
System Requirements Specification

for

Bluetooth Protocol Analytical Research

Version 2.1

**Gianna Scarangelli
Matthew Irvin
Carolina Terre
Jonah Rowell
Connal Grace**

Embry Riddle Aeronautical University Daytona Beach

3/3/2024

Table of Contents

1. Introduction	4
1.1 Purpose	4
1.2 Document Conventions	4
1.3 Intended Audience and Reading Suggestions	4
1.4 Product Scope	5
1.5 References	5
2. Overall Description	5
2.1 Product Perspective	5
2.2 Product Functions	6
2.3 User Classes and Characteristics	7
2.4 Operating Environment	7
2.5 Design and Implementation Constraints	7
2.6 User Documentation	7
2.7 Assumptions and Dependencies	8
3. External Interface Requirements	8
3.1 User Interfaces	8
3.2 Hardware Interfaces	12
3.3 Software Interfaces	12
3.4 Communications Interfaces	13
4. System Features	13
4.1 Test Plan	13
4.2 Bluetooth Connectivity	14
4.3 Zigbee Connectivity	14
5. Other Nonfunctional Requirements	15
5.1 Performance Requirements	15
5.2 Safety Requirements	15
5.3 Security Requirements	15
5.4 Software Quality Attributes	16
5.5 Business Rules	16
6. Other Requirements	16

Revision History

Name	Date	Reason For Changes	Version
Initial Release	9/29/2023	Initial document draft.	1.0
First Semester First Revision	10/31/2023	Updates & edits from feedback.	1.1
First Semester Second Revision	11/21/2023	Updates & edits from feedback.	1.2
Flipper Zero Revision	2/05/2024	Changes for new semester 2 work and product end goal.	2.0
Second Semester Revision	3/03/2024	Updates & edits from feedback. Edited sections 1.1, 1.3, 2.1-2, 2.6, 3.1-4, 4.1-3 for consistency and clarity.	2.1

1. Introduction

This product is designed to transmit and test various wireless protocols, including Bluetooth, Bluetooth Low Energy, and Zigbee. The goal of this product is to create several testbeds where these wireless protocols can safely be transmitted, data collected, and attempts to perform various attacks can be performed. From doing this, these protocols are ideally exploited with aviation use cases in mind. These use cases include a Bluetooth audio system that may be presented to passengers, Bluetooth Low Energy tracking devices like airtags, among other possible use cases.

1.1 Purpose

This document should define all the requirements and conditions revolving around the protocols that Bluetooth, Bluetooth Low Energy (BLE), and Zigbee follow. It will additionally follow the physical connections between the server/client as well as the user interface in which to view the findings.

This document is the most up to date as of Sunday, 03-03-2024, revision 2.1. It is the second iteration of this document, as approved by the Boeing Bluetooth Team.

1.2 Document Conventions

Currently no standard conventions are being used, which is scheduled to change as further research about the topic is uncovered. Each requirement will have a specific priority in the long term of this project, as sprints continue this is also set to change based on the progress or a change in priority.

All text in this document, aside from headers, use the same font and text size. However, note that any underlined words indicate a definition.

1.3 Intended Audience and Reading Suggestions

The intended users for this document would be cybersecurity researchers, developers using these wireless configurations, as well as the everyday person who has Internet of Things (IoT) devices and utilizes them. This document will contain different sections: Section 2 will discuss the functions, prospected results of the research, and it most relevant to researchers; Section 3 focuses on how one would be able to use and read information from the product, and is most relevant to developers; Section 4 is the primary priorities attempted to achieve by the end of this semester; and Section 5 will be the standards that are to be followed as this system is developed. Section 1 is most relevant to researchers, while Sections 3 and 5 are most relevant to developers. Everyday people may be most interested in Sections 3 and 4.

1.4 Product Scope

With wireless IoT devices becoming ever more prevalent throughout the world, the goal is to search for vulnerabilities between clients and servers as they connect using Bluetooth, BLE or Zigbee and create a system able to aptly display these findings. The software will focus on the breach of the vulnerabilities, attempting to extract or disrupt information, and prioritizing integrity, availability, then confidentiality. Additionally, the information would be displayed in an orderly fashion to the user. Hardware will be used to simulate server/clients to further analyze possible physical constraints that may impact the security.

