UC SANTA CRUZ

# TRAVIS

(**T**elevision **R**emote **A**nd **V**oice **I**nterpretation **S**ystem)

Simplified Remote Control with Voice Recognition

Ann Sophie Abrahamsson, Nathan Banner, Lillian Gwendolyn, Katy Johnson,
Aidan Martens, Heath Robinson, Kanybek Tashtankulov
Group 9

# Executive Summary

The purpose of the TRAVIS Project is to build a simple, universal, voice-activated television remote control. Modern-day TV controllers have many buttons that are tiny and hard to press for people with limited mobility or sight. The TRAVIS Project aims to solve this problem by producing a voice-enabled TV remote with the bare minimum number of large, easy-to-press buttons. This goal is illustrated with the acronym, TRAVIS, which stands for Television Remote and Voice Interpretation System.

TRAVIS is intended to help out the elderly, the disabled, and any family or caretakers of theirs. This product is designed to be a quick solution for several problems related to browsing modern entertainment devices, including incorrect button inputs and difficulty with seeing the remote. TRAVIS will be a low-cost product that is simple to set up, long-lasting, and, most importantly, accessible because of its reliable voice-control accuracy - with TRAVIS, instead of searching for specific buttons you can just ask "Hey TRAVIS, can you play Wheel of Fortune" or "Hey TRAVIS, where did I leave the remote?"

This document includes TRAVIS' features, hardware, aesthetics, processes, design flows, and the results garnered from our prototype build and its subsequent testing. TRAVIS is currently built to run on a Raspberry Pi Zero 2 W Microcontroller, housed in a sleek standardized design with room for eight buttons, each larger than a quarter in diameter. TRAVIS has a battery door at the back to swap out batteries, and two perforated honeycombs on the top to facilitate audio input/output, organized like that of a phone. And to ensure the user's privacy, TRAVIS has an LED that indicates whether or not it is currently listening, which can be toggled off and on from the TRAVIS button near the upper-middle of the remote.

TRAVIS implements a variety of Python solutions that transfer user audio input through powerful speech recognition libraries, leaving us with text that we can read to determine the user's command, which is then sent out via InfraRed while the user is alerted of the command's success from the soothing voice of TRAVIS themself. Included in this are several powerful user-oriented features, allowing TRAVIS to have a quick setup time that only requires the manufacturer of the customer's television, or their old remote on hand to create a custom-tailored configuration. The remote also has a setting that allows the user to have TRAVIS automatically stop listening to them after a duration without any commands heard. A Find-My-TRAVIS feature is also included, which allows the user to locate their device without scrambling around their room.

The TRAVIS prototype showed that our design is as functional as our unit tests had concluded. Voice commands are successfully interpreted and processed, all of its buttons send IR signals with a long-range, and its physical shell is comfortable to hold with an intuitive button layout. While we learned in the process that TRAVIS' hearing is very hardware dependent, no feature discussed seemed impossible to implement.

With discussed changes to TRAVIS' speech processing to use Mozilla's DeepSpeech, the future of this project is all turning up TRAVIS!

# **Table of Contents**:

# Introduction

## Need Statement:

Elderly people have a hard time navigating modern entertainment devices.

## Goal Statement:

Make it easier for elderly people to navigate modern entertainment devices.
The product will be at least 50% recyclable

# Ethics Statement

We, in the fulfillment of our professional duties, shall:

1. Hold paramount the safety, health, and welfare of the public.
2. Perform services only in areas of our competence.
3. Issue public statements only in an objective and truthful manner.
4. Act for each employer or client as faithful agents or trustees.
5. Avoid deceptive acts.
6. Conduct ourselves honorably, responsibly, ethically, and lawfully to enhance the honor, reputation, and usefulness of the profession.
7. Create products that are sustainable and do not negatively impact the environment.
8. Maintain the privacy of our customers and will not use any voice recordings for malicious purposes, nor share or sell it to third parties.

Source: https://www.nspe.org/resources/ethics/code-ethics for 1-6.

# Persona 1:

Name: Edith

https://depositphotos.com/vector-images/old-lady-cartoon.html
Role: Elderly person with visual impairment
Background:

- 80-year-old woman who lives alone, has one daughter who visits twice a week to check on her
- She has macular degeneration and is therefore visually impaired
- Watches television every day but has trouble navigating channels because she isn't able to see the buttons on the remote.
- Calls daughter whenever she has technical problems

Goals/Behaviors:

- Wants to operate remote without needing assistance

# Persona 2:

Name: Susan

https://www.istockphoto.com/illustrations/older-business-woman

Role: Part-time caretaker of her elderly mother

Background:

- Works full-time at a business firm
- Visits mother twice a week when she has time, makes sure she is taking medication and fixes television problems

Goals/Behaviors:

- Wants mother to be able to use television independently
- Reduce the amount of times mother calls her during her busy day to fix television problems

# Persona 3:

Name: George

https://stardewvalleywiki.com/George
Role: Elderly person with a movement disability
Background:

- Wheelchair-bound after workplace injury
- Doesn’t like relying on others’ help
- Likes watching TV

Goals/Behaviors:

- Wants to watch TV without needing assistance
- Wants to be able to set up the device without help

# Existing Product Research:

Several similar products integrate individual features that TRAVIS is planned to include, but nothing that has them all together. The three main existing products are:

- Amazon's Fire TV Cube, which implements all of the voice applicable features on a stationery box with a somewhat simplified remote control.
    - This design, while very good on the voice side of things, requires a large amount of setup (requires a direct link to the TV and a local internet connection), and has a remote that can be challenging to use and understand.
- Logitech's Harmony Elite, which is a now-discontinued product line that acts as a smart home controller on a stationary box, in addition to a TV remote control and an optional phone app.
    - This design is very powerful and has Alexa voice integration, but is very complicated and challenging to use and set up the device, more focused on tech-enthused consumers that can afford the high price.
- Philips Universal Remote Control, which is a cheap universal remote control with a phone app that enables you to find it wherever it might be, in addition to acting as a smart home controller.
    - This design has no voice integration and a complicated remote but has a potentially simple setup through its phone app, and a feature that lets you find where it is.

Beyond these, there are several simplified universal remotes, many of which have a minimum number of rather large buttons, but all suffer from the issue of a complex setup. In short, there are no existing designs or products that meet the needs of our personas, and thus TRAVIS is a solution that has not otherwise been created.

# Design

## Objectives in Words:

Design a remote with a short setup time and accurate voice command response that quickly and accurately changes channels for the user. The remote should be inexpensive to manufacture, have a long battery life, moderate charge time, and large buttons for ease of use.

# Free Format Description of Design

TRAVIS is intended to provide the user with a simple, easy-to-use TV remote that delegates most of the more complicated functions to voice-activated features. The design we settled on aims to include the following main features:

- Large, Simplified Button Layout
- Voice Integrated Capabilities
    - Status LED to indicate if TRAVIS is listening, auto turn-off setting available
- Simple Set-Up
    - Either run through every model from a manufacturer or save signals from current remote
- Find-My-Travis
- Rechargeable or Replaceable Batteries

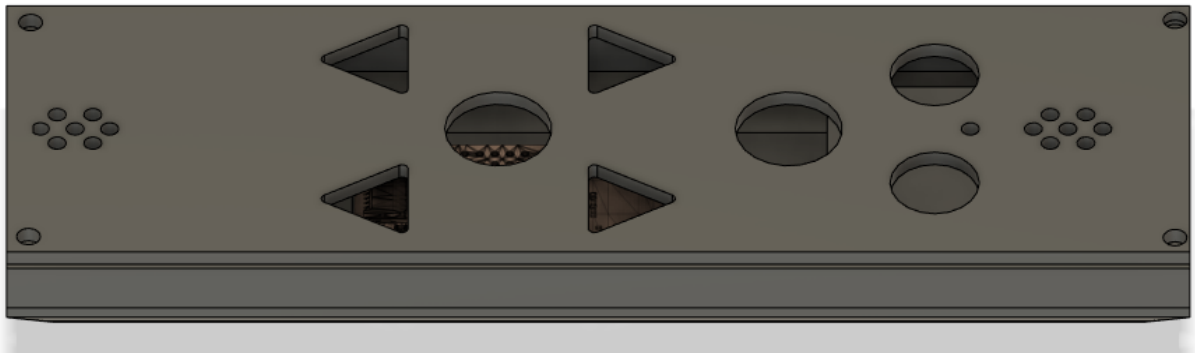
Piece by piece, TRAVIS was designed to maximize ease of use by exploring the capabilities brought on by voice-integrated capabilities. As such, since TRAVIS is capable of handling the most abstract functions of a TV remote via speech, it can use a limited button layout without sacrificing any functionality. Similarly, to aid our intended consumers in its use, all of the buttons are intended to be far larger than that of a standard television remote.

Similarly, as mentioned, TRAVIS is capable of handling even the most esoteric remote functions via speech access, from set-up to changing the channel directly to your favorite shows. Speech access can be enabled or disabled by pressing the TRAVIS button on the remote. To allay our customer's fears of their privacy being infringed upon, TRAVIS has an LED light indicating its current status - if the light is on, TRAVIS is listening, and if off then TRAVIS is deafened. Similarly, TRAVIS has a setting to enable or disable it automatically deafening after a duration of no commands heard.

As a side effect of this, TRAVIS can leverage its speech recognition capabilities to facilitate a speedy and efficient easy-to-use process - instead of reading through an instruction booklet for a specific television model's code to press into the remote, TRAVIS lets you name your television manufacturer (or make as well, if known) to then run through a list of possible remote configurations until the user finds one that works. Similarly, TRAVIS is capable of doing a custom configuration mode, involving the user pointing their old remote and TRAVIS at one another and pressing each button TRAVIS asks for, directly copying signals.

Additionally, because TRAVIS can listen to its users, it can also have a simple Find-My-TRAVIS feature, letting the consumer ask TRAVIS where it is to have it respond, helping users find their misplaced remote.

Finally, TRAVIS will be implemented with either rechargeable or replaceable batteries depending on the model.



The outside shell of TRAVIS is a simple 3D-printed model composed of 3 sections. The main body is a rectangular shape with slanted edges for ease of holding. It is meant to emulate the design of remotes currently on the market and be recognizable by consumers as a remote. One small feature on the body of the remote is a hole near the front where an LED would fit through to indicate when a button was pressed. There is also a rectangular cutout at the bottom of the remote where the battery door fits in. This battery door is removable and allows the user to replace the 9V battery. Our prototype did not make use of a battery, however, and was instead directly plugged into a power source. Wireless charging with a lithium-ion battery is an alternative power source we considered but would result in a more expensive model. We could potentially have two versions of the remote, one with a 9V battery and the other with wireless chargings. This current 3D model is designed for a 9V battery. The lid has large cut-outs where the buttons would be placed in a manufactured design. Each button would have an attachment that fits the shape of its respective hole. Two sets of holes are located on the top and bottom of the remote to allow microphone input and speaker output. Below the top set of holes is a hole for an LED to show the user when TRAVIS is listening for their voice. The bottom of the lid has two chambers that fit the speaker and microphone in place below the holes. In addition, holes are placed at the corners for screws to go through and keep the lid in place. Finally, a similar set of screws can be placed in the shell on a raised platform to secure the Raspberry Pi Zero 2 W to the inside of the shell.

# Design for Manufacture and Assembly

The external design for our remote is composed of two pieces (a top and bottom) that screw together to form the main remote shell, and a battery door to keep the battery in place. Our design includes 13 buttons:

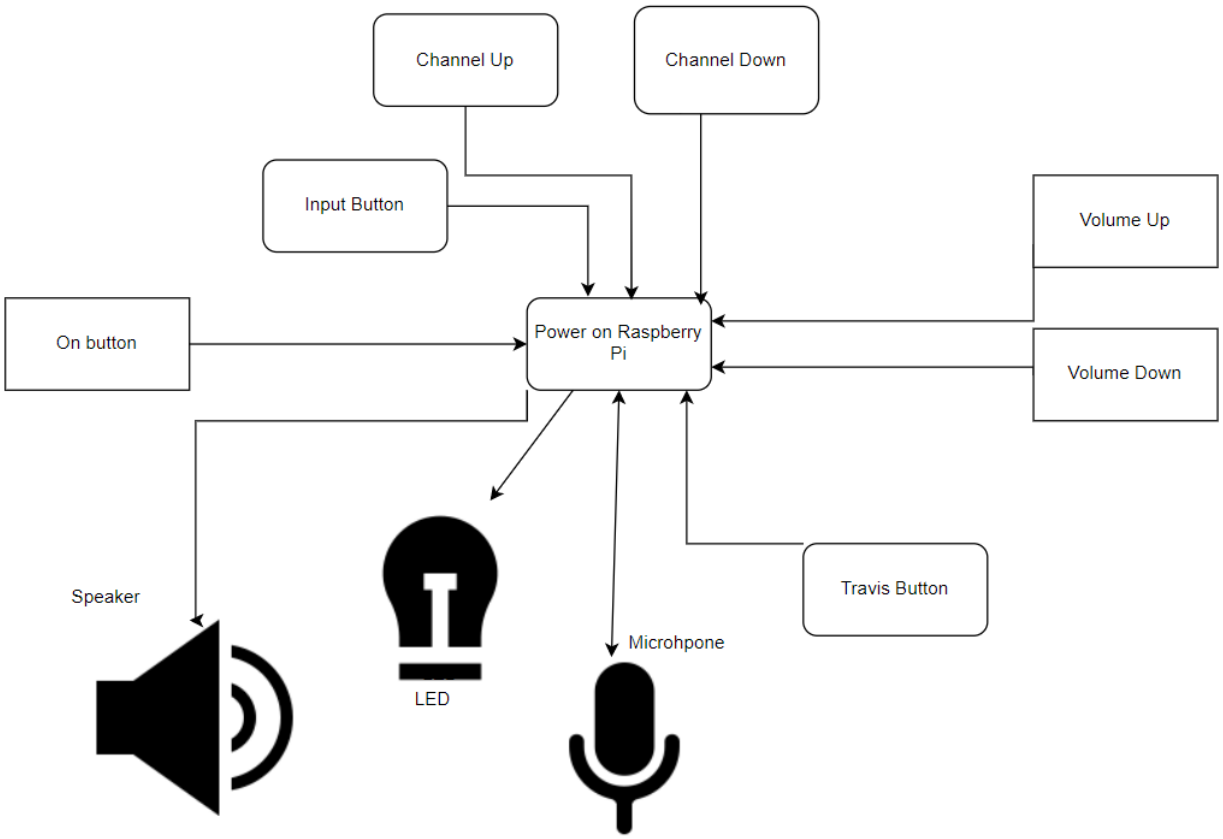
- TV On/Off
- Remote Voice Input Activated/Deactivated
- Cursor Up, Down, Left, and Right
- Select
- Channel Up and Down
- Volume Up and Down
- Source
- Mute

To keep the design simple, it is all 3D printed with plastic. Assembly is also kept simple only needing about 10 small screws to keep all components in place. In the section for “Free Format Description of Design,” the parts are described in greater detail. The largest time sink for assembly is the breadboard itself. The wiring can be time-consuming to finish and requires good familiarity with the device. Once this is learned, however, the assembly can be expedited. The small amount of buttons also leads to shorter and more efficient assembly.

