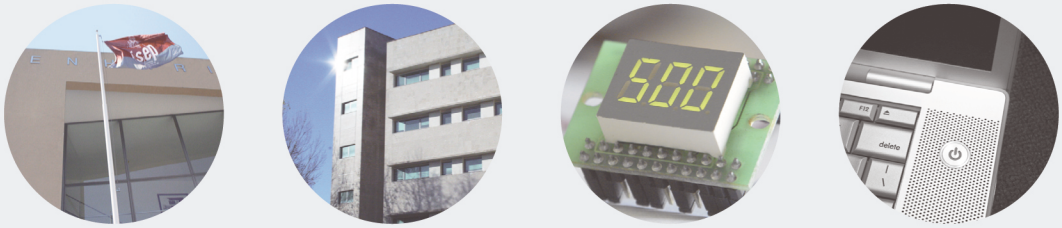
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Utilização da Tecnologia SDR para Autenticação de Dispositivos IoMT Baseada em RF Fingerprint

FILIPE MANUEL MOURA GOMES
julho de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Leveraging SDR Technology for Improved RF Fingerprint-based Authentication of IoMT Devices

Filipe Manuel Moura Gomes

Master in Electrical and Computer Engineering
Specialization Area of Automation and Systems

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

July, 2023

This dissertation partially satisfies the requirements of the Thesis/Dissertation course of the program Master in Electrical and Computer Engineering, Specialization Area of Automation and Systems.

Candidate: Filipe Manuel Moura Gomes, No. 1181059,
11810597@isep.ipp.pt

Scientific Guidance: Dr. Ricardo Severino, rar@isep.ipp.pt

Scientific Co-Guidance: Emmanuel António Carvalhido Lomba,
emmanuel@airlomba.net

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

July, 2023

Acknowledgements

I would like to thank Dr. Ricardo Severino for his guidance throughout this thesis and research work. During the time we worked together, I learned a lot on both a personal and professional level.

I also want to thank Eng. Emmanuel Lomba for all of his support throughout the project, which was critical in overcoming several problems and difficulties.

I want to thank all my friends and girlfriend for all the support, motivation and energy they brought me in this important phase of my life.

I want to thank in special my friend Stéphane, my project colleague, for always keeping me focused and motivating me to be a better person and professional.

Lastly, I want to thank my parents and sister, without your unconditional support none of this would be possible.

Abstract

The increasing reliance on the Internet of Medical Things (IoMT) raises great concern in terms of cybersecurity, either at the device's physical level or at the information and communication level. This is particularly important as these systems process very sensitive and private data, including personal health data from multiple patients such as real-time body measurements. Due to these concerns, cybersecurity mechanisms and strategies must be in place to protect these medical systems, defending them from compromising cyberattacks. Authentication is an essential cybersecurity technique for trustworthy IoMT communications. However, current authentication methods rely on upper-layer identity verification or key-based cryptography which can be inadequate to the heterogeneous Internet of Things (IoT) environments.

In our current research we aim at using Radio Frequency Fingerprinting for IoMT device authentication in medical applications to improve the cybersecurity of such mechanisms. This technique allows the authentication of medical devices by their physical layer characteristics, i.e. of their emitted signal. This shall be accomplished through the use of software-defined technologies, specifically Software-Defined Radio (SDR) gateways to implement security solutions while addressing ecosystem heterogeneity, in conjunction with Artificial Intelligence (AI) Edge/Cloud support to scale and leverage Machine Learning (ML) strategies.

This thesis focuses on the signal acquisition and feature extraction stages of the Radio Frequency Fingerprinting (RFF) process, relying upon SDR technology, which can effectively improve the feature extraction process in complex and challenging wireless environments, to support later highly accurate device authentication. This resulted in the creation of a dataset, which was unavailable before, to train and setup the RFF system.

Finally, the thesis specifies the integration of the RFF system with the medical IoT gateway.

Keywords: IoMT, SDR, RFF, Signal Processing, Feature Extraction, Dataset Creation

Resumo

A confiança crescente na IoMT suscita grande preocupação em termos de cibersegurança, quer ao nível físico do dispositivo quer ao nível da informação e comunicação. Isto é particularmente importante, uma vez que estes sistemas processam dados muito sensíveis e dados, incluindo dados pessoais de saúde de diversos pacientes, tais como dados em tempo real medidas do corpo. Devido a estas preocupações, os mecanismos e estratégias de ciber-segurança devem estar em vigor para proteger estes sistemas médicos, defendendo-os de ciberataques comprometedores. A autenticação é uma técnica essencial de ciber-segurança para as comunicações em sistemas IoMT de confiança. No entanto, os métodos de autenticação atuais focam-se na verificação de identidade na camada superior ou criptografia baseada em chaves que podem ser inadequadas para a ambientes IoT heterogéneos.

Neste trabalho, pretendemos utilizar a impressão digital por radiofrequência para autenticação de dispositivos IoMT para melhorar a flexibilidade de tais mecanismos. Isto permite a autenticação dos dispositivos médicos pelas suas características de camada física, ou seja, a partir do seu sinal emitido. Isto será conseguido através da utilização de tecnologias definidas por software, especificamente gateways de SDR para implementar soluções de segurança em conjunto com AI Edge/Cloud para escalar e alavancar estratégias de ML.

Esta tese foca-se nas fases de aquisição de sinal e extracção de características, com base na tecnologia SDR, que pode efectivamente melhorar o processo de extração de características em ambientes complexos e desafiantes para redes sem fios, de forma a apoiar posteriormente a autenticação precisa de dispositivos. O resultado foi a criação de um dataset, indisponível anteriormente, para treinar e configurar o sistema RFF.

Por último, a tese especifíca a integração do sistema RFF com a gateway IoT médica.

Palavras-Chave: IoMT, SDR, RFF, Processamento de sinal, Extração de características, Criação de dataset

Contents

List of Figures

List of Tables

List of Acronyms

ADB	Android Debug Bridge
AI	Artificial Intelligence
BRISK	Binary Robust Invariant Scalable Keypoints
CGRAN	Comprehensive GNU Radio Archive Network
CNN	Convolutional Neural Network
CTF	Constellation Trace Figure
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DSP	Digital Signal Processor
FFT	Fast Fourier Transform
FHIR	Fast Healthcare Interoperability Resources
FIR	Finite Impulse Response
FPGA	Field-programmable gate array
GPP	General Purpose Processor
GPU	Graphics Processing Unit
GRC	GNU Radio Companion
GUI	Graphical User Interface
HF	High Frequency
HL7	Health Level Seven International
HOG	Histogram of Oriented Gradients
IMD	Implantable Medical Devices

IoMT	Internet of Medical Things
IoT	Internet of Things
IQ	In-phase/Quadrature
LAN	Local Area Networks
LBP	Local Binary Pattern
ML	Machine Learning
OOT	Out of Tree
PLS	Physical Layer Security
PSD	Power Spectral Density
QPSK	Quadrature phase-shift keying
RF	Radio Frequency
RFF	Radio Frequency Fingerprinting
RFID	Radio Frequency Identification
SDK	Software Development Kit
SDR	Software-Defined Radio
SSID	Service Set Identifier
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
UHF	Ultra High Frequency
USB	Universal Serial Bus
VHF	Very High Frequency
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Overview

The IoT is a network of semi-autonomous devices with computational, networking, sensory, and actuation capabilities that connect to the physical world. It combines numerous technologies, such as wireless sensor networks, radio-frequency identification, and machine-to-machine communications. The IoT is becoming a ubiquitous reality, fueling applications in industries such as healthcare. [1].

Context-awareness will be critical in connecting the real world and digital computer entities as the Internet of Things expands. Environmental sensing, network connectivity, and data processing methods are all included. These advancements pave the way for a variety of complex IoT applications, including intelligent healthcare systems, efficient transportation systems, smart energy infrastructures, and intelligent buildings. The topology of IoT networks has been standardised, combining intelligent IoT-based application services with fundamental IoT sensor networks. Wireless networking technologies, as well as the incorporation of emerging technologies such as cloud platforms, are propelling the global expansion in the IoT industry. As a result of this trend, demand for linked IoT devices and application services is increasing [2].

While the number of deployed IoT devices continues to increase year after year, with 75 billion expected by 2025 [3], as shown in Figure 1.1, the number of interconnected devices per network will also increase exponentially. As the number of IoT

devices grows, so does the risk of exploitation, particularly in industrial and medical settings.

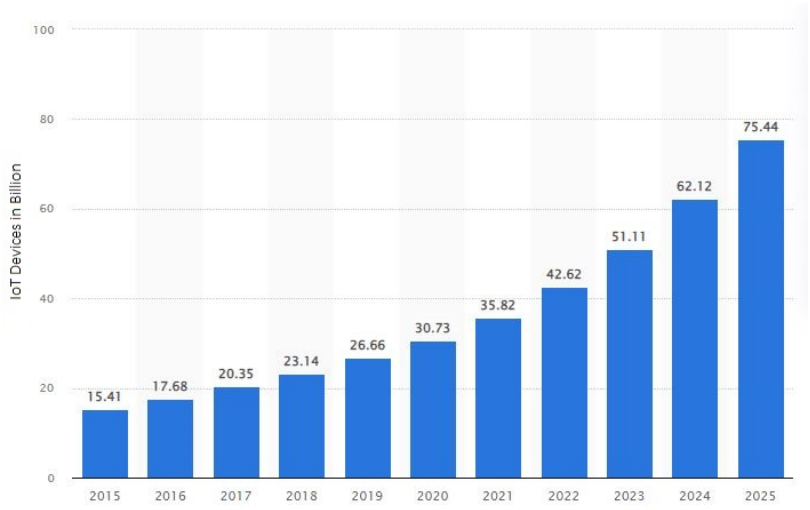


Figure 1.1: IoT Devices Statistics by 2025 [3]

Casual device use, failure to update passwords, and failure to complete regular device upgrades have all increased cybersecurity threats, allowing hostile programmes access to sensitive data within IoT networks. Inadequate security practises increase the likelihood of data breaches and other potential risks. The majority of security experts see IoT as a vulnerable target for cyber attacks due to the lack of security standards and policies in place. Despite the development of several security techniques to protect IoT devices from cyber threats, security rules are poorly documented. As a result, end-users are frequently unable to successfully implement defensive measures to prevent data breaches [4].

Since 2008, hackers have been developing various types of malware to infiltrate IoT devices. They've devised a variety of phishing techniques to trick employees or individuals into disclosing sensitive information. As a result of these high-profile hacks, business workstations and personal devices are routinely breached in terms of privacy. If device manufacturers and security experts identify these cyber threats correctly, they will be able to design and implement effective defensive mechanisms to either prevent or mitigate these cyber threats [4].

Unfortunately, most internet-connected IoT devices do not have the same level of resistance to infiltration, hacking, and sabotage attacks as other computing devices have developed over time. They, on the other hand, show a high level of vulnerability. Surprisingly, Figure 1.2 shows that nearly 70% of IoT devices have major security flaws, such as unsecured network services and weak password restrictions [5].

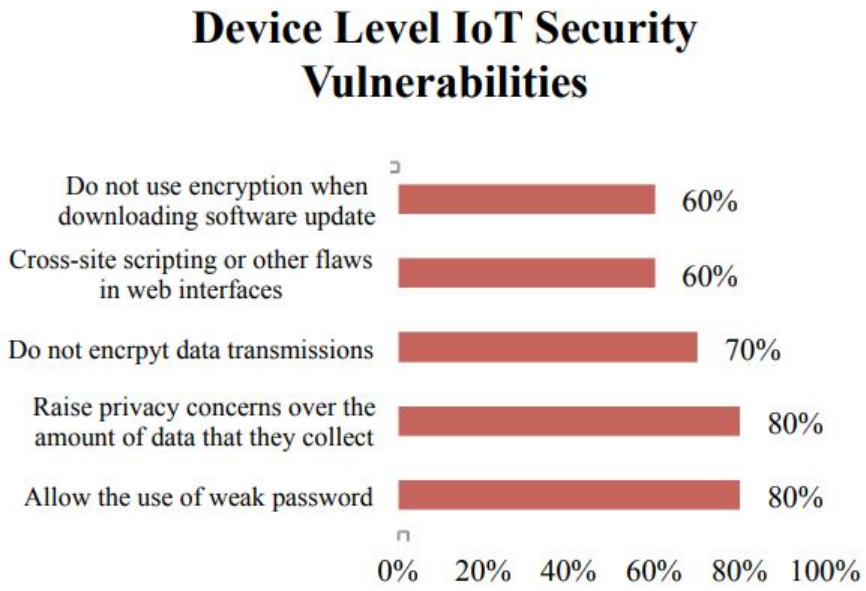


Figure 1.2: Device Level IoT Security Vulnerabilities [5]

The IoMT is an internet-connected device network that aims to improve healthcare services. IoMT uses small wearable or implantable sensors to collect vital physiological data and monitor patients' pathological details. It has numerous applications, ranging from implantable medical devices to wireless body area networks [6]. IoMT also has an impact on enhanced drug and equipment control, telemedicine, mobile healthcare, and health data management. By utilising technology such as Radio Frequency Identification (RFID) tags, it provides real-time monitoring, ensures authenticity, and aids in medical waste management. It has improved hospital information administration and is crucial to health management. Remote patient monitoring via implantable sensory nodes is revolutionising healthcare services [7].

Figure 1.3 depicts the increase in the number of active IoT connections in the European Union's healthcare sector for 2016, 2019, 2022, and 2025. The graph shows that these relationships have grown steadily over time. It is predicted that by 2025, there will be 10.34 million connections, up from 0.87 million in 2016 [8].

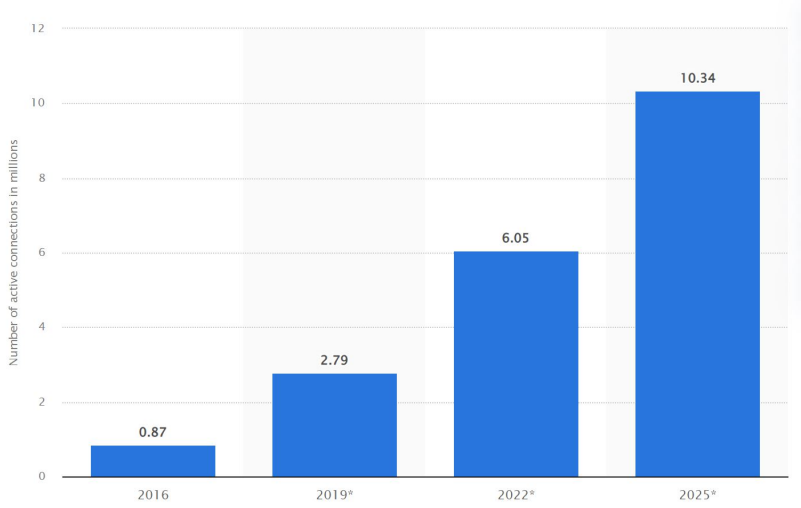


Figure 1.3: Number of active connections in medical IoT in the European Union [8]

These IoMT devices have begun to play an important role in extending human lives, thanks to advancements in 5G communications and IPv6 addressing. A number of security concerns arise when sensitive sensor data is transmitted from one medical IoT device to another. These IoMT devices are vulnerable to eavesdropping, hijacking, denial of service, message manipulation, device cloning, parameter configuration change, and power denial attacks. IoMT has limitations such as limited battery capacity, memory, and computing power. These constraints have the most significant impact on interoperability, availability and security. As a result, implementing secure IoT in the healthcare industry is critical [9].

Cyberattacks on IoMT equipment can have serious consequences ranging from data breaches to life-threatening disinformation. Denial of service attacks flood the system with traffic, causing data inaccessibility and tampering with patient information, which can lead to incorrect treatments and even patient fatalities. Router attacks jeopardise the safe transmission of critical healthcare data. Select Forwarding attacks involve the selective discarding of data packets, resulting in incomplete data and the possibility of misdiagnosis at the receiver's end. Sensor assaults can involve attackers replacing sensors with malicious ones, resulting in data manipulation and insertion of false data. An attacker impersonates legitimate data transmissions in replay attacks, resulting in false recognitions or duplications. Each of these attacks has the potential to result in unauthorised access, data manipulation, denial of continuous monitoring, data route changes, and data drops [10].

In the field of IoMT, where sensitive data is transferred on an unprecedented scale, the importance of strong authentication techniques cannot be overstated. Adoption of Radio Frequency Fingerprinting (RFF), which capitalises on each transmitter's unique, intrinsic properties during transmission, is a viable option. These

manufacturing-related characteristics distinguish each transmitter regardless of brand, model, or serial number [11]. As a result, datasets containing these distinct emissions from multiple devices can be compiled, providing a rich foundation for training machine learning algorithms. These algorithms can recognise and categorise devices based on their unique fingerprints, adding an extra layer of security.

Despite the potential associated with RFF techniques, there is an evident scarcity of Radio Frequency (RF) signal transmission datasets. This reveals a serious problem in this area, since these datasets are essential to train robust Machine Learning (ML) algorithms capable of efficiently identifying the emitting devices. Furthermore, existing software for Software-Defined Radio (SDR) implementations, such as GNU Radio and Matlab, do not have modules that directly perform the necessary processing for recording RF datasets. Additionally, it is necessary to develop ways to integrate RF techniques into relevant IoT systems that benefit from this implementation. This would reinforce the idea that these cybersecurity techniques do indeed have potential application in IoT systems.

This thesis is motivated by the need for greater security in IoMT systems, as well as the promise that RFF brings in this context. It proposes the development of an RFF-based architecture for identifying and authenticating IoMT devices. This work seeks to improve the security mechanisms of these systems through integrating an RFF approach into an IoMT infrastructure, thereby providing a secure mechanism for device authentication that complements existing protocols. The thesis will concentrate on the signal acquisition and feature extraction stages, relying on SDR technology to improve the feature extraction process in complex and demanding wireless situations, in order to facilitate highly accurate device authentication.

1.2 Research Context

This thesis is inline with the ForPharmacy and iCare4NextG projects, which are being carried out in the Porto Research, Technology, and Innovation Center (PORTIC) of the Polytechnic Institute of Porto (IPP) and Glintt. In which PORTIC was in charge of designing a cybersecurity technology that can be used in these two IoMT framed projects.

1.2.1 ForPharmacy Project

Because of their intimate interaction with the public, pharmacists are critical first-level healthcare points. With increased healthcare expenses, an ageing population, and an increase in the number of chronic diseases, this link is becoming increasingly crucial in tackling existing and future health issues, such as the COVID-19 pandemic. Telepharmacy, a type of telemedicine in the pharmacy sector, can open up new avenues for health-care delivery by allowing pharmacists to give distant care to

patients. The ForPharmacy project seeks to investigate and develop telepharmacy solutions in areas such as pharmacovigilance and therapeutic adherence, with the goal of increasing access to pharmacy care and improving patient safety through digitalization and improved communication among healthcare professionals.

1.2.2 iCare4NextG Project

Using data-driven methodologies, the iCare4NextG project intends to establish a service framework for better home care and wellbeing. To offer adaptable user solutions and a supportive business environment for new digital business models, a multi-track strategy will be used, combining service framework development, business modelling, and use case solutions. Prediction, prevention, personalization, involvement, integrated care, and home care will be central to the framework. The ageing population and rising chronic sickness necessitate functional preventive care at home, and the platform will be built on integration, standardisation, modularization, and flexibility principles. With the Covid-19 outbreak emphasising the necessity for home care, a number of current instances will drive platform development and demonstrate its viability. The framework includes abstraction layers for data gathering, management, and exploitation, enabling for the construction of customizable solutions using data from many sources.

In both projects, we are in charge of developing the medical gateway, connecting medical devices to it, as well as connecting it to cloud services for further communication with Glintt platforms. The work done for the thesis is significant because the developed RFF security mechanisms added authentication as a new component to the medical gateways in both projects.

1.2.3 Thesis Structure

This thesis begins with an introduction (Chapter 1), which presents a theoretical introduction to the subject, followed by the situation at hand and how this Thesis tackles the mentioned security concerns. In Chapter 2, there is a research foundation of the RFF that covers essential RFF concepts. Following that, in Chapter 3, this thesis describes some of the technologies used to lay the groundwork for the developed RFF architecture. Chapter 4 shows how the RFF architecture is integrated and how it works in the medical gateway. The Signal Acquisition and Feature Extraction Architecture, which is the main emphasis of this thesis, is presented in Chapter 5. The outcomes of the created work are disclosed and addressed in Chapter 6. Chapter 7 demonstrates the conclusions to be drawn from the established architecture, as well as a few suggestions for future work.

Chapter 2

Research Background

This chapter introduces the key concepts and background relevant to this work, as well as an overview of the most recent research on these topics.