1.5 References

1.5.1 Common Weakness Enumeration (CWE), from <https://cwe.mitre.org/>

1.5.2 National Vulnerability Database, from <https://nvd.nist.gov/>

2. Overall Description

Section 2 covers general product design in 2.1, the product functions in 2.2, user information in 2.3, and the operating environment in 2.4. Additionally, constraints are covered in 2.5, external documentation links are in 2.6, and assumptions are contained in 2.7.

2.1 Product Perspective

Our newly created product is self-contained to analyze the safety of Bluetooth and the signals between a server and a client. A detailed diagram on the Raspberry Pi system can be seen in Figure 1 below. It consists of a power source, display, keyboard, and SD card all connected to the centralized Raspberry Pi 4.

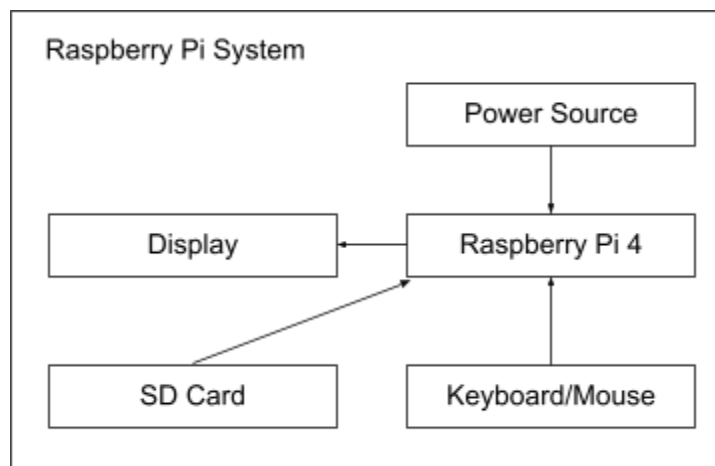


Figure 1: Diagram of the Raspberry Pi System

Diagrams of all additional systems, including Zigbee, jammer, spectrum analyzer, and Bluetooth,

can be seen in Figure 2 below. The Zigbee system consists of the Raspberry Pi system connected to two Zigbee development boards in a circular communication form. Both the jammer and spectrum analyzer systems consist of its over Raspberry Pi system connected to a HackRF One. The Bluetooth system consists of another Raspberry Pi system connected to two ESP 32's in a circular communication form.

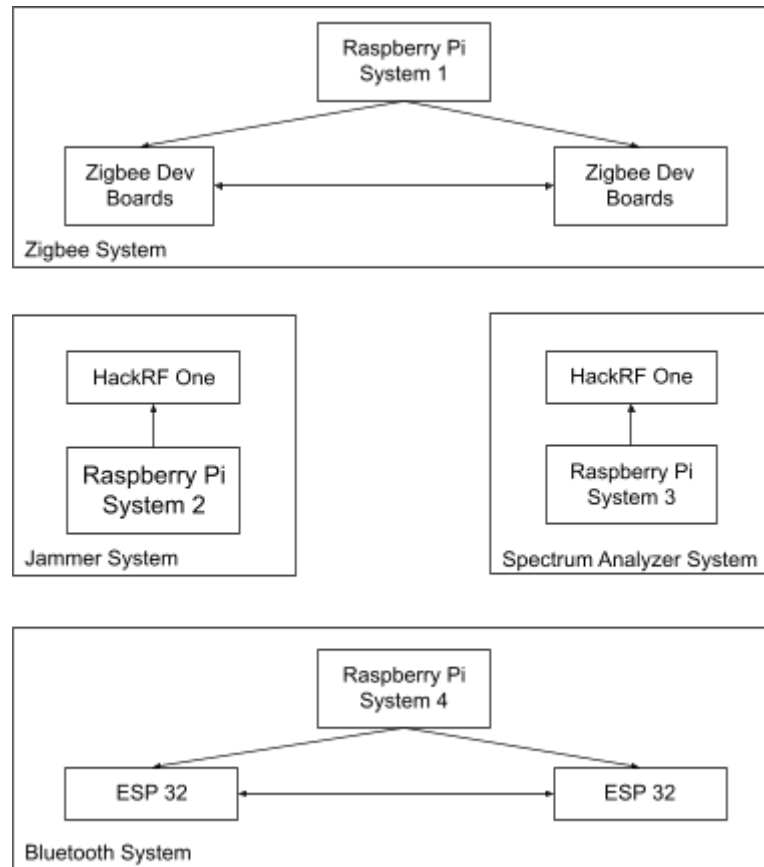


Figure 2: Diagrams of the Bluetooth, Zigbee, and Software Designed Radio Systems

2.2 Product Functions

- 2.2.1 The product should receive Bluetooth signals.
- 2.2.2 The signals should be identified and collected.
- 2.2.3 The HackRF One should intercept Bluetooth signals not intended for the device.
- 2.2.4 Using penetration methods to extract information.
- 2.2.5 The product should store and process information.
- 2.2.6 GNU Radio should display a graphical view of the transmitted frequency.