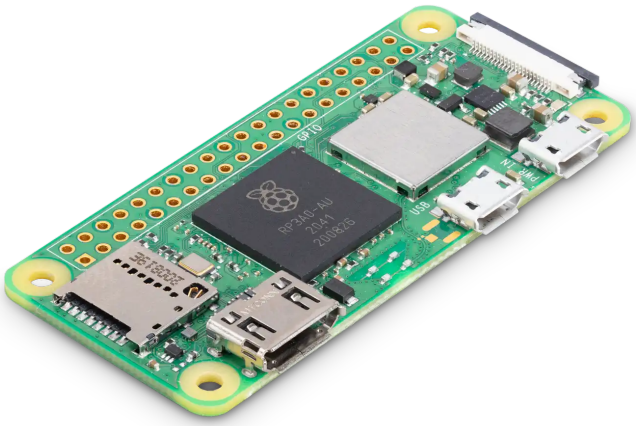
In this design, we tried to keep the number of components as minimal as possible. This is for the sake of the target user base of the elderly and disabled who might not want or understand a large range of functionalities. As a result, only the microphone, buttons, LED lights, speaker, and Raspberry Pi Zero 2 W need to fit into the chamber. There is plenty of space in the chamber for wires to fit inside and a slightly wider design could also accommodate the breadboard used in the prototype design.

Maintenance of the device is very simple. The user needs to put in a new battery whenever the device stops working. The remote should be kept in a dry, cool area to prevent damage to electronic components. The device should likewise not get wet. Finally avoid dropping the device, throwing it, or concussing it in any manner.

## Block Diagram & Technology




Essentially, everything runs through our Raspberry Pi Zero 2 W Microcontroller, with everything handled from general-purpose input/output pins powered by our software.



(Image Source: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/)

On the hardware side of things, the Raspberry Pi Zero 2 W was chosen as a powerful microcontroller with a small form factor, saving us both energy and space. In a more rigorous

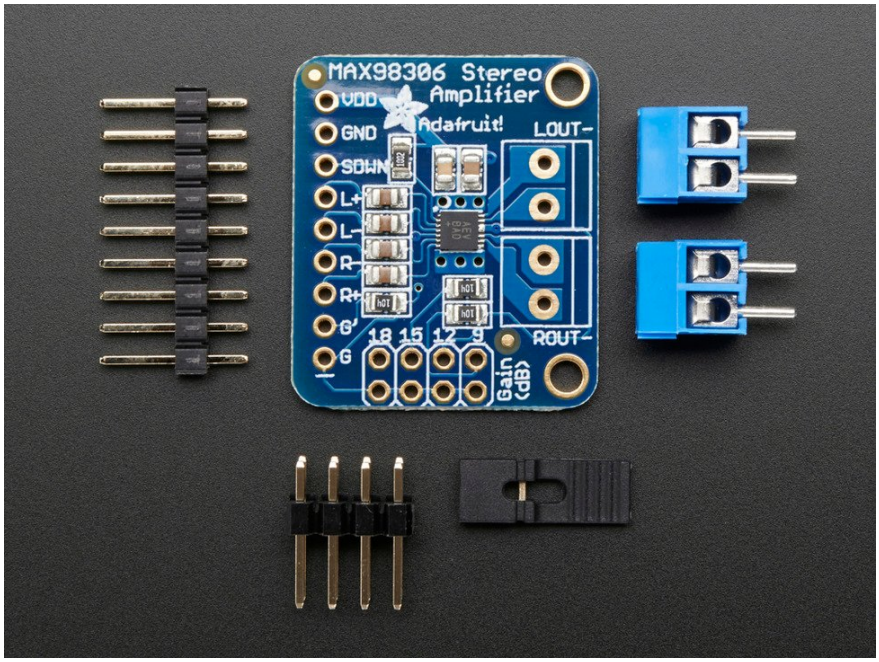
design, this is likely to be replaced with a custom microcontroller, but changing it would involve changing a lot of code and hardware integration.

(Image Source: https://www.adafruit.com/product/1119)

The buttons used thus far are some simple switches which, in a final design, will be covered with proper caps to fill out their size on our front panel, as well as feature the names of each button.

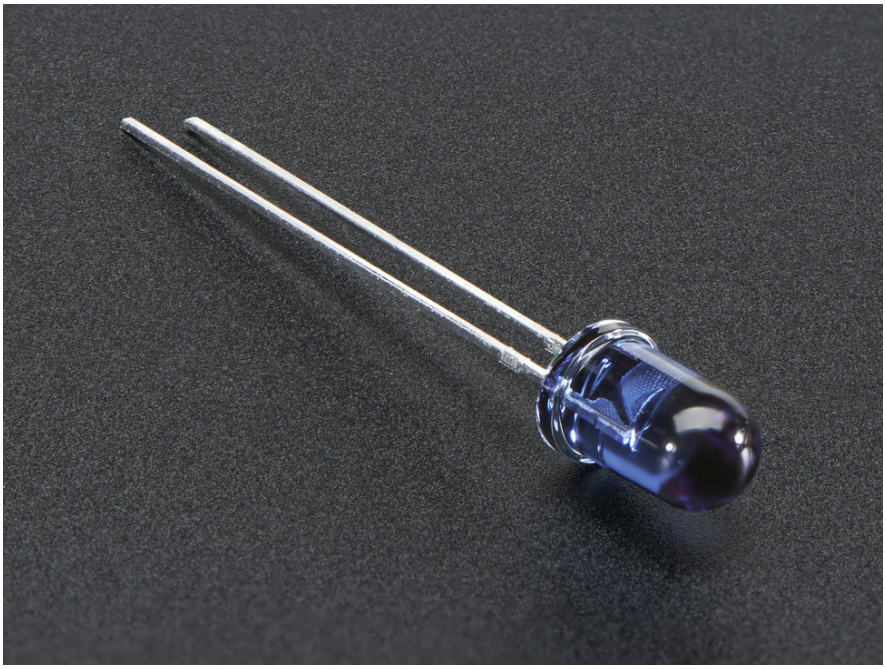
(Image Source: https://www.adafruit.com/product/1890)

Our speaker was chosen to be as simple as possible to minimize costs, energy usage, and space usage, and plugged into our audio amplifier to ensure the volume was at a reasonable level.




(Image Source: https://www.adafruit.com/product/987)

We used the MAX98306 audio amplifier to ensure our speaker speaks at an audible, variable volume. This amplifier was perhaps a bit overkill for our purposes, but it gave us some room to expand and test if we wanted to change various settings.



(Image Source: https://www.adafruit.com/product/387)

For our InfraRed output used to interface with television IR receivers, we used a simple IR LED hooked up to a transistor to amplify its power, providing us with a very large cone in which TRAVIS could successfully transmit an IR signal.

(Image Source: https://www.amazon.com/dp/B01MQ2AA0X)

As a quick fix, we used a Youmi USB microphone to handle our prototype's audio input, as we did not get enough time to try out our actual intended microphone chip. This USB microphone is slotted into a port provided by our microcontroller, providing a poor-quality but workable audio input stream.

Now onto software - because we used a Raspberry Pi microcontroller, we decided to write all of our code in Python to match with the language used in a lot of Raspberry Pi applications (such as RPI.GPIO, used to control the microcontroller's GPIO pins via software).

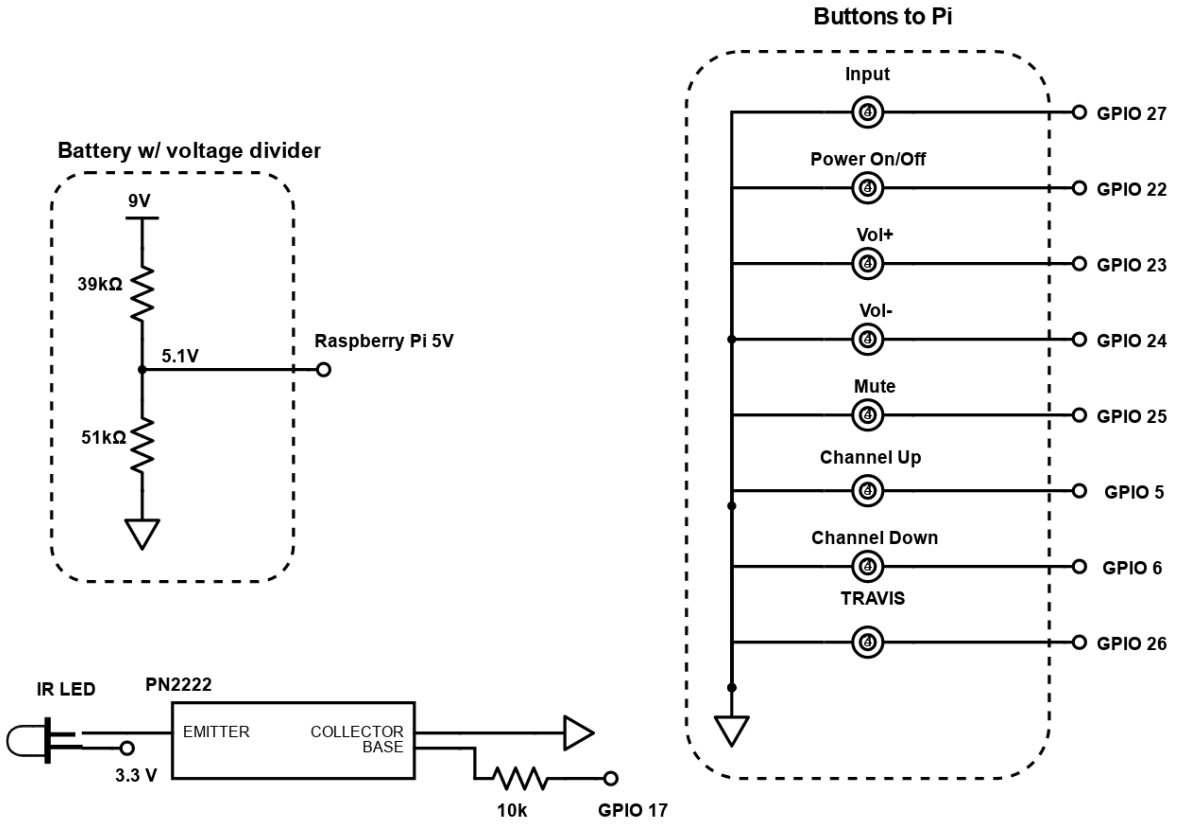
Similarly, our text to speech sent to our speaker was handled by the Python PyTTSx3 library, and our speech to text processing from our microphone was handled by the Python SpeechRecognition library, which implements several speech recognition softwares - of the six it implements, we use Google's speech recognition when TRAVIS has a wireless connection and CMUSphinx's speech recognition when TRAVIS is offline. A future version of TRAVIS may implement Mozilla's DeepSpeech software to train to the customer's voice in real-time, although TRAVIS' current prototype does not currently do so.

Beyond this, once the user's speech has been turned into text TRAVIS runs it through a giant RegEx block, checking it against a list of commands before signaling the rest of the code to try the specified command.

As TRAVIS is a television remote, it needs some way of signaling the television with various commands from its software - this is handled by the Linux Infrared Remote Control library (LIRC), which sends commands specified by the software off to a daemon that matches it up with a given remote style before shooting the signal out from its set GPIO ports.

And lastly, all of this is set up to run from our Dockerfile, which lets manufacturers simply run the Docker to generate a clean, working installation of TRAVIS onto their microcontrollers, as well as letting us test on fresh installations.

# Wiring Diagram

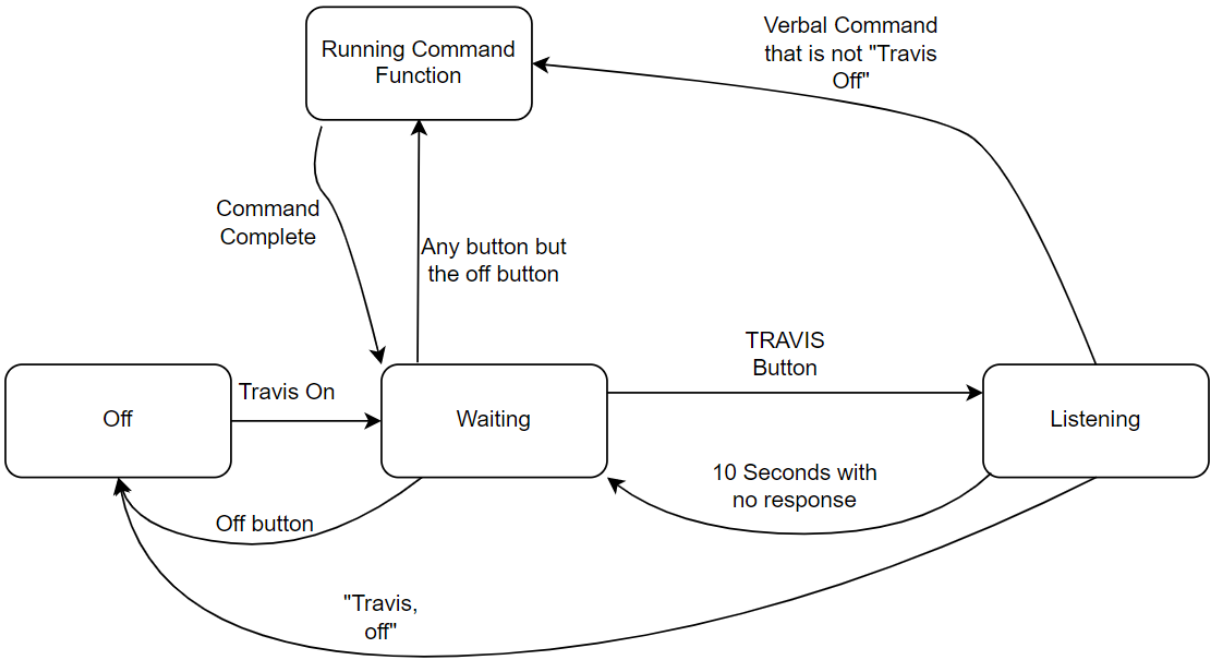



Shown is the hardware wiring diagram for the remote control. The far-left circuit is a voltage divider used to step down the 9V source voltage from the battery to a 5.1V source that powers the Raspberry Pi Zero 2 W. Ground in the schematic is connected to one of the ground pins on the microcontroller. The bottom left circuit shows how the Raspberry Pi sends the IR signals to the television

The IR LED signal is output in PWM format from the GPIO 17 pin through a resistor into the base of the PN2222 transistor. The cathode is connected to the emitter of the transistor, and the reason for this is to increase the range of the IR signal being output from the LED. Power is supplied to the anode of the LED by the 3.3V pin of the Raspberry Pi, and likewise for the collector node of the transistor.

