# Guidance Glasses Final Presentation

# Table of Contents

01

# Project Overview

# Problem and Existing Solutions

Current cityscapes are not designed to accommodate the visually impaired. Cityscapes are filled with multiple obstacles that move in and out of people's' paths, making it a dangerous environment to navigate. Current solutions still have many issues:

- Traditional white canes are unable to detect objects that are outside of the user's immediate range
- Current smart canes are unable to recognize objects and are only able to measure distances
- Existing smart glasses have high costs and are unable to measure distances
- Crosswalks still remain one of the most dangerous places for the visually impaired to traverse

# Scope and Approach

The goal of the guidance glasses is to **assist visually impaired people** in being able to **safely cross the street**. The glasses use a combination of information provided by a camera, and microphone to assess whether it's okay to cross the street. The ultrasonic provides additional information about obstacles in front of the user. This is then communicated to the user through vibrations from haptic feedback motors.

- The camera uses a custom-trained YOLOv5 model to detect and recognize pedestrian stop or walk signs.
- The microphone listens to sounds that might come from pedestrian walk and stop signals to add to the algorithm's confidence level on if it is a walk signal or not. It also measures the decibels of ambient sounds to determine if there is a car.
- The ultrasonic detects any objects within its range and how far they are to determine if an obstacle is in the way.
- Haptic feedback motors located on both temples then communicate all of this through vibrations of varying frequencies and intensities.

# Clarifications

- No longer using the Jetson Nano, Adafruit I2S MEMS Microphone, and IMX219-160 Camera in demonstration
  - Issues with interfacing components
- No gated recurrent unit for determining weights
  - Shifted focus to individual modules and main integration
- Final demonstration does not have PCB
  - Shipping - delayed in transit

# 02

# Hardware

# Frame and Temple

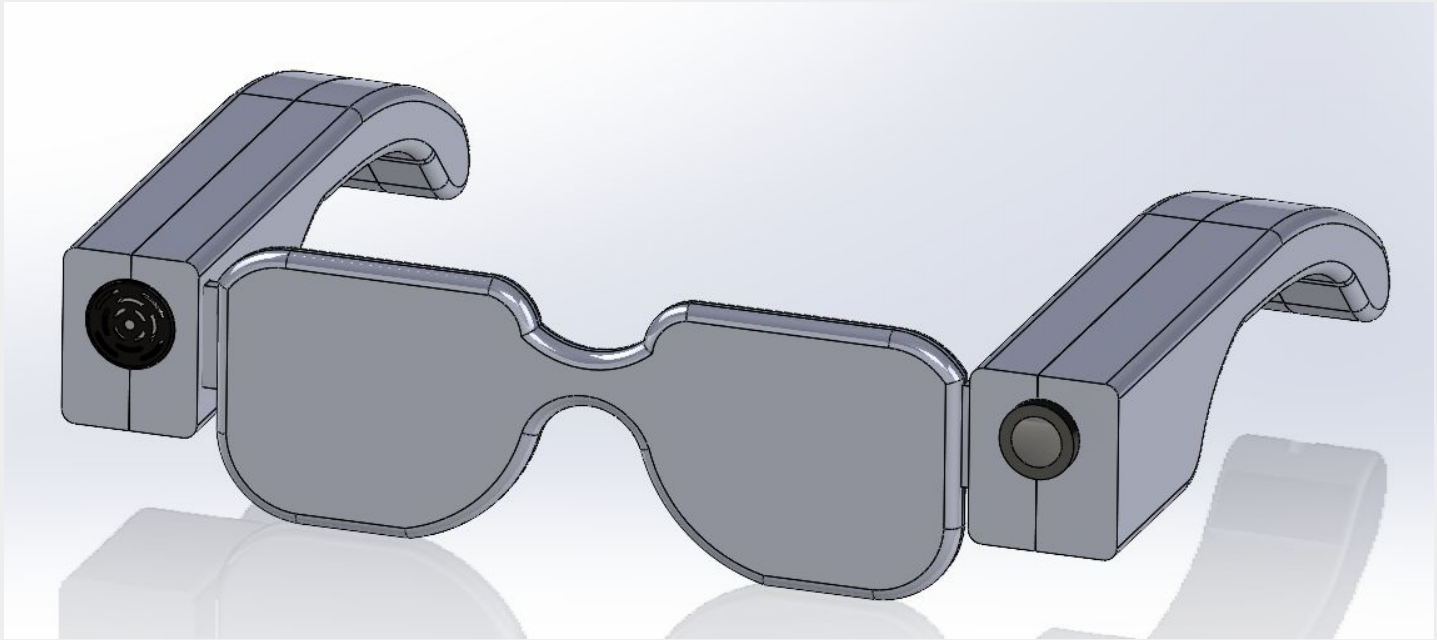


Figure 1: Full Assembly of Frame and Temple

# Frame and Temple

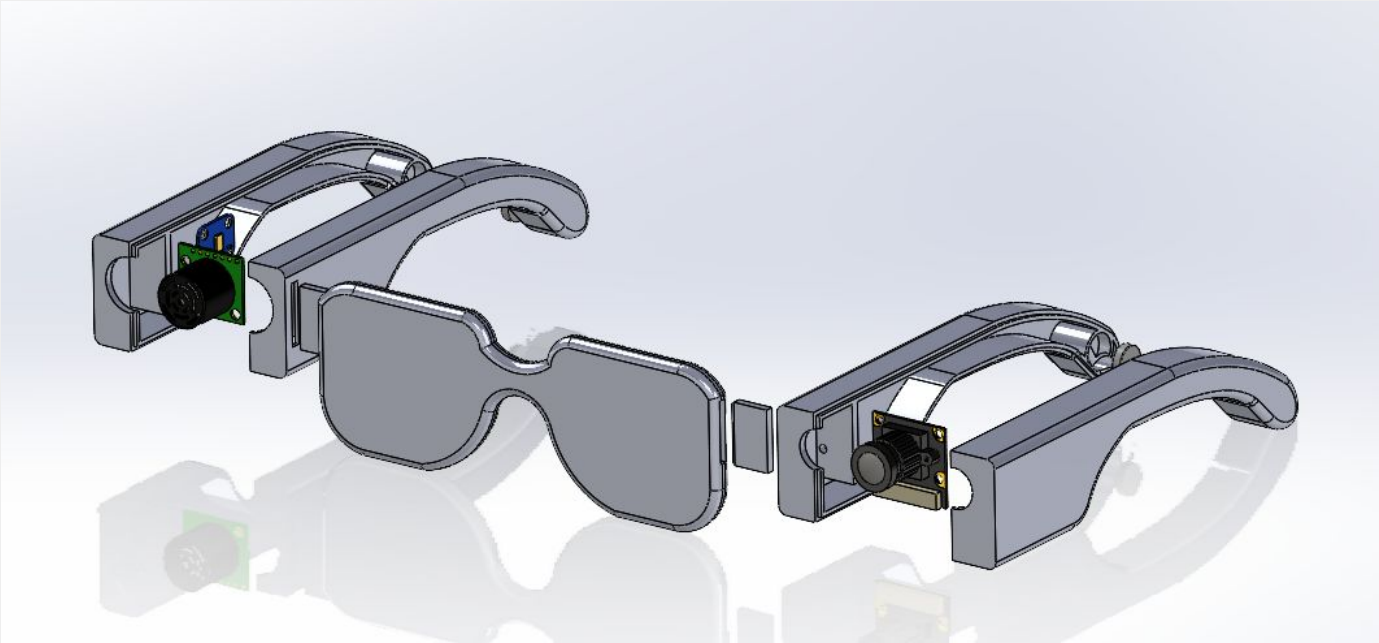
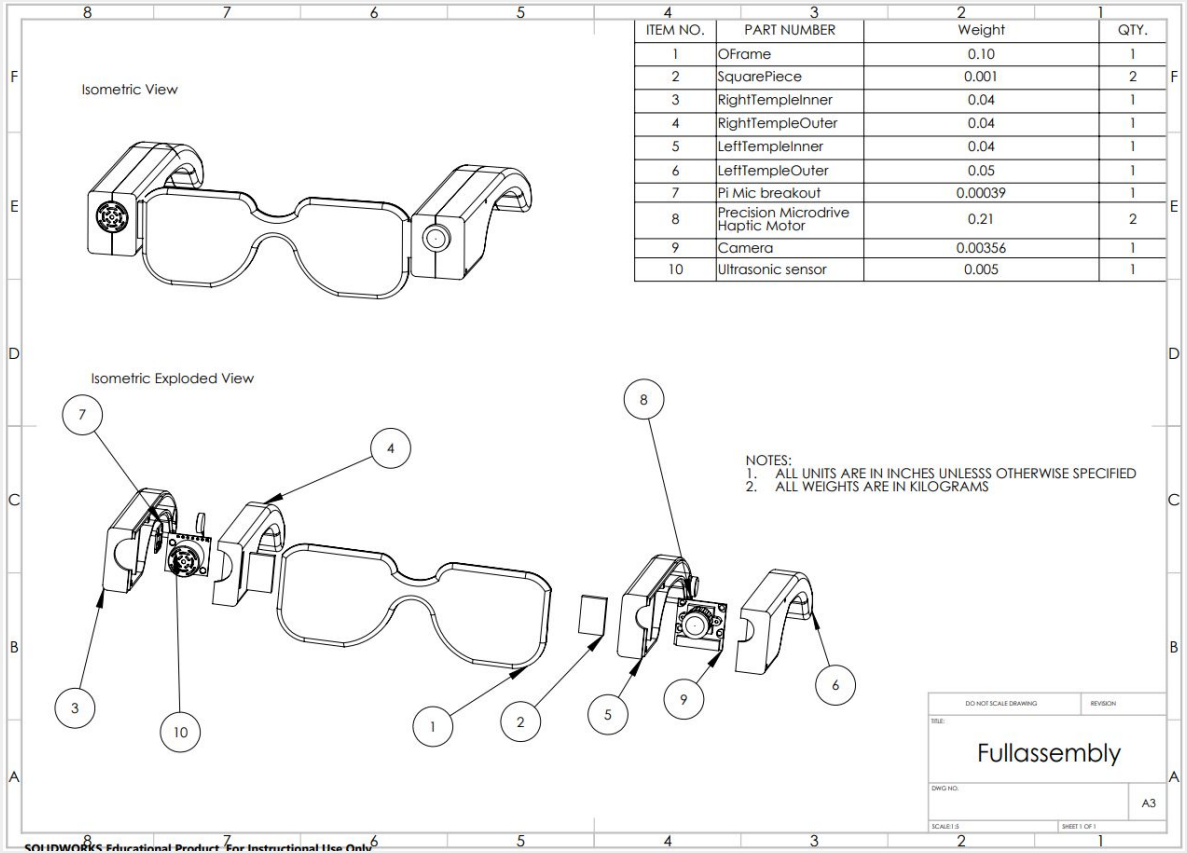


