HID Computer Remote with Convenient Shortcut User Interface

James Mulvenna(100965629)
Jori El-Saikali(100943888)

Table of Contents

1. Introduction

1.1 Context

The topic of this report includes but is not limited to modifying the Linux USB Boot Protocol keyboard driver. This driver utilizes asynchronous callbacks, dynamically allocated memory, synchronization and a complex API. Currently the latest HID devices maintain both a simple software and hardware interface, however generally come at a soaring price[1]. To put our project goals into perspective we'll begin with an average day example; you want to watch television from bed and simply can't afford the newest smart television. Our goal in this project is to design a simplistic remote with shortcut functionality to resolve these issues. Outlined are the major components which include, the driver, remote, and user interface along with elaboration on the challenges involved in the process. Finally, we draw conclusions, and engage in final statements.

1.2 Problem Statement

We were determined to design a HID for Ubuntu version 14.04 that provided shortcut functionality at the click of a button for the ordinary light computer user. In the present-day, convenience is a critical factor for typical consumers, as BLS (Bureau of Labour Statistics) proved that work related time spent for employed persons aged 25 to 54 is about 8.9 hours daily[2] which works out to 135 days annually. Our goal was to create a simplistic device that implements shortcut functionality instantaneously at the user's fingertips. We gained motivation to tackle this issue through statistics and studies based on the user's needs/wants. Facts show there are approximately 3.5 billion internet users globally[3] proving technology is accelerating faster than ever. We've found that the technology market lacks CPU devices with shortcut functionality. These are only a few supporting facts guiding our motivations, leading to the relevancy of this project.

1.3 Result

Our goal was to create a HID for Ubuntu version 14.04 with shortcut functionality to support convenience, and improve consumer CPU efficiency. Computing projects can be intimidating at first, as seen in lower year classes such as COMP2404, and COMP2402. One of the main concepts we grasped was the ability to take a large issue and break it down into smaller approachable issues. Working with the kernel was frightening, however by applying this concept we were able to disregard a lot of unrelated sectors of the lower level OS that had no relation to creating a HID. In the end we achieved our goal by breaking down our problem into the following smaller issues shown in Figure 1.

[1] Rosoff, Matt. "Every type of tech product has gotten cheaper over the last two decades — except for one." *Business Insider*. Business Insider, 14 Oct. 2015. Web. 13 Nov. 2016.
[2] "About the Charts." *U.S. Bureau of Labor Statistics*. U.S. Bureau of Labor Statistics, 26 Oct. 2015. Web. 27 Nov. 2016.
[3] "Internet Users." *Number of Internet Users (2016) - Internet Live Stats*. N.p., 1 July 2013. Web. 27 Nov. 2016.

Issues	**Goal**
Brainstorm	Break up our individual goals, and understand the problem
Statistical	Gain statistical knowledge
Implementation	Put knowledge into practice, code the program
Room for improvement	Gain feedback from peers
Closing	Touch up, and finalize our project

Figure 1: Issues

We found the most meaningful issues were the “Brainstorming” and “Room for improvement” issues. The “Brainstorming” issue helped us clinch where the project was headed and why. Meanwhile the “Room for improvement” issue was set so that we could use real consumer feedback to improve our final product. We found breaking up the project into smaller issues helped us accomplish our goal, leaving us satisfied with the USB remote device we produced.

1.4 Outline

The report in its entirety covers various concepts. Section 2 demonstrates background information about the usbkbd.c module and its relevancy to our project. Section 3 expresses extensively the challenges we faced, as well as the paths we’ve taken to achieve our goals and retrieved result. We further study this result as we engage in evaluation studies covered in Section 4. Finally, in Section 5 we trace our concluding words by disputing COMP3000 relevancy in addition to endeavoring in future work.

2 Background Information

Virtually every keyboard HID in Linux uses the driver usbkbd.c (Universal Serial Bus Keyboard)[4]. This module was created by Vojtech Pavlik and is located in Linux under the usbhid directory. For our project we worked particularly close with the usbkbd module as it reflected our device and our final goal. Moving forward, we’ll go in depth with what functionality is needed to build a usb keyboard driver from the ground up and how it applies to the kernel space. First off, when dealing with a keyboard you need to register the keys. Each individual key reflects a corresponding keycode in that the kernel space registers. The following picture shown in Figure 2 will provide a visual understanding of which buttons are used on the device we created, and the corresponding key codes the kernel interprets.

[4] Pavlik, Vojtech. Linux/drivers/hid/usbhid/usbkbd.c - Linux cross reference - free electrons. n.d. Web. 5 Dec. 2016.

