Multi-purpose IoT Security Guard Edge Node

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor in CyberSecurity

Ahmed Rasheed Jamhour (320210605136)

Sami Mohammed Saad (320210605072)

Supervised by

Dr. Ashraf Mohammed Alkhresheh

Committee Member Names

Dr. Eman Al-Qtiemat

Dr. Shorouq Al-Eidi

June 2025

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to everyone who contributed to the successful completion of the Multi-purpose IoT Security Guard Edge Node project.

First and foremost, We would like to thank and praise Allah (God Almighty) for granting us the strength, patience, and guidance throughout the journey of completing this project. Without His countless blessings and support, none of this would have been possible.

We extend our heartfelt thanks to Tafila Technical University for providing our with the opportunity and resources to undertake this project. The support from the university administration has been invaluable.

We immensely grateful to our project advisor, Dr. Ashraf Alkhresheh, for his continuous guidance, insightful feedback, and unwavering support throughout the development of this platform. His expertise and encouragement have been instrumental in shaping the direction and success of this project.

Additionally, I appreciate the assistance and cooperation of the faculty and staff of the Computer Science and Engineering Departments.

We would also like to thank the developers and contributors of the open-source tools and libraries that we utilized in this project. Their work has significantly accelerated our project development process and enhanced the functionality of the platform.

Lastly, we extend our gratitude to our families and friends for their patience, understanding, and encouragement throughout this journey. Their support has been a constant source of motivation.

Thank you all for your unwavering support, encouragement, and belief in our vision.

Pledge

This is to certify that the project entitled “Multi-Purpose IoT Secuity Gurde Edge Node” has been completed in partial fulfillment of the requirements for the Bachelor’s degree in ***Cyber Security at the Department of Computer Science, Faculty of Information and Communication Technology, Tafila Technical University.***

All analysis, design, and system development work presented in this project was carried out by the undersigned. Furthermore, this project has not been submitted to any other college or university and remains the sole intellectual property of the project team. **No external party has any right to claim ownership of this work now or in the future.**

Student 1 (Ahmed Rasheed Jamhour)

Student 2 (Sami Mohammed Saad)

Table of Contents

Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Introduction

1.1 Overview

Our IoT Guard Edge Node (IoT-GEN) is a compact and cost effective device that emulates a defensive edge not guard against a wide range of wireless and radio frequency (RF) attacks. It serves as both an educational and prototyped protective platform. allowing cybersecurity students, professionals and researchers to emulate real-world and it's corresponding security controls without the need for expensive equipment or time-consuming setups. By integrating modern technologies, this project provides advanced security features, real-time signal analysis, and practical testing capabilities—empowering users to better understand vulnerabilities and develop robust defense strategies.

1.2 Project Motivation

The motivation behind the project originates from the growing need to protect wireless systems—especially car key remotes and smart home devices—from evolving radio frequency (RF) and Wi-Fi-based attacks. As daily life becomes increasingly dependent on these technologies[1], threats such as replay, jamming, and fixed-code cloning[2] have become more widespread and dangerous the following research questions shape the road map of our project:

Q1: What are the reasons behind our choice to develop this project?

We chose to develop this project to address the increasing vulnerabilities in wireless communication systems, particularly those operating at 433 MHz only, as it is the only widely available and legally permitted RF frequency band for public use in our country. These systems are often exploited using simple tools to perform replay or jamming attacks, which may result in unauthorized access to vehicles or property. Our goal is to build a low-cost, portable, and intelligent device using add reference for the reader to know what is raspberry pi zero 2W[3] and RF modules to emulate these attacks and—most importantly—develop detection and defense strategies in real-time.

Q2: Why is our project important?

This project is crucial because it empowers cybersecurity professionals, researchers, and hobbyists with a hands-on tool to simulate real-world threats and analyze wireless signal behaviors. It also acts as a protective mechanism by detecting jamming attempts, capturing unauthorized signals, and preventing misuse through pattern analysis. Furthermore, the integration of a Wi-Fi test and handshake attack module enhances the system's scope, enabling the testing of WPA2[4] security using deauthentication and handshake capture techniques[5].

Q3: What is the new idea that has been proposed by this project?

Our innovation lies in combining RF signal analysis, simple and lightweight machine learning-based pattern prediction, and real-time attack simulation into one integrated system. Features such as:

- Real-time RF signal spectrum monitoring
- Replay and cloning of fixed RF signals
- Jamming and jamming detection using FFT[6] analysis
- Anomaly-based behavior detection
- Wi-Fi handshake capture and analysis
- Simulated LSTM[6]-based pattern prediction and reuse

make this device a unique blend of offensive and defensive cybersecurity capabilities. It is not only a testbed for learning and research but also a potential base for developing future embedded RF security solutions.

1.3 Problem Statement

Modern wireless systems—such as car key remotes, smart door systems, and IoT-based home automation—heavily rely on unsecured radio frequency bands like 433 MHz for communication. These systems, while convenient, are highly susceptible to signal interception, replay attacks, fixed-code cloning[2], and jamming attempts. Unfortunately, most commercial security solutions are either expensive, limited in functionality, or fail to provide real-time defensive mechanisms tailored to these threats.

Several core problems emerge from this gap:

- **Lack of Real-Time Threat Visibility:** Most users and systems lack the tools to detect and visualize RF attacks such as replay or jamming in real-time.
- **No Affordable Testing Platform:** There is a scarcity of low-cost, open-source platforms that allow researchers or students to simulate real-world RF/Wi-Fi attacks and develop protective responses.
- **Weak Defense against Jamming and Cloning:** Existing devices typically cannot distinguish between valid signals and malicious interference.

Our project addresses these challenges by developing a multi-purpose IoT security device using affordable components like the Raspberry Pi Zero 2W and RF modules. The IoT-GEN provides tools to:

- **Capture and analyze RF signals**

- **Perform and detect jamming attacks**
- **Clone fixed-code remotes**
- **Visualize signal behavior in real-time**
- **Simulate Wi-Fi handshake attacks**
- **Leverage anomaly-based detection models**

Through this approach, we offer a practical, educational, and defensive solution to bridge the gap between RF attack simulation and effective cybersecurity response—empowering users to understand, detect, and mitigate threats within constrained environments.

1.4 Project Aim and Objectives

The primary aim of this project is to develop an affordable, real-time, and multi-functional IoT-based security device capable of simulating and detecting wireless attacks such as replay, jamming, cloning, and Wi-Fi handshake attacks. By leveraging a Raspberry Pi Zero 2W and RF modules, the system enables researchers and security analysts to better understand the behavior of wireless threats and evaluate practical defense techniques.