2.2.7 HackRF Spectrum Analyzer will monitor the frequency between 2.4 GHz and 2.5 GHz.

2.2.8 The product should transmit BLE signals.

2.3 User Classes and Characteristics

Cybersecurity researchers who have previous knowledge of wireless communications would be responsible for the review of the results. With the information, they would be able to perform an in-depth analysis of the findings to further improve the security of the signals. Presentation of the information in an easy-to-understand form to the general person is also a necessity.

2.4 Operating Environment

The software will operate running Raspberry Pis that display the information on a mini monitor through an HDMI connection. Using Python, a user will interact with the Raspberry Pis which must be able to connect to Kali Linux 2023.3 to collect data as well as to interact with it. Bluetooth 5.0 will create and send signals between the multiple Pis.

2.5 Design and Implementation Constraints

- Systems must be able to interact with Bluetooth devices within 5 feet
- System must be able to operate at the start of a Bluetooth connection
- System must be able to interact with Zigbee devices within 5 feet
- System must be able to operate at the start of a Zigbee connection
- HackRF must be able to operate on same frequency as Bluetooth and Zigbee
- System must be able to interact like BLE devices within 5 feet
- System must be able to operate at the start of a BLE connection

2.6 User Documentation

The following documents are all relevant to product development and testing. User manuals and other device documentation are used to properly set up the test bed devices.

2.6.1 Zigbee Development Board User Manual:
<https://www.ti.com/lit/ug/swru209b/swru209b.pdf>

2.6.2 ESP32 User Manual:
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

2.6.3 HackRF One documentation:
<https://hackrf.readthedocs.io/en/latest/>

2.6.4 Raspberry Pi Documentation:

<https://www.raspberrypi.com/documentation/>

2.6.5 Kali Linux Documentation:

<https://www.kali.org/docs/>

2.6.6 HackRF Spectrum Analyzer GitHub:

<https://github.com/pavsa/hackrf-spectrum-analyzer>

2.6.7 Flipper Zero Documentation:

<https://docs.flipper.net/>

2.6.8 Flipper Zero External Firmware (Flipper Xtreme):

<https://github.com/Flipper-XFW/Xtreme-Firmware>

2.7 Assumptions and Dependencies

2.7.1 There are no external RF signals on the same or similar frequencies.

2.7.2 Transmitter and receiver are within 5 feet of each other.

2.7.3 There are no objects or walls between transmitter and receiver.

2.7.4 The user has downloaded and installed all required software.


3. External Interface Requirements

Section 3 covers information regarding all system interfaces. Section 3.1 reviews user software interfaces and 3.3 outlines further software interfaces. Section 3.2 outlines hardware interfaces while 3.4 outlines further communication hardware interfaces.

3.1 User Interfaces

The following requirements describe how respective user interfaces for each product component are displayed and utilized.

3.1.1 Arduino IDE: Arduino IDE shall be used to push code to the ESP32. An example of this IDE with a sample program can be seen in Figure 3 below.

A screenshot of the Arduino IDE interface. The title bar reads 'BlinkingLight | Arduino 1.8.13'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for opening files, saving, compiling, uploading, and erasing. The main text area shows a C++ sketch for a BLE server. The code includes headers for BLEDevice, BLEUtils, and BLEServer. It defines a SERVICE_UUID and a CHARACTERISTIC_UUID. The setup function initializes the serial port and prints a message. The main loop initializes the BLE device, creates a server, a service, and a characteristic, and then starts the service and advertising. The status bar at the bottom indicates 'DOT: ESP32 DEVKIT V1, 80MHz, 52160B, None, Disabled on COM8'.

```
// Code was edited from https://tutorials-raspberrypi.com/esp32-bluetooth-connection-to-esp8266-and-raspberry-pi/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

#define SERVICE_UUID          "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID   "beb5483e-36e1-4688-b7f5-ea07361b26a8"

BLEServer *pServer;
BLECharacteristic *pCharacteristic;
int messageCounter = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE work!");

  BLEDevice::init("ESP32 AS A BLE");
  pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ |
    BLECharacteristic::PROPERTY_WRITE
  );

  pService->start();
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
```

