System Design Document

For

Bluetooth Protocol Analytical Research

Gianna Scarangelli
Matthew Irvin
Carolina Terre
Jonah Rowell
Connal Grace

Version/Author	Date
v1.0	9/29
v2.0	10/31
v3.0	11/21

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

This document details the architectural and design elements for the project focused on investigating vulnerabilities in Bluetooth 5 and Zigbee protocols within a server-client configuration using Raspberry Pis or similar hardware.

1.2 Project Executive Summary

This section provides an overview of the wireless IoT vulnerabilities investigation project from a management perspective, showcasing the framework within which the system design was conceived.

1.2.1 System Overview

The system aims to explore, identify, and document vulnerabilities inherent to wireless IoT technologies, focusing primarily on the Bluetooth 5 and Zigbee protocols. Through a server-client configuration, orchestrated using Raspberry Pis (or similar hardware), the project delves into the security implications tied to the use of advertising IDs in wireless communication. See Figure 1.

High-Level System Architecture: A simplified high-level system architecture diagram is illustrated below, breaking down the system into discernable subsystems.

Server Subsystem: Hosted on a Raspberry Pi (or similar hardware), this subsystem will house the central server facilitating communication with client devices.

Client Subsystem: Comprising various devices like Raspberry Pis, Zigbee Dev Boards, and ESP32s, this subsystem will mimic IoT devices connecting to the server via Bluetooth 5 or Zigbee protocols.

Analysis Subsystem: Utilizing Software Radios (SDRs) like HackRF, Flipper Zero, and laptops/servers, this subsystem will conduct vulnerability analysis, capturing and analyzing the wireless traffic.

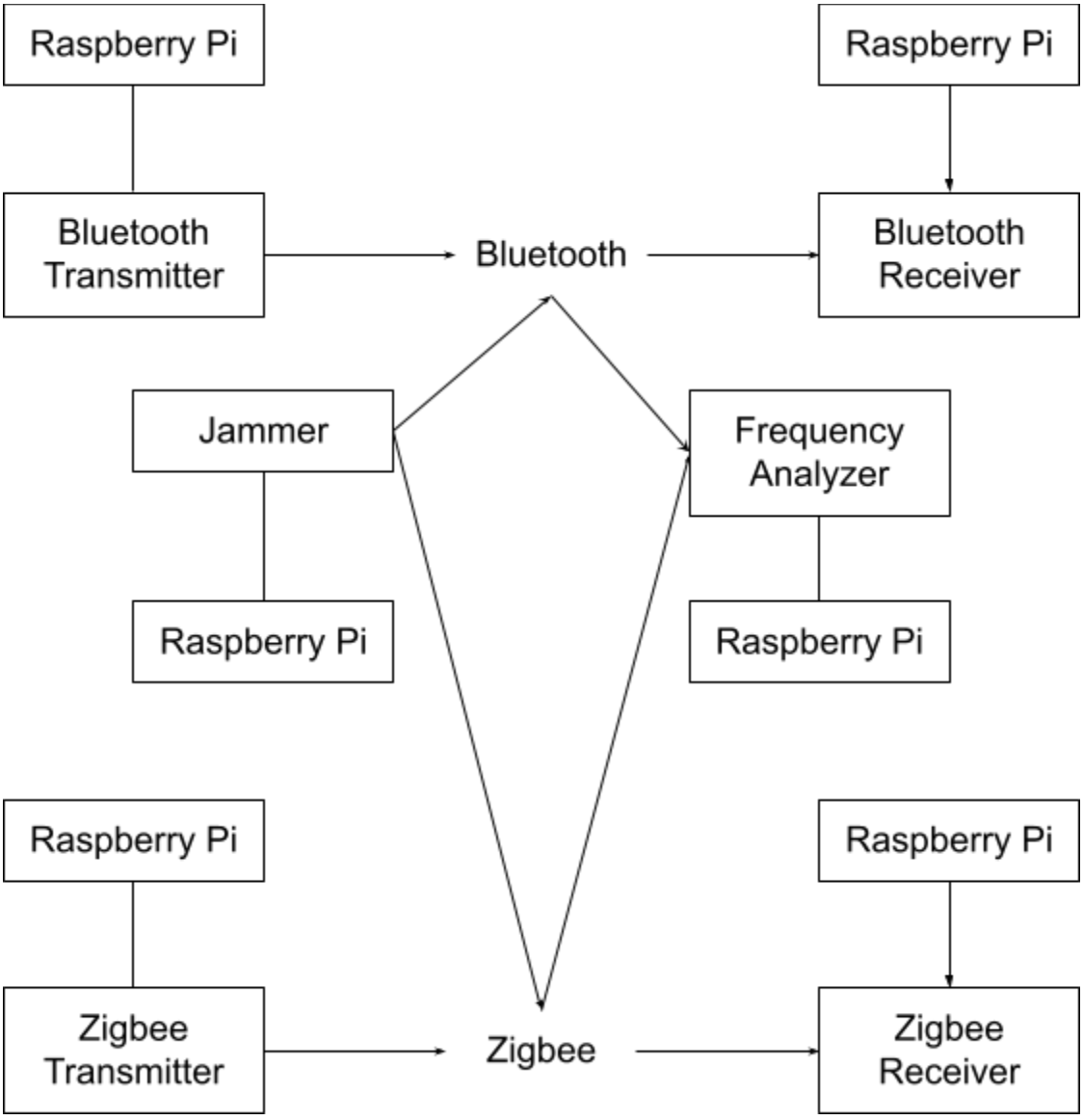

Figure 1: High-Level System Architecture Diagram

1.2.1.1 Reference to Requirements Traceability Matrix (RTM)

A thorough reference to the Requirements Traceability Matrix (RTM) in the Functional Requirements Document (FRD) is provided to map the allocation of functional requirements into this design document, ensuring a systematic transition from requirements to design.

