

|  |
| --- |
| **Economical Duck White Paper**  **Team 404:**  Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan  CMP311: Professional Project Development & Delivery  2018/19 |

Abstract

This paper is a follow on from the *Team 404 – Project proposal* paper. Connor Duncan, Lewis Goor, Sunny Sherry and Christopher Clark decided to undertake the challenge of implementing the functionality of a Rubber Ducky onto a Raspberry Pi Zero and calling it the Economical Duck. The device was targeted towards university students and would be used as an educational tool. The benefits of this include teaching students the dangers of connecting unknown devices to machines and enhancing programming knowledge.

The team set out over a period of 14 weeks to complete the Economical Duck. The project was split up with Christopher Clark being the Team Leader, Sunny Sherry undertaking the role of Hardware Specialist and Lewis Goor and Connor Duncan being the Software Specialists. Two scripts were created that a user would be able to switch between depending on their programming knowledge – basic knowledge or advanced knowledge, and they were accompanied by an extensive manual detailing how to use each version. The basic script came with its own syntax and functions, thus requiring very little programming knowledge. The advanced script would allow the user to import the Economical Duck Library, let them code in Python, and use the Economical Duck functions where required.

After completion of the Economical Duck, focus group testing was undertaken. Each participant was given the opportunity to test both an Economical Duck and a Rubber Ducky. They would then fill out a survey that would ask to score the following categories out of five: functionality, reliability, usability, efficiency, maintainability and portability. These sections would then be weighed against an importance rating given by the team prior. In the previous stage of planning, it was estimated that the Economical Duck would receive a weighted score of 84 out of 115. Upon completion of the focus group testing, it was found that it scored a weighted score of 88 thus surpassing the team’s expectations. However, the Rubber Ducky collected a total score of 95 after testing. This score surpassed the target set previously in the planning phase meaning that the project was a success.

Contents

[1 Introduction 4](#_Toc6331764)

[1.1 Background 4](#_Toc6331765)

[1.2 Aim 5](#_Toc6331766)

[2 Development 6](#_Toc6331767)

[2.1 Overview of Development 6](#_Toc6331768)

[2.2 Investigation 7](#_Toc6331769)

[2.3 Implementation 9](#_Toc6331770)

[2.3.1 Hardware 9](#_Toc6331771)

[2.3.2 Software 10](#_Toc6331772)

[3 Testing 13](#_Toc6331773)

[3.1 Software Testing 13](#_Toc6331774)

[3.2 Hardware Testing 14](#_Toc6331775)

[3.3 MacOS Testing 15](#_Toc6331776)

[3.4 Focus Group Testing 15](#_Toc6331777)

[4 Discussion 18](#_Toc6331778)

[4.1 General Discussion 18](#_Toc6331779)

[4.2 Countermeasures 19](#_Toc6331780)

[4.3 Conclusions 20](#_Toc6331781)

[4.4 Future Work 20](#_Toc6331782)

[4.5 Call to action 20](#_Toc6331783)

[5 References 22](#_Toc6331784)

[Appendices 23](#_Toc6331785)

[Appendix A – Lookup Tables 23](#_Toc6331786)

[Appendix B – Project Deliverables and Agreement Form 28](#_Toc6331787)

[Appendix C - Minutes 30](#_Toc6331788)

# Introduction

## Background

The brief that was provided by Abertay University was to create a physical penetration-testing device. One of the main reasons that Team 404 decided on creating the device was so that it would be suitable for educational purposes such as allowing lecturers to use the device in their classes.

To gain background knowledge into this field, Team 404 researched devices such as the “Rubber Ducky” by Hak5™ to review the syntax of the product and gain an understanding into what would be expected of the Economical Duck in terms of the commands used and its overall functionality. (Hak5, 2019)

The syntax produced was made to be used on Python and would run on a Raspberry Pi Zero; a small functional computer that would have the capability of acting as though it was a keyboard when plugged into a target machine.

The Rubber Ducky itself cannot be considered a viable option for Abertay University to hand out to Ethical Hacking students as an educational tool due to the cost being extremely high. Whereas Team 404 look to create a much cheaper alternative version.

As seen on the Hak5™ website, the price per unit of a Rubber Ducky is $44.99, which is approximately £35. If a class were to have two people to one device, the cost would still range from £350 to £550 before shipping and import costs, and for a device that would only be used a couple of times per academic year, it would not be considered efficient for the Universities’ budget to stretch to this amount. However, a Raspberry Pi Zero Kit works out at approximately £22.50 per unit, which is almost half the cost of a Rubber Ducky. As well as this, the Raspberry Pi Zero could be used multiple times and gives more flexibility for use in different projects and modules due to its extensive functionality.

The Economical Duck has two modes for the user to choose between when giving the Raspberry Pi Zero commands to execute on the target PC – a basic mode and an advanced mode. The basic mode is for programming beginners who do not have an in-depth knowledge into Python, whereas the advanced mode is for users who have a good understanding of programming as the functions used are in a Python Library, which grants the user more flexibility with the code. For example, below is a snippet of code that types “echo Hello World” onto the target machine;

**Basic Mode -** TYPE echo Hello World

**Advanced Mode -** eco.type("echo Hello World")

## Aim

The main purpose of this document is to document and provide proof to the team’s client, Dr. Ethan Bayne, regarding the development, and creation of the Economical Duck product.

Team 404 will supply this proof by delivering a list of deliverables and agreed requirements to Ethan Bayne within the deadline of 23rd April.

These deliverables consist of:

* The white paper which must contain information regarding investigation and research, code design in the form of pseudocode, development documentation, evidence of testing and focus group results.
* The developed source code that the Economical Duck uses which contains both the hardware and software interaction.
* The Economical Duck manuals that must clearly convey how to use the Economical Duck and all of its features.
* A fully working Raspberry Pi Zero containing the Economical Duck image.

Regarding these deliverables, the agreed requirements for them are as follows:

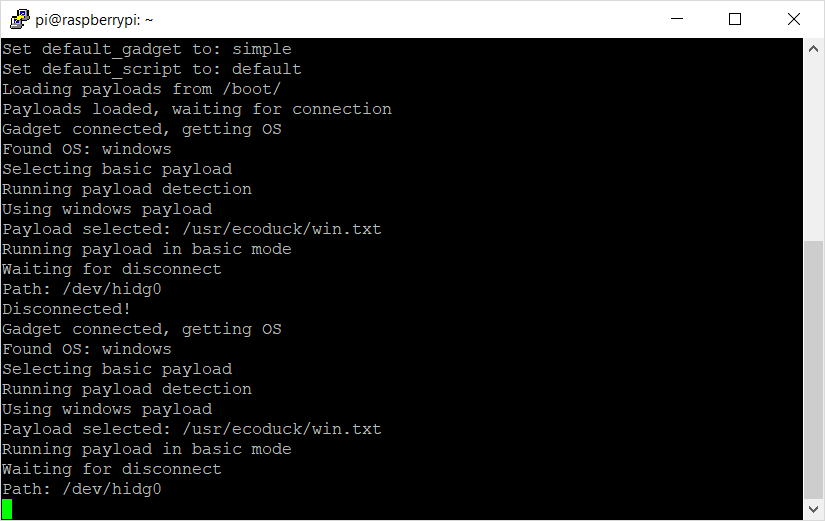
* The Economical Duck manuals must be easily understood by Ethical Hacking Students
* Cost of Economical Duck must be significantly less than Rubber Ducky from Hak5™.
* Device must be able to output specified keypresses that are defined by a file created by an end user.
* Data must be able to be transferred to the mass storage located on the Economical Duck
* End user must be able to access data previously transferred to the mass storage located on the Economical Duck
* Weighted quality score of the product should be similar to Rubber Ducky in focus group testing.

# Development

## Overview of Development

In order to undertake the project at hand, Team 404 split into two separate groups – Software Development and Hardware Development. The Software Development team consisted of Lewis Goor, Connor Duncan and Christopher Clark, while the Hardware Development team was made up of Sunny Sherry. Lewis, Connor and Christopher all had prior experience with software development, yet minimal experience with hardware development. However, Sunny did have prior experience dealing with hardware devices and therefore took it upon himself to undertake the role as Hardware Specialist. Hardware development consisted of setting up and configuring the Raspberry Pi Zero to be prepared for the software.

During development the Software Development team debated over how to create the EDS. The debate had two sides – creating their own language or creating a Python Library. After the debate, the team decided that by creating their own language, they will have created all of the functions for the Python Library. The only difference would be how the two were structured. Based on this knowledge Team 404 decided to create their own language and the Python Library.

During the software development, Connor Duncan developed the function that would create the HID packets to send to the hardware device, Lewis Goor was responsible for creating the individual functions that the end user would make use of, and Christopher Clark created the functions that would interpret the lines of the payload file and determine what commands the end user wants to perform. Christopher also created the repeat function, with the assistance of Sunny Sherry. In addition to this, Lewis and Connor created two lookup tables that mapped each keypress to a scan code. These lookup tables were then widely used by the function that created the HID packets.