Figure 2: Exploded View of Frame and Temple

# Frame and Temple




| ITEM NO. | PART NUMBER | Weight | QTY. |
|---|---|---|---|
| 1 | OFrame | 0.10 | 1 |
| 2 | SquarePiece | 0.001 | 2 |
| 3 | RightTempleInner | 0.04 | 1 |
| 4 | RightTempleOuter | 0.04 | 1 |
| 5 | LeftTempleInner | 0.04 | 1 |
| 6 | LeftTempleOuter | 0.05 | 1 |
| 7 | Pi Mic breakout | 0.00039 | 1 |
| 8 | Precision Microdrive Haptic Motor | 0.21 | 2 |
| 9 | Camera | 0.00356 | 1 |
| 10 | Ultrasonic sensor | 0.005 | 1 |

Figure 3: Full Assembly Drawing

# O-Frame

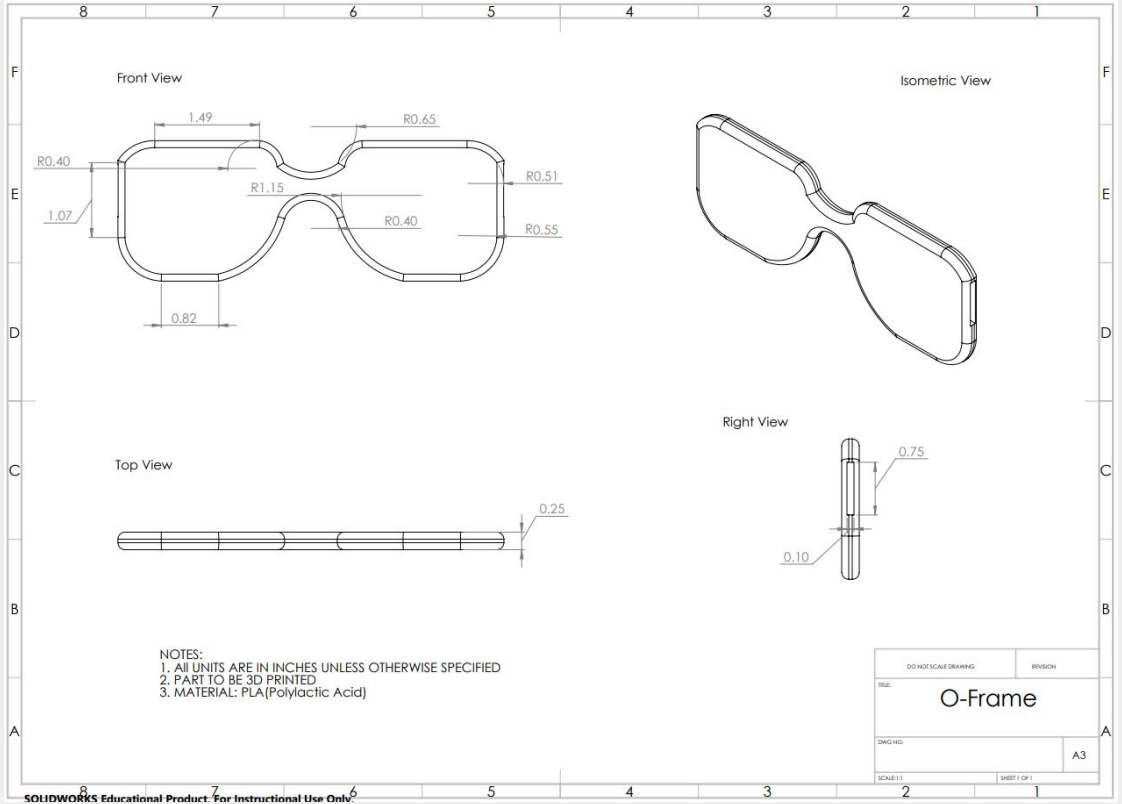



Figure 4: O-Frame Drawing

# Left Temple Outer and Inner

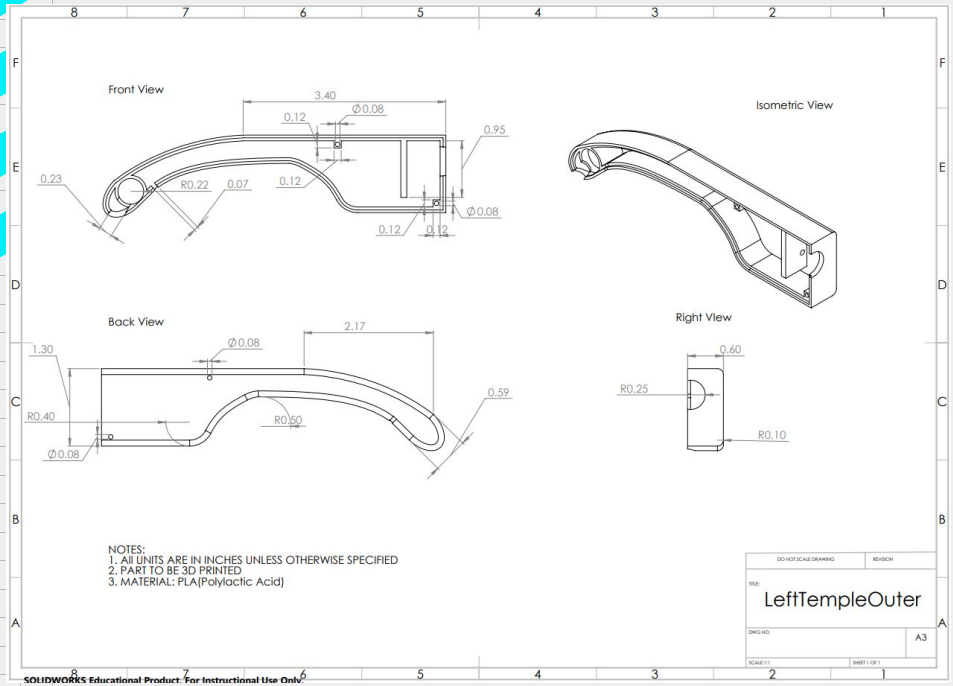



Figure 5: Left Temple Outer

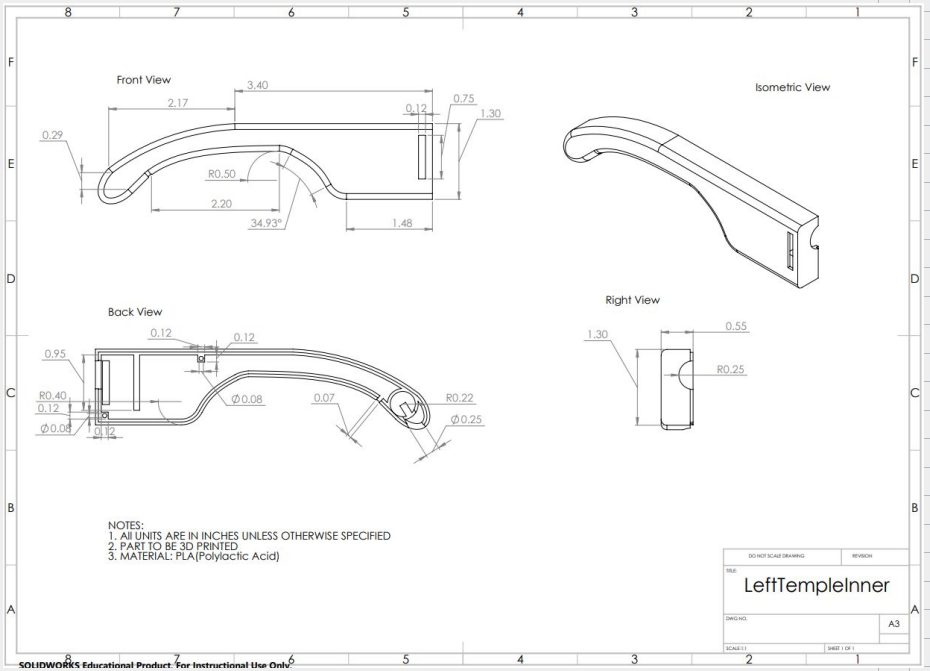



Figure 6: Left Temple Inner

# Right Temple Outer and Inner

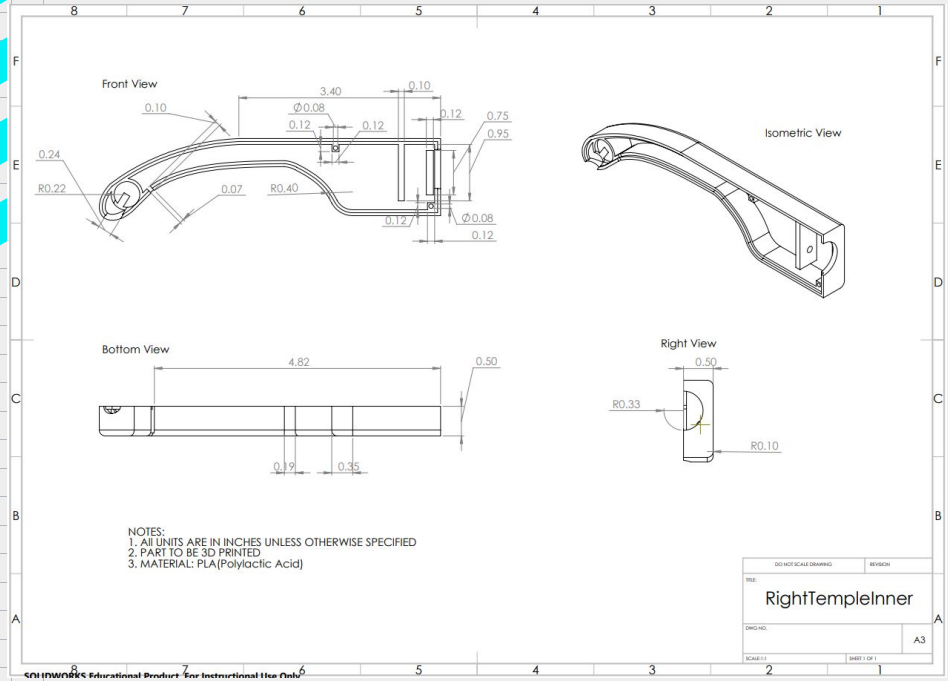