1.2.2 Design Constraints

Investigating vulnerabilities within wireless IoT technologies, notably within Bluetooth 5 and Zigbee protocols, is restrained by several notable constraints: legality, budget, technology, time, and IT department approvals. The budget constraint of $25K necessitates prudent fiscal management for procuring vital equipment such as Software Defined Radios (SDRs), Zigbee Dev Boards, and Raspberry Pis. The technological

boundary is drawn around Bluetooth 5 and Zigbee protocols, with a further obstacle of limited access to existing information on Bluetooth vulnerabilities, requiring a more thorough and potentially creative investigative approach. The time constraint is across a division of two semesters, enforcing a strict timeline for each critical phase: research, planning, and testing/findings. Additionally, the requisite approvals from the IT department add a procedural hurdle, as not all desired resources or methods may receive a green light, possibly leading to adjustments or deviations in the planned approach. Another task is compliance with the standards of testing Bluetooth security legally. To test Bluetooth security, the project must adhere to Bluetooth SIG (Special Interest Group) specifications, while also not interfering with other signals. Each phase hence demands a meticulous orchestration of time, technological resources, budget, and adherence to organizational protocols to ensure a thorough investigation and insightful documentation of findings. The outlined constraints underscore the imperative for a meticulously structured, well-coordinated, and flexible approach to surmount these challenges, ensuring the project's objectives are duly met within the delineated framework.

1.2.3 Future Contingencies

This project, aimed at investigating vulnerabilities in wireless IoT technologies using a server-client configuration with Raspberry Pis and exploring vulnerabilities through Zigbee or Bluetooth 5 advertising IDs, may face several contingencies. First, equipment procurement delays might arise due to budget approval or supply chain disruptions, mitigated by initiating the procurement process early and identifying alternative suppliers. Second, the IT department might not approve some tools or methodologies, potentially hindering progress. Engaging with the IT department from the project's inception and preparing alternative methodologies can mitigate this risk. Third, the opaque nature of information regarding Bluetooth vulnerabilities might lead to undiscovered vulnerabilities; engaging with external experts or reviewing recent publications could provide deeper insights. Fourth, technical limitations with Raspberry Pis or other chosen hardware might inhibit the full exploration of vulnerabilities; having backup hardware options or considering upgrades can alleviate this issue. Fifth, project expenses may exceed the allocated budget due to unforeseen costs, which can be mitigated by meticulous budget tracking, allocating a contingency fund, and prioritizing essential expenditures. Lastly, project phases might extend beyond stipulated timelines due to unforeseen challenges; implementing a well-structured project management approach, utilizing monitoring tools, and allocating time buffers in the schedule can help in navigating through these challenges. This single-paragraph contingency section outlines potential hurdles and mitigation strategies to ensure the project navigates unforeseen challenges efficiently for the successful and timely achievement of its objectives.

1.3 Document Organization

The purpose of this document is to provide a clear overview of the system design for investigating vulnerabilities in wireless IoT technologies utilizing a server-client setup. An introduction and executive summary of the project are presented at the start, and then sections on the high-level system architecture, design limitations, contingencies, and the precise hardware and software design follow. It also covers system interactions, user interfaces, and security issues, particularly as they relate to Bluetooth and Zigbee

technologies. The document also discusses the testing phases, suggested equipment purchases, a breakdown of tasks by semester, and a glossary to clarify technical terms at the end, ensuring a thorough understanding of the project's design framework and the justification for various design decisions.

1.4 Project References

This section provides a bibliography of key project references and deliverables that have been produced before this point.

1.5 Glossary

Flipper Zero: A tool to read, store, and emulate remote signals
Kali Linux: An open-source operating system aimed at Penetration Testing and Security Auditing.
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

2 SYSTEM ARCHITECTURE

This section describes and gives an overview of the systems and subsystems used in this project.

2.1 System Hardware Architecture

This section describes the overall hardware of the system, a list of hardware devices used, and connectivity diagrams.

Zigbee Development Boards(IOTZTB-DK0006): Development boards for testing Zigbee signals and devices
HackRF One: Software-defined radio used for transmitting signals
ESP32 2.4GHz Dual-Core: Microcontroller with onboard Wi-Fi and Bluetooth
Raspberry Pi 4 8GB RAM: Microcomputer used to transmit and receive Bluetooth signals as well as to control various tasks within the system
SD Cards: Small storage devices for Raspberry Pi
SAMSUNG SSD T7 Portable External Solid State Drive 1TB: Large portable storage devices used for file storage on Raspberry Pis.
Peripherals: Displays, mouse, and keyboard to control Raspberry Pis as well as a case to protect the Raspberry Pi. An SD card reader with a USB adapter will be needed to connect the SD card to computers

The Raspberry Pi system (Figure 2) 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.