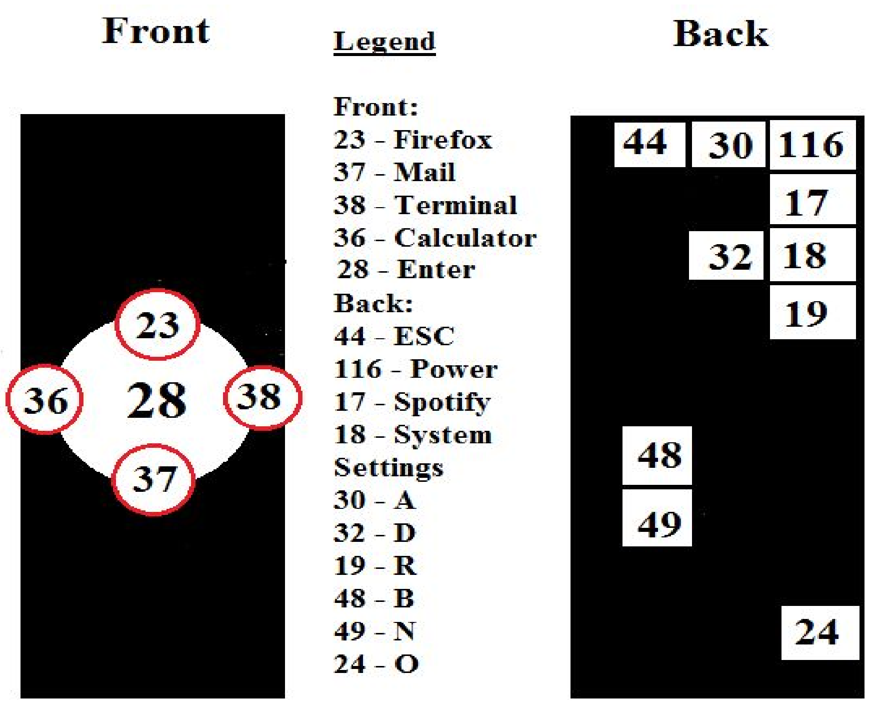
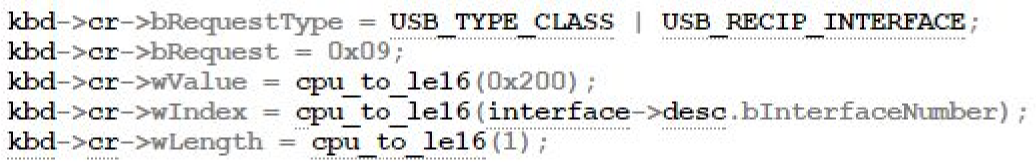


Figure 2: Keycodes

Immediately upon insertion, the keyboard interface calls a probe function. By definition a probe is to physically explore or examine, meaning the probe function is the entire structure forming the API. The probe function assesses the USB device information (including endpoints, vendor id, product id, and much more) to determine if it can initialize a URB. The URB is used to send and receive keyboard key-press/key-release data from the device itself. Finally the probe function registers the device making it accessible to other computer applications such as our shortcut applications. Following the probe function, when the device is used, the open function is called to submit the IRQ which is caught by the keyboard IRQ function. Before proceeding, if there is space available, memory must be allocated. If the space requirement is met, the keyboard IRQ function loops over the total keys, parsing the data into 8 bits a key. Finally, the keyboard IRQ reports key-presses/key-releases to the syslog. Upon close, a function called free memory is called to cleanup the memory that has been used over the lifetime of the API . Last but not least, id tables and disconnect functions are also used to document relevant information to the syslog. In kernel programming two crucial factors are efficiency and consistency. It is challenging to implement functionality without the use of comparable modules. One concept we noticed while working hand-in-hand with the usbkbd.c module is that programming practices weren’t always used.
For example, not using functions when applicable, or disregarding programming comments.



```
kbd->cr->bRequestType = USB_TYPE_CLASS | USB_RECIP_INTERFACE;
kbd->cr->bRequest = 0x09;
kbd->cr->wValue = cpu_to_le16(0x200);
kbd->cr->wIndex = cpu_to_le16(interface->desc.bInterfaceNumber);
kbd->cr->wLength = cpu_to_le16(1);
```

To back up this argument the source "Learn Cpp"[5] says functions provide organization, reusability, and primarily abstraction. Since this module is open source, we consider it imperative to use proper programming practices in that programmers will have a smoother time innovating the code. However, programming practices are opinionated depending on your location and time in computational history and this argument is not intended to insult the author.