Objectives

- **Analyze Wireless Frequencies and Detect Attacks**
    - **Objective:** Capture and analyze RF signals—specifically at 433 MHz—to identify potential threats such as replay, jamming, and cloning.
    - **Outcom**e: Enable real-time threat detection through signal visualization and statistical filtering, providing better insight into wireless vulnerabilities.
- **Develop an Embedded Defensive IoT-GEN using Raspberry Pi**
    - **Objective**: Design and build a portable hardware-based security tool using Raspberry Pi Zero 2W, integrating RF modules, a TFT[7] screen, and joystick[8] navigation.
    - **Outcome**: Deliver a self-contained, interactive IoT-GEN for performing both offensive simulations and defensive actions directly in the field.

- **Simulate Real World Wireless Attacks to Better Understanding and Countering Them.**
    - **Objective**: Reproduce real-world attack scenarios such as fixed code replay, signal cloning, jamming, and WPA2 handshake capture to study their behavior and impact.
    - **Outcome**: Provide a controlled testing environment for researchers and students to learn attack techniques and evaluate security postures.
- **Implement Real World Protection Mechanisms against Wireless Threats**
    - **Objective**: Apply intelligent defense strategies such as jamming detection, anomaly-based behavior analysis, and selective blocking of

malicious signals.
  - **Outcome**: Equip the system with countermeasures that proactively defend against RF and Wi-Fi attacks, increasing resilience of wireless devises.

1.5 Project Scope

Boundaries

This project focuses on building an integrated RF and Wi-Fi security tool with offensive and defensive capabilities using a Raspberry Pi Zero 2W and various peripheral modules such as TFT[7] screen ,STX882[9] and SRX882[10]. The scope includes:

RF Signal Capturing and Analysis

- Real-time sampling and edge detection of 433 MHz signals.
- Conversion of captured signal into binary patterns.
- Signals filtering based on based on standard deviation and pulse thresholds.

RF-Based Attack Emulation

- Replay of captured fixed-code signals.
- Reactive jamming of legitimate car key remotes.
- Signal injection and cloning of basic remotes.

Wi-Fi Attack Module

- Scanning for WPA/WPA2 networks.
- Deauthentication and handshake capture.
- Saving .cap files for offline cracking analysis.

Security Response Features

- Jamming detection using FFT and noise percentage analysis.
- Behavioral anomaly detection for unauthorized RF usage.
- Real-time threat visualization and alerts.

User Interface

- Joystick-based navigation.
- Menu displayed on a 128x128 TFT screen.
- Simple interaction to launch tests and view results.

Responsibilities

Team Members

- **Hardware & Signal Processing Developer:** Sami Saad
  - Responsibilities: Setting up RF components, implementing signal capture and filtering mechanisms, handling jamming simulations, and integrating the power and control systems.
- **Software & Interface Developer:** Ahmad Rasheed
  - Responsibilities: Implementing the Wi-Fi attack module, designing the joystick-controlled graphical interface, managing system logic in Python,

and integrating defensive features such as jamming detection and signal pattern analysis.

- **Project Supervisor: Dr. Ashraf Alkhrasha**
    - Responsibilities: Providing technical guidance and oversight, reviewing implementation stages, ensuring the project meets cybersecurity and academic research standards.

1.6 Physical Design & Integration

The physical design of the **IoT-GEN** combines compact hardware integration with custom-built casing and optimized internal wiring to ensure portability, durability, and operational efficiency. The IoT-GEN was engineered to support RF and Wi-Fi attack simulation, real-time signal analysis, and active defense mechanisms within a fully enclosed and organized setup

Hardware Components and Wiring

The system is built around a **Raspberry Pi Zero 2W**, chosen for its small footprint, sufficient processing power, and GPIO versatility. The key components integrated into the system include:

- **Receivers and transmitters:**

Figure 1 shows the transmitter and receiver circuits that we used in this

RF Transmitter (STX882)[9]

The Transmitter requires power from a 3.3V source, so VCC was connected to 3.3V on the Pi. GND was connected to GND, and the data source was connected to GPIO20.

RF Receiver (SRX882)[10]

The Receiver requires power from a 3.3V source, so VCC was connected to 3.3V on the Pi. GND was connected to GND, and the data source was connected to GPIO21.

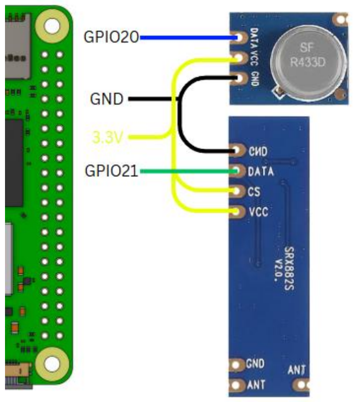


Fig. 1 Transmitter And Receiver

- **Control and display IoT Guard Edge Node:**

Figure 2 shows the TFT display and SPI interface that we used in this project.

TFT Display 128×128 (ST7735 driver) via **SPI**[7] interface:

SCK → GPIO11	SDA → GPIO10
DC → GPIO2	RES → GPIO25
CS → GPIO8	BL → GPIO18

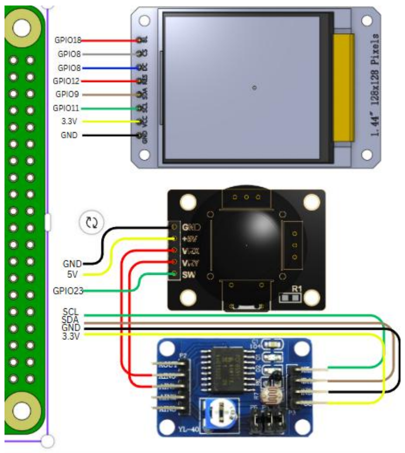


Fig. 2 Control And Display

Analog Joystick[8] connected via **ADC[12] I2C** module:

VRx → A0 VRy → A1

SW → GPIO16 SDA → GPIO2

SCL → GPIO3

- **Power connection shape:**

  Figure 3 shows the components of the power circuit that we used in this project.