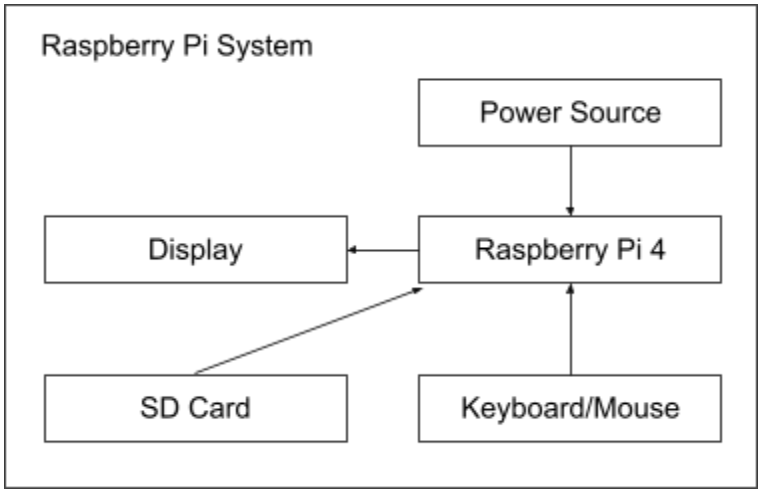

Figure 2: Diagram of the Raspberry Pi System

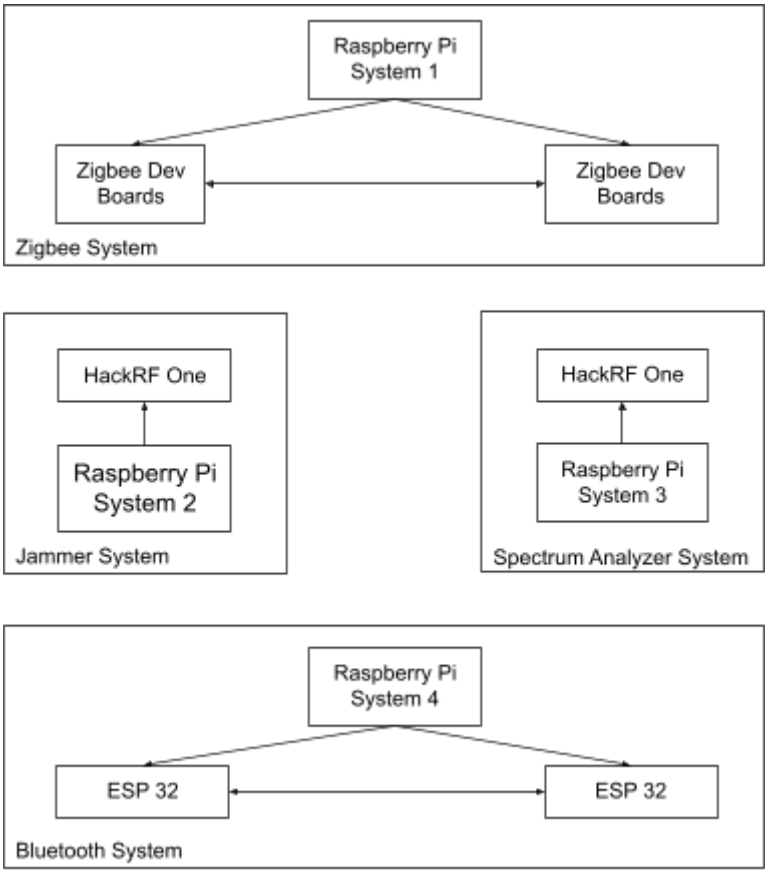

Figure 3: Diagrams of the Bluetooth, Zigbee, and Software Designed Radio Systems

2.2 System Software Architecture

Each Raspberry Pi will be preloaded with Kali Linux on the SD card as well as 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 installed. See Figure 4 below.

Kali Linux 2023.3: Linux-based operating system can be used for pen testing
Raspberry PI OS 2023-10-10: Operating system for Raspberry Pi
GNU Radio 3.10.8.0: Software used to control software-defined radios such as the HackRF One
HackRF Tools 2023.01.1: Command line to interact with HackRF
Arduino IDE 2.2.1: Used to program ESP32
Zigbee Development Kit Software 3.0: Software used to control Zigbee boards