2.1 Overview of Healthcare Communication Security

IoT is a collection of technologies that enable a wide range of applications, devices, and objects to connect with one another via network technology. IoMT is a primary application of IoT, and it offers a variety of services to the healthcare business, including Implantable Medical Devices (IMD), RFID Tags, and wearable medical devices. The transmission of critical IoMT data from one device to another raises many security concerns. Side-channel attack, tag cloning, denial of service, tampering devices, rogue access, sniffing, and brute force can target these devices, violating patient data and also raising safety concerns. However it is challenging to address these as IoMT has restrictions such as limited battery capacity, memory, and computational power [9]. We further elaborate on these attacks and prospective solutions in the next sections.

2.1.1 IoMT Requirements

IoMT is a dynamic ecosystem of linked medical tools, services, and procedures that collaborate to improve the provision of healthcare. To protect data and guarantee the system's efficient operation, it is crucial to create and maintain specific requirements as this sophisticated network continues to expand. These specifications are

intended to protect the privacy and confidentiality of patient data, ensure data integrity, ensure system availability, confirm user actions through non-repudiation, verify user identities through authentication, manage user permissions through authorization, and maintain anonymity when necessary. The following are the details of these needs in further detail [12]:

- Confidentiality: In the context of the IoMT, confidentiality refers to the protection of patient health data exchanged with medical experts. This information must be safeguarded against unauthorised access, suspicious surveillance, and malicious users who may harm the patient or exploit the data. Despite broad advice from IoMT standard protocols, it is critical to implement network access control and data encryption to ensure confidentiality.
- Privacy: In the context of IoMT, privacy refers to the safeguarding of confidential patient information against unauthorised disclosure and nefarious use. Currently, many nations have laws in place governing the gathering and archiving of patient health data, such as the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR). IoMT systems uphold these privacy laws and give users access to their private information.
- Integrity: Protecting patient data from unauthorised additions, deletions, or revisions is essential for IoMT healthcare systems. By doing this, the data is protected from manipulation both during wireless transmission and when it is at rest. Healthcare organisations are becoming more and more aware of the value of data integrity as medical data is used to depict diagnosis, treatments, and health problems. This integrity must be preserved through safeguards against unauthorised data changes.
- Availability: The readiness of servers and medical equipment to offer users services and data as needed is referred to as availability. This is especially important in healthcare systems because it's required to monitor patients continuously. Systems should have backup plans in place for uncertain data storage or transmission routes, especially against Denial of Service (DoS) or Distributed Denial-of-Service (DDoS) attacks, to ensure availability.
- Non-Repudiation: Non-repudiation refers to the system's ability to hold users accountable for their actions and prevent them from retracting previous system usage. It evaluates the system's ability to confirm or deny the occurrence of an activity. Non-repudiation is usually achieved through the use of digital signature methodologies.
- Authentication: A user's identity must be verified when they log into the system. Verifying that a user is really the original source of the delivered data

is the goal of message authentication. The most secure type of authentication is mutual authentication, which calls for client and server authentication before secure data transfer. Due to resource restrictions in some IoMT devices, lightweight authentication techniques are growing in popularity.

- Authorization: Given the sensitive nature of medical data, it is vital to prevent unauthorised access. Only trustworthy entities with the necessary credentials should be granted authorization to perform specific tasks, such as sending commands to IoMT devices or updating their software or security patches.

- Anonymity: This criteria makes sure that the identity of the patient or doctor is kept provate when speaking with unauthorised system users.The names of the doctor and patient should be kept secret during their interaction. Passive attacks should only expose behaviour of the user, not their identity.

2.1.2 IoMT Architecture Layers

IoMT systems typically have three main layers as depicted in Figure 2.1: perception, network, and application. These layers cover all of the steps through which data passes, from data collection via sensors and wearable devices to storage, processing, and visualisation by the patient and medical staff.

The perception layer is the IoMT architecture's foundational layer. This layer ensures the exact sensing of health indicators. To read and gather data about the patients, several sorts of sensors could be employed, particularly those that can be implanted or attached to the body, such as a pacemaker, smart watch, and so on.

The acquired data is then delivered to the network layer as raw values using various communication protocols. This second layer transmits data to processing units (i.e., the cloud) via the IoT gateway.

The data analysis is subsequently performed at the cloud, which represents the application layer. If certain changes in the patient's health data are noticed, this has significant implications. The changes are then presented to the physicians for emergency response (such as quantity, prescription, or change of dosage of different medications) and to the patients via remote monitoring, such as in a smartphone or a dedicated access point [12].

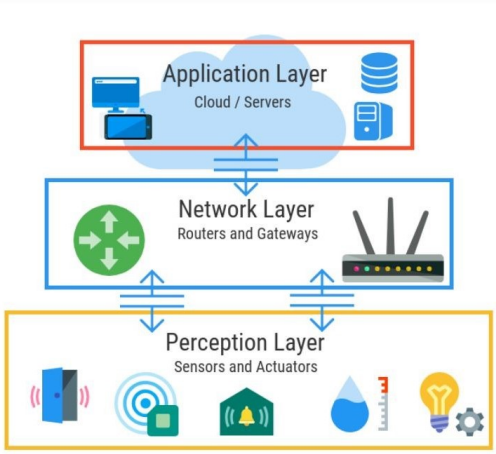


Figure 2.1: IoMT Architecture Layers [13]

2.1.3 Security Breaches in IoMT

There are numerous potential attacks in the IoMT space. These attacks differ not just in their strategy and methodology but also in how they affect the IoMT requirements. The Perception, Network, and Application layers of the IoMT architecture are the three categories that can be used to group these possible threats. The tables that follow give a brief summary of these attacks, their descriptions, and the areas of the requirements they affect [12]:

Table 2.1: IoMT potential attacks at Perception layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Side-channel attack	The side channels of the encryption device are used to gather data.	Confidentiality, Integrity
Tampering devices	To change the data (modification in a device utilising RFID or a communication link), the IoMT device is physically accessible.	Confidentiality, Integrity
Tag cloning	An attacker may copy data from a tag that has already been used or utilise data that was gained through a successful side-channel attack. The cloned tag might be used to get access to a restricted location or data, including private medical records.	Confidentiality, Authorization, Integrity
Sensor tracking	This attack violates the privacy of the patients. Through unprotected devices, attackers may acquire patient locations or create phoney GPS data. Sensitive patient information may also be revealed by other sensors, including those for wheelchair management, fall detection, and remote monitoring systems.	Confidentiality, Authorization, Integrity, Privacy

Table 2.2: IoMT potential attacks at Network layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Eavesdropping	A hacker intercepts and follows the hardware and communications needed to get data. There are numerous uses for the data that was taken illegally.	Confidentiality, Non-repudiation, Privacy
Replay	An attacker can intercept a signed packet and send it many times by using an authentication message that was previously sent between two authorised users.	Authorization
Man-in-the-middle	In order to gain access to their private data, this cyberattack attacks communication between two IoMT devices. Before the intercepted data is transferred to its intended recipient, the attacker can see it or change it.	Confidentiality, Authorization
Rogue access	In this attack, a fake gateway is placed within the wireless network's coverage area to grant access to authorised users and intercept traffic.	
DoS/DDoS	A DDoS attack, which differs from DoS attacks from a single node in that it comes from several sources, floods a target with messages or connection requests in order to prevent genuine users from using the service.	Availability
Sinkhole	In this attack, a rogue node draws traffic by promising improved connection quality. Other attacks, including eavesdropping or selective forwarding, can be initiated after a successful execution.	
Sniffing	Data being exchanged between two nodes is passively intercepted during a sniffing attack. This enables the attacker to watch while data is transferred across system layers.	Confidentiality
Selective Forwarding	A rogue node may change, delete, or only forward particular messages to other nodes in the network during an attack. As a result, the recipient only receives partial information.	All

Table 2.3: IoMT potential attacks at Application layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Brute Force	Automated programmes are frequently used by attackers to create a variety of password combinations until they succeed. IoMT devices are seriously vulnerable to the dictionary attack.	Confidentiality, Integrity
SQL injection	In this attack, a flawed SQL statement is introduced into the application's backend database. Sensitive patient data may be compromised or changed in a successful attack.	All
Account hijacking	Many IoT devices interact across networks in transparent text or with poor encryption. An attacker can steal an account by intercepting the packet during end-user authentication.	Confidentiality, Integrity
Ransomware	Ransomware encrypts important data and requests a ransom to decrypt it. Attackers can encrypt private data, such as health records, and keep the decryption key in exchange for money.	Integrity, Availability

2.1.4 Security Solutions

Although there are various methods for attacking IoT devices, several methods for reducing or even cancelling some attacks have been devised. There are three types of solutions to consider: communication layer, network routing, transport layer and application layer.

Communication Layer Security Solutions

A fundamental part of deploying an IoT network is ensuring secure end-to-end connectivity. One proposed method for enabling such security is to use compressed IP security. The Encapsulation Security Payload (ESP) and the Authentication Header (AH) are used in this method to authenticate and encrypt messages between the regular internet and the sensor network. The results show that employing established IPv6 techniques, compressed IPsec can assure message integrity. Furthermore, they developed an ESP for IPsec/6LoWPAN, which they compared to link-layer security for IEEE802.15.4.The authors discovered that while IPsec delivers greater security and has a faster response time than link-layer security, it consumes more energy than link-layer security when tested on an IEEE 802.15.4 testbed [14].

Another protocol for IoT security was proposed, which included a feasible authentication solution based on Elliptic Curve Cryptography and leveraging a secure key function (ECC). They also implemented Role-Based Access Control (RBAC) in IoT network applications for access control rules. However, the proposed security

analysis was not realistically applied, and IoT sensor nodes had a large connection overhead [14].

Network Routing Security Solutions

The proliferation of IoT devices has increased the demand for security solutions to protect these devices from cyber-attacks. Despite the implementation of encryption and authentication, IoT networks remain vulnerable to a variety of attacks, including sybil, black-hole, sinkhole, fragmentation, selective-forwarding, and man-in-the-middle attacks. Intrusion detection systems (IDSs) that can be put at every node of Low-power and Lossy Networks have been proposed to detect these attacks (LLN). However, because the nodes have limited resources, IDSs must be optimised for each physical object [14].

Some researchers have proposed lightweight intrusion detection systems (IDSs) based on harmful pattern detection, which evaluate detection in terms of energy usage and execution time using two schemes: early detection and auxiliary shifting. Another option is INTI, a distributed IDS architecture that combines the "watchdog" and "reputation and trust" approaches to find and mitigate attacks [14].

Others have developed hybrid intrusion detection systems to detect selective forwarding, sinkhole, and wormhole attacks. These IDSs are made up of primary IDS modules, a 6LoWPAN mapper, and a mini-firewall at constrained devices and border routers. To identify attacks, the proposed IDSs employ a variety of techniques such as monitoring routing traffic, evaluating signal intensity, and identifying aberrant activity [14].

While these IDSs have yielded encouraging outcomes, they are not without limits. Some intrusion detection systems, for example, have a high false-positive rate, spend more energy during an attack, or have decreased accuracy as network size increases. As a result, the selection of an appropriate IDS is determined by the unique attack scenario and resource restrictions [14].

In one study, a network is divided into numerous clusters and monitored as a group. The IDS is positioned at the top of each cluster and is responsible for monitoring the cluster based on the set role. This minimises computation and storage, and their results suggest that IDS can detect network threats more effectively with a small amount of overhead [14]. Another study proposes an efficient, secure route optimisation protocol for the Proxy Mobile IPv6 (PMIPv6), which improves on the existing routing protocol (PMIPv6) in terms of security, particularly authentication, complete forward secrecy, key exchange, and privacy when supporting the protocol mutual. Their approach assures secure transmission while minimising packet loss, latency, and delay [14].

A third paper suggests a novel trust mechanism based on the SecTrust-RPL routing protocol, which was deployed in a test bed experiment. The suggested

technique detects both sybil and rank attacks. The authors, however, did not assess the impact of their approach on performance, energy consumption, and performance overhead [14].

Another paper describes a wormhole attack detection technique for the RPL IoT routing protocol. Their intrusion detection system (IDS) is installed at both the border router (BR) and the host sensor nodes, and they analysed the influence of their IDS on the success rate of detecting the wormhole attack. The results showed that with eight sensor nodes, the true positive rate was high, but fell as the number of sensor nodes increased.

Another study presented a hybrid intrusion detection system (IDS) that targets sinkhole and cloneID threats and analysed its IDS impact in terms of performance and detection. The detection rate, according to the authors, was 100%. However, the statistics show that as the number of sensor nodes increased, the detection rate declined, and the energy and power consumption was higher than the prior IDS they tested [14].

Another study utilised an RPL-based IoT with genetic programming to develop an intrusion detection framework and architecture. Their detection strategy focuses on two types of attacks: version number and hello flood attacks, and their detection system detects intrusions with high accuracy and low false positives.

Transport and Application Layer Security Solutions

Some research articles' authors performed a comparison investigation of the security aspects of MQTT and CoAP, focusing on the transport protocols used, specifically MQTT with TLS and CoAP with DTLS. The investigation took into account security mechanisms such Raw Public Key, Certificates, and Pre-Shared Key. They discovered that MQTT provides a diverse security alternative to lightweight and PSK certificates, but certificate-based encryption and authentication remain the most secure [14].

One of the papers examines the use of RSA cryptography on sensing devices through the use of trusted-platform modules (TPM). The authors used a DTLS cypher suite TLS-RSA-with-AES-128-CBC-SHA to evaluate the system in terms of latency, energy usage, and memory. Another publication investigated this proposal further in wireless sensor networks (WSN). Another proposal for a transport layer authentication security scheme based on the ECC algorithm [14].

One study developed Lithe, a DTLS/CoAP integrated protocol that uses a compression algorithm to reduce header overhead and is more efficient than conventional CoAP/DTLS. However, it is vulnerable to DoS attacks. Another paper describes E-Lithe, a lightweight DTLS for IoT that customises the DTLS packet to reduce energy consumption and execution time. Another proposed system combines identity-based encryption and a Diffie-Hellman encryption algorithm to provide authentication and

integrity security without the use of a digital certificate. Another study tries to improve the CoAP protocol's hash function for integrity security in smart home applications, and a novel architecture suggests the use of cryptography methods rather than DTLS to ensure data confidentiality and integrity in CoAP-based smart home apps [14].

2.1.5 Towards Methods of Authentication

Wireless communication networks have grown in popularity and development in recent years. Attacks that leverage information security vulnerabilities such as identity impersonation, replay attacks, and device cloning continue to emerge. Accurately identifying and authenticating the objects of the IoT is the primary problem.

Due to many key obstacles, such as high computational complexity, high communication overheads, high latency, and inadequate security of resource-constrained mobile IoT devices for healthcare applications, authentication with typical cryptographic approaches does not appear to be practical. Key management is only appropriate for classical cryptosystems. Traditional cryptography systems' static properties make it impossible to detect unauthorised organisations reusing compromised keys.

Additionally, mechanisms like this have risks of protocol security loopholes and key leakage. Key leakage and security issues discovered at the application layer can be mitigated by physical layer authentication. Physical layer authentication technology offers more solutions to wireless communication security issues. Physical layer security authentication technology research is still in its early phases, and its core theory has not kept pace with other wireless communication technologies' progress.

5G and beyond networks have enabled rapid progress in the development of IoT devices on diverse platforms. Because of their low cost and small size, such devices are useful for a wide range of applications [15].

The use of ML authentication approaches on the physical layer has piqued the interest of academia and industry. This approach smartly and efficiently secures 5G-enabled portable devices. For authenticating entities, ML-enabled approaches such as supervised and unsupervised learning and Deep Learning (DL) are useful. Because machine learning approaches take into account unique multidimensional qualities, they can provide more secure, better authorised, fairly accessible, highly dependable, dynamic, and self-driven device verification for an unknown network state. Poor security and lengthy delays can have an impact on the functioning of vital applications like healthcare. Such challenges can be addressed with a quick cross-authentication solution that makes use of SDN-aware joint cryptographic and non-cryptographic features [15].

All of the previously mentioned concepts might be realised by implementing RFF authentication, which would strengthen the safety measures of those systems without limiting the performance of the IoMT devices [15].

As an example, WiFi has become a common communication medium for linking numerous wireless devices in Local Area Networks (LAN) and the IoT due to its ease of setup. Unfortunately, because of the openness of radio transmission, the disclosed security issues have become increasingly critical. Insecure authentication methods, implementation attacks, side channel attacks, impersonation, and the replay attack are among the security issues. Because these attacks can occur in and below the data link layer, soft-identifiers such as passwords, Service Set Identifier (SSID), and/or MAC/IP addresses are vulnerable to spoofing. As a result, it is critical to develop an effective approach for detecting and preventing rogue WiFi connections [16].

Physical Layer Security (PLS) has recently emerged as a promising paradigm for securing WiFi connections. PLS uses device and wireless channel characteristics to authenticate users and encrypt communication data. RFF is one of the PLS technologies that is being used to supplement existing multifactor authentication techniques at the device level in order to combat forgery and related dangers [16].

2.2 Radio Frequency Fingerprint

The method of recognising the hardware characteristics and specific features or signatures inherent in radio frequency waves broadcast via a wireless channel is known as RFF. The manufacturing process for chip components produces hardware flaws that make each emitter unique, regardless of brand, model, or serial number [17]. These flaws are necessary to make the cloning process challenging. Using RFF to discriminate between illicit and legitimate devices is a new physical layer strategy for protecting communication system security. RFF can be utilised for physical identification and access authentication of wireless equipment, just like everyone has their own unique biometric fingerprint characteristics. Figure 2.2 depicts a typical RFF identification.

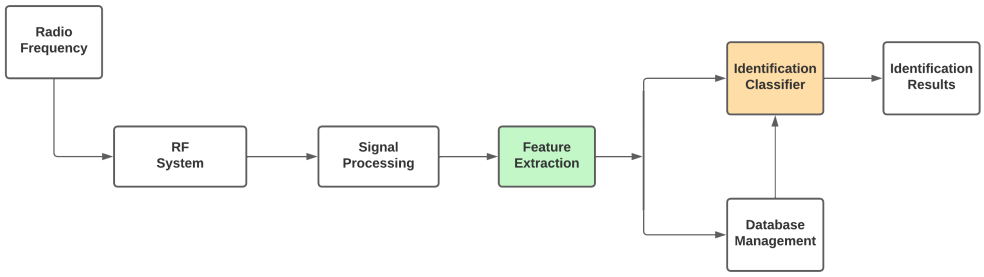


Figure 2.2: Typical RFF identification process

The procedure pulls features from a given radiation source from the received signal time series for classification and recognition, which is fundamentally a pattern recognition problem, as shown in figure 2.2. Looking at the diagram again, we may resume the process as follows [18]:

1. The RF signal segment utilised to extract the radio RFF is intercepted and preprocessed.

2. Following the acquisition of the signal to be classified, the RFF can be obtained using various transform domain approaches such as frequency domain analysis, time-frequency analysis, fractal, high order spectrum, constellations, or RFF can also be recovered in the modulation domain.

3. The deep learning method can also be used to process the signal directly. As it can perform some feature selection or feature dimensionality reduction, as well as feature extraction and classifier creation.

2.2.1 Signal Processing

The RF signal is intercepted and preprocessed, as indicated in Section 2.2. In recent years, Software-Defined Radio (SDR) have been employed in this step. A SDR is a programmable transceiver that may operate numerous wireless communication protocols without the requirement for hardware changes or updates. Progress in the SDR sector has resulted in a wide range of protocol development and applications, with a higher emphasis on programmability, flexibility, portability, and energy efficiency in cellular, WiFi, and M2M communication [19]. SDR have two benefits over lab-grade receivers for RFF. The first is cost; lab-grade receivers can cost up to twice as much as SDR. The second is general-purpose accessibility. SDR is made to offer a base and modular capacity that can be improved by building processing chains out of functional blocks to quickly add or integrate additional features [20].

In phase/Quadrature Samples

In-phase/Quadrature (IQ) samples are a representation of the preprocessed and recorded RF signal. Comparing the "in-phase" component to the "quadrature" component, a phase shift of 90 degrees has occurred. Each sample can be expressed as a complex number, where the real part increases the in-phase component and the imaginary part increases the quadrature component.

