DUBLIN CITY UNIVERSITY

SCHOOL OF ELECTRONIC ENGINEERING

Universal Design For An Internet Radio Appliance

Iseoluwani Alexander Dare

March 2022

BACHELOR OF ENGINEERING

IN

ELECTRONIC AND COMPUTER ENGINEERING

MAJORING IN

THE INTERNET OF THINGS

Supervised by Dr Barry McMullin

Acknowledgements

I would like to thank my supervisor Dr Barry McMullin for his guidance, enthusiasm and commitment helping me to get to the completion of this project. Finally I would like to express my deep appreciation to Dr Robert Clare for providing me the a suitable work environment and equipment necessary to allow me to work throught this project.

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the DCU Academic Integrity and Plagiarism at https://www.dcu.ie/system/files/2020-09/1_-_integrity_and_plagiarism_policy_ovpaa-v4.pdf and IEEE referencing guidelines found at https://loop.dcu.ie/mod/url/view.php?id=1401800 .

Name: ***Iseoluwani Alexander Dare*** Date:***27th February 2022***

Abstract

“When we design for disability first, we often stumble upon solutions that are not only inclusive but also are often better than when we design them for the norm”.

This report details the design, implementation and testing stages of designing an Internet Radio Appliance with the main motive of design choices being the the pursuit of implementing concepts of Universal Design to tailor easy access for profile individuals with disabilities.

The proposed solution, makes use of a Raspberry Pi 3 Model B+, an LCD display, stereo amplifier and dual push buttons for operativity.

The results proved the appliance to be system-optimized but quite power consumptive. In conclusion, all aims were accomplished and plans for further developments were initiated.

Table of Contents

Table of Figures

Chapter 1 – Introduction

An *Internet Radio* is described as an online radio which contains live or pre-recorded broadcasts from all over the world that are streamed over the internet [1]. There are appliances that provide internet-radio streaming, but there is a problem which brings more of a special purpose to this appliance. There is a requirement that principles of *Universal Design* need to be applied to tailor for easy access and usability by users with some particular, identified profiles of disabilities.

An internet radio in comparison to its other options (FM, AM, etc.) offers thousands of stations for listening and is not limited to its region/location. In addition, it offers finer audio quality. This is because the bandwidth is not limited by its broadcaster requirements and internet stations make use of more efficients codecs [2]. This codecs used allows for data to be compressed to allow faster transmission and decompresses data recieved. Example: BBC Radio 3 is encoded using AAC (Advanced Audio Coding) which specifically is a multi-channel perceptual audio coder that compresses audio signals and still maintains excellent quality. [3] With the internet becoming more available to people around the world, there are more advantages that an internet-radio has over other varieties of radios.

A general internet radio appliance may have design specifications that limit particular profiles of consumer individuals from having easy access to the appliance. To ensure that some profiled individuals have easy access, the design of an internet radio can be paired with an outer collective of designed appliances which makes it possible that by the collection of appliances, applications, 3rd party products, etc., there brings multiple pathways that brings universal access to the main appliance.

To provide *Universal Access* to the general appliance, it requires the concept of *Universal Design.* In regards to making any product or service, when applying principles of Universal Design, it must be ensured that there are no barriers that prevent the user from interaction regardless of their age, size, ability or disability [4]. This design strategy essentially makes the general appliance accessible to the chosen field of consumers, by meeting their ideal design needs and specifications.

The overall aim is to design, build and deploy an appliance which will have an overall design that provide pathways of universal access for the main appliance, the internet radio. Design choices will be motivated by by the selection of personas to be later discussed.

Chapter 2 - Technical Background

2.1 Universal Design

The ideal solution for an appliance to meet the project's requires a *Universal Design* approach to make an appliance or appliances that offer universal access to for the main product. Universal Design should ensure that the appliance is designed to adapt everyone in with this identified profile without any change additionally, especially for special needs [5].

2.1.1 Accessibility and Persona Based Design

Accessibility is concerned with a user's level of access to products and services [6]. The concept of accessibility allows for designers to implement tools and techniques that employ in creation of inclusive experiences, in this case, an internet radio appliance. The consideration of accessibility takes into consideration the user's abilities, challenges faced, the technology they are using and to engage with the appliance and the environment in which they do so [7].

To be able to deal with accessibility, there needs to be and understanding of the demands and issues faced by the consumer. A method of building this understanding is by the use of personas [7]. The use of personas are a common ground for reference in the design stage as they accentuate the needs required to provide accessibility. Along with the reference drawn out by the use of personas, it is a requirement by the design brief to implement concepts of universal design as guidance for our persona based designs.

2.1.2 Principles of Universal Design

To be able to evaluate a design, there are 7 principles that researchers, designers and enginers led by Robert Mace formulated [8]:

1. Equitable Use: The appliance is useful and appropriate for people of diverse abilities.
2. Flexibility in Use: The appliance has a wide range of individual preferences and abilities.

3. Simple and Intuitive Use: The appliance is easy to use and understand regards of the user's experience, knowledge, language skills or concentration levels.
4. Perceptible Information: The appliance communicates essential information in an effective manner to the user regardless of ambient conditions or the user's sensory conditions.
5. Tolerance for Error: The appliance minimizes hazards and the adverse consequences of accidental and unintended actions.
6. Low Physical Effort: The appliance can be interacted with efficiently and comfortably with minimum fatigue.
7. Size and Space for Approach and Use: The appliance's size and space is ideal for the user to approach, reach, manipulate and use regardless of the user's abilities.

The afformentioned principles detail universal design from different perspectives, where the consumer is placed in a position of primary consideration. With this design concept, products and services are able to be made for everyone to be used regardless of differences that can be found between all humans.

2.1.3 Existing Problems of Design in Internet Radio

For the wide range of consumers, the design of a radio appliance may not tailor for ease of access to all members in the consumer's field of interest. To be able to decide on personas which can be used for design options to implement universal access, it is essential to understand limitations and barriers that may be faced with these type of appliances. The barriers can be narrowed down to three categories [9]:

1. Barriers Releated to Computer to User Output
2. Barriers Related to User to Computer Input
3. Barriers Related to Understanding the Appliance

The first barrier relates to the limitations a user may have when trying to pass their input with the computer end. **Mobility** may be an issue for one when trying to give their input with a computer. The user may have little to no use of their hands, they may only be able to use one finger, they may be making use of equipment to point, etc. **Various Learning Disabilities**

may hinder some users, they may be hyper-sensitive to background noise, the user may be dyslexic and other sorts of learning difficulties that may hinder user input.

The second relates to limitations a user may face when trying to interpret the message/response a computer is passing. Regarding **Mobility**, users will not be presented with any challenges when it comes to screen output unless there may be a significant amount of navigation required to get to the output. Regarding **Blindness and Low Vision**, a user will experience challenges when trying to view the output passed via screen. Regarding **Hearing and Speech Impairments**, audio output will be a challenge to pass to the user. People with **Learning Disabilities** may need a more adaptive design to be able to make up for some limitations they may face with "Visual Impairments", "Hearing Impairments" and "Difficulties Interpreting Visual Material".

The last barrier relates with a user that has no prior experience of using a given application. They possibly may not have assistance from a third party to understand how the application works due to the same reasons in described in the previous barriers. This essentially would require a design in which all people with particular disabilities/limitations will be able to "independently" understand how the given application works will need to be drawn up.

2.1.4 Apple Universal Access and Persona Based Design Case Study

A perfect example of a design showcasing Universal Access is Apple's iOS operating system. Their built in Universal Access feature provides users with visual impairments, hearing impairments or physical disabilities the ease of access to computing and mobile ability with their design.

Figure 2.1 – Apple Universal Control

Apple's design approach for universal access touch on 3 personas:

Apple Universal Control

1. Vision: The design implements a speech facility with the ability that reads what happens on the screen and read text from the screen. Screen zooming, Inverse Colors (Dark Mode), display and text rescaling**,** braille display and on-screen simplicity are all features included in the design for the ease of access to comuting for visually impaired consumers [10].

2. Mobility: Voice control allows for a seamless process to operate the appliance, simplicity in touchscreen navigation, assistive touch and external hardware keyboards cater for consumers with demands related to their mobility [11].

3. Hearing: Exterior listening devices, hearing aids compatibility and sensory alerts lead the way for entry to computing and the full experience for consumers of hearing impairment [12].

Apple’s consideration for universal access relate to their success as being one of the biggest tech companies in the world and the having the number one largest revenue [13]. Their blueprint of their use of personas makes pathways for users that generally face limitations when using similar appliances. They have built a collective of appliances, Apple Hearing Aids, Apple AirPods and other peripherals that aim to allow for all consumers to access their headlined applances regardless of their abilities.

2.2 Internet Radio Appliance Hardware

The mandatory hardware required to build an *Internet Radio Appliance* would consist of the following components:

- A single-board microcomputer (e.g Raspberry Pi, BeagleBone Black, ASUS Tinker Board).
- A display unit (e.g an LCD display, OLED display, touch-screen, etc).
- Push Buttons and Rotary Encoders for the user interaction.
- An Audio Amplifier.
- Power Supply Unit.

2.2.1 Embedded Systems and The Raspberry Pi

There are many embedded systems that can be configured to operate as the appliance to be designed, with some being more suitable than others.

The BeagleBone Black is a low-cost community-driven platform that supports Linux and Android and is suitable for hobbysists and developers. The board consists of a 1GHz ARM processor, 512MB of DDR3 Ram, 4GB of on-board flash storage, a 3D graphics accelerator and dual 32 bit microcontrollers. Connections given allows for 2 46-pin GPIO headers [14].

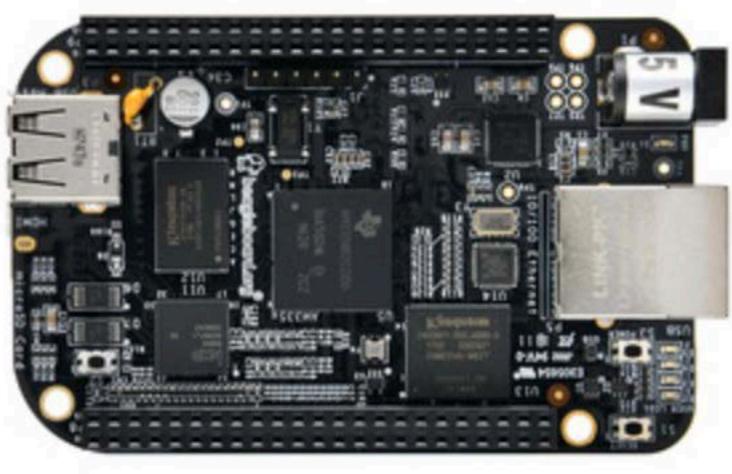

Figure 2.2 – BeagleBoard Black

The ASUS Tinker board is an ARM-based single board computer with market-leading performance. The board consists of a 1.8GHz ARM based CPU, 2GB of dual channel

DDR3 RAM and a 600MHz GPU. Understandibly, this is one of the best performace-based microcomputers on the market [15].

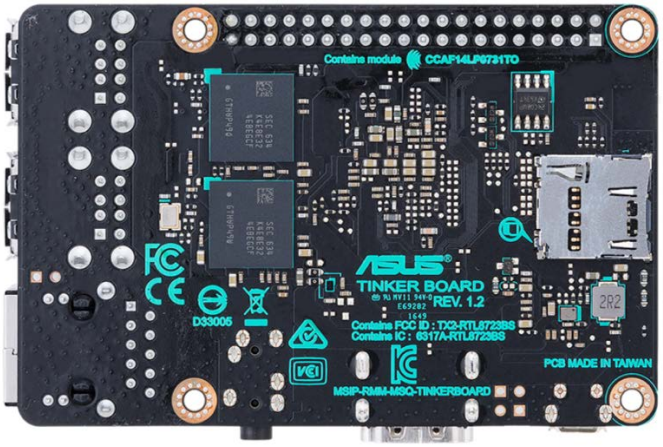

Figure 2.3 – ASUS Tinker Board

For the sake of this project, the Rapberry Pi 3 B+ will be the selection for the microcomputer. The Raspberry Pi is a small and affordable single-board microcomputer developed in the United Kingdom by the Raspberry Pi Foundation [16]. It was developed originally with the intention of teaching basic software programming in school but has not become a popular choice for engineering projects.