During the hardware development, Sunny configured the USB gadget driver on the Raspberry Pi Zero to be treated as a keyboard, mass storage device and network adapter, then developed plug-in detection to determine when to activate the device. For networking to work reliably OS detection was developed to create the correct networking gadget for each OS. Once all of the features Team 404 wanted were developed, a script was created to standardise the installation of the Economical Duck. This included configuring the device to run payloads from boot, and a configuration file to change how the payloads are selected and how the device is setup before executing a payload. Finally, a debug screen was created to give a verbose output of what the Economical Duck is doing (see Figure 1).

Christopher Clark took it upon himself to learn how to use GitHub and teach the other members how to branch and merge properly. Once this was done all documentation related to the Economical Duck itself (such as code and manuals) was uploaded to GitHub and worked on from there by branching off the repository and merging again to update the master branch. By doing this, the team gains the ability to all work on the same file and handle file collisions in a simple and straightforward manner, as well as receiving good version control because of the ability to roll back to previous updates if an unwanted change took place to a file that was updated.

Figure 1 - Economical Duck debugger

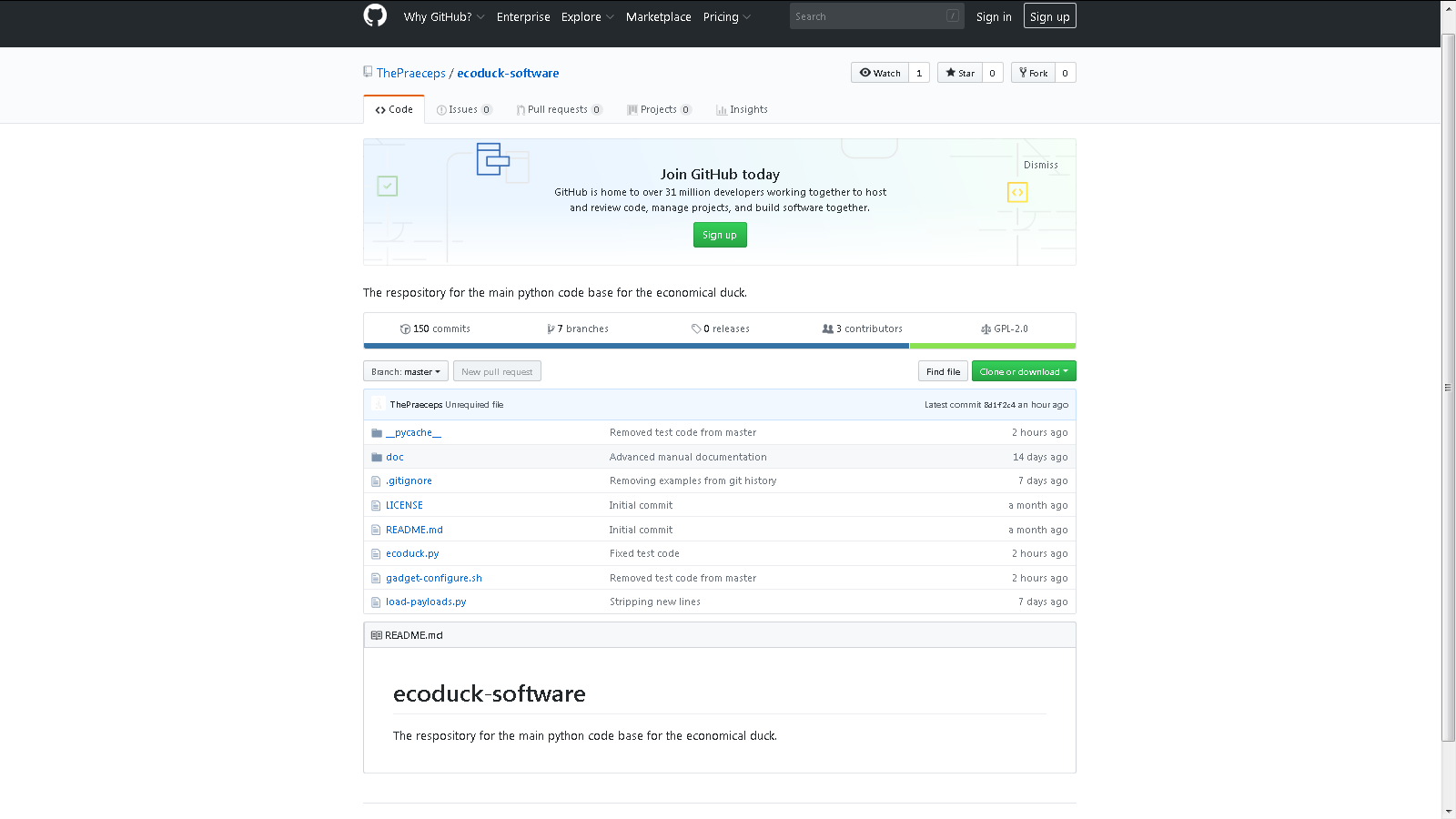


Figure 2 - Economical Duck GitHub Repository

To fulfil the agreed requirements, the Economical Duck manual deliverable was written up at this stage by the whole team and included how to use the product, how to write a payload using Basic and Advanced EDS, and how to configure the Economical Duck to give it certain behaviour.

Focus group testing took place as the final step once the Economical Duck and product manuals were created, and the results were recorded in accordance to the ISO 9126 software quality characteristics (see Focus Group under Discussion for more information) to determine the product’s success.

## Investigation

To understand what functionality Team 404 wanted implemented into the Economical Duck, the syntax of the Rubber Ducky by Hak5™ was reviewed to determine what sort of commands and services would be required of the Economical Duck. (Cheerla, 2017)  
These commands were then narrowed down to the following unique commands that the Economical Duck will use as its ‘syntax’;

**PRESS** – Any text after this command will be treated as unique keypress combinations to create HID packets (e.g. ‘PRESS LCTRL+LALT+delete’ would press the Left Control, Left Alt and Delete key down in tandem).

**TYPE** – All text entered after this command is treated as individual characters which would be sent as HID packets to the target PC (e.g. ‘TYPE Hello World’ would type ‘Hello World’ on the target PC).

**DELAY** – Economical Duck would pause for X seconds before continuing to execute more code (e.g. ‘DELAY 3’ would cause the Economical Duck to wait 3 seconds before continuing to read from the file).

**CMT** – Text inserted after this command would be treated as a comment which will be ignored by the Economical Duck script (e.g. ‘CMT This is a comment’ will cause the script to ignore the words “This is a comment” and any other words on the line). Originally, it was discussed as the command ‘COM’, however Team 404 decided that these three letters could be misinterpreted as ‘command’ and was therefore changed to ‘CMT’ instead.

**REPEAT** – This command repeats all commands after it X times until it reaches an ‘END’ string within the payload (see example below which repeats ‘Hello World’ four times). At first Team 404 only thought to support singular ‘REPEAT’ commands, however after further research it was deemed possible to create recursive nested ‘REPEAT’ commands to allow for yet more automation and functionality.

The following command will be the ‘Basic’ version of the Economical Duck Script (EDS):

REPEAT 4   
TYPE Hello World   
END

This language will be intended for users who do not have much experience coding in Python. An ‘advanced’ form of functionality will be available for users who have a developed understanding of programming. In the advanced version users can import the EDS commands as a Python Library, granting users more flexibility with the code allowing it to perform in different ways which is not normally possible using the ‘Basic’ version.

From here, Sunny had to research how to configure the Raspberry Pi Zero and allow these features to function on a hardware level. This involved figuring out how to set up the device to act as a USB composite gadget that will combine the functionality of a USB Ethernet adapter, a mass storage device, and a human interface device (HID). It was found that the “LibComposite” kernel module would best allow for configuring the Raspberry Pi Zero to do this by interacting with files on a pseudo-file system. (Tobi, 2016)

The team also had to figure out how to detect when the device has connected to a computer and ready to send the payload. After some research, it was found that PCs send a HID output report containing the current state of the toggle keys on the computer. This can then be used to synchronize the status of the Caps Lock LED with the other input devices connected to the PC. This allowed the team to capture this packet from the Pi and use it to determine if the keyboard was able to update by sending a “capslock” HID packet to the target PC, then seeing if a response was sent back.

Once testing began on the networking part of the device it was discovered that the operating system of the target would need to be identified to reliably create a network connection. Research was undertaken into determining if it was possible to reliably fingerprint what operating system a USB guest device was being plugged into. A project was then found for a Raspberry Pi Zero which used a kernel modification to output the USB connection packets to a log which could be used to determine what operating system was connected. (wismna, 2017). Ultimately it was decided to develop this functionality so that the device could use different payloads on different operating systems as it is unlikely that a payload would work across multiple platforms.

Once the functionality of the device was mature, Sunny decided that an installation script was required to ensure that every device could be configured easily and reliably act in the same way. This would be made in bash and a fresh installation setup of Raspbian would be configured to work with the Economical Duck Software.