The IQ convention is an alternate approach to describe magnitude and phase of a RF signal, which leads to the use of complex numbers and the ability to represent them on a complex plane. A complex number also has a magnitude and phase, which makes more sense when viewed as a vector rather than a point. This representation is better explained with an example. Consider the Figure 2.3 and imagine that the

point $0.7-0.4j$ is to be transmitted. The following equation can be used to represent the resulting signal to be transmitted.

$$x(t) = I\cos(2\pi ft) + Q\sin(2\pi ft) = 0.7\cos(2\pi ft) + -0.4\sin(2\pi ft).$$

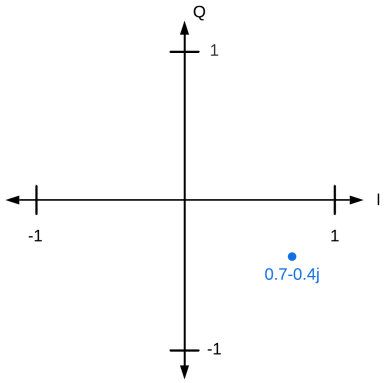


Figure 2.3: IQ complex number representation

A trigonometry identity, $a\cos(x) + b\sin(x) = A\cos(x - \phi)$, can be employed, where A is our magnitude discovered with $\sqrt{I^2 + Q^2}$ and ϕ is the phase, equal to $\tan^{-1}(Q/I)$, resulting in the equation:

$$x(t) = 0.806\cos(2\pi ft + 0.519).$$

Despite the fact that it begins with a complex number, the real component is conveyed, which is required because the imaginary component is not transmitted with electromagnetic waves. The complex numbers are useful for obtaining and plotting the features required to proceed in the fingerprint process [21]. The SDR collects the RF signals, but these captures must be passed to a computing device. There are numerous ways to do this step, ranging from writing code to capture and process these data to using software that already includes a major percentage of the necessary functionality, such as GNU Radio [22].

2.2.2 Feature Extraction

Following the acquisition of the signals, the following phase involves the extraction of unique features from various portions of the signal. There are various techniques for extracting multiple elements from an RF signal. According to [23], these features, as shown in Figure 2.4 can be classified into three types: transient-based, steady-state-based, and other approaches based on various signal portions employed for feature extraction.

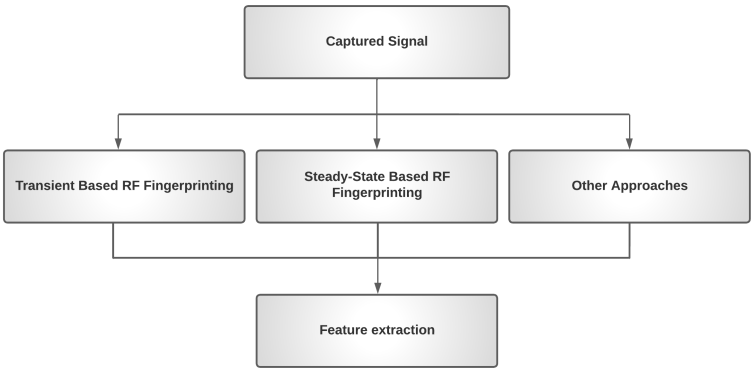


Figure 2.4: Feature Extraction

Transient-based RF fingerprinting techniques make use of the transition between turning on and off a transmitter prior to the actual transfer of signal data. This method necessitates precise transient extraction (start point and duration). Transient extraction methods are heavily influenced by signal noise, hardware, and the distance between the SDR and the device.

Only when the transient is precisely retrieved does transient-based analysis provide high reliabilty regarding the fingerprint procedure. The lack of transient analysis can make distinguishing devices from the same manufacturer more difficult (same model). For good transient extraction, very high sampling rates are required, necessitating expensive receiver structures, and isolating the transient signal and locating the start point in channel noise are extremely difficult due to non-stationary properties [24].

The unique features retrieved from the modulated component of the signal are the focus of steady state-based techniques. For physical layer identification, mostly five unique aspects of the modulated signal are used: frequency error, synchronised correlation, IQ origin offset, and amplitude and phase faults [25]. The following considerations must be made: The steady-state signal is not shared by all emitters. The transient signal is constantly there throughout transmission. However, as previously indicated, a greater sample rate is needed to retrieve the transient signal.

In comparison to the other methods, they usually require unique wireless technology and/or extract different signal and logical layer features, some of which will be detailed and explained:

- In [26], a Constellation Trace Figure (CTF) feature is shown. In practise, the non-linear behaviour of the amplifier at the transmitter will significantly distort the RF signal, resulting in non-linear signal offset at the receiver, including IQ offset and phase offset. These characteristics are often taken from the traditional constellation map. However, there is a flaw in the latter; it only represents the samples at the decision point. As a result, the CTF method was

introduced, in which the base point of the IQ axis on CTF is established as a central point and the CTF is equally split by a fixed angle from the central point.

- A technique was developed in [27] to give further protection against spoofing attacks. It focuses on identifying authorised objects and using their fingerprints to detect prospective intruders in the wireless environment by analysing the Power Spectral Density of physical signals.

2.2.3 Machine Learning Approaches for Identification and Classification

Depending on the ML method and features used during training, different results will be produced. As a result, modifications to feature extraction, signal processing, or the machine learning method itself may be required. Below are three examples that demonstrate this process:

Differential Contour Stellar-Based RFF

The fingerprinting of devices using the WiFi communication protocol is covered in [28]. They employ images created from signal capture to train a machine learning system to determine which device the image belongs to. These images are created by a differential process of in-phase and quadrature signals. Following the differentiation of the signal, the production of the designated differential contour stellar, as shown in Figure 2.5, was carried out.

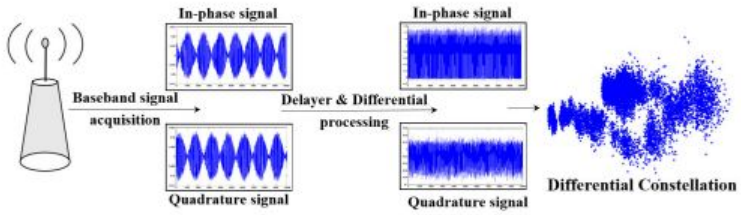


Figure 2.5: Creation of differential Contour Stellar Image [28].

Following the generation of the images, 400 images were picked at random from each class, 320 were used to train the Deep Neural Network (DNN), and 80 were utilised for validation. For the 20 WiFi devices utilised in this testing, this resulted in a recognition success rate of 90.4%.

Imaging Time Series for Internet of Things Radio Frequency Fingerprinting

The RFF process in IoT communication devices is highlighted in [29], with 9 nRF-24LU1+ devices used to construct IoT sensor networks. The way authors approach to fingerprinting is centred on transforming time series into images. The distance between two locations in the time series, from the start of the burst to a different amount of samples, is iterative. Three complex number distances were used: order 2 Minkowski distance (Euclidean distance), order 3 Minkowski distance, and the Chebyshev distance. The generated images are then processed to extract basic image processing features such as Histogram of Oriented Gradients (HOG), Binary Robust Invariant Scalable Keypoints (BRISK), Speeded-Up Robust Features (SURF), and Local Binary Pattern (LBP).

Different accuracies were obtained as a result of the three distances employed in the complex numbers, as well as the amount of samples used for each class. The precision for Minkowski order 3 (size 500 samples) was about 98.36%, for Minkowski order 2 (size 800 samples) was approximately 94.83%, and for Chebyschev (size 200) was approximately 92.02%. A Support Vector Machine (SVM) was employed as the machine learning algorithm for these results. Following these experiments, a comparison was done in which the traditional statistical features used for RFF were employed instead of the images. They discovered that models trained with images had higher accuracy, and it is important to note that the disadvantage of image conversion is related to conversion time.

RF Fingerprint Reconition Based On Spectrum Waterfall Diagram

According to the article [30], they proceeded to translate the radio signals into two-dimensional images based on earlier work. As a result, there is an issue with radio signal detection for picture recognition. MATLAB simulations were utilised to produce the dataset used for image conversion. The signal dataset uses Quadrature phase-shift keying (QPSK) modulation, has a sample frequency of 20Mbps, a sampling interval of 0.05us, a signal-to-noise ratio of 10dB, and 52 subcarriers. There are four pilot signals and the rest are data signals, as 20 separate devices were constructed. These signals were then translated into 300x300 spectrum waterfall diagrams, which were put into the artificial intelligence programme that they created.

It should be noted that a dataset of 100,000 images was generated for the 20 devices, 80000 of which were used for training and 20,000 for validation. With this model they were able to obtain an accuracy of 89.01%. To compare their results with other approaches, they decided to use a different Convolutional Neural Network

(CNN) model in which they fed IQ data in raw format to the model, obtaining an accuracy of only 58.80% and increasing the training time by about three times.

2.2.4 Dataset Creation for Radio Frequency Fingerprinting

Creating a dataset for radio frequency fingerprinting is an essential step in developing and evaluating RF fingerprinting systems. A dataset is a collection of RF fingerprints, along with associated metadata, which are used to train and test RF fingerprinting algorithms. The quality and diversity of the dataset can have a significant impact on the performance of RF fingerprinting algorithms, and therefore, the design and creation of the dataset is a crucial aspect of RF fingerprinting research.

Data Collection

The first step in creating a dataset for RF fingerprinting is to collect data from a variety of devices and environments. The data should be collected using a consistent set of protocols and equipment, ensuring that the data is comparable and of high quality. Also, the data should be collected in a variety of environments and in the presence of different types of interference, to ensure that the dataset is representative of the range of conditions in which RF fingerprinting will be used. The distinctive or unique features of transmitted signals do not fluctuate over short time scales (days, weeks and months). However, it is vital to analyse the changes in the emitted signals over long periods of time (usually years) due to hardware-oriented flaws in the devices, which may modify the properties of the emitted signal [31].

Data Preprocessing

After collecting the data, it should be preprocessed to remove noise and unwanted signals while highlighting the relevant features. Filtering, demodulation, and feature extraction are examples of techniques that can be used. To ensure that the data is comparable and that the features extracted are relevant to the RF fingerprinting task, it is critical that the data preprocessing is consistent across the dataset [31].

Data Anotation

The data is then annotated with metadata such as device type, environment, and other pertinent information. This metadata is used to train and test RF fingerprinting algorithms, as well as to assess their performance. It is critical that the annotation is consistent across the dataset to ensure that the data is comparable and that the metadata is relevant to the RF fingerprinting task [31].

Dataset Splitting

After the data has been preprocessed and annotated, it should be divided into two sets: a training set and a test set. The training set is used to train RF fingerprinting algorithms, and the test set is used to assess the algorithms' performance. It is important to ensure that the data is randomly divided and that the data in the training and test sets is representative of the entire dataset.

Example Wireless Communication Datasets

There are few datasets available to train ML models applied to RFF in wireless communications. Moreover, those that exist are often private, which makes it difficult to obtain such datasets that can serve as guides.

An example of a dataset created for this type of applications was the dataset presented by A. Jagannath, Z. Kane and J. Jagannath [32]. It consists of a real-world radio RF dataset of commercial off-the-shelf Bluetooth and WiFi emitters. This dataset gathers emissions from two laptops and eight commercial chips.

Another interesting dataset on wireless communications is presented by G. Shen, J. Zhang and A. Marshall [33]. This dataset is made for a RFF system based on deep learning techniques and gathers RF signals collected from sixty commercial-off-the-shelf LoRa devices.

Chapter 3

Software and Technologies

This chapter discusses the primary technologies employed in the development project, including hardware and software. The concept of SDR, the most significant technology in the acquisition and processing of RFF signals, is the main topic of this chapter.

3.1 Software Defined Radio

Software Defined Radio SDR is a radio communication technology. In contrast to hardware-based solutions, this technology is based on software-defined wireless protocols. This translates to supporting various features and functionalities, such as updating and upgrading via reprogramming, without the need to replace the hardware on which they are implemented. This opens the door to the development of multi-band, multi-functional wireless devices. The high demand for SDR is driven by network interoperability, readiness to adapt to future updates and new protocols, and, most importantly, lower hardware and development costs [19].

SDRs have been studied for several years, and researchers are working to find better ways to implement them to optimise their processing and energy efficiency. SDRs are built on various hardware platforms, including General Purpose Processor (GPP), Graphics Processing Unit (GPU), Digital Signal Processor (DSP), and Field-programmable gate array (FPGA). Each of these platforms comes with its own set of challenges. Some of these challenges include maximising the computational power

of the chosen hardware platform, minimising power consumption, simplifying the design process, and minimising the cost of tools and equipment [19].

3.1.1 History and commercial products

The concept of SDR platforms arose with Joe Mitola in 1992. Later on, in 1996, the *Spectrum Ware software radio project* was carried out at MIT, and a prototype receiver of a Global System for Mobile Communications (GSM) base station was developed. In 1999 the US government started investing in this technology for military use by financing the production of SDR SpeakEasy I/II radios. Another big step in the SDR technology was the creation of the company Ettus in 2004. This company began the production of Universal Software Radio Peripheral (USRP) hardware [34]. The USRPs are very popular and good quality SDRs and its use has grown to become one of the most widely used platfoms nowadays in this area. An example of on of their most recent models is the USRP N320, released in 2020 and represented in Figure 3.1. Other companies that produce high quality SDRs are Crimson with models like Crimson TNG and Crimson Cyan or Deepwave Digital with the AIR-T SDR.

Figure 3.1: USRP-N320 by Ettus [35]

There are other SDR platforms that can be included in a low-cost category such as FUNCube, RTL, RSPduo, AirSpy, HackRF, BladeRF, LimeRF and ADALM-PLUTO. The last one is developed by Analog Devices and will be used in the current project. These devices, along with free software such as GNU Radio and driver integration for devices signal processing platforms like Matlab and Labview, have facilitated the global development of SDR technology [34]. For further equipment analysis, Table 3.1 compares commercially available SDR platforms and their main characteristics. The compared characteristics are ADC resolution and sample rate, transciever and receiver functionalities, frequency range, maximum bandwidth, and software integration. As expected, SDRs presented as low-cost have less functionality

and more limited specifications compared to high-performance SDRs. The better quality in higher cost SDRs is reflected in advantages regarding the captured and transmitted signals. For example, in the capture of WiFi signals, using a higher quality SDR like the USRP N320 allows capturing the full bandwidth of the signal (which is 22 MHz), since it can capture signals with bandwidths up to 100 MHz. The signal will also be captured at a good resolution, as its ADC resolution is 250 millisamples per second. In the case of a low-cost SDR such as ADALM-PLUTO, the signal obtained will not contain the entire WiFi signal, as its maximum bandwidth is 20 MHz. The resolution won't be as good either, since its ADC resolution is lower (61.4 millisamples per second). Despite these constraints, low-cost SDRs may be adequate to do the required tasks. In the case of WiFi, even though the ADALM-PLUTO cannot capture the complete bandwidth of the signal, signal characteristics can be identified in the portion that has been collected.

As an example, the Analog Devices ADALM-PLUTO Active Learning Module (PlutoSDR), depicted in Figure 3.2, is a tool that can be used to introduce fundamentals of SDR, RF, and communications as advanced topics in electrical engineering in a self-paced or instructor-led setting. A number of software packages, including MATLAB, Simulink, and GNU Radio, offer an innovative GUI, allowing for intuitive use and minimising the learning curve. It is based on the AD9363 and has one receive channel and one transmit channel that can be operated in full duplex, capable of generating or measuring RF analogue signals from 325 to 3800 MHz at up to 61.44 Mega Samples per Second (MSPS) and a bandwidth of 20 MHz. With the default firmware, the PlutoSDR is completely self-contained and powered entirely by USB. It supports OS X, Windows, and Linux and allows for the exploration and understanding of RF systems in a variety of environments [36]. In this thesis, we rely on PlutoSDR as the RF signal capture and processing device. As such, it plays a crucial role in the developed RFF device authentication architecture.

Figure 3.2: ADALM-Pluto by Analog Devices [37]

Platform	ADC/ DAC [Bits]	ADC/ DAC [MS/s]	Tx/Rx	Fmin-Fmax [MHz]	Max RF Bandwidth [MHz]	Matlab	GNU Radio
FUNcube Dongle	16/-	96 kHz	Rx	64–1700	80 kHz	no	yes
RTL	8/-	3.2/-	Rx	25–1750	3.2	yes	yes
RSPduo	14/-	10.66/-	Rx	0.1–2000	10	no	yes
Airspy-mini	12/-	20/-	Rx	24–1700	5	no	yes
Airspy-R2	12/-	20/-	Rx	24–1700	9	no	yes
HackRF One	8/10	20/20	Tx-Rx	1–6000	20	no	yes
ADALM-PLUTO	12/12	61.4/61.4	Tx-Rx	325–3800	20	yes	yes
BladeRF 2.0 Micro	12/12	61.4/61.4	Tx-Rx	47–6000	56	no	yes
LimeSDR	12/12	61.44/61.44	Tx-Rx	0.1–3800	61.44	no	yes
AD-FMCOMMS4-EBZ	16/16	61.4/61.4	Tx-Rx	70–6000	56	yes	yes
USRP-1	12/14	64/128	Tx-Rx	DC-6000	Daughterboards 40	yes	yes
PicoSDR	12/12	80/80	Tx-Rx	56–6000	56	no	yes
WARP-V3	12/12	100/170	Tx-Rx	2400/5000	40	yes	yes
USRP-N210	14/16	100/400	Tx-Rx	DC-6000	Daughterboards 40/80/120/160	yes	yes
TMDSSFFSDR	14/16	125/500	Tx-Rx	360–960	20	yes	yes
USRP X310	14/16	200/800	Tx-Rx	DC-6000	Daughterboards 40/80/120/160	yes	yes
USRP-2974	14/16	200/800	Tx-Rx	DC-6000	160	yes	yes
AIR-T	12/10	245.7/245.7	Tx-Rx	300–6000	100	no	yes
Sidekiq X4	16/14	245.7/245.7	Tx-Rx	1–6000	200	no	yes
USRP N320	14/16	250/250	Tx-Rx	3–6000	200	yes	yes
CRIMSON Cyan	16/16	325000/6000	Tx-Rx	0.5–18000	1000	no	yes

Table 3.1: Software Defined radio platforms comparison [34]

3.2 GNU Radio

GNU Radio is a free and open-source software development toolkit that contains signal processing blocks for the implementation of software-defined radios and signal processing systems. It can be used to construct a broad variety of radio systems, ranging from simple radios for receiving and transmitting signals in the High Frequency (HF), Very High Frequency (VHF), and Ultra High Frequency (UHF) bands to more complex systems for use in radar, satellite, and other applications [38].

The GNU Radio Companion (GRC) is a flowgraph editor that may be used to create and execute GNU Radio flowgraphs. GRC employs .grc files, which are then converted into Python .py flowgraphs [39]. It consists of a large number of pre-designed (in C++ for improved computation speed) signal processing modules that may be joined together to make a radio system, as well as an GUI for creating and testing the system. Figure 3.3 shows an example of a GNU Radio flowgraph in which the Funcube dongle Pro+ is used as the source for GNU Radio, capturing the RF signal, and then there are blocks connected to it such as filters, waterfall blocks in which the IQ signal can be visualised, and sinks through which the IQ data is stored in a file.

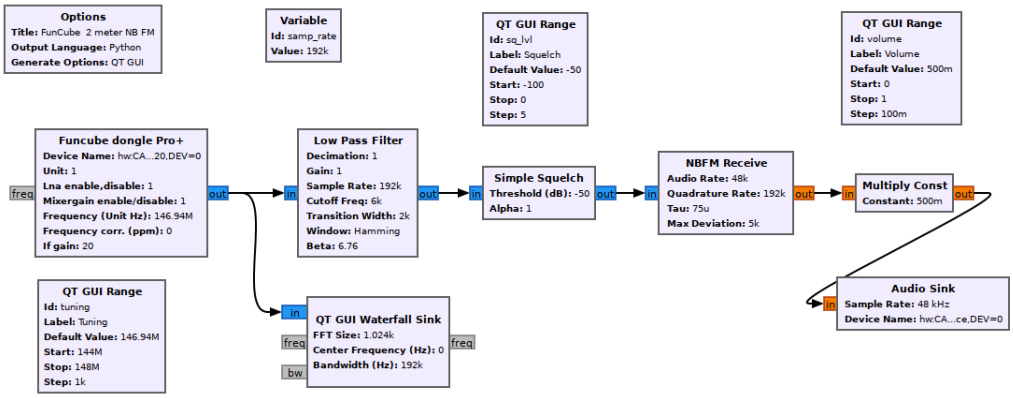


Figure 3.3: GNU Radio flowgraph example [40]

GNU Radio also allows users to create their own modules, known as Out of Tree (OOT) modules. An out-of-tree module is a GNU Radio component that does not live within the GNU Radio source tree. Typically, they serve cases where the user wants to extend GNU Radio with their own functions and blocks. This allows the user to maintain the code himself and have additional functionality alongside the main code. The OOT modules can be written in C++ or Python. The modules can be shared with the community at the Comprehensive GNU Radio Archive Network (CGRAN), where multiple OOT projects are hosted [41]. There are many types of blocks that can be created as a OOT module:

- Sync - The sync block is a block type used to create blocks that consume and produce an equal amount of objects per port. A sync block can have any number of inputs or outputs. A source is a sync block that has no inputs. A sink is a sync block that has no outputs.
- Decimator - Another sort of fixed rate block is the decimation block, in which the number of input items is a fixed multiple of the number of output items.
- Interpolator - Another sort of fixed rate block is the interpolation block, in which the number of output items is a fixed multiple of the number of input items.
- General - There is no correlation between the number of input items and the number of output items in the basic block. The basic block serves as the basis for all other blocks. When the other blocks are not appropriate, users should opt to inherit from the basic block [42].