Figure 7: Right Temple Inner

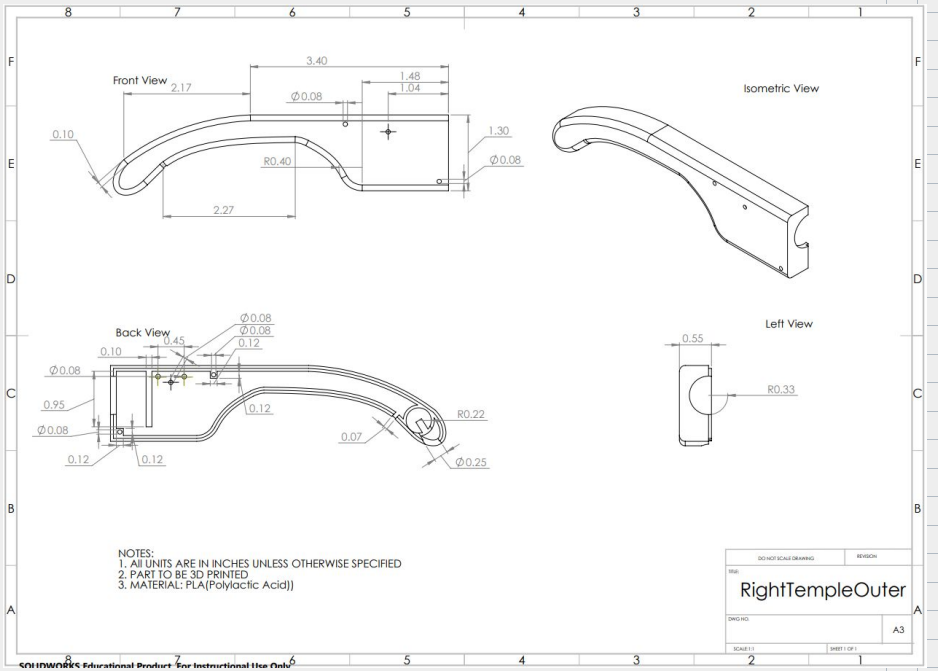



Figure 8: Right Temple Outer

# Assembly Drawing

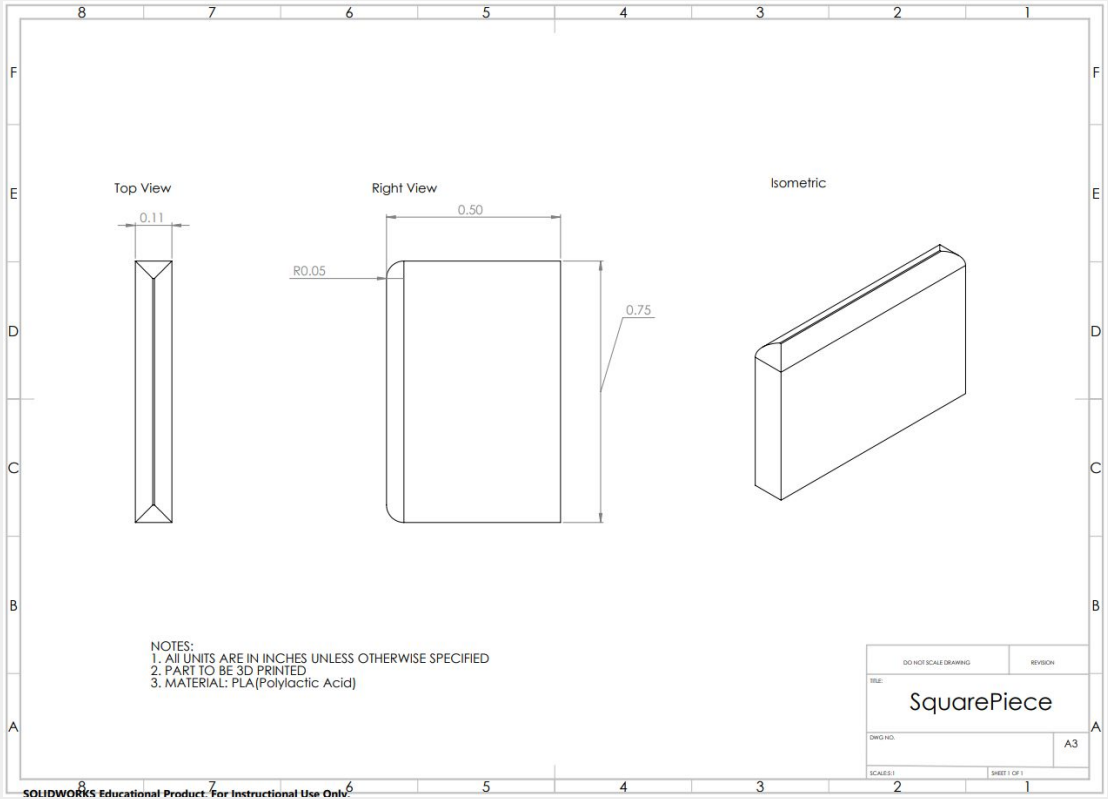



Figure 9: Square Piece

# Jetson Nano and PCB Housing

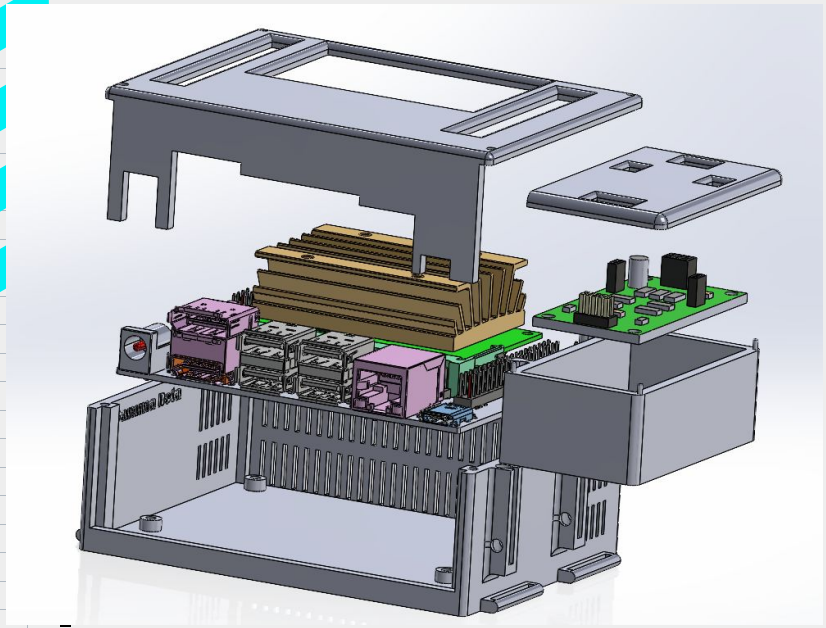


Figure 10: Exploded View of  Jetson Nano and PCB Housing

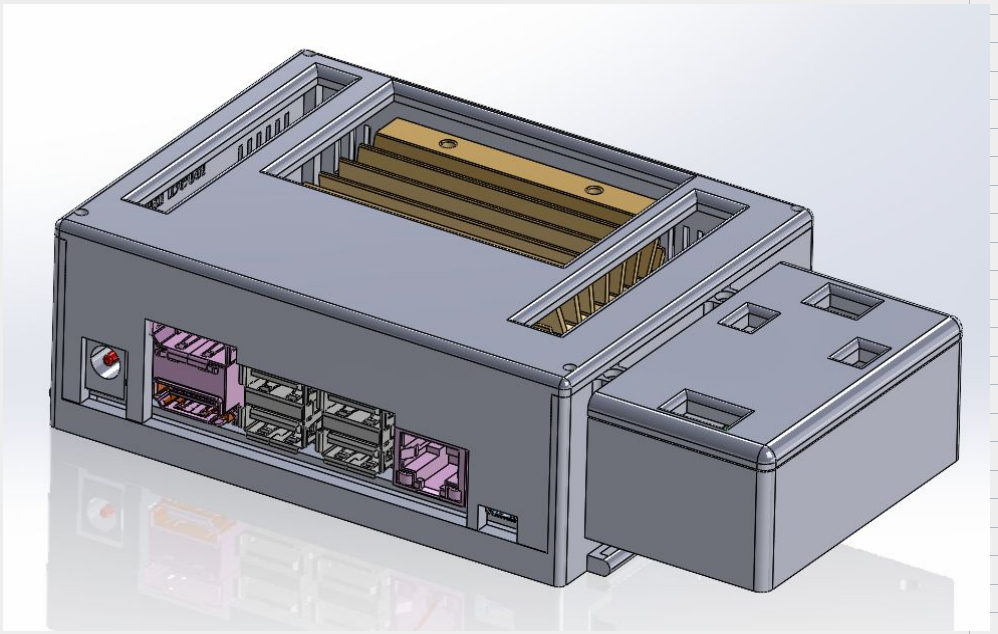
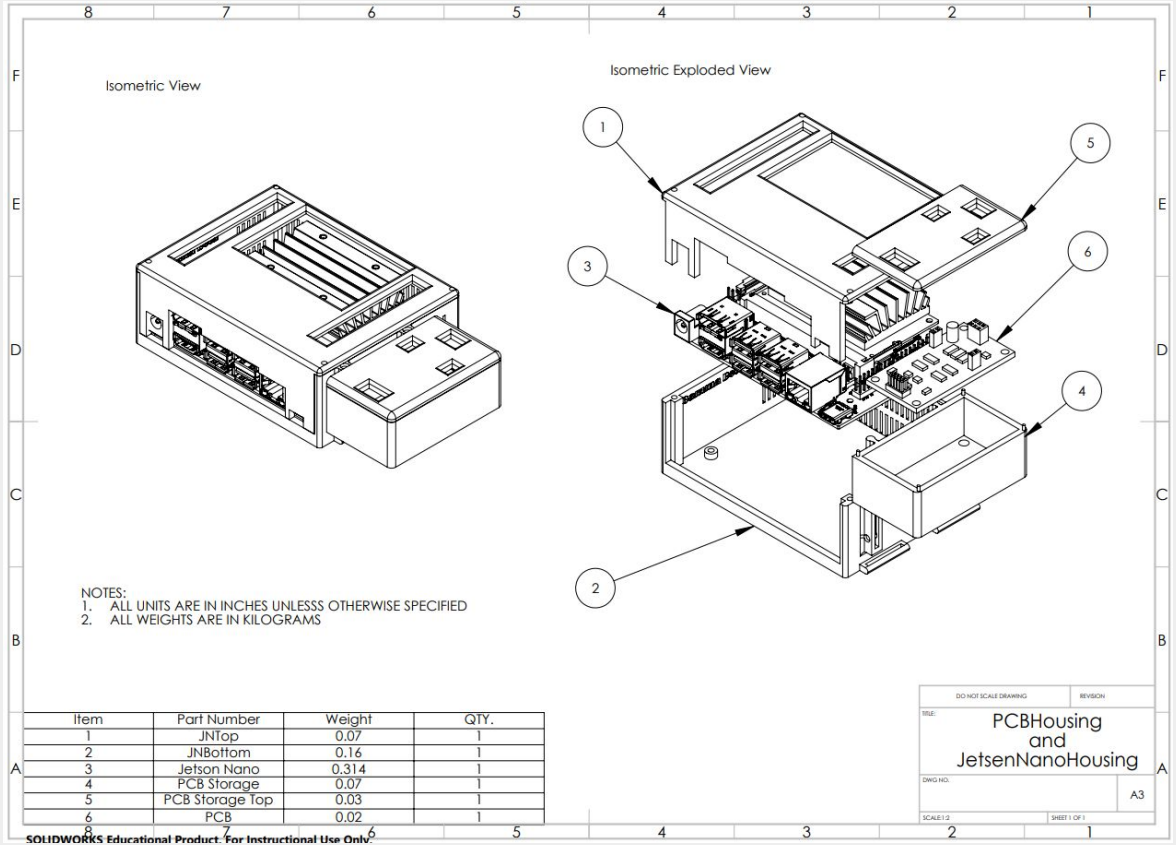