Figure 3: Arduino IDE

3.1.2 Zigbee Software: Zigbee Development software shall be used to control the Zigbee Boards to send a signal.

3.1.3 HackRF Tools: HackRF Tools shall be used to control the HackRF One.

3.1.4 GNU Radio: GNU Radio shall be used to send RF signals via the HackRF One. Users can configure what frequency and range the signals are being sent as well as view Bluetooth packets and signals. See Figure 4 below.

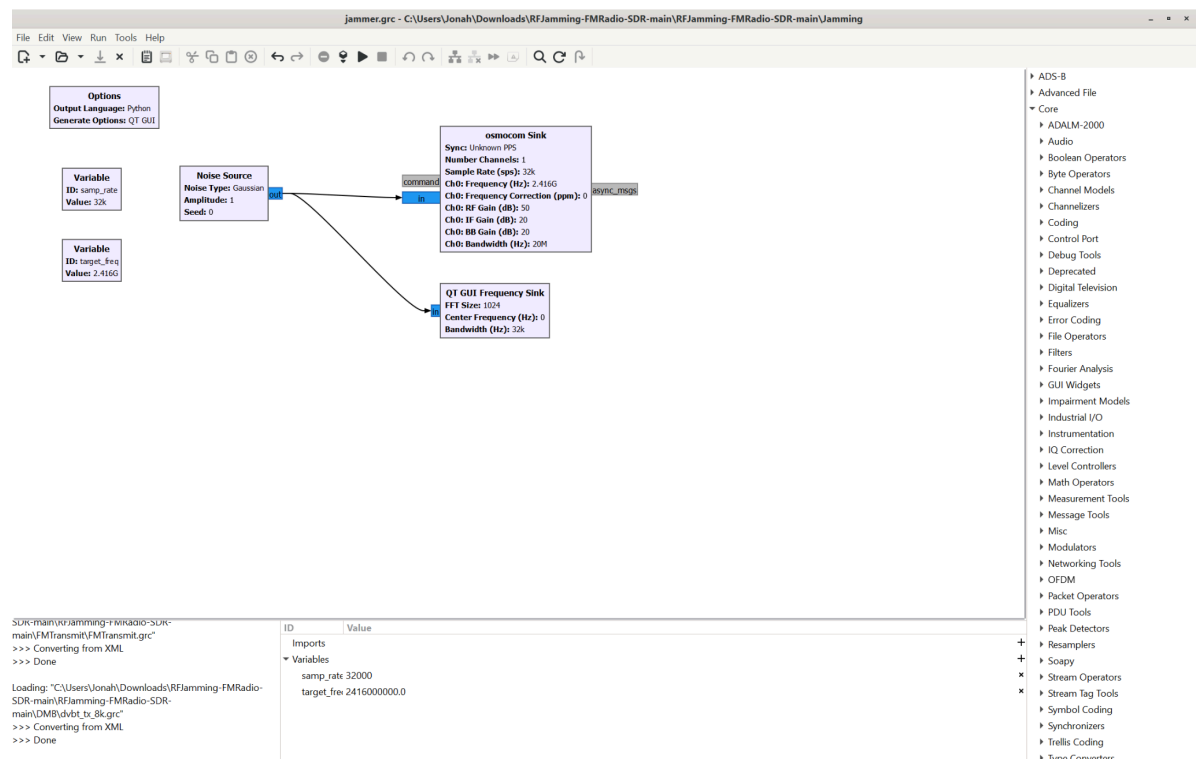


Figure 4: GNU Radio Jammer File

3.1.5 HackRF Spectrum Analyzer: The spectrum analyzer shall be used to determine if the HackRF One is correctly transmitting on the right frequency as well as view the data transmitted by Bluetooth and Zigbee devices. A sample output of this software interface can be seen in Figure 5 below.