Lastly, it was decided that the user should be able to configure the device easily from a Windows PC, and have the payloads execute without any interaction from the user. Research into this was conducted, and it was found that the Raspberry Pi Zero had a boot partition that was formatted as fat32 and therefore could be accessed from a Windows computer. It was decided to use this to allow users to put a payload in it, which would be copied across to the main working directory on boot. An “init.d” script was found to be the best way to have the Economical Duck software execute on boot.

## Implementation

The code for the Economical Duck was developed in two separate areas (hardware and software), then merged into one master file later throughout the course of the project.

### Hardware

**Gadgets**

The first gadget to be implemented was the one that provided the HID functionality to act as a keyboard. This was so the team could demonstrate functionality like the Rubber Ducky at the project pitch. Once this functionality was proved to be reliably working, the team moved on to making the mass storage and Ethernet gadgets function. Implementing this consisted of making a bash script to interact with the ‘LibComposite’ file system to create the gadgets.

As UNIX and Windows systems required different ethernet adapters, this resulted in two network interfaces on the Raspberry Pi Zero. These were combined using ‘OpenVSwitch’ to make them a single interface to simplify networking with the target device. This was used as the interface for the DHCP server to listen on and allows the Raspberry Pi Zero to have a consistent IP address regardless of the target’s operating system.

**Insertion Detection**

Next a feature was developed to run a script when the device is connected to the target machine. As found in the investigation, the output report from a PC when the caps lock toggle state changes can be used to detect when the device is inserted. Initially this was implemented in bash by calling the ‘cat’ command on the device endpoint of the HID gadget. This result was then piped to run the ‘echo’ command which would terminate the ‘cat’ command once a HID packet is received from the target computer. This meant the device had been successfully connected and could continue execution of the payload.

Once the project had been further developed this functionality was then moved over to Python. This was done using Python’s built in ‘open’ function. This would attempt to actively test the device by sending a HID packet with the caps lock key toggled and seeing if a response was received. This response would then be analysed to see if the caps lock key had been enabled or disabled, to ensure it was disabled for the rest of the payload.

This same function was used to test if the device had been removed. This was implemented by adding a timeout to the function which caused the test function to return false if the script did not receive a response in the fixed amount of time.

**OS Fingerprinting**

The script in the ‘HackPi’ repository for modifying the kernel module which provides USB gadget functionality did not function as the ‘rpi-source’ command is outdated and no longer maintained. As such, the team had to develop a custom script in order to automatically patch the kernel. This cloned the kernel from the Raspberry Pi Zero repository (Raspberry Pi Foundation, 2019). The script then applied the ‘gadget.patch’ patch to the dwc2 source file and built the required tools to make the module, which was then compiled.

After the modified module had been loaded, it was found to work as expected with the ‘fingerprint.sh’ script provided in the module. This was modified to enable the Windows and Linux detection functionality.

Once the software was more developed, this functionality was recreated in Python by using the same logic found in the “fingerprint.sh” script.

**Installation**

Once the project was sufficiently mature, the team decided that an installation script would be required to enable standardisation of the installation process to increase the reliability of the devices that were configured. This then updated the Raspberry Pi Zero and installed the required tools for the project which were then configured as needed. The script then had to reboot to allow the updated kernel to be loaded. This meant that an additional installation service was then created to allow the script to continue running after the reboot. This script would then patch the kernel and set up the ‘ecoduck-software’ repository in a common path (/usr/ecoduck), and then create a service to load the payloads and wait for connections on boot.

### Software

**Interpreter**

This phase involved reading a text file containing an Economical Duck Script and executing it in the form of keypresses by using the unique commands and dictionary tables created.

**Lookup tables**

This phase involved the creation of dictionary tables in Python that contain the hexadecimal representation of the binary data related to every key that is available on a keyboard. Thus, it was one of the first pieces of development as all code produced after these dictionary tables would be using the resources created in this phase.

Originally, the plan was to have all the hexadecimal information contained within one dictionary table, however it was then decided that it would be more logical to create two separate tables to determine whether the Shift key was being used in conjunction with other keypresses or not.

Below is an example of two lookup tables that are used to determine the hex representation of the keys that are to be pressed (LookUpScanCode), and whether it would require a shift flag to be set for the capital letter (LookUpShiftLayer).

LookUpScanCode = {   
 “a” :0x04  
 “b” :0x05

...  
}

LookUpShiftLayer = {   
 “A” :“a”  
 “B” :“b”

...  
}

See ‘Appendix A – Lookup Tables’ to view the full table.

**createHIDpacket()**

The createHIDpacket function is used to generate the HID packet that will be sent to the target computer. It takes in the keypresses that will be used and any modifiers that have been set. The first loop in this function is called “for key in KeyList”. This runs through all of the keypresses set and works out if they are contained within the LookUpScanCode dictionary, or the LookUpShiftLayer dictionary. If they are contained in the latter, then the “LSHIFT” modifier is set to true. The hex values gained from the dictionaries are then stored in an array called “ScanCodes”.

The following loop “for modifier, state in ModifierList” runs through all of the modifiers and determines if they have been set or not. This then populates the first byte of the HID packet. If the user has set the debug level to greater than three, then they will be able to see all the modifiers and their states when being set. This byte is then converted from a string to an integer and then converted into a binary literal.

When a HID packet is created, the second byte is always set to null. Therefore, a null byte is created to append to the HID packet.

After the first two bytes have been added to the HID packet, the remaining keypresses hex values are encoded and appended on the end. Once this stage has completed, if the HID packet is less than eight bytes, then the remainder of the packet is padded with null bytes.

**sendHIDPacket()**

The sendHIDPacket function is used to send the HID packet to the target computer. It takes a binary string and writes it to the device endpoint of the HID gadget, which will send it to the target PC.

Figure 3 shows a visual representation of the code’s logic flow regarding the keypress injector:

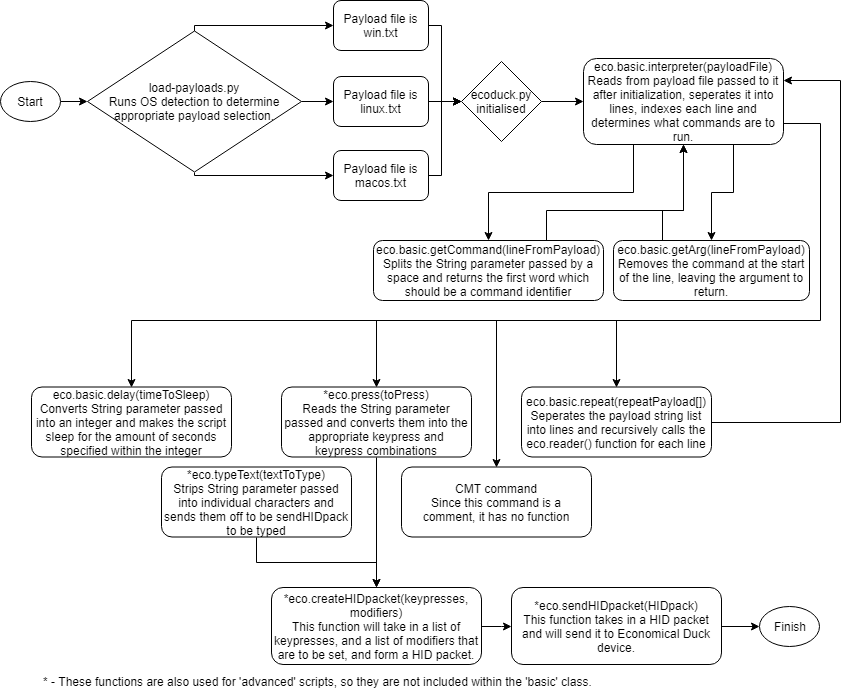


Figure 3 - Visual representation of the Economical Duck's keypress injector code

# Testing

## Software Testing

Software testing occurred at several different stages throughout the development of the Economical Duck. This included testing each function created for the EDS and ensuring that the hexadecimal representations of each keypress in the dictionaries were all accurate and up-to-date.

To carry out the testing of the keypresses, each scan code was individually sent to the target computer. The hexadecimal representation of each keypress’ accuracy would depend on the outcome on the target PC. Each keypress would be used relative to its purpose. For example, the ‘backspace’ key would not be tested if no characters were present to be deleted. This meant that for certain keypresses, specific requirements were needed.

To carry out the testing on the HID packet creation function, certain characters were hard coded and sent to the function. If the function was able to successfully convert the characters into HID packets, then the function worked. These values would then be manually checked against the calculated correct values to see if they matched. A wide range of inputs were given, including inputs with different modifiers set (‘RSHIFT’, ‘LALT’ etc.)

To carry out testing of the HID packet send function, human crafted packets were sent using the function, and the result on the target PC checked. This found that the function was sending the packets correctly with packets including both modifiers and regular scan codes.