  • **Lithium Battery (150mAh) – 3.7V** connected via **TP4056 charger** module(Type-C)

  • **Boost converter** to provide stable **5V** to the Raspberry Pi

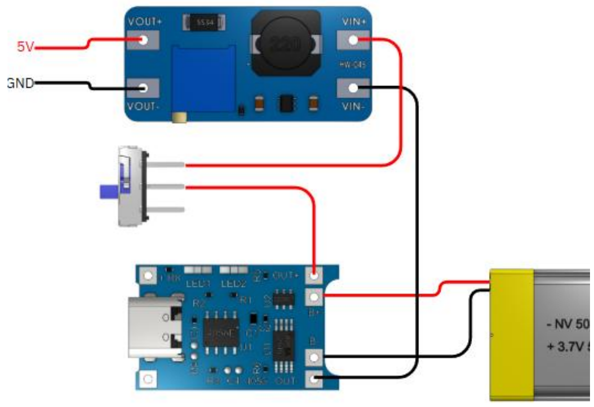


Fig. 3 Power circuit

3D Printed Enclosure

The enclosure of the IoT-GEN is 3D printed using durable PETG[10] material, ensuring it is both lightweight and robust. The design includes:

- Mount points for securing the Raspberry Pi and other components
- Dedicated ports for the TFT screen, joystick, and power connections
- A sliding lid for easy access and maintenance

Design tools such as CAD.OnShape and Blender were used for 3D modeling, and the case was printed using a standard FDM 3D printer with 0.2mm layer resolution as shown in figure 4.

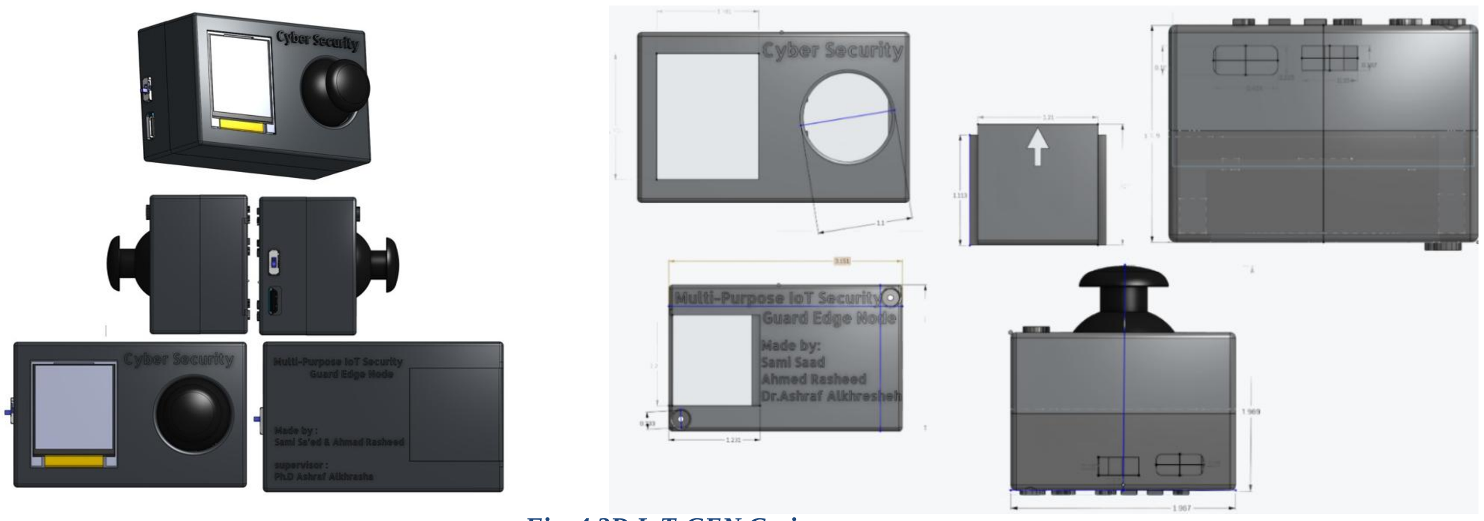


Fig. 4 3D IoT-GEN Casin

1.7 Software Stack and Integration

The software system is built using Python 3 on a customized Raspbian OS[13] environment.

Libraries and Tools Used

1. **pigpio** – For precise GPIO control[14].
2. **rpi_rf** – For decoding RF pulses into binary data[15].
3. **luma.lcd –** To display menus and signal data on the TFT screen[16].
4. **matplotlib, NumPy, scipy.fftpack** – For signal plotting, statistical analysis, and jamming detection via FFT[17].
5. **Adafruit_ADS1x15, smbus2** – For reading joystick movement through the ADS1115 module[18].
6. **subprocess, os, signal, threading** – For launching background processes (e.g., aircrack-ng), safe exits, and multitasking[19].
7. **systemd** – For running scripts automatically on boot[20].

We made all python scripts modular, with separate files including: RF capture, jamming control, Wi-Fi attack routines, and UI logic. The interface supports joystick-based navigation and provides live feedback from RF and Wi-Fi operations.

Operating System and Storage

The operating system that we chose to run on Raspberry Pi OS is **Raspbian**[13], a Debian-based distribution optimized for performance, security, and stability on embedded devises. Key reasons for choosing Raspbian:

1. Low resource consumption — ideal for Raspberry Pi Zero 2W
2. Full support for Python and GPIO libraries
3. Seamless integration with SPI, I2C, and UART peripherals
4. Customizability and wide community support

The IoT-GEN is equipped with a **64 GB** microSD card, providing ample storage for:

   a. Captured .cap files from Wi-Fi handshake attacks
   b. RF signal logs and analysis data
   c. System logs, Python modules, and updates

This setup allows the IoT-GEN to operate completely offline, log sensitive data securely, and maintain high reliability over extended periods of field

1.8 System Architecture and Functional Design

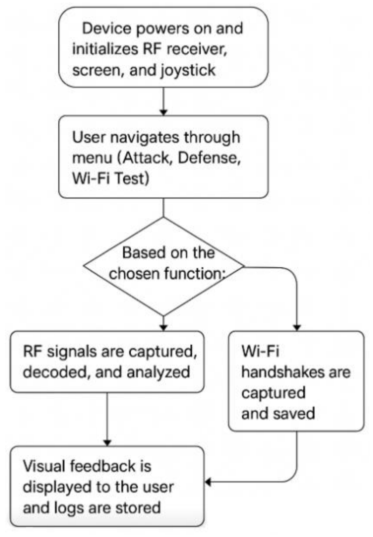
The architecture of the system is designed around a modular, layered approach that connects physical components with software modules to enable real-time signal analysis, attack simulation, and defense execution. The system is divided into the following functional layers:

1. **Input Layer – Signal Acquisition and User Interaction**
    - RF Receiver (SRX882): Continuously monitors 433 MHz band, capturing digital pulses from nearby RF remotes.
    - Joystick Module (via ADS1115): Provides analog from the user navigating between available features (e.g., tests, attack modes).
    - Wi-Fi Scanner: Detects nearby WPA2 networks during Wi-Fi attack testing.