3 Result

Throughout all of our previous explained issues, we never forgot our main goal which was to create a USB remote with a simplistic design and easy to use interface. In summary, our project was broken down into a few major components explained in Figure 3.

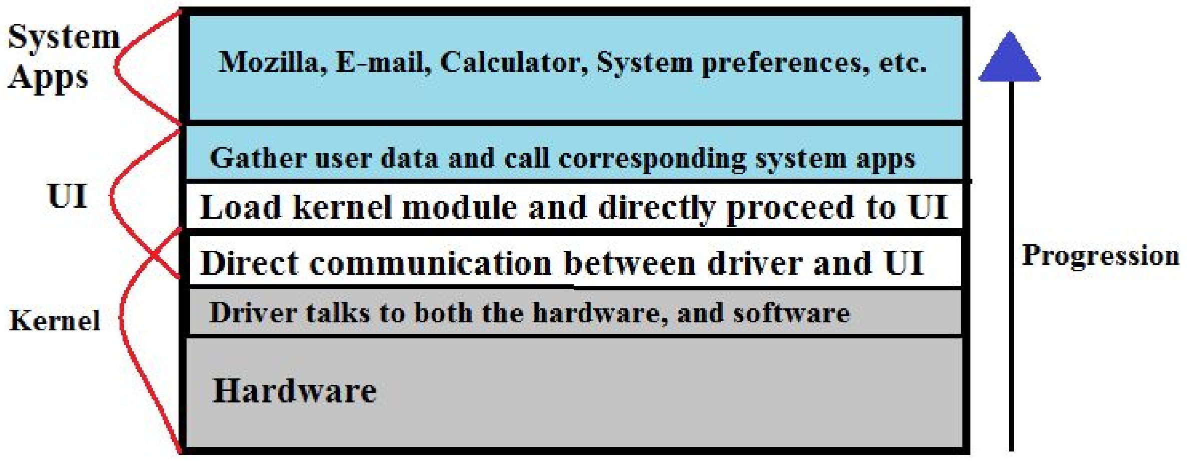
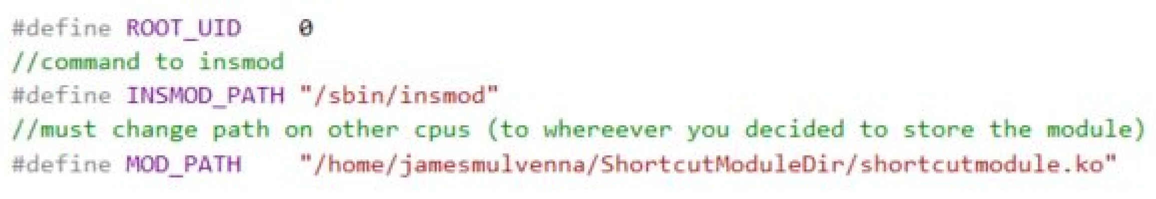

System Apps
UI
Kernel
Mozilla, E-mail, Calculator, System preferences, etc.
Gather user data and call corresponding system apps
Load kernel module and directly proceed to UI
Direct communication between driver and UI
Driver talks to both the hardware, and software
Hardware
Progression

Figure 3: Software architecture

As seen in Figure 3, the obtained result was built from the ground up. The first challenges we faced were designing the driver to accommodate the hardware. This was accomplished in the kernel space, which required various system callbacks to register the device and its requests. After implementing the driver we dealt with driver-user interface communication. This was certainly our toughest challenge, as it is problematic for kernel space to communicate with user space. This issue was resolved through inserting the module through the user space and for the sake of its importance the following code snippets are explained in detail showing how this was executed.

[5] Ishana. 1.4b — why functions are useful, and how to use them effectively. Learn C++, 5 Aug. 2015. Web. 5 Dec. 2016.