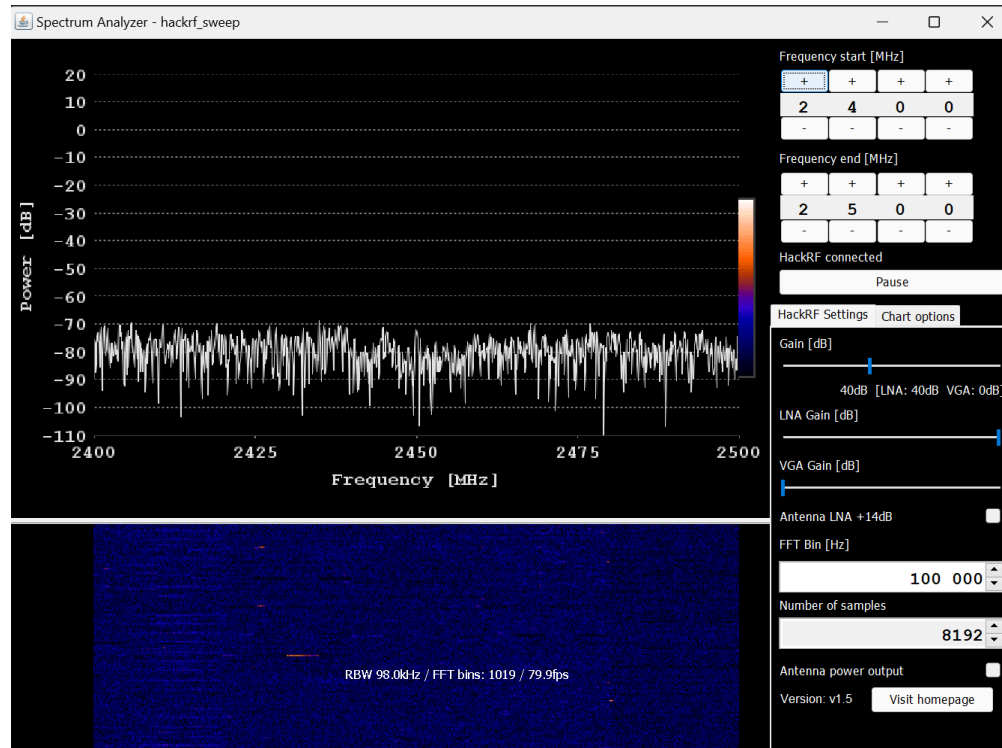


Figure 5: GNU HackRF Spectrum Analyzer

3.1.6 **qFlipper Desktop Application:** New firmware can be edited and downloaded onto Flipper Zero when the Flipper Zero is USB connected to the given laptop. Figure 6 below displays where Flipper Zero files may be edited in the software.

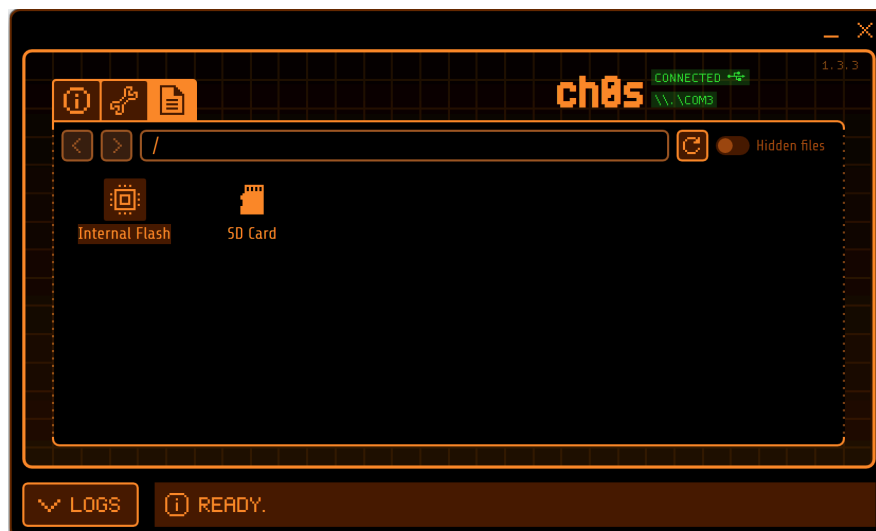


Figure 6: qFlipper Application

3.2 Hardware Interfaces

The Raspberry Pi system will contain all of the components needed to operate the Raspberry Pi as a standalone computer - consisting of the power source, display, keyboard, and mouse. An SD card will be used to run the applications and operating system of the Raspberry Pi. Four Raspberry Pis will be used to run the ESP32 and Zigbee modules, as well as one more Raspberry Pi for the HackRF Module.