2. **Processing Layer – Signal Analysis and Decision Logic**
    - Bit Conversion & Filtering Module:
      Converts captured RF pulses to binary patterns. Applies thresholds such as MIN_PULSES, STD_DEV, and bit length filters to eliminate noise.
    - Attack Emulation Engine:
      Performs replay of fixed-code signals, sends jamming bursts, or clones signals using RF transmitter.
    - Jamming Detection Module:
      Uses FFT to analyze spectrum noise. Detects spikes in background power to log jamming events.
    - Anomaly Based Behavior:
      Monitors signal patterns and behaviors over time to flag deviations or abnormal signal directions.
    - Rolling Code Pattern Emulation and Prediction:
      The system captures multiple signal sequences from the same car key to emulate rolling code behavior. By analyzing the differences between these captured patterns, the system attempts to learn the underlying logic used in the code progression. A custom LSTM-based machine learning model is then applied to predict future valid key patterns. This technique allows the IoT-GEN to perform educated guesses on the next rolling codes, enabling advanced analysis and highlighting the risks of predictive replay attacks.
3. **Output Layer – Interface and Feedback**
   TFT Display (128×128):
   Renders real-time RF status, user menus, threat alerts, and visual feedback.
   Data Logging:
   Handshake files, RF captures, and jamming logs are stored locally on the 64 GB microSD card And Google Drive use GoogleAPI .

4. **System Environment**
    - Core Controller: Raspberry Pi Zero 2W
    - Operating System: Raspberry Pi OS Lite (Raspbian)
    - Programming Language: Python 3
    - Storage: 64 GB microSD for persistent data logging
    - Startup Mechanism: Scripts run on boot via system command.

5. **Summary of System Flow**
   The system operational logic is clearly illustrated in Figure 5, which provides a comprehensive overview of the entire workflow.



Device powers on and initializes RF receiver, screen, and joystick
User navigates through menu (Attack, Defense, Wi-Fi Test)
Based on the chosen function:
RF signals are captured, decoded, and analyzed
Wi-Fi handshakes are captured and saved
Visual feedback is displayed to the user and logs are stored

Fig. 5 Data Flow Summary

1.9 Project Schedule

The development of the Multi-purpose IoT Security Guard Edge Node was divided into five main phases, ensuring that each attack type and defense mechanism was carefully studied, implemented, and validated through real testing as shown in table 1.

Table 1. Project Schedule

Phase	Duration	Description
Phase 1: Research & Setup	Mar 1 – Mar 15, 2025	Studying RF attack types, configuring Raspberry Pi OS, hardware wiring
Phase 2: Signal Capture & Replay	Mar 16 – April 10, 2025	Capturing car key signals, decoding fixed-code remotes, implementing replay
Phase 3: Attack Emulation	April 11 – May 1, 2025	Performing jamming, cloning, and Wi-Fi deauth/handshake attacks
Phase 4: Defense Mechanisms	May 1 – May 12, 2025	Developing jamming detection, anomaly behavior detection, alert system
Phase 5: Interface & Testing	May 13 – May 29, 2025	Building TFT interface, joystick control, running full IoT-GEN tests

Project Timeline Overview

1. Weekly Meetings: Every Tuesday from 10:00 AM to 11:30 AM.
2. Project Duration: From Mar 1, 2024, to May 30, 2025.

Final Project Submission

Date: June 2, 2025

CHAPTER 2: System Features and Functional Modules

Explanation of the GUI structure, test functions, attack simulation, and RF behaviors.

2.1 Intro Module

Purpose: Acts as a welcoming screen and navigation root.
Content: project title, and transition to main menu.
Notes: Smooth transition animations using joystick.

2.2 Security Part

Jamming Detection : Detects sudden noise spikes on 433 MHz and Alerts user visually and logs timestamp.
Capture My RF Key: Divided into 24-bit | 32-bit | 64-bit | 128-bit signal filters, each mode applies MIN_BITS_LEN / MAX_BITS_LEN [21] thresholds and saves the data to a file (key.txt).
Capture with Rolling Key: Captures multiple key presses from rolling code remotes, saves bit sequences and prepares them for analysis or **LSTM**-based pattern prediction.
Reuse My RF Key: Sends back captured signal through RF transmitter.

2.3 Attack Part

Jamming Attack: Sends noise continuously or during legitimate signal bursts and prevents car or door receiver from receiving valid signal.

Capture RF Key: Divided into 24-bit | 32-bit | 64-bit | 128-bit signal filters, each mode applies MIN_BITS_LEN / MAX_BITS_LEN thresholds and saves data to a file (key.txt).

Capture with Rolling Key: Captures multiple key presses from rolling code remotes, saves bit sequences and prepares them for analysis or **LSTM**-based pattern prediction.

Reuse My RF Key: Sends back captured signal through RF transmitter.

2.4 Wi-Fi test module

Scans for WPA/WPA2 networks, lets user choose SSID, sends deauth packets to target, captures handshake and saves .cap file, notifies user upon success.

CHAPTER 3: Signal Analysis and Experimental Results

3.1 Signal Acquisition Methodology

To perform accurate RF signal analysis, the IoT-GEN uses the SRX882 receiver connected to GPIO21 to monitor the 433 MHz frequency band. The incoming signal is sampled in real time using pigpio edge detection.

Each rising or falling edge is timestamped using microsecond-precision ticks. This raw sequence of ticks is used to:

- Calculate pulse widths
- Detect noise patterns
- Identify valid signal frames
- Trigger jamming detection or replay logic

The goal is to differentiate **between environmental noise** and actual signal pulses emitted by remote keys.

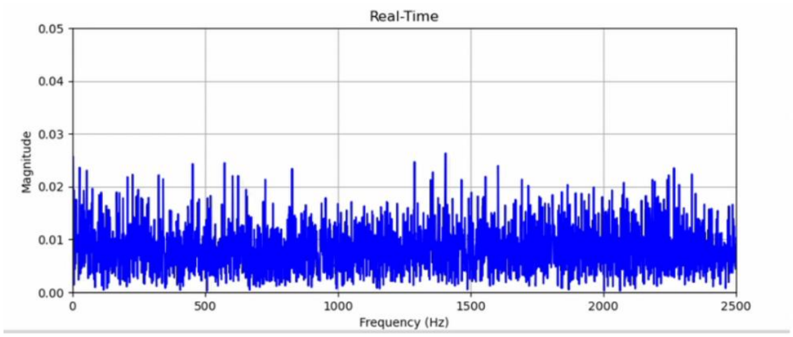


Fig. 7 Environmental Noise Signal

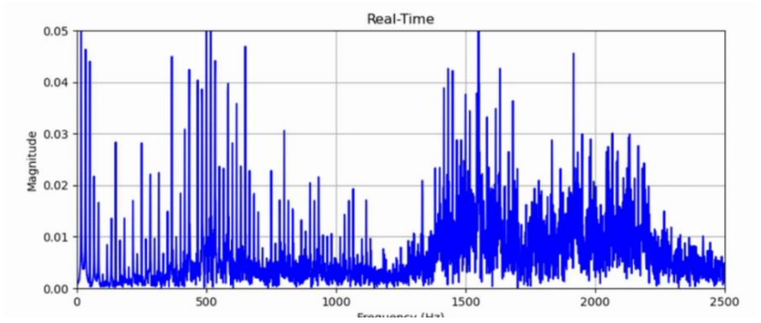


Fig. 6 Signal From The Car Key Without Filter Noise

3.2 Comparison Between Noise and Valid Signals

Table 2 lists the set of measurement criteria that we applied, their purpose along with example arbitrary values we.