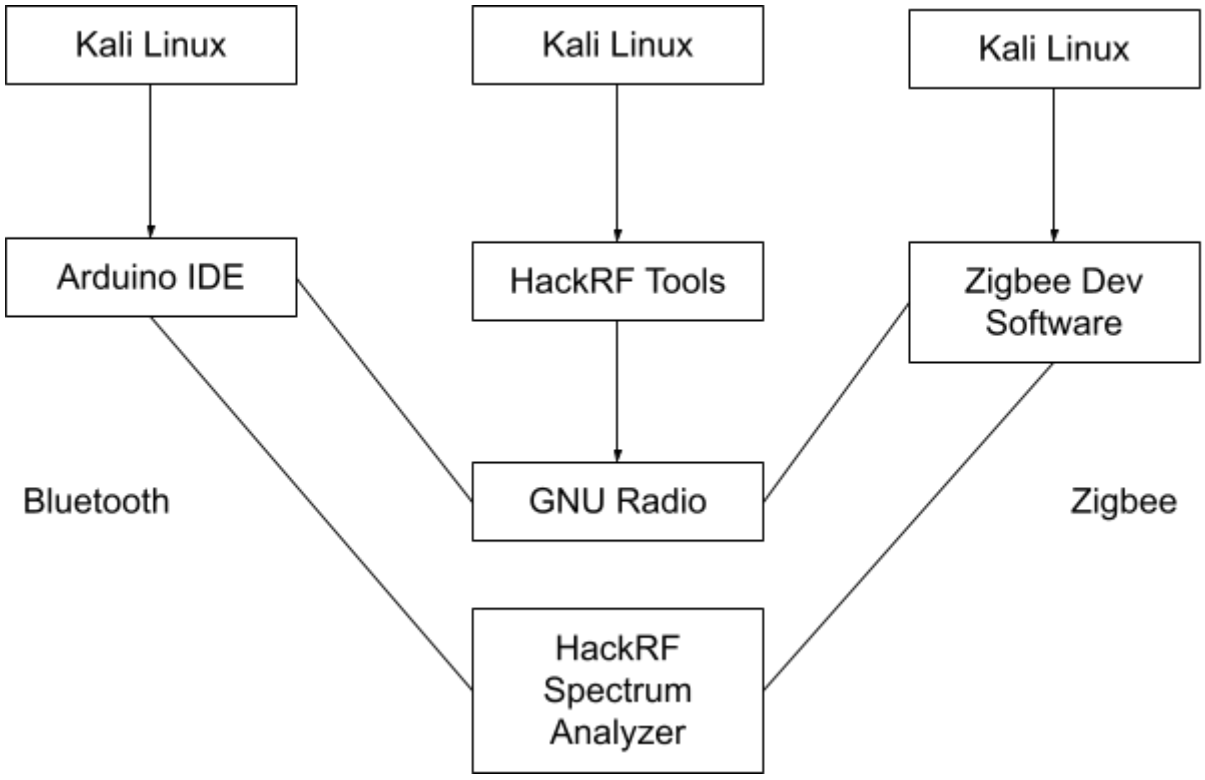

Figure 4: System Software Architecture Diagram

2.3 Internal Communications Architecture

Each Raspberry Pi will operate 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.

Zigbee Development Boards: Mesh networking for Zigbee devices
HackRF One: Radiofrequency communication
ESP32: Wi-Fi and Bluetooth communication
Raspberry Pi: Bluetooth, local network, and internet communication
SD Cards and SSDs: Data storage and retrieval
Peripherals: USB communication
SD Card Reader: USB data transfer to computers

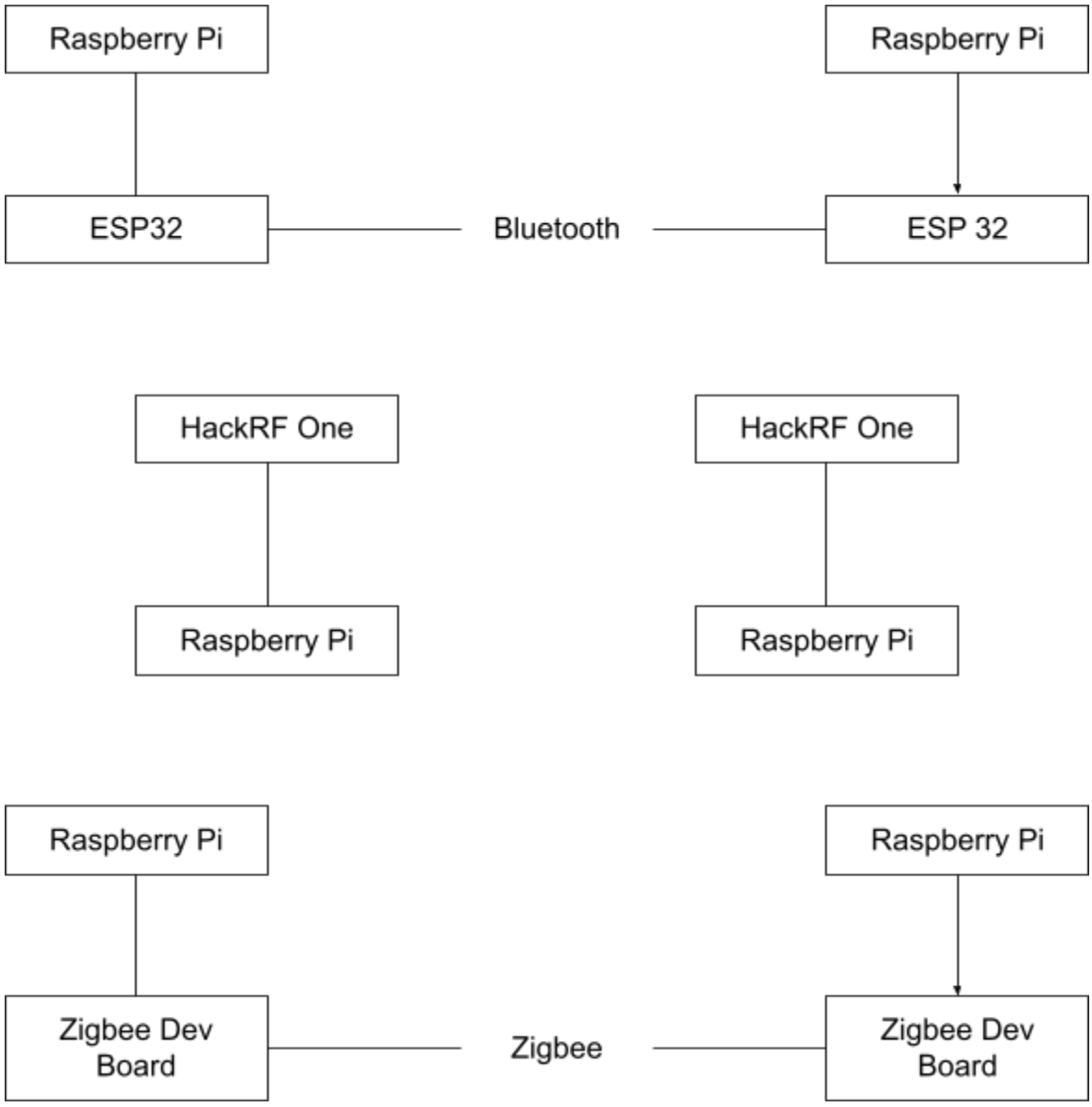

Figure 5: Diagram of Internal Communications Architecture

3 HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator.

3.1 Inputs

The user is required to provide the frequency that is being monitored and the frequency that is being used (for Bluetooth and Zigbee it will be 2.4 GHz to 2.5 GHz). The frequency range in HackRF Spectrum Analyzer will need to be set to the desired frequency range, in our case we will use 2.4 GHz to 2.5 GHz. In GNU Radio, the target frequency will need to be inputted depending on where the data is being transmitted as found in the spectrum analyzer as well as the data rate depending on which device is being jammed.

