 are presented the mean correlation coefficient, the standard deviation, maximum and minimum values (in percentage) between CardioWheel signals and clinical Lead-I and Lead-II, respectively.

Table 5.1: Signal Correlation Lead-I With CardioWheel (Non-Pathological Signals)

Correlation	**Standard Deviation (%)**	**Max (%)**	**Min (%)**	**Mean (%)**
Signal Correlation	19.99	96.96	30.38	83.19
R-Peak Correlation	02.01	100.0	94.20	99.18

When compared against Lead-I, the signal correlation has an average above 80%, and when comparing just the R-Peak time series, the correlation is almost 100%. These results are not as significant when compared against Lead-II (4.3). This is expected because CardioWheel obtains the signal in Lead-I also.

Table 5.2: Signal Correlation Lead-II With CardioWheel (Non-Patologic Signals)

Correlation	**Standard Deviation (%)**	**Max (%)**	**Min (%)**	**Mean (%)**
Signal Correlation	14.14	91.80	41.59	71.30
R-Peak Correlation	00.07	99.99	99.70	99.98

We also observed that even tho the CardioWheel is sampling in Lead-I the R-Peak correlation is even better when compared to Lead-II, this fact is most likely due to the sample size difference between the two. In Lead-II the signal obtained is, as evidenced by Figure 5.4, basically four times the size. Having more information to compare to the CardioWheel acquisition is beneficial because it tests bigger sampling portions than Lead-I.

Additionally, to analyse the quality of the CardioWheel signal, a series of quality metrics following [66] were used. The selected metrics quantify the signal quality in terms of signal power on the interest band ($pSQI$ - equation 5.1), in terms of the kurtosis ($kSQI$ - equation 5.3), and finally regarding the baseline wander ($basSQI$ - equation 5.5). It is also important to note that although the R-Peak detection match ($qSQI$) metric is also crucial, we did not consider it for this test because the process used to test the R-Peak correlation as the objective of testing the same metric differently.

$$p\text{SQI} = \frac{\int_{f=5Hz}^{f=15Hz} P(f)df}{\int_{f=5Hz}^{f=40Hz} P(f)df} \tag{5.1}$$

$$ECG \begin{cases} \text{optimal}, & p\text{SQI} \in [l_1, l_2] \\ \text{suspicious}, & p\text{SQI} \in [l_3, l_1] \\ \text{unqualified}, & p\text{SQI} > l_2, \text{ or } p\text{SQI} < l_3 \end{cases} \tag{5.2}$$

$$k\text{SQI} = v_4 = \frac{E\left\{(x-\mu_x)^4\right\}}{\sigma^4} \tag{5.3}$$

$$ECG \begin{cases} \text{optimal}, & k\text{SQI} > 5 \\ \text{unqualified}, & k\text{SQI} \leqslant 5 \end{cases} \tag{5.4}$$

$$\text{bas SQI} = \frac{1 - \int_{f=0Hz}^{f=1Hz} P(f)df}{\int_{f=0Hz}^{f=40Hz} P(f)df} \tag{5.5}$$

$$ECG \begin{cases} \text{optimal}, & basSQI \in]0.95, 1] \\ \text{suspicious}, & basSQI \in [0.9, 0.95] \\ \text{unqualified}, & basSQI < 0.9 \end{cases} \tag{5.6}$$

The results are shown in table 5.3 and show that the signals present very good $pSQI$ and $kSQI$, while not so good $basSQI$. This fact is explained since dry electrodes are used in the acquisition, which creates additional baseline wander artefacts.

Table 5.3: Baseline Values (Non-Patologic CardioWheel Signals)

Level Type	**Value Mean**	**Score**
Power Level (pSQI)	1.0	Optimal
Baseline Level (basSQI)	0.93	Suspicious
Kurtosis Level (kSQI)	20.43	Optimal

We conducted a final test to validate the system's stability over time. We set up a scenario as shown in Figure 5.8, where we used a CardioId Gateway to measure the average connection duration for the CardioWheel. We conducted eight tests, each time ensuring that the device was placed less than one meter away from the Gateway and had a fully charged battery. Our test results showed that the system could transmit for an average of 24 hours and 4 seconds, as shown in Table 5.4, before the battery was completely drained and without firmware crashes or failures to transmit. With these results, we can conclude that the system can transmit reliably, with the only limiting factor being the battery capacity, which is only tied to the device consumption and not its software reliability.

