System Design Document

For

Bluetooth Protocol Analytical Research

Gianna Scarangelli
Matthew Irvin
Carolina Terre
Jonah Rowell
Connal Grace

Version/Author	Date
v1	9/29

TABLE OF CONTENT

1 INTRODUCTION.....3
1.1 Purpose and Scope.....3
1.2 Project Executive Summary.....3
1.2.1 System Overview.....3
1.2.2 Design Constraints.....3
1.2.3 Future Contingencies.....3
1.3 Document Organization.....3
1.4 Project References.....4
1.5 Glossary.....4
2 SYSTEM ARCHITECTURE.....4
2.1 System Hardware Architecture.....4
2.2 System Software Architecture.....4
2.3 Internal Communications Architecture.....4
3 HUMAN-MACHINE INTERFACE.....4
3.1 Inputs.....5
3.2 Outputs.....5
4 DETAILED DESIGN.....5
4.1 Hardware Detailed Design.....6
4.2 Software Detailed Design.....6
4.3 Internal Communications Detailed Design.....7
5 EXTERNAL INTERFACES.....7
5.1 Interface Architecture.....7
5.2 Interface Detailed Design.....8
6 SYSTEM INTEGRITY CONTROLS.....8

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.

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.
[we can also insert a diagram here]
The architecture diagram showcases the interaction between the server, client, and analysis subsystems while illustrating external interfaces that might be pertinent, such as the interface with a database for storing test results and analysis data.

2.2 Context Diagram
The context diagram further simplifies the system's interaction with external entities and encapsulates the flow of information amongst the subsystems.

[Insert Context Diagram Here]

2.3 External System Interfaces

Database Interface: A secure database interface will be essential for storing, retrieving, and analyzing data generated during the testing phase.
Monitoring Interface: An interface that allows real-time monitoring of the system's operations, ensuring the integrity and availability of the testing environment.
2.4 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.

[Refer to RTM in the Functional Requirements Document]

This System Overview provides a high-level understanding of the project's architecture, setting a foundation for the in-depth design discussion in the subsequent sections of this document.

Make sure to include your diagrams where indicated, and adapt the narrative and subsystem breakdown to better fit your project's unique specifications and requirements.

1.2.2 Design Constraints

Investigating vulnerabilities within wireless IoT technologies, notably within Bluetooth 5 and Zigbee protocols, is restrained by several notable constraints: 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. 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

Supply a glossary of all terms and abbreviations used in this document. If the glossary is several pages in length, it may be included as an appendix.

2 SYSTEM ARCHITECTURE

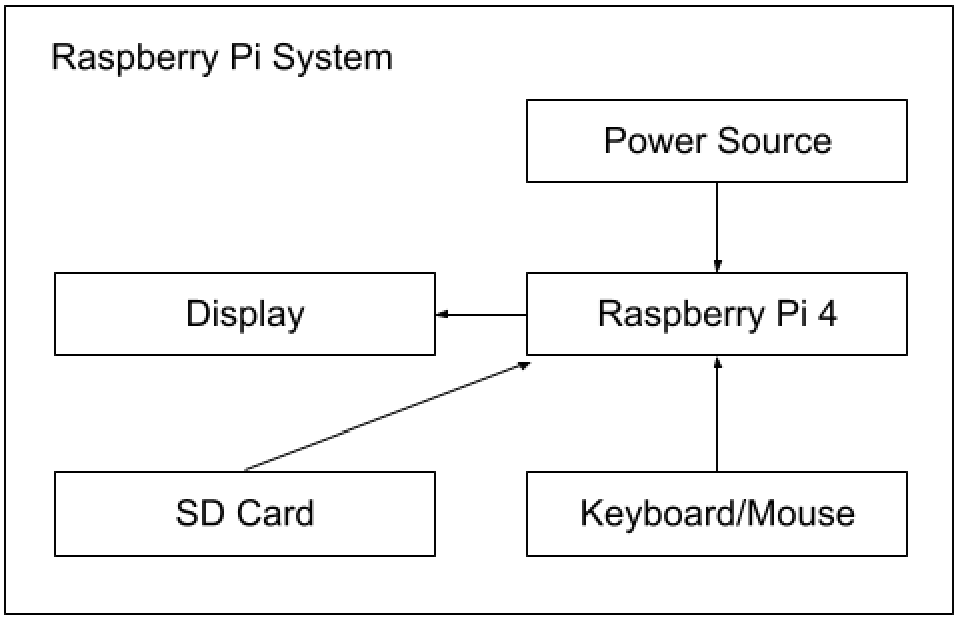
This section describes and gives an overview of the systems and subsystems used in this project.