The Raspberry Pi is described to be the size of a credit card bordered with input and output ports. There are 2 Models for the Raspberry Pi:

1. Model B which is the first generation released in Feb 2012.
2. Model A which was a cheaper version released after Model B.

There is also another Model (Model B+), released in 2014 which is a new and improved design of Model B.

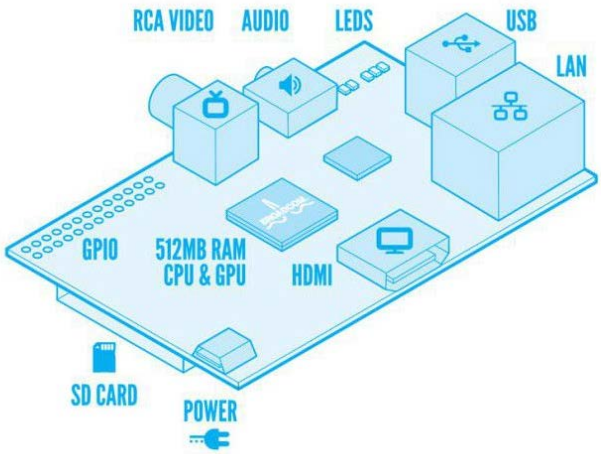

Figure 2.4 – Raspberry Pi Model B Diagram

The Raspberry Pi Model B's board consists of a chip that houses the Central Processing Unit with an *ARMv6 Instruction Set*, Broadcom 1080p30 Graphics Processing Unit, Digital Signal Processor, Secure Digital Random Access Memory (either 256mb or 512mb) and a USB 2.0 port. The I/O ports on the board include a HDMI port, CS Camera port, DSI Display Port, 4 USB 2.0 ports, 40 GPIO pins, Micro USB Power Input port, 3.5mm Audio Ouput Jack and a LAN port. The board also includes Bluetooth 4.1 and WiFi.

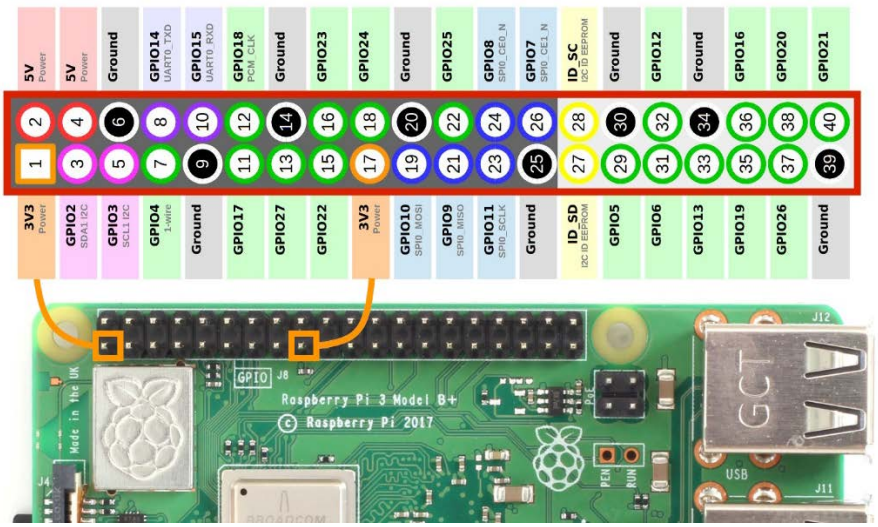

Figure 2.5 – Raspberry Pi GPIO Header

A feature on the Raspberry Pi that stands out is the two rows of GPIO (General Purpose Input/Output) pins. The board contains 40 pins (26 on the early models) [17]. The GPIO pins on the intergrated circuit allow for digital input or output functions. Other than the voltage and ground pins, these pins will have no predefined purpose

which allows the development of functions for each of these pins, examples being [18]:

- Controlling LEDs
- Reading Switches
- Controlling Sensors

The possibilities of functions that the GPIO functions can carry out are endless in bringing a physical interface between the Pi and the outside world interactions.

2.2.2 Display Devices in Computer Graphics

The Raspberry Pi's visual capabilities makes it a suitable choice for a multimedia device. For this project, display may not be an essential requirement but is beneficial for displaying the metadata and information for radio stations. Examples of displays that are compatible with the Raspberry Pi are:

- **Character LCD Displays**
- LED Displays
- HDMI Display
- OLED Display

Liquid Crystal Display (LCD) screens are an array of small pixels which are manipulated to display information [19]. Character LCD displays exist in many appliances today such as music keyboards, calculators, old gaming consoles, etc coming in various sizes as small as an 8x1 display. The affordable price for these displays make them a very popular choice for engineering projects.

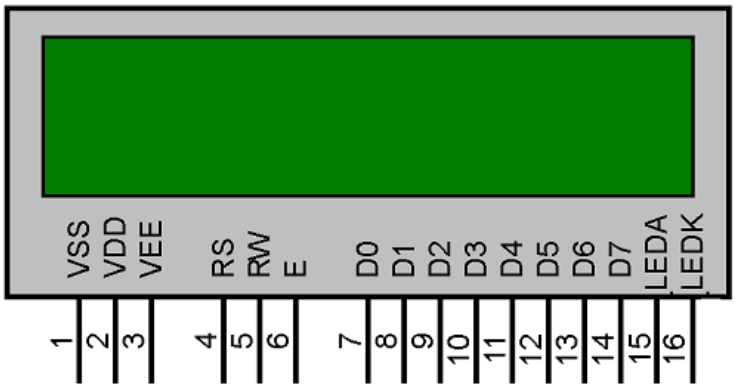

Figure 2.6 – LCD Display Module Pinout

Generally most Character LCD Displays have 14 pins. Figure 2 illustrates the 14 pins of an LCD Display Module's pins. Pin 1 is for ground and Pin 2 is for +5V. Essentially these 2 pins provide power to the display module. Pin 3 is a receiver pin for analog voltage which is used to set the contrast of the screen. In majority cases this is controlled by a potentiomenter.

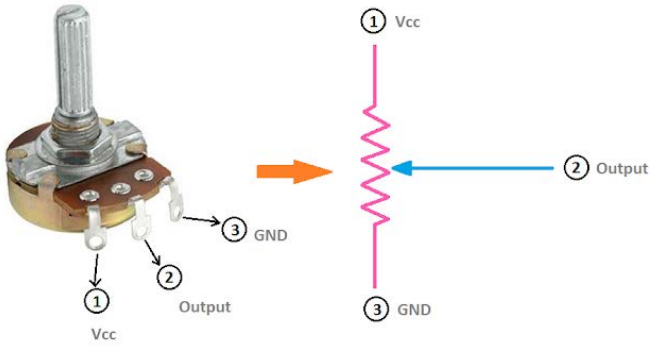

Figure 2.7 – Potentiometer Diagram and Circuit

Figure 3 indicates the 3 pins of a potentiometer where the middle pin for contrast would be connected to pin 3 of the display module and the 2 other pins would go to the ground and 5V pins of the display module. As the potentiometer is adjusted the signal will between 0V and 5V which adjusts the contrast.

Pin 4 is register select which allows for the following 2 types of data:

- Text Data (ASCII Characters)
- Instructions (turn on cursor, turn off display, display graphic, etc.)

Pin 5 is read/write which allows information to displayed on the screen or retrieved from the screen. For information to be read, voltage must be high and then low for writing. Pin 7 to 14 are the data bits which create an 8-bit binary number, each being set either high or low. Pin 6 is the enable pin. The enable pin's function is to send all the information set by the other pins to the screen.

2.2.3 Audio Amplifiers

The Raspberry Pi does not have an on-board built in audio amplifier but features a stereo 3.5mm socket on both the A and B models. Since the on-board audio jack is said to have

limitations, it is ideal to opt for an external amplifier. There are a variety of options that can be used when selecting an audio amplifier:

- Set of PC speakers with an amplifier.
- Digital Audio Converter (DAC) and Amplifier
- Dedicated AB or Class D Stereo Amplifier
- Bluetooth Speakers
- Wired Headphones

PC speakers and Wired Headphones can be connected directly to the on-board audio jack or USB port on the Pi board and the bluetooth support from the board allows for bluetooth speaker devices to be connected.

The sound output from the audio jack of the Raspberry Pi is known to be limited in regards to sound quality and output, hence a DAC may be used. DAC utilize Pulse Code Modulation which is a digital scheme for transmitting analog data and converting it to digital form [20]. PCM is based on the sampling theorem. Each analog sample is assigned a binary code. The analog samples in most cases are referred to as Pulse Amplitude Modulation samples [21]. The binary code being the output is a power of 2 bits. Once the binary output reaches the receiver PCM converts and processes the binary code back into an analog waveform.

In most cases a DAC card would have an in-built amplifier (of Class D-type amplification most of the time).

2.2.4 Power Supply Units

The Raspberry Pi recommends 5V @ 2A as a minimum for stability [22]. The power supply for the appliance is very important as low voltage can results in components not being able to function. There are 2 safe approaches to powering the Raspberry Pi:

1. Micro USB Port (Can be powered by a simple Micro-USB cable)
2. GPIO Ports (Pin 2/Pin 4 allows for positive voltage and Pin 6 is ground)

2.3 Background of Relevant Softwares

The Raspberry Pi is a linux-based embedded system. For building an Internet Radio Appliance, there are various softwares compatible with the Raspberry Pi and Linux OS that can be used to configure the device to act as an Internet Radio. There are also some software concepts thst need to be understanded to build the software.

2.3.1 Daemon

Daemons are utility programs on Linux based systems that run silently in the background to monitor and take care of certain subsystems to ensure that the operating system runs properly e.g. a printer daemon would monitor the printing services [23]. This is an important concept to note as it may be a requirement to make utility programs for the software side of the application.

2.3.2 Music Player Daemon

MPD is a server-side audio application, available on Debian based Linux platforms, such as Raspbian. MPD allows for playback of a number of audio formats, including WAV, FLAC, and MP3, and supports both FIFO, and HTTPD streaming which allows for internet station urls to be streamed. The MPD also comes with configuaration files that allow for environmental objects to be set such as the the audio output device, music directory, audio format and more [24].

2.3.3 O!MPD

O!MPD is an open source MPD client based on PHP and MySQL. It is a web user interface that control MPD and browse through the music library [25]. The interface has a responsive design that works with all devices, a user can search browse through the library of contents and a user can play any content from the MPD. For this appliance, this is a perfect solution to control the configuration and operation of the appliance from a mobile phone or laptop.

2.3.4 LIRC

Linux Infrared Remote Control (also referred to as LIRC) is a software that allows for the decoding and sending of infra-red signals from supported remote controls [26]. The software runs a daemon that decodes IR signals received by the Raspberry Pi. These signals are passed to the Pi using a capture device that is paired with the board. Once the info is received, it can then provide commands to hardware and its components. An application that works in conjuction with this software is IR-Record. This software records controller inputs and maps it to controller key commands from LIRC.

Chapter 3 - Design of Internet Radio Appliance

This project consists of three main design phases. The first phase entails around the design of the accessibility-based persona. The second phase, requires the hardware design, and lastly, the software design. Outline are the proposed solutions to these three phases.