3.3 IoMT Gateway

The Open Web Application Security Project (OWASP) defines the typical components of IoMT solutions as depicted in Figure 3.4 [43]:

1. Objectives: According to the FDA, connected medical devices (also known as IoMT endpoints) are medical devices that are linked to hospital networks, the Internet, or other medical devices. For completeness, the current work examines non medical devices that can be employed in IoMT situations, such as environmental sensors.
2. Gateways: These are networking devices that help connect weak endpoints to the backend by acting as a bridge network.
3. Back-end: Current IoT systems rely on back-end servers to run the IoMT solution, as well as process and store data.
4. Mobile devices/applications: Mobile devices/applications are often used in IoT systems to provide remote control of endpoints and back-end management as well as rapid notifications.

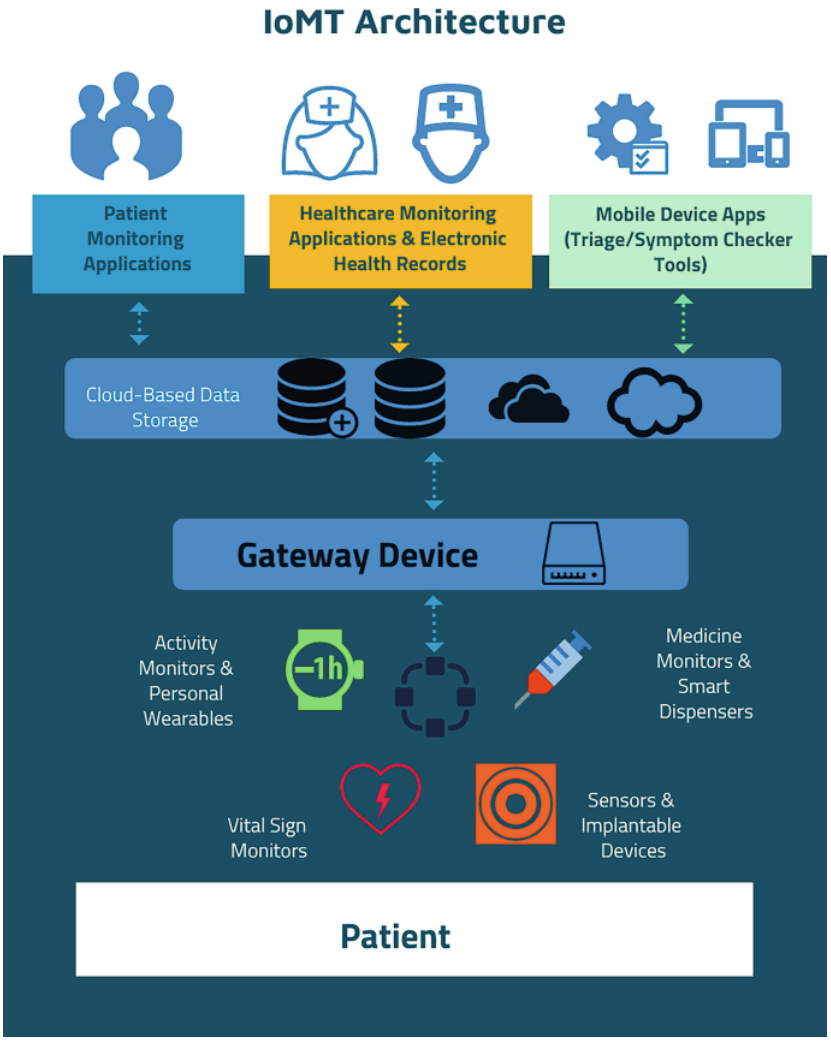


Figure 3.4: The IoMT landscape [44]

The components required in an IoMT solution vary based on the solution. As illustrated in Figure 3.4, IoMT solutions are categorised as follows [43].

- Wearable (e.g., heart monitors), implantable (e.g., embedded cardiac function monitors), ambient (e.g., door sensors), or fixed devices (e.g., computerised tomography scanners). Most current endpoints include networking capabilities, but some do not and, as a result, require a gateway.
- Platforms, which are typically cloud-based platforms designed to make smart devices and apps more accessible. IoMT solutions provide administrators with centralised back-end management features such as ecosystem administration, backup of reports and analytics, and online interfaces.
- Edge computing is used to evaluate acquired data (e.g., medical analytics) and interface with other systems via services (i.e., mobile or web applications)

(e.g., accounting and insurance). IoMT solutions are typically a combination of these sorts.

The development of a medical gateway is critical in the context of IoMT because it is the primary component in achieving the establishment of IoMT in the first place. To better understand the medical gateway, consider Figure 3.5. Medical devices are responsible for collecting medical measurements from users, and the edge iot gateway is responsible for connecting to these devices and storing the acquired data. The data is then structured in order to be delivered to the appropriate cloud services or API, where the sensitive data is debited for further study by the medical team. The medical team, in turn, visualises the data and offers information or alerts about the data it has processed; these alerts are transmitted from the cloud/API to the various gateways, allowing users or healthcare assistants to take the right action.

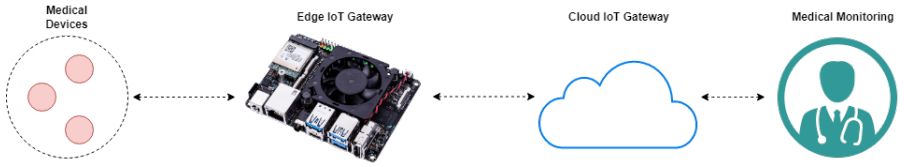


Figure 3.5: Medical Gateway

Medical devices can now communicate via Bluetooth. Software Development Kit (SDK) that enable the integration of the device into the appropriate gateway, allowing data to be received and then delivered to cloud services, are necessary to establish communication between a medical gateway and the medical devices. To accomplish this, the gateway must support an operating system capable of running these SDK. In the case of our built gateway, the ASUS ThinkerBoard Edge R, it supports the Android operating system, which is essential for integration because the medical devices we utilise include the Android SDK. When measurements are done, the SDK offered allow you to terminate the corresponding connections. Allowing these devices to connect to the medical gateway only when measurements from the user are required.

After receiving data from the appropriate devices, these are stored in the medical gateway for two reasons: first, to allow the user to view the most recent measurements, and second, in the event that it is not possible to send medical data to cloud/API services, to be saved for when the network conditions are reunified. It will also be possible to get alerts or messages from medical equipment originating from cloud or API services, informing the user of procedures or medication dosages.

All gateway data will be received here, in relation to the cloud service. Following receipt of these data, they are formatted for Fast Healthcare Interoperability Resources (FHIR) format for interoperability purposes, allowing them to be sent to

the appropriate medical equipment and API. FHIR is an Health Level Seven International (HL7) standard for the exchange, integration, sharing, and retrieval of electronic health information. These guidelines benefit clinical practises as well as the management, delivery, and evaluation of health care services [45, 46]. FHIR's compatibility with RESTful APIs, which allows it to take use of current web-based communication protocols, is one of its most appealing features. This combination allows FHIR to facilitate the exchange of healthcare data over common internet technologies such as HTTP and JSON. This implies that healthcare systems that support FHIR can connect more easily with other health and non-health systems [47].

If it is necessary to install data analysis modules, such as certain values that do not fall within the established norms for a specific user, notifications will be sent to the user or directly to medical teams in the event that an immediate response is required.

After receiving the data sent by the appropriate cloud services, medical teams can perform the necessary analyses by sending feedback, or even the names of the medications and dosages that patients must take, allowing these teams to work from a distance, reducing the burden placed on hospital staff.

The edge and the cloud are the two parts of a medical gateway. A project colleague developed the edge component, while I developed the cloud component.

The SDK integration was critical in the development of the Edge Gateway, enabling extensive interaction with the Android application. This integration was critical for collecting and processing data from a variety of medical devices, significantly improving the program's functionality. Bluetooth enabled the devices to connect to the application, ensuring reliable real-time data exchange. The program saved the medical equipment data for two reasons: giving users access to their most recent health indicators and storing data locally when there was no internet connection. OAuth2 procedures were used to secure data, ensure safe transmission, and manage individual data access permissions. To clearly present collected data, a user-friendly software interface and graphical user interface were developed.

Figure 3.6 shows the edge gateway, consisting of an ASUS Thinker Edge R board and a touch screen display.

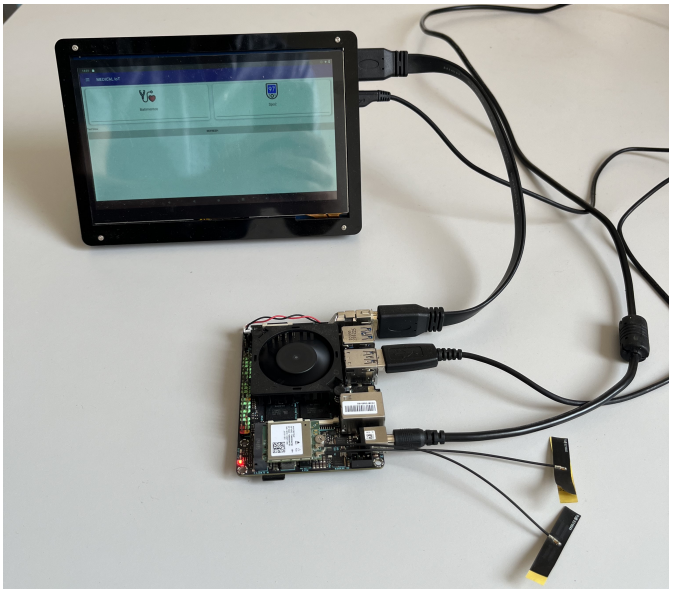


Figure 3.6: Edge Gateway

The cloud gateway component corresponds to the entire cloud environment of the developed medical gateway. The cloud gateway receives the health data collected by the edge gateway and handles all communication processes with external systems where the medical gateway can be integrated.

The cloud environment is based on Microsoft Azure services, where communication with the edge gateway takes place securely via the Azure IoT Hub platform.

The collected data can then be sent to external medical monitoring systems after establishing a secure HTTPS connection via POST method. FHIR format, suitable for health data communication, is used for sending data. The data in FHIR format is then sent as a JSON object to the external system, which should have a FHIR database.

Figure 3.7 represents a draft of the cloud gateway design from connecting to the Edge Gateway using the IoT Hub to sending the medical data in FHIR format to external medical monitoring systems.

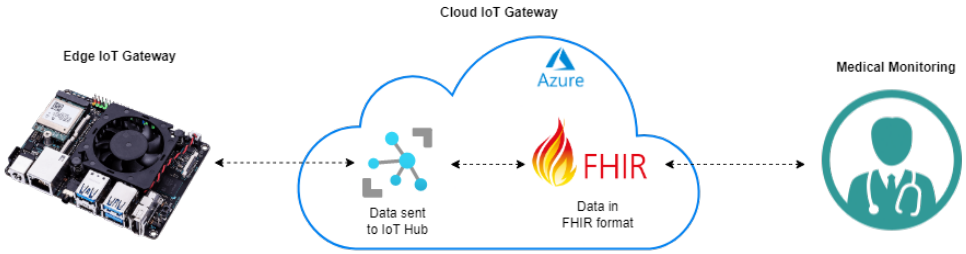


Figure 3.7: Cloud Gateway Draft

3.4 RFF ML Model Overview

Machine Learning is required for feature and classification purposes in RFF authentication. Machine learning can recognise patterns and other features inherent in data. In this project, a colleague developed CNN models that were used to achieve device classification. Through the use of Tensorflow the models were trained and evaluated with the created dataset. Finally, a new decision-making process is created, combining multiple features to classify and authenticate a device, achieving a new state of the art in the RFF spectrum. This development was then used to validate the created dataset as well as the signal acquisition and feature extration architecture.

3.5 IoT Devices

The devices discussed in this subsection were used as WiFi and 433 MHz signal transmitters. They play a critical role both in the creation of the signal dataset as well as in the validation of the developed architecture. The devices were chosen based on characteristics such as programmability, configurability, and communication capabilities.

3.5.1 ESP32

The ESP32, shown in Figure 3.8, is a powerful system on chip microcontroller developed by Espressif Systems that includes integrated Wi-Fi 802.11 b/g/n, dual mode Bluetooth 4.2, and a variety of peripherals, which makes it ideal for IoT applications.The chip has a dual-core CPU, clocked in different version up to 240 MHz. It features 36 GPIO pins, 16 PWM channels, and 4MB of flash memory [48]. This device was used in this project as a WiFi signal transmitter, signals to be captured by the ADALM-PLUTO SDR.

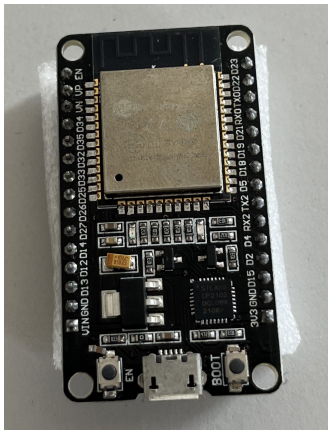


Figure 3.8: ESP32 board

3.5.2 ER400TRS

The easyRadio ER400TRS transceiver, shown in Figure 3.9, which features Easy-Radio technology, offers high-performance data transmission up to 250 metres and operates at a frequency of 433 MHz. It operates at low voltage and has low power consumption, and its serial input and output communication simplifies interfacing with host systems. Its embedded software significantly reduces design and development time [49]. This device was used in this project as a 433 MHz signal transmitter, signals to be captured by the ADALM-PLUTO SDR.

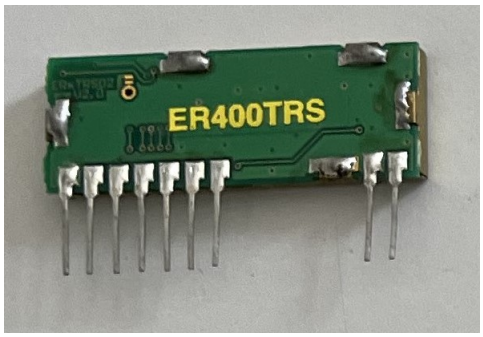


Figure 3.9: ER400TRS transciever

Chapter 4

RFF Architecture

4.1 IoMT Security RFF Gateway Architecture

The architecture we propose consists of a security gateway that employs RFF technology and works in conjunction with an IoMT gateway. The IoMT gateway is in charge of receiving data from medical devices such as oximeters and thermometers, as well as further analysing this data. The gateway can be divided into edge and cloud stages, with the cloud performing more demanding computation.

The security RFF gateway is designed to deal with IoMT system security threats making use of RFF and ML techniques. It must capture and recognise the signals that the devices send to the IoMT Gateway.

Consider the following scenario: a genuine user is operating an authorised medical device that is linked to the Edge Gateway. Simultaneously, an unauthorised entity attempts to exploit this active session by connecting a rogue device to the Edge Gateway in order to inject fake data into the system. The goal is to bypass authentication and introduce unauthorised data directly. The security gateway must include security mechanisms to counter such threats, notifying the IoMT gateway that the communication was not validated. This process should take place in real time to detect security threats as soon as possible. The system should also have an active mode that can demand the device to communicate based on specific parameters for better identity detection. At the same time packet monitoring is carried out, where the MAC address and IP of the devices communicating on the network are monitored as additional security.

The proposed RFF Security Gateway block diagram is depicted in Figure 4.1. It begins with signal capture using an SDR (ADALM-PLUTO), followed by feature extraction using SDR software (GNU Radio in this case). The first loopback, signal acquisition controller, refers to the previously described active mode concept. Following the feature extraction stage, the system proceeds to the generation of fingerprints, where the device identity should be present. This fingerprint is classified based on the device emitting the signal using ML. Finally, a second loopback is in charge of controlling the fingerprint generator for better classification results. This thesis mainly addresses the work done on the components coloured in the figure (SDR and feature extractor). The loopbacks in the diagram have not been developed and are considered future work.

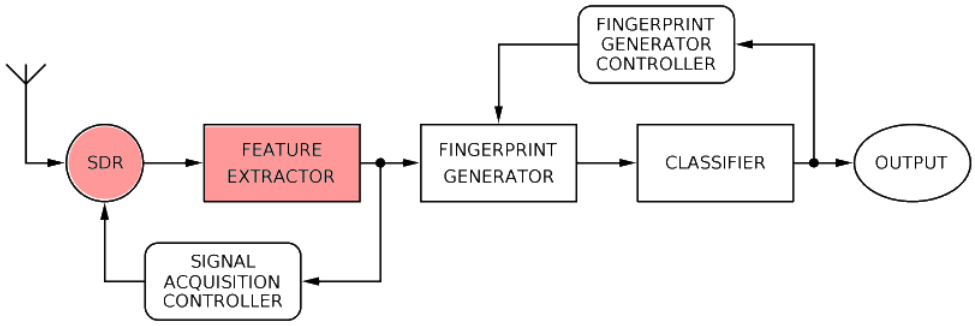


Figure 4.1: Block diagram of the proposed architecture

4.1.1 RFF Integration in Medical Gateway

Figure 4.2 depicts the process of incorporating RFF into the medical gateway. In our example, the SDR, represented by the ADALM-PLUTO, takes the lead in recording signals sent by the device attempting to authenticate. These signals are then promptly received and processed.

The computer manages the features of these signals, which are subsequently categorised using CNN models. The results of the CNN models are compared to the device specified by the medical gateway. This process provides secure exchanges and the implementation of additional security procedures, allowing for reliable authentication.

We created a unique communication protocol to strengthen this security even more, both the PC and medical gateway are connected with a USB connection Section 4.2 will go over the specifics of this protocol.

As previously described, the RFF integration procedure into the medical gateway is extremely relevant to a wide range of systems that integrate similar architectural aspects, particularly those that use a gateway for communication. The use of an SDR for signal capture, along with computer-based signal management, guarantees a dependable and dynamic solution.

Furthermore, the development of a different communication protocol strengthens the system. This protocol not only provides for easy network interactions, but it also supports secure exchanges, which is critical for system integrity.

As a result, this integrated model provides a customizable template that can be changed to strengthen security and increase efficiency in a variety of systems centred on a gateway architecture. It emphasises the potential for scalability and adaptability in similar or related sectors, hence accelerating the further implementation of such technology integrations.

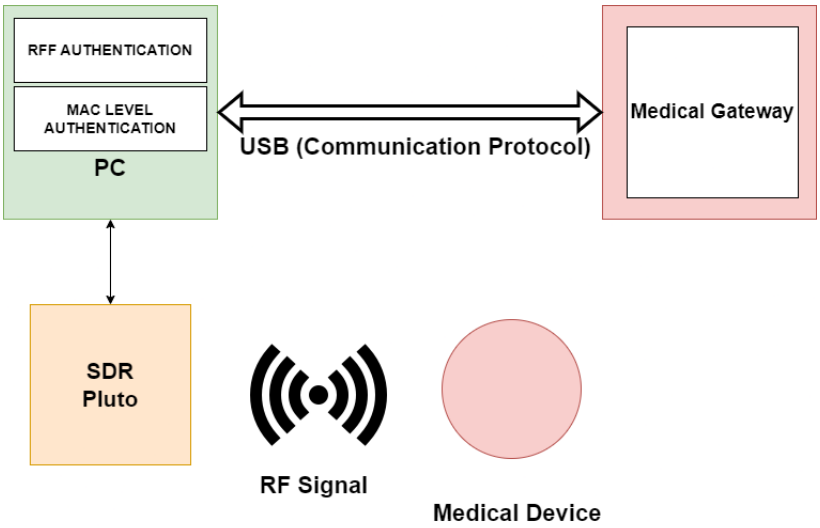


Figure 4.2: RFF Integration in Medical Gateway

4.2 Communication Protocol

Because of the sensitive nature of the data it maintains, the USB-connected edge IoMT Gateway requires a reliable, secure connection protocol. In this case, the PC authenticates the edge gateway medical device using RFF. The messages of this protocol are sent by Android Debug Bridge (ADB), a versatile command-line tool that allows communication between an Android device and other devices, allowing for a variety of device actions ranging from simple message communication to app installation and debugging [50]. The messages sent by ADB are simply strings containing the content to be transmitted. For example, when the gateway has a device to authenticate, the identification number of this device is sent.

