| **Group** 13 | *Spring 2024 Design Challenge* |
| --- | --- |
| **Major:** | **Team members:** |
| *ITC* | *David Kajuna* |
| *CEG* | *Joshua Andrews* |
| *CS* | *Jason Bynum* |
| *ITC*  *CS* | *John Armlovich*  *Kennedy Eziolise* |

**HW/SW Design Trade**

Hardware/software design choice to be made (include its identification within your system architecture):

**HW Design Options**

1. **Rebel Raspberry Pi**

|  | **Design Options** | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***Raspberry Pi 4, 2GB*** | ***Raspberry Pi 4, 4GB*** | ***Raspberry Pi 4, 4GB, 64Gb Memory*** | ***Raspberry Pi 4, 4GB, 256GB Memory*** |
| ***Req. 10.4*** | Could have transmission and reception capabilities. | Could have transmission and reception capabilities. | Could have transmission and reception capabilities. | Could have transmission and reception capabilities. |
| ***Const. 20***  ***(Budget)*** | Low cost  (54$ for Pi and Server Connection) | Low cost  ($64 for Pi and Server Connection) | High cost  ($71.3 for Pi and Server Connection) | High cost with no return.  ($101.51 for Pi and Server Connection) |
| ***Req. 10.5*** | Not enough memory to contain all the programs | Not enough memory to contain all the programs | Enough storage to contain all programs | Enough storage to contain all programs |
| ***Req. 20.4*** | No Operating System | No Operating System | Can use OpenSSL for AES | Can use OpenSSL for AES |
| ***Req. 10.1***  ***(GPIO pins)*** | Can interface with transceiver system | Can interface with transceiver system | Can interface with transceiver system | Can interface with transceiver system |
| ***Std. 70*** | Cannot store .png files in the correct format | Cannot store .png files in the correct format | .png files can be stored in the correct format | .png files can be stored in the correct format |

The Raspberry Pi 4 with 4GB of RAM and 64GB of removable memory will be used for this hardware block. This device would be able to fill all requirements for the least necessary cost of the devices decided upon. The software planned on being used requires the device to have some form of storage and therefore the Raspberry Pis without memory cannot be used. Any size of memory larger than 64GB would be more expensive than what is needed and therefore the Raspberry Pi with 256GB of memory is not used.

1. **Li-Fi Transceiver System**
   1. **Transmission Capabilities**

|  | **Design Options** | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***400nm laser*** | ***535nm Laser*** | ***650nm laser*** | ***470nm LED light*** |
| ***Req. 10.1*** | Can implement Li-Fi transmission | Can implement Li-Fi transmission | Can implement Li-Fi transmission | Can implement Li-Fi transmission |
| ***Req. 10.9*** | Part is low power, high cost | Part is low power, high cost | Part is low power and cost | Part is low power and cost |
| ***Req. 10.2*** | Lossless transmission | Lossless transmission | Lossless transmission | Lossless transmission |
| ***Req 10.4*** | Can interface with Raspberry Pi | Can interface with Raspberry Pi | Can interface with Raspberry Pi | Can interface with Raspberry Pi |
| ***Reg. 10.10*** | Input power of < 5V | Input power of < 5V | Input power of 5V | Input power of < 5V |
| ***Std. 10*** | Follows typical Li-Fi standard | Follows typical Li-Fi standard | Follows typical Li-Fi standard | Follows typical Li-Fi standard |
| ***Const. 50*** | Can transmit beyond 5 meters | Can transmit beyond 5 meters | Can transmit beyond 5 meters | Can not transmit beyond 5 meters |
| ***Const. 20 (Budget)*** | Higher Cost  ($42.97) | Higher Cost  ($42.97) | Low Cost  ($36.92) | Low Cost  ($36.92 20-Pack) |

* 1. **Reception Capabilities**

|  | **Design Options** | | |
| --- | --- | --- | --- |
| ***Criteria*** | ***5V, 1A, Solar Cell*** | ***6V, .8A, Solar Cell*** | ***Photodiode Sensor*** |
| ***Req. 10.1*** | Can implement Li-Fi reception | Can implement Li-Fi reception | Can implement Li-Fi reception |
| ***Req. 10.2*** | Lossless Reception | Lossless Reception | Lossless Reception |
| ***Req 10.4*** | Can interface with Raspberry Pi | Can interface with Raspberry Pi | Can interface with Raspberry Pi |
| ***Req. 10.8*** | No obstructions present | No obstructions present | No obstructions present, needs excessive precision |
| ***Req 10.9*** | low cost, low power | Low cost, low power | Low cost, low power |
| ***Req 10.10*** | Takes 5V input | Takes 5V input after manipulation | Takes > 5V input |
| ***Const. 20***  ***(Budget)*** | Low Cost  ($8.95 for laser and implementation) | Higher Cost  ($20.95 for laser and implementation) | Higher Cost  ($24.97 for laser and implementation) |
| ***Std. 10*** | Follows typical Li-Fi standard | Follows typical Li-Fi standard | Follows typical Li-Fi standard |