3.1 Design of Accessibility Personas

The ideal proposal for a solution to this project would display the concept of "Universal Access". To apply these concepts, three identified personas were designed to give an idea of the design specifications and features that may need to be added and the considerations to be made when choosing hardware and software solutions for this project. The personas given do not necessarily have the purpose of covering a specific individual but to cover a general consumer group that may have the same abilities and may require the same considerations to be made in the design process. *The keywords/phrases marked in bold relate to the Principles of Universal Design stated in Section 2.1.*

Name	Jack
Could Be	A Wheelchair User
Appreciates	Appreciates appliances that require less physical effort
Could Use	A Remote Control, Portable Appliance
Avoid Using	Appliances that require physical effort e.g. having to move between the appliance and your resting point.

Table 3.1 – Persona 1

The first persona, Jack, requires the consumer's mobility to be put into consideration. As stated, this a person that cannot easily move from one place to another and may need an appliance that allows them to operate it from the position they are settled at. Jack would benefit from a design that is **Portable** or a design that requires **Low Physical Effort.**

Name	Rachel
Could Be	Hearing Impairments
Appreciates	Compatibility with external listening devices
Could Use	Compatibility with listening devices, e.g hearing aid, headphones (wireless preferably)
Avoid Using	Appliances that require a lot of buttons and are complicated just for the general operations.

Table 3.2 – Persona 2

The second persona, Rachel, requires **Perception of Information** to be placed as a consideration. The appliance in design cannot provide any functionality to a user that is completely deaf but regarding an individual that still has hearing ability, the appliance may still be paired with external appliances that can bring accessibility for the described persona.

Name	Mark
Could Be	Visually Impaired
Appreciates	Programs that do not rely soley on display for info
Could Use	Audio Display Reader
Avoid Using	Appliances that rely soley on display for information

Table 3.3 – Persona 3

The third persona, Mark, requires the consumer's **Perception of Information** to be put into consideration for the design process. This persona mentioned relates to someone that may not have the best of abilities with vision and would prefer **Flexibility in Use** so that the information passed from the appliance can be conducted through audio format.

3.2 Hardware Design

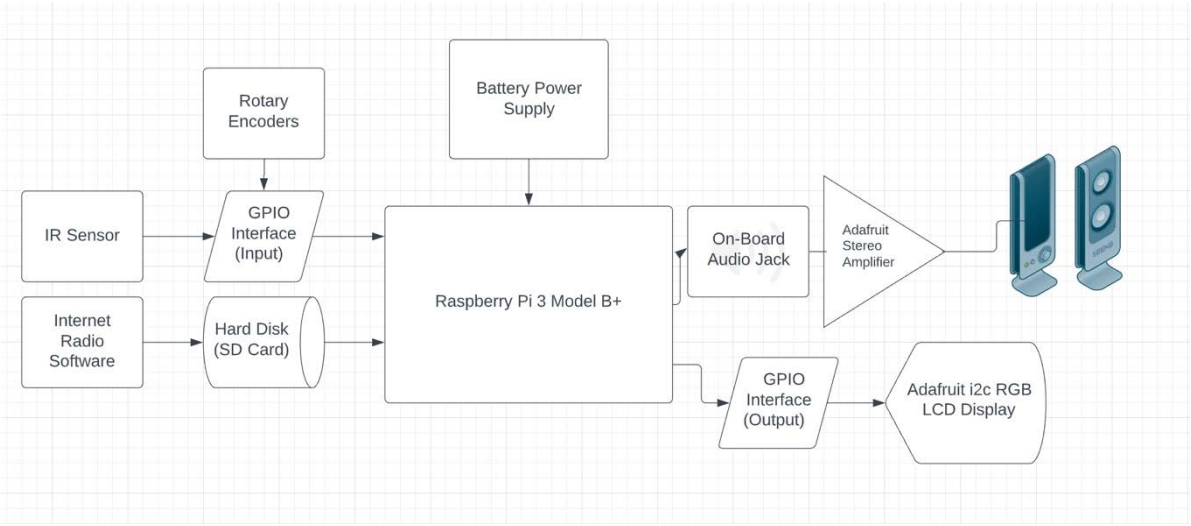

Figure 3.4 – Hardware System Block Diagram

The proposed solution for the hardware design is centered around the Raspberry Pi 3 Model B+. The internet radio software runs on the device which is stored in the slotted SD Card for storage. The utilizes an external stereo amplifier and a set of stereo speakers for audio output and an Adafruit LCD Display for visual output.

The appliance will be operated via two rotary encoders connected to the GPIO inputs and also by remote controller that has been configured with an infra-red sensor. The entire appliance will be supplied with a battery-based power supply. Connections between appliances are made via jumper wires (male to male, female to female, male to female, etc.). Table outlines the necessary hardware and quantities for the hardware design.

Components	Quantity
Raspberry Pi 3 Model B+	1
Adafruit RGB LCD Plate	1
Adafruit MAX98306 Stereo Amplifier	1
Four-inch Loudspeaker	2
Rotary Encoder	2
Battery Power Supply	1
TSOP1738 IR Reciever	1
IR Remote	1

Table 3.5 – Required Hardware Components List

3.3 Software Design

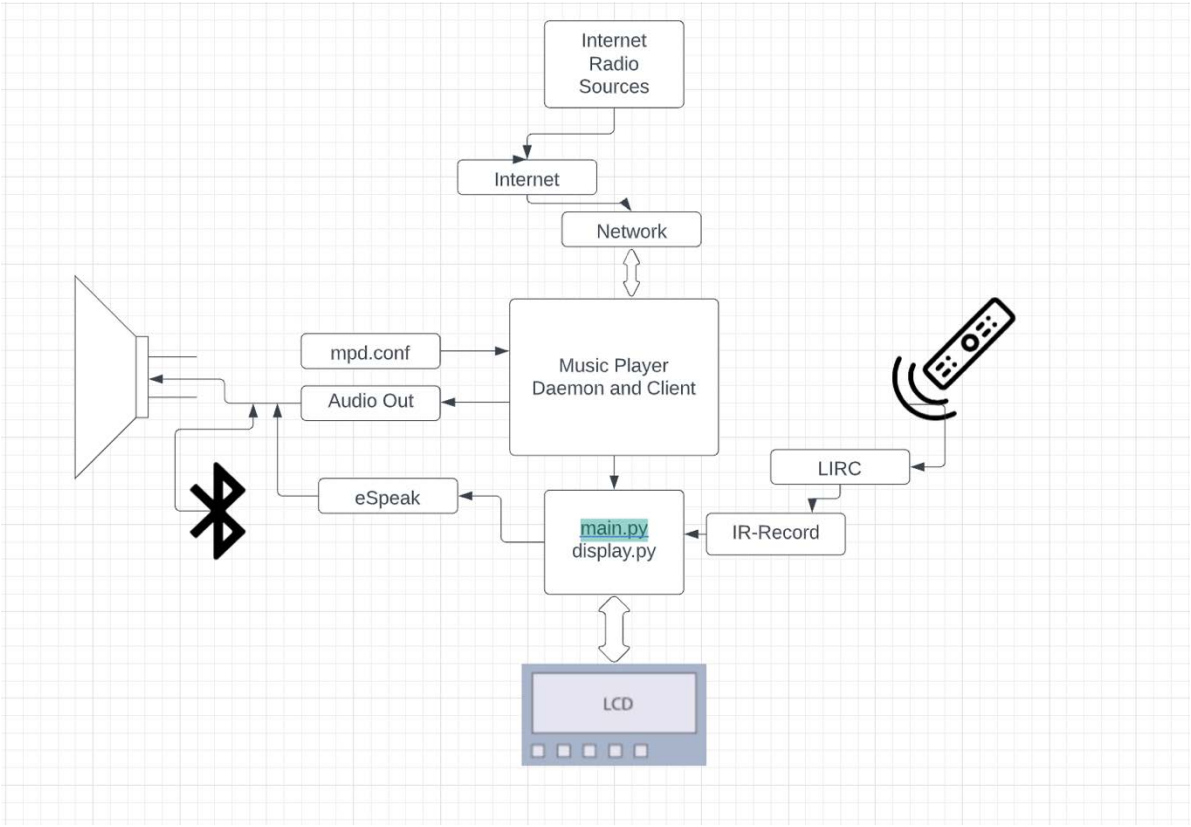

Figure 3.6 – Hardware System Block Diagram

The Internet Radio proposed software solution utilizes open source softwares and a python application that would be run as a background daemon when the Raspberry Pi is turned on. The Python application was daemonized by creating a cron job that calls the script.

The first essential software needed for the design was Music Player Daemon. This was responsible for making it possible for audio to be played, organizing a playlist and maintenance of the music database. To be able to access this software, the client, MPC, also was installed.

For consumer's thay are visually impaired or blind, it was proposed that a speech facility would be implemented to allow them to have the the displayed information to be perceived via audio. The software, eSpeak was used to solve this problem.

To allow operability of the appliance with a remote control, LIRC was included in the software design solution. After the necessary testing, ir-record, a utility program that records controller inputs, was used for configuring the buttons from the remote control.

A requirement necessary was for podcasts to be possible to be streamed from the appliance. The proposed solution for this was by implementing an open source software, Raspotify. This software allows the appliance to run as a spotify client allowing users to connect their spotify to the Pi and listen to podcasts that can be found on Spotify. According to eMarketers, Spotify is the top application used by podcast listeners with around 28.2 million monthly listeners from the US [27]. The problem being that, you require a Spotify account to be able to to use this feature may limit some consumers from accessing the podcasts feature of the appliance but from statistics it clearly shows that majority of podcast listeners already own a Spotify account.

The proposed proposed application application is collection of two .py applications. A display class which is responsible for driving text to the LCD Display and the main application that is responsible for the logic operation of the radio appliance.

Chapter 4- Implementation and Testing

4.1 Implementation of Hardware Design

Outlined is the implementation of all the afformentioned design choices to create the hardware necessary for the design of the appliance.

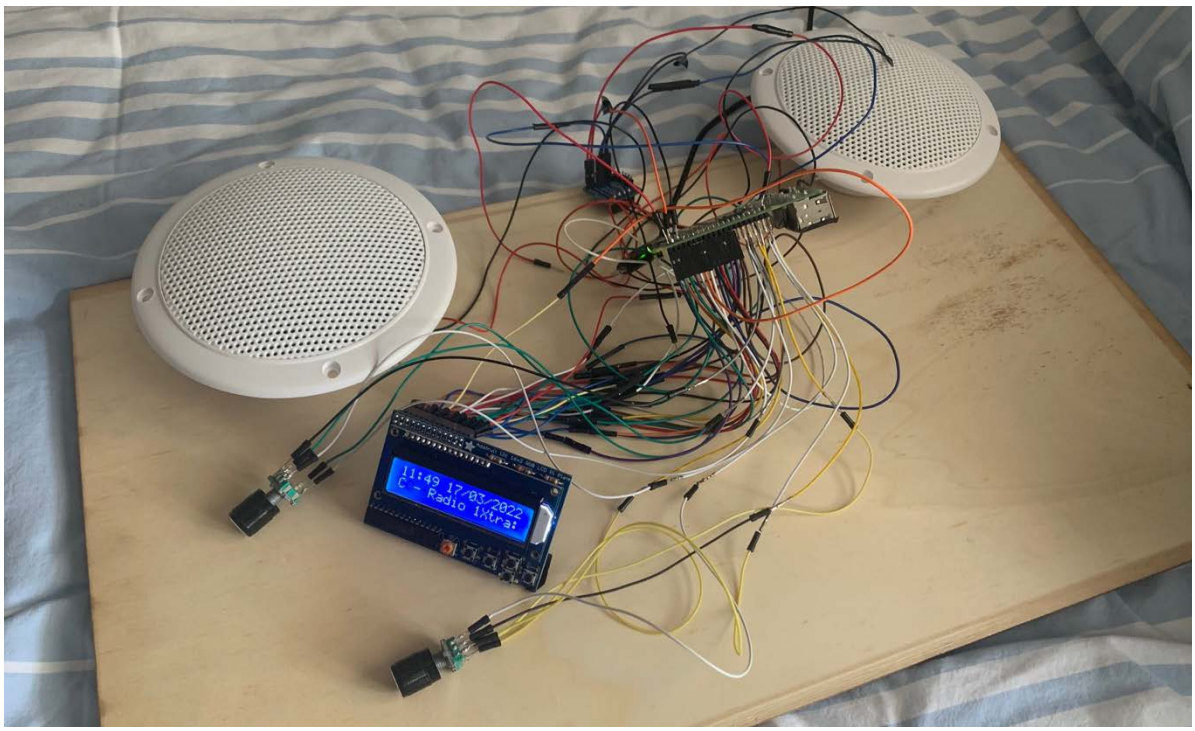

Figure 4.1 – Implemented Hardware Design

4.1.1 Implementation of Audio Hardware Design

The hardware design solution presents the implementation of a single Raspberry Pi 3 Model B+ with an Adafruit Stereo 3.7W Class D Audio Amplifier which gives output to the stereo speakers.

