**ECE 458**

**Spring 2020**

**Final Report**

**ECE – 6 TV Auto Commercial Mute System (MuteBot)**

**Report Submitted: April 29, 2020**

We, the undersigned, certify that we contributed to the generation of this report and attest to the validity of the data herein:

**Team Members:**

Steven Ferreira \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Thomas Morrissey \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Kevin Prairie \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zachary Taylor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Customer:**

Dr. Paul J. Fortier

**Advisor:**

Dr. David P. Rancour

**Table of Contents**

[1 Project Overview 5](#_Toc39058212)

[2 Customer Requirements 5](#_Toc39058213)

[3 Engineering Requirements 6](#_Toc39058214)

[3.1 Requirements 6](#_Toc39058215)

[3.2 Constraints 9](#_Toc39058216)

[3.3 Standards 10](#_Toc39058217)

[3.4 Ethical Issues 11](#_Toc39058218)

[4 Functional Overview 11](#_Toc39058219)

[5 Alternatives Evaluated 12](#_Toc39058220)

[5.1 Processors/Development Board 12](#_Toc39058221)

[5.2 Interfaces 14](#_Toc39058222)

[6 Technical Design Description of Selected System 15](#_Toc39058223)

[6.1 Detailed System Diagram of Overall System 15](#_Toc39058224)

[6.2 Commercial System Detection 16](#_Toc39058225)

[6.3 TV Remote 17](#_Toc39058226)

[6.4 Program Flow 18](#_Toc39058227)

[7 Test Plan and Results 21](#_Toc39058228)

[7.1 VCRM 21](#_Toc39058229)

[7.2 Test Cases and Test Results 21](#_Toc39058230)

[7.3 Test Summary 29](#_Toc39058231)

[8 Risk Discussion 30](#_Toc39058232)

[9 Plan, Schedule, and Costs 31](#_Toc39058233)

[9.1 Plan and Schedule – Final 31](#_Toc39058234)

[9.2 Actual Hours vs. Planned Hours 31](#_Toc39058235)

[9.3 Cost Summary 32](#_Toc39058236)

[10 Summary 33](#_Toc39058237)

[11 Lessons Learned 34](#_Toc39058238)

[11.1 Lessons Learned 34](#_Toc39058239)

[11.2 Impacts 34](#_Toc39058240)

[11.3 Innovation 36](#_Toc39058241)

[11.4 Steps to Success 36](#_Toc39058242)

[12 Documentation 37](#_Toc39058243)

**List of Tables**

[Table 1 - Program Customer Requirements 5](#_Toc39058244)

[Table 2 - System Customer Requirements 6](#_Toc39058245)

[Table 3 - Engineering Requirements 6](#_Toc39058246)

[Table 4 - Processor/Development Board Comparison 12](#_Toc39058247)

[Table 5 - Processor/Development Board Strengths vs. Weaknesses 12](#_Toc39058248)

[Table 6 - Interfaces Strengths vs Weaknesses 14](#_Toc39058249)

[Table 7: Verification Cross-Reference Matrix 21](#_Toc39058250)

[Table 8 - MuteBot Setup Test 22](#_Toc39058251)

[Table 9 - System Remote Test 23](#_Toc39058252)

[Table 10 - Commercial Detection Test 24](#_Toc39058253)

[Table 11 - TV Remote Test 26](#_Toc39058254)

[Table 12 – Enclosure Test 27](#_Toc39058255)

[Table 13 - Cost Analysis Test 28](#_Toc39058256)

[Table 14 - Program File Integrity Test 28](#_Toc39058257)

[Table 15 - Program File Type Test 29](#_Toc39058258)

[Table 16 - Test Summary 30](#_Toc39058259)

[Table 17: Potential Problems & Risk Analysis 30](#_Toc39058260)

[Table 18 – Project Cost Table 32](#_Toc39058261)

[Table 19 - Product Cost Table 33](#_Toc39058262)

**List of Figures**

[Figure 1 - System Diagram 15](#_Toc39058265)

[Figure 2: Program Flow of commercialDetector 20](#_Toc39058266)

# Project Overview

The MuteBot system was created to solve a common problem while watching television. When a television program ends and a commercial begins, the user is immediately bombarded with big businesses trying to sell them a product. This system would solve this issue of forced consumerism, which disturbs the user’s viewing experience. The objective of this project is to have a system that can successfully detect volume spikes, silent frames, and volume averages. With these values, the system should be able to detect a commercial break. The MuteBot system would use a 3.5mm audio jack to detect audio abnormalities when commercials occur. Using the input from TV audio, the MuteBot’s detection program would detect a commercial and mute or lower the TV volume using the system’s IR transmitter. The impact of a successful system would put an end to commercials by muting the TV without the user’s need of using a TV remote to manually mute it.

Due to unforeseen circumstances, the MuteBot system could not be retrieved from the lab quickly and as a result, a Python program was created. This Python program, created in PyCharm, would detect a commercial during a pre-recorded TV program and delete the commercials entirely. This results in a continuous viewing experience without interruption by big business. The impact of a successful program would eliminate the need to watch commercials. The user would be able to watch the entire pre-recorded TV program without the need to mute/lower the volume of the commercial or manually skip the commercials.

# Customer Requirements

Due to the unforeseen circumstances of COVID-19, the customer requirements have changed drastically. Instead of having a set of one customer requirements, the group and the customer agreed to have two separate requirement tables. The program customer requirements, seen in Table 1, highlight the criteria the program written must achieve to be deemed a success. The system customer requirements, seen in Table 2, highlights the requirements that the overall system should have met if this project continued under normal operation. All program requirements are prefaced with a P and system requirements are preface with a S.

Table - Program Customer Requirements

|  |  |
| --- | --- |
| Program Customer Requirement Number | Requirement Description |
| P.1 | Program must react preemptively. |
| P.2 | Program must remove all commercials from the .mp4 file |
| P.3 | Program must not remove any part of the main media. |
| P.4 | Program must output a single .mp4 file. |
| P.5 | Program must accept .mp4 files for input |

From the test results, the system customer requirements have been altered to omit some of the requirements due to the programming side of the project now handling these requirements. The other requirements were left to show how the system’s schematics and current work conducted would have still satisfied the customer’s needs.

Table - System Customer Requirements

|  |  |
| --- | --- |
| System Customer Requirement Number | Requirement Description |
| S.1 | System must not obstruct the TV Screen. It must be able to be kept out of sight. |
| S.2 | System must be simple to initially set up. |
| S.3 | System must have a user-friendly interface/remote. |
| S.4 | System must contain options to mute completely or lower volume.  ~~System must allow user to choose to mute completely or lower volume by a set amount when a commercial is detected.~~  *This was changed to combine S.8 into S.4* |
| S.5 | System cost must be competitive with competition. |
| S.6 | System must refrain from any alterations to the TV or its remote control. |
| ~~S.7~~ | ~~System must react to a commercial break within 1 second for improvement of accuracy.~~  *This was omitted due to the program side of the project now handling this requirement* |
| ~~S.8~~ | ~~System must mute or lower volume upon break and unmute or return to original volume upon return.~~  *This was omitted due to the program side of the project now handling this requirement* |

# Engineering Requirements

## Requirements

The engineering requirements have been adjusted to accommodate the program side of the project.

Table - Engineering Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prog.  Rqmt. # | Customer Requirement | Engineering Requirement | Justification/Comments | Test Method (IADT) |
| P.1 | Program must react preemptively. | The user must not be able to see any commercials and must be able to watch the main media without interruption. | The program should ensure the user gets a clean and pleasurable viewing experience. | Inspection/Test:  Run the file through the program, observe if the outputting file has any commercial breaks |
| P.2 | Program must remove all commercials from the .mp4 file | The .mp4 file must delete all video from the point the main media cuts away to the point right before the main media returns. | The program should ensure the user does not miss any of the main media while also ensuring the user does not see any advertisement. | Inspection/Test:  Run the file through the program, observe if the outputting file has any commercial breaks |
| P.3 | Program must not remove any part of the main media. | The .mp4 file must keep all video before the point the main media cuts away to the point right after the main media returns. | The program should ensure the user does not miss any of the main media while also ensuring the user does not see any advertisement. | Inspection/Test:  Run the file through the program, observe if the outputting file has cut any main media from the input file. |
| P.4 | Program must output a single .mp4 file. | The output must be one single .mp4 file and not multiple files. | The user will not want to interrupt their viewing and should just have to click a single file to watch all of the selected media. | Inspection:  Observe if there is more than one file on program completion. |
| P.5 | Program must accept .mp4 files for input. | The input to the program MUST be in a .mp4 format. | Working with one format makes the program prone to less errors. | Test:  Ensure the format outputted from the program is in .mp4. |
| P.5.1 |  | Other formats must be rejected and inform the user via a console window that they are invalid and a .mp4 must be used. | Informing the user of an .mp4 will ensure they use the correct format. | Inspection/Test:  Test program by putting in wrong file type. Ensure error message pops up to user. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| System  Rqmt. # | Customer Requirement | Engineering Requirement | Justification/Comments | Test Method (IADT) |
| S.1 | System must not obstruct the TV Screen. It must be able to be kept out of sight. | System can have a cable box at most, rest of system must remain behind the TV. | The system must not be obtrusive to ensure the user has a clear and not obstructed viewing experience. | Inspection:  User-Review/Observation  This will be measured via a questionnaire. Any category in the questionnaire labeled 3 or lower will be deemed not acceptable and the next step for improvement. |
| S.2 | System must be simple to initially set up. | The engineers are only to assume the users know how to: plug in an HDMI cord, | If the system is similar to design to a TV, then the user will find it easier to work with something of similar design. | Inspection/Analysis:  User-Review/Test Subject  This will be measured via a questionnaire. Any category in the questionnaire labeled 3 or lower will be deemed not acceptable and the next step for improvement. |
| S.2.1 |  | Power Cord, | Same as above. | Same as above |
| S.2.2 |  | Interface with a TV remote | Same as above. | Same as above |
| S.3 | System must have a user-friendly interface/remote. | The user interface must not consist of anything that would not already be on a TV remote or cable box. | A simple user interface allows ease-of-use and broadens the potential consumers. | Inspection/Analysis:  User-Review/Test Subject  This will be measured via a questionnaire. Any category in the questionnaire labeled 3 or lower will be deemed not acceptable and the next step for improvement. |
| S.4 | System must contain options to mute completely or lower volume. | The system must have a user interface that enables the user to choose between a volume of 0% (mute), or a volume 10 units lower than the original volume | Giving options to the user allows for a more customizable experience. | Test & Demo: |
| S.5 | System cost must be competitive with competition. | The system must remain anywhere from $40 - $100 retail cost. | A low retail costs attracts more customers and makes the product more able for mass production. | Analysis:  Components and materials cost will be analyzed using Excel. The final product will be compared to competition (MuteMagic, MuTR) |
| S.5.1 |  | Thus, the manufacturer cost is estimated to be between $24-$67. | A lower manufacturer costs aims for a larger profit. | Analysis:  Same as above. |
| S.6 | System must refrain from any alterations to the TV or its remote control. | The TV and remote must remain the same as originally purchased. | Altering the TV or remote would require too much of the user and is not fit for mass production. | Inspection:  User-Review  This will be measured via a questionnaire. Any category in the questionnaire labeled 3 or lower will be deemed not acceptable and the next step for improvement. |

## Constraints

The list below shows the seven constraints for the MuteBot system as well as the constraints added due to pandemic.

1. Form Factor (Same size or smaller than an Apple TV or Roku Ultra)

* Apple TV: Height-1.4 in, Width-3.9 in, Depth-3.9 in
* Roku Ultra: Height-0.8 in, Width-4.9 in, Depth-4.9 in

1. TV cannot be altered or changed in any way (Removing or modifying parts or remote)
2. Location- should not be visible (mounted to the back of the TV)
3. Outputs on the TV (Digital Optical Audio cable, 3.5mm Jack)
4. Budget-must be in same price range as the competitors (MuteMagic $39.95, MUTR $30 with a subscription of $50 per year)
5. Television provided (Westinghouse HDTV)
6. Constrained to the capabilities of python.
7. ~~The ongoing outbreak of COVID-19 has disrupted the timing, resource availability, and meeting availability for the team. Section 11.2 provides a greater detailed impact on the project’s entirety.~~

COVID:

1. Remote Communication – Zoom or Discord
2. One set of hardware
3. No live demonstration only remote/video
4. Harder to get lab assistance due to not having access to all faculty like before
5. Lack of lab equipment such as oscilloscopes, power supplies, and other tools.
6. Harder to conduct group meetings
7. Harder to stay informed/updated
8. Time Fame – Lost a large amount of progress

## Standards

The list below shows the current standards for the system.