In order to test the ‘delay’ function, the function was called with a parameter of 2 seconds. If the program were successfully delayed for 2 seconds before it would continue running, then it would confirm that the function worked as intended. For example, the delay would be set to 2 seconds, with ‘hello world’ to be displayed after this delay.

In order to test the ‘type’ function, the function was called with a parameter of a string of keypresses. These keypresses would then be sent individually to the ‘createHIDpacket’ function. If the program were able to interpret the keypresses and display them on the target PC, then the function would be deemed a success.

To test the ‘press’ function, a string was passed into the function as a parameter. The string that was passed into this function is split up depending on where the ‘+’ symbol is located. The reason behind splitting the string up was to determine what modifiers were specified and what keys were specified. During testing the string ‘LALT+RCTRL+delete’ was entered. ‘LALT’ and ‘LCTRL’ would be split and added to a list called ‘ModifierList’, and ‘delete’ would be added to a keypress list. These parameters were then sent to the ‘createHIDpacket’ function. To test that this function worked, the example above was sent to the target and as a result, the task manager was started, proving the function to work successfully.

**Issues Resolved**

During the testing of the ‘eco.basic.interpreter(listOfStrings)’ function, some unexpected output occurred when using the ‘PRESS’ command.   
A payload which reads ‘PRESS Enter’ should cause the interpreter to pass the string ‘Enter’ to the ‘eco.basic.press(str)’ function, however after not getting the output desired, debugging the code to display the raw output in terminal proved that the press function was being passed the string ‘nter’.

This was due to the way in which the interpreter function was developed, as it was set up to strip the characters ‘PRESS’ from the left side of the string, however this meant that a string such as ‘PRESS Enter’ would strip ‘PRESS E’.

Below is a snippet of the original code:

elif eco.commandFinder(stripped) == "PRESS":

currentLine = currentLine + stripped.lstrip("PRESS ")

eco.press(currentLine)

This was changed so that it calls a function called ‘getArg()’ to acquire the argument which needs to be passed to the press function.  
Rather than stripping characters from the string, ‘getArg()’ finds the location of the first space within the string which acts as the separator and sets the starting location of the string one character after the space.

Below is a snippet of the patched code:

elif current\_command == "PRESS":

eco.press(current\_arg)

def getArg(line):

arg = line[line.find(" ")+1:];

return arg

This code outputs in an expected fashion (outputs ‘Enter’ instead of ‘nter’) and no longer risks stripping certain characters within an argument.

Further testing of the software discovered that there was a conflict with using the ‘+’ symbol within the ‘press’ function. During time of testing there was no way for the user to enter the ‘+’ symbol without it being interpreted as a splitter for the strings in the function. To combat this, the word ‘plus’ was entered into the lookup table. By doing this, it allows the user to type the word ‘plus’, and it will be interpreted as the literal value for the plus symbol.

## Hardware Testing

Once the gadgets were configured, testing was done to make sure that the device was functioning as expected. The HID gadget was tested by creating a bash script to send manually created HID packets that would type “Hello World!” This was tested in a text editor on each operating system to ensure that the device was acting as anticipated. The USB Mass storage was tested by transferring a 100MB file of random data to the device, then transferring it back on each operating system. The hash of this file was checked before and after to see if any corruption had occurred. 5 run-throughs were then completed on each of the operating systems, in which the device was found to be working successfully.

The networking required the most significant testing of all the gadgets. It was not possible to create a single gadget which provided both UNIX and Windows Ethernet USB gadgets. As such, operating system detection was then added to select the correct gadget with just a single type of Ethernet gadget. When this was done, all operating systems identified the gadget and configured it automatically. Once a DHCP server was added to the Economical Duck, it was also checked that the target machine would also attempt to lease an IP address from the server running on it. This was found to be successful with no issues. Once networking was found to be set up correctly, ‘iPerf’, a networking performance tester, was ran on both Windows and Linux machines to see if the gadgets functioned properly. It was found that network had high speeds of over 80MB/s with no packet loss, meaning the networking was working correctly.

The installation script required a lot of testing to get to a working state. Initially there were issues with accessing the internet after the reboot. By switching to an ‘init.d’ instead of an ‘rc.local’ script, this fixed the issue by allowing requirements for the script to be defined. However, the connection would still often fail with a ‘gnutls\_handshake failed’ error, despite being able to ping a working connection and occasionally still successfully cloning other repositories. This was fixed by cloning over SSH instead of HTTPS, which bypassed that part of the git client.

## MacOS Testing

During the creation of the Economical Duck, testing was undertaken to get the EDS working on a Mac device. Due to the Mac keyboard layout being different, plans were made to create a separate Lookup table that contained all of the scan codes for the Mac keys.

For the Economical Duck to work, a HID packet containing the scan code of the caps lock key is sent to the target PC. If the target replies with the state of the caps lock toggle key, then the Raspberry Pi Zero has successfully connected, and the payload script can then be executed. However, when testing on a Mac, it was found that the Economical Duck could not detect when it was plugged in. Through testing it was found that when a caps lock HID packet was sent, the Pi would not receive a response. This presented the problem of the Economical Duck not being able to tell if it had successfully connected to the Mac or not.

Debugging then occurred to find out why this was, and a normal keyboard was connected to the Mac to do so. It was found that if the connected keyboard set its caps lock key to true, then it could type in capitals. However, the Mac’s internal keyboard would not type in capitals if the external keyboards caps lock key were set to true. This revealed that Mac’s treat external keyboards differently to Windows, in that Mac maintains an individual state of the toggle keys on each keyboard and does not communicate these states to the keyboards. Due to this problem, Mac was found not to be compatible with the Economical Duck in its current form.

## Focus Group Testing

Focus group testing took place to ensure the quality of the product created was up to the required standard and met the user’s needs to learn and use the product effectively.

Volunteers for the focus group testing were taken in pairs and given 15 minutes to perform a task. This task involved opening the command prompt using the run window and type “hello world” with the device. One volunteer would start with the Rubber Ducky, and the other would start with the Economical Duck. Once their task is completed, the volunteers swap devices and try the same task again. This was to try and eliminate bias created by using one device and transitioning over to the next. This could be due to them having a better understanding of using this type of device after creating the script on the first device, or due to the user getting confused by the differences between the two devices.

After finishing with each device, the volunteers were given a survey to fill out which contained the six categories from the ISO 9126 software quality characteristics example (see list below). These categories were each put onto a survey to be rated from 1 to 5, with 1 being ‘Poor’ and 5 being ‘Great’:

**Functionality** – This covers the number of features built into the device (operating system compatibility, additional features, etc.).  
**Reliability** – This regards how users felt about the reliability of the device (hardware failure, software doesn't do what's expected etc.).  
**Usability** – How easy it was to learn how to use the device, as well as how simple it was to use.  
**Efficiency** – How quickly it boots, how fast it does what users want.  
**Maintainability** – How easy it is to debug and how easy it is to add features to it.  
**Portability** – Physical size, ability to be used on other operating systems, etc.

Below is the comparison of speculative and actual results after performing the focus group test:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product Quality | Importance Rating | **Economical Duck**  Quality Score | Weighted Score | **Rubber Ducky**  Quality Score | Weighted Score |
| **Functionality** | 4 | 4 | 16 | 3 | 12 |
| **Reliability** | 4 | 3 | 12 | 4 | 16 |
| **Usability** | 4 | 4 | 16 | 4 | 16 |
| **Efficiency** | 4 | 3 | 12 | 5 | 20 |
| **Maintainability** | 4 | 4 | 16 | 1 | 4 |
| **Portability** | 3 | 4 | 12 | 5 | 15 |
| **Overall Totals** |  | | 84 |  | 83 |

Table 1 - Speculative results from project planning and prototyping phase

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product Quality | Importance Rating | **Economical Duck**  Quality Score | Weighted Score | **Rubber Ducky**  Quality Score | Weighted Score |
| **Functionality** | 4 | 4 | 16 | 4 | 16 |
| **Reliability** | 4 | 5 | 20 | 4 | 16 |
| **Usability** | 4 | 3 | 12 | 4 | 16 |
| **Efficiency** | 4 | 3 | 12 | 5 | 20 |
| **Maintainability** | 4 | 4 | 16 | 3 | 12 |
| **Portability** | 3 | 4 | 12 | 5 | 15 |
| **Overall Totals** |  | | 88 |  | 95 |

Table 2 - Results from focus group testing

As seen in the focus testing results in Table 2, and comparing them against Table 1, the Economical Duck has surpassed the projected results from the planning phase of this project (evidenced by the higher Weighted Score total).

The speculative results were an estimation of what the team thought each device could achieve. This was an educated guess based on the team’s prior experience with the Rubber Ducky and what Team 404 believed they could create based on their planning. This explains why the Rubber Ducky was projected to have a lower score than the focus group showed.

# Discussion

## General Discussion

Overall Team 404 feel like the investigation went exceptionally well because they felt as though they had a clear understanding of what needed to be accomplished and how to go about doing it. They felt it went well as throughout the development process they had the information required to create the product.