Figure 4 is the top view of the Stereo Amplifier. There are 9 inputs but for this design only 6 will be made use of:

- Voltage Direct Current
- Ground
- Both the Inverting and Non-Inverting Audio Right Inputs
- Both the Inverting and Non-Inverting Audio Left Inputs

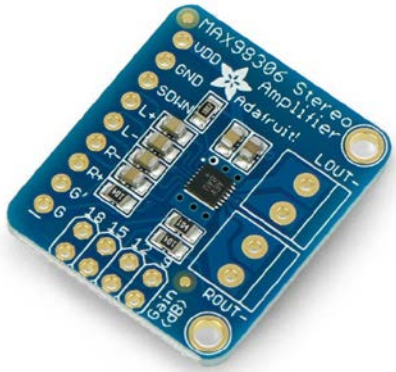

Figure 4.2 – MAX98306 Stereo Amplifier Top View

Making Reference to Figure 2 that showcases the GPIO headers of the Raspberry Pi, the input of the Voltage Direct Current comes straight from pin 4 of the Raspberry Pi which supplies 5V of direct current. Ground comes from pin 39 of the GPIOs.

For the connections releating to audio output of the raspberry pi for the input of the amplifier, the easiest approach would be to use two sets of GPIO pins but theis couls cause conflict later if there were a lack of pins unused for future development. Instead, the Inverting and Non-Inverting points were wired to the on-board audio jack. From the schematics of the Raspberry Pi, it was discovered that PP25 related to the left audio output of the audio jack and PP26 relates to the right. PP6 was then used for the two inverting audio inputs.

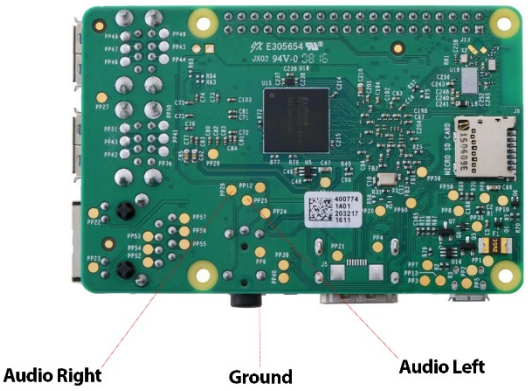

Figure 4.3 – Audio Jack Connection Points

Figure 6 demonstrates the wiring of the amplifiers output to the two speaker's inputs. The amplier has 4 output connections for audio (L+, L-, R+, R-) which connect to the corresponding input for the two loudspeakers.

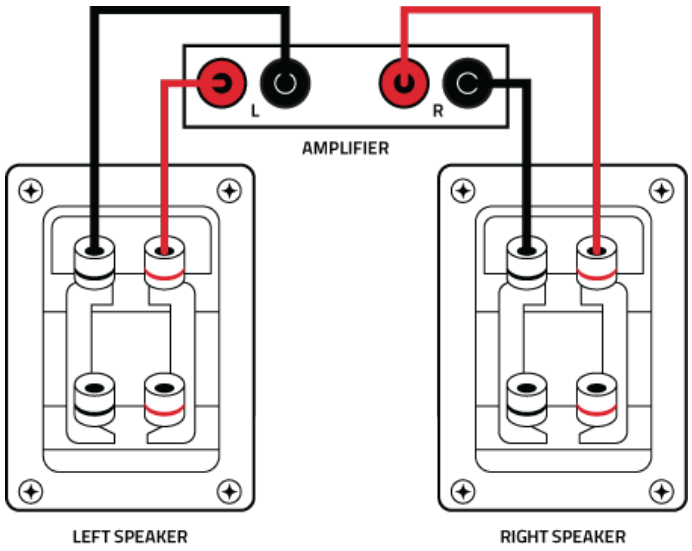

Figure 4.4 – Stereo Speakers and Amplifier Wiring

A selection of stereo speakers were proposed rather than a mono speaker. The sound quality of a stereo speaker is deemed to be fuller and have a more dynamic soundstage than a mono speaker. It also has the ability to allower a listener to be able to pinpoint sounds coming from multiple directions. This factor itself is key in the listening experience as most radio stations of nowadays use ominibidirectional microphones which allows which allow for this listening experience. Having two audio signals may also give an impression that the audio sounds louder which may give a better experience for the user [28]. There is no ultimate reason for one being significantly better than the other but the stereo setup may add to the consumer's listening experience.

4.1.2 Implemention of Display Hardware Design

This appliance does not require any video or visual graphics to be displayed, hence why a Character LCD Display was selected. The component to satisfy this selection is an Adafruit RGB LCD with a keypad. This plate when put together allows for the control of the 16x2

Character LCD, 3 backlight pins and 5 push buttons. This hardware required that the components are soldered to the plate for assembly.

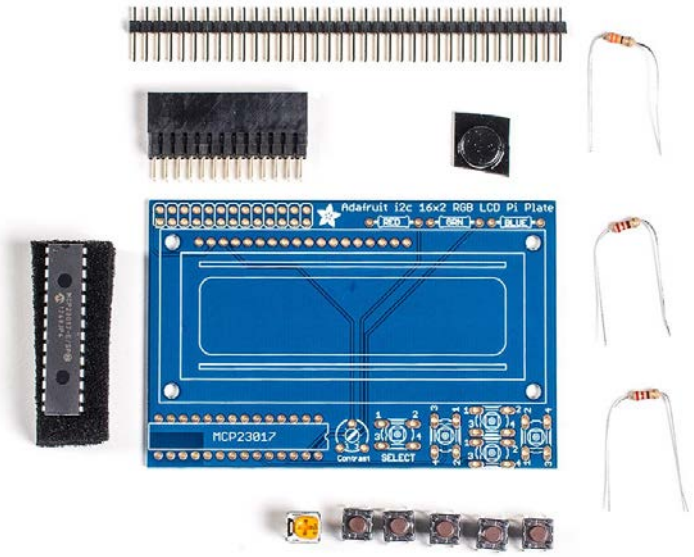

Figure 4.5 – Adafruit i2c RGB LCD Pi Plate Disassembled

After the assembly of the LCD Plate, the male pins were directly wired to the female pins from the GPIO interface. This piece of hardware gives for the display of the metadata from the currently playing station or other media.

Figure 4.6 – Adafruit i2c RGB LCD Pi Plate Assembled

4.1.3 Implementation of Operative Hardware Design

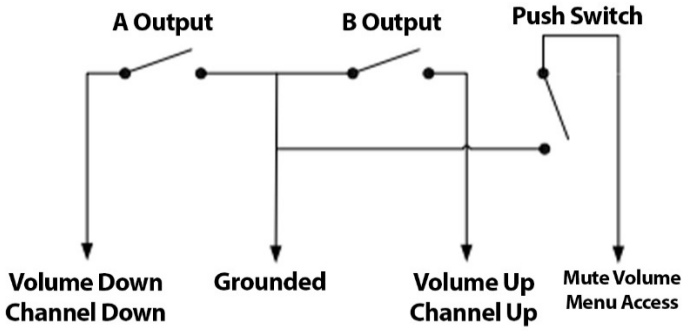

Figure 4.7 – Diagram for Rotary Encoder Wiring

The proposed design solution was the use of two *Incremental Rotary Encoders*. They provide cyclical output when the encoder is rotated and also give an out when the encoder is pressed/clicked [29]. The rotary encoders have 5 pins (A, B, Ground, Switch 1 and Switch 2) which are wired to the Raspberry Pi's GPIO for their functions to be carried out.

Rotary Pins	Raspbertty Pi GPIO	Function
A Output (Rotary 1)	Pin 40 (GPIO21)	Volume Up
B Output (Rotary 1)	Pin 38 (GPIO20)	Volume Down
Common Ground (Rotary 1)	Pin 34 (GND)	Ground
Switch 1 (Rotary 1)	Pin 36 (GPIO16)	Mute Volume
Switch 2 (Rotary 1)	Pin 30 (GND)	Ground
A Output (Rotary 1)	Pin 29 (GPIO5)	Channel Up
B Output (Rotary 1)	Pin 31 (GPIO6)	Channel Down
Common Ground (Rotary 1)	Pin 39 (GND)	Ground
Switch 1 (Rotary 1)	Pin 32 (GPIO12)	Switch to Radio/Spotify
Switch 2 (Rotary 1)	Pin 39 (GND)	Ground

Table 4.8 – Rotary Encoder Wiring and Operation

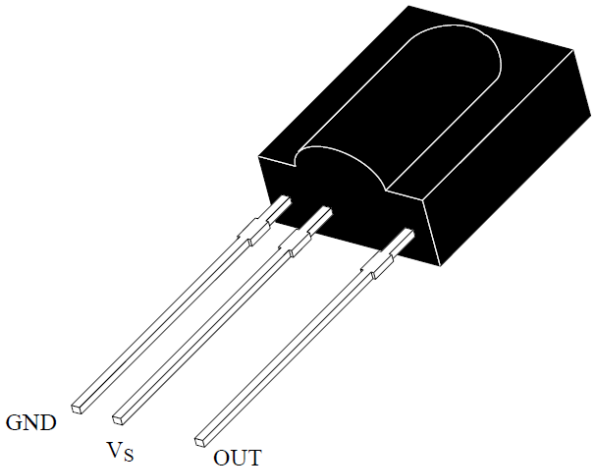

Figure 4.9 – TSOP1738 IR Reciever Pinout

An additional operational component is the Infra-Red sensor. The model of the receiver is the TSOP1738 Reciever. The specifications of the wiring of the component is displayed below in the table. The receiver is powered by the 3.3V supply from the Pi and the output is wired to GPIO5.

Sensor Pins	Raspbery Pi GPIO
GND	Pin 39 (GND)
VS	Pin 1 (3.3V)
OUT	Pin 29 (GPIO5)

Figure 4.10 – IR Reciver to RPi Connection

4.1.4 Power Supply

The Raspberry Pi 3 Model B+ can either be powered by the GPIO pins or via a micro-usb cable. To make portability a possible implementation in this device, it was proposed to go with a battery powered device.