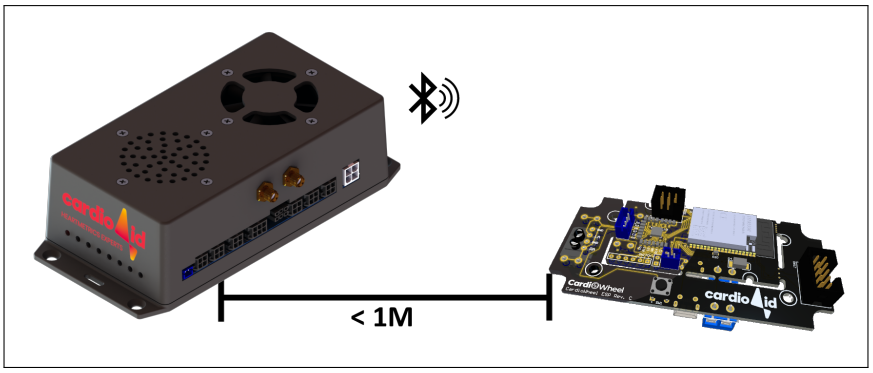


Figure 5.8: Setup test

Table 5.4: CardioWheel Up Time In Seconds - Obtained from testing eight runs

Units	**Mean**	**Max**	**Min**
Seconds	86403.50	89591.00	79074.00
Hours, minutes and seconds	24:00:04	24:53:11	21:57:54

6

Conclusion

In the preceding chapters, we explored the process of designing a reliable and secure architecture for a medical CPS. This dissertation work was driven by the increasing use of CPS for critical applications, especially for medical uses where reliability is paramount. In the medical field, non-communicable diseases (NCDs) accounted for a significant portion of global deaths in 2019, with cardiovascular diseases (CVDs) being a leading cause. Early diagnosis is crucial. Road accidents also rank among the top global causes of death and injuries. To address these challenges, CardioID Technologies utilizes electrocardiogram (ECG) data for CVD diagnosis, monitoring driver drowsiness and biometric identification. Reliable and secure systems are critical for these use cases, especially regarding data accuracy. This chapter overviews the result of the research work and outlines its possible limitations.

To accomplish the objectives of this work, we followed software architectural design and formal validation methodologies, obtaining favourable results for all the heuristics created. After testing and validating the architecture, we created a prototype implementing it and stress-tested it by evaluating how long the system could reliably transmit data. With this test we concluded that in the tests conditions the system was stable to run the entirety of its battery without crashes or communication fails. After this test, we ran some trials at Santa Marta Hospital to test the stability of the sampling frequency and the signal quality obtained by the system compared to a medical ECG machine. These tests consisted of obtaining simultaneously ECG on the CardioWheel and on a Medical ECG machine. The trial concluded that the signal obtained by the

CardioWheel was comparable with a Medical ECG, besides possible baseline wander artefacts, it maintains the features of the signal, giving the possibility to use this signal for the proposed use cases.

With the results obtained in the validation and verification process, we can conclude that all the requirements are met and that the system works reliably and as expected.

6.1 Future work

It is essential to acknowledge the limitations of our study. The heuristics created extend to all the edge cases acknowledged by the authors of this dissertation. This will most likely miss some corner cases, which can cause critical errors during the operation of the system, this can be solved by improving the granularity of the heuristics defined as well as adding more to the list as future work. Other future work that can improve the solution can be exploring the actual energy consumption of the system and implementing measures to extend its operation time, we can also minimise and make the CPU frequency more efficient and appropriate for this application as for this work we are using the maximum CPU frequency available for the MCU (240 MHz). Another thing to consider as future work is minimizing the size of the queue used to transport ADC data through the various tasks available in the implementation, a quick test revealed that the average occupancy rate of this queue is approximately 10% which means that we could certainly reduce its size in an effort to reduce memory consumption.

References

[1] F. Majéric, B. Gonzalvo, and L. Bossuet, "Jtag fault injection attack", *IEEE Embedded Systems Letters*, vol. 10, no. 3, 2018. DOI: 10.1109/LES.2017.2771206.