1. CALM Act: Commercial Advertisement Loudness Mitigation Act:
   1. <https://www.provideocoalition.com/the-calm-act-commercial-advertisement-loudness-mitigation/>
   2. The FCC set and monitor the loudness of commercials. The ITU, International Telecommunication Union, created standard audio measurements for content that is being broadcasted
2. ITU-R BS.1170:
   1. <https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1770-4-201510-I!!PDF-E.pdf>
   2. Ways to measure commercial audio loudness and true-peak audio level
3. IEEE Code of Ethics
   1. <https://www.ieee.org/about/corporate/governance/p7-8.html>
   2. The responsibilities in which all engineers are expected to follow that are expressed in a code of ethics.
4. Betamax Case: Sony Corp. of America v. Universal City Studios, Inc.
   1. <https://www.oyez.org/cases/1982/81-1687>
   2. Ruled recording TV legal
5. Copyright Laws and Television:
   1. <https://yourbusiness.azcentral.com/copyright-laws-television-16286.html>
   2. TV cable programs have copyrights to a program that can be violated (file sharing and sales, dependent on each program)
6. HDMI Specification Version 1.4a
   1. https://www.hdmi.org/manufacturer/hdmi\_1\_4/index.aspx
   2. HDMI standards and specifications that define the required waveforms and video format.
7. American National Standard ANSI/SCTE 07 2006, American National Standard ANSI/SCTE 124 2006
   1. <https://www.scte.org/documents/pdf/Standards/ANSISCTE072006.pdf>
   2. Digital Transmission Standard for Cable Television
   3. <https://www.scte.org/documents/pdf/Standards/ANSISCTE1242006.pdf>
   4. Specifications and standards for the F type connector used for cable television
8. ITU-T L.1002 (10/2016)
   1. https://www.itu.int/itu-t/recommendations/rec.aspx?rec=12131
   2. Standards for external universal power adapters
9. Standard IEC958
   1. Digital audio interface, standard for digital optical audio cables
   2. <http://www.epanorama.net/documents/audio/spdif.html>
10. TRRRS Standards including P.382
    1. Standards for the 3.5mm connector
    2. P.382 TRRRS connectors for new integration including multiple sources and noise canceling
    3. https://www.itu.int/itu-t/workprog/wp\_item.aspx?isn=9990
11. Infrared Data Association (IrDA)
    1. Standards and specifications for IR transmitter and receiver communication
    2. <https://www.novell.com/documentation/suse91/suselinux-adminguide/html/ch08s03.html>
12. IEEE 802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks
    1. Standard for RF modules including the 3 pin RF module
    2. Short range devices have unlicensed ISM/SRD bands like RF remotes
    3. <https://standards.ieee.org/standard/802_15_4-2015.html>

## Ethical Issues

The ethical issues involved with a commercial detection system resides with the ethics of swindling big business. As a result of the system, any advertisements, which these businesses pay for, will not longer be heard or seen by their target audience. Thus, as the engineers, the group is arguably stealing from these businesses. Likewise, if this product became popular it would decrease the amount of business advertisement companies and television channels receive as big business would look to other means of outreach. This would result in a loss of revenue for both types of businesses. However, as the consumer, the engineers feel as if they should seek out what to purchase rather than being suggested what to purchase by big business.

# Functional Overview

The original MuteBot system was in the process of being designed to mute or lower the volume of a commercial within a TV broadcast. An IR signal were to be sent from the MuteBot to the TV. The IR signal sent from the Raspberry Pi, using the IR blaster, would either send a mute command to the Westinghouse TV or a volume up or down command. The specific signal would be sent based on the user setting chosen. The system would have an input from the 3.5 mm jack that is connected to the TV. This addition allows for the volume output of the TV to be sent to the MuteBot. The volume from the TV was then to be inputted into the Raspberry Pi 3b and read by the commercial detection program. However, due to unforeseen circumstances the MuteBot was not completed to expectations. Refer to section 12, week 11/17-11/24, for the functional diagram of the MuteBot.

Instead a program was written for a pre-recorded broadcast of “The Price is Right.” The program that was created was made to take out the commercials within the broadcast. The program detects different scenes and then the commercials from that file are then to be removed from the output. This is expanded upon further in section 6.4.

# Alternatives Evaluated

For the MuteBot system, there are multiple major components that have different solution paths. Those alternative solutions have been evaluated and compared to one another in order to narrow down the best choice to meet the project’s customer and engineering requirements. The two major aspects of the project include the processor/development board and TV interface. Both aspects have different risks associated with them based upon the impact they have on the system as a whole.

## Processors/Development Board

Table 4 provides the overview of alternatives for choosing the processor/development board used to store audio/video data and send signals to mute/decrease the volume of the TV during a commercial. The systems below include the ATmega328pb Xplained mini, BeagleBone Black, Raspberry Pi 4 Model B(2GB), and a custom PCB with the ATmega328pb chip.

**Table 4 - Processor/Development Board Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Processor/Development Board | ATmega328pb Xplained mini | BeagleBone Black | Raspberry Pi 4 Model B (2 GB) | Custom PCB design with ATmega328pb chip |
| Processor Speed | 20 MHZ | 1 GHz | 1.5 GHz | 20MHZ |
| Power | 1.8-5.5V DC  or USB | 5V DC or USB | USB C 5V DC | 5V DC or USB |
| DDR Memory | 32 KB | 512 MB | 2GB LPDDR4 | 32 KB |
| Cost | $9.26 | $56.96 | $48.01 | $51.26 |
| Connection | 32 interface pins | 92 connection points | 40 dedicated interface pins | 32 interface pins |

Table 5 describes the strengths and weaknesses of each system. This helped determine which system was best suited for the team’s project and best met the customer and engineering requirements.

**Table 5 - Processor/Development Board Strengths vs. Weaknesses**

|  |  |  |
| --- | --- | --- |
| Processor/Development Board | Strengths | Weaknesses |
| ATmega328pb Xplained mini | * Cost * Familiar (ECE 263) * Programming languages (Atmel Studio for C, C++, Assembly) * Form Factor | * Speed in order to receive high frequency audio and video waveforms * Memory in order to store data * Connections- lowest amount at 32 |
| BeagleBone Black | * Expandable hardware with 92 pins * On board storage- 4GB * Expandable storage – microSD card * Speed- 1GHz * Programming languages (Python, C, C++) | * Form Factor * Cost * Speed- will depend on how fast it needs to be to capture coaxial/HDMI signals |
| Raspberry Pi 4 Model B (2 GB) | * Speed- 1.5 GHz * Memory * Storage- expandable storage with microSD card * Interfaces- HDMI, USB * Programming languages (Python, Scratch, C, C++) | * Cost * Runs on Linux but Windows can be installed * Form Factor- largest development board |
| Custom PCB design with ATmega 328pb chip | * Prioritize ATmega chip use and speed to focus on video and audio detection and analysis * Components needed for the PCB can be limited to only needed I/O (IR/RF receiver and transmitter, coaxial/HDMI input, audio output) | * Cost- includes $50 PCB, Atmega chip, 5V power adapter, labor to build * Time- more time to create PCB with risk of failure * Speed- will depend on how fast it needs to be to capture coaxial/HDMI signals |

Based upon the engineering requirements, in order to receive high frequency video and audio signals, a high-speed chip was needed to perform the task. With speed being the number one concern, the ATmega328pb Xplained mini will be deleted from the possible boards. As referenced in Table 5, the speed of the ATmega328pb was 20 MHz. This speed would not be enough to receive video and audio signals, process them, and send the proper mute/decrease volume command to the television before a commercial is played. Using cost as a customer requirement, a low-cost auto-mute system would be ideal. This would eliminate the custom PCB design due to both the lower processor frequency and the amount of money needed to manufacture the PCB. This price would also have to include the power adapter, processor, the PCB itself, and the labor required to design the PCB. This price would also increase if revisions were needed to the design. The team decided to use a Raspberry Pi 3 due to a cheaper price with similar speeds to the Raspberry Pi 4.

## Interfaces

Table 6 provides the overview of alternatives for choosing the interface and the strengths and weaknesses of each available option for the system. The interfaces below include the HDMI cable, Coax, 3.5mm audio jack, and the digital optical audio cable.

**Table 6 - Interfaces Strengths vs Weaknesses**

|  |  |  |
| --- | --- | --- |
| Interfaces | Strengths | Weaknesses |
| HDMI | * Produces both audio and video on one cable * Universally available (Standard for TV) | * Highly Encrypted via HDCP * Fast Transfer Speeds (10 to 18 Gbps for 4k) * Would require large memory capacity |
| Coax | * Universal (Works for antenna and cable box) * Produces both audio and video * Slow Transfer Speed (10Mbps) | * Cost (Would require a coax splitter) * Encryption via the cable provider |
| 3.5mm Audio Jack | * No Encryption * No Transfer Speeds (Works on Audio Frequencies) | * Only Audio Signals * Not Universal (Many TVs do not have one) |
| Digital Optical Audio Cable (TOSLINK) |  | * Encrypted * Not Universal * Only Audio Signals * Expensive to implement * 125 Mbps transfer rate |

Table 6 describes the strengths and weaknesses of each available interface for the system. The choice for the final system is the 3.5mm audio jack. TOSLINK has no strengths and does not offer anything that the other interfaces do not have. HDMI is a close second to coaxial, however the transfer speeds of HDMI are far faster than coaxial, which would raise the cost of a processor significantly. Coaxial was also too fast to process due to the many channels coaxial provides. HDMI uses high-bandwidth digital content protection (HDCP). This makes using this interface much more difficult as the system would have to pass an authenticity check. With 3.5mm audio jacks the biggest advantage is their lack of encryption and lack of transfer rates. It is for these reasons that the 3.5mm audio jack is chosen for the interface of the MuteBot.

In conclusion, we decided to choose the Raspberry Pi 3 and 3.5mm audio jack for the MuteBot system. These two components gave the best path to successfully detecting and alerting the TV audio. Due to the span of over a month of not being able to retrieve the MuteBot and equipment, the team was addressed with a new alternative. Alternative one was to create a program which would detect and eliminate commercials during a pre-recorded TV program. The second alternative was to spend the couple weeks after getting the equipment and try to complete the project from home. Due to the high risk, the team decided to choose the Python program. This would allow enough time to get a working product rather than an uncompleted MuteBot. This choice would show the customer that with more time, we could implement it into the MuteBot system to mute commercials during a live broadcast rather than a pre-recorded program.

# Technical Design Description of Selected System

## Detailed System Diagram of Overall System

The TV commercial auto mute system shown in figure 1 was broken down into each subsystem. These subsystems would complete each of the customer requirements for the project. The first subsystem provides the signal processing and volume control and broken down into each of the inputs and hardware. The second subsystem was the mounting system for the MuteBot which covered another customer requirement. The third subsystem provided the remote control and breaks it down into each category to cover the remaining customer requirements. Due to the unforeseen circumstances, customer requirements have changed but in general, the system design below still follows the relative operational flow of MuteBot.

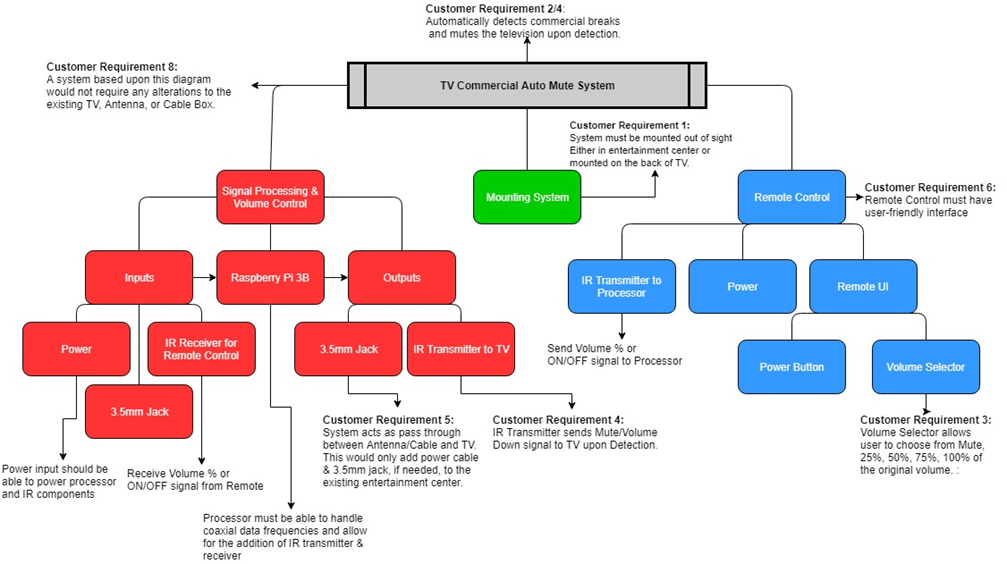


Figure 1 - System Diagram

## Commercial System Detection

The commercial detection utilized audio signals from a 3.5mm audio jack from the TV. With this audio, a python program, built in the PyCharm IDE, would recognize multiple triggers and trigger the IR Shield to either mute or lower the volume on the Westinghouse Television. The first and main trigger that the commercial detection used was silent frames. Silent frames were considered any audio that was at or below 3 on the systems sound scale. The system sound scale made was a result of the stream.read() command in python. Stream.Read() allows for the reading of real-time audio from a source.