3.2 Outputs

When HackRF Spectrum Analyzer is opened the user will be presented with a graphical view of the data being transmitted on the selected frequency as well as the power in dB and frequency in MHz. When the GNU Radio jammer is run the user will be presented with a graph that shows the power level and frequency that is being sent through the HackRF one. In this graph, the user can adjust the zoom and move through past data transmissions

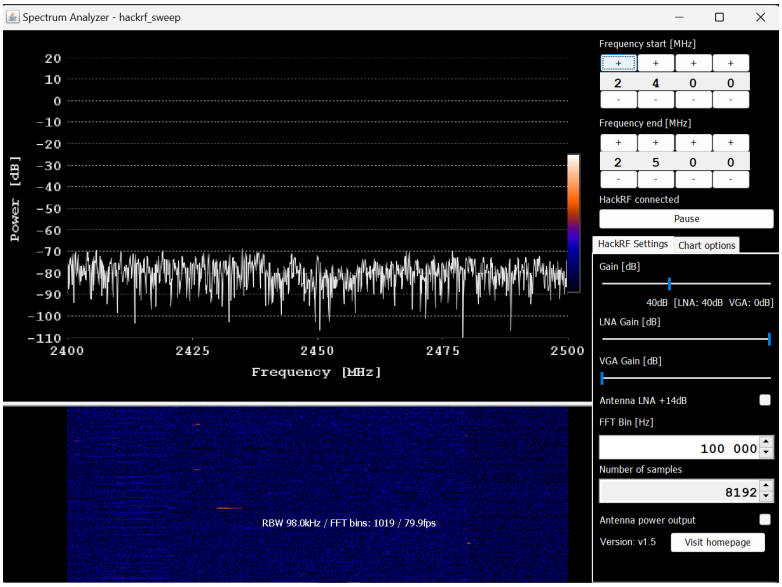

Figure 6: View of HackRF Spectrum Analyzer

4 DETAILED DESIGN

This section provides the information needed for a system development team to build and integrate the hardware components, code, and integrate the software modules, and interconnect the hardware and software segments into a functional product.

4.1 Hardware Detailed Design

The system is composed of several identical modules that connect to form either a Bluetooth or Zigbee network.

Each module that is part of the Bluetooth network is composed of a Raspberry Pi system and an ESP32 development board. Furthermore, each Raspberry Pi system is composed of a Raspberry Pi 4, a power supply, a display, a keyboard, and a mouse.

The mouse is an official Raspberry Pi mouse. The mouse connects to the Raspberry Pi board using a USB connection. The male USB connector is integrated into the mouse and will connect with a female USB port on the Raspberry Pi board. The mouse receives its power through the USB connection.

The keyboard is an official Raspberry Pi keyboard. The mouse connects to the Raspberry Pi board using a USB connection. Both the keyboard and the board have female USB ports and will be connected using a USB cable. The keyboard receives its power through the USB connection.

The display is an ELECROW 7-inch mini monitor. The display connects to the Raspberry Pi board using a HDMI connection. The display contains a female micro USB port and connects to the board using a micro USB to HDMI adapter which connects to the HDMI port on the board. The display receives its power through the HDMI connection. The display has a resolution of 1024x600 pixels and a refresh rate of 60Hz.

The power supply is the CanaKit 3.5A USB-C Raspberry Pi 4 Power Supply. The power supply connects to the Raspberry Pi board using a USB connection. The male USB connector is integrated into the supply and will connect with a female micro USB port on the Raspberry Pi board. See Figure 7.