Figure 11: Assembly of Jetson Nano and PCB Housing

# Jetson Nano and PCB Housing




| Item | Part Number | Weight | QTY. |
|---|---|---|---|
| 1 | JNTop | 0.07 | 1 |
| 2 | JNBottom | 0.16 | 1 |
| 3 | Jetson Nano | 0.314 | 1 |
| 4 | PCB Storage | 0.07 | 1 |
| 5 | PCB Storage Top | 0.03 | 1 |
| 6 | PCB | 0.02 | 1 |

Figure 12: PCB Housing and Jetson Nano Housing Drawing

# Updated Jetson Nano Housing

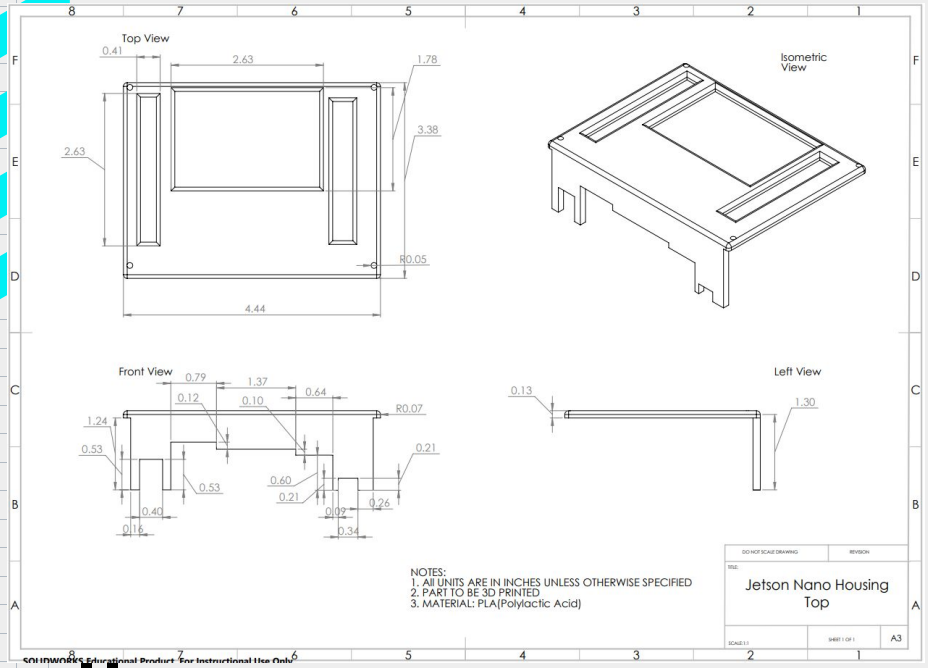



Figure 13: Jetson Nano Top Housing

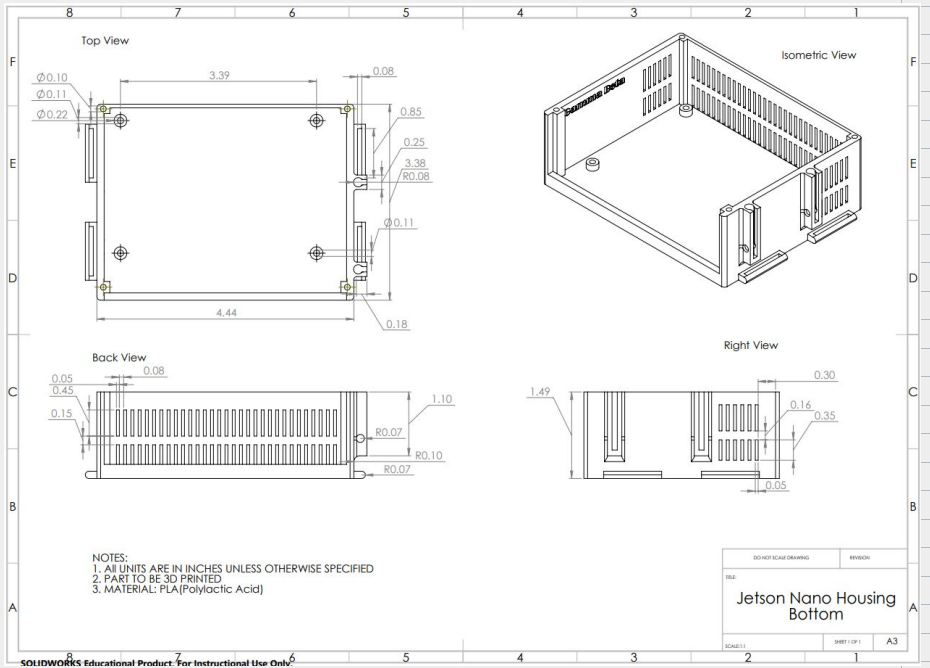



Figure 14: Jetson Nano Bottom Housing