The communication protocol must be designed so that the Edge Gateway can designate which device it wants to authenticate during medical measures. This implies that the protocol should provide different messages or commands that enable this operation. Following the conclusion of the measurement, the PC must communicate the results of the authentication procedure to the edge gateway. As a result, the communication protocol must be set up to allow for authentication result confirmation. Fundamentally, the communication protocol should enable three types of messages to be exchanged between the PC and the edge gateway:

1. A request for authentication of a specific device from the edge gateway to the PC.

2. A notification from the PC to the edge gateway indicating that the authentication process is in progress.

3. A last message from the PC informing the edge gateway about the outcome of the authentication process.

Furthermore, acknowledgement responses must be supported by the protocol. These confirmation responses are critical for maintaining communication integrity and reliability.

A reliable communication protocol must also be capable of identifying and handling possible threats. For example, the protocol must be able to detect spoofing efforts, in which an attacker attempts to pose as a legitimate device. In such cases, the protocol should include provisions for rejecting authentication and alerting the system.

A well-designed, secure communication protocol will ensure that all information transfers between the PC and the Edge Gateway are reliable and secure, reducing the possibility of errors or intercepts. Such a strategy will safeguard the privacy and integrity of medical measurements, improving the system's overall reliability and efficacy.

4.2.1 Packet monitoring

Packet monitoring, often known as "packet sniffing", is critical for detecting and blocking unauthorised network activity. The importance of packet sniffing in the system in this context rests mostly in detecting and warning about attempts by unauthorised or "rogue" devices to enter the network. This is especially important when these devices attempt to replicate MAC addresses, which is a typical practise in network spoofing attacks. A packet monitoring script was developed that runs in the background on the PC during the whole communication process between the devices and the gateway. The script (shown in Listing 4.1) was developed taking the following points into account:

1. Unauthorised Device Detection: Packet sniffing allows the system to identify data packets from unknown or suspicious MAC addresses. As a result, if an unauthorised device attempts to connect to the network, the system will be able to detect it quickly.

2. Duplicate MAC Address Alerting: When devices attempt to duplicate MAC addresses, packet sniffer can detect the existence of two or more devices on the network with the same MAC address. When this happens, the system can be notified so that necessary action can be taken.

3. Spoofing Attack Prevention: Detecting devices with duplicate MAC addresses may signal a spoofing attempt. MAC spoofing is a popular method for disguising an unauthorised device as a valid one. Packet sniffing can assist avoid these attacks by detecting MAC address duplication.

4. Maintaining Network Integrity: Packet sniffing contributes to network integrity and security by enabling for the rapid detection of suspicious or unauthorised activities.

5. Real-Time Traffic Analysis: Packet sniffing allows for real-time traffic analysis, which can be critical for detecting and mitigating developing network security threats.

```
if not packet.haslayer(Ether):
        return

    src_mac = packet[Ether].src
    if src_mac not in known_mac_addresses:
        print(f"Alert: Unknown MAC address detected: {src_mac}")

    if src_mac not in device_counts:
        device_counts[src_mac] = {'count': 0, 'ip': None}
    device_counts[src_mac]['count'] += 1

    if packet.haslayer(IP):
        src_ip = packet[IP].src
        device_counts[src_mac]['ip'] = src_ip
        if src_mac in device_ips:
            if src_ip not in device_ips[src_mac]:
                device_ips[src_mac].append(src_ip)
                print(f"Alert: Duplicate IP detected for MAC: {
    src_mac}")
        else:
            device_ips[src_mac] = [src_ip]
```

Listing 4.1: Packet sniffing

4.2.2 Communication scenarios

In a successful authentication scenario, depicted in Figure 4.3, we have a key interaction between a medical device and the medical gateway for the transmission of health data collected from the patient. From the moment the patient initiates the measurement process at the gateway, the medical device will constantly send health data collected from the patient. An oximeter, for example, will send data such as SpO2 and pulse rate. When this communication begins, the RFF platform will also begin the device authentication process in parallel. The platform communicates the device's successful authentication to the medical gateway, which then sends the medical data it has collected during that time to the cloud service, as it has passed through all the system security protocol.

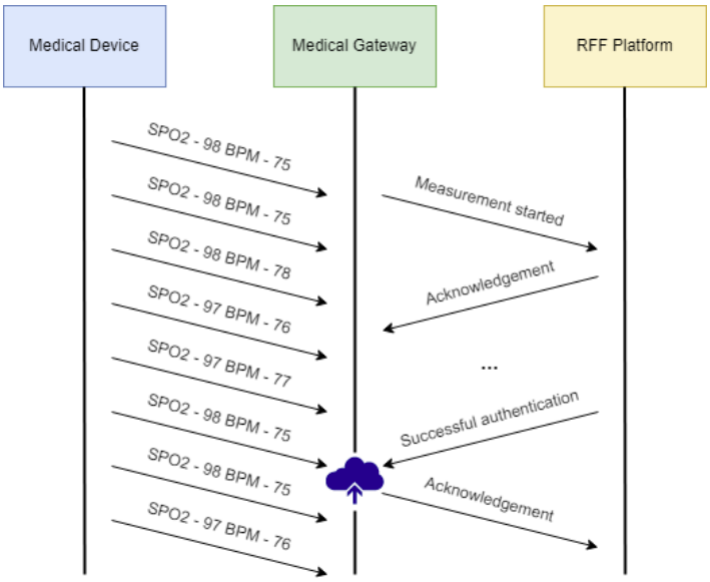


Figure 4.3: Successful authentication

In a failed authentication scenario, depicted in Figure 4.4, we have a potential attack on the system by a malicious device. This device passed the previous packet sniffing test, which involved spoofing its IP address and MAC adress, being only detected as malicious by RFF methods. The malignant device will constantly send manipulated data from the moment the patient starts the measurement process at the gateway. Once this communication starts, the RFF platform will also initiate the device authentication process. As soon as the device fails the authentication process, the platform communicates it to the medical gateway, which will not consider this data, not even sending it to the cloud service, since it has not passed through the whole system security protocol.

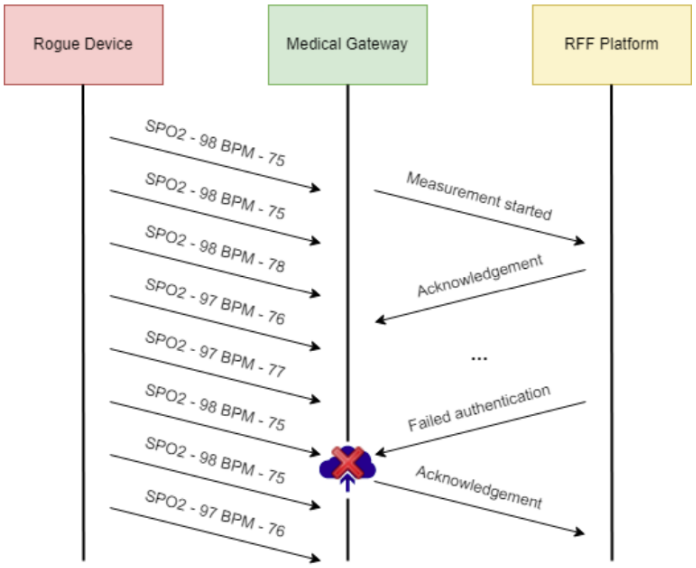


Figure 4.4: Failed authentication

4.2.3 Last Safety Measure - 433MHz Transceivers

Medical devices are made to communicate with the gateway over WiFi, and as such, the authentication system must identify the devices by the WiFi signals they transmit. If this does not happen, the system includes an RFF method for signals from the ER400TRS devices, that communicate in the 433MHz frequency range. In this method, the RFF system will read signals in the described range, and the medical device to be authenticated must emit a signal in this frequency in order to be recognised. This emission is performed using ER400TRS transceivers that must be connected to the device to be authenticated.

The 433MHz security method is activated when the WiFi signal from the medical device is not authenticated in three signal readings by the RFF platform. In this method the RFF platform is considerably more effective, since ER400TRS transmitters have significant signal differences between them in features such as frequency offset or clock synchronisation.

If the system cannot authenticate the device even using this method, the system is assumed to be under attack, and measurements at the medical gateway are blocked.

Chapter 5

Signal Acquisition and Feature Extraction Architecture

This chapter refers to the RFF architecture's signal acquisition and feature extraction processes. This section is critical to the production of the dataset as well as the RFF operation. This system is nearly entirely built over GNU Radio and with ADALM Pluto as an SDR recording the signal. The signal acquisition system is defined in a GRC flowgraph, which includes a graphical user interface where the acquired signal can be seen in a waterfall graph. A custom GNU Radio OOT Block was constructed in order to further improve the RFF system. Its concept and creation are also explained in this chapter.

5.1 GNU Radio Flowgraph

The GNU Radio flowgraph created for this aspect of the architecture is divided into three major components: the source, signal processing and sinking, and signal plotting. The full flowgraph is depicted in figure 5.1 and each of its components are going to be described in subsections 5.1.1, 5.1.2 and 5.1.3.

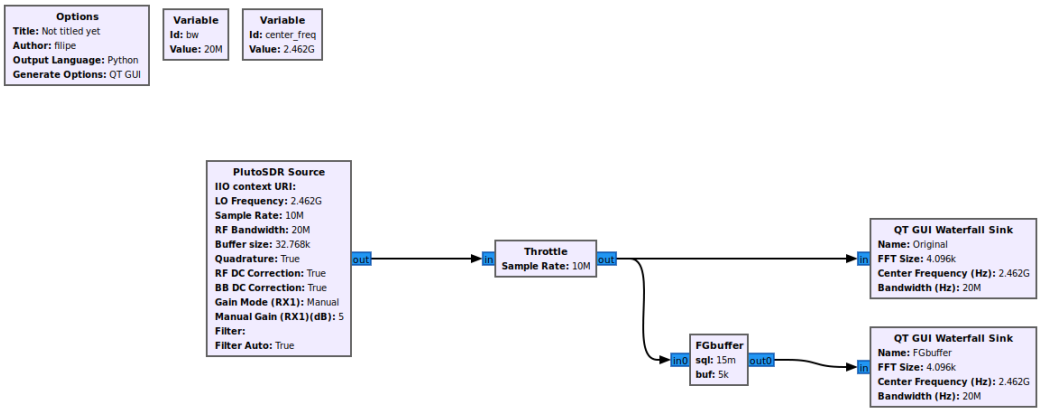


Figure 5.1: GNU Radio flowgraph

5.1.1 Signal Source

The source part is defined by the Pluto SDR Source GNU Radio source block. This block was designed specifically for the ADALM Pluto SDR when used as a source device. It has several configurable parameters, as shown in figure 5.2. In this situation, the most important ones are the LO Frequency, Sample Rate, Bandwidth, Buffer Size, Gain Mode and Filter and they can be described as follows [51]:

- LO Frequency - This parameter selects the receiver local oscillator frequency. For example, in order to capture WiFi signal bursts in channel 11, we must select the centre of the channel frequency, which is 2.462 GHz;
- Sample Rate - The sample rate setting specifies how many samples the Pluto SDR can receive per second. This determines how much bandwidth the SDR receives at once;
- Bandwidth - This setting configures the receiver analog filter in the Pluto SDR;
- Buffer Size - This parameter defines the size of the internal buffer in samples. The block will only output one buffer of samples at a time;
- Gain Mode - This parameter allows the selection of one of the available gain modes (manual, slow attack, hybrid and fast attack). For spectrum sensing type applications such as this, manual gain is the best option, since it enables for accurate detection of signal presence, and it's relative power;
- Filter - This argument allows you to load a Finite Impulse Response (FIR) filter configuration from a file. This is not used because the raw signal is required for this application.

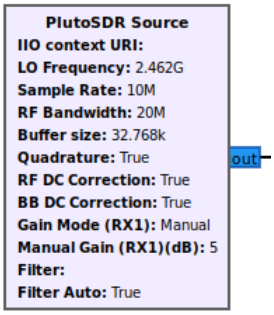


Figure 5.2: PlutoSDR Source block

5.1.2 Signal Processing and Sinking

The signal processing and sinking component is the part of the flowgraph that is in charge of receiving data from Pluto SDR and then processing and storing it. This component must perform these tasks continuously as the signal is captured. The signal processing must remove the unnecessary parts of the captured content while preserving the useful content. Since there is no premade block that performs these functions, a custom OOT block was created from scratch. Figure 5.3 depicts this block, which was named FGbuffer. Its creation from scratch and operation will be detailed in sections 5.2.

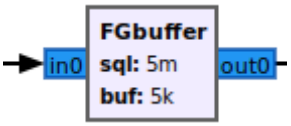


Figure 5.3: Signal processing and sinking block (*FGBuffer*)

5.1.3 Signal plotting

In order to allow visualization of the original and recorded signal by the block in a graphical way, the QT GUI Waterfall Sink block available in the original GNU Radio blocks was used. These blocks (depicted in figure 5.4) have 3 adjustable parameters [52, 53]:

- Fast Fourier Transform (FFT) Size - Defines the size of the FFT to compute and display;
- Center Frequency - Specifies center frequency of the signal to be displayed;
- Bandwidth - Configures the bandwidth of signal to be displayed.

It should be noted that these parameters only affect the plot that is displayed to the user; they have no effect on the signal. They must be configured in such a way that the signal is presented as clearly as feasible. To display a WiFi signal

transmitted on channel 11, for example, the centre frequency must be set to 2.462 GHz and the bandwidth to 20MHz (WiFi signals have a bandwidth of 22GHz, but Pluto can only capture 20MHz at a time [54]).

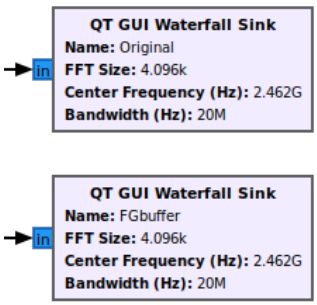


Figure 5.4: QT GUI Waterfall Sink blocks

In this flowgraph, QT GUI Waterfall Sink blocks are used to generate Waterfall plots that allows the user to check the signal acquisition by the Pluto SDR and the recording of the signal by the created OOT block. The user interface is shown in figure 5.5, where the upper graph shows the original signal obtained and the lower graph shows the processed signal as it is recorded in a file.

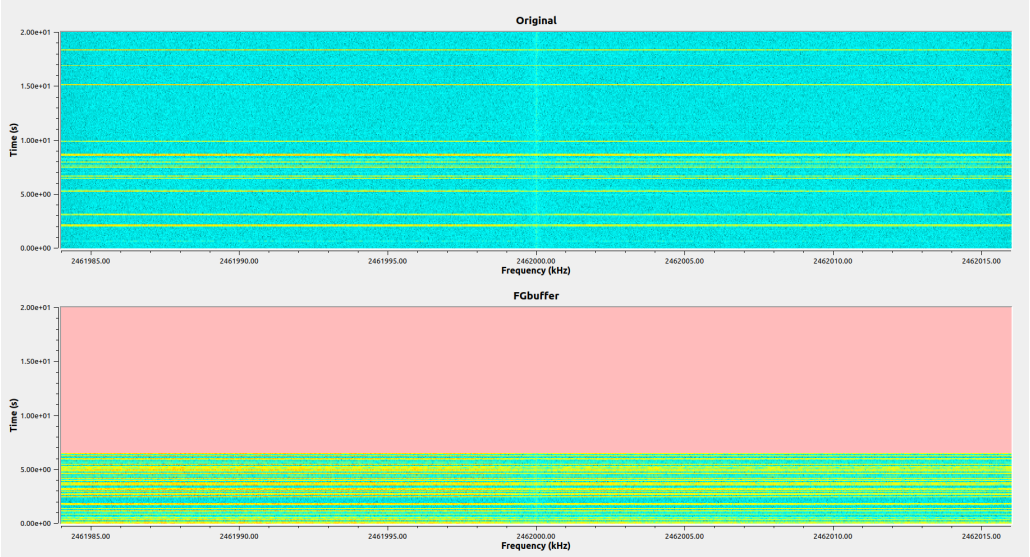


Figure 5.5: GNU Radio Waterfall Plots

5.2 Custom Block Concept and Implementation

The block that has been created is intended to meet the signal processing and sinking requirements described in subsection 5.1.2. The main goal is to have a block capable of identifying the RF signal bursts received by the Pluto SDR and store them in a file for later analysis. This collection must occur on the fly, and in the most optimized

way possible, i.e., collecting the necessary signal while cutting the intervals in which no transmissions occur. This way we have a file with all the necessary content to later feed the machine learning model while using less storage space, which is critical since .iq raw files take up a lot of storage.

Initially, the Power Squelch block was used, which allowed discarding signal samples that fell below a user-defined threshold in dBs in the GRC while saving samples that fell above this threshold. The signal was then routed to a sink block, which saved the processed signal. This method can be used, however, it has a significant limitation: it will cut the transients of RF signals if they are below the threshold. This could significantly reduce the efficiency of the machine learning model, since the generated files will have bursts with the transient cut off, and the transient contains important and distinguishing features regarding device identification.

With this in mind, we decided to create an OOT block that fulfilled these requirements, since no block offered by GNU Radio did so. The idea was then to create a block that would receive the data stream coming from Pluto SDR, store this data in a circular buffer and process the samples that actually belong to RF signal bursts, discarding the empty samples. It is also important to preserve the transients of the collected bursts in order to retain as many features of each signal burst as possible. This signal transient preservation is achieved by making this threshold more "flexible". This is possible if, instead of comparing values to the threshold, we compare the average value of the values in the circular buffer. The cuts are less abrupt this way, and the transient in most signal bursts is preserved.

5.2.1 Using *gr-modtool*

GNU Radio includes *gr-modtool*, a software tool for creating OOT modules. An OOT module is a collection of customised GNU Radio blocks. In order to create an OOT module called *FGSink* using *gr-modtool*, the following command was used:

```
$ gr_modtool newmod FGSink
```

The directory *FGSink* is created which contains all of the skeleton code for an OOT module, however it does not yet have any blocks. We must now use the following command to move to the newly created directory:

```
$ cd FGSink
```

The desired custom block can now be developed within the newly created directory. To create a block named *FGBuffer* inside the module, the following command was used:

```
$ gr_modtool add FGBuffer
```

The command will start a questionnaire to determine how the block should be defined, including the block type, language, and parameters.

The general block is the most appropriate type of block to select in this case because there is no correlation between the number of input items and the number of output items in the developed block:

```
GNU Radio module name identified: FGSink
('sink', 'source', 'sync', 'decimator', 'interpolator',
'general', 'tagged_stream', 'hier', 'noblock')

Enter block type: general
```

Following that, the programming language used to write the block's code must be chosen. Python was used in this case:

```
Language (python/cpp): python
Language: Python
Block/code identifier: FGBuffer
```

The 'Block/code identifier' parameter simply identifies the name of the block and its code file.

The questionnaire then asks for the copyright holder's name or organisation, as well as the argument list for the Python function. These parameters are unimportant and can be ignored because they can be changed later directly in the code. It also provides Python quality assurance code, which was not used.

Following this configuration, the block code files will be generated: *FGBuffer.py*, which describes the block's operation, and *FGSink_FGBuffer.block.yml*, which defines the block's interface for GRC. The *CMakeLists.txt* file has been updated so that both files are installed when the module is compiled and installed. The file generation is depicted in the console:

```
Adding file 'python/customModule/FGBuffer.py'...
Adding file 'grc/FGSink_FGBuffer.block.yml'...
Editing grc/CMakeLists.txt...
```

We then proceeded to edit these code files in order to shape the block for the desired purpose. This process is explored next.

5.2.2 Custom Block Code

The code defining the functionality of the block is present in the files *FGBuffer.py* and *FGSink_FGBuffer.block.yml*.

FGBuffer.py code

The code of a GNU Radio block in python has two essential functions: the *___init___()* function and the *work()* function.