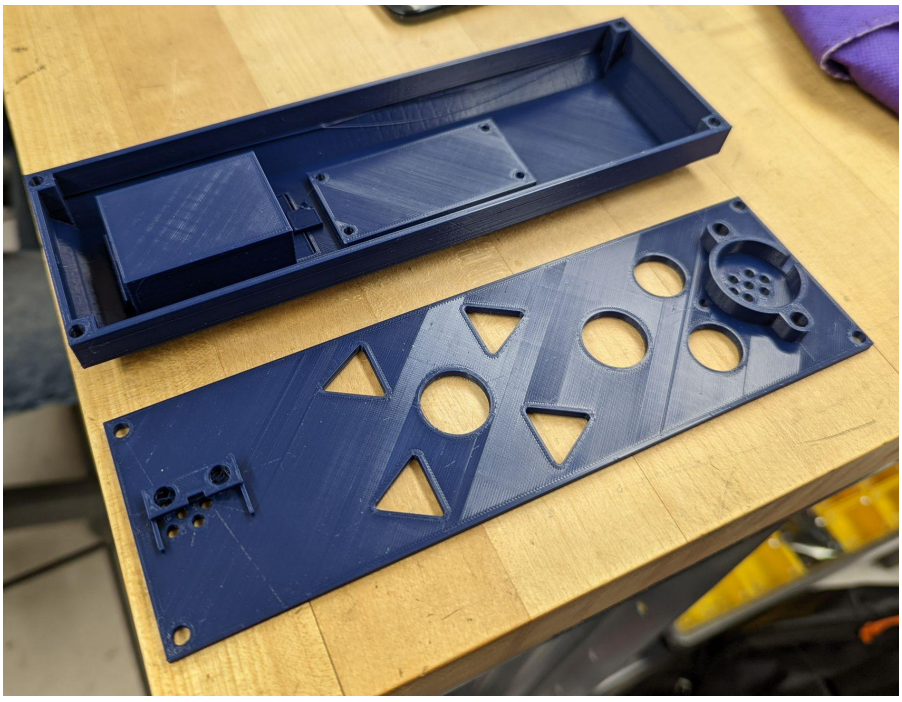
The far-right circuit is the configuration used to communicate button presses to the Raspberry Pi. The buttons are active low and the GPIO pins they are connected to are configured as INPUT_PULLUP by default. This means that the GPIO pin states are high until their corresponding buttons are pressed, which pulls their state low. The code will react when the button reading is low.

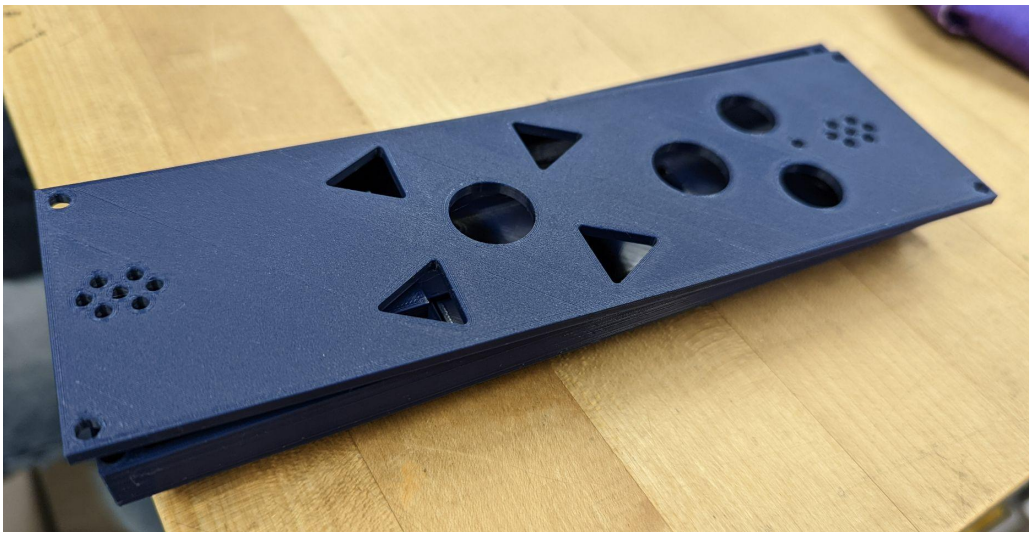
# State Transition Diagram




Initially TRAVIS is in the off state. Once it is turned on it waits for any button to be pressed. If any of the physical buttons such as volume/channel up and down buttons are pressed, the command is processed, and once complete TRAVIS goes back to waiting. If while waiting, the TRAVIS button is pressed, it starts listening for a voice command. If a command is given it processes the command again and goes back to waiting for a button to be pressed. If no command was spoken or recognized within 10 seconds of hitting the TRAVIS button, it stops listening and goes back to waiting. TRAVIS can also be turned off at any moment by either pressing the off button on the remote or saying ‘TRAVIS off” while it is listening for a voice command.

# Assembly Instructions



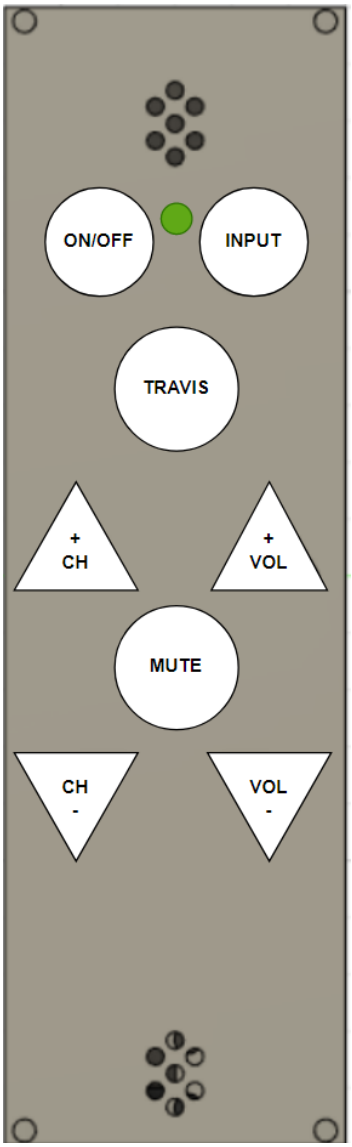


Travis assembly comes in three parts. The top, or the lid, of the device, the main body, and the battery door. The main body is the largest rectangular piece with slanted sides towards its bottom. First, attach buttons to the lid portion of the device. These will be soldered in place. Wires will attach to the buttons from below to connect to the Raspberry Pi Zero 2 W. Next, the microprocessor will be placed above the rectangular raised portion of plastic in the middle of the main body. Align the holes in it with the holes in the plastic. Screw-in four M3 x 5mm screws into the holes until held firmly in place. Next, take the microphone and place it into the rectangular cubby at the bottom of the lid. Align the holes on it with those in the plastic and screw in two more screws to hold it still. Next, place the speaker in the circular chamber. Wire

the speaker to the Raspberry Pi Zero 2 W and cover the chamber with the chamber lid. Use two screws to keep it in place. Take both of the LEDs and place them into the holes located in the picture below. Solder them to the inside of the main body and wire them to the raspberry pi. Lastly, attach the lid over the top of the main body, lining up the four holes in the corner. Attach with four final screws. The battery door will clip into the bottom of the main body using a plastic clip.

# Button Layout and Available Voice Commands

- Button Layout
    a. Power ON/OFF
    b. Input
    c. Volume UP
    d. Volume DOWN
    e. Mute
    f. Channel UP
    g. Channel DOWN
    h. TRAVIS
- Available Voice Commands
    a. "*TRAVIS, initialize*"
        - Result: establishes the connection between TV and remote
    b. "*TRAVIS, change channel to (CHANNEL NAME OR NUMBER)*"
        - Result: changes channel using either channel name or number
    c. "*TRAVIS, power ON*" or "*TRAVIS, television ON*"
        - Result: turns the TV on
    d. "*TRAVIS, power OFF" or "TRAVIS, television OFF*"
        - Result: turns the TV off
    e. "*TRAVIS, mute*"
        - Result: mutes the TV's sound
    f. "*TRAVIS, RAISE volume" or "TRAVIS, volume UP*"
        - Result: turns the volume of the TV up by 1
    g. "*TRAVIS, LOWER volume*" or "*TRAVIS, volume DOWN*"
        - Result: turns the volume of the TV down by 1
    h. "*TRAVIS, channel UP*"
        - Result: increases the channel number by 1
    i. "*TRAVIS, channel DOWN*"
        - Result: decreases the channel number by 1
    j. "*TRAVIS, assign (NEW NAME) to (CHANNEL NUMBER)*"
        - Result: Assigns a name to the given channel number



# Setup Instructions

If taking TRAVIS out of the box, setup is accessed immediately upon providing TRAVIS some batteries (or charging the one currently inside of it) and turning TRAVIS on for the first time.

If the user is instead setting up after TRAVIS has been previously set,

1. Press the “TRAVIS” button on the remote
2. After the green LED at the top of the remote turns on, say “TRAVIS, initialize”

From there, TRAVIS will prompt the user to either say their television’s manufacturer or say “TRAVIS, initialize custom”

Either way, this will take TRAVIS into its setup mode. If the user says their manufacturer, i.e. Samsung, TRAVIS will try various buttons automatically, telling the user expected changes.

- If the user says each action was successful, TRAVIS will let the user know that setup has successfully finished and save the remote style.

- If the user says an action was unsuccessful, TRAVIS will instead restart the testing process with another remote style of the same brand, repeating until it gets through setup successfully.

If no style is successful or the user tells TRAVIS to use a custom style, TRAVIS will prompt the user to push their current TV remote and TRAVIS together such that their infrared transceivers are pointed at one another. Once the user confirms this, TRAVIS will ask the user to try a single button, e.g. prompting the user to press the power on/off button, saving the result as it goes. Once the user has entered every button on their remote (or told TRAVIS their remote doesn’t have such a button), TRAVIS will let the user know that setup has successfully finished and save the custom remote style.

# Budget for Prototype

| Final Parts List | | | | | | | |
|---|---|---|---|---|---|---|---|
| Name | Link | Amount | Cost (Each) | Tax + Shipping | Cost (Total) | Availiabilty/Lead Time | Reasoning |
| Raspberry Pi Zero 2 W | https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/ | 1 | $15.00 | | $15.00 | ~2 week lead time<br>out of stock, on notification list | Microcontroller is necessary to be able to program on and provide a hub for associated embedded devices; $50-70 on eBay(scum)<br>This specific RP provides good power for the price and is small enough to comfortably fit within a remote controller. Katy is on list for: pishop.us, sparkfun |
| Tactile Switch Buttons (12mm square, 6mm tall) x 10 pack | https://www.adafruit.com/product/1119 | 1 | $2.50 | | $2.50 | | Remote buttons |
| Mini Metal Speaker w/ Wires - 8 ohm 0.5W | https://www.adafruit.com/product/1890 | 1 | $1.95 | | $1.95 | | Volume output so the device can confirm commands |
| Stereo 3.7W Class D Audio Amplifier | https://www.adafruit.com/product/987 | 1 | $8.95 | | $8.95 | | Other option for speaker |
| Super-bright 5mm IR LED - 940nm | https://www.adafruit.com/product/387 | 1 | $0.75 | $0.35 | $1.10 | | Light indicator for when button is pressed to show it is working |
| Mini USB 2.0 Microphone Mic for Laptop/Desktop PCS - Skype/Voice Recognition Software Driver-Free Audio Receiver Adapter for MSN PC Notebook | https://www.amazon.com | 1 | $7.99 | | $7.99 | | Used as backup microphone |
| TOTAL | | | $37.14 | | $37.49 | | |

Our budget includes a Raspberry Pi Zero 2 W microprocessor, a microphone for voice input, LEDs for light indicators, buttons for manual remote input and activating TRAVIS, and a speaker & Class D Audio Amplifier for potential vocal feedback from the remote. We included wireless charging components in the initial budget, but after reviewing the costs we realized that the expenses involving wireless charging doubled the cost of our prototype. We decided to use regular batteries for our project and implement the wireless charging features later if we have time to decide if it is worth the extra expense. Our final budget for a single remote, as seen above, summed up to $37.49. This is a little expensive given we wanted it to cost about $10 but the Raspberry Pi alone was already $15 making this inapt. Another factor in the price is the Audio Amplifier at $8.95 and the Microphone at $7.99. In the future, less expensive parts could be sought after and we could get our own microprocessor manufactured.