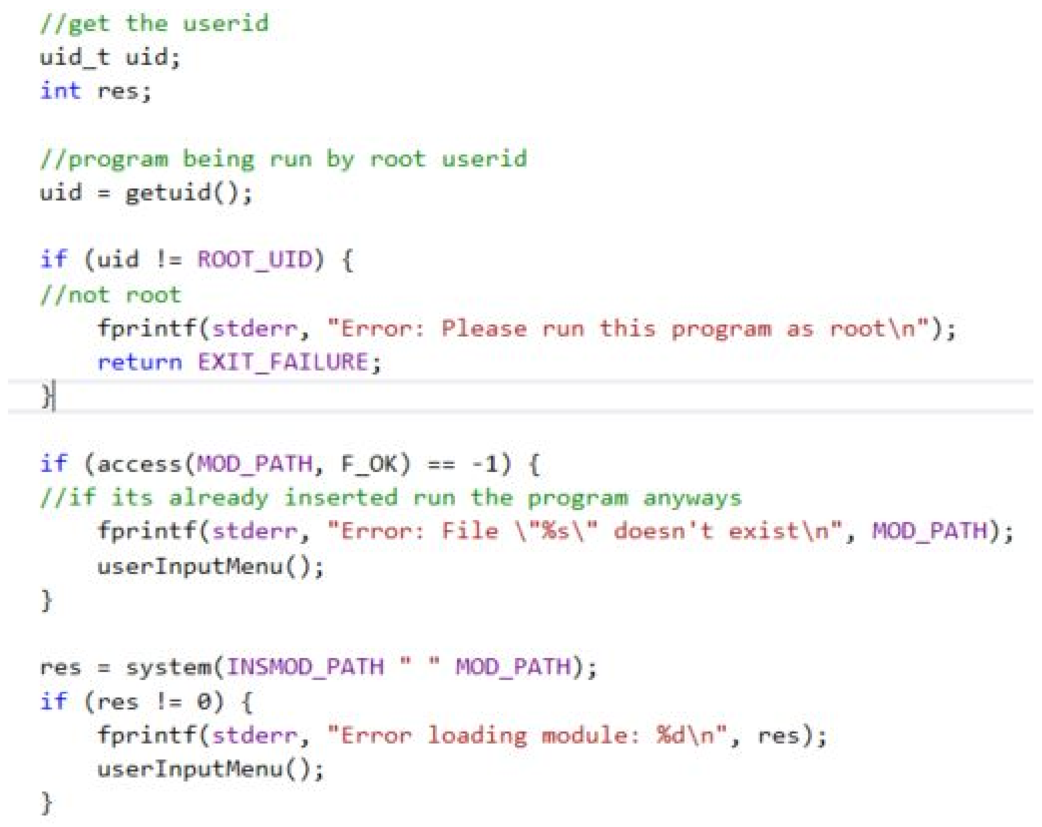
```
#define ROOT_UID    0
//command to insmod
#define INSMOD_PATH "/sbin/insmod"
//must change path on other cpus (to whereever you decided to store the module)
#define MOD_PATH    "/home/jamesmulvenna/ShortcutModuleDir/shortcutmodule.ko"
```

Figure 4: Code snippet 1

Disregarding programming comments, Figure 4 is explained line by line:

1. ROOT_UID is a variable used to later check if the current user is root on the system
2. INSMOD_PATH is the system call used to insert any module on the system
3. MOD_PATH is the location of our module on the system

Using these variables, Figure 5 executed in the following fashion.