The 650nm laser in conjunction with the 5V@1A solar cell will be used for our Li-Fi transceiver system. The 650nm laser takes in the correct input power as opposed to the other possible transmission modules as well as being low-cost. The 650nm laser can also transmit the required 5 meters from the Empire Lab. The 5V@1A solar cell takes in the required 5V input while being low power and low cost. Moreover, The 5V solar cell will be able to take in the required input without circuit manipulation in comparison to the other reception devices.

1. **Mobile Device**

|  | **Design Options** | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***iPhone 12 Pro*** | ***Samsung Galaxy S23+*** | ***Samsung Galaxy S23+ Emulator*** | ***iPhone 12 Pro Emulator*** |
| ***Const. 20***  ***(Budget)*** | Personally Owned | Personally Owned | Little to no cost to implement | Cost of operating system |
| ***Req. 50.1*** | Able to display scrollable table (Limited) | Able to display scrollable table | Able to display scrollable table | Able to display scrollable table (Limited) |
| ***Req. 50.2*** | Able to run user-created software (Limited) | Able to run user-created software | Able to run user-created software | Able to run user-created software (Limited) |
| ***Req. 50.4*** | Able to connect to the server over Wi-Fi | Able to connect to the server over Wi-Fi | Able to connect to the server over Wi-Fi | Able to connect to the server over Wi-Fi |

A personally owned Samsung Galaxy S23+ will be used for the mobile device. Due to low familiarity with Swift coding language, iPhones would be harder to implement. An Android mobile device would be easier to implement due to already being personally owned, and a higher familiarity with the C++ coding language. We also have a physical device available for use so we will choose that over an emulator.

1. **Rebel Server**

|  | **Design Options** | |
| --- | --- | --- |
| ***Criteria*** | ***Windows*** | ***Linux*** |
| ***Std. 60*** | OS can follow HTTPS protocols | OS can follow HTTPS protocols |
| ***Req. 50*** | Hosted Server can allow downloads | Hosted Server can allow downloads |
| ***Req 50.4*** | Can host a server | Can host a server |
| ***Const. 50*** | The server can be placed within 5 meters of Empire Lab | The server can be placed within 5 meters of Empire Lab |
| ***Req 50.5*** | The server can be publicly connected using Nginx (Limited) | The server can be publiclyconnected using Nginx |

A Linux OS will be used for the Rebel Server. A Linux system is easier to implement due to more familiarity with coding and running websites and servers. Linux servers are much less of a hassle when setting up and operating. Nginx software will be used to host the server which the group has experience running in Linux.

**SW Design Options**

1. **Death Star Image Evaluation application**

|  | **Design Options** | | |
| --- | --- | --- | --- |
| ***Criteria*** | ***Python*** | ***C++*** | ***Java*** |
| ***Std. 70*** | pillow library can process .png images | pillow library can process .png images | pillow library can process .png images |
| ***Std. 20*** | Follows Python coding standards | Does not follow Python coding standards | Does not follow Python coding standards |
| ***Req. 30*** | pillow library can process .png images | OpenCV library can process .png images | OpenCV library can process .png images |
| ***Req. 30.1*** | The library can theoretically identify Death Star images | The library can theoretically identify Death Star images | The library can theoretically identify Death Star images |
| ***Req 30.3*** | Can be run on Raspberry Pi | Can be run on Raspberry Pi | Can be run on Raspberry Pi |

Python will be used in conjunction with the pillow image processing library to develop the Image Evaluation Application. Proof of work has been shown to verify that the pillow libraries will be able to find Death Star pictures in groups of similar objects. OpenCV and therefore Java and C++ would be much harder to implement than Python because Pillow is created for Python specifically whereas OpenCV is not.