Table 2. Filtering Criteria

Parameter	*Purpose*	*Example Value*
MIN_PULSES	*Minimum number of edges to consider "a frame"*	*20*
MAX_STD_DEV	*Max allowed µs variation among pulse intervals (noise filter)*	*200 µs*
MIN_BITS_LEN	*Minimum bit-pattern length after conversion*	*24 bits*
MAX_BITS_LEN	*Maximum bit-pattern length*	*96 bits*
REPEAT_SUPPRESSION_MS	*Ignore duplicate frames within this time*	*500 ms*

The following figures 8-9 demonstrate the signal after applying the filtering process. Environmental noise is significantly reduced, allowing for clearer identification of valid RF key signals.

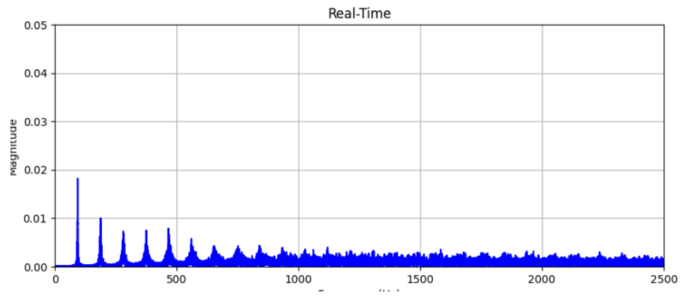


Fig. 9 Environmental Noise Filter Signal

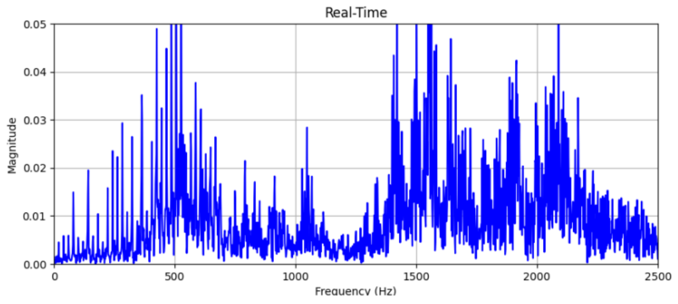


Fig.8 Signal From The Car Key With Filter Noise

3.3 Signal Pattern Extraction and Visualization

Once valid pulses are extracted, their durations are converted to binary using pulse width thresholds. Each signal frame is translated into a bit string (e.g., 101011010100…), which is then logged and analyzed.
The following process is applied:

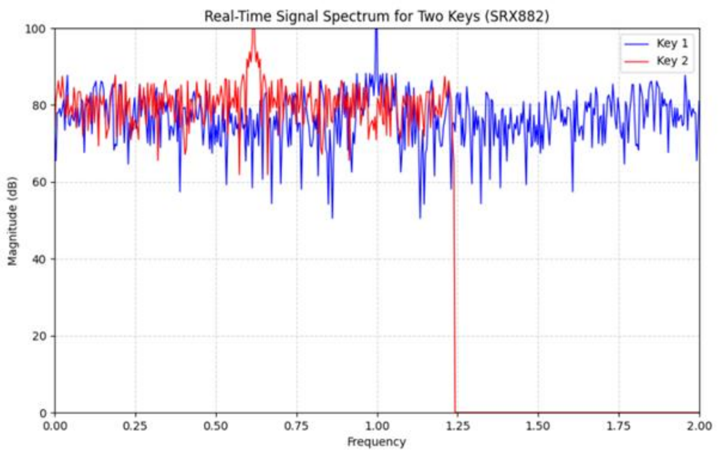
1. Capture edges (ticks)
2. Calculate pulse durations: This refers to measuring the time difference between consecutive rising and falling edges of the captured RF signal pulses. The result is the duration of each pulse in microseconds, which forms the basis for binary classification.

3. Apply median-based threshold
4. Convert to binary (0/1): A median-based thresholding method is used to digitize the pulse durations. If a pulse duration is greater than the median threshold, it is classified as ‘1’; otherwise, it is classified as ‘0’. This simplifies signal interpretation into a clear bit pattern.
5. Log and analyze
6. The valid keys are saved for cloning or replay in future tests.

3.4 Signal Interference and Multi-Key Analysis

To study real-world interference, we tested the system by activating two car key remotes simultaneously in the IoT-GEN proximity. Surprisingly, the system could distinguish between the overlapping signals because of:

- Differences in signal start times
- Slight variation in pulse shapes
- Time spacing between transmissions
- This confirmed that the minor signal overlaps do not necessarily corrupt valid signal reception.



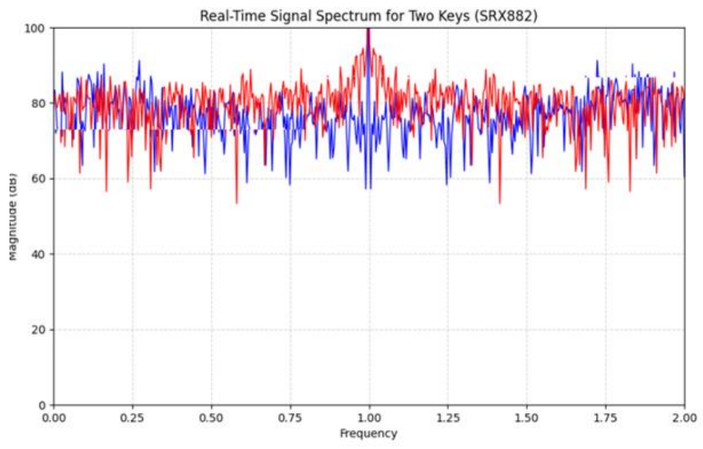


Fig.10 Overlapping RF Signals From Two Remotes

3.5 RF Jamming Behavior Analysis

During jamming attacks, the RF spectrum becomes saturated with high-density pulses. This prevents valid signals from being received properly. The jamming module emits continuous or burst-based signals via the STX882 transmitter. As shown in figure 11, we observed that:

- The FFT analysis shows a significant rise in noise floor
- All legitimate pulses become indistinguishable
- The receiver logs only unstable or short pulse patterns
- This behavior triggers the Jamming Detection Module, which logs the event and notifies the user via the screen for further manual analysis and human decision making.