The *___init___()* function essentially defines the essential parameters that the block will have in GRC, as well as the inputs and outputs. The code for the *___init___()* function can be found in listing 5.1. The parameters in this case will be *sql_lvl* (parameter that defines the squelch level applied to the input signal) and *buf_len* (parameter that defines the size of the circular buffer used by the block to save the input items). The inputs and outputs, *in_sig* and *out_sig* respectively, are defined as 64-bit complex type variables.

```
1  def __init__(self, sql_lvl=0.015, buf_len=10000):
2      gr.sync_block.__init__(self,
3                             name="FGbuffer",
4                             in_sig=[np.complex64],
5                             out_sig=[np.complex64])
6
7      self.sql_lvl = sql_lvl
8      self.buf_len = buf_len
9      self.buffer = deque(np.zeros(self.buf_len, dtype=np.complex64)
       , maxlen=self.buf_len)
10     self.sig_state = 0
11     self.file_saved = 1
12     self.out0 = []
13     self.mean_buffer = 0
```

Listing 5.1: ___init___() function

The core processing function in any GNU Radio OOT Block is the *work()* function. It receives input samples, processes them, and produces output samples. In the developed block, *FGBuffer*, it is no exception. The *work()* function of this block is displayed in listing 5.2. During processing, the function maintains a buffer of samples and calculates the mean value of the buffer. If the mean value exceeds a predefined threshold (*sql_lvl*), the presence of a signal is detected and the samples are collected into an output buffer. Once the mean value falls below the threshold, the collected signal is saved to a file and the state is reset. The ongoing signal samples are also outputted for signal plotting in a QT GUI Waterfall Sink, as explained previously in subsection 5.1.3. This function provides the necessary functionality for

signal detection and storage, allowing the *FGbuffer* block in GNU Radio to analyse and process complex data streams.

```
def work(self, input_items, output_items):
    in0 = input_items[0]
    out0 = output_items[0]
    chunk_size = len(in0)
    out_index = 0
    for i in range(chunk_size):
        sample = in0[i]
        # update mean buffer
        old_val = np.abs(self.buffer[0])
        new_val = np.abs(sample)
        self.mean_buffer = self.mean_buffer - old_val / self.
    buf_len + new_val / self.buf_len
        # push data in buffer
        self.buffer.append(sample)
        if self.mean_buffer >= self.sql_lvl:
            if self.sig_state == 0:
                self.out0 = list(self.buffer)
                self.sig_state = 1
                self.file_saved = 0
            elif self.sig_state == 1:
                self.out0.append(sample)
        else:
            if self.file_saved == 0:
                # save output in iq file
                filename = "/home/filipe/Documents/all_signals.iq"
                with open(filename, 'ab') as f:
                    np.array(self.out0, dtype=np.complex64).tofile
    (f)
                self.file_saved = 1
            self.sig_state = 0
        # signal output as it is being saved (for waterfall sink)
        if self.sig_state == 1:
            out0[out_index] = sample
            out_index += 1
    self.consume_each(chunk_size)
    return out_index
```

Listing 5.2: work() function

FGSink_FGBuffer.block.yml code

As a whole, the YAML code displayed in listing 5.3 defines the *FGbuffer* block with the two created parameters (*sql_lvl* and *buf_len*), one input port (*in0*), and one output port (*out0*), allowing the block to process complex data streams.

```
id: FGSink_FGbuffer
label: FGbuffer
category: '[FGSink]'

templates:
  imports: import FGSink
  make: FGSink.FGbuffer()

parameters:
- id: sql_lvl
  label: sql
  dtype: float
  default: 0.015
- id: buf_len
  label: buf
  dtype: float
  default: 100000

inputs:
- label: in0
  domain: stream
  dtype: complex

outputs:
- label: out0
  domain: stream
  dtype: complex

file_format: 1
```

Listing 5.3: FGSink_FGBuffer.block.yml code

5.3 Dataset Creation for Machine Learning

Many of the most recent advances in machine learning are being driven by researchers in computer vision, voice recognition, natural language processing, medical imagery, and finance. Unfortunately, radio signal processing has been noticeably absent from much of this recent work, but the potential for applying recent advances in machine learning to the radio domain right now is enormous. Researchers in communications are increasingly considering these methods, but they lack common benchmarks and open datasets for evaluating advances, as seen in other application areas [55].

Given the scarcity of datasets for signal applications, a dataset for the communication signals under consideration (WiFi and 433 MHz) was created. This subsection describes the creation of this dataset as well as the extraction of features from it

in order to prepare the correct feeding of the ML algorithm developed by another colleague in the project.

5.3.1 Collection methodology

This subsection discusses the several aspects of the dataset creation process, such as the collection site, hardware and software setup, and the collection procedure itself. This procedure was carried out taking into account the conditions available for its realization, which are not optimal in terms of signal collection without noise due to the collection site and hardware limitations.

Collection Site

The signal sampling was carried out in an office room where external interference exists, in order to simulate a real environment where these external signal influences also exist. This external noise is most noticeable in the 2.4GHz frequency range, where WiFi and Bluetooth signals operate. This factor causes the obtained dataset to have more noise, which hinders the learning process performed later by the ML model, but also makes it more efficient in real environments where these noises are common.

Hardware Setup

In terms of hardware, a Pluto SDR is used as a signal receiver. The SDR is coupled with a common external 2.4 GHz antenna, since its original antenna is designed to operate in the 824-894 MHz and 1710-2170 MHz bands, which does not include the 2.4 GHz band of the WiFi signal. The Pluto SDR is connected to a computer, to which the signal is sent via Universal Serial Bus (USB) input. The collection setup is depicted in figure 5.6.

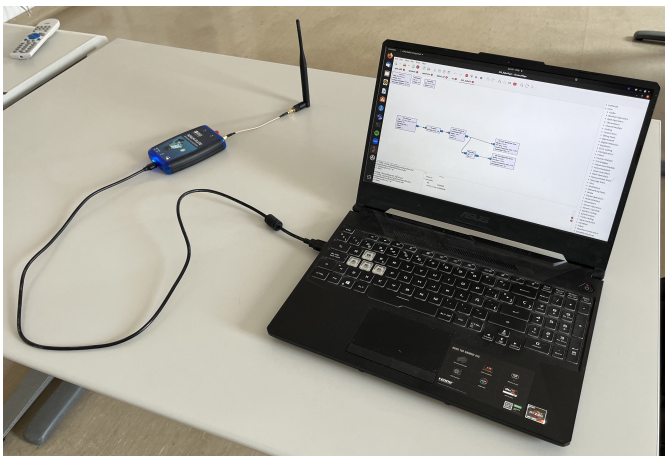


Figure 5.6: Collection setup

As signal transmitters, three ESP32 2.4GHz Dual-Mode WiFi and Bluetooth Development Boards and one Microsoft Surface Book 2 computer were used for WiFi transmissions. For 433 MHz transmitions, four ER400TRS transceivers were used. These devices were chosen to see if it was possible to tell the difference between completely different devices and devices of the same model.

Software Setup

At the software level, GNU Radio was used to receive data sent by the Pluto SDR and perform the necessary digital processing, as well as file collection for the dataset. Two different GNU Radio flowgraphs were used in signal capture, depending on the type of signal to be captured (433MHz or WiFi). Figure 5.7 depicts the flowgraph used to capture WiFi signals, while Figure 5.8 depicts the flowgraph used to capture 433 MHz signals.

To capture WiFi signals, the Pluto SDR is configured with the central frequency of channel 11 of 2.462 GHz and with 20 MHz of bandwidth in order to take advantage of the maximum bandwidth of Pluto SDR. This method will not allow the capture the entire bandwidth of the channel, which is 22 MHz, but it will allow the extraction of important features of the WiFi signal.

As for capturing signals in the 433 MHz range, the Pluto SDR is configured with that same central frequency, and with 2 MHz of bandwidth, since this is enough to get all the signal transmitted by the devices. In terms of the custom block, the *sql_lvl* parameter was adjusted for each application based on performance tests.

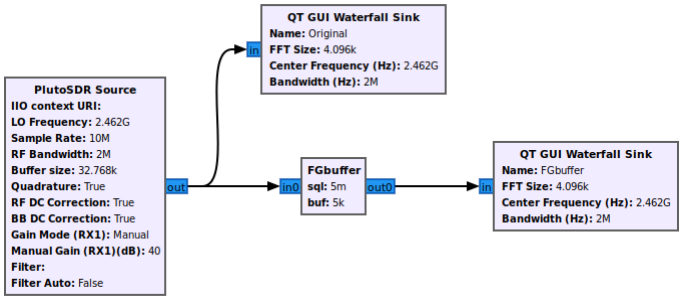


Figure 5.7: Flowgraph for WiFi signal capture

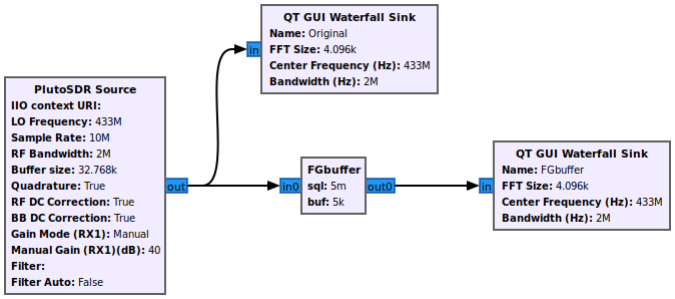


Figure 5.8: Flowgraph for 433 MHz signal capture

Data Collection Procedure for Wifi Signals

As for WIFi signals data collection, a private Wireless Local Area Network (WLAN) was created where the devices send ICMP pings to an access point, while the signal is captured by the SDR. The access point is a Buffalo WZR2-G300N that creates a network on WiFi channel 11, to which the communicating devices connect. The devices were configured to send pings repeatedly with defined time intervals.

The devices communicated with the access point one at a time, so there were no concurrent communications. This way, we know exactly which device generated the signal to be written to the dataset.

For each device, signals sent at different distances between the device and the SDR were recorded. This allows the system to detect the communicating device regardless of distance or signal amplitude. The recording distances between the device and the access point were 2, 2.5, and 3 metres. As expected, the captured signal had less amplitude the further the device was from the access point.

At the end of the recordings, 500 samples of 10MB per device were obtained, leading to a total size of 20 GB in the WiFi dataset.

Data Collection Procedure for 433 MHz Signals

Regarding data collection of signals transmitted in the 433 MHz frequency range, ESP32s connected to ER400TRS transcievers per serial port were used. These transcievers are used to emit signals in the frequency range described, and this transmission is also amenable to RFF techniques for device identification. These emissions are then picked up by the Pluto SDR, which is centred on this frequency range when performing the capture.

The 433 MHz signal capture approach was similar to that of WiFi signal capture, meaning that the SDR pluto captured the signal emitted by the devices at various distances (the same emitting distances of 2, 2.5, and 3 metres were performed). The collected dataset has about the same dimensions as the WiFi dataset, with 500 samples per device.

Dataset anotation and splitting

The dataset was saved in IQ files, a format suitable for representing RFF signals and serving as a basis for the generation of images that highlight the features of each signal. As for the annotation criteria, the files were recorded in order to identify the type of emitted signal (WiFi or 433MHz), the emitting device, its distance from the Pluto SDR, and the recording time. The files are then named in the format *EmittedSignal_Device_Distance_Time.iq*, in order to identify all the recording parameters. For example, the file containing the recording of ESP32 number 1 transmitting WiFi signals at a distance of 2 meters from the Pluto SDR for 8min will be called *WiFi_esp1_2m_8min.iq*.

After the data has been collected, processed and annotated, the files should be divided into a training set and a test set, as mentioned in subsection 2.2.4. This division was done in a random manner, so that the two sets are representative of the original dataset.

5.3.2 Signal Features

We then proceed to extract features from the dataset files that we have created. Different electromagnetic waves will be produced by each device, resulting in different signals. These signals contain distinguishing characteristics that can be extracted and used to identify the emitting device. These features are highlighted by being represented by different kinds of images in order to feed the machine learning model. This section will explain these images and their creation.

Constellation

The constellation images can help with the analysis of tiny differences in captured signals that can be used as RFF features. By examining the real (I) and imaginary (Q) components of the complex-valued data points, the constellation plots can reveal distinctive patterns or abnormalities in the signal that can be attributed to a specific transmitter. These different patterns can be used as input features for machine learning algorithms to identify and classify devices based on their RFF.

The code in Listing 5.4 is used to read complex-valued IQ data files and create a constellation plot to display the data. A constellation plot is a scatter plot of the real (I) and imaginary (Q) components of complex-valued data points.

```
1 data = np.fromfile(filename, np.complex64)
2 plt.rcParams["figure.figsize"] = (100,100)
3 plt.plot(np.real(data), np.imag(data), '.')
4 plt.grid(True, which='both'))
```

Listing 5.4: Constellation Plot

Figure 5.1 depicts an example of a constellation image generated by the code, which plots the real and imaginary components of the IQ file. This image is suitable for feeding the ML model.

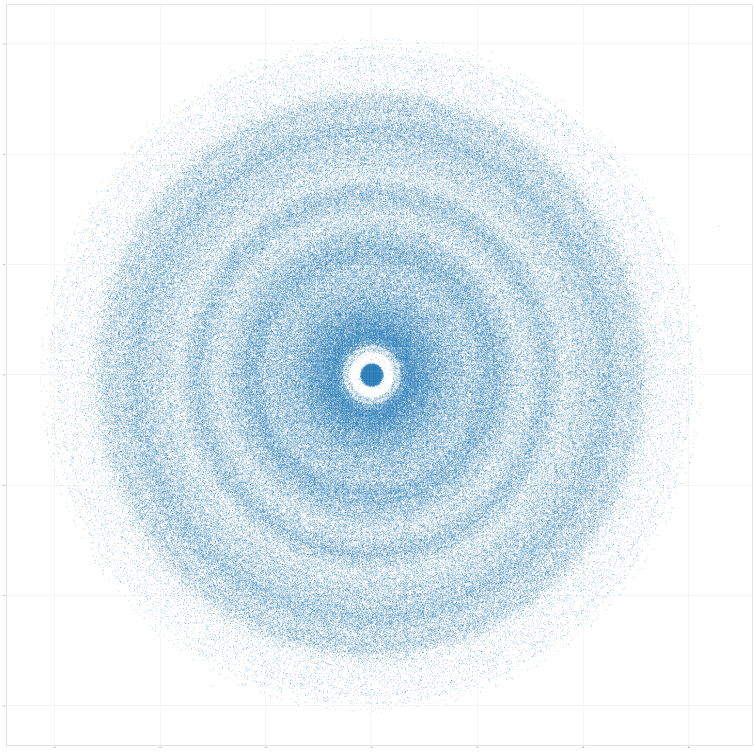


Figure 5.9: Constellation image from an ESP32 WiFi signal

Amplitude

The amplitude images can help with the analysis of tiny variations in the amplitude of the signal over time, which can then be used as RFF features. These plots depict the signal's amplitude changes, revealing distinct patterns or anomalies that can be traced to certain devices or transmitters.

Listing 5.5 contains the code that loads complex-valued IQ data, calculates signal amplitude, and plots the amplitude of the signal over time. An example of a plotted image is depicted in figure 5.10.

```
1 data = np.fromfile(f, np.complex64)
2 signal = np.abs(dat.real + 1j * dat.imag)
3 plt.rcParams["figure.figsize"] = (100,100)
4 plt.plot(signal)
```

Listing 5.5: Amplitude Plot

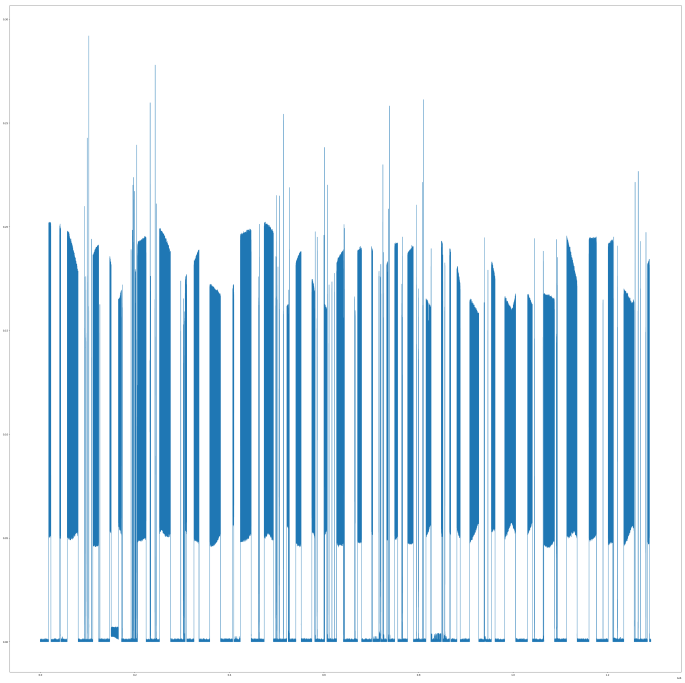


Figure 5.10: Amplitude image from an ESP32 WiFi signal

Power Spectral Density

The PSD is an interesting feature for RFF since it displays the distribution of signal strength across a range of frequencies, revealing the distinctive characteristics of each wireless device.

Listing 5.6 provides a code for computing and displaying the PSD of RF signals. Figure 5.11 depicts an example of a generated PSD image.

```
data = np.fromfile(filename, np.complex64)
PSD = np.abs(np.fft.fft(data))*2 / (NFs)
PSD_log = 10.0*np.log10(PSD)
PSD_shifted = np.fft.fftshift(PSD_log)
f += center_freq
plt.rcParams["figure.figsize"] = (100, 100)
plt.plot(f, PSD_shifted)
```

Listing 5.6: Power Spectral Density Plot

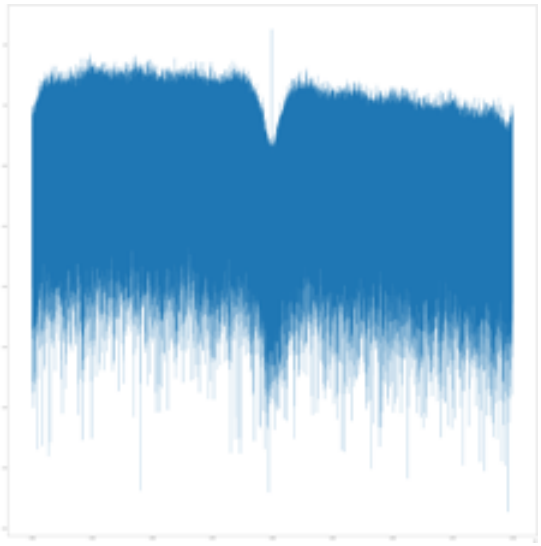


Figure 5.11: PSD image from an ESP32 WiFi signal

Differential Constellation

Differential constellation images are also employed in machine learning applications. Listing 5.7 shows a code used to generate differential constellation images. The complex data is imported via the IQ file in this approach, and the differential signal is computed by multiplying each sample by the complex conjugate of the next sample, followed by a scatter plot of the real and imaginary components of the differential signal. Figure 5.12 depicts the resultant image.

```
data = np.fromfile(file, np.complex64)
I_data = data.real
Q_data = data.imag
n = 1
differential_signal = []
for i in range(len(I_data) - n):
    X_t = I_data[i] + 1j * Q_data[i]
    X_tn = I_data[i+n] + 1j * Q_data[i+n]
    D_t = X_t * np.conj(X_tn)
    differential_signal.append(D_t)
plt.rcParams["figure.figsize"] = (100,100)
plt.scatter(np.real(differential_signal), np.imag(
    differential_signal))
```

Listing 5.7: Differential Constellation Plot

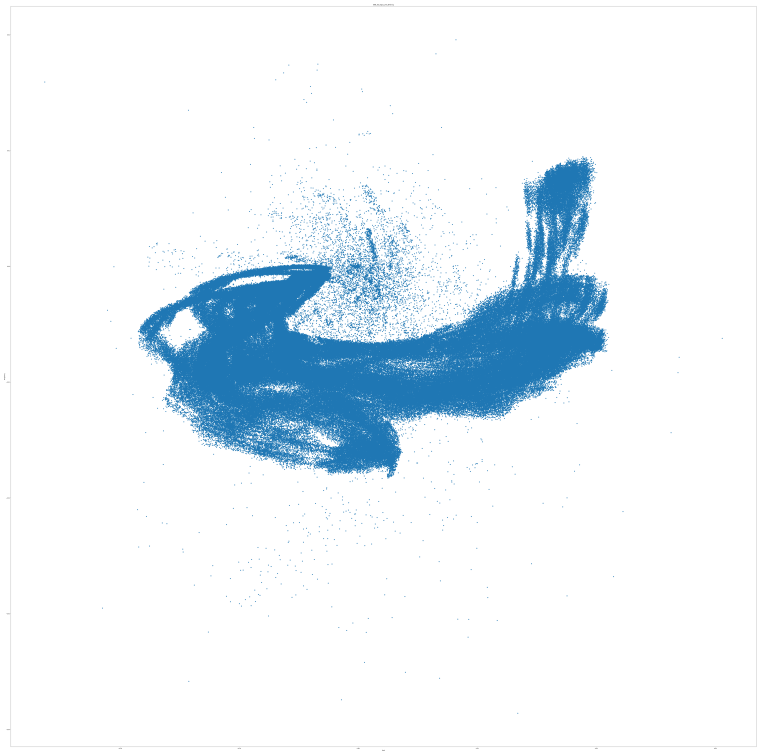


Figure 5.12: Differential constellation image from an ESP32 WiFi signal

Spectrogram

Spectrogram images of a wireless signal can give important insights on its frequency content and modulation features, which can aid in the process of producing and analysing its RFF.