“data = stream.read(CHUNK, exception\_on\_overflow=False)”

However, the data that is read comes in as hex values that mean nothing in terms of manipulation and triggers. Thus, the hex values needed to be converted to Float32.

“decoded = np.fromstring(data, 'Float32')”

The problem now is that the float numbers are now exponential. This means if the volume was high, the system would read numbers of near 120,000, but if it was half that volume it would read numbers around 150. Thus, this did not give an accurate representation of what the volume was actual at. To solve this issue, these numbers had to be converted from an exponential increase to a linear increase. This would create a readable scale. Thus, to accomplish this, the group used a log function. The problem now lied that these numbers could also be negative, representing the wave form of positive and negative decibels. To solve this, the group squared the numbers received and scaled it by 10,000 to give non decimal numbers. The number was then rounded to the nearest integer.

“decodedsquared = decoded \* decoded \* 10000”

“logrules = (np.log2((decodedsquared \* 100 + 1)) / np.log2(4)) \* 10”

“volume = np.round(np.mean(logrules))”

After these commands volume could be read on a 0-100 scale. 100 never gets reached as that would require an abnormally loud audio file. 0 gets reached when totally silent. The other numbers varied based upon the audio at the time of reading. With these numbers the system could detect silent frame transitions accurately as well as get moving averages for the volume. Silent frames were detected as follows:

“silent.append(volume)”

This command made an array of the volume over a time frame.

“if i % 2 == 0:”

“if np.max(silent) < 20 and np.min(silent) == 0:”

Over two reads, if the minimum was 0 and the max was 20, the program would detect a silent frame. This was done due to the volatility of silent frames, sometimes they were extremely fast and sometimes they were slow. Thus, two reads, one being max and one being mix, allowed the system to see a decrease in volume and then an arrival at 0. This reliably caught transitions and mitigated false transitions. The program would then sleep for one second to ensure the trigger did not trigger again causing the television to unmute.

For moving averages, the program made a second array.

“average.append(volume)”

This array is kept separate from the silent array since the silent array gets cleared on each trigger. This array gets cleared after 10 reads and then saved to a variable. This variable is then compared to the next 10 reads. However, this never became useful as it was highly dependent on the commercial that followed the transition.

“if j % 10 == 0:”

“if m == 0”

“x = np.mean(average)”

“m = 1”

“if m == 1”

“y = np.mean(average)”

“m = 0”

## TV Remote

For the MuteBot to communicate and interact with the provided Westinghouse Television, IR signals would have to be sent from the MuteBot’s IR transmitter to the TV’s IR receiver. The IR transmitter and receiver used on the MuteBot system was the Geekworm Raspberry Pi IR remote expansion board. This included an IR transmitter, receiver, and two push buttons. This was installed by attaching the IR shield to the Raspberry Pi so that the IR shield board hovers over the Raspberry Pi. Then, plug in the IR shield at the farthest pin down towards ‘J8’ on the Raspberry Pi. Since the MuteBot needs power and connection to the TV’s 3.5mm jack, a universal IR remote was part of the design. The IR remote would boot up the system, start/end commercial detection, and toggle between mute and ten volume clicks lower than the original volume. Communication between the IR remote and MuteBot was done by using the Geekworm IR expansion board. The IR transmitter would send signals to the MuteBot’s IR receiver.

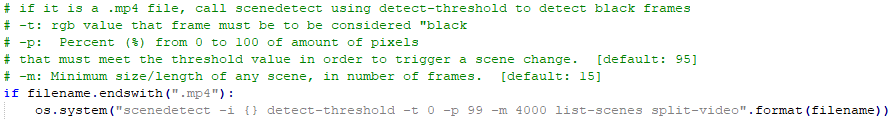
Sending IR signals to and from the MuteBot required software to initially set up IR expansion board and to send IR signals to the Westinghouse TV to mute or lower its volume. The software used for this process is called LIRC. This piece of software runs the IR communication on the expansion board. For a detailed set of instruction required to initialize the IR expansion board, it is found in section 12, 11/24-12/1. When sending a mute/volume down command to the Westinghouse TV, a specific file for the Westinghouse remote had to be created and inserted into the LIRC software’s “lircd” config file. This can be shown as well in the group’s github located at the end of this report. From prototyping with the irsend –count=8 SEND\_ONCE ESTINGHOUSE KEY\_VOLUMEDOWN and irsend –count=8 SEND\_ONCE ESTINGHOUSE KEY\_VOLUMEUP, the count number in the command line changed the TV volume up and down when pressed. This number was chosen to send ten volume down and up commands to the Westinghouse TV.

While developing and testing the MuteBot’s IR communication, the team decided to remove the universal IR remote from the design. This was due to compatibility issues with LIRC. In the LIRC configuration file lirc\_options.config, the device name would correlate to either using the expansion board’s IR receiver or transmitter. If the device name were lirc0, the IR transmitter could be used. If the device name were lirc1, the IR receiver could be used. The team brainstormed ideas of how to write a program that would address this issue. The idea was to go into the lirc\_options file and automatically change the device name. This would allow both the IR transmitter and receiver to be used. Thus, the remote would be able to both send a command to the Pi as well as the Pi send commands to the TV. However, it was found through research that when the device name was changed, the entire Raspberry Pi needed to reboot to apply the changes needed to change communication. From these finding, the team met with the customer and decided to use the IR expansion board’s two buttons to start/end commercial detection, and to toggle between either muting or lowering the TV volume when a commercial occurs.

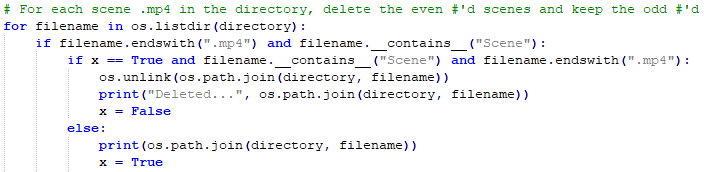
The two buttons including on the IR expansion board were connected to pins 27 and 22 on the Raspberry Pi 3B. The functionality would be the same as the IR remote but now only the IR transmitter was needed to communicate with the TV. This solved the issue of starting/stopping the commercial detection program. Python was used to control the button presses as well as issue the commands. When button 1 is pressed the MuteBot will toggle the commercial detection algorithm. Button 2 would change from muting the TV to reducing the TV volume by 10 levels. This can be shown in Table 9 of the Test Results which shows the tests ran for the TV remote communication. A detailed look of the Python script for the button pressed program can be found in the group’s github. In conclusion, for the final MuteBot system, a button was substituted for the IR remote due to LIRC constraints in using both the IR transmitter and receiver simultaneously.

## Program Flow

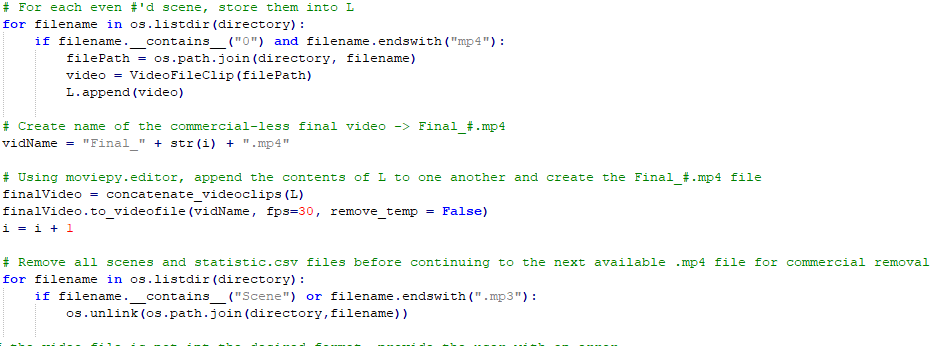
In order to work around the unfortunate events of COVID-19, the team decided to create a program that would be able to eliminate commercials from a prerecorded show. Similar to how comcast and TiVo eliminate commercials from their offerings, black frame detection was used to identify transitions between show and commercial break. To do so, the PySceneDetect library was used to read a .mp4, detect black frames, and split the video into scenes based upon those detections. The Detect-Threshold command allows a user to set a specific RGB value, percentage of pixels, and minimum length of frames that are required for the program to trigger on a set of black frames. This allows for expanded customization and the ability to pinpoint the best threshold levels. Figure 2 provides the flow of the new commercialDetector program. At start, provided the executing folder of the program, it will begin by parsing through the folder and running the scene detect function on any .mp4 files it finds. If there are other video file formats, it will provide the user an error message asking for .mp4 files only. When run, the scenedetect function will load and downscale the video and begin averaging the RGB value of each frame and comparing it to the given threshold. If the segment of frames meets the requirements given by the parameters, a scene detection will trigger. Below is the function call that was used in the team’s implementation:



Once all detections have been made, the function then splits the given .mp4 file into separate scene .mp4 files based upon those detections. At this point, the program parses through the folder and deletes the even numbered scenes. As traditionally a recorded show would begin with a segment of the show, then transition to a commercial break. Then it would alternate between show and commercial break for the remainder of the show. Using this assuming, deleting the even numbered scenes would delete the commercial break segments. Below is the loop the team used to delete the commercial break scenes:



After all commercial break segments had been deleted, the show segments are stored into an array of .mp4 and later appended into a singular final.mp4 file to output. After the final.mp4 video is written into the original folder, the remaining files used to create the final.mp4 are then deleted from the folder and the program moves on to the next available .mp4 file to eliminate commercials from. Below is the implementation the team used to create the final video and remove the additional files from the directory:



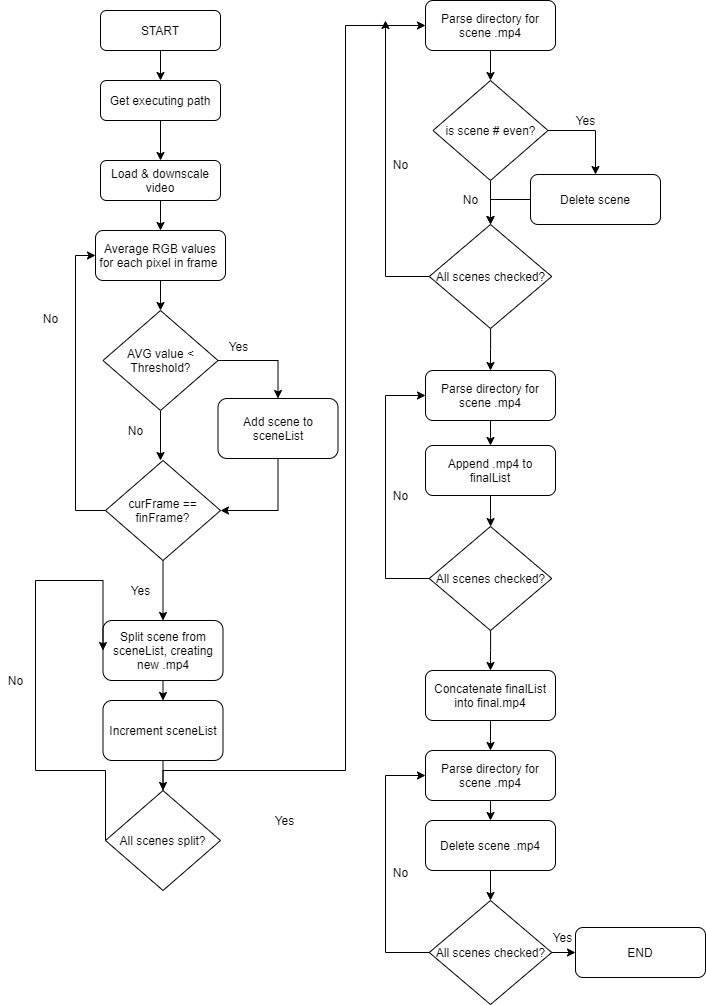


Figure 2: Program Flow of commercialDetector

# Test Plan and Results

## VCRM

Table 4 provides a verification cross-reference matrix in order to highlight the different verification methods being used to verify each of the engineering requirements for the system. The test mapping column shows which table in the section 7.2 to refer to regarding that engineering requirement. Table 10 and 11 refer to the groups progress prior to the pandemic. It refers to S.7 and S.8, which have been omitted for this report in favor of the programming requirements. In the test mapping, UR refers to user-review where the user will be given the questionnaire shown in section 12, week 10/20-10/27. If the review scores less than a 3 in any category that category will be subject to improvement. The questionnaire was never able to be used as a result of the pandemic and lack of a prototype to hand out.