[2] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen, "Bias: Bluetooth impersonation attacks", in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pages 549–562. DOI: 10.1109/SP40000.2020.00093.

[3] Vishal A. Thakor, Mohammad Abdur Razzaque, and Muhammad R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities", *IEEE Access*, vol. 9, 2021. DOI: 10.1109/ACCESS.2021.3052867.

[4] *Single-lead, heart rate monitor front end*, AD8232, Analog Devices, 2013.

[5] Joseba A Agirre, L Etxeberria, R Barbosa, S Basagiannis, G Giantamidis, Thomas Bauer, E Ferrari, M Labayen Esnaola, V Orani, Johnny Öberg, *et al.*, "The valu3s ecsel project: Verification and validation of automated systems safety and security", *Microprocessors and microsystems*, vol. 87, page 104 349, 2021.

[6] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo, "Gift: A small present: Towards reaching the limit of lightweight encryption", in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, Springer, 2017.

[7] Alberto Battistello, "On the security of embedded systems against physical attacks", Ph.D. dissertation, Université Paris-Saclay, 2016.

[8] Gerd Behrmann, Alexandre David, and Kim G Larsen, "A tutorial on uppaal", *Formal methods for the design of real-time systems*, pages 200–236, 2004.

[9] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks, *Uppaal 4.0*. Los Alamitos, CA: IEEE Computer Society, 2006.

[10] *Ultra-low power sensor hub incl. integrated imu*, BHI160d, Rev. 1.0, Bosch Sensortec, Jun. 2016.

[11] Bluetooth SIG. "Bluetooth technology overview". (Jan. 2023), [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

[12] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe, "Present: An ultra-lightweight block cipher", in *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*, Springer, 2007.

[13] Cambridge, "Cambridge Business English Dictionary", in Cambridge University Press, Nov. 2011.

[14] António Jorge Janicas Cerca, "Fatigue and drowsiness detection using inertial sensors and electrocardiogram", Ph.D. dissertation, Instituto Superior de Engenharia de Lisboa, 2018.

[15] Michael Barr. "Bug-killing coding standard rules for embedded c". (May 2016), [Online]. Available: https://barrgroup.com/embedded-systems/how-to/bug-killing-standards-for-embedded-c.

[16] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers, *The simon and speck families of lightweight block ciphers*, Cryptology ePrint Archive, Paper 2013/404, https://eprint.iacr.org/2013/404, 2013. [Online]. Available: https://eprint.iacr.org/2013/404.

[17] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede, *Rectangle: A bit-slice lightweight block cipher suitable for multiple platforms*, Cryptology ePrint Archive, Paper 2014/084, https://eprint.iacr.org/2014/084, 2014. DOI: 10.1007/s11432-015-5459-7. [Online]. Available: https://eprint.iacr.org/2014/084.

[18] Sourav Das, *Halka: A lightweight, software friendly block cipher using ultra-lightweight 8-bit s-box*, Cryptology ePrint Archive, Paper 2014/110, https://eprint.iacr.org/2014/110, 2014. [Online]. Available: https://eprint.iacr.org/2014/110.

[19] Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov, *Design strategies for arx with provable bounds: Sparx and lax (full version)*, Cryptology ePrint Archive, Paper 2016/984, https://eprint.iacr.org/2016/984, 2016. DOI: 10.1007/978-3-662-53887-6_18. [Online]. Available: https://eprint.iacr.org/2016/984.

[20] World Health Organization. "Cardiovascular diseases (cvds)". (Jun. 2021), [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[21] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen, "Nessie proposal: Noekeon", in *First open NESSIE workshop*, 2000.

[22] Zamira Daw, Rance Cleaveland, and Marcus Vetter, "Formal verification of software-based medical devices considering medical guidelines", *International journal of computer assisted radiology and surgery*, vol. 9, 2014.

[23] Christophe De Canniere, Orr Dunkelman, and Miroslav Knežević, "Katan and ktantan—a family of small and efficient hardware-oriented block ciphers", in *Cryptographic Hardware and Embedded Systems-CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, Springer, 2009.