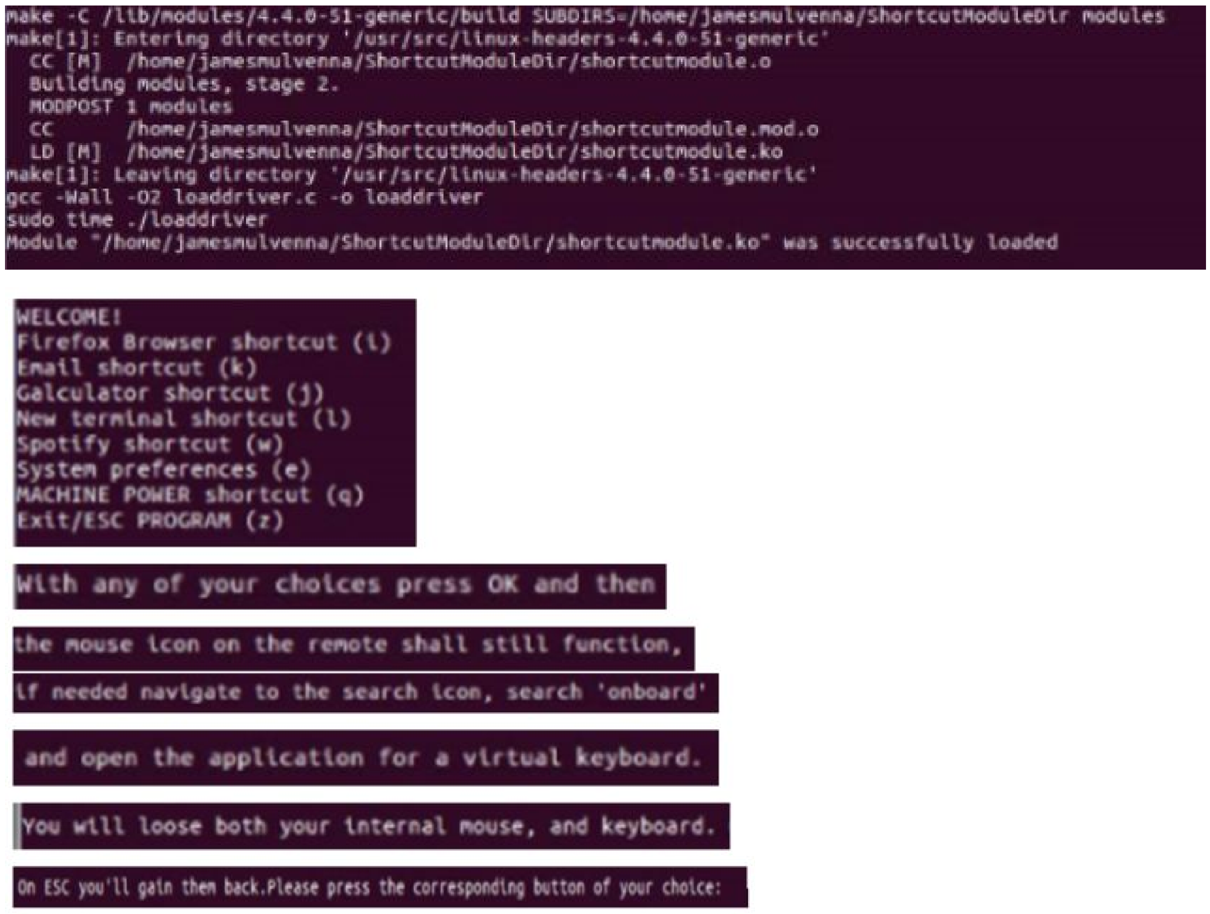
```
//get the userid
uid_t uid;
int res;

//program being run by root userid
uid = getuid();

if (uid != ROOT_UID) {
//not root
    fprintf(stderr, "Error: Please run this program as root\n");
    return EXIT_FAILURE;
}

if (access(MOD_PATH, F_OK) == -1) {
//if its already inserted run the program anyways
    fprintf(stderr, "Error: File \"%s\" doesn't exist\n", MOD_PATH);
    userInputMenu();
}

res = system(INSMOD_PATH " " MOD_PATH);
if (res != 0) {
    fprintf(stderr, "Error loading module: %d\n", res);
    userInputMenu();
}
```

Figure 5: Code snippet 2

4. Uid_t uid, res respectively are the current user ID, and the return value of the system calls used
5. First we check if the user is root, if there not we fail the insertion (need root privileges to insert modules to kernel space
6. Essentially the next snippet checks if the module is already inserted, in this case we ignore that it has and enable the user to proceed

7. Next, insert the module according to the path, if the value doesn't equal 0, we conclude it either is already inserted so cause an error, however still enable the user to proceed
8. If all else, the module is inserted, and we enable the user to proceed

Moving forward, as we neared our final goal, an emphasis was put on the user interface, constructing it with minimal confusion. Studies show an exceptional user interface is the backbone to software, which we consider true. Things to look for to determine if a user interface is acceptable are: load times, prioritized information, and a clear simple path[6]. A great user interface is so crucial because it is the medium in which data transfer between the API and user happens. Knowing this, built was an interface that was simplistic and easy to master. Our final user interface is comprised into actual screenshots in Figure 6.



```
make -C /lib/modules/4.4.0-51-generic/build SUBDIRS=/home/jamesmulvenna/ShortcutModuleDir modules
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-51-generic'
  CC [M]  /home/jamesmulvenna/ShortcutModuleDir/shortcutmodule.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jamesmulvenna/ShortcutModuleDir/shortcutmodule.mod.o
  LD [M]  /home/jamesmulvenna/ShortcutModuleDir/shortcutmodule.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-51-generic'
gcc -Wall -O2 loaddriver.c -o loaddriver
sudo time ./loaddriver
Module "/home/jamesmulvenna/ShortcutModuleDir/shortcutmodule.ko" was successfully loaded
```

```
WELCOME!
Firefox Browser shortcut (i)
Email shortcut (k)
Calculator shortcut (j)
New terminal shortcut (l)
Spotify shortcut (w)
System preferences (e)
MACHINE POWER shortcut (q)
Exit/ESC PROGRAM (z)
```

```
With any of your choices press OK and then
```

```
the mouse icon on the remote shall still function,
if needed navigate to the search icon, search 'onboard'
```

```
and open the application for a virtual keyboard.
```

```
You will loose both your internal mouse, and keyboard.
```

```
On ESC you'll gain then back.Please press the corresponding button of your choice:
```