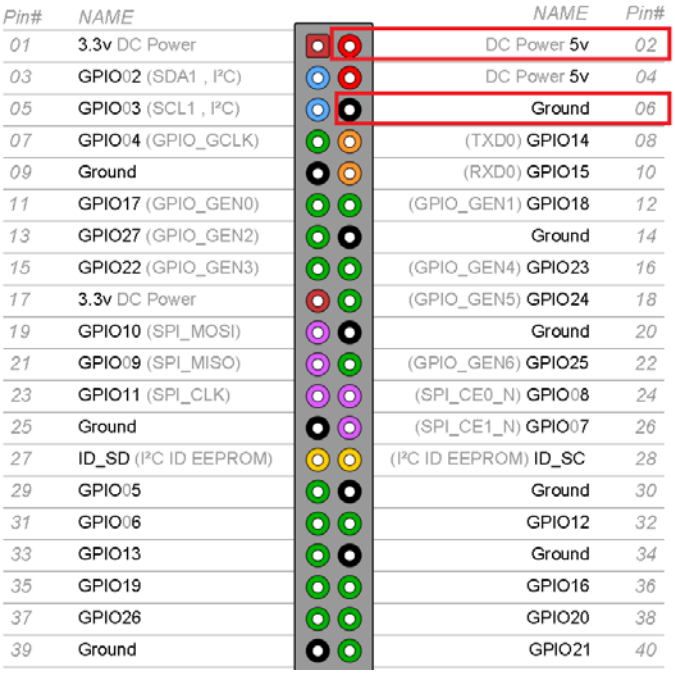

Figure 4.11 – GPIO Pins for Power Supply

The Raspberry takes a DC Power of 5V and the battery in use is a 9V battery (6x AA Batteries). A step down voltage converter was put in place to ensure that the Raspberry Pi would get the safe suggested 5V DC. It also ensures that there would be a maximum of a 1.5A output. After the testing stage, it was not feasible to power the appliance with commercial consumer batteries. The power supply was then switched to a rechargeable power bank supply connected to the micro-input.

4.2 Implementation of Hardware Testing

To ensure the development of an efficient design, characterestic of the appliance needed to be subject to testing. Some testing stage was executed by the use linux-based softwares and cli commands.

To conduct testing of hardware and extract results, Putty was utilized. Putty is a Windows based, Telnet and SSH client. This application is used for the SSH client it provides as a service. SSH allows for the logging in to a multi-user computer from another computer, over a network [30]. This allows instrutions to be carried out on the Raspberry Pi from a client device. Some of the services for testing are outlined below.

4.2.1 System Performance Testing

Conky is a system monitor for the Raspberry Pi, for monitoring system variables such as the CPU status, memory, disk storage, temperatures and more. These collected results will then be necessary to ensure the safety of the appliance components and the users of the appliance.

4.2.2 Wireless Connectivity Testing

Wireless Connectivity requires testing regarding the WiFi and Bluetooth. With the components being enclosed in a box, it is a consideration to be made that the wireless features are still effective with the surrounding barriers.

5 WiFi Download/Upload speed tests were conducted for the overall network performance to be presented. 10000 Latency tests between the appliance and https://google.com to track for any packets lost. Both operations were conducted using the listed commands. Both tests were executed with and without the case.

```
Speedtest-cli // used for calculating Down/Up speed
Ping www.google.com // used for latency tests
```

Listing 4.12 – WiFi Command Test

The Bluetooth range was executed through physical testing. Estimated distances between the appliance and Bluetooth audio device, to discover the range at which there is a distortion in the audio signal. The device used for Bluetooth audio were Apple Airpods which are standardised to be capable of 10m range. This tests was executed with and without the case.

4.2.3 Power Supply Testing

To mathematically compute the the specifications of the required power supply, the current drawn from the raspberry pi was recorded with an ampmeter. A 10% surplus margin of error is applied to the recorded current. The number of hours required for optimal usage is multiplied by the calculated current to discern on a value for the desired mAh specification for the choice of battery.

```
X mA = (Current Drawn by Raspberry Pi) x (1.1)
Y mAh = (X mA) x (Number of Hours Operating)
```

Listing 4.13 – Battery Requirement Calculation

4.2.4 Display Functionaility Testing

To test the LCD display, the test program was written in Python to test the transmission RGB and display of text from the Pi to the display unit. The result displayed the simple text “Hello World” and colored background.

```
import board
import busio
import adafruit_character_lcd.character_lcd_rgb_i2c as
character_lcd
lcd_columns = 16
lcd_rows = 2
i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns,
lcd_rows)
lcd.color = [100, 0, 0]
lcd.message = "Hello\nCircuitPython"
```

Listing 4.14 – Python Display Test Program

4.2.5 Audio Functionaility Testing

To test the stereo speakers and amplifier functionality is tested by running the *speaker-test* command in the Pi terminal. The expected result of a fully functional amplifies sound alternating between the left and right speakers.

```
speaker-test -c2 -twav -l7
```

Listing 4.15 – Linux Speaker Test Command

4.3 Implementation of Software Design

To implement the proposed solution, the following outlined software prerequisite were required:

- Music Player Daemon
- Music Player Client
- eSpeak
- GPIOZero
- Adafruit Character LCD Python Module

4.3.1 MPD Setup

MPD and MPC were installed from the Linux command line. The mpd configuration file needed to be manually configured for audio output because of corruption issues due to the Bullseye operating system. Listing 1 outline the ammendments to the configuration file.

```
audio_output {
  type            "alsa"
  name            "On Board Audio Jack”
  device          "hw:0,0"
  mixer_type      "software"
  mixer_device    "default"
  mixer_control   "PCM"
  mixer_index     "0"
}
```

Listing 4.16 – Python Display Test Program

An m3u was created that stores the playlist of internet radio stations. The MPC client allows for the following operations:

```
Mpc play – play operation
Mpc pause – pause operation
Mpc next – next operation
Mpc prev – previous operation
Mpc add – add to playlist
Mpc load – load playlist
```

Listing 4.17 – Music Player Client Operation

These operations were implemented in the Python application by making functions that call these commands within the application. This was implemented by the use of the OS module in python which provides functions for interacting with the operating system [31].

4.3.2 Radio Program

Main.py is the top level for the software program which provides all the logic for operating the radio. As displayed in Figure, there is communication between the program and the LCD Display, eSpeak functionablity, Alsamixer, Music Player Client, Rotary Encoders and LIRC.

Regarding the communication with the rotary encoders, this is achieved with the a Python module, GPIOZero. This allows allows for an interface to be established between the Python application and GPIO devices [32]. The table below initializes the Rotary Encoder GPIO pins with a variable name. GPIOZero provides a function that allows the use of the function **when_pressed** which allows for a function to be called when the GPIO pin is active ‘high’. These functions relate to the logic operations of the radio device.

```
buttonForMute = Button(16)
buttonForVolumeUp = Button(21)
buttonForVolumeDown = Button(20)
buttonForNext = Button(5)
buttonForPrev = Button(6)
buttonForSwitch = Button(12)

buttonForVolumeUp.when_pressed = self.increaseVolume
buttonForVolumeDown.when_pressed = self.decreaseVolume
buttonForMute.when_pressed = self.muteVolume
buttonForNext.when_pressed = self.nextStation
buttonForPrev.when_pressed = self.previousStation
buttonForSwitch.when_pressed = self.switchSource
```

Listing 4.18 – Rotary Encoder Software Implementation

The application communicates with the LCD display through display.py. The application initializes the LCD display by the use of generic Python supplied from the Adafruit docs [33]. A function is designed to allow messages to be passed through and displayed on both lines. The function takes two parameters, the first and second line and uses the message function from the initialized LCD to display these lines. This function is called within main.py for the display of:

- Currently Playing Track
- Volume Display
- Date and Time

```
def out(self,lcd_line_1,lcd_line_2):
        # wipe LCD screen before text displayed
        self.lcd.clear()
        while True:
            self.lcd.message = lcd_line_1 + lcd_line_2
```

Listing 4.19 – LCD Software Implementation

To operate with the Music Player Client, the OS Python module was used to excute system commands within the application. This allowed for the generic audio player commands to be executed:

- Next Station
- Previous Station
- Play/Pause Media
- Stop Player

```
def play(self):
    os.system("mpc play")

def pause(self):
    os.system("mpc pause")

def stop(self):
    os.system("mpc stop")

def nextStation(self):
    self.play()
    os.system("mpc next")
    self.speakCurrent()

def previousStation(self):
    self.play()
    os.system("mpc prev")
    self.speakCurrent()
```

Listing 4.20 – Music Player Client Implementation

For the implementation of eSpeak, a function was designed to communicate the contents of the display to the user via audio means. Similar to the implementation of the mpc operability, the OS Python module is used to call the espeak function. The speech function when spotify is running will read the track name or podcast episode and for the normal radio mode, will read the current radio station.

```
def speakCurrent(self):
    if spotifyRunning:
        track = self.getSpotifyTrack()
        os.system("espeak -ven+f2 -k5 -s130 -a20 " + track +"
--stdout | aplay")
    else:
        currentStation = self.run_cmd("mpc current")
        os.system("espeak -ven+f2 -k5 -s130 -a20 " +
currentStation +" --stdout | aplay")
```

Listing 4.21 – eSpeak Implementation

To implement the control of the appliance's volume levels, system commands were called from functions that allowed for an increase/decrease in volume, muting the volume and retrieving the current volume levels.

```
def increaseVolume(self):
    if buttonForVolumeDown.is_pressed:
        os.system("amixer sset Digital 3%+")
        self.displayVolume()

def decreaseVolume(self):
    if buttonForVolumeUp.is_pressed:
        os.system("amixer sset Digital 3%-")
        self.displayVolume()
        sleep()
        self.displayCurrent()

def muteVolume(self):
    if buttonForMute.is_pressed:
        os.system("amixer set Master toggle")
        self.displayVolume()
        sleep()
        self.displayCurrent()
```

Listing 4.22 – Volume Control Software Implementation

The software allows for a switch between two modes (Radio Stations and Spotify). Spotify allows for the streaming of podcasts and songs through an open source software, Raspotify. The switch source function is accesed by the push on the rotary encoder. If Spotify is running, then all the processes relating to Spotify will be ended and the Music Player Client will startup with the radio stations. In the parallel case, if sources are switched, the Music Player Client will stop and the Spotify client will begin operation alongside with a journal that keeps track of the current metadata for Spotify which has the use case of displaying the current track/podcast episode on the LCD display.

```
def switchSource(self):
    global spotifyRunning
    if spotifyRunning:
        os.system("killall journalctl")
        os.system("sudo systemctl stop raspotify.service")
        os.system("mpc play")
        spotifyRunning = True
    else:
        os.system("mpc stop")
        os.system("sudo systemctl start raspotify.service")
        self.startJournalWatch()
        spotifyRunning = False
```

Listing 4.23 – Spotify Implementation

LIRC was used to setup the infra-red sensor and remote which allowed for pulses from the remote to be received by the sensor. IR-Record, another open-source software, was then used to to map these remote control pulses with system commands related to Alsamixer for volume control, MPC for tracks navigation and eSpeak for speech facility.

After the creation of the Python application, a new Cron job was created. The "crontab" was modified and the following line from Listing was added to the file to ensure that the script would always run in the background at the booting of the Pi.

```
@reboot python /home/pi/MyScript.py &
```

Listing 4.24 – Cron Job Creation

4.3.2 Web Interface

O!MPD was used to develop a web interface that allows for control over the Music Player Client. The application's implementation constantly checks the current playing track allowing the track to be automatically updated when changed through the web interface. The web interface is accessible through entering the IP address of the appliance into a browser.