In hindsight the team would carry out a more in-depth investigation on how the HID protocol interacts with different platforms. By doing this it would have allowed them to find the limitation contained within the Mac OS and potentially reduced development time by not needing to carry out testing on Mac devices, or alternatively finding a solution to the problem.

Additionally, the team feels like they could have carried out further research on how to structure manuals and effectively communicate technical information in them. This would have resulted in the writing progress being much simpler and quicker rather than only relying on feedback from user testing.

During the implementation stage Team 404 believe that they maintained a good and consistent pace throughout the creation of the Economical Duck. This was proven by the majority of the requirements being fulfilled by week 4.

The team used Slack to communicate any concerns that they may have had throughout development, and coupled this with weekly meetings to ensure that each team member understood what was expected of them. In addition to this the team met with the client every week to give updates on the progress that had been made.

Overall the teamwork within the group was very strong and the team maintained a positive frame of mind throughout. The team felt like they were well supported by their fellow teammates and if they had any concerns then they could effectively voice them.  
Some problems arose within the team due to a misunderstanding of certain Python functions, which caused problems when testing the code. This could have been avoided by having a more in depth understanding of how Python works and more research into any unknown functions.

The team believes that they carried out thorough testing on both the hardware and software components of the Economical Duck. The testing stage covered all the features and was able to find several bugs that would have otherwise hampered the user experience. However, the testing stage did not find all bugs and some did still arise during the focus group testing stage. This could have been avoided by testing more types of unexpected data.

Team 404 feels that the focus group testing was a success and gave sufficiently reliable results. While carrying out the testing, a satisfactory sample size was used given the timeframe and limited pool of participants. The majority of users enjoyed using the product and quickly understood how to use the device. However, despite this, the Rubber Ducky still achieved a higher weighted quality score than the Economical Duck. While the team believes that the difference in cost between the two devices is enough to account for the difference in scores, with more thorough focus group testing the team predicts that the difference in scores would be closer.

The focus group testing applicants had 15 minutes with each device, which is not necessarily enough time to demonstrate all of the features that the Economical Duck possesses outside of a typical keypress injector like the Rubber Ducky, such as the mass storage device and network adapter capabilities. This is due to the time and attention constraints which affected how much time the team allocated for each applicant to spend with the device. If the participants had more time with the device, the team believe they would rate the functionality of the Economical Duck much higher than the Rubber Ducky, opposed to matching it which current testing showed.

In addition, the weighting of the scores was decided based upon its effectiveness as a penetration testing device. As this device is mainly aimed at educational users, a large amount of the areas where the Rubber Ducky scores higher, such as efficiency and portability, are less relevant. There is much less pressure in an educational environment than an on-site penetration test as there is no need to hide the use of the device. As such, how long it takes to boot, and the size of the device, is far less relevant. If the team were to adjust these scores towards purely education, then the scores would be much closer.

Team 404 believes that with these adjustments, the Economical Duck would match or beat the quality scores of the Rubber Ducky.

In addition, how many applicants Team 404 could gather for the test was limited by the time of year. This decreases Team 404’s confidence intervals in the values gathered by the focus group testing. This was because a large number of ideal candidates were unable to give the time required for the testing due to other university commitments. This was unavoidable however as it was necessary for the focus group testing to occur on a mature version of the product, and the team believes that the results are reliable enough to show that the complete product meets the user’s needs.

Overall, all of the standards set out in the requirements stage were met. Focus group testing proved that the manuals were easy to follow, concise and gave all of the required information to carry out their task. The cost remained significantly lower than the Rubber Ducky and worked effectively as a keypress injector and a USB mass storage device as shown in the testing. Data contained within the mass storage device can be accessed by the Economical Duck when required. Finally, the weighted quality score exceeded the expectations and was within the estimated ±10% boundary, thus deeming the project a success.

## Countermeasures

There are not many countermeasures that can be used to prevent the Economical Duck from carrying out its payload.

Team 404 speculates that an advanced intrusion detection system that detects the speed that a keyboard is being typed at as well as the time consistency between each keypress could potentially discover that the keyboard in question is not being operated by a human and blocks or subtly modify input from it. This will effectively cease all functionality of the keypress injector on the Economical Duck, thus preventing the script from running.

Additionally, USB devices allowed on a computer can be white listed to prevent unknown devices being plugged into the computer. This would disable the device, unless it was modified to match the USB id of another device on the computer.

It is also possible to configure computers to not automatically connect to new networking adapters. This would mean the computer would not make a DHCP request to the new USB ethernet adapter and therefore the networking functionality of the Economical Duck would not function as expected.

## Conclusions

After several weeks of development, the Economical Duck was completed. It successfully allowed the full functionality of a Rubber Ducky and still allowed the user to use the device as a Raspberry Pi Zero, all while being significantly cheaper than a Rubber Ducky, making it an affordable alternative.

This was shown by means of focus group testing. A focus group was created to test the performance of the Economical Duck against a Rubber Ducky, in which students were given both devices to test and a survey to answer at the end to see the outcome of certain standards such as reliability, usability and flexibility. From the results, it was shown that the Economical Duck narrowly lost out to the Rubber Ducky. However, Team 404 still see this as a positive result due to the device performing better than anticipated, when compared to the previous scores as seen in Section 4.2.

By using this product, it would allow the client to supply physical penetration devices to their students at a fraction of what the normal cost would be. This means that they can then use these devices for educational purposes to teach students the dangers of connecting an unknown device into their computer.  
Due to significant documentation being included, and the Economical Ducks code being open source, students will also be able to take it upon themselves to develop their own projects and add-ons for the device.

## Future Work

If the team had been allocated more time to work on the Economical Duck, then they would have looked into the following:

**Localization Detection**

This would have involved analysing the keyboard entered (e.g. American keyboard, British keyboard etc.) and map the keys and scan codes accordingly. Whereas currently the Economical Duck makes the assumption that the keyboard layout will be set to British.

**Support for macOS**

Getting the Economical Duck to run on a Mac’s operating system was a large part of the testing that was undertaken. As previously mentioned, it was proven not possible at this current point in time. However, had the team been given more time, they would have continued their work into searching for a way for the keyboard to successfully recognise when it has been connected to a macOS device.

**More robust configuration settings**

Currently, the Economical Duck’s configuration settings are prone to being affected by bad user input which can result in crashing the device. If Team 404 had more time to work with, it would be possible to create a less crash prone version which falls back to hard-coded defaults should user input settings be invalid.

## Call to action

For free training on how to use the device, as well as any additional information, please contact Team 404 using the details below:

Christopher Clark 1603282@uad.ac.uk

Sunny Sherry 1600092@uad.ac.uk

Connor Duncan 1602893@uad.ac.uk

Lewis Goor 1600580@uad.ac.uk

# References

Cheerla, K. (2017). *Duckyscript*. Retrieved March 19, 2019, from https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Duckyscript

Hak5. (2019). *Rubber Ducky*. Retrieved from https://shop.hak5.org/products/usb-rubber-ducky-deluxe

Raspberry Pi Foundation. (2019). *raspberrypi/linux*. Retrieved 03 29, 2019, from https://github.com/raspberrypi/linux

Tobi, A. B. (2016). *COMPOSITE USB GADGETS ON THE RASPBERRY PI ZERO*. Retrieved 01 21, 2019, from https://www.isticktoit.net/?p=1383

wismna. (2017). *HackPi*. Retrieved 01 21, 2019, from https://github.com/wismna/HackPi

# Appendices

## Appendix A – Lookup Tables

|  |
| --- |
| LookUpScanCode = { |
|  |

|  |
| --- |
| "a":"\x04", |
|  |

|  |
| --- |
| "b":"\x05", |
|  |

|  |
| --- |
| "c":"\x06", |
|  |

|  |
| --- |
| "d":"\x07", |
|  |

|  |
| --- |
| "e":"\x08", |
|  |

|  |
| --- |
| "f":"\x09", |
|  |

|  |
| --- |
| "g":"\x0a", |
|  |

|  |
| --- |
| "h":"\x0b", |
|  |

|  |
| --- |
| "i":"\x0c", |
|  |

|  |
| --- |
| "j":"\x0d", |
|  |

|  |
| --- |
| "k":"\x0e", |
|  |

|  |
| --- |
| "l":"\x0f", |
|  |

|  |
| --- |
| "m":"\x10", |
|  |

|  |
| --- |
| "n":"\x11", |
|  |

|  |
| --- |
| "o":"\x12", |
|  |

|  |
| --- |
| "p":"\x13", |
|  |

|  |
| --- |
| "q":"\x14", |
|  |

|  |
| --- |
| "r":"\x15", |
|  |

|  |
| --- |
| "s":"\x16", |
|  |

|  |
| --- |
| "t":"\x17", |
|  |

|  |
| --- |
| "u":"\x18", |
|  |

|  |
| --- |
| "v":"\x19", |
|  |

|  |
| --- |
| "w":"\x1a", |
|  |

|  |
| --- |
| "x":"\x1b", |
|  |

|  |
| --- |
| "y":"\x1c", |
|  |

|  |
| --- |
| "z":"\x1d", |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "f1":"\x3a", |
|  |