Figure 6: User interface

As you can see the information is brief however clear. The user is equipped with how the program is being loaded, as well as a simple list of options coupled with corresponding keys. Moreover, simple messages regarding how to execute your choice, the consequences taken

[6] "Focusing on UI/UX and the importance of good design | Up&Up." *Creative & Design*. Up&Up, 29 July 2014. Web. 6 Dec. 2016.

and how to escape the program are all provided. It may not be suitable for people who haven’t used a computer previously, however recognize it is abstract and maintains an “easy to use” design. Additionally, we prove it loads in a fraction of a second**.** Over a series of 10 tests the average load time was 0.002s**.** The time had come, we had a finished result in which we felt satisfied. Withal, we decided to gain consumer feedback with a sequence of evaluations to see where our product stood. In Figure 7 the consumer feedback averages are shown.

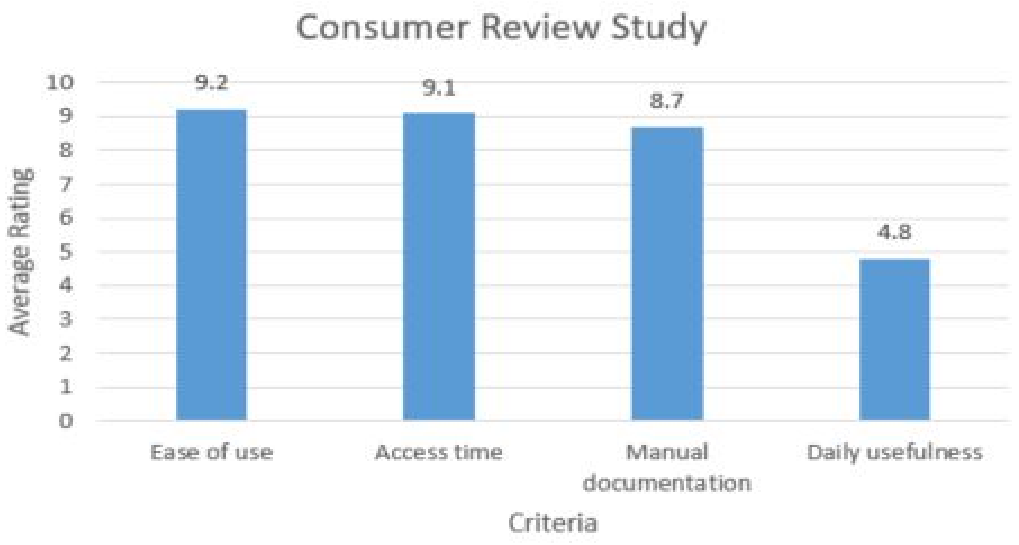


Figure 7: Feedback results

By thoroughly analysing our results, we noticed the poor usefulness rating. With this, we realized our consumer review study focused solely on the student segment whereas we should have included a wider range of ages for more accurate results. We then concluded our project by touching up main components to provide the finest product viable. As a whole, we are remarkably fulfilled with the results.

4 Evaluation

Considering our project was purely user based, it was essential to conduct a consumer review evaluation. We knew our projects strengths and weaknesses, however knowing our peers had different perspectives we distributed an evaluation form to gain the information needed to take our project to the next level. A copy of the evaluation form is in the appendix and consists of various tasks and questions, as well as a comment section. First off, the tasks all measured the time taken to open an application via the respective remote shortcut versus the time taken to open the same application via trackpad.

The following graph in Figure 8 shows the trends our data produced.