The code in Listing 5.8 can be used to read the complex numbers and plot the spectrogram of the signal, yielding the image shown in Figure 5.13.

```
1 data = np.fromfile(f, np.complex64)
2 plt.specgram(data, NFFT=1024, Fs=2000000)
```

Listing 5.8: Spectrograms Plot

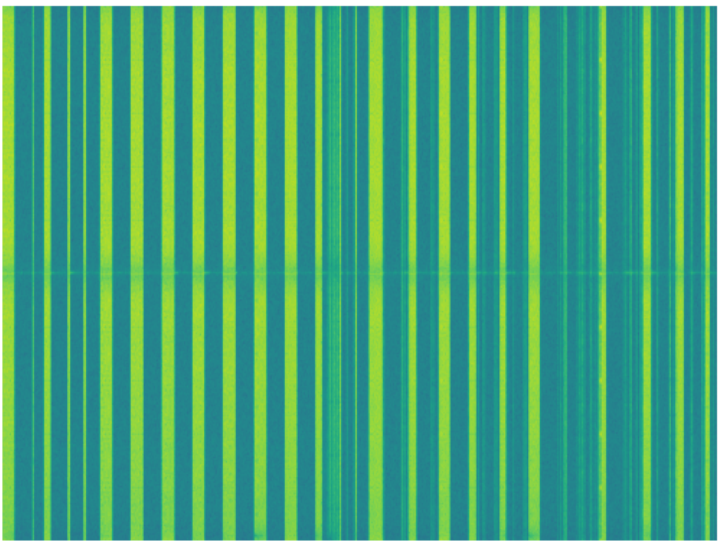


Figure 5.13: Spectrogram image from an ESP32 WiFi signal

Considering the offered features have diverse characteristics, their effectiveness will be dependent on the communication protocol of the communicating device. The attributes of each communication protocol are analysed, and the ones with the best performance are used. This ensures that the best results are obtained and that ML resources are not wasted on inefficient features.

Final Dataset Considerations

As mentioned in section 5.3.1, the dataset for each of the communication methods (WiFi and 433 MHz) ended up with 500 samples per device. From these 500 samples, 2500 images were generated per device, corresponding to the 5 features presented. Next, data augmentation techniques were used to increase the number of images without needing to increase the dataset obtained. Data augmentation is a well-known strategy, particularly for image-based applications, because it artificially increases the diversity and size of the training dataset. By making various adjustments to the input images, it is possible to generate new training samples that aid the ML model in learning more generalised features.

The method for implementing data augmentation is outlined below and shown in Listing 5.9:

1. The input image file is first opened using the Python Imaging Library (PIL) function *Image.open*.

2. To resize the image to the proper dimensions (224 × 224), the *resize()* function was used. Because of this standardisation, the ML model will always receive images with the same input shape.

3. Each image generates three new augmented images by rotating the existing one 90 degrees three times. This is performed by executing the *rotate()* function with an angle of '90 * (i+1)' degrees while iterating through a range of three (i=0,1,2). As a result, the original image is produced in three rotated variants at 90, 180, and 270 degrees.

4. Using the *save()* function, the rotated photos are then stored with a new file name that includes the augmentation index.

```
im = Image.open(file_name)
im = im.resize((224, 224))

for i in range(3):
    rotated_im = im.rotate(90 * (i+1))
    save_path = os.path.splitext(file_name)[0] + f'_augmented_{i}.
    png'
    rotated_im.save(save_path)
```

Listing 5.9: Data augmentation script

After data augmentation, images quadrupled to 10000 in each device. This means that the total dataset for each of the communication methods contains 40000 images.

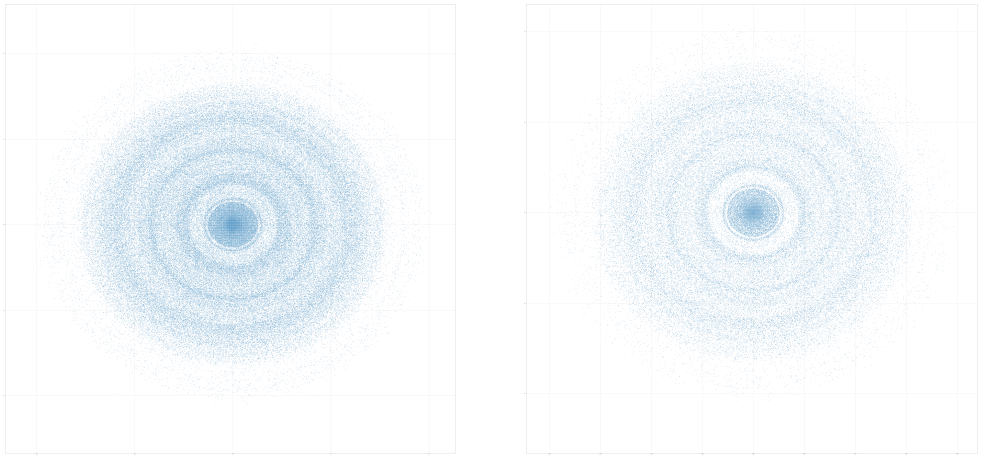
Chapter 6

Results

6.1 Feature Extraction Evaluation

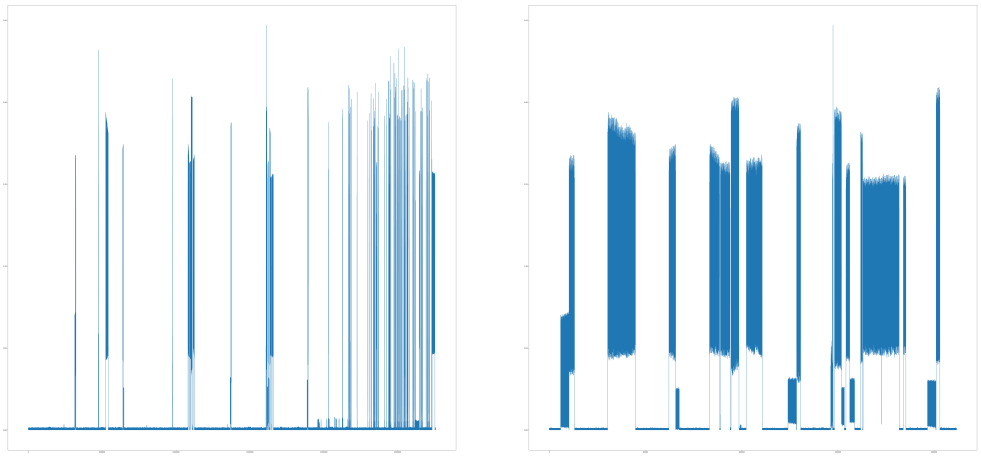
The Signal Acquisition and Feature Extraction Architecture produced very satisfactory results, both in terms of creating better images to feed the ML model and in terms of resource management, more properly disk storage management.

Figures 6.1, 6.2, 6.3, 6.4, and 6.5 display images generated from the capture of WiFi signals for each of the features worked on. The images obtained directly from raw signal capture and images generated by the developed architecture can be compared in each of these figures. Just by observing and comparing the images, it is possible to perceive an evident reduction in signal noise, since they are less saturated and have greater detail. A high saturation in these images indicates the existence of an excessive amount of noise in the signal, making it difficult to extract signal features from them.



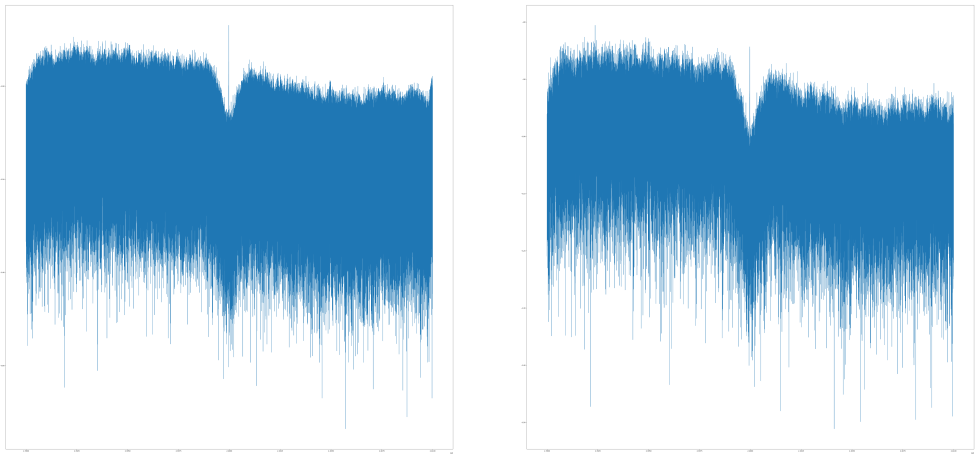
(a) (b)

Figure 6.1: Constellation image before (a) and after (b) custom block



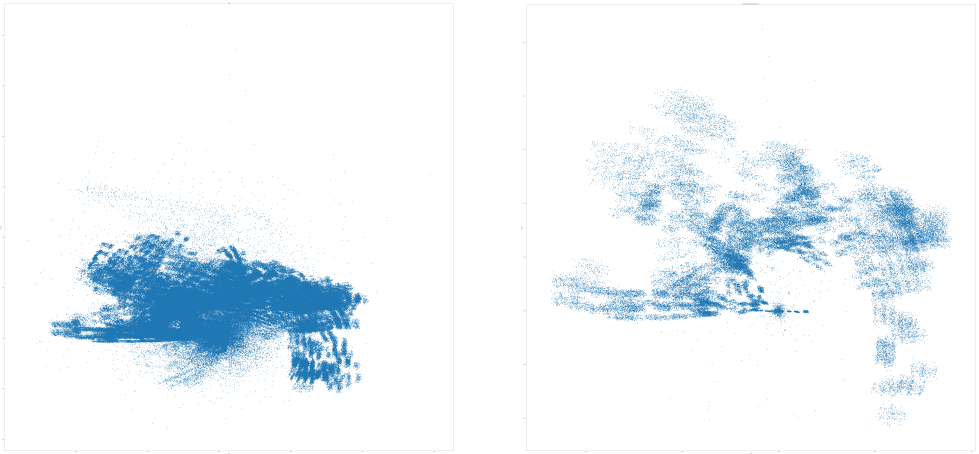
(a) (b)

Figure 6.2: Amplitude image before (a) and after (b) custom block



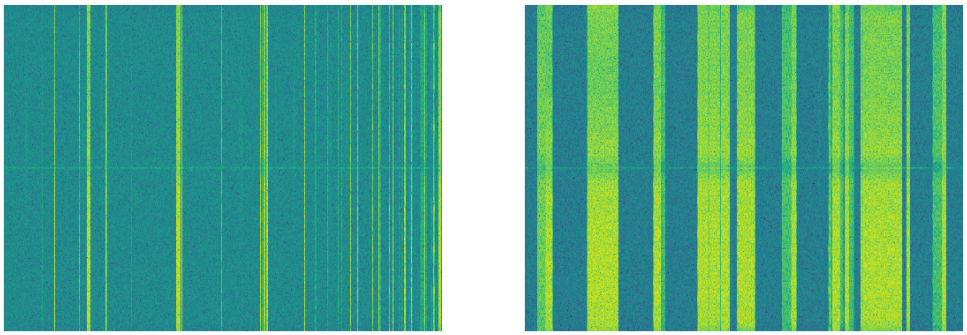
(a) (b)

Figure 6.3: PSD image before (a) and after (b) custom block



(a) (b)

Figure 6.4: Differential constellation image before (a) and after (b) custom block



(a) (b)

Figure 6.5: Spectrogram image before (a) and after (b) custom block

The results of the ML model developed by the project colleague were used to validate the developed custom block. The evaluation metrics for each of the features used to feed the ML model for WiFi device classification are shown in Tables 6.1 and 6.2. Table 6.1 represents the evaluation performed prior to the implementation of the custom block, while Table 6.2 represents the evaluation performed after the implementation of the custom block. The metrics presented are accuracy and loss.

Accuracy is a method for assessing the performance of a classification model. It is most commonly expressed as a percentage. Accuracy is defined as the number of predictions where the predicted value equals the true value. During the training phase, accuracy is frequently graphed and monitored, though the value is frequently associated with overall or final model accuracy.

A loss function, also known as a cost function, considers the probabilities or uncertainty of a prediction based on how far it differs from the true value. This provides a more nuanced picture of how well the model is performing. Loss, unlike accuracy, is not a percentage; rather, it is the sum of the errors made for each sample in training or validation sets. In the training process, loss is frequently used to find the "best" parameter values for the model. The goal of the training process is to reduce this value [56].

Feature	Accuracy	Loss
Constellation	89,00%	0,3367
PSD	71,84%	0,7434
Spectrogram	62,95%	0,8925
Amplitude	76,32%	0,7107
Differential Constellation	74,42%	0,6611

Table 6.1: Feature evaluation before custom block for WiFi devices

Feature	Accuracy	Loss
Constellation	90,87%	0,2566
PSD	76,58%	0,5781
Spectrogram	71,53%	0,7213
Amplitude	74,90%	0,7790
Differential Constellation	80,47%	0,5127

Table 6.2: Feature evaluation after custom block for WiFi devices

Tables 6.3 and 6.4 show the evaluation metrics for each of the features used to feed the ML model in the ER400TRS transcievers classification case. Table 6.3 refers to the evaluation done before the custom block implementation, and Table 6.4

refers to the evaluation done after the custom block implementation. The presented metrics are also accuracy and loss.

Feature	Accuracy	Loss
Constellation	96,76%	0,0945
PSD	93,75%	0,2275
Spectrogram	85,50%	0,2875
Amplitude	79,00%	0,4945
Differential Amplitude	78,47%	0,4469

Table 6.3: Feature evaluation before custom block for ER400TRS transcievers

Feature	Accuracy	Loss
Constellation	98,37%	0,0430
PSD	95,00%	0,1608
Spectrogram	88,00%	0,2480
Amplitude	80,62%	0,5137
Differential Amplitude	85,58%	0,3986

Table 6.4: Feature evaluation after custom block for ER400TRS transcievers

This evidence can be further corroborated by the graphs in figures 6.6 and 6.7 for WiFi, as well as in figures 6.8 and 6.9 for ER400TRS devices. These graphs show the accuracy gains and loss decreases regarding device identification by the ML model when it is fed with images from the developed architecture.

In the graph of figure 6.6, referring to the classification of devices communicating over WiFi, accuracy gains in all features are verified except amplitude, with an accuracy drop of 1,42%. This decrease is not considered problematic, since it was concluded that amplitude is not a good feature to use for this sort of communication. The feature is extremely sensitive to external factors such as noise and even the distance between the device and the SDR that captures the signal, which is undesirable. The percentage accuracy gains in the features Spectrogram (8,58%), Differential Constellation (6,05%), PSD (4,74%), and Constellation (1,87%) are notable.

Figure 6.7 depicts the loss decreases in classification of WiFi devices. Loss decreases in all features are verified: 0,0801 in Constellation, 0,1653 in PSD, 0,1712 in Spectrogram, 0,197 in Amplitude, and 0,1484 in Differential Constellation.

Figure 6.8 depicts the accuracy increases in classification of ER400TRS devices operating in the 433 MHz frequency band. Accuracy gains in all features are verified:

increases of 1,61% in Constellation, 1,25% in PSD, 2,5% in Spectrogram, 1,62% in Amplitude, and specially 7,11% in Differential Constellation.

After reviewing Figure 6.7, it is clear that the noise reduction block has significantly contributed to the decrease in loss values for the majority of features, demonstrating its importance in optimising the system's performance. Furthermore, the poor performance of the Amplitude feature emphasises the importance of selecting appropriate features for RFF systems to ensure optimal performance.

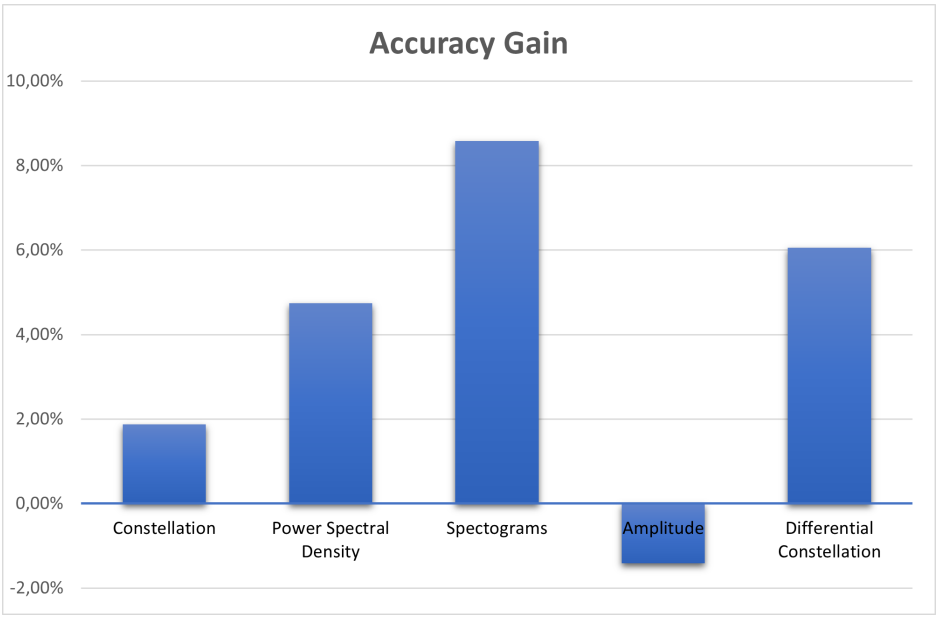


Figure 6.6: Accuracy gain in WiFi devices

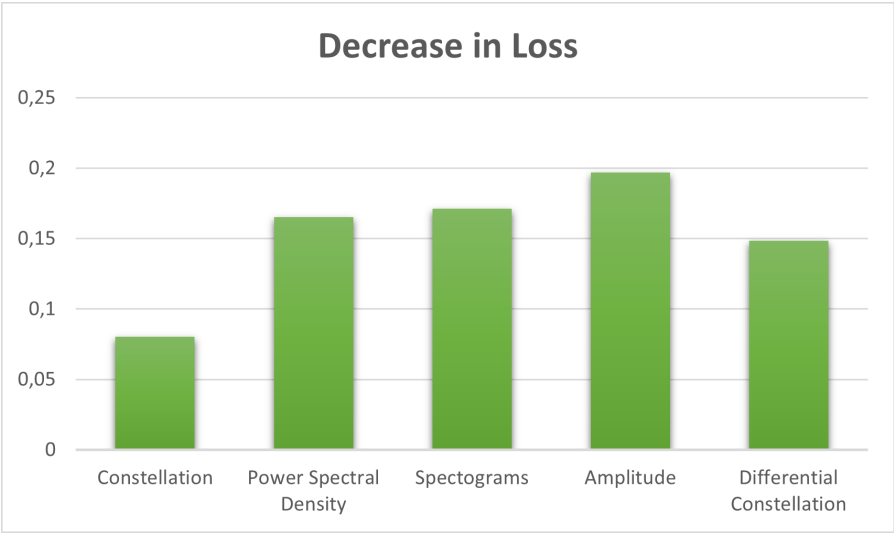


Figure 6.7: Loss decrease in WiFi devices

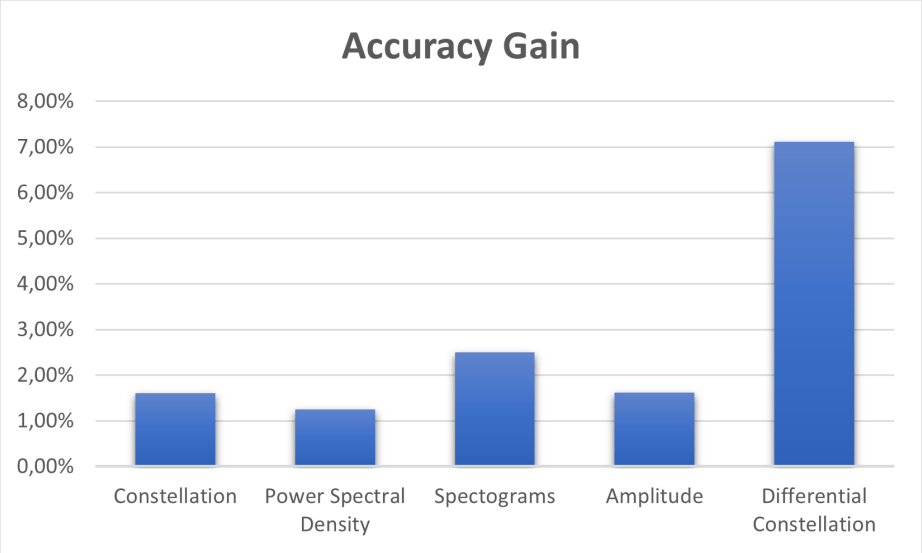


Figure 6.8: Accuracy gain in ER400TRS devices

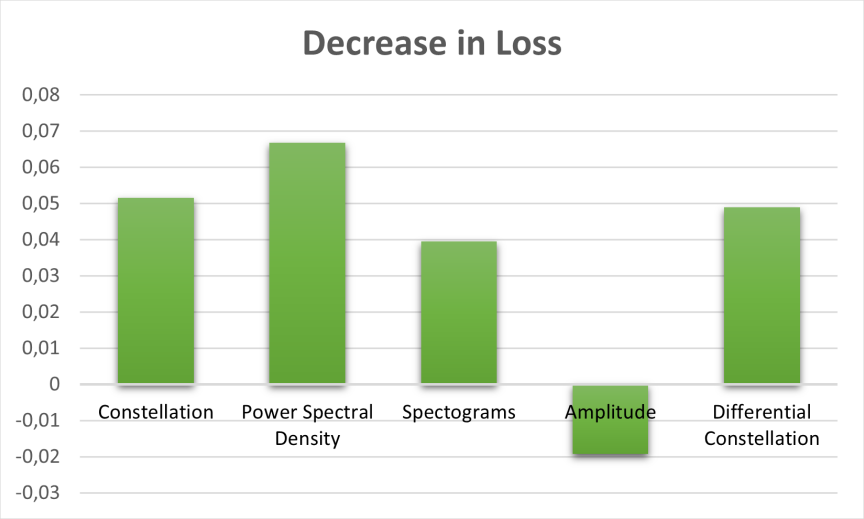


Figure 6.9: Loss decrease in ER400TRS devices

6.2 RFF Architecture Evaluation

6.2.1 Execution Time

The viability of this architecture also involves the certification that its execution time is satisfactory. If the authentication of the device takes too long, there will be a large time gap where it is not known whether the information transmitted by the device is reliable or not. As such, the execution time of the complete authentication process was evaluated, that is, from the signal capture process transmitted by the device up to its classification.