|  |
| --- |
| "f2":"\x3b", |
|  |

|  |
| --- |
| "f3":"\x3c", |
|  |

|  |
| --- |
| "f4":"\x3d", |
|  |

|  |
| --- |
| "f5":"\x3e", |
|  |

|  |
| --- |
| "f6":"\x3f", |
|  |

|  |
| --- |
| "f7":"\x40", |
|  |

|  |
| --- |
| "f8":"\x41", |
|  |

|  |
| --- |
| "f9":"\x42", |
|  |

|  |
| --- |
| "f10":"\x43", |
|  |

|  |
| --- |
| "f11":"\x44", |
|  |

|  |
| --- |
| "f12":"\x45", |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "1":"\x1e", |
|  |

|  |
| --- |
| "2":"\x1f", |
|  |

|  |
| --- |
| "3":"\x20", |
|  |

|  |
| --- |
| "4":"\x21", |
|  |

|  |
| --- |
| "5":"\x22", |
|  |

|  |
| --- |
| "6":"\x23", |
|  |

|  |
| --- |
| "7":"\x24", |
|  |

|  |
| --- |
| "8":"\x25", |
|  |

|  |
| --- |
| "9":"\x26", |
|  |

|  |
| --- |
| "0":"\x27", |
|  |

|  |
| --- |
| "enter":"\x28", |
|  |

|  |
| --- |
| "esc":"\x29", |
|  |

|  |
| --- |
| "backspace":"\x2a", |
|  |

|  |
| --- |
| "tab":"\x2b", |
|  |

|  |
| --- |
| "space":"\x2c", |
|  |

|  |
| --- |
| " ":"\x2c", |
|  |

|  |
| --- |
| "-":"\x2d", |
|  |

|  |
| --- |
| "=":"\x2e", |
|  |

|  |
| --- |
| "[":"\x2f", |
|  |

|  |
| --- |
| "]":"\x30", |
|  |

|  |
| --- |
| "\\":"\x31", |
|  |

|  |
| --- |
| "#":"\x32", |
|  |

|  |
| --- |
| ";":"\x33", |
|  |

|  |
| --- |
| "'":"\x34", |
|  |

|  |
| --- |
| "`":"\x35", |
|  |

|  |
| --- |
| ",":"\x36", |
|  |

|  |
| --- |
| ".":"\x37", |
|  |

|  |
| --- |
| "/":"\x38", |
|  |

|  |
| --- |
| "caps\_lock":"\x39", |
|  |

|  |
| --- |
| "print\_screen":"\x46", |
|  |

|  |
| --- |
| "scroll\_lock":"\x47", |
|  |

|  |
| --- |
| "pause":"\x48", |
|  |

|  |
| --- |
| "insert":"\x49", |
|  |

|  |
| --- |
| "home":"\x4a", |
|  |

|  |
| --- |
| "page\_up":"\x4b", |
|  |

|  |
| --- |
| "delete":"\x4c", |
|  |

|  |
| --- |
| "end":"\x4d", |
|  |

|  |
| --- |
| "page\_down":"\x4e", |
|  |

|  |
| --- |
| "right\_arrow":"\x4f", |
|  |

|  |
| --- |
| "left\_arrow":"\x50", |
|  |

|  |
| --- |
| "down\_arrow":"\x51", |
|  |

|  |
| --- |
| "up\_arrow":"\x52", |
|  |

|  |
| --- |
| "keypad\_numlock":"\x53", |
|  |

|  |
| --- |
| "keypad\_/":"\x54", |
|  |

|  |
| --- |
| "keypad\_\*":"\x55", |
|  |

|  |
| --- |
| "keypad\_-":"\x56", |
|  |

|  |
| --- |
| "keypad\_+":"\x57", |
|  |

|  |
| --- |
| "keypad\_enter":"\x58", |
|  |

|  |
| --- |
| "keypad\_1":"\x59", |
|  |

|  |
| --- |
| "keypad\_2":"\x5a", |
|  |

|  |
| --- |
| "keypad\_3":"\x5b", |
|  |

|  |
| --- |
| "keypad\_4":"\x5c", |
|  |

|  |
| --- |
| "keypad\_5":"\x5d", |
|  |

|  |
| --- |
| "keypad\_6":"\x5e", |
|  |

|  |
| --- |
| "keypad\_7":"\x5f", |
|  |

|  |
| --- |
| "keypad\_8":"\x60", |
|  |

|  |
| --- |
| "keypad\_8":"\x61", |
|  |

|  |
| --- |
| "keypad\_0":"\x62", |
|  |

|  |
| --- |
| "keypad\_.":"\x63" |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| LookUpShiftLayer = { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "A":"a", |
|  |

|  |
| --- |
| "B":"b", |
|  |

|  |
| --- |
| "C":"c", |
|  |

|  |
| --- |
| "D":"d", |
|  |

|  |
| --- |
| "E":"e", |
|  |

|  |
| --- |
| "F":"f", |
|  |

|  |
| --- |
| "G":"g", |
|  |

|  |
| --- |
| "H":"h", |
|  |

|  |
| --- |
| "I":"i", |
|  |

|  |
| --- |
| "J":"j", |
|  |

|  |
| --- |
| "K":"k", |
|  |

|  |
| --- |
| "L":"l", |
|  |

|  |
| --- |
| "M":"m", |
|  |

|  |
| --- |
| "N":"n", |
|  |

|  |
| --- |
| "O":"o", |
|  |

|  |
| --- |
| "P":"p", |
|  |

|  |
| --- |
| "Q":"q", |
|  |

|  |
| --- |
| "R":"r", |
|  |

|  |
| --- |
| "S":"s", |
|  |

|  |
| --- |
| "T":"t", |
|  |

|  |
| --- |
| "U":"u", |
|  |

|  |
| --- |
| "V":"v", |
|  |

|  |
| --- |
| "W":"w", |
|  |

|  |
| --- |
| "X":"x", |
|  |

|  |
| --- |
| "Y":"y", |
|  |

|  |
| --- |
| "Z":"z", |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "\_":"-", |
|  |

|  |
| --- |
| "plus":"=", |
|  |

|  |
| --- |
| "+":"=", |
|  |

|  |
| --- |
| "{":"[", |
|  |

|  |
| --- |
| "}":"]", |
|  |

|  |
| --- |
| ":":";", |
|  |

|  |
| --- |
| "@":"'", |
|  |

|  |
| --- |
| "~":"#", |
|  |

|  |
| --- |
| "<":",", |
|  |

|  |
| --- |
| ">":".", |
|  |

|  |
| --- |
| "?":"/", |
|  |

|  |
| --- |
| "|":"\\", |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "shift\_backspace":"backspace", |
|  |

|  |
| --- |
| "shift\_keypad\_delete":"keypad\_.", |
|  |

|  |
| --- |
| "shift\_keypad\_0":"keypad\_0", |
|  |

|  |
| --- |
| "shift\_keypad\_1":"keypad\_1", |
|  |

|  |
| --- |
| "shift\_keypad\_2":"keypad\_2", |
|  |

|  |
| --- |
| "shift\_keypad\_3":"keypad\_3", |
|  |

|  |
| --- |
| "shift\_keypad\_4":"keypad\_4", |
|  |

|  |
| --- |
| "shift\_keypad\_6":"keypad\_6", |
|  |

|  |
| --- |
| "shift\_keypad\_7":"keypad\_7", |
|  |

|  |
| --- |
| "shift\_keypad\_8":"keypad\_8", |
|  |

|  |
| --- |
| "shift\_keypad\_9":"keypad\_9", |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| "!":"1", |
|  |

|  |
| --- |
| "\"":"2", |
|  |

|  |
| --- |
| "¬":"`", |
|  |

|  |
| --- |
| "£":"3", |
|  |

|  |
| --- |
| "$":"4", |
|  |

|  |
| --- |
| "%":"5", |
|  |

|  |
| --- |
| "^":"6", |
|  |

|  |
| --- |
| "&":"7", |
|  |

|  |
| --- |
| "\*":"8", |
|  |

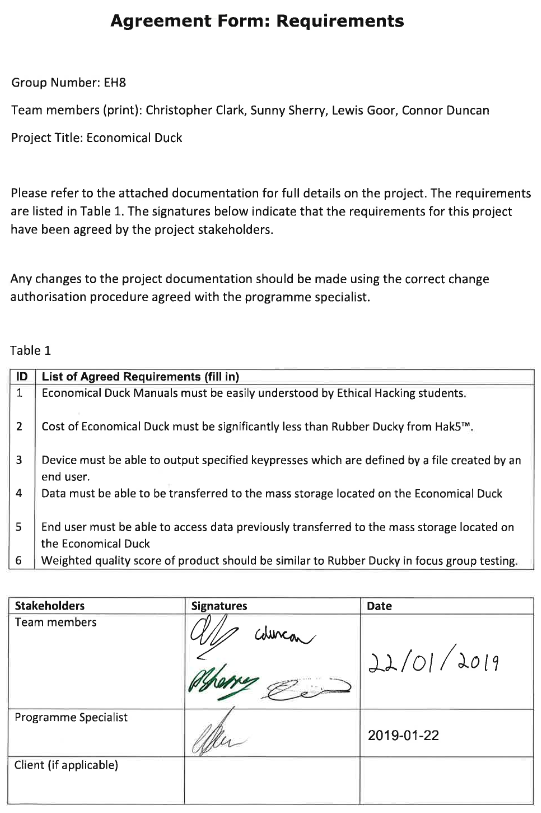
|  |
| --- |
| "(":"9", |
|  |

|  |
| --- |
| ")":"0" |
|  |

}

## Appendix B – Project Deliverables and Agreement Form





## Appendix C - Minutes

Date and Venue: 15/01/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting the team reiterated all the goals of the project and what it is that they are looking to achieve. The team discussed with Dr. Ethan Bayne all of their deliverables that were to be achieved come completion of the project. The team also questioned what needs to be included within the white paper. After discussing this with Dr. Ethan Bayne it was found that it should include the investigation, development, implementation, testing and user metric-based results.