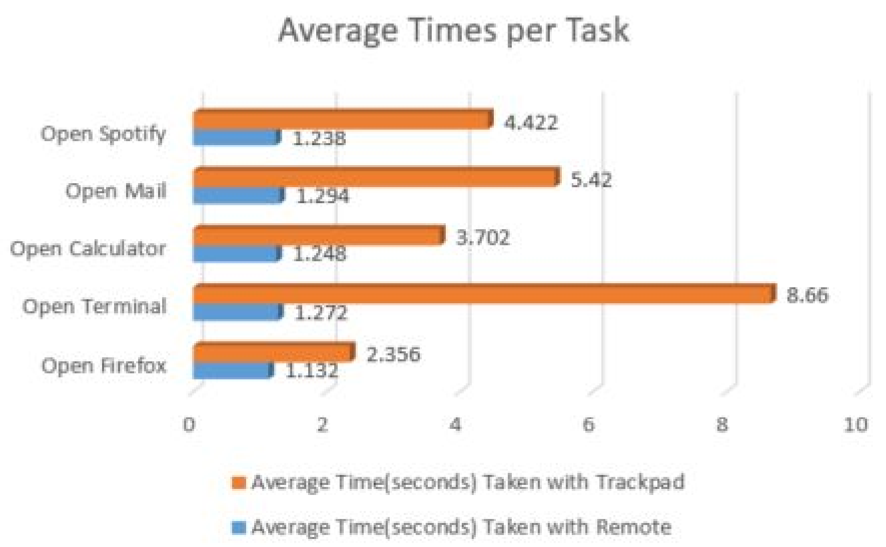


Figure 8: Graph of average time taken per task

Evidently, in every case the remote was indeed more swift. Nonetheless this analysis could have improved by incorporating the time taken to run the remote program needed to perform the respective tasks. Following this section of the evaluation we further analyzed using a questionnaire. The following chart in Figure 9 displays the consumers perspective rated out of 10 regarding usability, speed, documentation and usage.

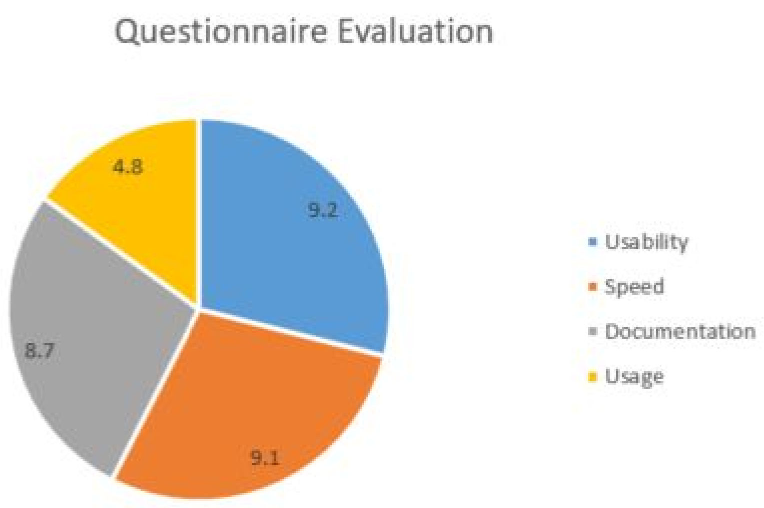


Figure 9: Pie chart displaying strengths and weaknesses

It is easy to see where our strengths and weaknesses lie. Obviously usage was our main weakness, yet we disagree. The results taken from the evaluation were conducted in situations where you surely wouldn’t demand the remote. We noticed that the questionnaire lacked scenarios such as if you are putting on a presentation, or watching a movie where the remote would be relevant. The final examination taken were comments. The comments were established so that the examinee could leave their concerns without being questioned. The main concerns included the remote was more appropriate for non tech savvies and adding functionality to customize the remote to the users liking. We took these particular conditions into discussion for further future work.

5 Conclusion

5.1 Summary

All in all, our pathway to designing a shortcut remote accompanied by a user interface and driver was a success in its entirety. From brainstorming to consumer feedback reviews and all things inbetween helped assist us in meeting our final goal. Throughout the process, we learnt how to apply concepts we previously possessed to new material pressing on. In today's market we see higher priced electronics and lack the basics such as remotes. It seems you need to buy a full electronic setup to obtain what you demand. As a team, we decided to put a stop to this because as it stands, the average adult spends over 142.5 minutes or about 10%[7] of their day in front of a computer screen. Something as simple as a remote with shortcuts can provide instantaneous support and efficiency. As a whole, we received positive acknowledgement from our peers in regards to our software and completed our primary goals which comprised of modifying the usbkbd.c module, designing a satisfactory remote and creating an endorsed user interface.

5.2 Relevance

Course topics starting from day 1 including OS structure and system calls, process ids, processes, memory management, linux kernel module, input/output, devices and many more lectures were all relevant. Applications from system calls such as software architecture, or process ids and processes like system calls along with locating processes by id were all used to create the user interface. Coming next, we used memory management by freeing/allocating memory accordingly. Meanwhile directly practising linux kernel programming by registering devices, and recording input/output events. Many more course subject were also immensely relevant.