1. **Empire Lab Pi Image Transceiving Application**

|  | Design Options | | |
| --- | --- | --- | --- |
| ***Criteria*** | ***Python*** | ***C++*** | ***C*** |
| ***Const. 10*** | Interfaces with Li-Fi system using WiringPi Library (limited) | Interfaces with Li-Fi system using WiringPi Library | Interfaces with Li-Fi system using WiringPi Library |
| ***Req. 10.5*** | Applicable software language | Applicable software language | Applicable software language |
| ***Req. 10.4*** | Will use both received and transmitted data | Will use both received and transmitted data | Will use both received and transmitted data |
| ***Req. 10.7*** | Can transmit file formatting data | Can transmit file formatting data | Can transmit file formatting data |
| ***Std. 40*** | Does not follow C++ coding standards | Follows C++ coding standards | Follows C coding standards |

C coding language will be used to implement the Image Transceiving Application. This is because C interacts with the GPIO pins of the Raspberry Pi with the least overhead work using an external library compared to other languages. Moreover, code found by the group during research which will be used as a baseline, used C for their work.

1. **Empire Data Verification Application**

|  | Design Options | |
| --- | --- | --- |
| **Criteria** | **Bash** | **Korn Shell** |
| **Req. 20** | Can implement Openssl AES-256 encryption/decryption | Can implement Openssl AES-256 encryption/decryption (limited) |
| **Req. 20.1** | Can compare binary file values | Can compare binary file values |
| **Req. 20.2** | Can implement Openssl AES-256 encryption/decryption | Can implement Openssl AES-256 encryption/decryption (limited) |
| **Req. 40** | Can implement scripting and md5 | Can implement scripting and md5 |
| **Req. 40.1** | Can implement scripting and md5 | Can implement scripting and md5 |
| **Req. 40.2** | Bash shell is preloaded on Empire Raspberry Pi OS | It would have to be installed separately on Empire Raspberry Pi. |
| **Std. 50** | AES-256 follows common encryption standard | AES-256 follows common encryption standard |

Bash scripting will be used to implement the Empire Data Verification Application. This is because bash is more comfortable for the design team to implement. Additionally, bash is native to the common Raspberry Pi operating system where Korn shell would have to be installed separately.

1. **Rebel Data Verification Application**

|  | Design Options | |
| --- | --- | --- |
| **Criteria** | **Bash** | **Korn Shell** |
| **Req. 20** | Can implement Openssl AES-256 encryption/decryption | Can implement Openssl AES-256 encryption/decryption (limited) |
| **Req. 20.1** | Can compare binary file values | Can compare binary file values |
| **Req. 20.2** | Can implement Openssl AES-256 encryption/decryption | Can implement Openssl AES-256 encryption/decryption (limited) |
| **Req. 40** | Can implement scripting and md5 | Can implement scripting and md5 |
| **Req. 40.1** | Can implement scripting and md5 | Can implement scripting and md5 |
| **Req. 40.2** | Bash shell is preloaded on Empire Raspberry Pi OS | It would have to be installed separately on Empire Raspberry Pi. |
| **Std. 50** | AES-256 follows common encryption standard | AES-256 follows common encryption standard |

Bash scripting will be used to implement the Rebel Data Verification Application. This is because bash is more comfortable for the design team to implement. Additionally, bash is native to the common Raspberry Pi operating system where Korn shell would have to be installed separately.

1. **Rebel Pi Image Transceiving Application**

|  | Design Options | | |
| --- | --- | --- | --- |
| ***Criteria*** | ***Python*** | ***C++*** | ***C*** |
| ***Const. 10*** | Interfaces with Li-Fi system using WiringPi Library (limited) | Interfaces with Li-Fi system using WiringPi Library | Interfaces with Li-Fi system using WiringPi Library |
| ***Req. 10.5*** | Applicable software language | Applicable software language | Applicable software language |
| ***Req. 10.4*** | Will use both received and transmitted data | Will use both received and transmitted data | Will use both received and transmitted data |
| ***Req. 10.7*** | Can transmit file formatting data | Can transmit file formatting data | Can transmit file formatting data |
| ***Std. 40*** | Does not follow C++ coding standards | Follows C++ coding standards | Follows C coding standards |

C coding language will be used to implement the Image Transceiving Application. This is because C interacts with the GPIO pins of the Raspberry Pi with the least overhead work using an external library compared to other languages. Moreover, code found by the group during research which will be used as a baseline, used C for their work.

