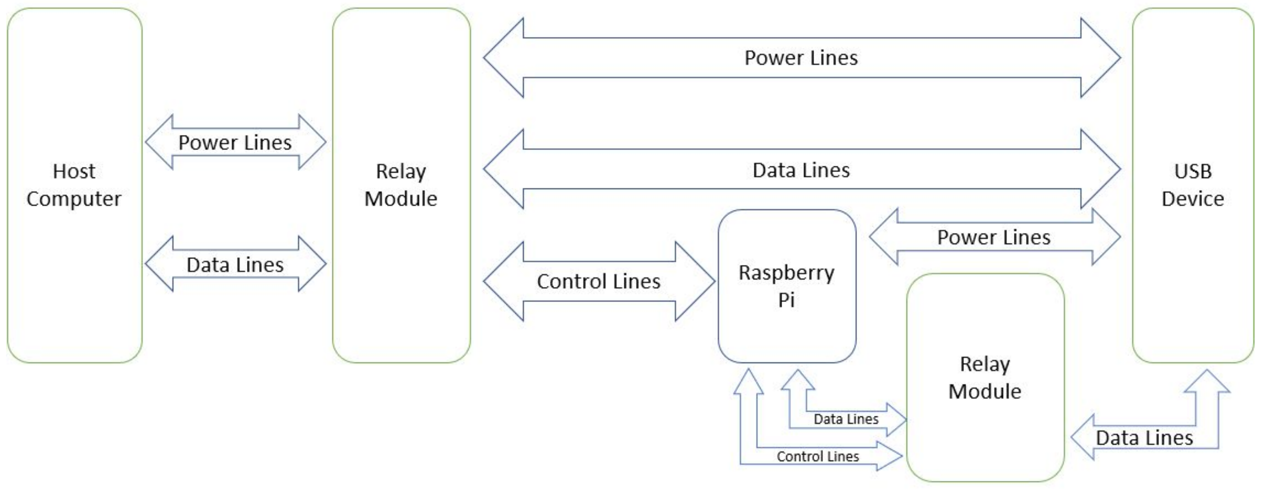
Chapter 3

3.1 Introduction

A more detailed examination of USafeB's proposed approach and design decisions is presented in this chapter. Section 3.2 outlines the general logical flow of the project as well as showcases its . Section 3.3 explores the system's targeted features in detail, in addition to demonstrating its functional and non-functional requirements. Section 3.4 explores key design decisions that were made and describes the process behind their selection.

3.2 Design Overview



(Figure 2. Overall design of USafeB)

USafeB's design philosophy is derived from the original requirement of having an isolated sandbox for USB devices. The goal for an isolated middleware to mediate between a host computer and a USB device is achieved by creating a circuit which ensures that the host will only be able to gain access to a device only if it's provided with the middleware's permission.

Once a user connects their USB device on USafeB's hardware, it remains logically disconnected from their host device until USafeB's hardware issues the appropriate commands to the modules which physically conduct the separation. It is then the host computer will be allowed to access the device's data. This is done without the hardware itself behaving is a tunnel between the two devices. Once permission is granted, a direct connection between the host and the USB device will be enabled.

If there is anything that the user wishes to extract from a USB device that was deemed unsafe, the files on the device itself will be rendered unexecutable so that the direct connection can still be established.

Using USafeB will offer users a safe USB security solution that's portable across all platforms and is easy to deploy. This is especially true in regards to computer users who are not so as to properly secure their own devices, accommodating them is one of the primary goals of USafeB. They receive a prepackaged system that simply receives a USB drive, scans it, gives the red or green light, then offers safe file exchange functionality, or forwards the entire drive.

3.3 System Features

3.3.1 Requirements Overview

Actors	**Description**
Logically separated hardware circuit	The circuit design should allow the USB device to be examined in isolation from the host computer until permission for connection is established.
Software scanning mechanism	The middleware should have the ability to detect multiple attacks successfully so as to not allow malicious files to go unnoticed.

(Table 5: An overview of the project's requirements)

3.3.2 Functional Requirements

- USafeB should be able to connect to the Internet.
- USafeB should have a reliable scanning/filtering mechanism that could be used successfully to defend against malicious attack vectors.
- USafeB should be able to forward files that are allowed and deny files which are not.
- USafeB should implement defenses against attacks described in Table 4, Chapter 2.