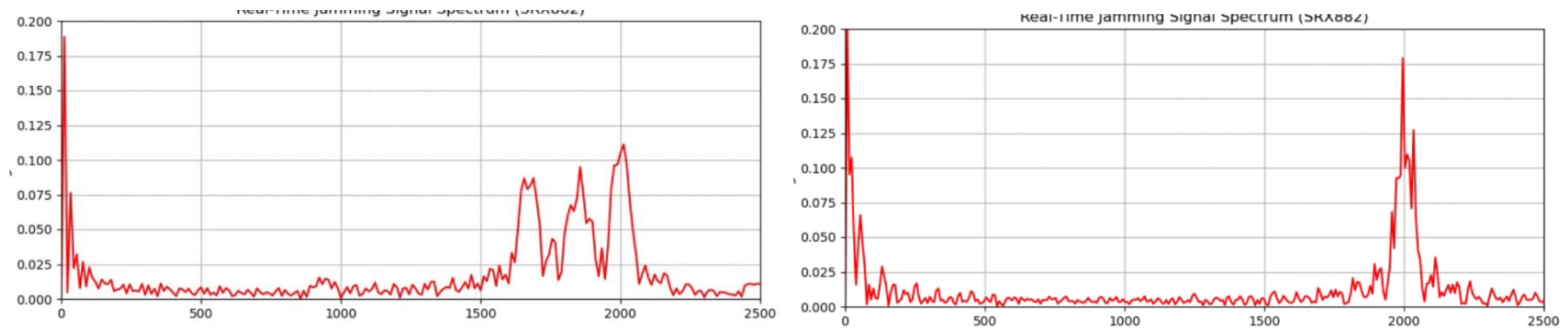


Fig.11 visualization during RF jamming

3.6 Summary of Experimental Findings

The filtering logic was effective in removing background RF noise and isolating valid transmissions to confirm our words through Figure 6-9.
Manual signal inspection revealed consistent patterns in fixed-code keys, while rolling-code keys showed variation across presses.
Interference analysis showed minimal risk in low-density environments, but jamming caused total signal failure.

CHAPTER 4: Defensive Mechanisms and Security Innovations

4.1 Overview

As part of the project’s second phase, we shifted from attack emulation to protection strategy. The result was the implementation of three innovative defensive mechanisms built directly into the same IoT-GEN. These mechanisms are designed to detect real-world wireless threats, respond autonomously, and enhance physical layer awareness—something not commonly found in commercial RF-based security systems.

The **three main** defense mechanisms we developed are:

1. Jamming Detection System.
2. Anomaly-Based Behavioral Detection.
3. Secure Data Storage and Owner-Only Access via Google Drive API.

4.2 Jamming Detection System

Jamming is a common attack where a IoT-GEN emits continuous or intermittent noise signals to prevent legitimate communication. Our detection system uses:

1. Fast Fourier Transform (FFT) on live RF data.
2. Baseline noise thresholding.
3. Dynamic signal stability analysis.

How it works

- The IoT-GEN continuously samples the RF environment at 433 MHz.
- If a significant increase in noise floor or sudden absence of valid pulses is detected, it flags the event.
- When noise levels exceed 68%, it is logged as a confirmed jamming attack This threshold was determined experimentally based on a local simulation we conducted. In our test, we simultaneously pressed four RF remotes in close proximity, generating a continuous stream of overlapping signals. The observed noise saturation under this extreme condition reached a maximum of 58%.

  In real-world scenarios, such synchronized interference is unlikely, as most RF key usage involves a single transmission at a time. To account for environmental variations and ensure reliable detection, we added a 10% safety margin, setting the jamming detection threshold at 68%, in the absence of standardized or cited values for environmental RF saturation rates.

Features

- Logs timestamp and signal stats
- Displays on-screen alert
- Stored securely for later audit. Captured signal data and timestamps are securely stored on the device and backed up for later audit.

4.3 Anomaly-Based Behavioral Detection

This is the most advanced and novel protection system we implemented. It introduces contextual awareness to our RF-based systems guard system , enabling the IoT-GEN to differentiate between a legitimate owner and a potential attacker based on their behavior in the car proximity.

Concept

By placing **multiple RF receivers** around the **car**, we can analyze:

1. Signal direction (which antenna first received the pulse)
2. Time of day and day of the week
3. Estimated signal distance
4. Angle of entry and unlock frequency

While this setup may introduce additional hardware costs, our designed guard system remains low-cost overall, as it utilizes affordable RF modules and minimal infrastructure.

Example Scenario:

The legitimate owner typically unlocks the car from the **left side**, at **7:30 AM** weekdays. An attacker captures the code and replays it at **2:00 AM** from the **back side**.

This deviation in signal angle + time + frequency is flagged as **Anomaly behavior**.

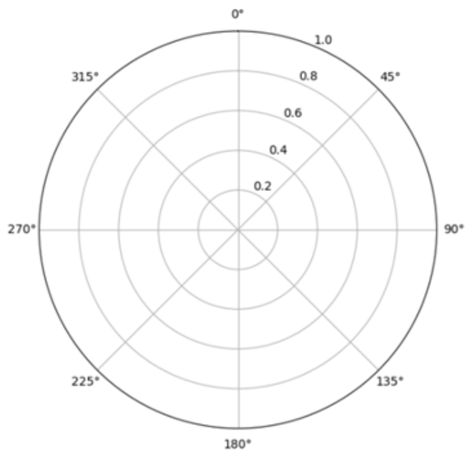


Fig 12.a Baseline (no signal received).