Table 7: Verification Cross-Reference Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Customer Requirement | Engineering Requirement | Test Mapping | Inspection | Analysis | Demonstration | Test |
| P.1 | P.1 | **14** | **X** |  |  | **X** |
| P.2 | P.2 | **14** | **X** |  |  | **X** |
| P.3 | P.3 | **14** | **X** |  |  | **X** |
| P.4 | P.4 | **15** | **X** |  |  |  |
| P.5 | P.5 | **15** |  |  |  | **X** |
|  | P.5.1 | **15** | **X** |  |  | **X** |
| S.1 | S.1 | **12, UR** | **X** |  |  |  |
| S.2 | S.2 | **8, UR** | **X** | **X** |  |  |
|  | S.2.1 | **8, UR** | **X** | **X** |  |  |
|  | S.2.2 | **8, UR** | **X** | **X** |  |  |
| S.3 | S.3 | **8, UR** | **X** | **X** |  |  |
| S.4 | S.4 | **9** |  |  | **X** | **X** |
| S.5 | S.5 | **13** |  | **X** |  |  |
| S.5.1 | S.5.1 | **13** |  | **X** |  |  |
| S.6 | S.6 | **8** | **X** |  |  |  |
| ~~S.7~~ | ~~S.7~~ | **~~10, 11~~** |  |  | **~~X~~** | **~~X~~** |
| ~~S.8~~ | ~~S.8~~ | **~~10, 11~~** |  |  | **~~X~~** | **~~X~~** |
|  | ~~S.8.1~~ | **~~10, 11~~** |  |  | **~~X~~** | **~~X~~** |

## Test Cases and Test Results

Table 8 shows the initial setup test for the MuteBot system. This test has been completed and done many times as the group has had to setup the MuteBot system on multiple different occasions to continue work on the project. In the “Observed Result” column, more descriptive results have been added.

Table - MuteBot Setup Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass  Fail |
| 1. | **Plug** the **Westinghouse TV Power cord** into an outlet. | The Westinghouse TV has Power | Red LED turns on, located at the bottom right of the TV. | Pass |
| 2. | Using the **Westinghouse TV** remote **press the Red Power Button** | The Westinghouse TV powers up (Blue Light indicates Westinghouse TV is on) | Blue light has turned on and the Westinghouse logo appears on the TV display | Pass |
| 3. | **Plug the HDMI cable** into both the **Raspberry Pi** along with the **Westinghouse TV** | Display between the two systems are connected (After Power is applied) | Cable is firmly connected. | Pass |
| 4. | **Plug** a **keyboard** and a **computer mouse** into the **bottom two USB ports** on the **Raspberry Pi** | Mouse and Keyboard will now work on the Raspberry Pi (After power is applied) | Cable is firmly connected into USB port. | Pass |
| 5. | **Plug in** the **USB soundcard** into the top right USB slot | USB soundcard now inputs into the Raspberry pi | Soundcard is firmly connected. | Pass |
| 6. | Plug one end of **3.5 mm jack** into the **Westinghouse TV** and plug the other end into the **pink port on the USB sound port** | The Raspberry Pi can get sound in from the Westinghouse TV (After Power is applied) | Cable is firmly connected. | Pass |
| 7. | Attach the **IR shield** to the **Raspberry Pi** so that the **IR shield board hovers over the Raspberry Pi**, Plug in the IR shield at the **farthest pin down towards ‘J8’** on the Raspberry Pi | The IR shield can now interact with the Raspberry Pi (After Power is applied) | IR Shield is firmly connected. | Pass |
| 8. | **Plug** in one end of **a micro USB cable** into an **outlet** and **plug** the other end of the **micro USB cable** into the **micro USB port on the Raspberry Pi** | Starts up the Raspberry Pi and shows the Raspberry Pi desktop | Pi starts, keyboard now interacts with pi, display shows the Pi logo | Pass |

Table 9 shows the system remote test for the MuteBot system. The “action” column has been changed to more accurately show the test performed. The system remote’s purpose is to send a frequency from the Pi to the TV that mimicked that of the remote that is provided with the TV. This allowed for commands to be sent independently from the Pi, which allowed for muting and volume lowering of the TV.

Table - System Remote Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass/  Fail |
| 1. | Click the **TOP BUTTON** on the IR Shield.  ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON 1~~**  *This change was made since the system remote was not an actual remote, but rather buttons on the Pi that mimicked the same frequency as a remote.* | Python console output  displays “++Algorithm is ON, Mute is selected” | “++Algorithm is ON, Mute is selected” is shown on the computer’s display. | Pass, see comments for clarification |
| 2. | Click the **TOP BUTTON** on the IR Shield.  ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON 1~~**  *This change was made since the system remote was not an actual remote, but rather buttons on the Pi that mimicked the same frequency as a remote.* | Python console output displays “++Algorithm is OFF” | “++Algorithm is OFF” is shown on the computer’s display. | Pass, see comments for clarification |
| 3. | Click the **LOWER BUTTON** on the IR Shield.  ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON 2~~**  *This change was made since the system remote was not an actual remote, but rather buttons on the Pi that mimicked the same frequency as a remote.* | Python console output displays “++Reduced Volume ON” | “++Reduced Volume ON” is displayed onto the computer screen. | Pass, see comments for clarification |
| 4. | Click the **LOWER BUTTON** on the IR Shield.  ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON 2~~**  *This change was made since the system remote was not an actual remote, but rather buttons on the Pi that mimicked the same frequency as a remote.* | Python console output displays “++Reduced Volume OFF” | “++Reduced Volume OFF” is shown on the computer’s display | Pass, see comments for clarification |
|  | On the Desktop, select **StartingPycharm.sh** | Execute File dialog box appears | The Execute File dialog box is shown on the computer monitor. | Pass |
|  | Select **Execute** in the Execute File dialog box | Pycharm opens to last opened file | Pycharm boots up and displays the python code from the most recent file | Pass |
|  | Select **File** in top left corner, Select **Open** | Open File or Project window opens | The File or Project window is shown on the computer monitor | Pass |
|  | Type in search bar **/home/pi/remotecontrol\_v1/buttonPressed.py**  Then press **OK** | The buttonPressed program will open | The buttonPressed project file opens after clicking OK. The python code from this file is shown on the computer monitor. | Pass |
|  | Select **Run** at the top of the screen, Select **Run “buttonPressed”** | Python console will open and button presses will now be read | PyCharm’s console window appears on the computer monitor and waits for buttons to be pressed from the MuteBot system. | Pass |

Table 10 shows the commercial detection test for the MuteBot system. Step 3 and step 4 have failed. These have never been corrected due to the impact of COVID-19, which is expanded upon in section 11.2. Prior to the pandemic, step 3 failed due to the number of false positives given. The system did succeed in detecting a cut to a commercial reliably. However, there was no way for the program to realize when the next detection was indeed a return to the regular programming. Detections would happen every time a commercial switched to another commercial. Due to the volatile and unpredictable nature of on-air television, there was no way for the system to reliably predict when a return was going to happen using only audio. Commercial time, frequency, and transition time varied by channel, show, and reception. Therefore, there were too many variables to find a trend in the amount of time available. Thus, the return algorithm was never fully implemented given the constraints of both working time and capabilities.

Table - Commercial Detection Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass  Fail |
| 1. | ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON NAME 1~~**  Click the **TOP BUTTON** on the IR Shield. | Output to terminal reads **“++Algorithm is ON”** | Terminal reads **“++Algorithm is ON** | Pass |
| 2. | ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON NAME 2~~**  Click the **LOWER BUTTON** on the IR Shield. | Output to terminal reads **“++Reduced Volume ON”** | Terminal reads “**++Reduced Volume ON”** | Pass |
| 3. | Upon a transition to a commercial break observe the terminal | Output to terminal reads **“++Commercial Detected Volume Lowered”** | Inconsistent Behavior | Fail |
| 4. | Upon a transition back to TV programming observe the terminal | Output to terminal reads **“++Return Detected Volume Restored”** | Inconsistent and not fully implemented | Fail |
| 5. | ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON NAME 2~~**  Click the **LOWER BUTTON** on the IR Shield. | Output to terminal reads **“++Reduced Volume OFF”** | Terminal reads “**++Reduced Volume OFF”** | Pass |
| 6. | Upon a transition to a commercial break observe the terminal | Output to terminal reads **“++Commercial Detected Muted”** | Inconsistent Behavior | Fail |
| 7. | Upon a transition back to TV programming observe the terminal | Output to terminal reads **“++Return Detected Unmuted”** | Inconsistent and not fully implemented | Fail |
| 8. | Point the system remote to the Raspberry Pi and press **BUTTON NAME** | Output to terminal reads **“++Algorithm is OFF”** | Terminal reads **“++Algorithm is OFF”** | Pass |

Table 11 shows the TV remote test for the MuteBot system. Step 2 and step 3 is a pass/fail due to the lack of a fully implemented commercial detection algorithm. The IR transmitter on the Pi properly sent a mute command to the TV when a detection happened, but it also sent when the commercial detection algorithm had a false positive. Thus, this created a unpleasurable viewing experience to the user and did not work to its fullest potential. However, it did accomplish the main goal of sending a mimicked signal. This is the same case for step 5 and 6. The IR transmitter correctly sent exactly 10 volume down commands whenever a detection was found, and this setting was set to on.

Table - TV Remote Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass/  Fail |
| 1. | ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON NAME 1~~**  Click the **TOP BUTTON** on the IR Shield. | Commercial detection program will begin to search for commercials with mute selected | ~~Commercial detection program is not completed at this time.~~  Commercial detection starts, console window reading “**++Algorithm is ON.”** | Pass |
| 2. | When commercial is detected send a mute command from IR transmitter to the Westinghouse TV | Westinghouse TV displays mute, Audio symbol with X over it, and mutes volume output | Westinghouse TV audio has been stopped and can no longer hear TV audio. Audio symbol with X over it is shown on the bottom left corner of the TV display. | Pass/Fail |
| 3. | When the commercial end is detected send a mute command from IR transmitter to the Westinghouse TV | Westinghouse TV display for mute disappears and audio level is restored | Westinghouse TV audio has been restored and can hear the TV program. The mute icon on the TV has disappeared. | Pass/Fail |
| 4. | ~~Point the system remote to the Raspberry Pi and press~~ **~~BUTTON 2~~**  Click the **LOWER BUTTON** on the IR Shield. | Commercial detection program will begin to search for commercials with volume decrease selected | ~~Commercial detection program is not completed at this time.~~  Commercial detection continues with the lowered volume option, console window reading “**++Reduced Volume ON.”** | Pass |
| 5. | When commercial is detected irsend a KEY VOLUMEDOWN command from IR transmitter to the Westinghouse TV | Westinghouse TV displays volume level bar going down ten button presses | Westinghouse TV audio from the speakers has been lowered. This is also indicated with a volume level bar shown on the bottom of the TV display. A vertical bar indicating the current audio level is moved by 10 spaces to the left. | Pass/Fail |
| 6. | When the commercial end is detected irsend a KEY VOLUMEUP command from IR transmitter to the Westinghouse TV | Westinghouse TV displays volume level bar going up returning to its previous volume | The volume level bar reappears on the TV display and the vertical bar goes to the right 10 spaces. The TV audio has been increased to its original volume level. | Pass/Fail |
|  | On the Desktop, select **StartingPycharm.sh** | Execute File dialog box appears | The File or Project window is shown on the computer monitor | Pass |
|  | Select **Execute** in the Execute File dialog box | PyCharm opens to last opened file | The final project file has not been created. It opens up the most recent project file | Fail |

Table 12 is incomplete due to not having the means to accomplish it. A schematic has been made of what the 3D print would have looked like. It can be found in section 12 under 3/8-3/14. It is predicted that both results would have passed if the group was able to print this enclosure. The schematic made was to scale and taking measurements on a TV stand allowed us to assume it would not be obtrusive.

Table – Enclosure Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass  Fail |
| 1. | Attach both sides of the shell onto the Raspberry Pi | Raspberry Pi is enclosed with all exterior ports exposed | MuteBot system’s enclosure is not 3D-printed and assembled | Incomplete |
| 2. | Place the Enclosure 2 in. in front of the Westinghouse TV’s IR sensor | Westinghouse TVs display and IR sensor is not obstructed from the viewer | MuteBot system’s enclosure is not 3D-printed and assembled | Incomplete |

Table 13 shows the cost analysis test for the MuteBot system. Step 2 and 3 have failed, by being over the target range by nearly $55 assuming a 35% manufacturing cost on the initial resources. However, this comes with a disclaimer. The target number was arrived upon using MuteBot’s competitors, Mutr and MuteMagic. MuteMagic was made years ago and no longer work on today’s televisions, thus their cost can not be fairly compared. Mutr is still in development and has been in development for 3 years now. It also comes with the caveat of a $50 yearly subscription. There product has not been released to the public yet either, thus invalidating their cost. Therefore, the group feels the cost currently could be decreased, but still lies within a competitive range.