# Life Cycle Assessment

## Energy Use

- **Manufacturing:** 11 hours to print shell, average power draw for 3D printers is 70W, 0.77kWh for 11-hour print
    - Source:https://3dprinterly.com/how-much-electric-power-does-a-3d-printer-use/
- **Operation:** When idling, the zero 2 W uses about 120 mA @5V, which is about 600mW. With our software running offline, the Pi itself would not draw more than 449 mA as this is the average current draw when all cores are stressed, and the inclusion of our amp + speaker setup at a gain of about 12 dB averages at about 150mA, so the average power consumption would be around 840mW
    - Source: https://www.cnx-software.com/2021/12/09/raspberry-pi-zero-2-w-power-consumption/
    - https://learn.adafruit.com/stereo-3-7w-class-d-audio-amplifier/downloads
    -

## Resource Use

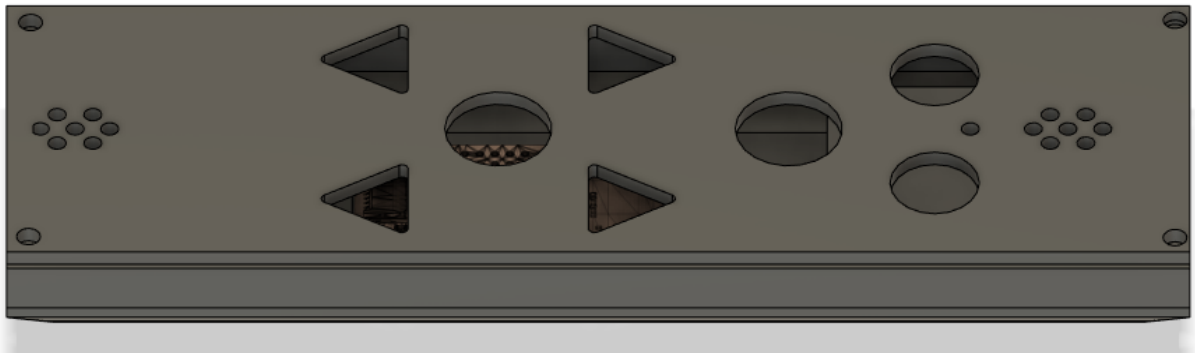
- **Material source**:
    - Raspberry Pi (based on A+ model): circuit board and CPU made of Silicon Dioxide, found in sand, and mined copper. Electrical parts are made of tin and precious metals. Materials used while processing raw materials include epoxy resin, photoresist, and tin.
        - Source: http://www.designlife-cycle.com/raspberry-pi
    - Shell: PLA plastic: vegetable-based plastic material, uses cornstarch as raw material. Made from fermented plant starch and primary natural material in the filament.
        - Source:https://www.sculpteo.com/en/materials/fdm-material/pla-material/#:~:text=PLA%20plastic%20or%20polylactic%20acid,material%20used%20in%203D%20printing.
- **Material usage**:
    - **Manufacture:** 65g filament for printing, Raspberry Pi, speaker, microphone, LED, miscellaneous wires
    - **Operation:** 9V batteries
- **Transportation**: Parts had to be transported to residence before assembly.
- **Renewable**: Powered by electricity, potentially renewable depending on the power source. Parts of internal hardware are recyclable but not renewable. Shell filament is made with renewable raw materials.
- **Disposal**: Shell is partially biodegradable(dependent on filament). Shell has the potential to be recycled but needs special treatment because of its lower melting point than other plastics. Shell filament can also be reused. Components on Raspberry Pi and in other hardware can be reused. Metals on PCB are extracted via acid leaching and recycled or

reused, but leftover residues go to landfills. Some plastics of internal hardware can be recycled.

- Source:https://www.sciencedirect.com/science/article/pii/S1878029616301499#:~:text=PCBs%20contain%20a%20lot%20of,reused%20based%20on%20their%20status.
- Source: https://all3dp.com/2/is-pla-recyclable/

# Models and Simulations

Modeling - To make sure everything physically fit together, we had TRAVIS modeled in CAD. This included our non-3D printed parts such as LEDs, the RPi, and buttons.



Simulations - We had a few “simulations” to test various parts without assembling our whole project. For one, our voice recognition could be run independently since it didn’t require any special hardware. This would essentially simulate pressing our voice command button since it would listen, text-to-speech, and then print out what it interpreted.

Another way we could simulate events was by using irsend, the LIRC command for manually sending an IR signal from the command line. We could use this to simulate our program sending commands to make sure the communication layer of our project was functional.

A few simpler ones were also used, such as connecting a GPIO pin to ground with a wire to simulate a button press or using a regular LED to visually tell if a signal was sent without requiring a TV to interpret it.

# Evaluation

# Functional Prototype

Our functional prototype was ultimately successful at handling our most important and challenging features, though we did learn from a lot of mistakes made along the way.

The prototype implemented the following of our intended features:

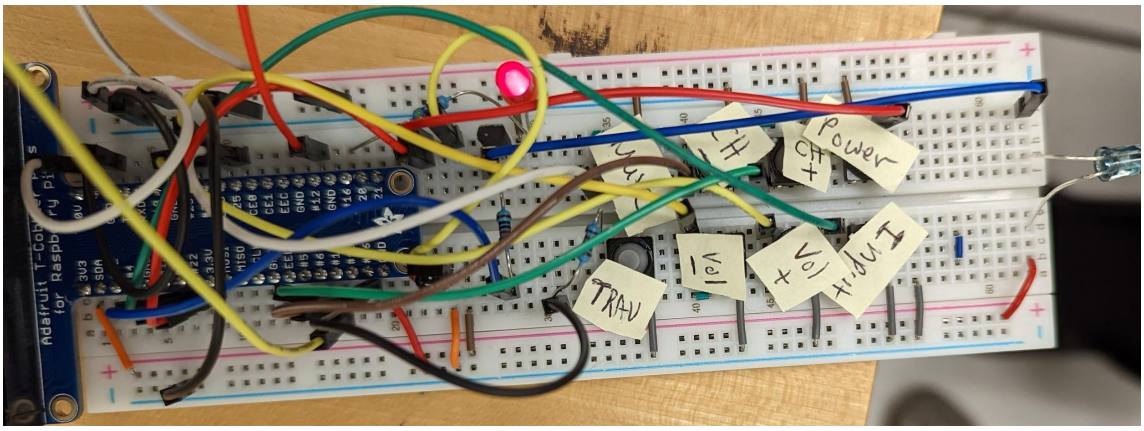
- Large, Simplified Button Layout
- Voice Integrated Capabilities
    - Status LED to indicate if TRAVIS is listening, auto turn-off always on
- Simple Set-Up

Several features were not implemented, either not fully implemented or not at all implemented:

- TRAVIS auto-deafen as a setting
- Set-up running through every option from a manufacturer or saving signals from current remote
- Find-My-TRAVIS
- Rechargeable or Replaceable Batteries

Most of these features that didn't make the cut were not necessary to demonstrate the viability of the project and were instead left as an exercise for a full implementation. Having TRAVIS stop listening automatically only as a setting, having a giant list of manufacturers for setup, having Find-My-TRAVIS, and working off of a battery all were very low priorities for our fast-paced in-person demonstrations.

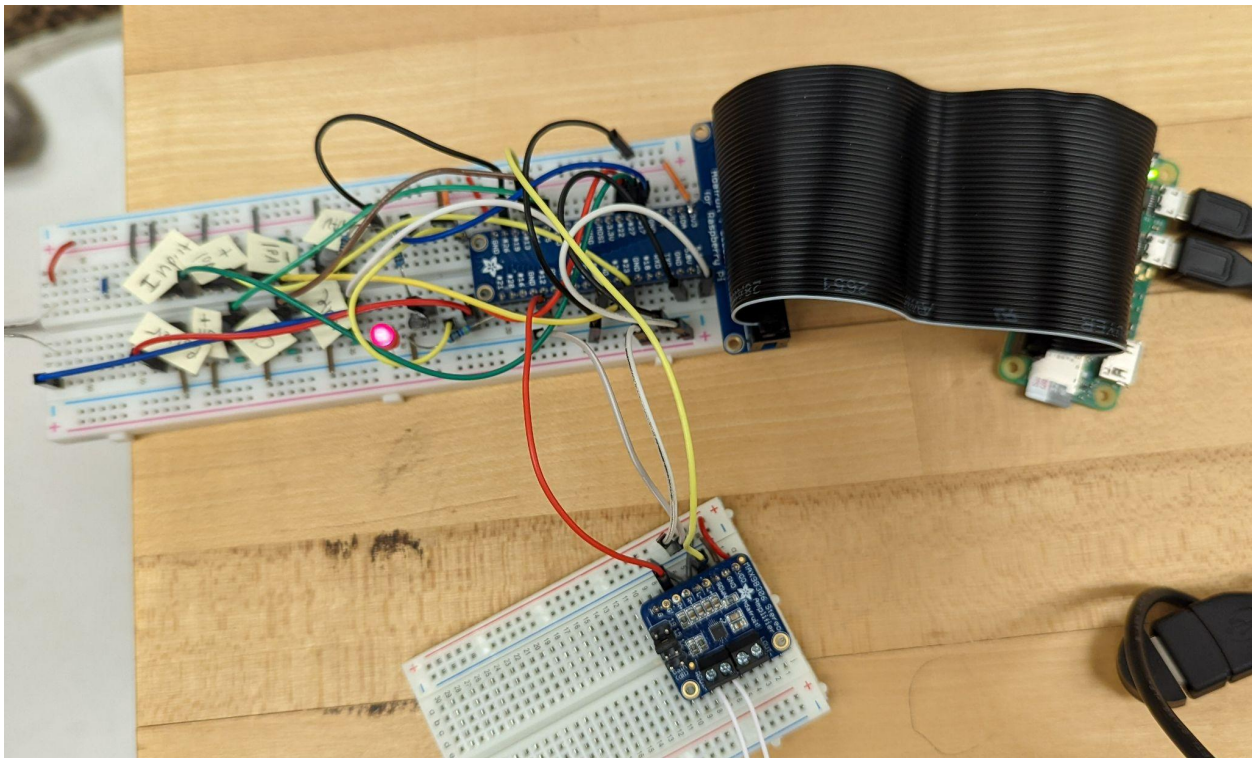
The prototype's primary feature is its voice recognition. This was done using Google's voice recognition libraries so that processing for the voice commands could be done with an internet connection. Alternatively, the device can use CMUSphinx without an internet connection. However, CMUSphinx is less accurate than Google. An IR-emitting LED sends signals to the television and a receiver takes in signals from the television. Below is a picture showing the wired prototype used in our presentation. The buttons shown are volume up & down, channel up & down, power, input, mute, and a button labeled "TRAVIS" to activate voice commands.



We designed our remote to have full functionality in both manual and voice mode. We programmed this functionality using python, but our voice recognition accuracy was less than what we hoped for. Ironically, due to our poor prototype microphone, TRAVIS had trouble interpreting the word “TRAVIS,” which the user says before a command. Text-to-speech would sometimes hear “TRAVIS” as a completely different word or would ignore the word. This did not come up in testing as our laptop microphones were leagues above the quick fix we used for our prototype and, as such, sent much more accurate sound data off to our speech processing software.

Similarly, TRAVIS’ audio output, while functional, could be a little hard to hear due to the speaker’s vibrations resulting in itself being vibrated due to its lack of proper enclosure, a flaw we imagine will disappear when soldered into a PCB.

An important command we included was the “TRAVIS initialize” command which syncs the remote with the TV model specified. For the sake of the prototype, it was specified to sync up with the Samsung brand TV provided to us. The prototype also could assign certain channels to numbers. For example, the user could say “Travis, assign CNN, twelve,” which would make it so channel 12 would go to CNN when using the TRAVIS remote.



# Testing

Testing our prototype was done in various stages. Some parts could be developed and tested in isolation, while others integrated too closely with our overall project and had to be run on our prototype as a whole.

One part which could be tested on its own was the voice interpretation. It didn't rely on any specialty hardware or software, since most laptops include a microphone and the libraries we used are cross-platform. This was very useful since it allowed us to develop parts at the same time without physical access to our model or building multiple, which would have been a challenge due to the limited availability of Raspberry Pi hardware. Once voice recognition and command interpretation were mostly working, it required much less thorough testing when integrated into our design than it would have if it needed to be developed on our hardware.

Aside from that, the first round of testing we had to do was just getting a baseline on the pi itself and how to interface with devices. We had to make sure all of our dependencies could be installed and the LIRC daemon could run. LIRC comes with a command-line tool to manually send IR signals which was very useful for testing TV communication with minimal setup required. We could also use a non-IR LED connected to the same pins, which allowed us to visually see if a signal was being sent without needing to worry about if there was an issue between the IR LED and the TV.

The last round of testing involved all of the parts working together. Specifically, we had to first make sure that each button sent a signal to the TV correctly and consistently. Next, we had to make sure that voice commands were being interpreted (mostly) correctly and that they sent the proper signals to the TV. We had a series of phrases to test which had several variations in wording of the same commands and at least one that corresponded to each applicable command. Our program would output the transcription to a console and say which action it was doing through the speaker, so it was relatively easy to see at which stage a command went wrong (if it did).

Interestingly and much to our chagrin, TRAVIS had a much harder time hearing voice input than in our (laptop microphone-powered) unit tests, to the point that a lot of the time it would miss hearing its own name, and thus not recognize a command that was otherwise properly recorded had been spoken. This ended up being due to the poor quality microphone we used in the TRAVIS prototype as a quick fix for driver-related issues.

Thankfully everything else in the prototype worked about as fine as they did individually, from the range on our IR LED to the response time from our buttons and everything in between.