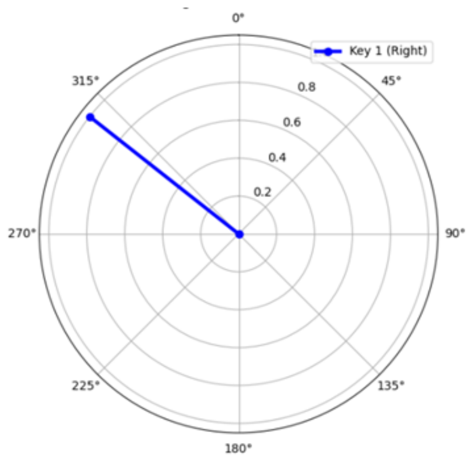


Fig 12.b Signal from right side (Key 1)

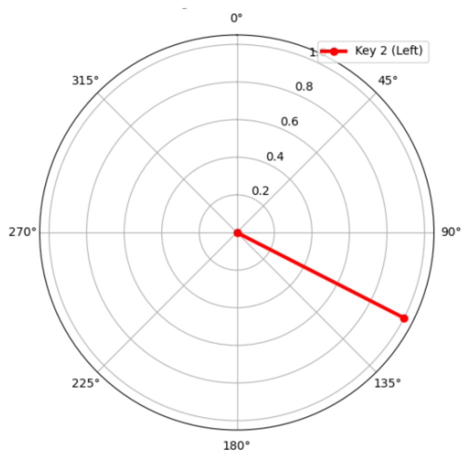


Fig 12.c Signal from left side (Key 2)

Fig 12 Signal direction detection using polar plots

4.4 Secure Data Storage via Google Drive API

All sensitive information captured by the IoT-GEN—RF keys, attack logs, jamming events, and anomaly traces—is stored securely using Google Drive API integration.

Key Features

- End-to-end encrypted transfer
- Organized storage by date and category
- Offline buffering if no connection is available

Stored data includes:

- Captured .bin RF key patterns
- Wi-Fi .cap handshake files
- Jamming detection logs
- Anomaly behavior records (timestamp, angle, trigger type)

This feature ensures that if the IoT-GEN is stolen or compromised, no one can retrieve the stored signals or logs except the verified owner.

CHAPTER 5: System Implementation and Testing

5.1 Testing Strategy and Environment

After completing the hardware and software integration of the IoT Security Guard Edge Node, we proceeded with extensive field testing to evaluate the performance of each attack and defense module. Testing was conducted in both indoor and outdoor environments, across multiple car models, signal ranges, and interference conditions.

All testing was executed on a IoT-GEN mentioned above after completely designed.

5.2 RF Attack Modules Testing

Test Objective: Capture a valid RF key signal and replay it to the target vehicle.

Test Result: Successful across multiple vehicles using fixed-code key fobs operating at 433 MHz.

Distance: Signals were captured from remote and successfully replayed to the target vehicles from up to 50 meters away in open environments.

Target Cars: Most vehicles manufactured before 2015 were vulnerable to cloning and replay attacks.

5.3 Rolling Code Emulation and Prediction

Test Challenge: Due to limited access to modern rolling-code key fobs, real-world tests were restricted.

Solution: We developed a custom Python-based emulation to generate rolling patterns based on realistic hopping algorithms[22] (hopping1 and hopping2). These algorithms simulate the dynamic changes in transmitted codes used in rolling code systems, where each transmission uses a new pseudo-random value derived from a synchronized counter shared between transmitter and receiver.

These patterns were then fed into an LSTM prediction model to: Identify the pattern sequence logic, Predict the next possible key Simulate the code progression of actual rolling-key systems

Outcome: The system successfully identified and predicted next key patterns in simulation, validating the approach for future real-world adaptation.

5.4 RF Jamming Module

Test Objective: Transmit interference pulses to block legitimate remote key communications.

Results by Distance:

5 meters: Legitimate signal fully blocked.
10 meters: Full jamming maintained.
25 meters: Partial blocking with minor leakage.
50 meters (open field): Reduced effect; some signals may pass.

Observation: Interference effectiveness decreased significantly in dense environments, indicating susceptibility to RF noise competition.
Additionally, the validity of this observation is supported by findings in previous studies [23], which confirm that interference effectiveness tends to decrease in dense environments due to increased RF signal competition.

5.5 Jamming Detection Results

1. Successfully detected and logged jamming activity at all tested distances.
2. Visual alerts were triggered when signal noise exceeded 55% threshold.
3. Logs saved to file and mirrored successfully to Google Drive via secure API integration.

5.6 Anomaly-Based Behavior Detection

Setup: Multiple receivers placed to emulate signal triangulation which is a method used to determine the location of a signal source by measuring its direction or distance from three or more known points (e.g., antennas or sensors).

Behavior Modeling: IoT-GEN recorded:

Time of unlock, Signal direction and Distance estimation (based on signal strength and arrival time)

Test Result: The system was able to differentiate between:

Legitimate owners with routine unlock patterns and abnormal unlock attempts Like wrong angle and odd hour.

This feature adds a behavioral intelligence layer to RF access control systems, providing better security even when encryption is not available.

5.7 Summary

The implementation and testing phase validated the system's ability to simulate realistic attacks, detect threats in real-time, and log sensitive data securely. Despite hardware limitations, the integration of signal analysis, behavior-based detection, and secure cloud storage proved the IoT-GEN value in both research and field applications.

CHAPTER 6: Conclusion and Results

6.1 Conclusion

This project aimed to develop a compact, affordable, and multifunctional IoT-GEN capable of emulating and defending against a variety of wireless attacks. By focusing on emulation rather than theoretical simulation, the system provided an authentic, hands-on security experience.

Built on the Raspberry Pi Zero 2W, the IoT-GEN combined signal capturing, RF attack modeling, and intelligent defense mechanisms into a single portable platform. Through careful engineering, practical testing, and smart design, the project succeeded in bridging the gap between cybersecurity research and real-world application.

One of the standout innovations of this work was the integration of anomaly-based behavioral detection, introducing a layer of intelligence that does not exist in most traditional RF systems. Furthermore, the use of secure Google Drive logging ensured that captured data remained protected, even in the presence of physical compromise.

Despite some limitations—such as the absence of real rolling-code keys for full testing—the simulated pattern generation and LSTM prediction model proved the theoretical viability of future code prediction systems.

The implementation and testing phase validated the system's ability to simulate realistic attacks, detect threats in real-time, and log sensitive data securely. Despite hardware limitations, the integration of signal analysis, behavior-based detection, and secure cloud storage proved the IoT-GEN value in both research and field applications table 3 summarizes the results of our e of re project.