Table - Cost Analysis Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass/  Fail |
| 1. | Document cost of competitors products. E.g. **MuteMagic, Mutr, etc.** | Costs average in range of $50-$100 | MuteMagic sells for $40 | Pass |
| 2. | Sum the **cost of resources + manufacturing cost** of MuteBot. | Cost sums in range $40-75 | $130.23 manufacturing cost | Fail |
| 3. | Multiply build cost by **“Sell price” factor** to calculate retail cost | Retail cost falls within $50-$100 range. | $195.34 with 50% profit margin | Fail |

Table 14 shows the program file integrity test for the program system. Step 4 has pass/failed due to one false positive. The significant drop in audio as well as darkness of the frame triggered a false positive queuing the program to not cut the remainder of that commercial. With more time, the group could get rid of this false positive by implementing a color average, which would take the average color of “X” amount of frames and compare that to when the program knows the television show is on, if the color differs by a large amount than it can be assumed this false positive is a commercial and not the return to the show. However, with the limited time frame as well as learning a new technique when it comes to this program, the group has not been able to implement this.

Table - Program File Integrity Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass/  Fail |
| 1. | Select the .mp4 name “TestFile.mp4” | Program begins to run parsing through the .mp4 file. | Program begins to run showing progression in the console window. | Pass |
| 2. | Once program is completed, observe a file is outputted | Program outputs file named “output1.mp4” | Program outputs single file named “output1.mp4” to target directory | Pass |
| 3. | Open the file using VLC media player. | New window opens and “output1.mp4” begins to play automatically | When double clicked, VLC media player begins playing automatically | Pass |
| 4. | Watch “output1.mp4” entirety ensuring no commercials are present throughout the media and all main media is kept intact. | All commercials that were in “TestFile.mp4” are no longer in “output1.mp4”, but all main media is still intact. | All main media is intact there is one false positive that triggers on one of the commercial breaks, triggering an early return | Pass/Fail |

Table - Program File Type Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Step | Action (Attach test data, diagrams, etc. as appropriate) | Expected Result | Observed Result | Pass/  Fail |
| 1. | Select directory that contains no “.mp4” files | Program rejects the file prompting console window to pop up. | Console window pops up | Pass |
| 2. | Observe if console window pops up and text reads “?No .mp4 files found. Select directory with .mp4 files” | Console window reads “?No .mp4 files found. Select directory with .mp4 files” | Console window reads “?No .mp4 files found. Select directory with .mp4 files” | Pass |
| 3. | Reopen the program and input the directory with “TestFile.mp4”. then wait for program to finish. | Program begins to run. Outputs .mp4 file on completion | Program beings to run | Pass |
| 4. | Observe to ensure the output file is in .mp4 format. | Output file is “output1.mp4” | File is outputted in .mp4 format | Pass |

## Test Summary

Overall, despite the failure of some of the tests, the group now understands what must be done in order to achieve success in those failed categories. The tests involving hardware have all passed except for having a remote that turns on the system from afar. However, this is deemed impossible to work with the selection of the current hardware due to the Raspberry Pi needing a restart each time it switches from IR Receiver mode to IR Transmitter mode. This is a minor setback as this only affects the quality of life of the product as it will just have to be started and modified from the device itself. The commercial detection tests, although not working, have results that appear promising with more time and additional fail-safes. The commercial detection is not predicted to have worked 100%, but with the addition of digital image processing this project can realistically be completed and in the markets. Further detail can be found in section 11.4 explaining what the results of these tests mean for the success of the project. The test summary can be seen in table 16.

Table 16 - Test Summary

|  |  |
| --- | --- |
| **Test** | **Pass/Fail** |
| Setup | PASS |
| System | PASS |
| Commercial | FAIL |
| TV | FAIL |
| Enclosure | INCOMPLETE |
| Cost | FAIL |
| Program File Integrity | PASS |
| Program File Type | PASS |

# Risk Discussion

Table 17 provides an overview of the risk and potential problems that are related to each of the major components of the overall system. The Raspberry Pi 3b was chosen and for the original MuteBot design this choice had worked out well with the design of the whole system. The Raspberry Pi 3b was also useful and helpful while creating the program to cut out commercials for “The Price is Right.” The only other high-risk potential problem within this project was COVID-19 emerging. This virus had blocked off access to lab along with face to face meetings with partners. This had a big impact on the project, which inherently increased the overall risk. The next biggest risk was choosing the programming language, the language that was agreed upon was python. Since both aspects of this project used python, the choice of this risk ended up working well. With the changes of this semester, the group was able to switch from real-time processing to post-processing with ease. The type of interface to interact with the TV carried a fair amount of risk. The final risk was a low-risk problem and that was the controller interface. IR was chosen, however due to the change of the project, this decision did not truly affect the final outcome.

Table 17: Potential Problems & Risk Analysis

|  |  |
| --- | --- |
| Potential Problems | Risk Factor |
| Choosing the incorrect processor/development board. | **High Risk**   * Increase time and cost to change design. * Effects other major components. |
| Pandemic causing the UMASS campus to shut down | **High Risk**   * No access to lab equipment along with project equipment. * Lesser quality meeting times since no hands on work to get done. |
| Choosing the incorrect programming language | **Medium Risk**   * Increase time and cost to change code for implementation * Time to learn new programming language if there is a steep learning curve |
| Choosing the incorrect TV interface | **Medium Risk**   * Increase time and cost to change design * Would affect detection program (input signals for detection) |
| Choosing the incorrect remote technology | **Low Risk**   * Increase time to change remote design * Both alternatives are compatible with all potential processors. |

# Plan, Schedule, and Costs

## Plan and Schedule – Final

At the start of the spring semester, the team created a schedule that includes task breakdowns, members responsible for tasks, planned and estimated start/end dates, and total hours. Throughout the semester, the team updated the Gantt chart and actual hours in order to track progress and identify potential bottlenecks in the project. Unfortunately, due to COVID-19, the project needed to be reworked as a whole. To highlight this, a new section was added to the schedule to track the team’s progress with the commercial detection alternative. Refer to the plan and schedule, ECE-6 Plan And Schedule.xlsx, which is provided in the same folder as this report.

## Actual Hours vs. Planned Hours

Through the duration of the project, the team recorded their actual hours spent on each task, report, and presentation. Compared to the initial planned hours for the project, the actual hours spent falls much shorter. Overall, the team had planned to spend 383 hours to complete the project this semester, for a labor cost of $5,745.00. However, after adjusting to the current situation, the team spent 167 hours, for a labor cost of $2505.00. Once the campus had shutdown and the team had lost access to the essential equipment needed, focus was quickly shifted to creating an alternative for the commercial detection subsystem. As this solution was a entirely separate approach, it could not be integrated with the previous remote control work that the team had done prior to COVID-19. Thus, the team was not able to start integration on the previous system or continue work on the system enclosure. This greatly skewed the number of actual hours as integration took up a large portion of the teams planned hours.

## Cost Summary

Table 18 provides a list of components and software used to create the MuteBot system. Next to each component is the price of the item or if the component was supplied by the university. Solidworks was replaced with Sketchup due to the prior experience and familiarity of using this piece of software. This is highlighted in green. All software that was used in the project was either free or provided by the university and is shown in the table below. Table 18 provides a list of components needed to build the MuteBot system. If the system were to be manufactured and sold to the global market, this table provides only the components needed to manufacture the product. The final price of the product with manufacturing cost was higher than anticipated and is reflected upon in greater detail in table 13 of the test results.

Table – Project Cost Table

|  |  |
| --- | --- |
| Resource | Cost |
| Raspberry Pi 3 | $38.36 |
| Geekworm Raspberry Pi IR remote expansion board | $12.09 |
| Sabrent USB external stereo sound adapter | $6.99 |
| Energizer 2025 Lithium Battery | $7.43 |
| Sandisk Ultra Plus 16G GB microSD card | $10.61 |
| HDMI to VGA adapter | Supplied by UMASS Dartmouth |
| LIRC (IR communication software) | Free software |
| 5V DC Power supply | $9.00 |
| AV Splitter | $5.00 |
| 3.5mm Audio Cable | $6.99 |
| Westinghouse HDTV & Remote | Provided by customer |
| TV Antenna | Provided by customer |
| HDMI Cable | Supplied by customer |
| 3D Printer | Supplied by UMASS Dartmouth |
| Atmel Studio 7.0 | Software provided by UMASS Dartmouth |
| Audacity | Free software |
| SketchUp | Free software |
| Total | $96.47 |

Table – Product Cost Table

|  |  |
| --- | --- |
| Resource | Cost |
| Raspberry Pi 3 | $38.36 |
| Geekworm Raspberry Pi IR remote expansion board | $12.09 |
| Sabrent USB external stereo sound adapter | $6.99 |
| Energizer 2025 Lithium Battery | $7.43 |
| Sandisk Ultra Plus 16G GB microSD card | $10.61 |
| 5V DC Power supply | $9.00 |
| AV Splitter | $5.00 |
| 3.5mm Audio Cable | $6.99 |
| Total | $96.47 |

# Summary

As an overview of the two semesters, the team was able to successfully detect commercials and eliminate them from the TV program. Using the Python program, 90% of all commercials were removed. Each scene of the TV program was created separately, and the scene associated with the commercials was then removed, resulting in a TV program with no commercials. Before the pandemic, the team was able to detect commercial audio spikes, but the rate of false positives was high. This was one aspect that the team planned to improve upon for the final product. Due to the inability to work on the project in the lab, the team was unable to continue to finalize the system’s design. Since the team transitioned to a new design of detecting commercials, the Python program, as mentioned previously, had some aspects that did not go in the team’s favor. This includes one false positive that triggered as a result of an abnormally dark commercial.

Some suggestions that the team would give for future work is to make sure the team brings all equipment needed with their team if they are leaving the campus for an extended period. One fault, that was partially the team’s responsibility, was not bringing the MuteBot home over break. However, given the novelty of such an event, the group does not believe anyone could have predicted this outcome. This caused the team to change project design ideas entirely and change many customer requirements. Another suggestion for future work would be to associate time in the schedule for possible setbacks that may occur. For instance, the pandemic eliminated time in the schedule and if time were planned for a possible occurrence, it could have changed the outcome of the project. The team has now reported on two separate project ideas that contain similarities. As a result, if both aspects worked to expectations, the customer would receive a product that worked for both post and real-time processing. In terms of real-time processing, if the customer were able to eliminate the false positives, the system would be able to work according to expectations. The python program, despite not working 100%, it still provides 90% accuracy. Thus, giving the customer a product usable on all pre-recorded television broadcasts. Overall, the project was successful in teaching the group how to be effective engineers as well as teaching aspects of real-time audio processing.

# Lessons Learned

## Lessons Learned

The group would like to mention how this affected our learning experience throughout the semester. In our group’s opinion, we learned more this semester with this pandemic than without. The essentials of engineering management and organization were already engrained with the previous reports of this semester as well as ECE 457. We feel as if we also accomplished enough work prior to the incident to gain a grasp of real-time audio processing. Although we did not get the final product we wanted, we do believe we gained the skills necessary to apply the concepts in the work force. With time, we do feel as if we could have a product capable of meeting all prior customer requirements. However, with the introduction of this pandemic we also learned how to overcome severe obstacles. We learned how to work remotely efficiently, how to communicate with each other, and how to adapt in the face of adversity. In the future, we also learned that we should avoid relying on one main goal without a backup plan, especially if that goal heavily relies on something that could be restricted. Thus, if we could foresee the future, we would have brought the hardware home prior to the pandemic or invested in extra hardware for us to work separately.

Overall, this pandemic has challenged us all educationally, mentally, and physically. Despite this, we feel we will come out stronger as not only scholars, but as people. Our product may not be the product we envisioned, but we came out learning more than we originally anticipated. We have learned the essentials of engineering production, management, teamwork, and scheduling from this course. In the work force, we can now apply the skills learned both in ECE 457 and ECE 458 to our prototyping and testing to be effective engineers. This pandemic may have hindered our progress, but it will not hinder us on our path as engineers.

## Impacts

COVID-19 was a completely unprecedented event that has hindered our project’s production immensely. For the most part, our group was on track to complete our prototype on April 25, 2020. We made monumental steps throughout the design process, leaving the month of March and April to further develop our algorithms to ensure greater accuracy and decrease the number of false positives. The group worked adamantly and efficiently over the beginning months of the semester since we aimed to complete the project early, leaving time for improvement beyond customer requirements if we desired. After class recess on March 6, Kevin and Zach continued to work on the algorithm portion of the project. Tom and Steve finalized the hardware portion of the project. We went into the lab three times the first week of spring break. Thus, we never saw an immediate need to move the equipment into one of our households. On the second week of spring break UMASSD announced that the campus will be shutdown. Following this week, Kevin and Zach requested to gain access to the lab equipment but were denied the access. On April 16 one of our members were given lab access, but at this point with only two weeks till presentation and the inability to work in person, it was not feasible for us to meet our deadlines. This section will discuss in-depth how we were affected, how we adapted, and what we learned.