# Appendix

# Brainstorming

- 3 design ideas: remote control, phone app, or box that plugs directly into TV
- Remote control:
    - Have a box that receives the signal, use remote to communicate with
    - Large buttons, easy to press
    - “Find remote” command lights up remote, possibly vibrates, makes chirping sound until the button is pressed or command “Okay remote, I found it” is given
- Voice-activated, give voice commands to change channels, lower volume, increase volume, turn on/off, source, mute
- AI is called TRAVIS (Television Remote and Voice Interpretation System)
- Initiate voice commands with “Hey Travis”
- LED lights up when a voice command is being processed, turns off when done
- Needs a detailed set of instructions to make setup easy
- May not need to require internet, Bluetooth connection?
- Simple to charge, charging stand, possibly wireless charging
- Different language settings
- Makes a sound when a voice command is initiated
- Have a button to press to initiate voice command so remote isn’t always listening
- Option to preset channels
- Be able to find a specific channel for the user, like if they say “Hey remote, put on Discovery Channel”
- Type in channel numbers for user
- For a specific program, like “The Amazing Dr. Pol”, tell the user what time the program is on and what channel
    - Ask the follow-up question “Would you like to go to this channel?”
- High contrast color palette for remote, black remote white buttons, white remote black buttons; yellow remote black buttons, black remote yellow buttons

# 6-3-5 Method

| Idea 1 | Idea 2 | Idea 3 |
|---|---|---|
| have a button on remote that can be pushed to locate it | charging stand for remote | App displays remote's current location and charge |
| Maybe a button that is separate from the remote to locate the remote. For example have the button integrated into the couch or wheelchair if they have one. Also have the button cause the remote to make a sound when it is pushed. | Make the charging stand a plate instead. Could be a large charging plate so that it is easy for people who cannot see well, to simply put it down without having to get it into an exact charging position. Maybe could even make like a little sidetable where the whole top of the able can charge the remote just by sitting on it. | Maybe have a camera embedded in the remote so when using the app and trying to find its location you can turn on its camera to see what the remote is seeing. |
| Bluetooth-connected trackers to locate devices embedded into remote, possibly some haptic feedback for those who can't hear well and see well, maybe the remote vibrates more as you get closer to it (Marco Polo remote) | Magnetic charging stand, would help if remote has a rubber casing so you could just toss it or place it on any part of the surface so it latches on | possibly a 360 fisheye camera? in case the camera side is facing down |
| Make it have a function where you can ask the remote where it is vocally And it'll respond if it hears you | A wireless and wired charging option like smartphones have | Have a tiny pinhole camera that cam see its surroundings? |
| yeah maybe like a wall mounted thing so you can put it next to your lightswitch so that whenever you hit it the remote beeps and vibrates and whatnot | probably just wireless or swap the batteries, maybe two different versions cause rechargeable batteries are hard to find anyways wireless charging is probably simpler to understand if u give it like a charging indicator or something | maybe give each remote a code so that the setup isnt wierd and you can just attach it to any unattached remote probably a security risk idk |
| Could give the remote a phone number that will buzz/ring the remote when you call it, so no extra devices needed. | Charging stand could have the button to locate remote since it'll probably be in one place mostly. | could use the camera for face recognition so it knows who's using it and can have personalized settings, probably pretty hard to actually do though |

| Idea 1 | Idea 2 |
|---|---|
| Box sends IR signals directly to tv from like 6 inches away so no lag or missed inputs. Could be connected to with a remote or app or just voice controlled. | Any of our ideas could search through a channel guide from the internet to find what's playing at what time, and if someone wants to watch something not on at the moment it could let them know when it would be on. Could maybe find some natural language processing to match what's said with show titles instead of basically speech to text and ctrl-f.This might've been said already but I can't remember. |
| use bluetooth connection instead of infrared. doesn't rely on direct line of site with the box or short distance | could have a "favorites" list for the user that tells them when their favorite show is on and on what channel; would need some way to periodically check channels and a display; could alternativey store the info and then have the voice command "hey TRAVIS, when are my shows on today?" |
| Bluetooth sounds good, other than that maybe put something in place so that not just anyone can tamper with the tv and change the channels. | Could also add a capability to remind users when a show is about to start. So you tell the box to tell you when this program is about to begin and then the box rings an alarm out to you or just speaks out "hey your show is about to start in 5 minutes, would you like me to put it on the tv? |
| Avoid tampering by having voice authentication, have them repeat vowels and consonants so it can recognize their mannerisms, finger print sensor on physical remote | have this work with multiple devices set around the house, or combine it with a wristband so the user knows if the show will start if they are not within tv proximity at the moment |
| Implement profiles for different people that use the remote | Could ave something like speeddial for channels the person really enjoys? Easily configurable |
| security setting probably isnt needed as old people would probably enjoy pranking people and like, its a tv who would mess with it | yeah like if we have a remote we can just put it on a button to go to whatever channel the favourite one is on, same goes for app |

| Idea 1 | Idea 2 | Idea 3 |
|---|---|---|
| TV Bar like the ones that attach for things like a wii | tv remote but you can point it at things and click on them + being able to use it like a normal remote | alexa/google/whatever voice box integration so like a module that attaches to it that lets it send tv signals but otherwise just uses its normal voice recognition capabilities |
| Use for gesture controls and a pointer? Coudl also build in some sport and party games lol | Oh this is pretty much what I said in the last column oops. Idk what else to add but this could make the controls more intuitive than a standard remote. | This would be cool, voice assistants usually have good smart home/external device control I think. If we were to do this we might be able to take advantage of the advanced language processing capabilities these companies have too. Could also probably connect with the users phone as well since that's usually linked with a hardware assistant. |
| have certain dances connected to turning switching to a preset channel, way for a senior to get a little bit of exercise; ex) the macarena means switch to Disney Channel | bruh. basically a wii remote, would needd to include calibration, have simple calibration mini game where you connect the dots to draw a picture? | if we use a google home we could use its voice recognition features potentially, not sure how that would work though but it would be the interface for communicating with the TV. |
| maybe also every hour or so it tells you to get up and do a little movement before it lets you continue watching. | idk what else to add sorry | maybe use it for more than just the tv, and have it be able to control multiple things in your home. |
| Group exercises for people in retirement homes watching TV, or for friends watching the same show at different locations, more incentive to move about if against ur pals? | uhhhhhh, so a Bluetooth controlled remote instead of IR? Idk an app works better idk how to specify this either | Improving upon the smart plug application. this exists with limitrd capabilities like turning on/off, on duration. could make it more applicable |
| Could allow for setting timers to automatically turn off the TV or something? Like how some radios have sleep timers | Add a smaller screen for you to use certain functionality with? At this point now it's a Wii U | Can control any smart devices in your house, including the television, light bulbs, locks, etc. This would fit best as an app or something |

| Idea 1 | Idea 2 | Idea 3 |
|---|---|---|
| A television addon similar to a Roku box where it adds extra voice-recognition functionality rather than replacing what the TV already has. | An app similar to the Apple Remote app which, instead of being button based, would be voice-controlled. | Something that goes on your wrist?? |
| Check if Roku lets you develop apps to work on it, maybe be able to work with already existing stuff and just hook up a microphone to it | See if we can find apple remote source code or a binary of it and just add voice functionality and have it translate into apple remote commands | maybe we can add those like wiimote straps to it like how old people have their glasses attached to their ears with a string so the remote gets lost less? |
| I think Roku remotes already have a microphone iirc. I think it's mostly only for entering text though so definitely room to add functionality like commands. | Might be easier to interface with than old fashioned tvs and channels | Maybe also have it able to tether to like a table or couch so it doesn't get lost |
| could use the text entering function to enter in shows on netflix/hulu/whatever, text entering could also be used to "manually" type a channel number on the TV | Keep in mind that some older people don't have smart TVs/ don't feel comfortable or are scared of using them. Would user need to be savvy or can someone set it up for them once and then they can operate it wj/o help? | have a clip(like a book light clip) w/ a tether for the remote, needs to be somewhat sturdy |
| idk what else to add, idk what roku is sorry. | Could try to make this without needing internet, just have all the functionality right inside the box. | make it comfortable so that it does not hurt after a while, also could make it a simple necklace that you can speak into that is directly connected to the tv, you would only take it off during bed on the nightstand to charge. |
| This exists, roku does let 3rd party apps develop for it, base roku remote is basic, idk all these ideas are remotes y'all lmao, roku attachments for ergonomics since the roku infrastructure is already decent | No internet is very limiting, makes it harder to update later on too. Idk why not include eye tracking instead, much more reliable, or combine the two for better accuracy | Have labels for easier identification, maybe braille on them too |

| Idea 1 | Idea 2 | Idea 3 |
|---|---|---|
| App to detect public facilities (bathroom, water fountain etc) | Electronic posture checker | Ergonomic phone cases (idk lol) |
| Built in map which shows where the nearest restrooms are? Could come in handy at amusement parks, malls, etc. | An app that detects your posture using some kind of outside electronic device on your back or something? I could use one of these lol | A phone case that melds to the shape of your hand in order to maximize comfort? Does that really need electronics tho or can it just be memory foam? lol |
| could make a service that will go to places for companies and walk around and mark things on google maps for them at like a cost or something | can see if we can just make one of those posture holding things with a bit of slack in them and one of those checkers so it holds you a bit but also makes sure you know when you're slacking and stuff | phone case combined stress ball and stim toy so you can like squeeze it when stressed and otherwise play with it like those silly putty things or those like goo balls that expand between ur fingers when u squeeze them |
| automate it somehow? Or some way to add a bunch of tags at once for places that are expanding quickly. It would be cool to see smaller but importand things on google maps but idk the process for adding stuff. | could track your posture throughout the day like a fitness or sleep tracker and let you know trends about your posture | case with slots for multiple types of tactile things to suit people's preferences |
| app that user can use to make a detailed map of a building or amusement park, could be large or small scale, map can be shared and if the map includes restaurants then the user can pull up yelp reviews | user can set a goal for how they want their posture to be, keeps track of whether posture is consistently at goal posture, if not gives tips on how to improve posture and possibly other health tips | compartments on phone case that have a latch, can be popped open or close, see through |
| could also add capabilities for the user to add a suggestion, if they found somehting not already on the map. | Could have something that tells you at what time of the day your posture is the worst or what activity you were doing during that time. | something that would be personal to you so that only you know how to open and close the compartments. |

| Idea 1 | Idea 2 | Idea 3 |
|---|---|---|
| Make everything built into the remote so that no internet is required. The current generation of old people did not grow up with internet and are too old to try to understand how to use computers or smartphones. | Add functionality to call people from their landline by using asking the remote to call them. Since some of the elderly have a hard time seeing, they can't type in a phone number so having a set of saved numbers and telling the remote to call could be cool. so basically it connects to the tv and the landline? | Add a functionality to video call, if internet capability is there. |
| Simplified remote voice assistant to navigate internet, kind of limited without internet. Have it use olden slang to resonate with the older generation | An all in one remote, replacing the use of touch screens with a remote control buttons for ergonomics | Camera embedded into remote i guess, this is basically a phone that is embedded into a remote... Idk |
| Make it very easily understandable, have a built in tutorial or guide to help them navigate the device. | Make it super universal so it doesn't just apply to television. Can be hooked up to other devices like smart bulbs and stuff. | Add ability to connect to smart phones so they can take pictures from afar?? Idk |
| could make this into a monetizeable thing so like we partner with cable companies to provide their customers with extra programmable chips for their remote to include the new channels and shows being offered lol | at that point it would probably be simpler to just make it a phone app | can make it like a laser pointer or something instead? |
| Do new channels ever get added or do they just move around to different bands/get replaced? Either way, could probably get cable companies to pay us for making their stuff more accessible. | (smart)phones already can call through voice commands so idk if it would be useful as a phone app | Camera could go on the tv and do video calls on the tv screen too? |
| make a detailed manual about how to use the device, try to make it local to the TV so old folks don't have to deal with the internet; TRAVIS responds to voice inputs like "Okay partner, navigating to channel 165". Has a southern accent. Good ol Travis | make it possible to connect to landlines in the house as well so can dial number or quick call someone for you, would be overly complicated | have camera on top of remote, hold up like a smartphone and display face on TV and person who is calling |

## Reflections on Brainstorming and 6-3-5

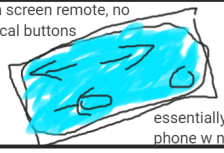
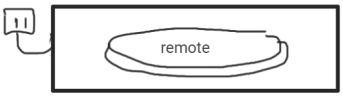
Since the group already had an idea of what we wanted to do, we found the brainstorming technique to be a bit more helpful. We were able to collaborate and receive feedback on ideas immediately instead of waiting to see how ideas developed with the 6-3-5 method. We felt that the 6-3-5 method would have worked better for us if we did it closer to the beginning because doing it later in the project felt like brainstorming with additional constraints.

# Morphological Chart

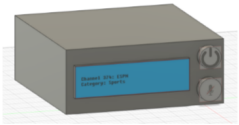
CSE123G9
Morphological chart
02/02/2022

TV Remote
Morphological Chart

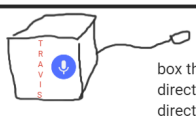
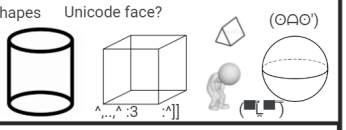
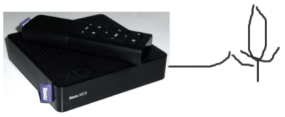
maybe dedicated buttons for favorite channels?

| | input method | button layout | other features |
|---|---|---|---|
| | general remote style  |  channel up/down, vol up/down, power, mute, voice button |  keypad input with cover if it gets fat-fingered often |
| | touch screen remote, no physical buttons  essentially phone w no OS |  above design + keypad |   wireless charging plate |
| | pure audio remote  | only audio related buttons  | if lost, LED flashes, remote vibrates, chirping noise   |

CSE123G9
Morphological chart
02/02/2022

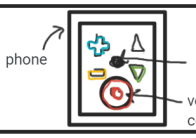
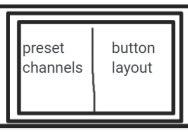


Alexa Box
Morphological Chart

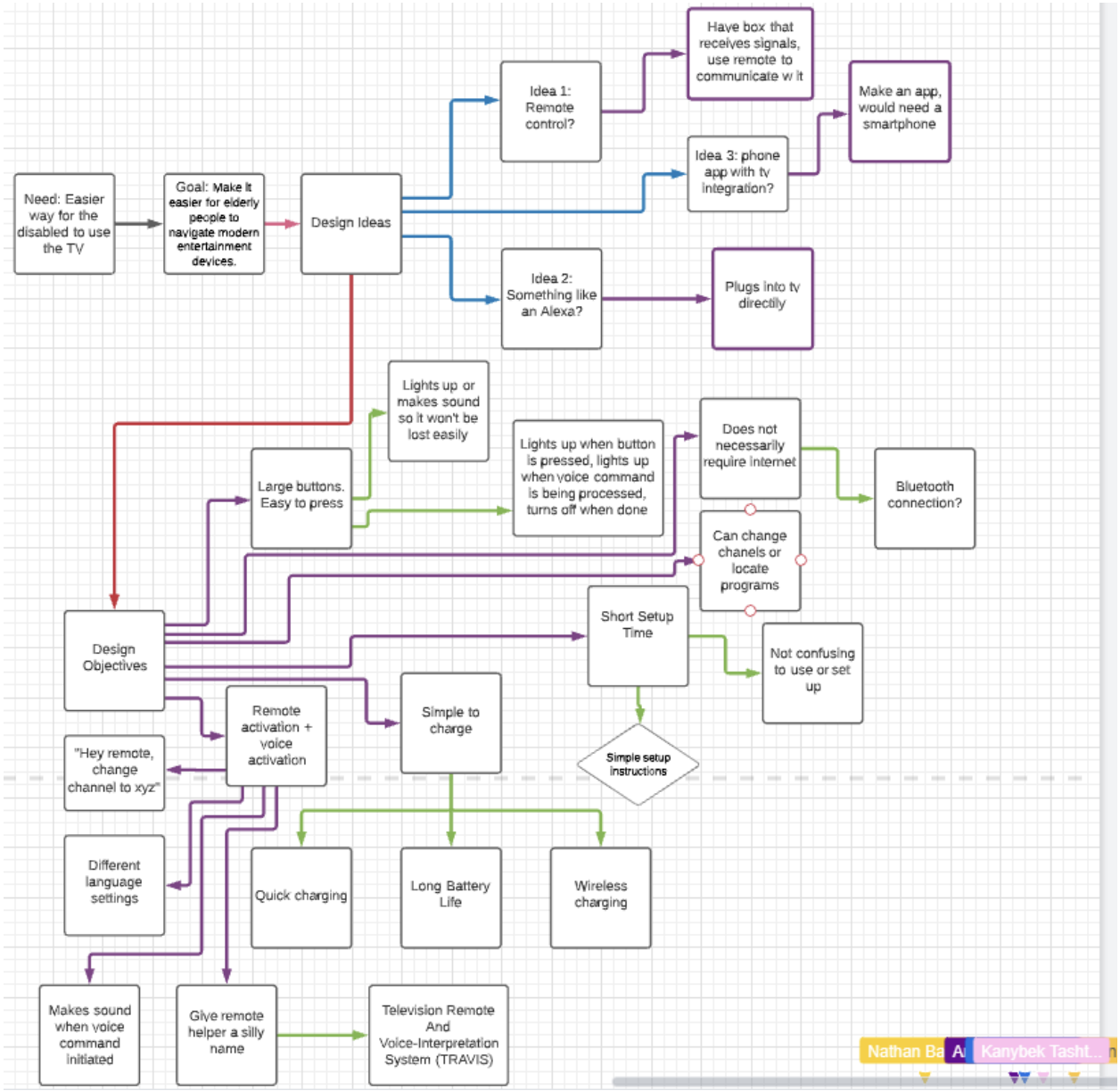
| | technology | TV connection style | T.R.A.V.I.S. Features |
|---|---|---|---|
| |   custom cable box |   box that plugs directly into TV w/ direct voice input | shapes Unicode face?   |
| |  roku/alexa app with potential add'tl hardware |   IoT? | Robot Lady voice. Generic but trustworthy<br>one of our voices with preset responses<br>Text to speech database?<br>texan accent, fall in love with TRAVIS? |
| |  custom alexa style soundbox |   Sends IR signals to TV/emulates remote inputs | have TRAVIS respond to input "Changing to Channel 145" or "I'm sorry, did you mean this?" small lad |