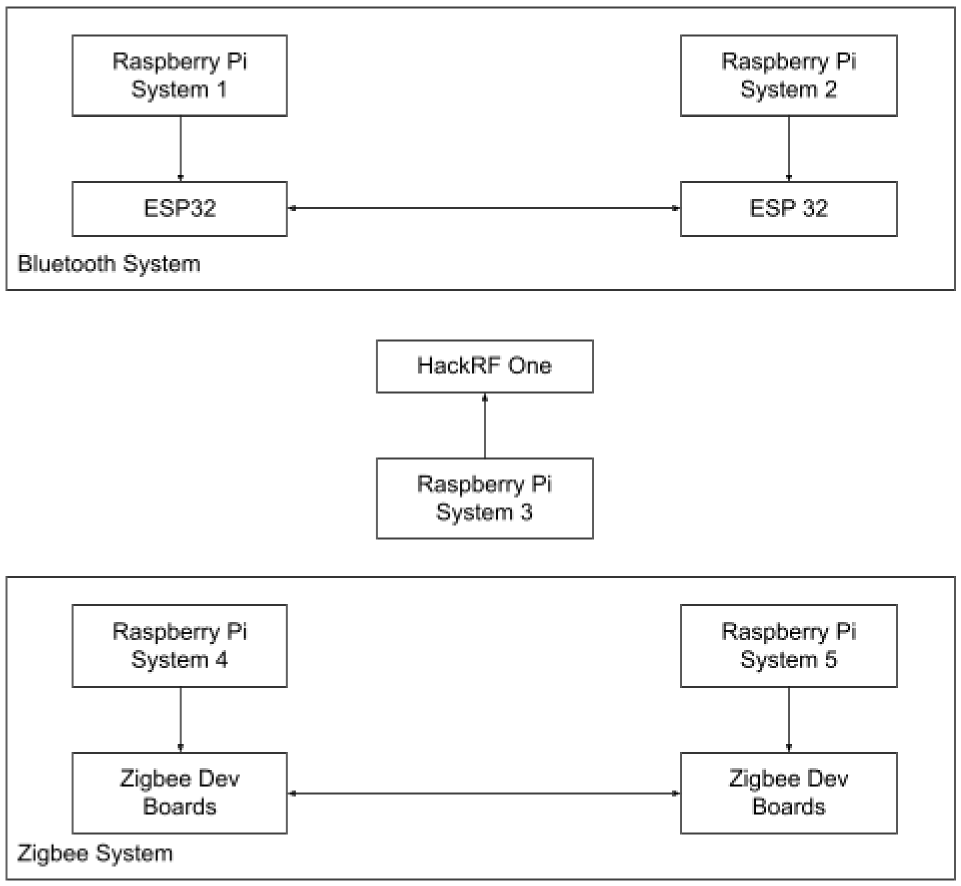
2.1 System Hardware Architecture

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

Zigbee Development Boards: Development boards for testing Zigbee signals and devices
HackRF One: Software-defined radio used for transmitting signals
ESP32: Microcontroller with onboard Wi-Fi and Bluetooth
Raspberry Pi: Microcomputer used to transmit and receive Bluetooth signals as well as to control various tasks within the system
SD Cards: Small storage devices for Rasberry Pi
SSD: Large portable storage devices used for file storage on Rasberry Pis.
Peripherals: Displays, mouse, and keyboard to control Rasberry Pis as well as a case to protect the Rasberry Pi. An SD card reader with a USB adapter will be needed to connect the SD card to computers

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 Rasberry 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.