3.2.1 Required Hardware:

The required hardware shall consist of the following:

3.2.1.1 Two Software Defined Radio (HackRF One)

3.2.1.2 Two ESP32 Bluetooth Modules

3.2.1.3 Two Zigbee Development Modules

3.2.1.4 Four Raspberry Pis

3.2.1.5 Four SD Cards

3.2.1.6 Three Flipper Zero's

3.2.2 Preloaded operating system and software shall be loaded on the SD cards for use on Raspberry Pis.

3.3 Software Interfaces

Each Raspberry Pi will be preloaded with Kali Linux on the SD card, along with the software to run the required module. Two Raspberry Pis will have Arduino IDE to run the ESP32s, two will have the Zigbee Development Software, and one will have the HackRF Tools and GNU Radio.

3.3.1 Four Raspberry Pis shall be loaded with Kali Linux (or other software) to run their respective hardware.

3.3.2 HackRF Tools shall be used to establish a connection with the HackRF One.

3.3.3 GNU Radio shall be used to generate and send RF signals using the HackRF One.

3.3.4 Data for the ESP32 and Zigbee Boards shall be sent via serial from the Raspberry Pis.

3.3.5 HackRF Spectrum analyzer shall be used to measure the data transmitted by the jammer using a HackRf One.

3.3.6 New firmware shall be installed onto Flipper Zero via qFlipper.

3.4 Communications Interfaces

Each Raspberry Pi will function using an operating system and software through an SD card. The Raspberry Pis will connect to either the ESP32 or Zigbee Development boards through USB type A to USB micro B. The HackRF will connect to the Raspberry Pi via USB A to USB type C. The two ESP32 modules will communicate via Bluetooth, and the Zigbee Development boards will communicate via Zigbee. The HackRF will send signals through a set frequency to intercept or block Bluetooth and Zigbee signals.

- 3.4.1 The HackRF, Zigbee, and ESP32 Bluetooth modules shall connect to the Raspberry Pis via USB type A to USB micro B.
- 3.4.2 The ESP32 modules shall communicate via Bluetooth.
- 3.4.3 The Zigbee modules shall communicate via Zigbee.
- 3.4.4 Peripherals such as the keyboard and mouse shall connect via USB type A.
- 3.4.5 The displays shall connect via micro HDMI to HDMI.
- 3.4.6 The Raspberry Pis shall be powered via a USB type C.
- 3.4.7 The Flipper Zeros shall communicate via BLE.

4. System Features

Section 4 provides further details into system features as introduced in Section 2.2. Section 4.1 presents a general system test plan, while 4.2 and 4.3 go into further descriptive detail of Bluetooth and Zigbee connectivity, respectively.

4.1 Test Plan

4.1.1 Description and Priority

The test plan provides the team with a plan of action to find and experiment with vulnerabilities and security flaws in Bluetooth and Zigbee. A higher priority is placed on Bluetooth security.

4.1.2 Stimulus/Response Sequences

There are no user actions or system responses for this feature.

4.1.3 Functional Requirements

- 4.1.3.1 The test plan shall provide team members with a plan to test vulnerabilities in the Bluetooth protocol.
- 4.1.3.2 The test plan shall provide team members with a plan to test vulnerabilities in the Zigbee protocol.
- 4.1.3.3 The test plan shall provide team members with a plan to test vulnerabilities in the BLE protocol.

4.2 Bluetooth Connectivity

4.2.1 Description and Priority

Raspberry Pi computers connected to ESP32 development boards will be connected using Bluetooth. Implementing this feature is a high priority because it is integral to the identity of the project.

4.2.2 Stimulus/Response Sequences

When a Raspberry Pi and ESP32 development board combination is already connected to the network, powered on, and in range of the network, it will connect to the network. When a Raspberry Pi and ESP32 development board combination that is not already connected to the network, and is in range of the network, is powered on, it will be able to be added to the network using Bluetooth pairing.