5.3 Future Work

Forthcoming, we will attend to our examinees comments such as adding customization, making it more useful to tech savvies and our own suggestion to make it

[7] Alvarez, Mark. The average American adult spends 8 1/2 hours a day staring into screens. 31 Mar. 2009. Web. 03 Nov. 2016.

multithreaded. Safe to say, customization would definitely enhance our product. In fact, studies show that 25% of consumers are interested in the added functionality[8] which we can confirm as our review study shows 30% of our ten consumers recommended customization. In that, tech savvies will be able to customize the remote to their personal needs, therefore everyone will gain more use. These are just some of the changes we will develop. Surely we'll encounter a number of issues along the way, as nothing will ever be perfect and there's always something you can do as a developer to innovate.

Contribution of Team Members

Issues	**Contribution (% James Mulvenna, % Jori El-Saikali)**
Brainstorm (Ideas)	(50, 50)
Statistical (Data)	(80, 20)
Implementation (Programming)	(85, 15)
Room for improvement (Consumer Reviews)	(20, 80)
Closing (Putting the pieces together)	(70, 30)

[8] Insights, Bain. "Having It Their Way: The Big Opportunity in Personalized Products." *Forbes*. Forbes, 5 Nov. 2013. Web. 29 Oct. 2016.

References

Rosoff, Matt. “Every type of tech product has gotten cheaper over the last two decades — except for one.” Business Insider. Business Insider, 14 Oct. 2015. Web. 13 Nov. 2016.

"About the Charts." U.S. Bureau of Labor Statistics. U.S. Bureau of Labor Statistics, 26 Oct. 2015. Web. 27 Nov. 2016.

"Internet Users." Number of Internet Users (2016) - Internet Live Stats. N.p., 1 July 2013. Web. 27 Nov. 2016.

Pavlik, Vojtech. Linux/drivers/hid/usbhid/usbkbd.c - Linux cross reference - free electrons. n.d. Web. 5 Dec. 2016.

Ishana. 1.4b — why functions are useful, and how to use them effectively. Learn C++, 5 Aug. 2015. Web. 5 Dec. 2016.

“Focusing on UI/UX and the importance of good design | Up&Up.” Creative & Design. Up&Up, 29 July 2014. Web. 6 Dec. 2016.

Alvarez, Mark. The average American adult spends 8 1/2 hours a day staring into screens. 31 Mar. 2009. Web. 03 Nov. 2016.

Insights, Bain. “Having It Their Way: The Big Opportunity in Personalized Products.” *Forbes*. Forbes, 5 Nov. 2013. Web. 29 Oct. 2016.

Appendix

List of Acronyms:

USB (Universal Serial Bus)
API (Application Program Interface)
URB (USB Request Block)
HID (Human Interface Device)
CPU (Central Processing Unit)
OS (Operating System)
IRQ (Interrupt Request)

HID Remote with Shortcuts Evaluation Form

Name: ______________________________

Date: ______________________________

Purpose:

The purpose of the evaluation form is to collect data regarding our project. The idea of our project is to make every day used applications easier to access without having much knowledge about computers.

Instructions:

You will be given a remote control device which you will using to complete a series of tasks provided below. Read through the manual to understand what each button does. Once all tasks are completed, you are to answer the questions provided on the following page.

Tasks:

Please complete every task listed below.

1. Open Firefox with a mouse/touchpad.
2. Open Firefox with the remote control provided.
3. Open the terminal with a mouse/touchpad.
4. Open the terminal with the remote control provided.
5. Open Galculator with a mouse/touchpad.
6. Open Galculator with the remote control provided.
7. Open Mail with the mouse/touchpad.
8. Open Mail with the remote control provided.
9. Open Spotify with the mouse/touchpad.
10. Open Spotify with the remote control provided.

Questionnaire:

In this section, you will be answering the following questions. Please circle the number (where applicable) you feel best answers that question. Please answer honestly.

1. How easy/difficult was the remote to use? (1 – Extremely difficult, 10 – Very easy)

1 2 3 4 5 6 7 8 9 10

2. Do you find the remote was faster/slower to access applications? (1 – Very Slow, 10 – Very Fast)

1	2	3	4	5	6	7	8	9	10

3. Was the manual well documented? (1 – Poorly documented, 10 – Very well documented)

1	2	3	4	5	6	7	8	9	10

4. Would you use this remote on an everyday basis? (1 – Never, 10 – Everyday)

1	2	3	4	5	6	7	8	9	10

5. Who do you feel this remote would best help and why? (Eg. Parents, Grandparents, etc.)

____
____
____
____
____

Comments:

____
____
____
____

HID Remote with Shortcuts
Results

Name of test subject: ______________________________

Results:

Question	Time it took to complete
1	
2	
3	
4	
5	
6	
7	
8	

9	
10	