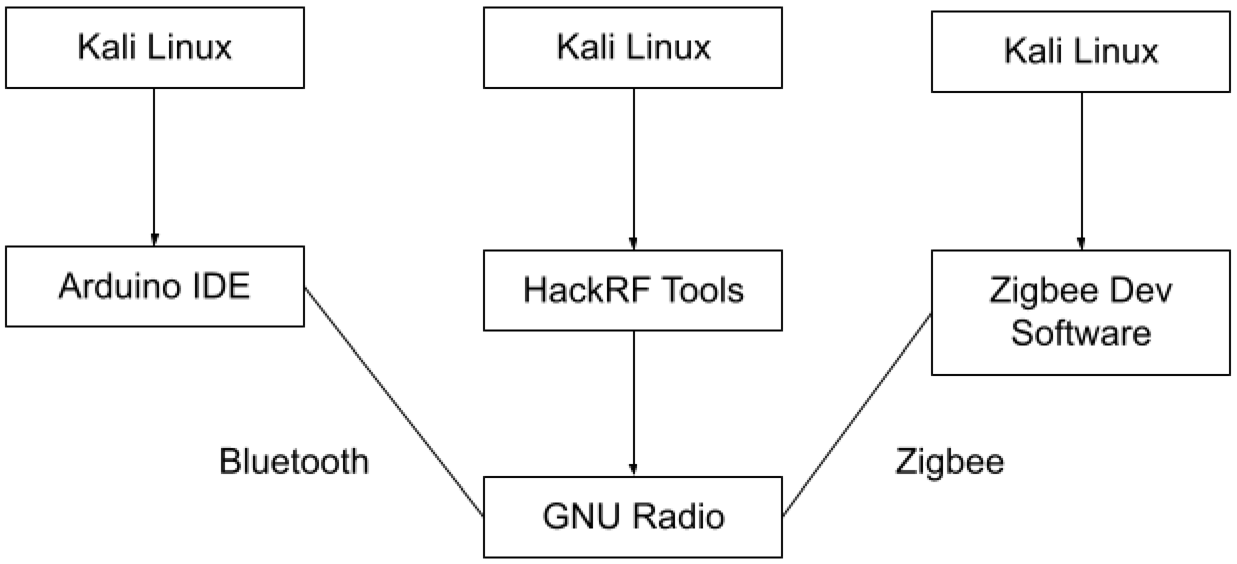




2.2 System Software Architecture

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 Aurdino IDE to run the ESP32s, Two will have the Zigbee Development Software and one will have the HackRF Tools and GNU Radio installed.

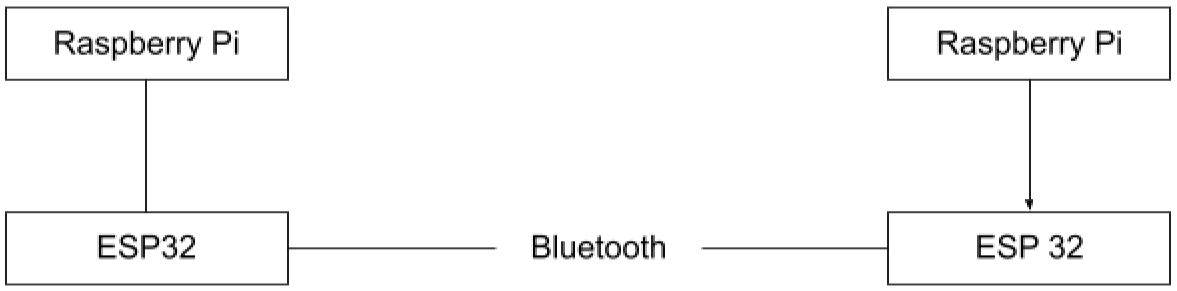
Kali: Linux-based operating system can be used for pen testing
Rasberry PI OS: Operating system for Rasberry Pi
GNU Radio: Software used to control software-defined radios such as the HackRF One
HackRF Tools: Command line to interact with HackRF
Audrino IDE: Used to program ESP32
Zigbee Development Kit Software: Software used to control Zigbee boards