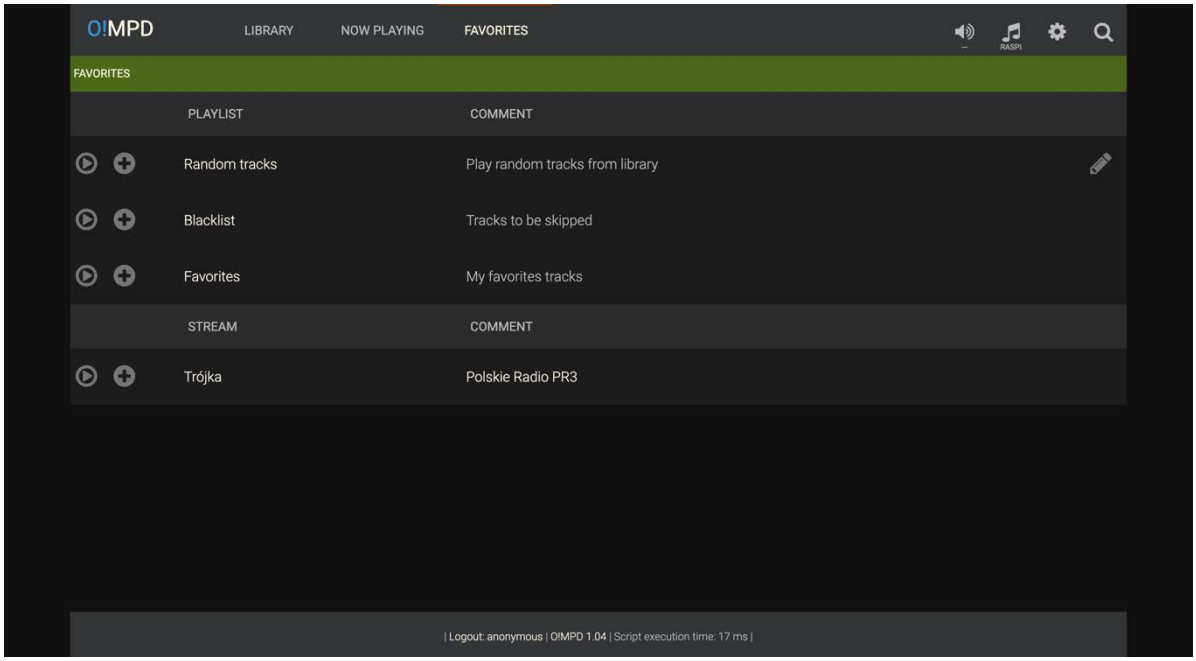

Figure 4.25 – O!MPD Web Interface

Chapter 5 - Results and Discussion

The following results and discussions formulated from the extraction of data from the testing stages of the hardware and software. Factors of accessibility are also evaluated on the basis whether they meet the requirements necessary for the constructed personas in the design stage.

5.1 System Performace Results

Conky was used to test the hardware system performances. The following system stats were extracted and recorded:

- CPU Usage
- Memory Usage
- CPU Frequency
- CPU Temperature
- Load Average

The stats were recorded during 3 modes of operation:

- Idle
- Normal Radio Usage
- Spotify Client Usage

CPU Frequency (MHz)			
	Minimum	Maximum	Average
CPU	600	600	600
CPU Temperature (°C)			
	Minimum	Maximum	Average
CPU	39.2	39.7	39.45
Memory Usage (Megabytes)			
	Minimum	Maximum	Average
Used	59.9	61.0	60.45
Free	863.1	862	62.55
CPU Usage (%)			
	Minimum	Maximum	Average
CPU1	0.0	0.7	1.1
CPU2	0.7	1.3	0.95
CPU3	0.0	0.6	2.6
CPU4	1.3	2.0	1.25
Load Average			
	Minimum	Maximum	Average
Load	0.24	0.6	0.42

Table 5.1 – Idle Sysyem Performance Test

Extracted from Conky are system related measurements of the internet radio appliance. The Raspberry Pi 3 Model B+ frequency when the system is idle, does not deviate from 600MHz. On average, the measured temperature is 39.45°C. The memory usage on average is 60MB compared to the allocated 1GB Ram. From the 4 CPU cores, the average CPU usage is 1.48% and there is an average system load of 0.42.

CPU Frequency (MHz)			
	Minimum	Maximum	Average
CPU	600.064	700	650.032
CPU Temperature (°C)			
	Minimum	Maximum	Average
CPU	41.9	42.4	42.15
Memory Usage (Megabytes)			
	Minimum	Maximum	Average
Used	79.8	80.8	80.3
Free	843.2	842.2	842.7
CPU Usage (%)			
	Minimum	Maximum	Average
CPU1	0.7	14.1	7.4
CPU2	2.2	7.6	4.9
CPU3	0.7	5.3	3.0
CPU4	0.9	7.1	4.0
Load Average			
	Minimum	Maximum	Average
Load	0.24	0.9	0.57

Table 5.2 – Sysyem Performance Test with Radio Program

The Raspberry Pi 3 Model B+ frequency when the system is running the radio softwae, has an average frequency of 650.032MHz. On average, the measured temperature is 42.15°C. The memory usage on average is 80.3MB compared to the allocated 1GB Ram. From the 4 CPU cores, the average CPU usage is 4.8% and there is an average system load of 0.57.

CPU Frequency (MHz)			
	Minimum	Maximum	Average
CPU	600	700	650
CPU Temperature (°C)			
	Minimum	Maximum	Average
CPU	40.8	40.9	40.75
Memory Usage (Megabytes)			
	Minimum	Maximum	Average
Used	92.7	95.6	94.15
Free	830.3	827.4	828.85
CPU Usage (%)			
	Minimum	Maximum	Average
CPU1	9.5	16.5	13
CPU2	2.2	7.6	4.9
CPU3	0.7	7.0	3.85
CPU4	0.9	11.3	6.1
Load Average			
	Minimum	Maximum	Average
Used	0.48	0.78	0.63

Table 5.3 – Sysyem Performance Test with Spotify Client

The Raspberry Pi 3 Model B+ frequency when the system is running the radio softwae, has an average frequency of 650MHz. On average, the measured temperature is 40.75°C. The memory usage on average is 94.15MB compared to the allocated 1GB Ram. From the 4 CPU cores, the average CPU usage is 6.96% and there is an average system load of 0.63.

5.2 System Performace Discussion

The idle state of the Raspberry Pi has system performance stats that are expected based on the manufacturer's performance review. While running the radio software, Table 5.2 depicts that the memory usage is increased approximately by 20MB with a 3% increase in CPU usage. Amongst, these results a narrative can be drawn that the radio software itself doesn't require much computing power.

As shown in Table 5.3, the change in state between the radio software and Spotify client software makes evident that it requires slightly more computing power as there is a 14MB increase in memory usage and 2% increase in CPU usage.

Regarding the temperature, the temperature varied between 39.2°C and 42.4°C. The Raspberry Pi Foundation officially recommend that the Raspberry Pi should be below 85°C [34]. Regarding CPU, the maximum frequency can reach up to 1.4GHz. In this usage, the median average was approximately 600MHz with some outliers at 700MHz. The CPU and Memory usage didn't ever seem to be significantly used up.

The stats drawn from the system performance analysis illustrate that the hardware and software implemented for this design doesn't lead to stressing any of the system components or their characteristic.

5.3 Wireless Connectivity Results

Terminal commands were used to test the wireless connectivity performances. The following system stats were extracted and recorded:

- Download Speed
- Upload Speed
- Latency (Ping)
- Bluetooth Range

The stats were recorded during 2 configurations of the appliance:

- With Enclosed Box
- Without Enclosed Box

Download Speed (Mbps)			
	Minimum	Maximum	Average
Speed	12.7	15.87	14.27
Upload Speed (Mbps)			
	Minimum	Maximum	Average
Speed	5.53	7.22	6.38
Latency (ms)			
	Minimum	Maximum	Mode Average
Used	27.9	35.9	31.4
Bluetooth Range (m)			
Distance Between Devices (m)		Distorted Audio Signal?	
1		No	
2		No	
3		No	
4		Yes	

Table 5.4 – Wireless Connectivity Test with Box

The average download and upload speeds were 14.27Mbps and 6.38Mbps respectively. The modal average latency was 31.4ms. The range at which a distortion in audio signals became apparent was at a 4m range.

Download Speed (Mbps)			
	Minimum	Maximum	Average
Speed	14.9	16.09	15.49
Upload Speed (Mbps)			
	Minimum	Maximum	Average
Speed	5.82	6.98	6.4
Latency (ms)			
	Minimum	Maximum	Mode Average
Used	25.2	38.6	29.4
Bluetooth Range (m)			
Distance Between Devices (m)		Distorted Audio Signal?	
1		No	
2		No	
3		No	
4		No	
5		No	
6		Yes	

Table 5.5 – Wireless Connectivity Test without Box

The average download and upload speeds were 15.49Mbps and 6.4Mbps respectively. The modal average latency was 29.4ms. The range at which a distortion in audio signals became apparent was at a 6m range.

5.4 Wireless Connectivity Discussion

The motive behind these tests were to understand whether the box that entrapped the appliance components had an influence in the wireless connectivity performances. As presented in Table 5.4 and 5.5, there was no significant change in WiFi performances but the range at which Bluetooth audio signals could be transmitted was limited by 2 meters. The Pi 3 Model B+ offers Bluetooth 4.2 which should offer a significantly increased range to even the data extracted from Table 5.5. The action take from this analysis was to

implement an external Bluetooth 5.0 USB adapter which when tested provided result that illustrated efficient Bluetooth auio operativity throughout a floor level with walls.

5.5 Power Supply Testing

The Raspberry Pi 3 Model B+ was measured to have an current draw of 660mA during general usage. The following table details the required mAh specification a battery power supply will need for the correlated usage of N number of hours. A 10% margin of error is applied.

Required Battery Capacity (mAh)	
Duration (hrs)	Battery Capacity (mAh)
1	726mAh
2	1,452mAh
3	2,178mAh
4	2,904mAh
5	3,630mAh
6	4,356mAh

Table 5.6 – Power Capacity Testing

5.6 Power Supply Discussion

The original proposal for a power supply was the use of a 9V battery and a voltage regulator for 5VDC. To power the appliance for a minimum of an hour required approximately 700mAh. The commercial 9V batteries on average have a capacity of 500mAh which would offer 42 mins of usage for the appliance. For a more practical approach for usage, ideally a minimum of 6 hours operation would permit an optimal lifespan for the appliance. This would require a battery that capacitates a minimum of 5000mAh.

5.7 Accessibility Analsis and Discussion

The three persona based designs challenged design process to include methods or features in which accessibility would be possible of the detailed personas.

Regarding *Mobility Impairment,* an infra-red sensor and remote were included in the design. The setupcof LIRC and mapping of commands with ir-record made possible for operative commands to be called from python application. This was tested as was shown to be effectively operative.

Regarding *Visual Impairment,* a speech facility was implemented so when turned on, the information displayed on the Character LCD would be spoken out. This was tested as was shown to be effectively operative.

Regarding *Hearing Impairment,* Bluetooth audio operation was implemented, tested and made efficient for audio from the appliance to be transmitted directly to in-ear appliances.

Chapter 6 – Ethics

The topic of internet radios and streaming services brings up for discussion of ethical problems such as audio piracy, revenue paid to artists and podcasters from streaming services and the ethical matters relating to viewpoints and stories shared on radio stations. As the solution of this project is topic of discussion for these ethical problems, this is a subject that needs to be explored to understand what impact this appliance may have for this ongoing discussion.

6.1 Audio Piracy

This designed solution gives consumers the freedom to listen to manually configured radio stations, podcasts and music of their choice. This then brings up the ethical question of where these audio forms are sourced from. Streaming services such as Spotify, Deezer, Youtube Music, etc, have made it possible for consumers to stream their favourite podcasts and music in a legal and ethical manner.