# PCB Housing

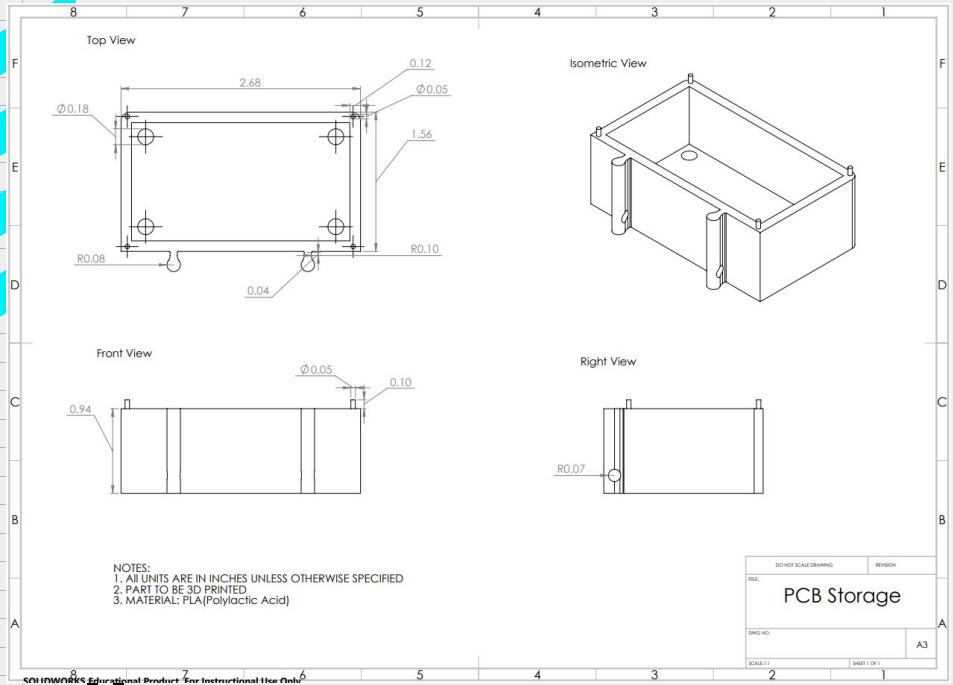



Figure 15: PCB Bottom Housing

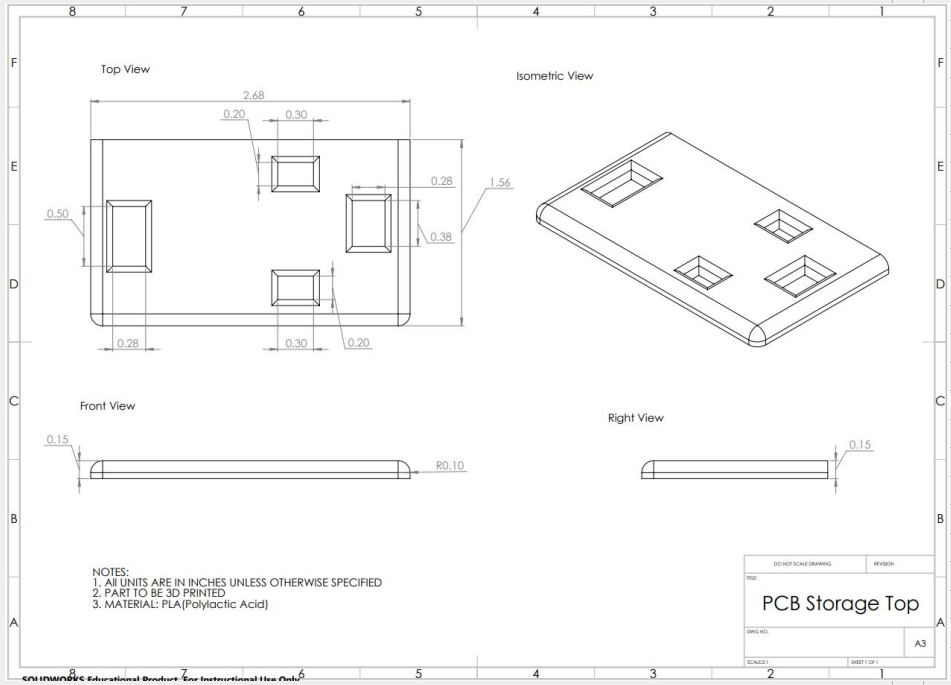



Figure 16: PCB Top Housing

# Temple and Component Assembly

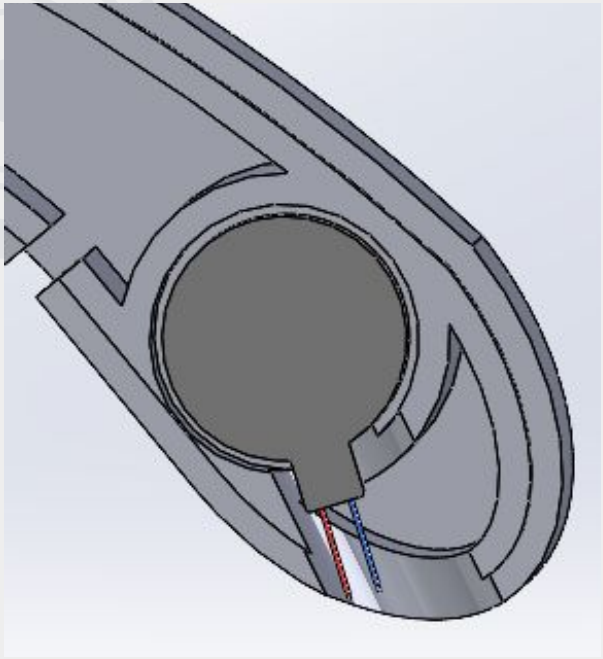


Figure 17: Haptic Motor

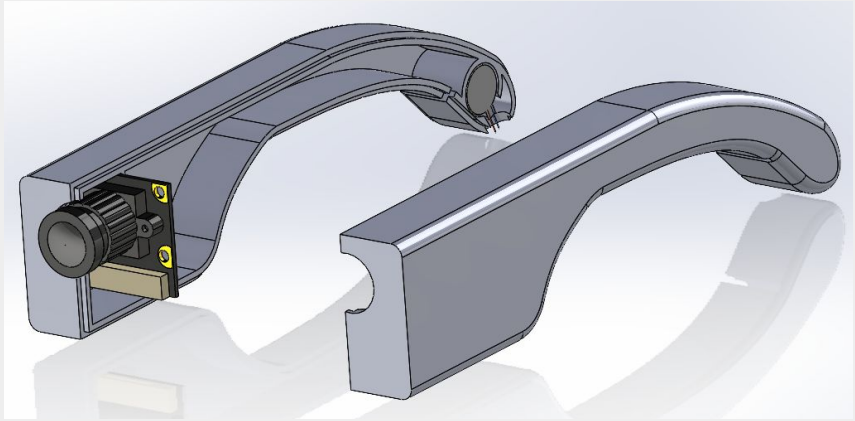


Figure 18: Camera - Exploded

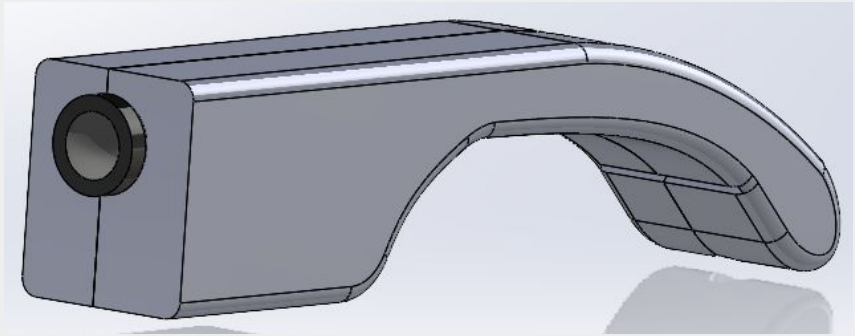


Figure 19: Camera (Left Temple)

# Temple and Component Assembly cont.

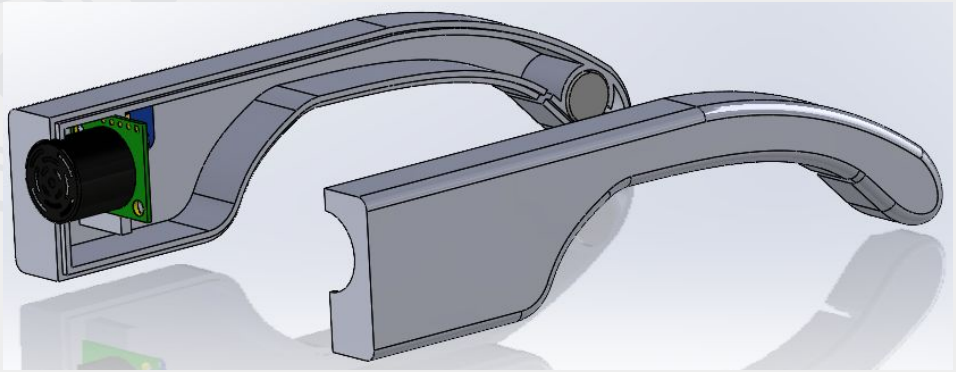


Figure 20: Ultrasonic Sensor- Exploded

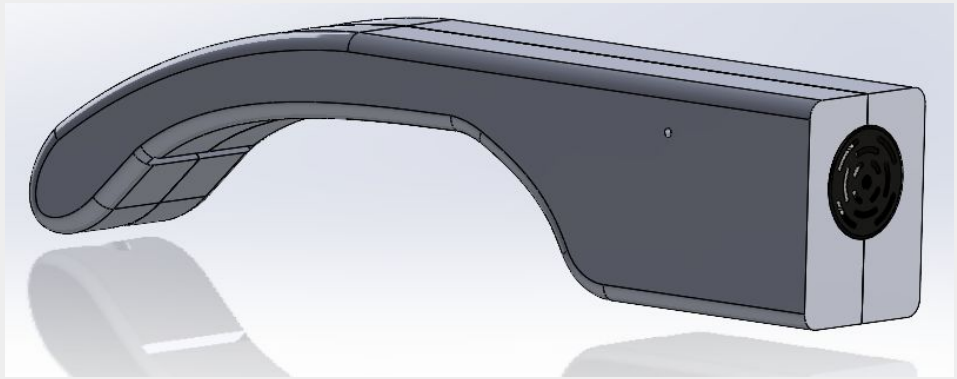


Figure 21: Ultrasonic (Right Temple)