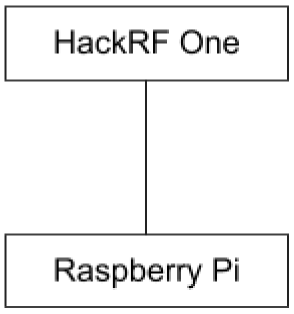


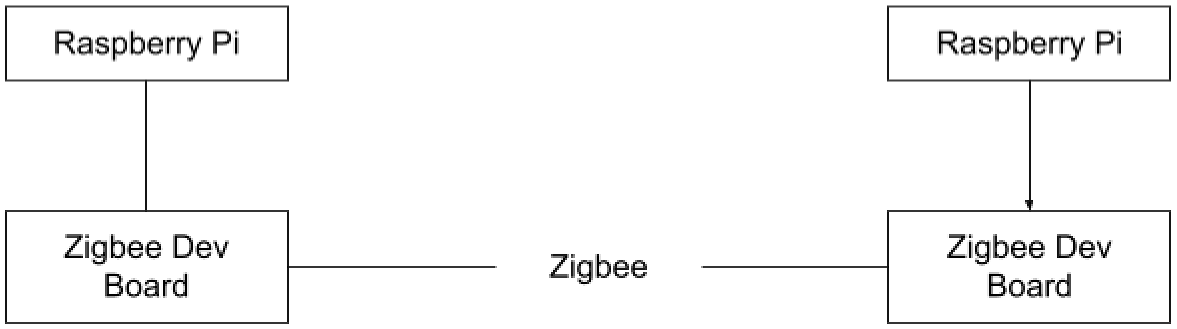
2.3 Internal Communications Architecture

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 in order 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







3 HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs.

3.1 Inputs

This section is a description of the input media used by the operator for providing information to the system; show a mapping to the high-level data flows described in

Section 1 .2.1, System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in Section 3, File and Database Design, may be referenced. Include data element definitions, or refer to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

- Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
- Description of any access restrictions or security considerations
- Each transaction name, code, and definition, if the system is a transaction-based processing system

3.2 Outputs

This section describes of the system output design relative to the user/operator; show a mapping to the high-level data flows described in Section 1.2.1. System outputs include reports, data display screens and GUIs, query results, etc. The output files are described in Section 3 and may be referenced in this section. The following should be provided, if appropriate:

- Identification of codes and names for reports and data display screens
- Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
- Description of the purpose of the output, including identification of the primary users
- Report distribution requirements, if any (include frequency for periodic reports)
- Description of any access restrictions or security considerations

4 DETAILED DESIGN

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

4.1 Hardware Detailed Design

The system is composed of several identical modules that connect together 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 to each other 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.

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. 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.

When the Raspberry Pi system is sending and receiving data over a Zigbee network a Zigbee development board is used instead of a ESP32 development board. The Zigbee development board connects to the Raspberry Pi board using a micro USB cable. 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.

In some 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. 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.

4.2 Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about the logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

- A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
- For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
- Data elements, record structures, and file structures associated with module input and output
- Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
- Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
- Report layout

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

External systems are any systems that are not within the scope of the system under development, regardless of whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

5.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

5.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

- The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
- Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
- Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
- Graphical representation of the connectivity between systems, showing the direction of data flow
- Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

6 SYSTEM INTEGRITY CONTROLS

Sensitive systems use information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of State programs or the privacy to which individuals are entitled.

Developers of sensitive State systems are required to develop specifications for the following minimum levels of control:

- Internal security to restrict access of critical data items to only those access types required by users
- Audit procedures to meet control, reporting, and retention period requirements for operational and management reports

- Application audit trails to dynamically audit retrieval access to designated critical data
- Standard Tables to be used or requested for validating data fields
- Verification processes for additions, deletions, or updates of critical data

Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.