4.2.3 Functional Requirements

4.2.3.1 A Raspberry Pi shall be able to transmit and receive data with another Raspberry Pi using Bluetooth.

4.2.3.2 The ESP32 modules shall transmit and receive data with each other using Bluetooth.

4.2.3.3 The Raspberry Pi operating system shall be loaded from an SD card.

4.2.3.4 The ESP32 modules shall be preloaded with the appropriate code to create a Bluetooth connection.

4.2.3.5 The Zigbee modules shall be able to transmit and receive data with each other using Zigbee.

4.2.3.6 The HackRF shall be able to detect Bluetooth packets on the 2.4 GHz band.

4.3 Zigbee Connectivity

4.3.1 Description and Priority:

Raspberry Pi computers connected to Zigbee development boards will be connected using Zigbee. Implementing this feature is a medium priority because it is considered a lower priority than Bluetooth connectivity.

4.3.2 Stimulus/Response Sequences

When a Raspberry Pi and Zigbee development board that is already connected to the network and in the range is powered on, it will connect to the Zigbee network. When a Raspberry Pi and Zigbee development board that is not already connected to the network and in the range is powered on, it will be able to connect to the existing Zigbee network.

4.3.3 Functional Requirements

4.3.3.1 A Raspberry Pi shall transmit and receive data with another Raspberry Pi using Zigbee.

5. Other Nonfunctional Requirements

Section 5 outlines all known nonfunctional requirements associated with this product. Section 5.1 covers requirements related to performance, 5.2 requirements related to safety, 5.3 requirements related to security, 5.4 requirements related to software quality, and finally 5.5 requirements related to business rules.

5.1 Performance Requirements

- 5.1.1 Bluetooth and Zigbee connection shall attempt to re-establish after the connection is lost.
- 5.1.2 Bluetooth transmission shall be either partially or fully stopped.
- 5.1.2 Bluetooth service shall be interrupted for a minimum of 1 second.
- 5.1.4 Zigbee service shall be interrupted for a minimum of 1 second.
- 5.1.5 Bluetooth data rate shall be a minimum of 1 MBps.
- 5.1.6 Bluetooth data rate shall be a maximum of 2 MBps.
- 5.1.8 Zigbee data rate shall be a minimum of 20 kbps.
- 5.1.9 Zigbee data rate shall be a maximum of 250 kbps.
- 5.1.10 RF signals shall be transmitted at the same frequency as Bluetooth.
- 5.1.11 RF signals shall be transmitted at the same frequency as Zigbee.
- 5.1.12 Raspberry Pi shall contain 128 GB of storage.
- 5.1.13 Service interrupt shall occur within 1 second of execution.
- 5.1.14 BLE data rate shall be maximum 1 MPps.

5.2 Safety Requirements

- 5.2.1 All Raspberry Pi shall remain in cases with fans running while powered.
- 5.2.2 All devices shall be powered down before handling.

5.3 Security Requirements

- 5.3.1 The system shall not be used within the range of extraneous electronic devices.
- 5.3.2 The system shall only be used in a controlled environment.
- 5.3.3 All testing shall be done on a closed network.

5.3.4 All transmitted RF signals shall remain within 2.4 GHz and 2.5 GHz.

5.3.5 Signal data shall be deleted after use.

5.3.6 Devices unrelated to the testing shall be removed from the room.

5.3.7 Personal devices shall not be connected to any of the systems.

5.4 Software Quality Attributes

5.4.1 RF signals shall be transmitted between 2.4 GHz and 2.5 GHz.

5.4.2 Bluetooth signals shall be transmitted between 2.4 GHz and 2.5 GHz.

5.4.3 Zigbee signals shall be transmitted between 2.4 GHz and 2.5 GHz.

5.4.4 The Spectrum Analyzer shall monitor a minimum frequency of 2.4 GHz.

5.4.5 The Spectrum Analyzer shall monitor a maximum frequency of 2.5 GHz.

5.4.6 BLE signals shall be transmitted between 2.4 GHz and 2.5 GHz

5.5 Business Rules

5.5.1 All testing shall be preapproved in a controlled environment.

5.5.2 Signals shall transmit a maximum of 20 feet.

6. Other Requirements

No further requirements not previously stated.

Appendix A: Glossary

Bluetooth - Short-Range Wireless Data Transmission Standard

Zigbee - Wireless Personal Area Network Standard

RF - Radio Frequency

IOT - Internet of Things

SDR - Software Defined Radio

USB - Universal Serial Bus

SDR - Software Defined Radio

BLE - Bluetooth Low Energy