Table 3. Results Summary

Module	Status	Details
Fixed Code Cloning	Successful	Fully captured and replayed signals from multiple pre-2015 cars
Rolling Code Simulation	Simulated	Patterns generated via hopping1/hopping2 and analyzed using LSTM
RF Jamming Attack	Effective	Stable jamming up to 25 meters, up to 50 meters in clear environments
Jamming Detection	Accurate	Triggered at 55%+ noise threshold with log and screen alert
Anomaly Detection	Functional	Differentiated real owner vs attacker based on direction, time, behavior
Secure Data Logging	Encrypted & Synced	Uploaded all files securely to Google Drive.
Wi-Fi Handshake Attack	Working	Captured WPA2 handshakes and saved .cap files from multiple networks
User Interface	Responsive	Full TFT + joystick navigation, organized by Attack/Security/Wi-Fi sections

6.2 Future Enhancements

While the current implementation of the IoT-GEN has successfully demonstrated practical defense and attack emulations on 433 MHz signals, several future enhancements can significantly expand its capabilities, resilience, and real-world applicability:

- **Multi-Frequency RF Support (315/433/868/915 MHz):**

  To ensure broader compatibility with global automotive systems and IoT devices, future versions of the device will include support for additional frequency bands. This will allow the device to interact with and defend against threats targeting a wider range of vehicles and smart home technologies, rather than being limited to the 433 MHz band.

- **Intrusion Prevention System (IPS) Integration:**

  A major future enhancement is the development of an embedded lightweight Intrusion Prevention System (IPS). This system would continuously monitor Wi-Fi activity, automatically block suspicious patterns or known attack behaviors, and isolate compromised modules. The IPS logic could be based on a rule engine and enhanced by behavior modeling through machine learning.

- **Access Point (Hotspot) Mode for Internet Connectivity:**

  To enhance usability and real-time remote monitoring, the device will be upgraded to function as a wireless access point. This mode would allow users to connect to the IoT-GEN using their smartphones or laptops directly, access logs, trigger tests, and visualize signal activity through a web-based dashboard—without needing external infrastructure.

Through these enhancements, the IoT Security Guard Edge Node can evolve from a focused testing tool into a full-fledged, adaptive wireless intrusion prevention system capable of safeguarding diverse smart environments and automotive systems around the world.

References

1- Pahlavan, K., Krishnamurthy, P. Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective. *Int J Wireless Inf Networks* 28, 3–19 (2021). https://doi.org/10.1007/s10776-020-00501-8.

2- Raspberry Pi Foundation. (2021). Raspberry Pi Zero 2 W – Product Brief. Retrieved from raspberry-pi-zero-2-w-product-brief.pdf

3- Csikor, L., Lim, H. W., Wong, J. W., Ramesh, S., Parameswarath, R. P., & Chan, M. C. (2022). RollBack: *A New Time-Agnostic Replay Attack Against the Automotive Remote Keyless Entry Systems*. arXiv preprint arXiv:2210.11923. https://arxiv.org/abs/2210.11923.

4- Mwangi, J., Cheruiyot, W., & Kimwel, M. (2015). *Security analysis of WPA2. Control Theory and Informatics*, 5(5), 1–6. Jomo Kenyatta University of Agriculture and Technology. http://www.iiste.org.

5- Shahadat, M. M. Z., Ali, M., & Mallik, A. (2023). *An approach on cracking WPA/WPA2 security of Wi-Fi with handshake attack*. Proceedings of the 17th Annual Paper Meet of Mechanical Engineering (APMME) 2023, Dhaka, Bangladesh.

6- Yemets, K., Izonin, I., & Dronyuk, I. (2025). Enhancing the FFT-LSTM Time-Series Forecasting Model via a Novel FFT-Based Feature Extraction–Extension Scheme. *Big Data and Cognitive Computing*, 9(2), 35. https://doi.org/10.3390/bdcc9020035

7- RS Components. (n.d.). 433 MHz RF Transmitter Module – Datasheet. Retrieved from https://docs.rs-online.com/40f4/A700000007964300.pdf

8- Components101. (n.d.). Joystick Module – Pinout and Details. Retrieved from https://components101.com/modules/joystick-module

9- Laskakit. (n.d.). STX882 RF Transmitter Module – Datasheet. Retrieved from https://www.laskakit.cz/user/related_files/stx882.pdf

10- Smart Prototyping. (n.d.). SRX882 RF Receiver Module – Datasheet. Retrieved from https://www.smart-prototyping.com/image/data/9_Modules/101416%20SRX882%20Low%20Consumption%20Strong%20Driving%20Force%20Super%20Heterodyne%20Receiver%20Module/SRX882%20Datasheet.pdf

11- Tarun Raj Kumar, P., Ramakrishna, G., & Nirmala Devi, E. (2025). Mechanical characterization of PETG-3D printed material for enhancement and scrutiny. *YMER*, 24(2), 301–310.

12- NXP Semiconductors. (n.d.). PCF8591: 8-bit A/D and D/A converter. Retrieved from https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf

13- Thompson, M., & Green, P. (2012). Raspbian: A Debian-based operating system for the Raspberry Pi. Raspbian.org. Retrieved from https://www.raspbian.org/

14- Jones, J. (2023). A history of GPIO usage on Raspberry Pi devices, and current best practices. Raspberry Pi Foundation. Retrieved from https://pip.raspberrypi.com/categories/685-whitepapers-app-notes/documents/RP-006553-WP/A-history-of-GPIO-usage-on-Raspberry-Pi-devices-and-current-best-practices.pdf

15- Amarasekara, B. (2021). RF-based home automation using Raspberry Pi (Part 1). Medium. Retrieved from https://binesh-amarasekara.medium.com/rf-based-home-automation-using-raspberry-pi-part-1-b2226c7068da

16- Hull, R. M. (2021). luma.lcd: Python module to drive LCD displays. GitHub. Retrieved from https://github.com/rm-hull/luma.lcd

17- Real Python. (2020). Fourier Transforms With scipy.fft: Python Signal Processing. Retrieved from https://realpython.com/python-scipy-fft/

18- Adafruit Industries. (2021). ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier. Retrieved from https://www.adafruit.com/product/1085

19- Python Software Foundation. (2021). Python 3.9.7 Documentation. Retrieved from https://docs.python.org/3/library/index.html

20- Oluwasegun, K. (2022). Running scripts on boot in Linux using systemd. Medium. Retrieved from https://zt4ff.medium.com/running-scripts-on-boot-in-linux-using-systemd-e10d3606f28f

21- Vanhoef, M., & Piessens, F. (2017). *Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2*. In Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS '17), 1313–1328. https://doi.org/10.1145/3133956.3134027

22- Ghanem, A. (2022). Security Analysis of Rolling Code-based Remote Keyless Entry Systems (Master's thesis, University of Victoria). Retrieved from https://dspace.library.uvic.ca/handle/1828/13914

23- Guo, J., Tang, B., Xu, J., Li, Q., Qin, Y., & Li, X. (2025). *A Secure Communication Protocol for Remote Keyless Entry System with Adaptive Adjustment of Transmission Parameters.* arXiv preprint arXiv:2504.09527. Retrieved from https://arxiv.org/abs/2504.09527