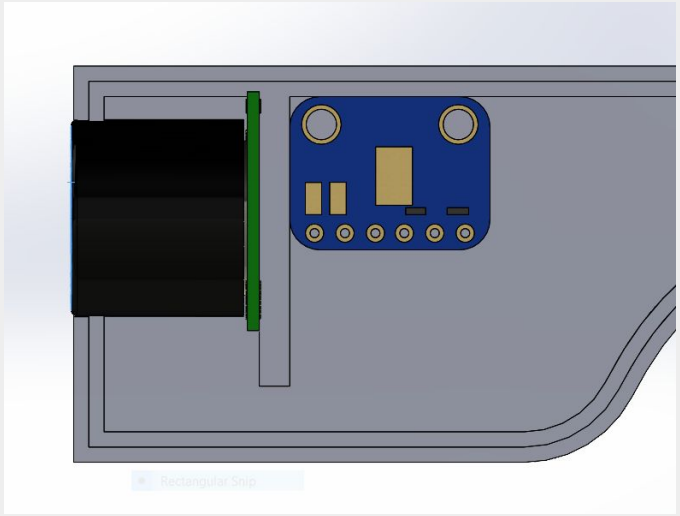


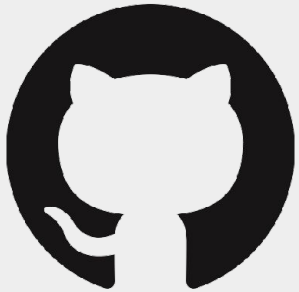
Figure 22: Ultrasonic Sensor and Microphone

# Future Improvements

- Creating temples that sit comfortably on users' ears.
- Experimenting with different filaments
  - Using Chinchilla filaments for parts contact with skin.
  - Using harder filaments for parts that need to be durable.
- Improving housing unit to become more portable
  - Better implementations onto the backpack.
  - More streamlined wiring holes to minimize unnecessary cutouts.
  - Minimizing exposed wires.
- Implementing hinges so that the glasses can be easier stored when not in use.
- Creating different styles frames to suit user's preference.

# 03

# Software Progress



GitHub

https://github.com/timbim1681/gb-glasses

# Camera Pseudocode

```
# Setting up of the model and camera
Load custom YOLOv5 model

SET camera and frame dimensions
SET GPIO pins and sensors

WHILE LOOP:
    READ a frame # Check if empty
        IF empty, skip frame due to mismatch

    SET results of frame capture # check if detected
        IF no detections, print "no detections" and Continue

    Apply model on each frame
    Start calculating a total for confidence interval
        IF the frame is walk:
            Add to total by interval
        IF the frame is don-t walk:
            Add to total by 0
        IF average interval over 20 frames > 70:
            Activate walk feedback
            Shut down camera process

Instead of counting how many frames there are,
Over 20 frames, if average is over 70
```

Figure 23: Glasses.py Pseudocode

# Speech to Text Pseudocode

```
# Updated Microphone Pseudocode

FUNCTION speechToText
    # Converts audioReading from microphone to text
    GET audioReading
    SET audioReading AS textReading

    # If the phrase 'walk sign is on' is heard, mic sends positive feedback
    IF 'walk sign is on' IN textReading THEN
        RETURN positiveFeedback

    # If the phrase 'wait' is heard, mic sends negative(warning) feedback
    ELSEIF 'wait' IN textReading THEN
        RETURN negativeFeedback

    # Audio feedback not heard/contains inconclusive data
    ELSE
        RETURN inconclusive
    ENDIF

ENDFUNCTION
```

Figure 24: speech_to_text.py Pseudocode

- Speech to text
  - Open audio reading from microphone
  - Convert section of audio to recognized text
  - Listens for speech of a walk sign being on to return positive feedback
  - Listens for speech of waiting to return negative feedback
  - If no speech or is just sound, it is inconclusive and will rerun
  - Continues to run until it hears speech indicating of a walk sign

# Decibel Pseudocode

```
23 FUNCTION detectDecibels
24     GET audioDecibels
25 
26     # If the volume is louder than a certain threshold, send negative(warning) feedback
27     IF audioDecibels > certainThreshold THEN
28         RETURN negativeFeedback
29 
30     # Audio volume does not entail threat (inconclusive data)
31     ELSE
32         RETURN inconclusive
33     ENDIF
34 
35 ENDFUNCTION
```

Figure 25: speech_to_text.py Pseudocode

- Decibel Detection
  - Record 3 second snippet of audio in .wav file
  - Calculate average decibels every second
  - If louder over the interval, car or object is moving towards you
  - If quieter over the interval, car or threat is moving away from you
  - Otherwise inconclusive where the object is moving towards

# Ultrasonic + Haptic Motor Pseudocode

```
Initalize pins, adafruit driver, array, and variables

Function Setup
    Initalize haptic motor
    Set pin input
    Initalize serial monitor

Function Sensor reading
    Measure duration of pulse recieved by ultrasonic
    Convert to centimeters
    Store values in array

Function Re-arrange centimeters
    Loop over input array of centimeters to place in ascending order

Function Filter
    Receive array of arranged centimeter values
    Keeps track of occurance of each value, maximum count, and current count
    Returns either mode of data or median if there is no mode/conflicting mode

Function loop [serves as main function]
    Calls on Sensor reading to grab distance values from ultrasonic
    Calls on Re-arrange centimeters to sort values
    Calls on Filter to obtain final centimeter measurement

    If centimeter value is between 400 and 320 centimeters
        cause haptic motor to buzz at 20% intensity

    If centimeter value is between 320 and 240 centimeters
        cause haptic motor to buzz at 40% intensity

    If centimeter value is between 240 and 160 centimeters
        cause haptic motor to buzz at 60% intensity

    If centimeter value is between 160 and 80 centimeters
        cause haptic motor to buzz at 80% intensity

    If centimeter value is between 80 and 0 centimeters
        cause haptic motor to buzz at 100% intensity
```

Figure 26: ultrasonic_Pwm.ino Pseudocode

- Distances values are obtained from sensor
- Values are sorted and filtered for accuracy
- Haptic motor vibrates with varying intensity based on filtered value

# Main Module Pseudocode

```
# Main module

FUNCTION main
    # Call camera function to get the aerage image recognition probabilty
    image_probability = camera()

    # Once it detects a walk sign, parallel calls for camera, ultrasonic sensor,
    # speech to text, and decibel calculator
    SYNC camera
    SYNC ultrasonic sensor
    SYNC speech-to-text
    SYNC decibel calculator

    # All sensors will continue to run throughout the crossing
    CALL camera
    CALL ultrasonic sensor
    CALL speech-to-text
    CALL decibel calculator

    # Only using speech-to-text, change image_probability based off what it has heard
    IF 'walk sign' in speech-to-text THEN
        image_probability = image_probability * 1.25 speech-to-text

    ELSEIF 'wait' in speech-to-text THEN
        image_probability = image_probability - (image_probability * 0.25)

    RETURN walk/don-t walk

ENDFUNCTION

# Call main function
if __name__ == '__main__':
    main()
```

Figure 27: Main.py Pseudocode

- Main module connecting all sensors
- Activates camera first
- Followed by continuous use of camera, microphone, and ultrasonic sensor
- Calculates certainty of crossing using camera and speech to text

# Roboflow

https://universe.roboflow.com/timbim1681/guidance-glasses-v2

# Updated Dataset & Model

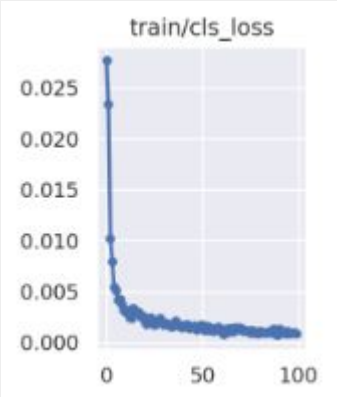



Figure 28.
Class Loss for Train Set (BC)