[24] Ngoc-Tuan Do, Van-Phuc Hoang, and Van Sang Doan, "Performance analysis of non-profiled side channel attack based on multi-layer perceptron using significant hamming weight labeling", in *Industrial Networks and Intelligent Systems: 8th EAI International Conference, INISCOM 2022, Virtual Event, April 21–22, 2022, Proceedings*, Springer, 2022.

[25] Jürgen Dobaj, Damjan Ekert, Jakub Stolfa, Svatopluk Stolfa, Georg Macher, and Richard Messnarz, "Cybersecurity threat analysis, risk assessment and design patterns for automotive networked embedded systems: A case study.", *J. Univers. Comput. Sci.*, vol. 27, no. 8, 2021.

[26] Daniel Engels, Markku-Juhani O Saarinen, Peter Schweitzer, and Eric M Smith, "The hummingbird-2 lightweight authenticated encryption algorithm", in *RFID. Security and Privacy: 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers 7*, Springer, 2012.

[27] Sinem Coleri Ergen, "Zigbee/ieee 802.15. 4 summary", *UC Berkeley, September*, vol. 10, no. 17, 2004.

[28] Rik Eshuis and Roel Wieringa, "Comparing petri net and activity diagram variants for workflow modelling – a quest for reactive petri nets", in *Petri Net Technology for Communication-Based Systems: Advances in Petri Nets*, Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ISBN: 978-3-540-40022-6. DOI: 10.1007/978-3-540-40022-6_16. [Online]. Available: https://doi.org/10.1007/978-3-540-40022-6_16.

[29] João Carlos Santos Fernandes, "Choosing the future of Lightweight Encryption algorithm ", eng, M.S. thesis, 2018.

[30] Mohammed El-hajj, Hussien Mousawi, and Ahmad Fadlallah, "Analysis of lightweight cryptographic algorithms on iot hardware platform", *Future Internet*, vol. 15, no. 2, 2023, ISSN: 1999-5903. DOI: 10.3390/fi15020054. [Online]. Available: https://www.mdpi.com/1999-5903/15/2/54.

[31] *Assigned numbers*, Bluetooth SIG Proprietary, Jul. 2023.

[32] Giann Spilere Nandi, David Pereira, José Proença, José Santos, Lourenço A Rodrigues, André Lourenço, and Eduardo Tovar, "Mars: A toolset for the safe and secure deployment of heterogeneous distributed systems", in *1st International Workshop on Explainability of Real-time Systems and their Analysis at the IEEE Real-Time Systems Symposium (RTSS 2022) in Houston, USA*, 2022. [Online]. Available: https://micro.ros.org/docs/overview/hardware/.

[33] *Guide to cyber threat modelling*, 2021.

[34] Basel Halak, *Hardware Supply Chain Security: Threat Modelling, Emerging Attacks and Countermeasures*. Springer Nature, 2021.

[35] Mohammad Kamrul Hasan, Muhammad Shafiq, Shayla Islam, Bishwajeet Pandey, Yousef A Baker El-Ebiary, Nazmus Shaker Nafi, R Ciro Rodriguez, and Doris Esenarro Vargas, "Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications", *Complexity*, vol. 2021, 2021.

[36] Zoran Hercigonja, "Comparative analysis of cryptographic algorithms", *International Journal of Digital Technology & Economy*, vol. 1, no. 2, 2016.

[37] "Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications", *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, 2016. DOI: 10.1109/IEEESTD.2016.7786995.

[38] Lvar Jacobson and James Rumbaugh Grady Booch, *The unified modeling language reference manual*. The unified modeling language reference manual, 2021.

[39] M Katagi and S Moriai, "The 128-bit blockcipher clefia", Tech. Rep., 2011.

[40] Yash M Kenia, "Cyber attacks on smart cars using sdr", 2019.

[41] Pramod Kumar, Lalit Kumar Singh, and Chiranjeev Kumar, "Suitability analysis of software reliability models for its applicability on npp systems", *Quality and Reliability Engineering International*, vol. 34, no. 8, 2018.

[42] Craig Larman and Don O'Hagan, *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development*. Prentice Hall PTR, 2004.

[43] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm, "New lightweight des variants", in *Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14*, Springer, 2007.

[44] Dean Leffingwell and Don Widrig, *Managing software requirements: a unified approach*. Addison-Wesley Professional, 2000.