The largest hindrance in our progress was the lack of access to equipment. Thus, we began work on our backup plan of creating a program that accomplished a similar goal. The only issue was that it was not real-time processing, but this was established with our customer before starting production. We did not have access to the equipment for a little over a month, only gaining access with two weeks left till our deadlines. Even with access being granted, it still wasn’t within WHO guidelines to have the four of us within a six-foot distance in one room with each other. We attempted to use other means of communication, such as Discord or Zoom, but with most of the content being on the Raspberry Pi, it was near impossible to see what the other user was doing as we couldn’t share screens on that computer and run our program effectively. This also always required our team lead to be present as he was the one with the equipment. Realizing the difficulty in this, we decided to stick with our program that closely reflected our original design as we could all work on in separately. It was also more efficient when in group meetings as it used our more powerful machines that allowed us to code and stream at the same time.

Each member individually struggled as well during this pandemic, which led us to not have our minds particularly engaged in the project as much as we would have on the campus. Kevin continued to work as an intern for General Dynamics and had his hours increased to 20 per a week during the pandemic. Tom had pre-existing health issues prior to this pandemic, which in turn led him to be more at risk to the virus, preventing him from interacting with us physically or the materials we had contact with. Steve is an essential worker with many of his colleagues leaving work during this pandemic, thus he had to pick up more hours in order to still fulfill obligations at work. Zach’s mother received treatments prior to this pandemic as a result of breast cancer, along with having preexisted immune system disorders. This has led her to be immunocompromised, thus anything that comes into his house needs to be disinfected and no one is allowed in or out without disinfecting. Thus, Zach could not have retrieved the equipment or gather with other people.

From a production standpoint with our new program, there presented new challenges that we had to solve in half the time of our previous product. Tom and Steve are electrical engineers, which to no fault of their own, made them less adept at coding. Thus, our program idea mainly relied on Kevin and Zach, while Steve and Tom worked on the report aspect as well as learned the design process in our new code. It also shifted our entire project away from real-time processing to post-processing, which resulted in completely revamping our code as the previous code was basically unusable. The inability to meet in person also hindered our progress as it does with any production.

## Innovation

The project required a complete overhaul as a result of COVID-19. With lack of access to the hardware for over a month and the lack of physical communication, the group had to find a way to communicate and engineer efficiently without the two. Thus, on March 23, when we realized that we will not be able to retrieve our equipment, we decided that coding a program to accomplish a similar goal is our best bet. We then designed the remainder of this project around this program, while theorizing how the program would be implemented into our overall system. We ensured the program utilized the similar techniques as our main system to detect the commercials. This is because we wanted to prove that it was indeed still possible to complete this project if this pandemic did not happen and time allowed for it. Along with the program, we added new documentation as well as keeping old documentation to indicate that the project went in a new direction. We decided the best course of action was to write and present this project as combination of both the program and system. This resulted in two different parts to certain sections in our report and presentation.

## Steps to Success

This section will explain what must be done for the project to be deemed a 100% success if continued in the future. On-air television is an extremely volatile and unpredictable environment. As described previously, the program this prototype used acted upon silent frames and audio abnormalities. These abnormalities consisted of audio spikes and high audio moving averages. However, these detection methods alone are not enough to create a stable and working product in the current day. These methods are obsolete since the CALM act was introduced, which is talked about in section 3.3. To summarize, this act made it so commercial average volumes could not be louder than that of regular television programming. Before this act, every commercial was loud enough to be easily detected using just audio. This product existed at the time called MuteMagic. This product is now obsolete due to this act.

With this information now gathered, digital image processing is a MUST to achieve the project goals. Asking Dr. Benjamin Viall about digital image processing back in October of 2019, he highly discouraged us due to the difficulty of such a concept. However, the group now realizes, despite the difficulty, not using it is not an option. Commercials, in terms of audio, are too variable to rely on. They do however always start with a silent frame and end with a silent frame. This makes it easy to see when commercials are switching or when a television program switches to a commercial break. There is no way to determine when one of those triggers is a trigger back to the television programming. The best way to do this was to implement a sleep(60 seconds) command when the first trigger was found. This would ensure the program would mute for a bare minimum of 60 seconds. However, this would prevent the user from hearing their show if the commercials ended earlier (which was rare). The downside lied when the commercials lasted longer than this time frame, which would cause the user to still hear commercials before their show returned. Also, if there were more triggers after these 60 seconds it would cause a sleep to happen again unless the sleep was disabled for 15 minutes after being activated, but this is still not ideal. Audio moving averages were useless due to the timeframe being worked with. In order to make the moving averages useful it would need a longer buffer period, but at that point the product is not providing its original goal. Thus, in accordance with the testing, there was no workaround using just audio.

Through the group’s research, digital image transitions could greatly improve the accuracy of the overall system. The first trigger that is easy to detect are black frames. If every pixel on the screen is under a threshold, then it is considered black, which in turn usually means there is a transition happening. This is a good leading indicator along with silent frames to be 100% sure there is a transition happening and not just a silent or dark section. Next is shot cuts, this can be done using color averages. Shows and movies tend to use the same relative color and progressively change the color over the course of the movie, unless there is a scene transition. However, these scene transitions happen infrequently over a 10 second period. Thus, taking a moving average of the average color of the pixels across the screen would allow for a trailing indicator. This means if the leading indicator, silent black frames, were detected, the trailing indicator would begin to check if this transition is indeed valid. If the color average was drastically different and kept changing drastically over the next 10 seconds than this is indeed a commercial. If the color average was remaining the same this could be a return to the show. Closed captioning digital image processing could also be used to trigger a commercial on certain key words. Words that are associated with major products/brands could be listed as a trigger word. This is like how Mutr operates. Potentially extending the buffer would only increase the accuracy of the overall system.

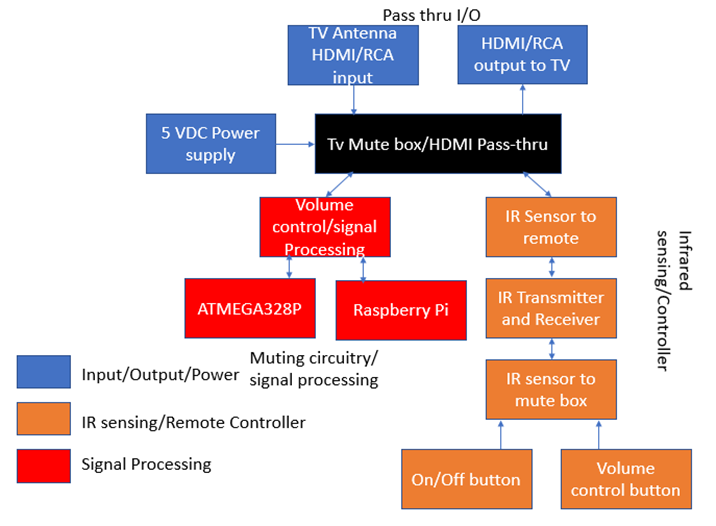
# Documentation

**9/15-9/22:**

* Started gathering resources and doing research towards project. Started working on customer requirements, Engineering Requirements, along with constraints and standards.
* Research on previous brands conducted:
  + MuteMagic (Audio Processing): <https://bgr.com/2015/05/25/mute-tv-commercials-automatically-how-to/>
  + MutR (White Listing): <https://mutr.com/>
  + Kommercial Killer (Machine Learning): <https://blog.adafruit.com/2018/06/20/kommercial-killer-silences-your-tv-during-commercials-using-trinket-oshpark-pcbs-via-circuitcellar/>

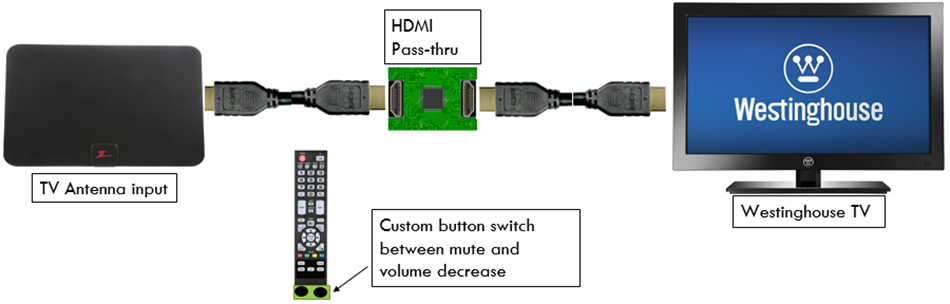
**9/22-9/29:**

* Finalized the customer requirements, Engineering Requirements, along with constraints and standards.
  + Finalized the continue of production using signal processing
* Original Flow chart: *see week 11/3-11/10 for functional chart and 4/19-4/26 for final flow chart*



**9/29-10/6:**

* Original Top-Level diagram: *see 11/17-11/24 for final top level diagram*



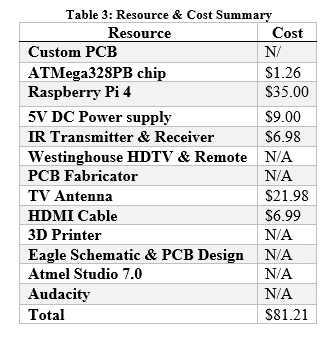
* Conducted research on statistics regarding on-air broadcasting:
  + Average break time: 92.63 seconds
  + Lowest break time: 12 seconds
  + Max break time: 157 seconds
  + Notes: There is an extreme amount of variability between the longest and shortest segments of television broadcasts.

**10/6-10/13:**

* Updated Engineering & Customer requirements, constraints, and cost summary
* Research conducted on signal processing:
  + All data from cable companies are encrypted and must be decrypted using their cable box
    - Therefore, all manipulation to those signals must be done after it outputs from the cable box.
  + Data speeds are very high, thus the processor chosen must be within these speeds
  + 3 Detection methods: Audio Levels, Black Frames, and Shotcuts

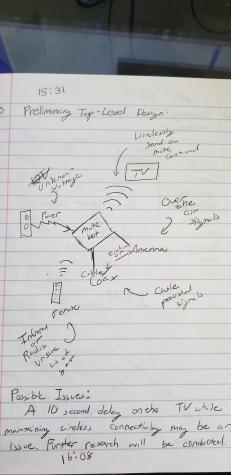
**10/13-10/20:**

* Began assessing different risks that go along with certain design choices. Started comparing certain parts, program languages, and project path.
* Customer requirement has changed. The system no longer must be preemptive as this was deemed too difficult.
  + The group has proposed to make it pseudo-preemptive, delaying the input to the TV by 10 seconds via a buffer. Therefore, it would appear preemptive to the user.
* Preliminary cost and resource table created:

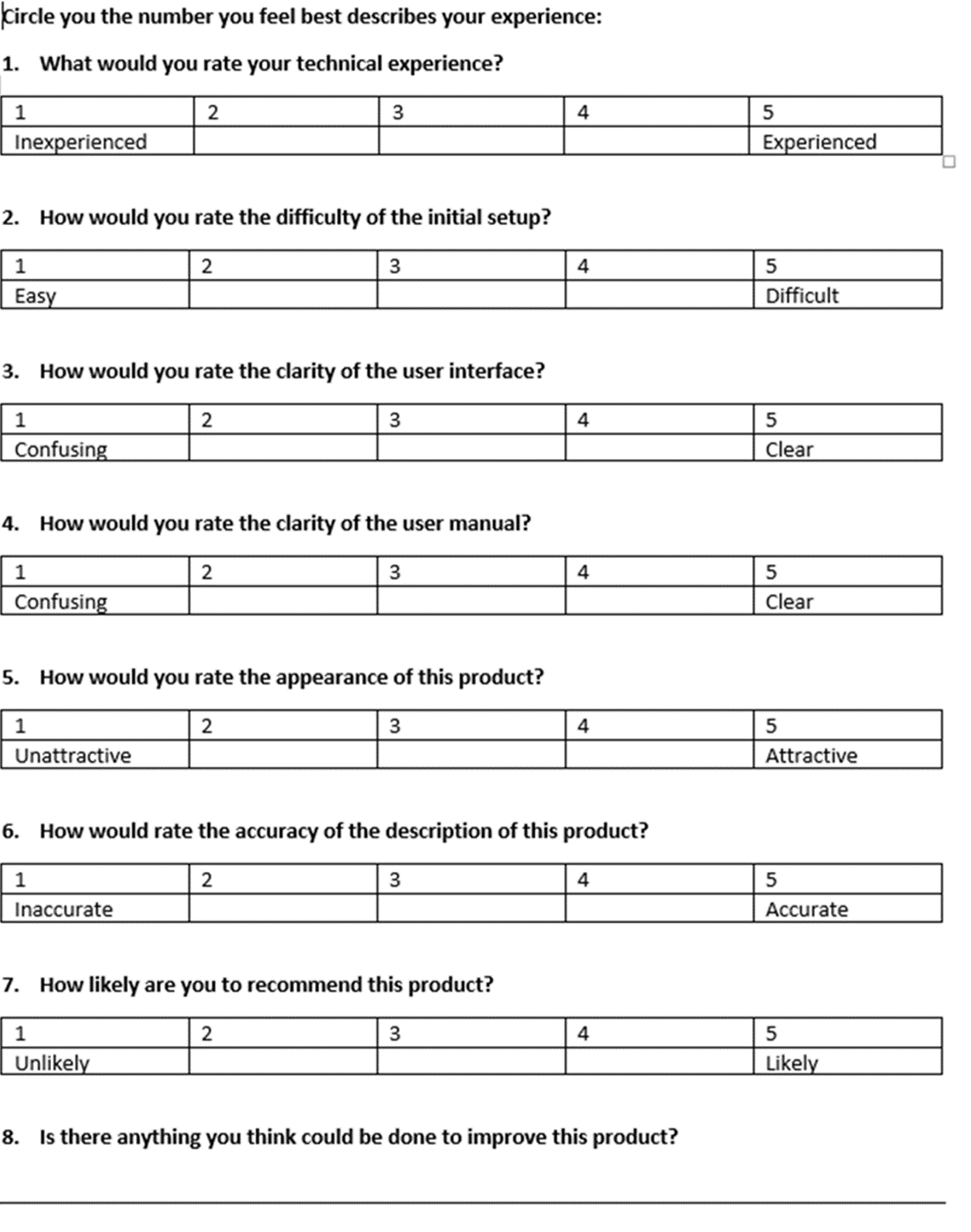


**10/20-10/27:**

* Researched deeper into the different parts that could be picked to assemble the project. Risks starting to be weighed against all options.
* Preliminary Top-Level Designs created:



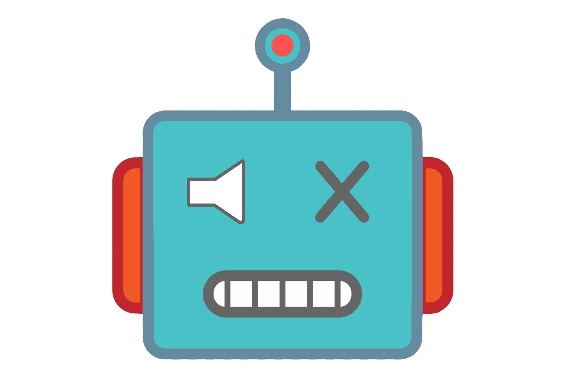
* Questionnaire created to ensure quality and ease of use of product upon completion:



**10/27-11/3:**

* IR Mitigation

|  |  |
| --- | --- |
| **Problem** | **Mitigation** |
| Problem initializing the Geekworm expansion board with Raspberry Pi | Increased hours of troubleshooting to communicate between IR expansion board and the Raspberry Pi with LIRC software |
| Issues with transmitting signals from Raspberry Pi to the Westinghouse TV | Prototype using the Westinghouse remote to create a custom configuration file |

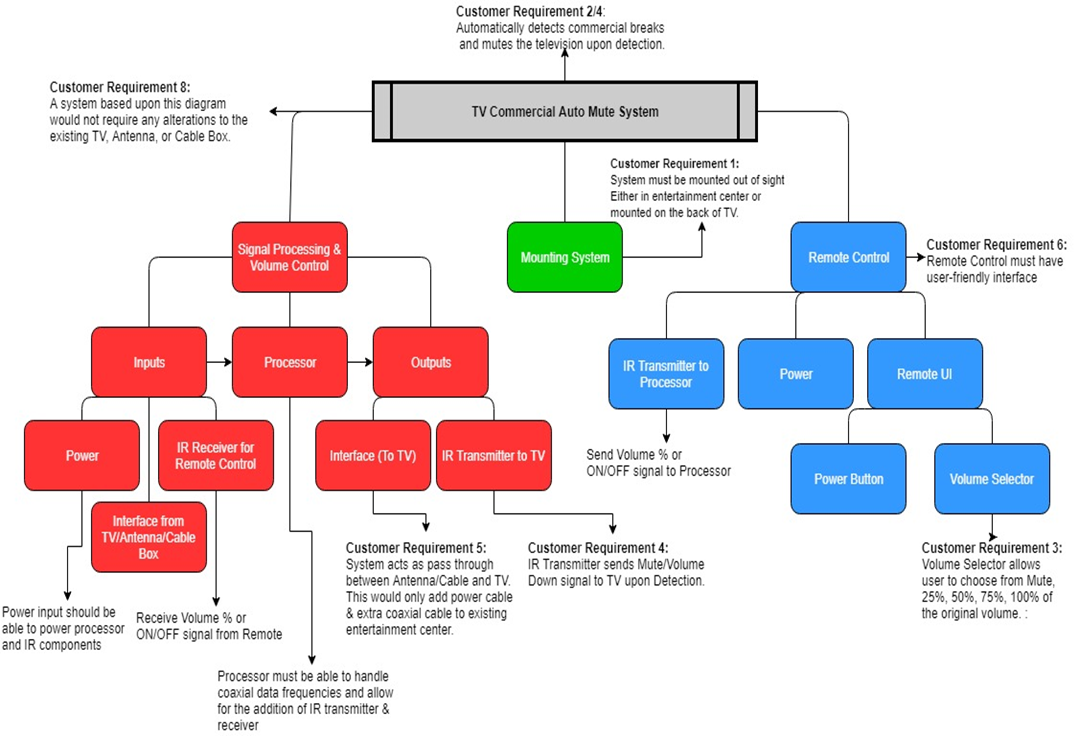
* Created Logo:

**11/3-11/10:**

* Finalized the CDR report and finished the presentation.

**11/10-11/17:**

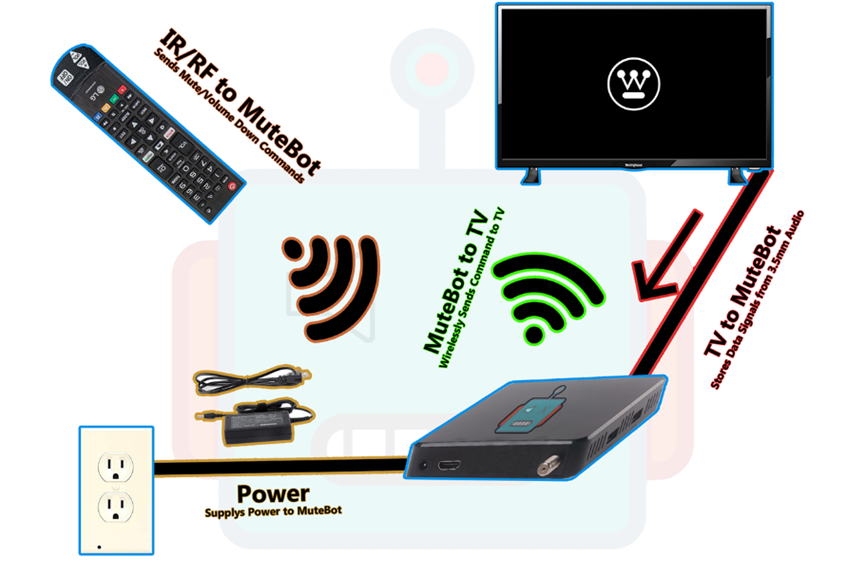
* Functional Flowchart Finalized:



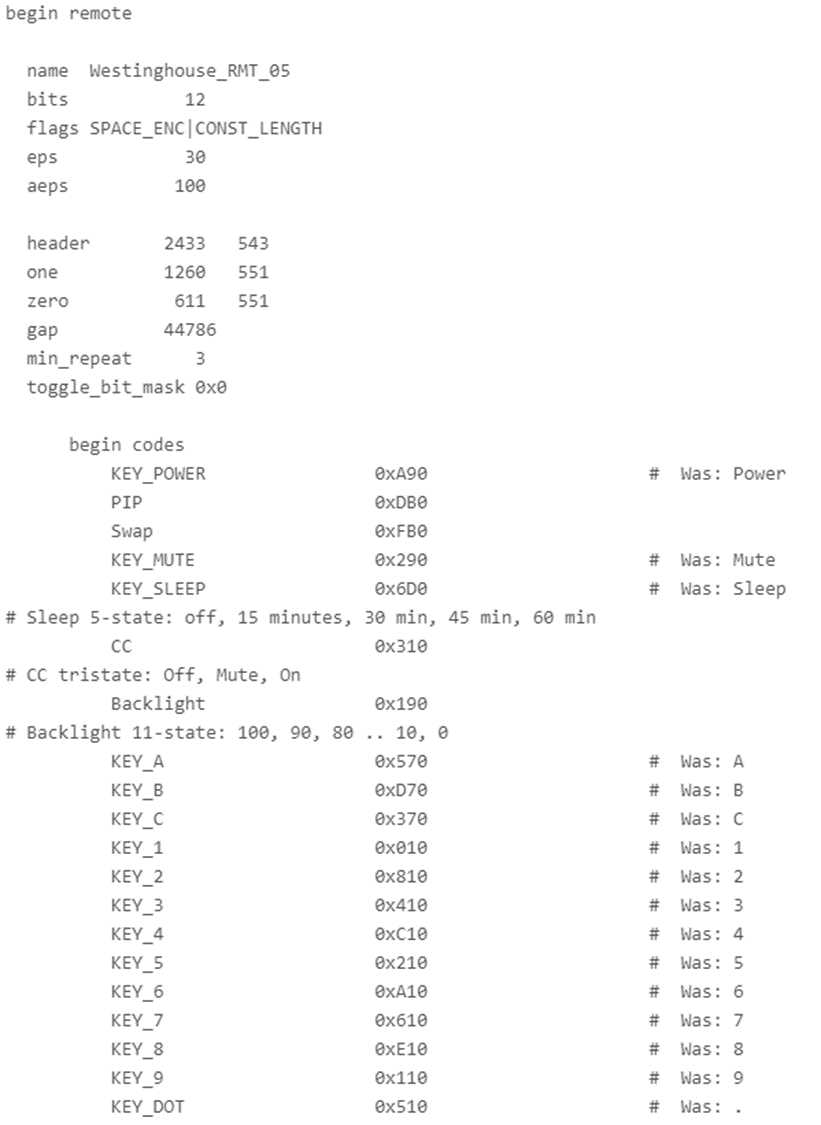
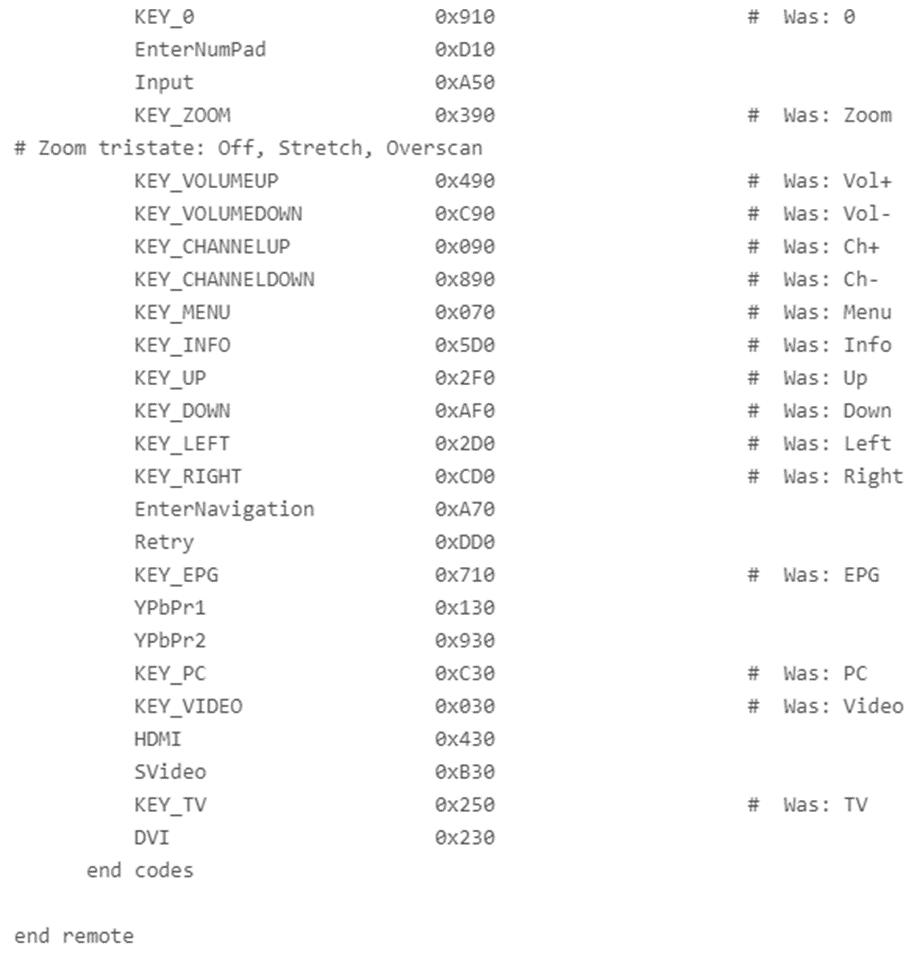
* Research conducted on coaxial cable decoding:
  + 3.58 Mhz = color
  + 4.5 Mhz = audio