CSE123G9
Morphological chart
02/02/2022

Phone App
Morphological Chart

| | Layout | Features | Way of Registering Commands |
|---|---|---|---|
| |   | one page with list of preset channels, one page of buttons, one page showing current channel number and program name | custom database/neural network training<br>Predicts what you want to watch based on past behavior |
| |   -Channel display -Mute button -Slider to switch between changing channel and volume | Search bar or search engine of available channels. Can input streaming services you are subscribed to. Will add them and your cable plan to app's search list. Will try to play what you choose | text to speech + speech to text - use already created open source software<br>Confirmation before continuing with command |
| | better for a tablet   | Has a favorites list that can tell the TV to alert the user when a show is playing on a specific channel or if their show is appearing on any channel<br>Favorite channel speeddial option | Options in-app to set alarms/reminders/ notifications for when a show is playing.<br>Sleep timer & time-limiting options |

# Mind Map


Need: Easier way for the disabled to use the TV
Goal: Make it easier for elderly people to navigate modern entertainment devices.
Design Ideas
Idea 1: Remote control?
Have box that receives signals, use remote to communicate w it
Idea 3: phone app with tv integration?
Make an app, would need a smartphone
Idea 2: Something like an Alexa?
Plugs into tv directily
Design Objectives
Large buttons. Easy to press
Lights up or makes sound so it won't be lost easily
Lights up when button is pressed, lights up when voice command is being processed, turns off when done
Does not necessarily require internet
Bluetooth connection?
Can change chanels or locate programs
Short Setup Time
Not confusing to use or set up
Simple setup instructions
Simple to charge
Remote activation + voice activation
"Hey remote, change channel to xyz"
Different language settings
Quick charging
Long Battery Life
Wireless charging
Makes sound when voice command initiated
Give remote helper a silly name
Television Remote And Voice-Interpretation System (TRAVIS)

## Design Objectives

| Design Objective | Units | Target/Range |
|---|---|---|
| Short setup time | minutes (Alexa set up as an example) | < 10 |
| Short times needed to change channels | seconds | < 20 |
| Inexpensive to manufacture | dollars | < 10 |
| Device Uptime | hours | > 20 |
| Task Startup | seconds | < 5 |
| Voice Recognition Accuracy | Percentage | >60% |
| Damage Likelihood (External) | Percentage | <10% |
| Average Lifetime (Internal) | Years | >2 |
| % Recyclable/Biodegradable Material | Percentage | >50% |

# Decision Table

| **Criteria** | **Weight** | **Design 1: Remote** | | **Design 2: TV Box** | | **Design 3: App** | |
|---|---|---|---|---|---|---|---|
| | | Raw | Weighted | Raw | Weighted | Raw | Weighted |
| Cost | 10 | 4 | 40 | 2 | 20 | 5 | 50 |
| Device Uptime | 10 | 4 | 40 | 5 | 50 | 2 | 20 |
| Task Startup Time | 15 | 1 | 15 | 4 | 60 | 3 | 45 |
| Voice Recognition Accuracy | 25 | 2 | 50 | 2 | 50 | 2 | 50 |
| Damage Likelihood (External) | 5 | 3 | 15 | 4 | 20 | 5 | 25 |
| Average Lifetime (Internal) | 10 | 5 | 50 | 2 | 20 | 5 | 50 |
| Recycleability | 5 | 5 | 25 | 1 | 5 | 5 | 25 |
| Setup Time | 15 | 5 | 75 | 5 | 75 | 5 | 75 |
| Manufacturing Cost | 5 | 4 | 20 | 2 | 10 | 5 | 25 |
| **Total** | **100** | **33** | **330** | **27** | **310** | **37** | **365** |

## Reflections on Decision Table

While coming up with various metrics, we realized that the inclusion of the app as the main design idea introduced a lot of factors that other designs couldn't compete with, i.e. that we would likely not need a manufacturing cost, that the app only breaks when the phone breaks, etc. Because this decision table must be quantifiable, and we would have trouble quantifying user experience at this stage, a lot of the flaws for the app version do not show through, and instead the strengths of the medium are overrepresented. For example, ease of use was not a factor that we could easily quantify but is a concern with the app, especially for older people who do not have smartphones. The app design also doesn't consider the cost of a smartphone, which can vary depending on brand and model, and the cost of a base product, other than a TV, does not apply to the other designs.

Similarly, because we have product costs and manufacturing costs, some parts of our designs are double-counted. We felt both criteria were too important to exclude, but it introduced an effect where money factored in higher than what we would have liked. Similarly, because our guesses for the average end-product cost and manufacturing costs were similar on the scales we used, they ended up just compounding.

At the end of the day, we convened and decided that the app, while good on paper, does not have the ease of use we want for our product. We decided to continue with the original remote control idea as we feel that it would score the best on the user-experience bracket because of the physical buttons involved. Besides that, we would much rather make a physical product over a phone app.

# Evaluation Scale

| Criteria | Units | Design 1<br>Remote | Design 2<br>TV Box | Design 3<br>App (only this app) |
|---|---|---|---|---|
| Cost | US Dollars [$] | 25 | 50 | 10 |
| Device Uptime | hours | > 1k | Always | 20 < x < 50 |
| Task Startup | Seconds | 5 | 1 - 2 | 2 - 3 |
| Voice Recognition Accuracy | Qualitative (% words recognized) | > 60 | > 60 | > 60 |
| Damage Likelihood (External) | % (Newtons of force? Passed drop tests?) | 10 | 2 | N/A |
| Average Lifetime (Internal) | Years | 10 | 5 | Could be infinite with proper updates |
| Recyclability | % material that is recyclable | > 90 | > 50 | N/A |
| Ease of use | Qualitative (Survey fill out?) | > 75% | > 75% | > 75% |
| Manufacturing Cost | US Dollars [$] | < 10 | < 25 | > 25 |
| Setup Time | Minutes | 5 | 5 | 2 |

## Evaluation Scale for Ranges: Numeric Scores

| **Cost Range** (Dollars $) | **Device Uptime Range** (Hours) | **Task Startup Time Range** (Seconds) | **Voice Recognition Accuracy Range** (%) | **Average Lifetime (Internal) Range** (Years) | **Recyclability Range** (Percent %) | **Numeric Score** |
|---|---|---|---|---|---|---|
| x <= 15 | Always | x <= 1 | 90 < x <= 100 | x >= 10 | 90 < x <= 100 | x5 |
| 15 < x <= 30 | 1k <= x | 1 < x <= 2 | 80 < x <= 90 | 8 <= x < 10 | 80 < x <= 90 | x4 |
| 30 < x <= 45 | 100 < x <= 1k | 2 < x <= 3 | 70 < x <= 80 | 6 <= x < 8 | 70 < x <= 80 | x3 |
| 45 < x <= 60 | 10 < x <= 100 | 3 < x <= 4 | 60 < x <=70 | 4 <= x < 6 | 60 < x <= 70 | x2 |
| 60 <= x | x <= 10 | 4 < x | x <= 60 | x <= 4 | x <= 60 | x1 |

| **Setup Time** (Minutes) | **Damage Likelihood (External)** (Percent %) | **Manufacturing Cost** ($) | **Numeric Score** |
|---|---|---|---|
| x <= 15 | N/A | x <= 5 | x5 |
| 15 < x <= 30 | x <= 5 | 5 < x <= 10 | x4 |
| 30 < x <= 45 | 5 < x <= 10 | 10 < x <= 15 | x3 |
| 45 < x <= 60 | 10 < x <= 15 | 20 < x <= 25 | x2 |
| 60 <= x | 15 <= x | x > 25 | x1 |

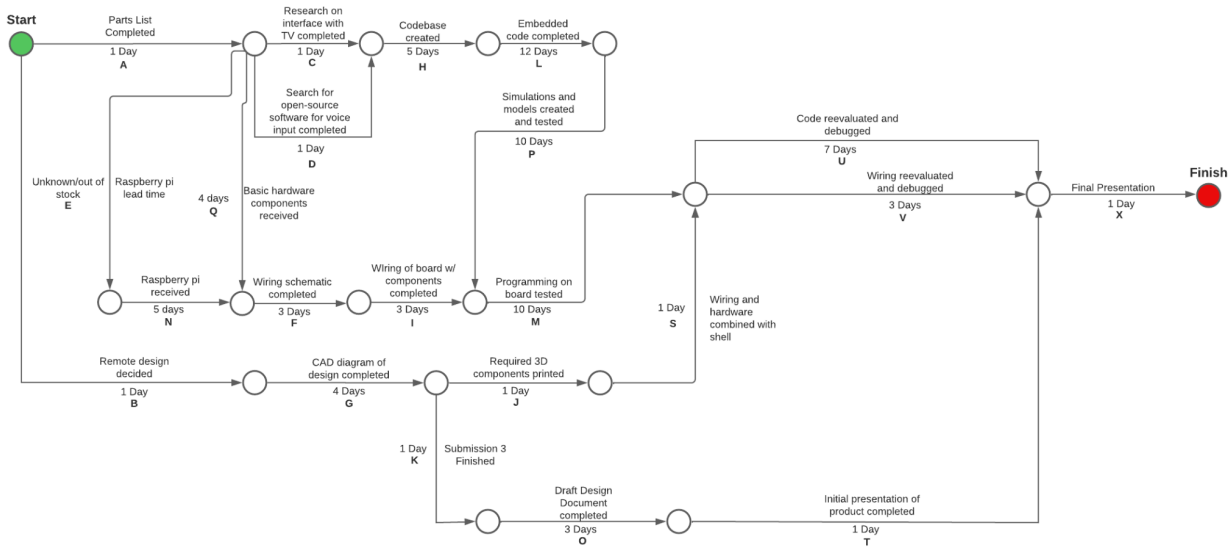
# Weights

| **Criteria** | **Weight** |
|---|---|
| Cost | 10 |
| Device Uptime | 10 |
| Task Startup Time | 15 |
| Voice Recognition Accuracy | 25 |
| Damage Likelihood (External) | 5 |
| Average Lifetime (Internal) | 10 |
| Recyclability | 5 |
| Setup Time | 15 |
| Manufacturing Cost | 5 |
| **Total** | 100 |

# Gantt Chart

| Task ID | Task Title | Task Members | Start Date | Due Date | Time Alotted (in Days) | % Task Completed | Phase (each phase is 3 weeks) |
|---|---|---|---|---|---|---|---|
| 1 | Submission 1 | All | 01/16/22 | 01/23/22 | 7 | 100 | 1 |
| 2 | Individual Contribution 1 | All(Individual) | 01/30/22 | 02/04/22 | 5 | 100 | 2 |
| 3 | Submission 2 | All | 01/24/22 | 02/06/22 | 12 | 100 | 2 |
| 4 | Submission 3 | All | 02/07/22 | 02/20/22 | 13 | 100 | 2 |
| 5 | Draft Design Document | All | 02/22/22 | 03/06/22 | 13 | 70 | 3 |
| 6 | Parts List Compilation | All | 02/04/22 | ongoing | -- | -- | 3 |
| 7 | Research on interface w/ TV | Nathan Banner, Ann Sophie | 02/04/22 | 03/18/22 | 42 | 80 | 3 |
| 8 | Codebase Creation | | | | | | |
| 9 | Embedded Code Creation | | | | | | |
| 10 | Search for Open-Source Voice Recognition Software | All | 02/04/22 | ongoing | -- | 30 | 3 |
| 11 | Finish Schematic | | | | | | |
| 12 | Simulation and Model Testing and Creation | | | | | | |
| 13 | Programming on Board Testing | | | | | | |
| 14 | Remote Design Decision | | | | | | |
| 15 | Finish CAD Design | | | | | | |
| 16 | Print 3D Components | | | | | | |
| 17 | Combine Hardware and Shell | | | | | | |
| 18 | Reevaluate and Debug Code | | | | | | |
| 19 | Reevaluate and Debug Wiring | | | | | | |
| 20 | Interim Presentation | All | 02/22/22 | 03/10/22 | 17 | 90 | 3 |

# Critical Path Method (CPM)




We have two notable critical paths in this project. The first path includes waiting on all of our parts to arrive and completing the wiring portion of the project. This path is A, Q, F, I, M, V and we estimate it will take about 24 days. We estimate it will take about 4 days for us to receive all of the basic hardware components. We are unsure how long it will take for us to receive the Raspberry Pi because it is out of stock so we did not include this event in our time estimate. We estimate it will take about 5 days for the Raspberry Pi to arrive once it is back in stock. The largest delay on this path is 10 days, which is how long we expect full testing of the wired and programmed board to take.