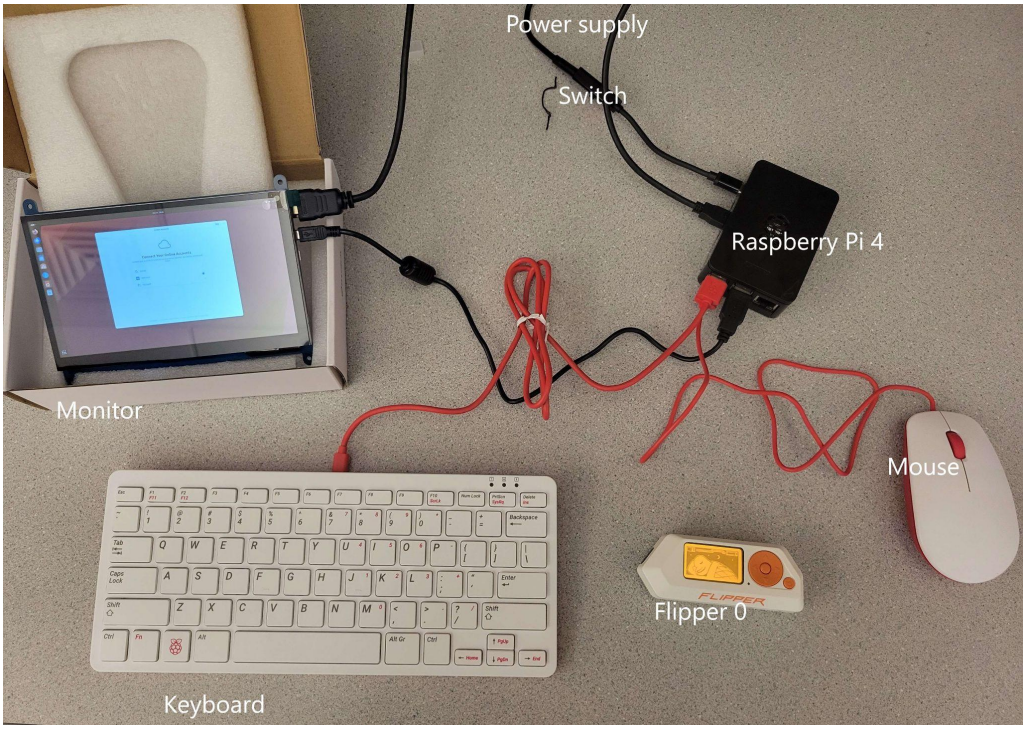

Figure 7: A Visual of the Raspberry Pi System

The Raspberry Pi systems connect to the Bluetooth network using ESP32 development boards. The ESP32 development board connects to the Raspberry Pi board using a micro USB cable. See Figure 8. Both components have female micro USB ports so a male-to-male micro USB cable will be used. This cable supplies power to the ESP32 development board. This ESP32 development board is what is used to transmit and receive data from other Raspberry Pi systems.

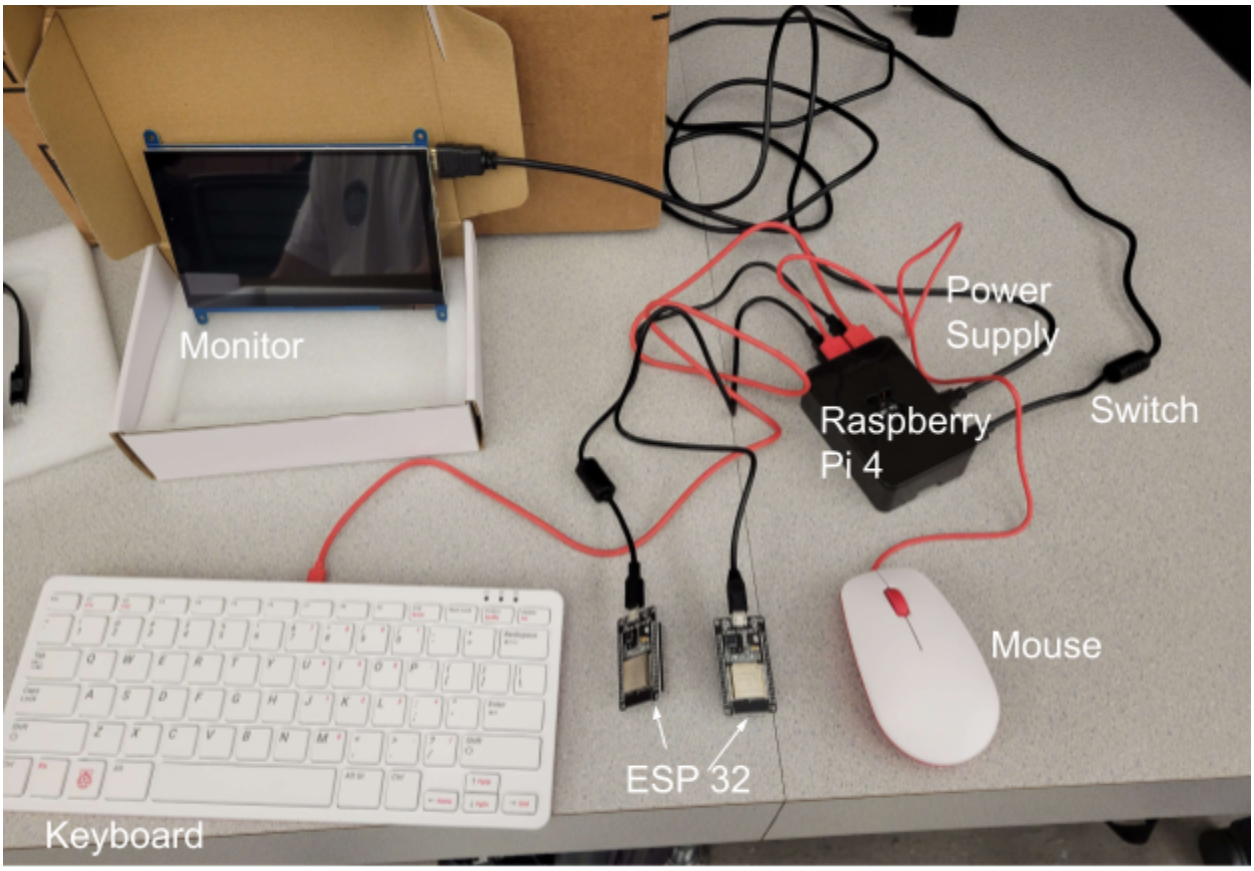

Figure 8: Bluetooth System

When the Raspberry Pi system is sending and receiving data over a Zigbee network a Zigbee development board is used instead of an ESP32 development board. The Zigbee development board connects to the Raspberry Pi board using a micro USB cable. See Figure 9. Both components have female micro USB ports so a male-to-male micro USB cable will be used. This cable supplies power to the Zigbee development board. This Zigbee development board is what is used to transmit and receive data from other Raspberry Pi systems.