**11/17-11/24:**

* Testing began on coaxial cable
  + Testing found that waveforms are not useful in this project
  + Dr. Benjamin Viall advised us to not use coaxial cables due to the high processing rates as a result of the many channels
  + Coaxial cable also did not provide enough information in regard to what channel a user was using, the volume, etc.
* Functional overview finalized:

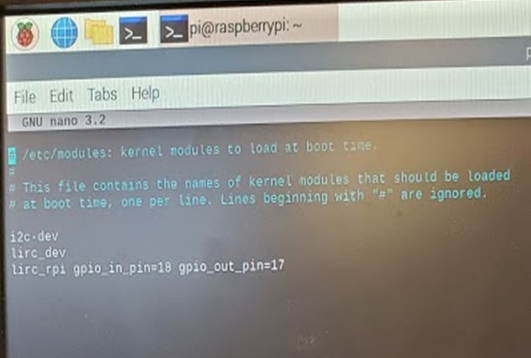


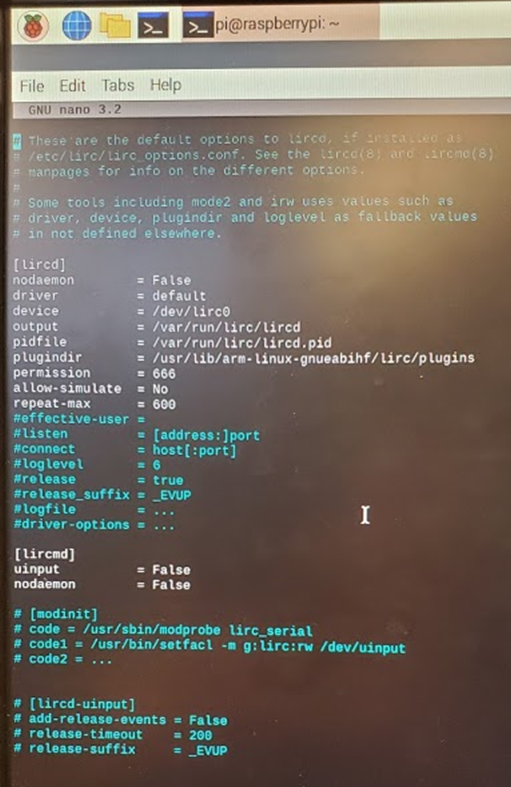
* Westinghouse remote configuration file (Used to find what hex values correlate with keys on the remote):

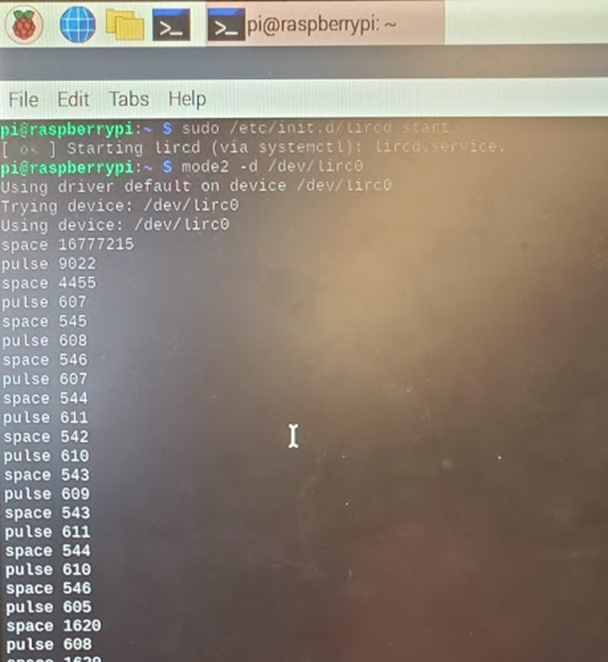


**11/24-12/1:**

* IR configuration file (used to set ports that connect to the Pi):



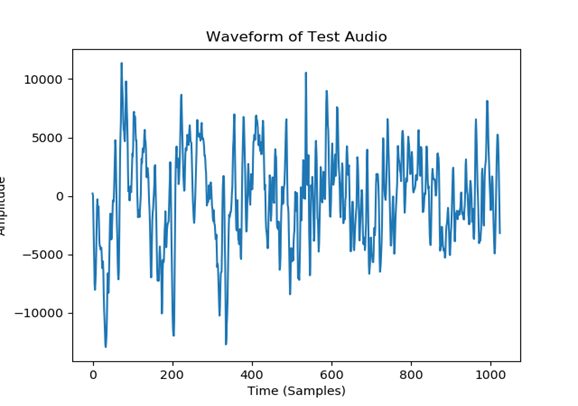
* 2nd IR configuration file (Lircd\_options, refer to 6.2 in report for further detail):
* LIRC commands and IR pulse (Testing done for IR Receiver):



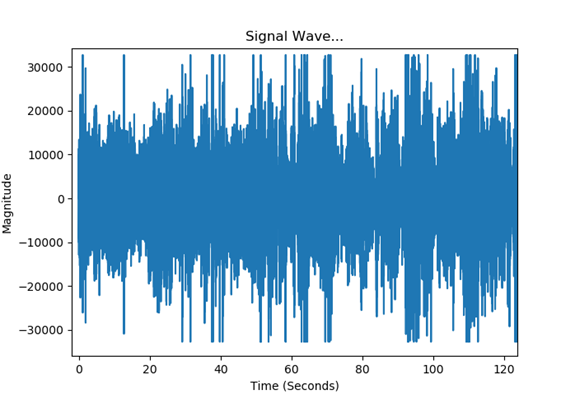
* All information regarding setup for the LIRC software can be found here: <https://gist.github.com/prasanthj/c15a5298eb682bde34961c322c95378b>

**12/1-12/8:**

* Worked on PDR report
* Python .wav audio file (First attempt at receiving real-time audio):



* .wav real time plot



**End of Fall semester**

*\*No work was conducted or planned for over the winter break\**

**Beginning of Spring Semester**

**1/19-1/26:**

* Started thinking of different tests that could have been run in order to test for all the requirements for the project.
* Read audio in through the 3.5 mm jack and plotted the audio via Matplotlib.
* Researched the LIRC library and researched how to initialize LIRC with the needed Westinghouse IR signal.
* Began fully developing the IDE:
  + Installed multiple libraries into the PyCharm (The Chosen IDE):
    - PyAudio
    - SciPy
    - NumPy
    - Math.lib
    - RPi.GPio

**1/26-2/2:**

* Started piecing together plans of how to run the tests, and still the process of making tests.
* Plotted and studied audio from the Westinghouse TV real-time
* Configured the LIRC to work on the Raspberry Pi

**2/2-2/9**

* Changed customer requirement from having a pseudo-preemptive to having a one second delay before the algorithm reacts.
* Changed customer requirement from having multiple options to have a 0% (on), 10 volume keys down, or 100% (off) when a commercial is detected
* Sent IR signal for volume up/down and mute from the Raspberry Pi to the TV.
* More Tests discussed and formulated.
* Updated LIRC to the most updated version. All information is available here on how to update LIRC: <https://gist.github.com/billpatrianakos/cb72e984d4730043fe79cbe5fc8f7941>

**2/9-2/16:**

* Worked on a program to detect for silent frames, standard deviation of average volume, and volume spikes.
* Wrote a program to lower the TV volume based upon a button press.
* Program written to toggle in and out of a detection loop.
* Configured new file that programmed the Westinghouse remote into the Pi.

**2/16-2/23:**

* Improved on program to detect any triggers of a commercial.
* Changed primary data type for sound bites to float32
* Learned different commands to use to send different IR signals a specified length of time

**2/23-3/1:**

* Worked on detection of commercials and triggers for the program
* Started to work on integration of the two parts of the project (IR & commercial detection program)
* Tested and optimized values for volume decrease percentages:

“irsend –count=\* SEND\_ONCE ESTINGHOUSE KEY\_VOLUMEDOWN”

* Count here refers to the amount of signals sent, this value needed to be tested to see what values correlate to the amount of volume clicks down from a remote. The table below shows what count values correlated to the number of clicks:

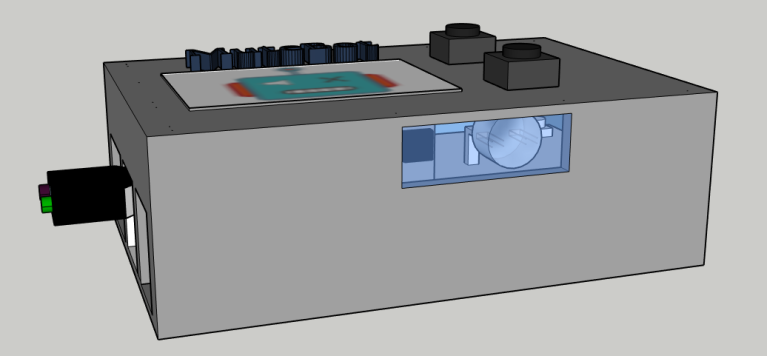
|  |  |  |
| --- | --- | --- |
| **Starting Volume** | **Count** | **End Volume** |
| 100% | 28 | 50% |
| 50% | 16 | 24% |
| 75% | 22 | 37% |
| 62% | 19 | 30% |
| 87% | 25 | 43% |
| 25% | 9 | 13% |
| 37% | 12 | 19% |
| 12% | 6 | 6% |
| 4% | 4 | 2% |

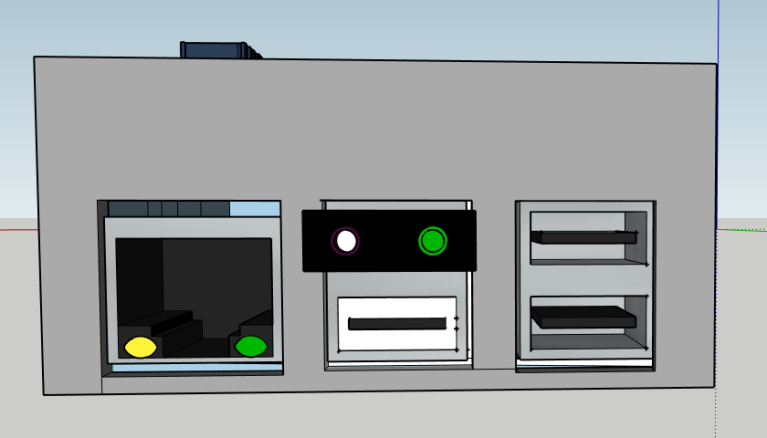
**3/1-3/8:**

* Finalized the group’s standings on different aspects of the project due to spring recess happening by week’s end
* Agreed upon plans for the project over the course of spring break

**3/8-3/14:**

* Created the algorithm that converts stream audio into a sound scale. Refer to section 6.2 for more information.
* Created a 3D printable model of the enclosure in SketchUp to be printed on the return from spring break.





**3/15-3/29:**

* Evaluated alternative options due to the closing of the labs and the inability to gain equipment.
* Did not do much this week as a result of the announcement of quarantine. Waiting to see if we can gain access to our equipment.

**3/29-4/5:**

* Equipment access was not allowed. Thus, work on a python program that establishes the same goal post-processing an .mp4 file has started.

**4/5-4/12:**

* Added in the first detection method, but it still has many false positives and misses some commercials. Parameters need to be adjusted and tested.

**4/12-4/19:**

* Program flow chart created. Refer to 6.4 for more information regarding the program and program flow.
* Testing sequence created for program.

**4/19-4/26:**

* Final Report outlined and work divided among group

**4/26-4/29:**

* Final Report finished
* Github Repository made
* Program finalized
* Presentation Conducted

End of spring semester

ALL CODE, CONFIG FILES, DIAGRAMS, SCHEMATICS AND REPORTS ARE IN THE GITHUB REPOSITORY AS FOLLOWS: <https://github.com/ztaylor28/ECE458Documentation>