[45] N.G. Leveson and C.S. Turner, "An investigation of the therac-25 accidents", *Computer*, vol. 26, no. 7, 1993. DOI: 10.1109/MC.1993.274940.

[46] LoRa Alliance, "Lorawan® distance world record broken, twice. 766 km (476 miles) using 25mw transmission power", *LORAWAN® NEWS*, Jun. 3, 2019. [Online]. Available: https://lora-alliance.org/lorawan-news/lorawanr-distance-world-record-broken-twice-766-km-476-miles-using-25mw-transmission/ (visited on 01/30/2023).

[47] André Lourenço, Ana Priscila Alves, Carlos Carreiras, Rui Policarpo Duarte, and Ana Fred, "Cardiowheel: Ecg biometrics on the steering wheel", in *Joint European conference on machine learning and knowledge discovery in databases*, Springer, 2015, pages 267–270.

[48] LUCAS. "Lucas chess compression system". (Jan. 2023), [Online]. Available: https://www.lucas-cpr.com/.

[49] Eliot Marshall, "Fatal error: How patriot overlooked a scud", *Science*, vol. 255, no. 5050, 1992.

[50] B. Esther Sunanda and P. Seetharamaiah, "Modeling of safety-critical systems using petri nets", *SIGSOFT Softw. Eng. Notes*, vol. 40, no. 1, 2015, ISSN: 0163-5948. DOI: 10.1145/2693208.2693238. [Online]. Available: https://doi.org/10.1145/2693208.2693238.

[51] Giann Nandi, David Pereira, José Proença, José Santos, Lourenço A Rodrigues, André Lourenço, and Eduardo Tovar, "Mars: A toolset for the safe and secure deployment of heterogeneous distributed systems", in *43rd IEEE Real-Time Systems Symposium, RTSS 2022*, 2022.

[52] Rui Qiao and Mark Seaborn, "A new approach for rowhammer attacks", in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016. DOI: 10.1109/HST.2016.7495576.

[53] Alan V Oppenheim and Ronald W Schafer, "Digital signal processing(book)", *Research supported by the Massachusetts Institute of Technology, Bell Telephone Laboratories, and Guggenheim Foundation. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975. 598 p*, 1975.

[54] Jonathan S Ostroff, "Formal methods for the specification and design of real-time safety critical systems", *Journal of Systems and Software*, vol. 18, no. 1, 1992.

[55] Luu Anh Tuan, Man Chun Zheng, and Quan Thanh Tho, "Modeling and verification of safety critical systems: A case study on pacemaker", in *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement*, 2010. DOI: 10.1109/SSIRI.2010.28.

[56] Doron A. Peled, "Formal methods", in *Handbook of Software Engineering*, Sungdeok Cha, Richard N. Taylor, and Kyochul Kang, Eds. Cham: Springer International Publishing, 2019, ISBN: 978-3-030-00262-6. DOI: 10.1007/978-3-030-00262-6_5. [Online]. Available: https://doi.org/10.1007/978-3-030-00262-6_5.

[57] Haidhar Athir Mohd Puat and Nor Azlina Abd Rahman, "Iomt: A review of pacemaker vulnerabilities and security strategy", *Journal of Physics: Conference Series*, vol. 1712, no. 1, 2020. DOI: 10.1088/1742-6596/1712/1/012009. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1712/1/012009.

[58] 2023. [Online]. Available: https://safetyculture.com/topics/risk-assessment/5x5-risk-matrix/.

[59] José Santos, André Lourenço, and Tiago Dias, "Reliability and security in well-being monitoring embedded systems", [Online]. Available: https://www.inforum2023.org/Atas/paper_4887/4887-CM.pdf.

[60] Ali Hasnain, Yame Asfia, and Sajid Gul Khawaja, "Power profiling-based side-channel attacks on fpga and countermeasures: A survey", in *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, 2022. DOI: 10.1109/ICoDT255437.2022.9787473.

[61] Federal Information Processing Standards, "Announcing the advanced encryption standard (aes)", *FIPS PUB*, 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[62] RH Thayer and M Dorfman, *Requirements engineering, edn*, 1977.