Challenges: Sunny Sherry and Christopher Clark debated over the definition of deliverables and what the deliverables should be. To settle this debate Dr. Ethan Bayne was consulted and he clarified what exactly needed to be put down for deliverables.

Team Dynamic: The team felt very confident about being able to complete the project within the deadline timeframe and looked forward to beginning creating the device.

Following Actions: The team will meet up in a week’s time to begin the creation of the Economical Duck.

Next Meeting: 22/01/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 22/01/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting, Connor Duncan, Lewis Goor and Christopher Clark discussed the Economical duck’s scripting language, what rules should be implemented and what format it should go under. This caused several debates within the team and often they were settled by consulting Sunny Sherry. Eventually all rules were settled upon and the format of the scripting language was established. Christopher worked on breaking down the EDS into three stages, he lead the development team and created templates of the functions required in the EDS. Christopher also began creating variables and defining the syntax for the EDS. In addition to this, he began creating the documentation. Connor Duncan and Lewis Goor begin coding the EDS and created a lookup table in Python. Contained within the lookup table was the scan code for every key on a windows keyboard. Sunny worked on allowing developers to remotely connect to the Raspberry Pi Zero via a network. This made the Raspberry Pi Zero easier to develop on for the user. He also managed to enable the USB mass storage function on the Raspberry Pi Zero, with it acting as a keyboard. In addition to this Sunny managed to get the OS foot printing working, allowing the Raspberry Pi Zero to detect what OS the target is running.

Challenges: In this meeting, Connor said that he was unsure about what exactly was looking to be achieved in the scripting language. After consulting with all members of the team, a clear picture was established for what was required. Another challenge that arose was the unit of measurement for one of the commands. Lewis, Connor and Sunny wanted the format of the delay key to be in milliseconds. However, Christopher wanted it to be in milliseconds multiplied by ten. After approximately five minutes of debate, a vote occurred for the unit to be used and it was decided that milliseconds would be used, by three votes to one.

Team Dynamic: In this meeting, the team faced a large number of challenges about the design of the EDS. This lead to a large amount of conflict, however the team debated upon each issue that arose and came to a democratic decision for each outcome. This lead to the team bonding better and feeling positive.

Following Actions: The team will meet up in one week’s time to continue development of the economical duck.

Next Meeting: 29/01/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 29/01/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting, Sunny proposed that the EDS is not used, and a Python Library is used instead. After approximately twenty minutes of debating, Dr. Ethan Bayne was consulted. It was decided that a basic and advanced option would be available to the user, with the basic option making use of the EDS and the advance option allowing the user to code in Python. Connor, Lewis and Christopher began coding the EDS.

Connor worked on creating a function that would accept characters/keypresses and convert them into their scan codes, and as a result create a HID packet to be sent.

Lewis worked on creating specific functions for the EDS. Lewis managed to create a type and delay function.  
Sunny discussed the work that he had undertaken over the past week with the group. Sunny helped the group with any challenges that they faced while they were coding, as he is experienced in Python. Sunny helped Chris with planning how the repeat function will work and helped Connor with sending the HID packet to the hardware. Sunny also set up a git repository for the group to upload all of their files to.

Christopher focused on developing the EDS class library and delegated the command functions to Lewis.

Challenges: The team experienced conflict when Sunny first introduced the Python Library idea. However, after debating for a while and consulting Dr. Ethan Bayne, a proposal was made that made both Sunny and the rest of the group happy. This proposal was to implement a basic and advanced option. The basic option would make use of the EDS and the advanced option would allow the user to code in Python and implement the EDS as a library. Sunny and Connor proposed that in the EDS command function, a space symbolizes a new command. Chris did not like this idea as he did not see why the user could not put the command on a new line. It was argued that by allowing a space to symbolize a new command, the user would have to write less, albeit both options having the same result. A vote was then taken and Sunny, Connor and Lewis all voted in favor of having a space as a symbol for a new command to be created.

Team Dynamic: In this meeting the team faced a lot of challenges that allowed for heated discussions. However, all challenges were successfully overcome, and a lot of progress was made for the EDS. This left the team feeling positive and successful.

Following Actions: In the next meeting, the team aims to continue programming the EDS.

Next Meeting: 05/02/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 05/02/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor

Absences: Connor Duncan

Discussed: In this meeting, Sunny Sherry started testing all of the installation scripts for the Economical Duck and focused mainly on the making the scripts be able to run after a reboot. Sunny also worked with Christopher Clark on making a repeat function for the EDS which will be capable of running both nested and sequential repeats of any size.

Lewis Goor began work on the press function which would decipher the keypresses from the user, before translating them into hid packets.

Christopher helped Lewis on the press function as well as collaborating with Sunny on the repeat function.

Connor Duncan worked independently on a function that will take in a list of strings, and create a HID packet that would change depending on the strings passed in.

Challenges: Connor Duncan was unable to attend this meeting due to being back home. To combat this, Connor understood exactly what tasks he needed to have completed by the meeting the following week and made sure that these were done on time.

Team Dynamic: The team remained positive in this meeting despite missing a member.

Following Actions: In the next meeting the team will collaborate with Connor and look to consolidate all the code into one program.

Next Meeting: 12/02/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 05/02/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting Connor Duncan explained to the team how his HID function works, what input is required to make it work and what the output looked like. After talking to Sunny Sherry, Connor needed to change the output of the function to better suit the needs of Sunny. Connor also helped Lewis Goor continue and finish developing the press function.

Sunny continued working on the installation scripts and making them work from a fresh install.

Christopher Clark learned and taught the other members in the team how to correctly use GitHub. Christopher also created a master file for all of the team to insert their weekly workings into. Christopher continued working on the repeat function and added extra code to this that Sunny had created.

Lewis Goor finished the creation of the press function and updated the type function to make use of the function that creates the HID packet.

Challenges: The team did not face any significant challenges in this meeting

Team Dynamic: In this meeting the team felt positive that they were making good progress towards the final product, and this in turn motivated the team.

Following Actions: In the next meeting the team hopes to be able to complete and test the code for the basic and advanced version of the Economical Duck.

Next Meeting: 12/02/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 12/02/18, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting Connor Duncan finished the code for the basic version of the Economical Duck and created the Python library for the advanced version.

Christopher Clark amended issues that were found to be contained within the class libraries and began writing about the planning and development stages of the Economical Duck in the white paper. Along with this Christopher also created a process diagram of the EDS basic code.

Sunny Sherry setup the Raspberry Pi Zero to enable testing to be carried out and to verify that the code worked successfully. Sunny also finished testing on the installation code. In addition to this Sunny helped the team with coding issues that arose and made amendments where code was incorrect.

Lewis Goor helped assist Christopher Clark in writing the white paper and helped complete the software code.

Challenges: Several coding issues arose while coding the EDS. To correct these issues the team consulted with each other to debug the code and fix the problem. For example, when testing the code, an unexpected output was displayed by the Raspberry Pi Zero. In order to find out what this issue was, the team went through the code together and found that one of the Python functions was being incorrectly used.

Team Dynamic: The team Dynamic was very strong throughout this meeting as a large amount of progress was made and the team felt that they were quickly and successfully approaching the end of the completion of the EDS.

Following Actions: In the next meeting the team looks to be able to give a demo of the basic and advanced version of the EDS to the client and begin working on the documentation and white paper.

Next Meeting: 26/02/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 26/02/18, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting, the team demonstrated their progress so far to the client. Sunny Sherry spent a large portion of the week developing the code created and creating the demo to display. This demo involved demonstrating the basic and advanced methods of the Economical Duck. In the basic method, the Raspberry Pi Zero would open up notepad and run “Hello World!” five times. For the advanced method, the Pi would open a command prompt, download a script from the Raspberry Pi Zero that would open up a reverse TCP shell, then copy all contents of a folder on the target to a folder that was named the same as the target computer. After demonstrating this to the client, the team began working on the White Paper and Connor Duncan started populating the Mac scan code dictionary.