The second notable critical path is the path for creating the code and completing testing. This path is A, C, D, H, L, P, M, U and takes 47 days. The largest time delay on this path is caused by programming and testing. The path may not take the full 47 days because some tasks, such as completing the embedded code and simulating and testing it, can be done concurrently by multiple members, reducing the total time of the critical path.

The delay caused by the Raspberry Pi obstructs part of the CPM, but we can still create code and perform most of the necessary testing with a different Raspberry Pi while we wait for the part.

# CPM Inputs

| Activity | Duration (day/s) | Preceded by |
|---|---|---|
| A | 1 | – |
| B | 1 | – |
| C | 1 | A |
| D | 1 | A |
| E | 1 | A |
| F | 3 | A, E, Q, N |
| G | 4 | B |
| H | 5 | A, D, C |
| I | 3 | A, E, Q, N, F |
| J | 1 | B, G |
| K | 1 | B, G |
| L | 12 | A, D, C, H |
| M | 10 | A, C, D, H, L, P, E, N, Q, F, I |
| N | 5 | A, E |
| O | 3 | B, G, K |
| P | 10 | A, C, D, H, L, P |
| Q | 4 | A |
| S | 1 | B, G, J |
| T | 1 | B, G, K, O |
| U | 7 | A, C, D, H, L, P, E, N, Q, F, I, M, B, G, J, S |
| V | 3 | A, C, D, H, L, P, E, N, Q, F, I, M, B, G, J, S |
| X | 1 | A, C, D, H, L, P, E, N, Q, F, I, M, B, G, J, S, K, O, T, U, V |

# Division of Labor During Prototype Phase

Most of the labor was split up via each contributor's interests, expertise, and most importantly availability. At the beginning of the project, Nathan displayed interest and research into our voice integration, so he was assigned to that. Similarly, Aidan, Heath, and Kenny displayed expertise in CAD. so they went and handled that. Katy handled wiring and physical integration, and Lillian & Ann Sophie handled the remainder of the code.

As time went by, Heath and Kenny eventually branched out into setup and integration, with Heath creating our Dockerfile and Kenny getting everything tested, and them also spending time handling each other's work. At the same time, Aidan handled almost all of our CAD, creating changes to it while sending it over to Heath to print. Katy handled some of our wirings while also getting our documentation ready, while Nathan, Lillian, and Ann Sophie handled the odd jobs left in between coding everything up.

Before, throughout, and after the prototype phase, there were tri-weekly meetings to share progress, get coordinated, and keep everyone on track, hosted by Lillian who also handled a lot of the team management and brought snacks. Lillian, Katy, and Kenny also handled parts acquisition, ordering additional supplies for use in the prototype.

# Collaboration

Collaboration was handled in three main ways. First, organizing, meeting, and almost every discussion took place in our discord server, where meeting times were figured out by using when2meet.com and then set in stone for the remainder of the quarter, in addition to a bunch of informal discussions on the server directly. Second, code was worked on via our GitHub where we stored the actual code used by TRAVIS, the Dockerfile used to set up TRAVIS, test code for various features, and previous design submissions to show off to our friends, family, and job interviews. Finally, our CAD was collaborated on via Fusion360 which also gave us a helpful 3d rendering of our most up-to-date version of the prototype and its shell.

Additionally, design submissions were handled via Google docs, sheets, and slides, and were stored in this Google Drive folder along with many pictures used for our submissions.

# Test Plan

## Product Durability

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters (w/ justification) | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | BSOE Lab 250/252 | | Testing durability of remote shell. | Fresh printed shell, Standard electronics layout, fully assembled product. | Can survive most drops and continue functioning | Drop height, drop force, surface dropped upon. | 1 Meter Drop height, No force beyond gravity, Concrete floor surface | Reported damage (internal and external), continued viability of product |
| 2 | | BSOE Lab 250/252 | | Testing durability of remote shell at different heights (when does shell pop open? physical wear and tear? | Remote shell w/o hardware, yard stick (for measuring heights), hollowed cardboard box (drop site barrier, keep pieces from scattering) | Remote will survive most drops with wear and tear directly proportional to an increase in height | Drop height, landing site | at least 5 drops from different heights up to 10ft, no external force acting on remote besides gravity, concrete landing site | Shell damage |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions | Procedure: Data Collection Method | Other Factors |
|---|---|---|---|---|---|---|---|---|
| Drop onto specific surface from X height up with Y specific additional downwards force. | This should (mostly) cover expected scenarios of our remote being dropped out of someone's hands or knocked off someone's table. Thus we need to see the expected damage from such actions to prevent future problems with our product | Sample taken after single drop, noteworthy results after continued drops have been noted in the summary | Report on seen external damage, open casing and check for internal damage (e.g. loosened wires), and test if remote shut off and if it is still operational within seemingly normal bounds. | 15 | 0. Wear PPE, clear area of easily damageable goods.<br>1. Grab remote, hold up to given height above the ground.<br>2. Let remote tumble out of hand, potentially add additional force if testing for such.<br>3. Note down the orientation and rotational speed (subjectively) at the moment the device hit the floor.<br>4. Pick up remote, inspect for exterior damage, note this down.<br>5. Unclasp casing, inspect for interior damage, note this down.<br>6. Reseal product, test for continued operation and additional perameters. Note down operational status and perceived issues, if any.<br>7. Repeat steps 1-6 until desired number of samples taken. | Safety glasses warn at all times to protect against potential flying debris, rubber gloves used when checking device's internal electrical components for damages to prevent against potential shocks. Additionally, clear area of easily damageable goods (e.g. glass bottles). | Exterior damage: Inspect for any scratches, chips, damaged or lost buttons, and obviously damaged portions.<br>Interior damage: Inspect for broken wires, sparking components, damaged soldering, loose components, and so on.<br>Continued operational viability: Check if remote is still on, and if so if it is still functioning within bounds. Note if any queries take longer than normal or if significant reductions in audio processing capabilities have occured. | Experiment went smoothly with no additional factors of note. |
| Drop remote shell from various heights and make a qualitative assessment of the damage, damage will compound unless different shells are used, start from lowest height | Testing to be able to ensure consumer that shell will adequately protect the inner hardware of the remote. Older or disabled people may be less dextrous with hands, ensure remote will survive being dropped. | Note visible wear and tear after each drop. | Note visible wear and tear on inside and outside of shell for each height | 3 at each height | 0) wear safety glasses, clear area<br>1) Drop remote from marked height, note visible damage<br>2) Repeat step 2 three times at each predetermined height<br>3) repair remote between each drop if possible (such as snapping remote back together) | Beware of shrapnel from remote, wear safety glasses if near drop zone | | |

| Dependent Variables' Outcomes | Summary |
|---|---|
| Y/N internal damage reported, Y/N external damage reported, product remained viable and performed within bounds of other test results. | Remote is robust to faults from dropping from low heights on very hard surfaces. |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | BSOE Lab 250/252 | | Testing water resistance of shell | Freshly printed shell, no electronics inside | The shell will most likely not keep<br>out water at all | Type of water, water volume location | A few litres of regular tap water | Volume of water inside the shell afterwards | |
| 4 | | BSOE Lab 250/252 | | Testing scratch resistance of shell | Partial fresh printed shell, no electronics inside | Shell will be reasonably scratch resistant to accidental scratches, hard scrape with sharp object will definitely scratch | Scratching implement | pocket knife, butter knife, hard plastic, other scratching implement | shell damage | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Submerge the remote shell in tap water for varying amounts of minutes. Record results | This should prove whether or not<br>the remote is durable enough to prevent water from ruining the electronics, which will be better for older and/or disabled folks sincespills may be common in the household | Measure the amount of water present inside the shell after testing has occured | See where the water accumulates if at all and where the weakpoints of the remote are that allow the most water in | Different situations of water tests (e.g. spilling a cup, dropping it into a tank of water, droplets spilling onto the remote, etc.). Most likely 5-8 tests | 1) Assemble the remote shell together<br>2) Either submerge the remote in water or pour it on top<br>3) Wait for varying amounts of time and dry the outside of the case<br>4) Check the inside for water entry points and how much accumulated<br>5) Repeat steps 1-4 for different types of liquid submergence/spread and for different timings | Make sure there are no electronics inside the remote for risk of electrocution and wasting supplies |
| Place partial of shell on hard surface, secure with tape or other means to prevent sliding, scratch portion with object | Shows how scratch resistant/resistant to damage the remote is. Particularly important for fancy implementation with wireless charging to show consumer product will last | Note outer shell after scraping at different pressures and/or with different implements | | 2 tests with each pressure and implement, 6-8 in total | 1) Secure remote shell to work surface<br>2) Scratch surface of shell at different pressures<br>3) Note depth of scratch or if scratch occured<br>4) Repeat as necessary<br>5) Clean work area of any debris, dispose of properly | Wear gloves when handling scratching implements, safety goggles recommended |

# Battery Life

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | BSOE Lab 250/252 | | Testing battery life w/ regular AA batteries (min) | 2 fresh AA batteries, fully completed standard electronics setup. | Battery lifetime will go down as voice input increases, but button-related input will have a much more minor effect on the battery life. | Button usage (% of time per minute), voice input usage (% of time per minute), TRAVIS voice responses enabled/disabled, TRAVIS voice input receiving/disabled. | Button usage @ 0%, voice input usage @ 10%, TRAVIS voice responses enabled, TRAVIS voice input receiving. | Duration batteries last before there is not enough power to continue functioning. |
| 2 | | BSOE Lab 250/252 | | Testing battery life with wireless charging (min) | Wireless receiver and transmitter, Li-ion battery, internal hardware | Battery lifetime will go down as voice input increases, but button-related input will have a much more minor effect on the battery life. | Button usage (% of time per minute), voice input usage (% of time per minute), TRAVIS voice responses enabled/disabled, TRAVIS voice input receiving/disabled. | Button usage @ 0%, voice input usage @ 10%, TRAVIS voice responses enabled, TRAVIS voice input receiving. | Duration batteries last before there is not enough power to continue functioning. |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions | Procedure: Data Collection Method |
|---|---|---|---|---|---|---|---|
| Provide automated voice and button input at provided levels of duration per minute with given voice settings. Have TRAVIS log every 10 minutes so that when TRAVIS runs out of battery we will be able to find within a reasonable amount of time how long it lasted. | By keeping this process automated we can have these tests running concurrently with other production and not need to worry about constantly checking in on it and thus wasting the time of our team. | Due to the duration of these tests, only a single sample per delta of independent variables will be taken, as these batteries may last a long amount of time and already have prescribed tolerances we can assume they will adhere to. | Upon TRAVIS power off, plug into stable power supply and copy the total time TRAVIS' log reports running for. | 1 | 0: Store TRAVIS within sound-dampened container nearby TV, ensure fresh batteries have been supplied to TRAVIS.<br>1: Set up laptop next to TRAVIS to input given settings and automated inputs.<br>2: Turn on TRAVIS and begin sending in signals via laptop.<br>3: Occasionally check in on TRAVIS to check when it runs out of power.<br>4: When TRAVIS runs out of power, hook TRAVIS up to a stable power supply and collect data from its timing log.<br>5: Repeat with different independent parameters. | TRAVIS to be kept within sound-dampened container so as to minimize audio interference for others using the same office space. | Data collection to occur by reading log file output by TRAVIS, then recording the duration it was on for within this spreadsheet. |
| Provide automated voice and button input at provided levels of duration per minute with given voice settings. Have TRAVIS log every 10 minutes so that when TRAVIS runs out of battery we will be able to find within a reasonable amount of time how long it lasted. | By keeping this process automated we can have these tests running concurrently with other production and not need to worry about constantly checking in on it and thus wasting the time of our team. | Due to the duration of these tests, only a single sample per delta of independent variables will be taken, as these batteries may last a long amount of time and already have prescribed tolerances we can assume they will adhere to. | Upon TRAVIS power off, plug into stable power supply and copy the total time TRAVIS' log reports running for. | 1 | 0: Store TRAVIS within sound-dampened container nearby TV, ensure fully charged battery has been supplied to TRAVIS.<br>1: Set up laptop next to TRAVIS to input given settings and automated inputs.<br>2: Turn on TRAVIS and begin sending in signals via laptop.<br>3: Occasionally check in on TRAVIS to check when it runs out of power.<br>4: When TRAVIS runs out of power, hook TRAVIS up to a stable power supply and collect data from its timing log.<br>5: Repeat with different independent parameters. | TRAVIS to be kept within sound-dampened container so as to minimize audio interference for others using the same office space. | Data collection to occur by reading log file output by TRAVIS, then recording the duration it was on for within this spreadsheet. |

# Battery Charging Speed

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | BSOE Lab 250/252 | | Testing charging speed with wireless charger | wireless transmitter/receiver, LIP battery and breakout board, stopwatch connected to state that says battery fully charged | | wireless transmitter/receiver, LIP battery and breakout board | | charging time |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) |
|---|---|---|---|---|---|
| place fully depleted LIP battery on charging station and time how long it takes for the battery to be fully charged | Determining how long it takes for remote to reach a full charge | | | | 1) setup up wireless charging station<br>2) place battery on charging station<br>3) wait until full charge is reported, write down time |

# Audio Recognition Accuracy

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | Closet or quiet room | | Testing audio recognition without background noise | remote control hardware | Remote will be able to interpret most voice commands without background noise | hardware, voice input | | Recognition quality (i.e. what words were picked up and what were not) |
| 2 | | BSOE Lab 250/252 | | Testing audio recognition with background noise | remote control, TV | Remote will be able to interpret some voice commands with obstacle of background noise | hardware, voice input | | Recognition quality (i.e. what words were picked up and what were not) |
| 3 | | BSOE Lab 250/252 | | Testing voice recognition accuracy for different voices (young, old, teenager, young adult, middle-aged) | remote control, voice recordings | Remote will have difficulty interpreting slow speech and accents | Voices, electronics used for picking up audio | Age ranges of the voices, speed & volume at which they are talking, accents | Recognition quality (i.e. what words were picked up and what were not) |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions | Procedure: Data Collection Method |
|---|---|---|---|---|---|---|---|
| Have someone speak predetermined phrase into microphone, have display show what the hardware heard and compare that to what was said | Testing if the voice recognition works when there are no external factors affecting its accuracy | Take notes of what words were understood by the system and what words either weren't or were omitted/misheard. | Note accuracy of what was heard vs what was said | 3 tests for each voice recording, have at least 5 example commands, total: 15 samples | 1)Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Have person speak predetermined phrases into microphone<br>3) Examine results of what was heard vs what was spoken, compare results | N/A | Google doc |
| Have someone speak predetermined phrase into microphone, have display show what the hardware heard and compare to what was said | Testing reliability of voice recognition with background noise. Is voice recognition reliable with other people in the room? With someone making noise in another room? | Take notes of what words were understood by the system and what words either weren't or were omitted/misheard. | Note accuracy of what was heard vs what was said | 3 tests for each voice recording, have at least 5 example commands, total: 15 samples | 1)Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Have person speak predetermined phrases into microphone<br>3) Examine results of what was heard vs what was spoken, compare results | N/A | Google doc |
| Have different voices play into the remote at differing speeds and volumes. Document results | Testing if the voice recognition is biased toward any voice. Testing how reliable voice recognition is for different voices (high-pitched, low-pitched, slow, fast, stilted) | Take notes of what words were understood by the system and what words either weren't or were omitted/misheard. | Report accuracy compared to a young adult talking at an average pace | 3 for each speed per voice type, accent, etc. | 1) Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Have tester speak into the microphone with a specific accent, vocal age type, speed, etc.<br>3) Examine results compared to what was spoken<br>4) Compare accuracy of results to a control group (i.e. young adult talking at moderate pace)<br>5) Repeat steps 1-4 for different age groups, accents, speeds, etc. | N/A | |