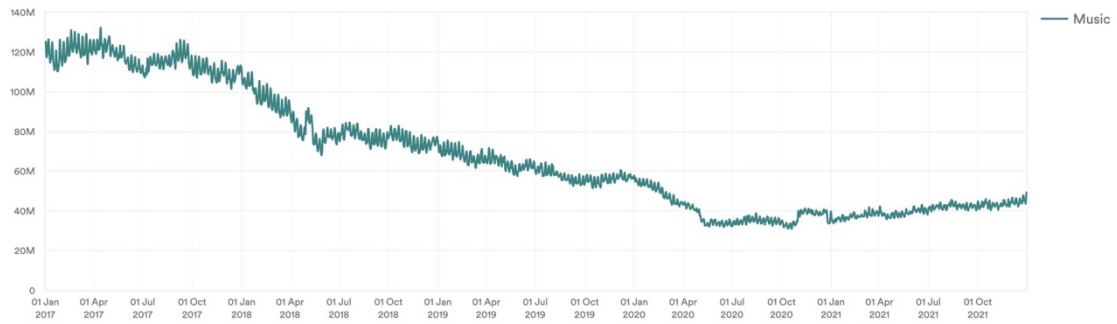

Figure 6.1 – Music Related Piracy 2017 to 2021

As seen in the figure, According to MUSO's data, there has been a 65% decrease in audio related piracy between the years of 2017 till 2021 [35]. These statistics show for a positive effect that the introduction of these streaming services have on the numbers relating to audio piracy as consumers can now buy subscriptions and access all content from their favourite podcasters and access with no exclusion given. Although the numbers have drastically decreased, there is evidence that there may be another trend seen in audio piracy as of

recent. There has been an 18.6% increase in audio piracy when comparing Q4 2020 to Q4 2021 [36].

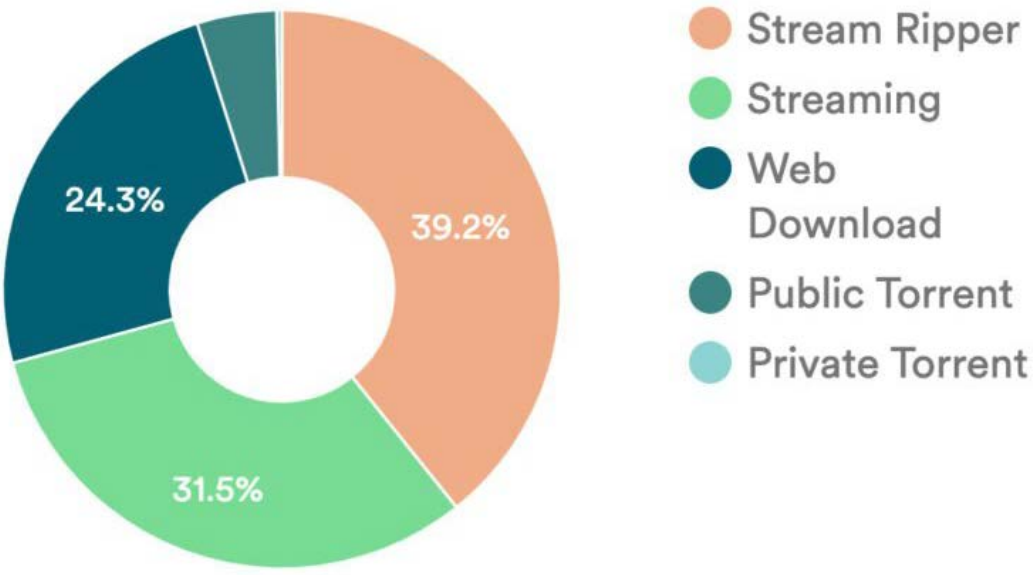

Figure 6.2 – Audio Piracy Delivery Methods

In 2022, with the emergence of audio piracy once again, it must be discovered what methods and platforms are driving the growth of audio piracy once again. The number 1 method for audio piracy now are websites and applications that provide stream ripping. Stream ripping is a method of obtaining a permanent copy of content that is streamed online [37]. This comes from people making copies of audio formats from our afformentioned streaming services and redistributing it. Although, many operators of these stream ripping platforms have been hit with legal action, there are still a wide variety of these services, that are resulting in hit being taken by the streaming platforms and the creators on these platforms.

Regarding the design of the appliance for this project, it has been ensured that the only way for users to stream podcasts and music comes from the ownership of a Spotify account. By this, we can ensure that all deliverance of streaming comes from a service that provide this practice in an ethical manner.

6.2 Ethical Issues Regarding Streaming Platforms

As previously discussed, there is a growing amount of streaming services helping to combat against audio piracy. While this is good, there has been some criticism faced by some of these services. Spotify, which is the largest audio streaming service, with a paying subscriber count of approximately 180 million as of Q4 2021 [38], has been the platform with the most receipt of these criticisms, especially regarding on their method of payment for their creators of the platform.

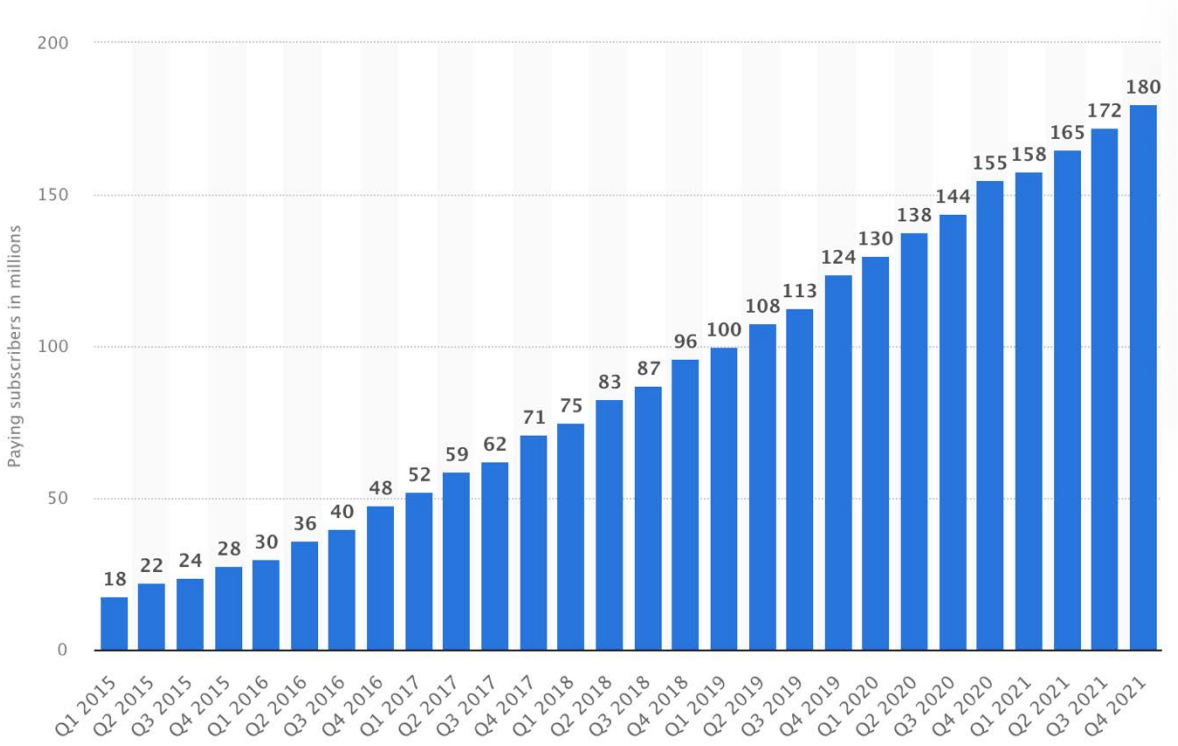

Figure 6.3 – Number of Paying Spotify Premium Subscribers

Studies discern that, more famous artists and podcasters earn more money from their streams fluctuating between $0.0033 to $0.0054 per stream. For every 1,000, that equates to between $3 to $5 [39]. The ethical matter associated with these platforms is that famous people are seen to be entitled to higher payouts for their streams, rather than the stream numbers being directly related to the payout without any outside factors. This is also the case for other platforms such as YouTube and Deezer. A less popular creator with with 1 million streams, may only receive $3000 as payment from Spotify.

The payment received to the creator after the cost for production and releasing these contents to the public may be seen as unjust and not sustainable for the creator. For a consumer and developer of a product that implements streaming services, it should be a concern to know whether the content creators are being financially supported by the services they provide content for. The problem then navigates to the ethical question, if we feel obliged to provide for the ethical wellbeing of the creators that we enjoy listening to.

Results from Spotify's Director of Economics has examines where the revenue made from Spotify subscribers is redistributed to creators. They showcase that the redistribution of revenue is beneficial to artists over a user-centric model and in addition, it exceeds this cost by 4.64% [40].

With the growing numbers of Spotify subscriptions from year to year as shown in Figure 10, it is ethically a key principle that creators are being paid a fair amount for the content they produce that attracts the subscribers.

Chapter 7 - Conclusions and Further Research

In conclusion, the overall aim for this project was achieved. An appliance capable of streaming radio stations, podcasts and other audio media was deployed. Following the completion of the design, implementation and testing of this appliance, it can be concluded that the Raspberry Pi is capable of being the central component of an Internet Radio Appliance. The results from the testing stage presume that there is no significant system demands while streaming from the device is ongoing. The minimal usage of the system capabilities lower the risk for wear in the appliance, provide safety for the user and bring the capability of additional features to be implemented.

Universal Accessibility was a reoccuring topic throughout the design process and implementation of this project. The three persona-designs motivated the design process of this appliance as it became priority to ensure that the persona's demands were met. Aformentioned in the Technical Background, Apple take pride in their design accessibility features by ensuring that most consumer groups can operate their appliances with ease of access. Working to cover three persona designs helps appreciate the design thought companies like Apple conduct to ensure no user is limited by design.

There is still significantly a vast amount of system resources available during operation on the Pi board which brings more streams to further develop this appliance. Testing proved the appliance to be very power demanding. An efficient power system would be viable to draw up to keep the portability feature. Another possible implementations that would be innovative and suitable for this appliance would be the addition of "Virtual Assistant Technology". This would bring forth the possibility for features such as voice control, tailored news reports, control of other appliances, etc. The direction this could take would tailor for ease of access to more fields of users mainly from the feature of voice control. The Python code designed to develop this program could also be redesigned in Swift to make an iOS application that allows for internet radios stations to be added and streamed. The same could be developed for Android.