Challenges: No challenges were faced by the team in this meeting.

Team Dynamic: The team gained positive feedback an all of the progress that had been completed so far and as a result felt motivated to continue the project.

Following Actions: In the next meeting the team will aim to continue the white paper and get the OS detection to work for a mac keyboard.

Next Meeting: 05/03/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 26/02/18, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting Sunny Sherry and Connor Duncan worked on getting the Economical Duck to work on a Mac keyboard. Connor used his own personal mac keyboard to providing testing on. Christopher Clark and Lewis Goor proceeded to work on the White paper. Lewis began filling out the software testing of the white paper. Christopher began reformatting the headings of the white paper, working on the aim section, adding notes for what information needs provided in each section of the paper and filling out the overview section.

Challenges: Sunny and Connor experienced several problems when it came to populating the mac scan code lookup table. Due to the mac keyboard layout being different, a lot of results were unexpected. Having taken the scan codes from the “Events.h” folder on the mac, it was found that these scan codes do not match for the desired keys. To find a solution to this Sunny created a program that would run through every hexadecimal combination and output the registered key on the mac. Connor would then enter all of these values into the mac scan code dictionary.

Another challenge that was faced was that the Economical duck does not work with the mac. This is due to the Economical Duck attempting to read in the value of the CAPS\_LOCK key, but due to the way that mac is configured the pi is unable to get the status of the key. This creates problems for the Economical Duck detecting if it is connected to the target computer, and when it can disconnect. In this meeting the team was unable to find a solution and so agreed to come back next week and decide if it was practical to carry on with attempting to get the Economical Duck working on macs.

Team Dynamic: Today the team felt tired and unmotivated towards writing the white paper. With the bulk of the program for the Pi already created, there was not much more left other than the write ups. However, each member understood that these are vital parts for the project and so did not complain when it came to undertake these tasks.

Following Actions: In the next meeting the team will aim to continue trying to get the EDS working on a mac and will continue writing up the white paper.

Next Meeting: 12/03/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 12/03/18, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting the team continued working on the white paper. Connor wrote about the createHIDpacket function, how it works and how it was tested. Lewis began writing the background section and Christopher began writing the basic manual. Sunny continued the testing of the mac script. However, after numerous failed attempts it was found that it would not be possible to get the EDS to work correctly on a mac.

Challenges: One of the challenges faced in this meeting was to do with the University network. Part of the way through the meeting the network crashed, resulting in the one drive stopping working. This meant that the team could not complete the White Paper at the same time and thus halted all progress for this meeting.

Team Dynamic: The team was feeling confident about their deadlines as they believed that they were making good progress towards the completion of the white paper, despite progress being halted for the day by the network.

Following Actions: In the next meeting the team will continue working towards completing the white paper.

Next Meeting: 19/03/18, 10:00am, Hacklab, Abertay University.

Date and Venue: 19/03/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting the team continued working on the White Paper and manuals. The advanced manual was completed, and the basic manual was also completed. Christopher was working on the basic manual and Connor was working on the advanced. Lewis was working on the White Paper and continued to create the background section. Sunny worked on developing the installation script and worked on bypassing a bug contained with TLS.

Challenges: One of the challenges faced in this meeting was getting the installation script to run correctly. In order to combat this challenge Sunny added several debugging features and made it more robust. This allowed Sunny to find that the bug was in the git tool, and then switched from TLS to SSH to allow the bug to be bypassed.

Team Dynamic: The team were happy with the progress that they have made and look forward to completing the documentation.

Following Actions: In the next meeting the team will continue their work towards the white paper, test the Raspberry Pi Zero for how easy it is to use and look to start thinking about focus group testing.

Next Meeting: 26/03/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 26/03/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting Sunny continued his work on the installation scripts and getting the EDS up and running to allow testing on. Connor continued working on the White Paper and filling out the future work and product testing sections. Lewis completed working on the background and moved on to writing up the general discussion section. Christopher delegated tasks to each team member throughout the meeting and began combining the basic and advanced manuals, updating the payloads section and inserting a new section called configuration, which covers the Economical Duck configuration settings.

Challenges: A challenge faced in this meeting was that outside of Sunny, no other team member had worked on the Economical Duck and did not fully understand how to set it up. To combat this Sunny explained to the whole team how it worked, however Lewis and Connor did not follow. Therefore, Christopher went through his understanding of how it worked to give a different perspective, and both Lewis and Connor then understood how to set up the Economical Duck.

Team Dynamic: The team dynamic was positive throughout this meeting, and the team looked forward to undertaking the product testing with focus groups.

Following Actions: In the following meetings the team hopes to carry out focus group testing with other members of the University.

Next Meeting: 02/04/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 02/04/19, 10:00am, Hacklab, Abertay University.

Present: Christopher Clark, Sunny Sherry, Lewis Goor, Connor Duncan

Absences: N/A

Discussed: In this meeting the team started to carry out their focus group testing and continue completing the white paper. Lewis and Connor worked together to work on the abstract, countermeasures, general discussion and conclusion sections. In addition to this they proof read Sunny’s work in the Hardware implementation section from the previous week. Christopher Clark organized the focus group testing and brought in a participant from the University to test the EDS and compare it to a normal Rubber Ducky. Christopher also updated the configuration section of the manual, added to the White Paper focus group section and looked over all of the sections in the White Paper. Sunny added in instructions on how to physically set up the device, amended any errors in the advanced manual and tracked down a bug that was causing the Economical Duck to crash in the net lab. At the end of the meeting, Christopher Clark and Sunny Sherry stayed behind to carry out focus group testing. They asked members around the university if they would like to participate, and then spent time explaining what the testing involved. A satisfactory number of participants were collected, and the focus group testing was completed.

Challenges: One of the challenges faced in this meeting was that when carrying out the focus group testing, the Economical Duck would crash for no apparent reason. In order to fix this Sunny tested the Economical Duck in the Hack Lab rather than the net lab and found that the device worked successfully. From carrying out these tasks, Sunny was able to identify where the bug was located and what could be done to fix this.

Team Dynamic: The team felt very positive in this meeting and were making very good progress towards the completion of the White Paper and the finalized product of the Economical Duck.

Following Actions: In the next meeting the team will look to complete the White paper.

Next Meeting: 09/04/19, 10:00am, Library, Abertay University.

Date and Venue: 09/04/19, 10:00am, Library, Abertay University.

Present: Lewis Goor, Connor Duncan

Absences: Christopher Clark, Sunny Sherry

Discussed: In this meeting, Lewis and Connor planned on continuing work on the white paper. However, the information from the focus group testing had not been uploaded and therefore no work could be completed on the white paper. Lewis and Connor then decided to reschedule the meeting for Thursday instead.

Challenges: Sunny Sherry and Christopher Clark were both not present in this meeting. Christopher had suffered from a migraine in the early hours of the morning and was unable to attend the meeting for this reason. Sunny was away on business thus not being able to make this meeting. Although Sunny had stated previously that he could not attend but would be able to work remotely if asked to.

Team Dynamic: Despite not being able to continue work on the white paper, Lewis and Connor were still confident that they would be able to finish the white paper within the next meeting

Following Actions: In the next meeting the team looks to finish the White Paper.

Next Meeting: 11/04/19, 10:00am, Hacklab, Abertay University.

Date and Venue: 11/04/19, 10:00am, Hacklab, Abertay University.

Present: Lewis Goor, Connor Duncan, Christopher Clark

Absences: Sunny Sherry

Discussed: In this meeting the present team members continued working on the White Paper. Christopher Clark input content for the focus group testing section, Connor Duncan updated the abstract to include the focus group testing, and Lewis Goor updated the conclusion to include the focus group testing.

Challenges: The team did not face any challenges in this meeting.

Team Dynamic: The team felt very positive as they completed the first draft of the White Paper.

Following Actions: In the next meeting the team will run through the White Paper and proof read it.

Next Meeting: 16/04/19, 10:00am, Library, Abertay University.

Date and Venue: 16/04/19, 10:00am, Library, Abertay University.

Present: Lewis Goor, Connor Duncan, Christopher Clark, Sunny Sherry

Absences: N/A

Discussed: In this meeting the team collected together to discuss the White Paper and proof read it. At this point it was realized that the countermeasures section still needed input from Sunny. Sunny added information to the counter measures section and checked the technical accuracy of the paper. Lewis and Connor went through the whole document and proof read it, checking for any spelling and grammar mistakes. Christopher updated the overview of development and clarified certain aspects of the report with Dr. Ethan Bayne.

Challenges: The team did not face any challenges in this meeting.

Team Dynamic: The team felt very accomplished as they completed the second draft of the White Paper in this meeting.

Following Actions: Submit the White Paper to Blackboard.