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 4 | | BSOE Lab 250/252 | | Testing volume range remote can accurately respond to | remote control, voice recordings | Remote will have trouble with quiet voices | voices, electronics used for picking up audio | | Recognition quality (i.e. what words were picked up and what were not) |
| 5 | | BSOE Lab 250/252 | | Testing which member/s of the group the remote responds to regularly | remote control, voice recordings of group members | Remote will respond more accurately to one member's voice over the others | voices, hardware | | Recognition quality (i.e. what words were picked up and what were not) |
| 6 | | BSOE Lab 250/252 | | Testing accuracy of responses to questions from user | remote control, voice recording or live input | Remote will have trouble responding to question if it varies too much from expected input | voices, hardware | | Recognition quality (i.e. what words were picked up and what were not) |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions |
|---|---|---|---|---|---|---|
| have the same audio clip played at different volumes, document difference in what was heard vs what was said | Testing how the remote handles a range of input volumes. Does the remote have trouble understanding very loud voices and very quiet voices? How does accuracy diminish with distance? | Note what words were understood by the system and what words were omitted/misheard | Note accuracy of what was heard vs what was said | 3 for each volume, test at quiet, medium, and loud, total = 9 | 1)Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Play audio clips at different volumes<br>3) Document what was heard vs what was said<br>4) Repeat test with different types of voices at different volumes | N/A |
| play audio clip of each member's voice saying the same phrase, document which voice was recognized most accurately | Finding a voice that the remote will reliably respond to. Testing to see if accuracy is affected by masculine or feminine voices. | Play audio clips, document what was heard vs what was said | Note accuracy of what was heard vs what was said | 3 for each member, 21 in total | 1) Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Play audio clip of each member's voice saying the same phrase<br>3) Document what was heard vs what was said<br>4) Repeat as necessary | N/A |
| play audio clips of variations of a question, document how remote responds | Testing to see if the remote can recognize a question/command and respond correctly and how program reacts if question isn't perfect | Play audio clips, document program response | Note accuracy of response to question | 3 for each audio clip, 3 variations of question, at least 9 | 1) Prepare voice recognition software and hardware with a program to purely translate audio into text<br>2) Play variations of audio input<br>3) Document how the program responded, record whether met expectation or not<br>4) Repeat as necessary | N/A |

# Buttons

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | Testing button durability | Buttons that will face a variety of different hits with different pressures, timings, and durations (i.e. being pressed repeatedly for n minutes) | The buttons should be able to withstand a multitude of different presses and should last for a good bit of time | Button types, pressure and duration of presses and how often they will be pressed | Varying amounts of force, duration, and on & off periods<br>Soft tactile buttons | Damage sustained and usability of the buttons after rigorous testing<br>Amount of bouncing present afterwards |
| 2 | | | | Testing button contact wear and tear | | | | | |
| 3 | | | | Testing button reactivity/response time | | | | | |
| 4 | | | | Testing accuracy/reliability of response to button input | | | | | |
| 5 | | | | Testing "ease of use" of buttons | | | | | |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions |
|---|---|---|---|---|---|---|
| Have one person apply various amounts of pressure and for different amounts of time<br>Note durability and usability damages | This should show how much the buttons can take before they stop working either partially or altogether. It will allow us to figure out if we need to get more durable buttons or if we need to come up with more protection for said buttons | Button tested manually after each test. Check functionality and effectiveness based on factory new | Hook up button to program that allows us to press the button and test its functionality. Have a factory new button ready for comparison between it and the freshly tested one. Document results | 3 timings for each level of pressure | 1) Have button disconnected and ready to be pressed<br>2) Time the duration of the press and have the participant use as specific of an amount of pressure as possible<br>3) Release button and begin analysis<br>4) Compare to factory new button for feeling test<br>5) Hook up to program to compare functionality to factory new button. See if extra bouncing is present<br>6) Repeat steps 1-5 with varying times and pressures | Make sure the button is completely disconnected from all electronics before beginning testing in order to prevent electrocution as well as material waste |

# Television Remote Control

| Test # | Date | Location | Members Present | Goal/Purpose | Parameters | Hypothesis/Expectations | Independent Variables | Independent Variables' Settings | Dependent Variables |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | Testing ability to communicate with different TVs via IR | IR-sending hardware and software | The IR system should be able to handle multiple TV systems | IR hardware components & software libraries; TV being used | TV make and model | Accuracy of the signals sent on different televisions |
| 2 | | | | Testing how reliable communication with one TV set is via IR | IR-sending hardware and software | The IR system should be reliable on a single TV system | IR hardware components & software libraries; TV being used | TV make and model | Accuracy of the signals sent |
| 3 | | | | Test time it takes to configure the remote w/ a smart TV | | | | | |
| 4 | | | | Test time it takes to configure the remote w/ a regular TV | | | | | |
| 5 | | | | Test time it takes for an unassociated person to confiure the remote | | | | | |

| Testing Method | Testing Method: Significance | Sampling Procedure | Sampling Procedure: Method | Sampling Procedure: # of Samples | Procedure (Step-by-step) | Procedure: Safety Precautions |
|---|---|---|---|---|---|---|
| Have one person send various different IR signals to different TVs and see how many work correctly | This should accurately tell us how well the IR-sending library works and whether or not the hardware functions for multiple TV makes and models | Each IR signal manually tested. Functionality should be compared with proper TV remote | Send one IR signal at a time and check for what happens on the TV, if anything. Document results | Test each IR signal. Try 3 times for each. Try 3 different televisions | 1) Have hardware and software connected.<br>2) Send IR signal to television<br>3) Analyze results and compare to factory new remote<br>4) Repeat steps 1-3 multiple times for all IR signals available on the remote<br>5) Repeat steps 1-4 for different TVs | N/A |
| Have one person send various different IR signals and see how many work correctly | This should accurately tell us how well the IR-sending library works and whether or not the hardware functions | Each IR signal manually tested. Functionality should be compared with proper TV remote | Send one IR signal at a time and check for what happens on the TV, if anything. Document results | Test each IR signal. Try 3 times for each | 1) Have hardware and software connected.<br>2) Send IR signal to television<br>3) Analyze results and compare to factory new remote<br>4) Repeat steps 1-3 multiple times for all IR signals available on the remote | N/A |

# Test Results

Audio Recognition:

Through rigorous testing, we found that Google's online speech-to-text worked the best in both close and far ranges, as well as with a wider range of voices. While Sphinx is functional and a well-optimized alternative when not connected to WiFi, it did not hold up with the older range of voices tested. Regarding distance, Google still held up better but Sphinx did not fall far behind; both performed relatively well in this area. On the board itself with a USB microphone, the speech recognition was not as accurate due to the quality of the audio being passed through, as well as some issues with speech-to-text latency that were also present in the test files. This, in turn, made it more crucial to use Google's online versus Sphinx's offline speech recognition.

Button Inputs:

Our button inputs worked relatively flawlessly from the beginning and required little testing. We just had to press each button once and make sure that the TV received the correct action or it was at least logged to the console. We also tried things like pressing a button as rapidly as possible or for varying periods to check edge cases, but due to the simple nature of buttons, nothing caused any issues.

Television Communication:

Interfacing with the IR LED involved running the LIRC daemon in the background, and then communicating with it from our program to send TV remote codes. LIRC was a bit finicky to set up, but we could easily test that the daemon was up and running and had the correct output pins by connecting a regular LED and checking it for flashes when sending a code. Once it was set up properly it never really gave any more issues. We also tested the limitations of the IR communication between our device and the TV by pressing buttons while holding the remote at various angles and distances. We found that with our transistor setup to power the LED, it gave about the same performance as a regular TV remote. It of course had the same limitations as well, requiring line of sight and relatively accurate aiming of the remote.

# Review

**Lillian**

What went well was that our research at the start, despite some hurdles getting everything together, actually worked out pretty well - basically, all of the technologies we researched in the first two weeks were what we actually used later down the line, we didn't need to ditch a software halfway through our prototype. Also, our CAD and Docker were handled super well and I'd like to think I did a good job making sure everyone was on track, though I could have done a bit better to remember our meeting times and give a little less slack after big due dates. On the other side of things, there was a lot of time during the project when everyone was busy with other things and we weren't getting a lot of work done, between one of our team members being stuck in mechatronics and another busy with personal matters, we were down a lot of our programmers and integration seemed like a far off nightmare. I'm not sure how to have handled that situation better, as it's not like I can give people an extra eight hours a day to jam onto our prototype, but perhaps I could have made sure that all of our team members were more communicative of when they would be out and when they would be back, as well as to make sure that in the final weeks that our prototype wasn't being hogged by one person, or that we had enough parts for a second prototype for other testing and integration work.

**Ann Sophie**

Overall looking back on the last two quarters I think that our project was well within the scope of this class. It was not too easy but also not so complicated that it could not be completed within 5 months. In the first quarter, we were really on top of things, having multiple long meetings a week where we all contributed to the assignments at hand. I do think that we could probably have started working on our prototype in the first quarter however it proved difficult with everyone's schedules even going into the second quarter. Since we are all on different tracks for graduation completing various different classes conflicting with each other's availability it proved difficult to get together and work on the prototype since most of the work really had to be done in person. I, unfortunately, had a lot of personal things going on during the second quarter which made it difficult to be there for my team. If I could do it again, given different circumstances I would love to contribute more to the actual prototype. My strengths are not in software and I would say I'm better at hardware but there were already too many people dealing with the hardware. My lack of experience in software really challenged me to work on that side of things. English is also not my first language which can make it difficult at times to try and communicate effectively.

**Katy**

In our first quarter of senior design, our team did well getting together early on and brainstorming ideas. If I were to do the first quarter over again, I would be more proactive about stepping back and encouraging other team members to write parts of the documentation. There were times when I would complete parts of assignments that I should have left to other team members. In our second quarter, I spent a lot of time doing Mechatronics and subsequently was barely available. My team was very accommodating and did a great job putting together a prototype while being down 2 team members consistently. If I were to do things differently I would carve out more time in my schedule to learn more about the software we used and to tidy up the breadboard wiring for the remote.

**Aidan**

In the first quarter, I think we did a pretty good job meeting our goals and finishing our deadlines. However, I wish I took a more proactive role in getting things done early. In the second quarter, I took on the responsibility of the CAD part of the design which was quite the learning experience. Fusion 360 was hard to learn at first but by the end of the quarter, I feel pretty proficient in it. That being said, I would have liked to do more of the coding side of things if I were to start over. I think the coding of the device is the most interesting and important part. I tend to enjoy my coding assignments so I think I could have had a lot of fun doing that.

**Heath**

I think that we did a good job overall of delivering on our goals. Even though some ideas had to be left out and our final product could have used a bit more polish, we did create a prototype that works well as a fully functional remote and can understand and interpret voice commands pretty reasonably. My role at the beginning was to create the CAD model since I had a 3D printer and had used Fusion360 before. After a while, Aiden took over the CAD (though I kept printing), and I moved more towards the infrastructure (? not really sure what to call it but setting up Balena to interact with the Pi remotely as well as Docker to keep a consistent environment for running our program). In hindsight, we never ended up having multiple Pis or breaking our current one, so I'm not sure how much it actually contributed to our project besides correctness and peace of mind. We were definitely a lot more active as a team for the first quarter, with regular meetings and solid communication and direction. In the second quarter we were a bit less focused, but also got more work done on the programming/wiring/assembly side (probably just because of how projects tend to develop though). I personally had a lot harder classes this quarter, and I know a few other members of the team had even more important stuff to worry about. If I were to do this project again, I would try to devote more energy to communicating as a team. I'm generally not the best communicator, but I think it would have made our project more cohesive and consistent if we spent more time talking about our choices and thoughts when working on our individual parts. If we had more time, I would have liked to assemble a prototype in the actual casing and maybe make the voice recognition have some sort of fuzzy matching from a list of options without transcribed text as an intermediate layer.

**Nathan**

I feel as though we did an excellent job finding the resources we needed to bring this project to life, such as the libraries and hardware necessary. I'm really glad we stuck to what we had and followed through with the project; it was overall fun to test and work with. The workload was mostly distributed evenly throughout the first quarter (though it became less so in the second), so I never felt like I was taking on too much at a time. If done differently, I wish I could have gotten more work done on the code (although this would require having more extensive knowledge of the Python coding language, which I do not). I also wish we had more time to test out the speech recognition on the board so we could have worked out more of the kinks, such as the latency between when the button is pressed and when it starts listening. Finally, I would like to have implemented the Deep Speech recognition software that can be trained rigorously; it was more functional and was able to look for specific words rather than interpreting everything, which would have allowed for greater accuracy and a wider range of accepted voices between distance, accents, and age.

**Kanybek**

I felt that the design process and division of labor worked well in the first few weeks of the quarter. I was in charge of handling the hardware setup for our prototype, which I was a bit slow to do initially, though once the ball was rolling I managed to get it set up. The main issues that arrived were when dealing with software hiccups, with missing libraries and drivers that were not intuitive to fix (due to not having in-depth experience with python), so I had to work on the software side of things as well since I had the prototype on hand most of the time. Our use of Docker was also challenging, and if I could have done it differently, I wish I would have gotten the hardware setup sooner and without Docker initially, in order to identify the issues sooner. The lack of time left towards the end meant that we had to forgo the rechargeable battery approach in our functional prototype demonstration, and that our microphone choice led to inaccurate voice-recognition sometimes.