[63] Wim Van Eck, "Electromagnetic radiation from video display units: An eavesdropping risk?", *Computers & Security*, vol. 4, no. 4, 1985.

[64] JE Wiggelinkhuizen, GJ Tretmans ESI, and E Hendriksen, "Feasibility of formal model checking in the vitatron environment", Ph.D. dissertation, Master thesis, Eindhoven University of Technology, 2007.

[65] Cindy George. "Micra av pacemaker regulates as a wireless implant in the heart". (Sep. 2020), [Online]. Available: https://www.tmc.edu/news/2020/09/micra-av-pacemaker-regulates-as-a-wireless-implant-in-the-heart/.

[66] Zhidong Zhao and Yefei Zhang, "Sqi quality evaluation mechanism of single-lead ecg signal based on simple heuristic fusion and fuzzy comprehensive evaluation", *Frontiers in physiology*, vol. 9, page 727, 2018.

Threat Modeling

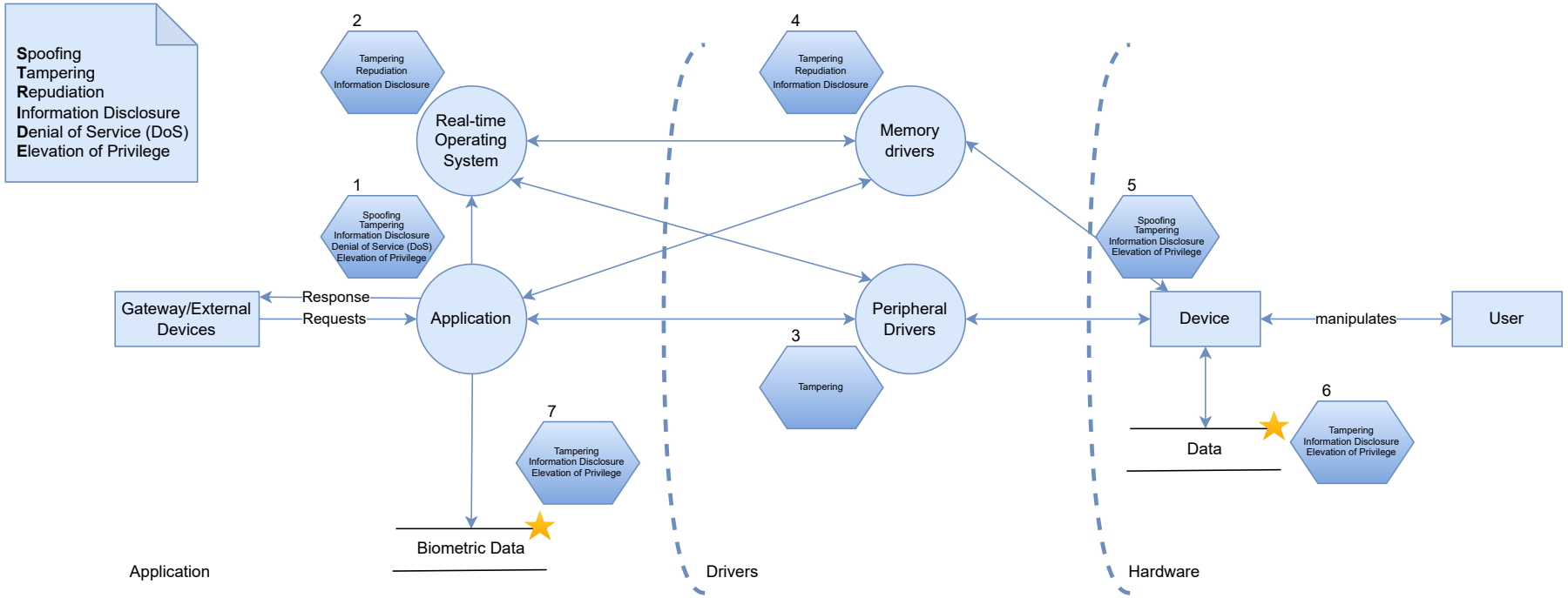


Figure A.1: Threat Modeling Diagram

B

Doxygen generated documentation

This documentation is CardioID's intellectual property, to have access to it contact CardioID Technologies.

ESP32-CardioWheel CardioID

Ver1.1

Generated by Doxygen 1.8.16