3.3.3 Non-functional Requirements

- USafeB should operate independently, regardless of the user's target platform.
- USafeB should be simple to use, not requiring elaborative user interaction.

3.4 Design Analysis

3.4.1 Hardware Considerations

Single board computer manufacturers, such as Raspberry[], Beagleboard[], UDOO[] among others, are fairly competitive. The decision to select Raspberry as USafeB's sandbox hardware was made upon the following considerations:

- Local availability in the Egyptian market
- Best community support among other options

And among the Raspberry products, there are several models that were surveyed to determine the best for USafeB as displayed in Table 6..

Raspberry Pi Model	**USB Ports**	**CPU**	**RAM**	**Price (EGP)**
Raspberry Pi 1 B	2x USB 2.0	700MHz	512MB	Unavailable
Raspberry Pi 1 B+	4x USB 2.0	700MHz	512MB	Unavailable
Raspberry Pi 1 A+	1x USB 2.0	700MHz	512MB	Unavailable
Raspberry Pi 2	4x USB 2.0	900MHz	1GB	Unavailable
Raspberry Pi Zero	1x micro-USB	1GHz	512MB	500
Raspberry Pi 3	4x USB 2.0	1.2GHz	1GB	1200
Raspberry Pi Zero W	1x micro-USB	1GHz	512MB	550
Raspberry Pi 3 B+	4x USB 2.0	1.4GHz	1GB	1500
Raspberry Pi 3 A+	1x USB 2.0	1.4GHz	512MB	1300
Raspberry Pi 4	2x USB 2.0 + 2x USB 3.0	1.5GHz	1GB, 2GB, or 4GB	1120

(Table 6: List of Raspberry Pi models and their specifications, along with their availability in the Egyptian markets)

The required model's selection was based on the following considerations:

- **High performance**: One of the primary goals of USafeB is to provide a portable, platform-independent device to protect USB hosts against malicious external attacks, and for that to yield the highest performance results, it would be preferable to use the fastest and latest processor.
- **RAM**: The Pi's operating system will reside on an external SD card, but a moderately high amount of RAM is still required for encryption features USafeB is going to implement.
- **USB ports**: While USB 3.0 ports might not be needed for immediate purposes, it is necessary to have for purposes of forward compatibility so that USafeB can eventually be improved to support and be tested on USB 3.0.
- **Price considerations**: Despite performance requirements, the goal is still to design a system that should become affordable. In which case, compromises should be made if possible, and unnecessary features should not be paid for.

Raspberry Pi 4 Model B (1GHz) is the most ideal for USafeB's purposes.

3.4.2 Software Considerations

After selecting Raspberry Pi, a survey was held to determine the operating system it should work on. The important thing was for it to be lightweight so as to not consume the device's resources and slow down its operation, and having the necessary packages that USafeB's codebase should rely on. The following table represents a certain set of options considered for USafeB:

Operating System	**Base**	**Package manager**
Raspbian	Debian	apt
Ubuntu Core	Debian	apt
Windows IoT Core	Windows	ADK Add-ons
Miniban OS	Debian	apt
RaspBSD	FreeBSD	pkg-install
Gentoo	Independent	Portage

Alpine Linux	Independent	apk

(Table 7: List of Operating Systems and their file formats)

The operating system requirement is less expensive to modify, so less risk is involved is flawed, thanks to the nature of Linux-based operating systems. Most of these options provide the basic requirements for the packages necessary to be easily available and for there to be no problems setting up and testing the design of USafeB.

That is a more difficult task to achieve on operating systems based on BSD such as RaspBSD, use musl libraries instead of glibc such as Alpine (which does not guarantee the packages required by USafeB to be functioning properly) as well as distributions that require more fine-tuning such as Gentoo in order to function properly. A decision was made to avoid the Microsoft option as Linux-based distributions are more efficient in the lower-levels.

With that in mind, a Debian-based distribution was preferred, and Raspbian, being the most popular distribution for Raspberry Pi devices, is fitting for the task. A decision may be made to migrate to Miniban OS at a later time to improve performance.

3.5 Conclusion

The previous sections should have given the reader the awareness necessary to follow through with the implementation details of the following chapter. The project's overall form was outlined along with the key considerations shadowing it. The upcoming chapters will deal with the realization, findings and performance evaluation of the requirements outlined above.