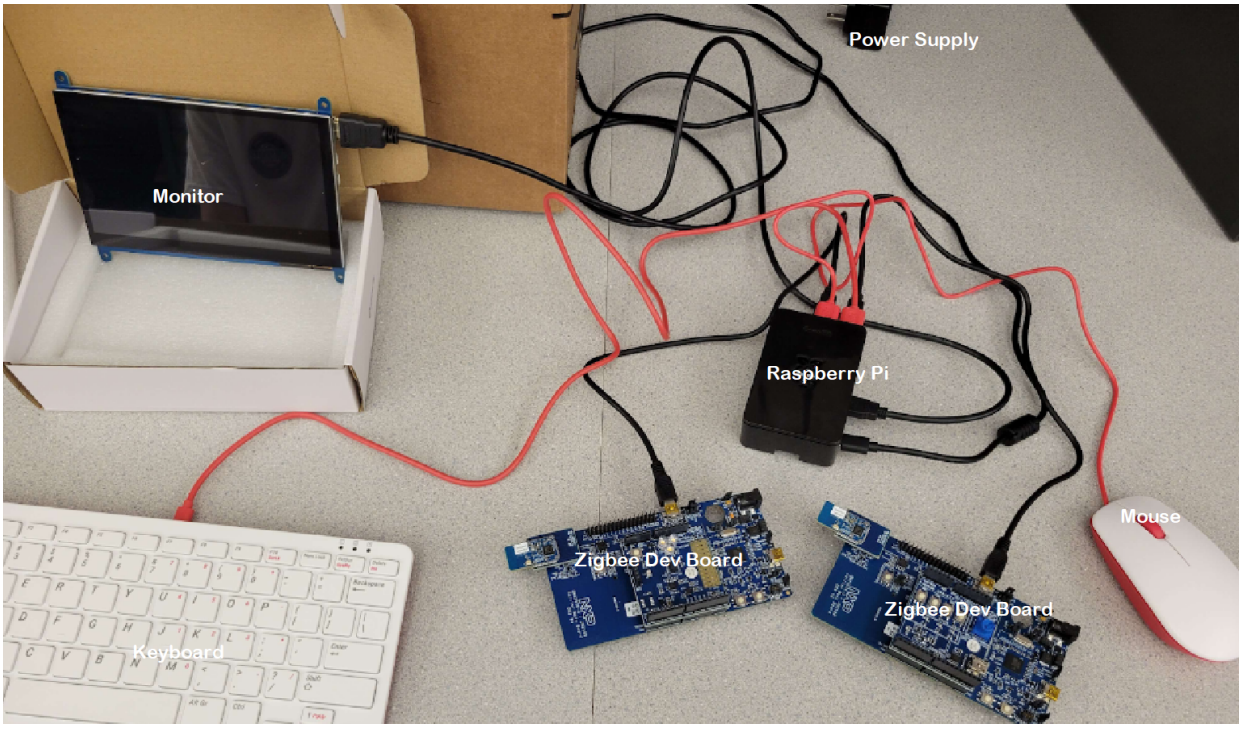

Figure 9: Zigbee System

In other cases, a Raspberry Pi system is connected to a HackRF One The HackRF connects to the Raspberry Pi board using a micro USB cable. See Figure 10. Both components have female micro USB ports so a male-to-male micro USB cable will be used. This cable supplies power to the HackRF One.