1. **Raspberry Pi to Server Interfacing**

|  | Design Options | |
| --- | --- | --- |
| **Criteria** | **Bash** | **Korn Shell** |
| **Req. 50** | Can move images to the server for hosting | Can move images to the server for hosting (limited) |
| **Req. 50.5** | Can move images to the server for hosting | Can move images to the server for hosting (limited) |
| **Std. 70** | Can send data as formatted .png’s | Can send data as formatted .png’s |

Bash will be used to move image data to the server. This is because Bash was used previously and there is no additional benefit of using a different shell script.

1. **Server Hosted Website**

|  | Design Options | | | | |
| --- | --- | --- | --- | --- | --- |
| ***Criteria*** | ***Nginx*** | ***Bash*** | ***Apache*** | ***AWS*** | ***Oracle*** |
| **Req. 50.4** | Can be created and be online during mobile download process | Can be created and be online during mobile download process | Can be created and be online during mobile download process | Can be created and be online during mobile download process | Can be created and be online during mobile download process |
| **Std. 60** | Can support HTTPS connections | Can support HTTPS connections (complicated) | Can support HTTPS connections | Can support HTTPS connections (high cost) | Can support HTTPS connections (high cost) |
| **Req. 50.5** | Can be publicly connected to and allow users to download images | Can be publicly connected to and allow users to download images | Can be publicly connected to and allow users to download images | Can be publicly connected to and allow users to download images | Can be publicly connected to and allow users to download images |

Though all web servers listed do not have any specific disadvantages, Nginx will be used as the web server where the website will be hosted as the team is more familiar with Nginx than other aforementioned web servers.

1. **Mobile Download Application**

|  | Design Options | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***Swift*** | ***C++*** | ***Rust*** | ***Python*** |
| ***Req. 50.4*** | Can connect to server using HTTPS | Can connect to server using HTTPS | Can connect to server using HTTPS | Can connect to server using HTTPS |
| ***Req. 50.5*** | Can pull data from the server using HTTPS | Can pull data from the server using HTTPS | Can pull data from the server using HTTPS (Limited) | Can pull data from the server using HTTPS |
| ***Std. 60*** | Can send data following HTTPS | Can send data following HTTPS | Can send data following HTTPS | Can send data following HTTPS |
| ***Std. 40*** | Does not follow C++ coding standards | Follows C++ coding standards | Does not follow C++ coding standards | Does not follow C++ coding standards |

C++ will be used for the Mobile Download Application due to its flexibility and the amount of tools that will be needed for downloads and HTTPS compatibility. We are also using an Android device and they use C++ for their app development so most resources would also use C++.

1. **Image Weakness Evaluation Application**

|  | Design Options | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***Swift*** | ***C++*** | ***Rust*** | ***Python*** |
| ***Req. 50*** | Theoretically possible but not proven | Possible using external libraries | Theoretically possible but not proven | Possible using external libraries |
| ***Req. 50.2*** | Theoretically possible but not proven | Possible using external libraries | Theoretically possible but not proven | Possible using external libraries |
| ***Req. 50.5*** | Allows for image download from URL | Allows for image download from URL | Allows for image download from URL (Limited) | Allows for image download from URL |
| ***Std. 60*** | Can send data following HTTPS | Can send data following HTTPS | Can send data following HTTPS | Can send data following HTTPS |
| ***Std. 40*** | Does not follow C++ coding standards | Follows C++ coding standards | Does not follow C++ coding standards | Does not follow C++ coding standards |
| ***Std. 70*** | Can send and store data in .png format | Can send and store data in .png format | Can send and store data in .png format | Can send and store data in .png format |

C++ will be used on the mobile device to implement the Weakness Evaluation Application. This is because C++ has familiar image evaluation libraries that can be used to find the red circles containing possible weaknesses. Swift and Rust also have these libraries but proof of work with their usage cannot be found. Python achieves the requirements but is not being used because the group is more comfortable with C++.

1. **Data Organization and Display Application**

|  | Design Options | | | |
| --- | --- | --- | --- | --- |
| ***Criteria*** | ***Swift*** | ***C++*** | ***Rust*** | ***Python*** |
| ***Req. 50.3*** | Possible using external libraries | Possible using external libraries | Theoretically possible but not proven | Possible using external libraries |
| ***Std. 40*** | Does not follow C++ coding standards | Follows C++ coding standards | Does not follow C++ coding standards | Does not follow C++ coding standards |
| ***Std. 70*** | Can send and store data in .png format | Can send and store data in .png format | Can send and store data in .png format | Can send and store data in .png format |

C++ will be used to program the Data Organization and Display Application on the mobile device. This is because the group is more comfortable working with C++ than Swift and Python. Rust theoretically could work however no proof of work was found by the group.