The two expected scenarios were considered: the scenario in which the WiFi device authenticates itself without problems and the scenario in which it is necessary to proceed to authentication using ER400TRS transcievers. Forty authentications were performed for the two envisaged scenarios.

It was concluded that, on average, the complete WiFi device authentication process takes 23,82 seconds. As for authentications that proceed to the second scenario using the ER400TRS transcievers, it was found that the average execution time is 37,21 seconds. These results are satisfactory considering that the execution times are not very long, allowing for device authentication during or shortly after data transmission to the gateway.

6.2.2 Final Classification

In order to assess the effectiveness of the architecture, an evaluation of the final results of the device classification was also performed. This evaluation was made by performing 40 authentications for each of the methods: WiFi devices authentication and ER400TRS transcievers authentication. The results of the authentications were examined, and it was determined whether the ML model correctly identified the communicating device. The obtained results are shown in Figure 6.10 (for WiFi devices authentication) and figure 6.11 (for ER400TRS transcievers authentication). It is possible to conclude that there was a high rate of true positives, indicating that the classifications were correct in the majority of cases for both authentication methods.

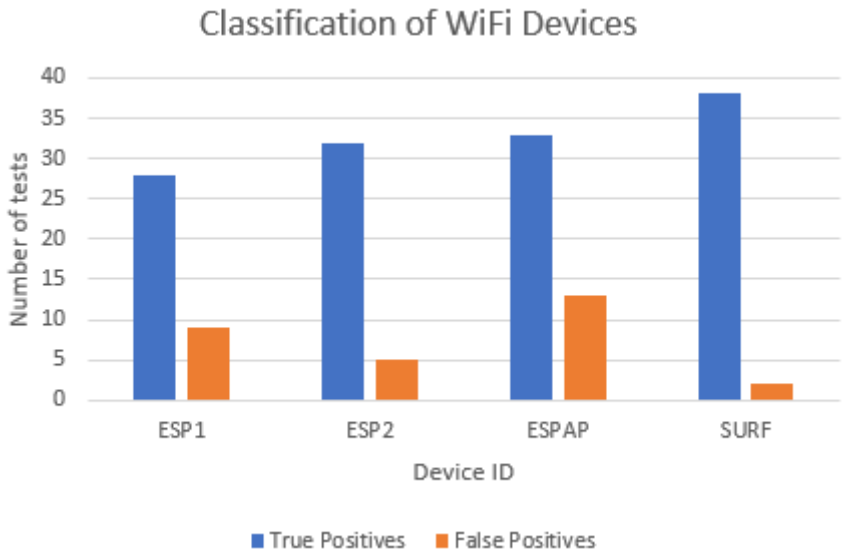


Figure 6.10: Classification of WiFi devices

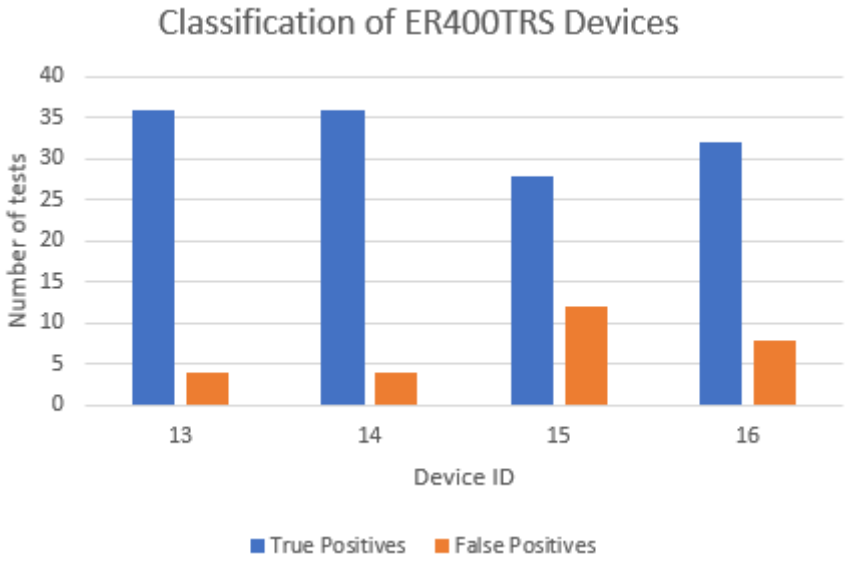


Figure 6.11: Classification of ER400TRS devices

Chapter 7

Conclusions

This thesis presented a SDR-based signal aquisition and processing architecture to enable RFF classification. It further proposed its integration over a real medical IoT gateway, to carry out device authentication. The resulting dataset, which was used to train an RFF ML model, is one of a kind because it takes advantage of the developed GNU Radio custom block. It demonstrates that SDR technology can effectively improve the feature extraction process from RF signals, despite the fact that this strategy has received little attention.

This architecture allows an improved collection of the desired signal, disregarding the noise present in its frequency range, which results in higher quality signal samples for device identification using RFF techniques.

This component fits into a larger scale architecture where it is framed with an ML component, developed by a project colleague, which feeds on the collected signals and is responsible for device classification. The effectiveness and relevance of this RFF architecture is proven by its integration in a medical gateway, allowing it to perform the authentication of the medical devices with which it communicates. This architecture demonstrates a very viable device authentication method for medical purposes, as medical systems need advanced cybersecurity methods to protect the confidential vital data they process.

RFF has emerged as a potential approach for authentication by leveraging the distinctive features of RF signals produced by hardware devices. However, no system is without faults, and possible vulnerabilities must be thoroughly examined before they can be exploited. In this context, we will look at a system that uses a CNN

model for RF fingerprinting and is designed to authenticate four different devices. Despite its unique approach, this system has two major security flaws that should be addressed: sensitivity to device imitation and inability to handle overlapping RFF from concurrent connection attempts. Understanding these flaws is critical to improving the security of RFF-baseds identification systems.

One of the most identifiable security flaws is the system's susceptibility to device impersonation. Because the CNN model is trained exclusively on four distinct classes corresponding to the four devices, it is fundamentally incapable of identifying unknown or novel devices. The model assumes that any device attempting to communicate is one of the four known devices. Although a rogue device will not normally pass RFF authentication, the signal it transmits may be similar enough to one of the classified devices. As a result, in rare cases, the system may misidentify it and grant access to an unauthorised device. To exploit this vulnerability, the rogue device would also need to duplicate the IP and MAC addresses of the device it is attempting to pass through in order for this anomaly to go undetected by the packet monitoring script, making the success of this attack highly unlikely.

The system's capacity to manage simultaneous communication attempts from various devices is the system's second key security flaw. The CNN model is currently incapable of distinguishing between many overlapping RF fingerprints. When two or more devices try to communicate at the same time, their RF fingerprints overlap, resulting in a complicated pattern that the model cannot comprehend. This could lead to inaccurate device identification and the danger of unauthorised device access, or it could result in genuine devices being denied access.

Future work could include creating a more complete dataset to make the defined architecture more robust. This is possible if the dataset is enriched with signals from a greater number of devices, as well as signals captured in various environments and with different types of noise. This enables the ML model to perform RFF on more devices and in more diverse environments.

To further validate the RFF authentication process, one could test with more devices by carying out spoofing and replay attacks, hence validating the system. This work is currently in progress, and the system will be further tested to validate its protection against the types of attacks it intends to tackle.

Another idea for scaling up this project would be the adaptation of this medical authentication architecture into other types of systems that have similar security requirements and deal with IoT communications that require device authentication, such as several kinds of industry 4.0 applications.

References

[1] F. Al-Turjman, M. H. Nawaz, and U. D. Ulusar, "Intelligence in the internet of medical things era: A systematic review of current and future trends," *Computer Communications*, vol. 150, pp. 644–660, 2020. [Cited on page 1]

[2] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi, "An overview of iot sensor data processing, fusion, and analysis techniques," *Sensors*, vol. 20, no. 21, 2020. [Cited on page 1]

[3] Statista1, "Internet of things - number of connected devices worldwide 2015-2025." Available at `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`, 2023. (Last accessed in 06/05/2023). [Cited on pages ix, 1, and 2]

[4] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "Iot privacy and security: Challenges and solutions," *Applied Sciences*, vol. 10, no. 12, 2020. [Cited on page 2]

[5] A. M. Gamundani, A. Phillips, and H. N. Muyingi, "An overview of potential authentication threats and attacks on internet of things(iot): A focus on smart home applications," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 50–57, 2018. [Cited on pages ix, 2, and 3]

[6] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things: a review," *Security and Communication Networks*, vol. 2018, pp. 1–9, 2018. [Cited on page 3]

[7] S. S. Mishra and A. Rasool, "Iot health care monitoring and tracking: A survey," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1052–1057, 2019. [Cited on page 3]

[8] Statista, "Number of internet of things (iot) active connections in healthcare in the european union (eu) in 2016, 2019, 2022 and 2025." Available at `https://www.statista.com/statistics/691848/iot-active-connections-in-healthcare-in-the-eu/`, 2023. (Last accessed in 07/05/2023). [Cited on pages ix, 3, and 4]

[9] R. Somasundaram and M. Thirugnanam, "Review of security challenges in healthcare internet of things," *Wireless Networks*, vol. 27, pp. 5503–5509, 2021. [Cited on pages 4 and 7]

[10] S. A. Butt, J. L. Diaz-Martinez, T. Jamal, A. Ali, E. De-La-Hoz-Franco, and M. Shoaib, "Iot smart health security threats," in *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, pp. 26–31, 2019. [Cited on page 4]

[11] E. Lomba, R. Severino, and A. F. Vilas, "Work in progress: Towards adaptive rf fingerprint-based authentication of iiot devices," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, IEEE, 2022. [Cited on page 5]

[12] R. Hireche, H. Mansouri, and A.-S. K. Pathan, "Security and privacy management in internet of medical things (iomt): A synthesis," *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 640–661, 2022. [Cited on pages xi, 8, 9, 10, 11, and 12]

[13] mPython, "Internet of things." Available at `https://mpython.readthedocs.io/en/master/tutorials/advance/iot/index.html`, 2023. (Last accessed in 28/06/2023). [Cited on pages ix and 10]

[14] P. K. Sadhu, V. P. Yanambaka, and A. Abdelgawad, "Internet of things: Security and solutions survey," *Sensors*, vol. 22, no. 19, 2022. [Cited on pages 12, 13, 14, and 15]

[15] A. H. Sodhro, A. I. Awad, J. van de Beek, and G. Nikolakopoulos, "Intelligent authentication of 5g healthcare devices: A survey," *Internet of Things*, vol. 20, p. 100610, 2022. [Cited on pages 15 and 16]

[16] G. Li, J. Yu, Y. Xing, and A. Hu, "Location-invariant physical layer identification approach for wifi devices," *IEEE Access*, vol. 7, pp. 106974–106986, 2019. [Cited on page 16]

[17] J. A. Gutierrez del Arroyo, B. J. Borghetti, and M. A. Temple, "Considerations for radio frequency fingerprinting across multiple frequency channels," *Sensors*, vol. 22, no. 6, 2022. [Cited on page 16]

[18] S. Wang, H. Jiang, X. Fang, Y. Ying, J. Li, and B. Zhang, "Radio frequency fingerprint identification based on deep complex residual network," *IEEE Access*, vol. 8, pp. 204417–204424, 2020. [Cited on page 17]

[19] R. Akeela and B. Dezfouli, "Software-defined radios: Architecture, state-of-the-art, and challenges," *Computer Communications*, vol. 128, pp. 106–125, 2018. [Cited on pages 17, 25, and 26]

[20] T. R. Smith, "Comparing rf fingerprinting performance of hobbyist and commercial-grade sdrs," Master's thesis, Wright State University, Ohio, 2020. [Cited on page 17]

[21] M. Lichtman, "Iq sampling." Available at `https://pysdr.org/content/sampling.html`, 2022. (Last accessed in 11/12/2022). [Cited on page 18]

[22] S. Katz and J. Flynn, "Using software defined radio (sdr) to demonstrate concepts in communications and signal processing courses," in *2009 39th IEEE Frontiers in Education Conference*, (San Antonio, TX, USA), pp. 1–6, October 2009. [Cited on page 18]

[23] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar, "A review of radio frequency fingerprinting techniques," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, pp. 222–233, 2020. [Cited on page 18]

[24] C. Zhao, T.-Y. Chi, L. Huang, Y. Yao, and S.-Y. Kuo, "Wireless local area network cards identification based on transient fingerprinting," *Wireless Communications and Mobile Computing*, vol. 13, no. 7, pp. 711–718, 2013. [Cited on page 19]

[25] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, (New York, NY, USA), p. 116–127, Association for Computing Machinery, 2008. [Cited on page 19]

[26] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid rf fingerprint extraction and device classification scheme," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 349–360, 2019. [Cited on page 19]

[27] F. Galtier, R. Cayre, G. Auriol, M. Kâaniche, and V. Nicomette, "A psd-based fingerprinting approach to detect iot device spoofing," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 40–49, 2020. [Cited on page 20]

[28] J. Li, Y. Ying, C. Ji, and B. Zhang, "Differential contour stellar-based radio frequency fingerprint identification for internet of things," *IEEE Access*, vol. 9, pp. 53745–53753, 2021. [Cited on pages ix and 20]

[29] G. Baldini, G. Steri, R. Giuliani, and C. Gentile, "Imaging time series for internet of things radio frequency fingerprinting," in *2017 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, IEEE, 2017. [Cited on page 21]

[30] D. Liu, M. Wang, and H. Wang, "Rf fingerprint recognition based on spectrum waterfall diagram," in *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 613–616, IEEE, 2021. [Cited on page 21]

[31] E. Uzundurukan, Y. Dalveren, and A. Kara, "A database for the radio frequency fingerprinting of bluetooth devices," *Data*, vol. 5, no. 2, 2020. [Cited on page 22]

[32] A. Jagannath, Z. Kane, and J. Jagannath, "Real-world commercial wifi and bluetooth dataset for rf fingerprinting," 2022. [Cited on page 23]

[33] G. Shen, J. Zhang, and A. Marshall, "Lora$_r ffi_d ataset,''$ 2022. [$Citedonpage$23]

[34] J. de Jesús Rugeles Uribe, E. P. Guillen, and L. S. Cardoso, "A technical review of wireless security for the internet of things: Software defined radio perspective," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 4122–4134, 2022. [Cited on pages xi, 26, and 28]

[35] E. Research, "Usrp n320." Available at `https://www.ettus.com/all-products/usrp-n320/`. (Last accessed in 26/1/2023). [Cited on pages ix and 26]

[36] A. Devices, "Adalm-pluto." Available at `https://wiki.analog.com/university/tools/pluto`. (Last accessed in 01/02/2023). [Cited on page 27]

[37] A. Devices, "Adalm-pluto." Available at `https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview`. (Last accessed in 26/01/2023). [Cited on pages ix and 27]

[38] R. Gandhiraj, R. Ram, and K. Soman, "Analog and digital modulation toolkit for software defined radio," *Procedia Engineering*, vol. 30, pp. 1155–1162, 2012. International Conference on Communication Technology and System Design 2011. [Cited on page 29]

[39] G. Radio, "Your first flowgraph." Available at `https://wiki.gnuradio.org/index.php?title=Your_First_Flowgraph`. (Last accessed in 17/04/2023). [Cited on page 29]

[40] B. Duggan, "Funcube 2 meter nb fm fg." Available at `https://wiki.gnuradio.org/index.phptitle=File:FunCube_2_meter_NB_FM_fg.png`, 2019. (Last accessed in 15/12/2022). [Cited on pages ix and 29]

[41] G. Radio, "Out of tree modules." Available at `https://wiki.gnuradio.org/index.php/OutOfTreeModules`. (Last accessed in 01/2/2023). [Cited on page 29]

[42] G. Radio, "Types of blocks." Available at `https://wiki.gnuradio.org/index.php/Types_of_Blocks`. (Last accessed in 15/04/2023). [Cited on page 30]

[43] F. Alsubaei, A. Abuhussein, V. Shandilya, and S. Shiva, "Iomt-saf: Internet of medical things security assessment framework," *Internet of Things*, vol. 8, p. 100123, 2019. [Cited on pages 30 and 31]

[44] Z. B. Caldwell, "The case for a security metric framework to rate cyber security effectiveness for internet of medical things (iomt)," in *Women Securing the Future with TIPPSS for Connected Healthcare: Trust, Identity, Privacy, Protection, Safety, Security*, pp. 63–81, Springer, 2022. [Cited on pages ix and 31]

[45] HL7, "Fhir overview." Available at `https://www.hl7.org/fhir/overview.html`, 2023. (Last accessed in 09/05/2023). [Cited on page 33]

[46] G. Alterovitz, J. Warner, P. Zhang, Y. Chen, M. Ullman-Cullere, D. Kreda, and I. S. Kohane, "SMART on FHIR Genomics: facilitating standardized clinico-genomic apps," *Journal of the American Medical Informatics Association*, vol. 22, pp. 1173–1178, 07 2015. [Cited on page 33]

[47] D. Bender and K. Sartipi, "Hl7 fhir: An agile and restful approach to healthcare information exchange," in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pp. 326–331, 2013. [Cited on page 33]

[48] M. Babiuch, P. Foltýnek, and P. Smutný, "Using the esp32 microcontroller for data processing," in *2019 20th International Carpathian Control Conference (ICCC)*, pp. 1–6, 2019. [Cited on page 35]

[49] Easy-Radio, *Easy-Radio ER400TRS Transceiver*, 2003. [Cited on page 36]

[50] A. Developers, "Android debug bridge (adb)." Available at `https://developer.android.com/tools/adb`, 2023. (Last accessed in 10/06/2023). [Cited on page 39]

[51] G. Radio, "Plutosdr source." Available at `https://wiki.gnuradio.org/index.php/PlutoSDR_Source`. (Last accessed in 02/05/2023). [Cited on page 46]

[52] G. Radio, "Qt gui waterfall sink." Available at `https://wiki.gnuradio.org/index.php/QT_GUI_Waterfall_Sink`. (Last accessed in 04/05/2023). [Cited on page 47]

[53] G. Radio, "Qt gui frequency sink." Available at `https://wiki.gnuradio.org/index.php?title=QT_GUI_Frequency_Sink`. (Last accessed in 04/05/2023). [Cited on page 47]

[54] A. Devices, "Adalm-pluto detailed specifications." Available at `https://wiki.analog.com/university/tools/pluto/devs/specs`. (Last accessed in 04/05/2023). [Cited on page 48]

[55] T. O'Shea and N. West, "Radio machine learning dataset generation with gnu radio," *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016. [Cited on page 53]

[56] A. Developers, "Accuracy and loss." Available at `https://machine-learning.paperspace.com/wiki/accuracy-and-loss`, 2019. (Last accessed in 13/06/2023). [Cited on page 68]