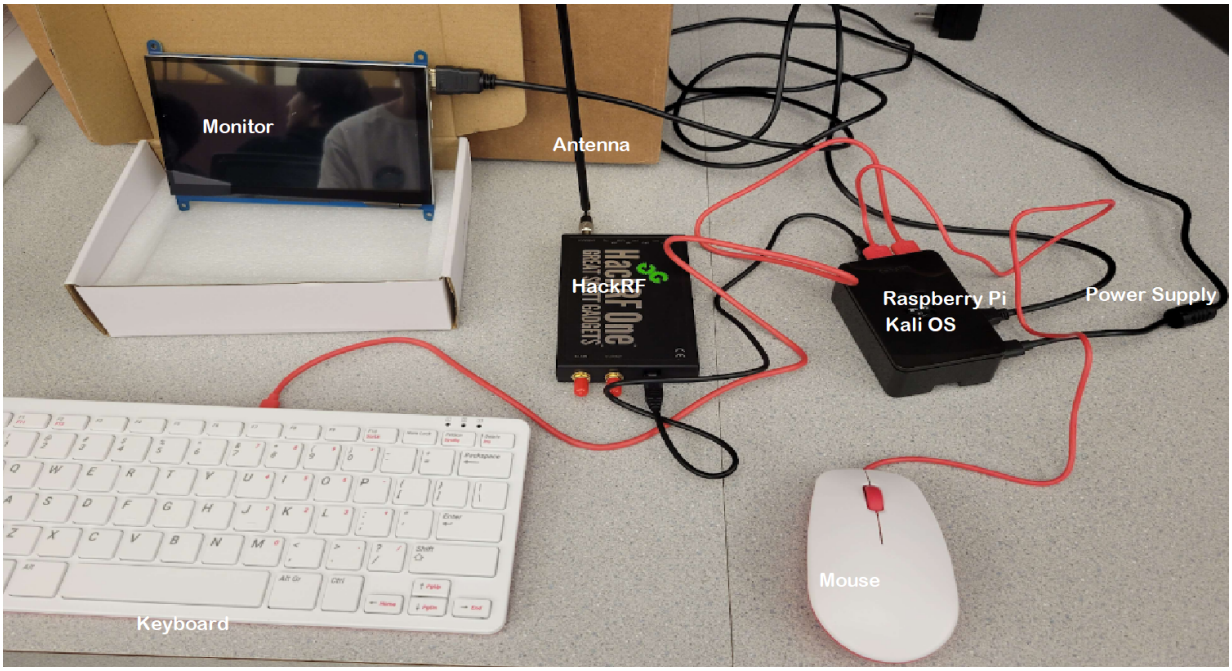

Figure 10: Software Defined Radio System

4.2 Software Detailed Design

The software is broken up into three sections: the jammer, frequency analyzer, and communication system. See Figure 11 below. The jammer system is responsible for transmitting RF signals on the 2.4 GHz to 2.5 GHz frequency. The frequency analyzer system is responsible for analyzing and decoding frequencies on the 2.4 GHz to 2.5 GHz to detect Bluetooth and Zigbee data transmission as well as confirm the jammer is working properly. The communication system is responsible for transmitting and receiving Zigbee and Bluetooth signals.

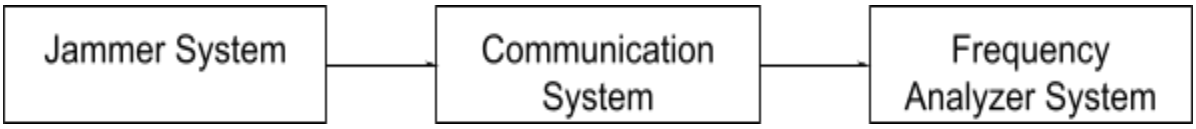

Figure 11: Software Architecture

4.2.1 Jammer System

The jammer system consists of two sub-systems: the RF signal generator and the control interface. See Figure 12 below. The RF signal generator is responsible for creating and transmitting RF signals in the 2.4 GHz to 2.5 GHz frequency band. It uses a high-frequency oscillator and a power amplifier to generate and transmit the signals. The control interface is a software component that allows the user to control the operation of the jammer system. It provides options to start or stop the jamming, adjust the frequency and power of the RF signals, and monitor the status of the jammer system.

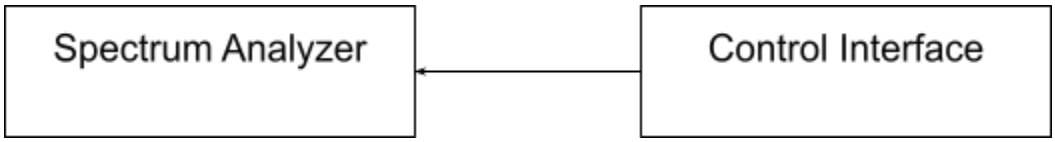

Figure 12: Jammer System

4.2.2 Frequency Analyzer System

The frequency analyzer system is designed to analyze and decode frequencies in the 2.4 GHz to 2.5 GHz band. See Figure 13 below. It consists of a spectrum analyzer and a decoding module. The spectrum analyzer scans the frequency band and captures the signals present in it. The decoding module then processes these signals to identify and decode any Bluetooth and Zigbee data transmissions. The system also verifies the operation of the jammer by detecting the presence of its RF signals in the frequency band.

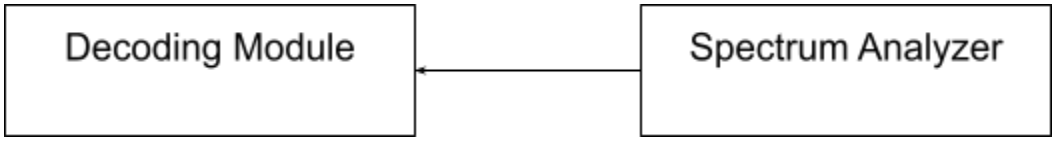

Figure 13: Frequency Analyzer System

4.2.3 Communication System

The communication system is responsible for handling Zigbee and Bluetooth communications. See Figure 14 below. It consists of two sub-systems: the Zigbee communication module and the Bluetooth communication module. The Zigbee communication module handles the transmission and reception of Zigbee signals. It uses a Zigbee transceiver and a microcontroller to encode and decode the Zigbee data. The Bluetooth communication module handles the transmission and reception of Bluetooth signals. It uses a Bluetooth module and a microcontroller to encode and decode the Bluetooth data. Both modules are designed to operate in the presence of the jammer system's RF signals.

Figure 14: Communication System

4.3 Internal Communications Detailed Design

Modules on the Bluetooth network communicate with each other following the specifications outlined by the Bluetooth protocol. Modules on the Zigbee network communicate with each other following the specifications outlined by the Zigbee protocol.

5 EXTERNAL INTERFACES

The system does not communicate with any external systems.

6 SYSTEM INTEGRITY CONTROLS

The system will pick up Bluetooth and Zigbee signals which are widely used by most IOT devices. These signals can be picked up by anyone with a frequency analyzer, which can be installed on most mobile devices and computers. The system will be used in a controlled environment to prevent signal interruption and potential interceptions. This will also ensure that the Bluetooth and Zigbee signals can be correctly detected by the HackRF Spectrum Analyzer. The data collected from the HackRF Spectrum Analyzer will be downloaded and stored locally on the Raspberry Pi for future analysis if needed. The data will be securely stored and deleted once used to prevent frequency data from exposure.

The Raspberry Pis will have a kill switch on the power supply in case of an emergency. If the kill switch is pressed on the power supply the Raspberry Pi will lose power instantly. This will also be used in case of a signal leak to prevent potential exposure of Zigbee, Bluetooth, or jamming signals.