Chapter 8 - Appendix

```
import os
import subprocess
from time import sleep
from FYP.radio import run
from gpiozero import Button
from display import Display
from subprocess import Popen, PIPE
from time import sleep
from datetime import datetime
import threading
import select

connection = False
display = Display()

# Rotary Encoder Mapping
buttonForMute = Button(16)
buttonForVolumeUp = Button(21)
buttonForVolumeDown = Button(20)
buttonForNext = Button(5)
buttonForPrev = Button(6)
buttonForSwitch = Button(12)
spotifyRunning = False

class Radio:

    def startup(self):
        # Turning on Appliance
        display.out("Alexander Radio","Powering On")

        # Waiting for connection
        while (not connection):
            display.out("Alexander Radio","Connecting to Network....")
            ipaddr = self.parse_ip()
            if len(ipaddr) > 1:
                connection = True

        # Display IP Address
        display.out("Connected to Internet","IP: " + ipaddr)
        sleep(2)
```

```
        # setup MPC when program starts running
        display.out("Loading: ","Radio Stations")
        sleep(2)
        Radio.run()

    def run(self):

        # Play MPC
        self.play()

        # Main program processing loop
        while True:
            try:

                # If Button/Output from Rotary Encoder, Carry out function
                self.displayCurrent() # Display Current Station with
Current Time
                buttonForVolumeUp.when_pressed = self.increaseVolume
                buttonForVolumeDown.when_pressed = self.decreaseVolume
                buttonForMute.when_pressed = self.muteVolume
                buttonForNext.when_pressed = self.nextStation
                buttonForPrev.when_pressed = self.previousStation
                buttonForSwitch.when_pressed = self.switchSource


            except KeyboardInterrupt:
                self.stop()


########## WIFI NETWORK CHECK #################

def find_interface(self):
    find_device = "ip addr show"
    interface_parse = self.run_cmd(find_device)
    for line in interface_parse.splitlines():
        if "state UP" in line:
            dev_name = line.split(':')[1]
    return dev_name

# find an active IP on the first LIVE network device
def parse_ip(self):
```

```
    find_ip = "ip addr show %s" % interface
    ip_parse = self.run_cmd(find_ip)
    for line in ip_parse.splitlines():
        if "inet " in line:
            ip = line.split(' ')[5]
            ip = ip.split('/')[0]
    return ip

# run unix shell command, return as ASCII
def run_cmd(self,cmd):
    p = Popen(cmd, shell=True, stdout=PIPE)
    output = p.communicate()[0]
    return output.decode('ascii')

######### RADIO OPERATIONS ######################

def poweroff(self):
    os.system("sudo shutdown -h now")

def play(self):
    os.system("mpc play")

def pause(self):
    os.system("mpc pause")

def stop(self):
    os.system("mpc stop")

def nextStation(self):
    self.play()
    os.system("mpc next")
    self.speakCurrent()

def previousStation(self):
    self.play()
    os.system("mpc prev")
    self.speakCurrent()

def increaseVolume(self):
    if buttonForVolumeDown.is_pressed:
        os.system("amixer sset Digital 3%+")
        self.displayVolume()
```

```
def decreaseVolume(self):
    if buttonForVolumeUp.is_pressed:
        os.system("amixer sset Digital 3%-")
        self.displayVolume()
        sleep()
        self.displayCurrent()

def muteVolume(self):
    if buttonForMute.is_pressed:
        os.system("amixer set Master toggle")
        self.displayVolume()
        sleep()
        self.displayCurrent()

def displayCurrent(self):
    if spotifyRunning:
        dateTime = datetime.now().strftime('%b %d  %H:%M:%S\n')
        track = self.getSpotifyTrack()
        display.out(dateTime,track)
    else:
        dateTime = datetime.now().strftime('%b %d  %H:%M:%S\n')
        currentStation = self.run_cmd("mpc current")
        display.out(dateTime,currentStation)

# Speech Facility using eSpeak
def speakCurrent(self):
    if spotifyRunning:
        track = self.getSpotifyTrack()
        os.system("espeak -ven+f2 -k5 -s130 -a20 " + track +" --stdout |
aplay")
    else:
        currentStation = self.run_cmd("mpc current")
        os.system("espeak -ven+f2 -k5 -s130 -a20 " + currentStation +" --
stdout | aplay")


def displayVolume(self):
    volume = self.run_cmd("mpc volume")
    display.out("Alexander Radio",volume)

# Switch between spotify and radio
```

```
def switchSource(self):
    global spotifyRunning
    if spotifyRunning:
        os.system("killall journalctl")
        os.system("sudo systemctl stop raspotify.service")
        os.system("mpc play")
        spotifyRunning = True
    else:
        os.system("mpc stop")
        os.system("sudo systemctl start raspotify.service")
        self.startJournalWatch()
        spotifyRunning = False


########## SPOTIFY MONITORING ##############

def startJournalWatch(self):
        t = threading.Thread(target=self.followJournal)
        t.daemon = True
        t.start()

def getSpotifyTrack(self):
    args = ['journalctl', '--lines', '0', '--follow',
'_SYSTEMD_UNIT=raspotify.service']
    f = subprocess.Popen(args, stdout=subprocess.PIPE)
    p = select.poll()
    p.register(f.stdout)

    while True:
        line = f.stdout.readline()
        line = (line.strip())
        line = line.lstrip().decode('utf-8')

        if "INFO:librespot" in line:
            try:
                elements = line.split('::')
                spotifyTrack = elements[1]
                return spotifyTrack
            except ValueError:
                pass

if __name__ == '__main__':
```

```
  Radio().startup()
```

Figure 8.1: Python Application (main.py)

```
from subprocess import Popen, PIPE
from time import sleep
from datetime import datetime
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd
import threading

class Display:
    # Rows and Cols configured for LCD display
    lcd_columns = 16
    lcd_rows = 2

    # compatible with all versions of RPI as of Jan. 2019
    lcd_rs = digitalio.DigitalInOut(board.D22)
    lcd_en = digitalio.DigitalInOut(board.D17)
    lcd_d4 = digitalio.DigitalInOut(board.D25)
    lcd_d5 = digitalio.DigitalInOut(board.D24)
    lcd_d6 = digitalio.DigitalInOut(board.D23)
    lcd_d7 = digitalio.DigitalInOut(board.D18)


    # Initialise the lcd class
    lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5,
lcd_d6, lcd_d7, lcd_columns, lcd_rows)

    # Turn on LCD Backlight
    lcd.backlight = True

    def __init__(self):
        threading.Thread.__init__(self)

    # Function to Display Text on both lines
    def out(self,lcd_line_1,lcd_line_2):
        # wipe LCD screen before text displayed
        self.lcd.clear()
        while True:
            self.lcd.message = lcd_line_1 + lcd_line_2
```

Figure 8.2: Python Application (display.py)

Figure 8.3: Completed Appliance

References

[1] Daniel Barnes (2019). The Complete Guide to Internet Radio [online]. Available from: https://ligo.co.uk/blog/internet-radio-complete-guide/

[2] Jack Schofield (2014). What are the options for radio in a digital age? [online]. Available from:https://www.theguardian.com/technology/askjack/2014/apr/25/what-are-the-options-for-radio-in-a-digital-age

[3] S. Quackenbush (2005). MPEG Advanced Audio Coding [online]. Available from: https://mpeg.chiariglione.org/standards/mpeg-2/advanced-audio-coding

[4] National Disability Authority (2020). What is Universal Design? [online]. Available from: https://universaldesign.ie/what-is-universal-design/

[5] Xin Zhang, Jianwu Zhang and Mei Yang, "The application study of universal design principle in the old people product," 2008 9th International Conference on Computer-Aided Industrial Design and Conceptual Design, 2008, pp. 120-123

[6] Alicia Crowther (2020). Creating Accessibility Personas [online]. Available from: https://uxdesign.cc/creating-accessibility-personas-e7749d4096b4

[7] Mark Boyes-Smith (2020). Bringing inclusive design to life through Personas [online]. Available from: https://uxdesign.cc/bringing-inclusive-design-to-life-through-personas-83ba26a41109

[8] National Disability Authority (2020). 7 Principles of Universal Design [online]. Available from: https://universaldesign.ie/what-is-universal-design/the-7-principles/

[9] Sheryl Burgstahler, Ph.D. (2012). Working Together: People with Disabilities and Computer Technology [online]. Available from: https://www.washington.edu/doit/sites/default/files/atoms/files/wtcomp.pdf

[10] Apple (2022). Vision For Every Point of View [online]. Available from: https://www.apple.com/accessibility/vision/

[11] Apple (2022). Mobility. A tap. A touch. A ton of possibilities. [online]. Available from: https://www.apple.com/accessibility/mobility/

[12] Apple (2022). Hearing. Catch every word, sign, or signal. [online]. Available from: https://www.apple.com/accessibility/hearing/

[13] Hendrikse, Reijer & Adriaans, Ilke & Klinge, Tobias & Fernandez, Rodrigo. (2021). The Big Techification of Everything. Science as Culture. 31. 59-71. 10.1080/09505431.2021.1984423.

[14] BeagleBoard (2018). Beaglebone black [online]. Available from: https://beagleboard.org/black

[15] ASUS (2018). Asus tinker board [online]. https://tinker-board.asus.com/product/tinker-board-s.html

[16] Raspberry Pi Foundation (2022). Raspberry Pi [online]. Available from: https://www.raspberrypi.org/

[17] Raspberry Pi Projects (2022). Physical Computing with Python [online]. Available from: https://projects.raspberrypi.org/en/projects/physical-computing/1

[18] Jeff Tranter (2019). An Introduction to GPIO Programming [online]. Available from: https://www.ics.com/blog/introduction-gpio-programming

[19] Tia D (2022). LCD-principle and applications [online]. Available from: https://www.academia.edu/16470045/LCD_principle_and_applications

[20] Stanbury, P., Whitaker, A. and Hall, S. (2017) "Instrumentation and control", *Principles of Fermentation Technology*, pp. 487-536.

[21] Tan, L. and Jiang, J. (2019) "Image Processing Basics", *Digital Signal Processing*, pp. 649-726. doi: 10.1016/b978-0-12-815071-9.00013-0.

[22] *How do I power my Raspberry Pi?* (2022). Available at: https://thepihut.com/blogs/raspberry-pi-tutorials/how-do-i-power-my-raspberry-pi#:~:text=The%20first%2C%20recommended%20and%20easiest,is%205.1V%20%40%202.5A

[23] Dyer, B. (2021) *What are Daemons in Linux? Why are They Used?*, *It's FOSS*. Available at: https://itsfoss.com/linux-daemons/#:~:text=What%20is%20a%20Daemon%20in,the%20operating%20system%20runs%20properly.

[24] *Music Player Daemon* (2022). Available at: https://www.musicpd.org

[25] *One app fits all - O!MPD* (2015). Available at: http://ompd.pl/one-app-fits-all

[26] *LIRC - Linux Infrared Remote Control* (2022). Available at: https://www.lirc.org

[27] Sarah Perez (2021). Spotify says US podcast listeners now use its service more than Apple Podcasts. Available from: https://techcrunch.com/2021/10/27/spotify-says-u-s-podcast-listeners-now-use-its-service-more-than-apple-podcasts/?guccounter=1#:~:text=According%20to%20eMarketer%27s%20recent%20data,million%20by%20a%20thin%20margin

[28] G., A. (2022) *Mono vs Stereo: Which Should You Go For? | Headphonesty*, *Headphonesty*. Available at: https://www.headphonesty.com/2022/01/what-is-the-difference-between-mono-and-stereo

[29] *Rotary Encoder* (2013). Available at: https://www.azosensors.com/article.aspx?ArticleID=312

[30] *Introduction to PuTTY* (2022). Available at: https://the.earth.li/~sgtatham/putty/0.76/htmldoc/Chapter1.html#you-what

[31] *OS Module in Python with Examples - GeeksforGeeks* (2016). Available at: https://www.geeksforgeeks.org/os-module-python-examples

[32] *gpiozero — GPIO Zero 1.6.2 Documentation* (2022). Available at: https://gpiozero.readthedocs.io/en/stable/

[33] *Drive a 16x2 LCD with the Raspberry Pi* (2022). Available at: https://learn.adafruit.com/drive-a-16x2-lcd-directly-with-a-raspberry-pi/python-code

[34] *Raspberry Pi Temperature Monitor* (2022). Available at: https://linuxhint.com/raspberry_pi_temperature_monitor/#:~:text=Officially%2C%20the%20Raspberry%20Pi%20Foundation,throttling%20at%2082%20degrees%20Celsius

[35] Stassen, M. (2022) *Music piracy has plummeted in the past 5 years. But in 2021, it slowly started growing again. - Music Business Worldwide*, *Music Business Worldwide*. Available at: https://www.musicbusinessworldwide.com/music-piracy-plummeted-in-the-past-5-years-but-in-2021-it-slowly-started-growing-again/#:~:text=Stream%2Dripping%20sites%2C%20which%20allow,music%20business%20in%20recent%20years

[36] *Spotify users - subscribers in 2021 | Statista* (2022). Available at: https://www.statista.com/statistics/244995/number-of-paying-spotify-subscribers/#:~:text=How%20many%20paid%20subscribers%20does,than%20doubled%20since%20early%202017

[37] (2022) *Orpheusaudioacademy.com*. Available at: https://www.orpheusaudioacademy.com/spotify-pay/#:~:text=You%20can%20expect%20to%20make,between%20%240.0033%20%2D%20%240.0054%20per%20stream