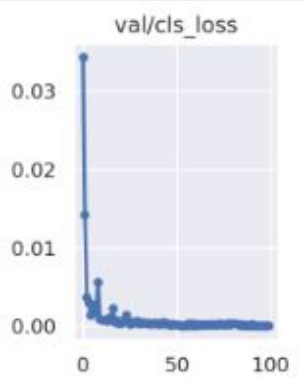



Figure 29.
Class Loss for Validation Set (BC)

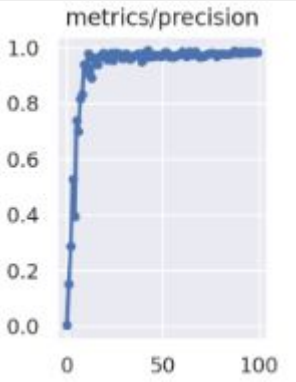



Figure 30.
Overall Precision (BC)

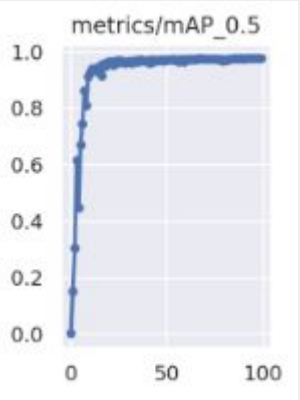



Figure 31.
Mean Average Precision (BC)

Class Loss: measures loss/error associated with object classification
Precision: measures accuracy of positive predictions within model
mAP: Mean Average Precision at an Intersection-over-Union (IoU) threshold of 0.5.
*Common evaluation metric for assessing accuracy & quality of predictions*

# Updated Dataset & Model

```
Validating runs/train/yolov5s_results2/weights/best.pt...
Fusing layers...
custom_YOLOv5s summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.81it/s]
                   all        101        102      0.988      0.921      0.959      0.767
            Don-t Walk        101         57      0.977      0.965      0.968      0.807
                  Walk        101         45          1      0.877      0.957      0.726
Results saved to runs/train/yolov5s_results2
CPU times: user 18.5 s, sys: 1.82 s, total: 20.3 s
Wall time: 25min 49s
```

Figure 32: Data from Original Model

```
Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
custom_YOLOv5s summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 5/5 [00:03<00:00,  1.56it/s]
                   all        131        135      0.982      0.965      0.976      0.803
            Don-t Walk        131         54      0.963      0.981      0.986      0.843
                  Walk        131         81          1      0.949      0.967      0.762
```

Figure 33. Data from Re-trained Model

# Updated Dataset & Model

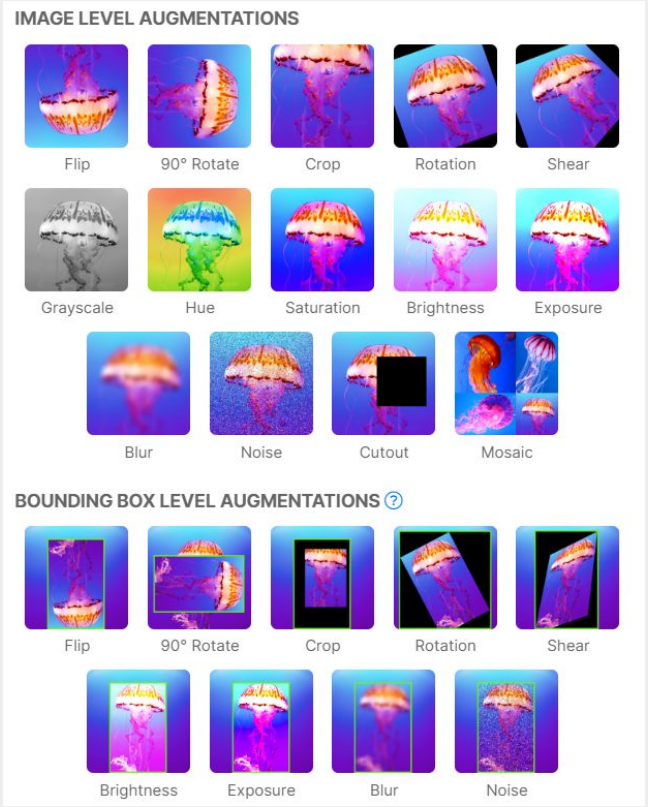



Figure 34: Image Augmentation

- Augmentations create new training examples for our models to learn from
- We chose the following augmentations:
  - Flip: Horizontal
  - Rotation: Between -10° & +10°
  - Shear: ±15° Horizontal, ±15° Vertical
  - Brightness: Between -25% and + 25%
  - Blur: Up to 3px
  - Bounding Box: Noise: Up to 5% of pixels
- Augmentations were selected to adhere to real-life situations where the crosswalk light orientation won't always be optimal
- Significant contributor to great mAP values

# Updated Dataset & Model pt. 2

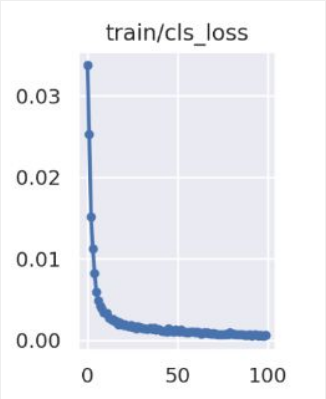



Figure 35: Class Loss for Train Set (MC)

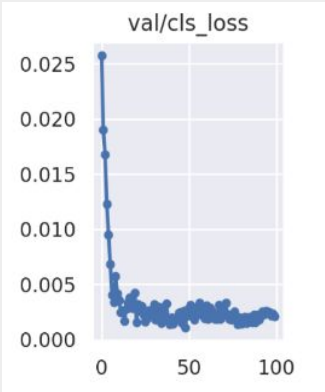



Figure 36: Class Loss for Validation Set (MC)

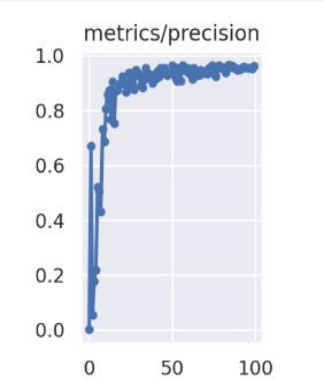



Figure 37: Overall Precision (MC)

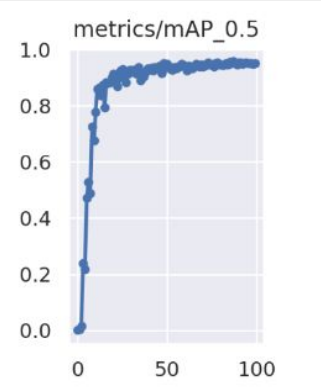



Figure 38: Mean Average Precision (MC)

Results from Multi-Classification

# Updated Dataset & Model pt. 2

```
custom_YOLOv5s summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 11/11 [00:04<00:00,  2.26it/s]
                   all        322        591       0.95      0.912      0.954      0.718
            Don-t Walk        322         57      0.964      0.965      0.985      0.841
                  Walk        322         45          1      0.881       0.95      0.741
                  cars        322        489      0.886       0.89      0.927      0.573
```

Figure 39: Data from Multi-Classification

- We added the ‘cars’ classification to have the model detect cars (or any road vehicles) that may pose a threat to the user crossing a crosswalk
- The ‘cars’ classification was achieved in exchange for an insignificant change in the mAP values for the other classes.

# Future Improvements

- Integrating all components with the Jetson Nano instead of using an Arduino
- Increasing the amount of detectable classes in the multi-classification version of the model
  - Avoiding losing mAP in re-training for more classes, avoiding overfitting the model
  - Re-programming the logic of the camera from relying on binary classification to a more complex and robust multi-classification algorithm
- Transforming and standardizing audio files into a specific format to be able to train a basic classification model
  - Discerning whether or not the microphone is hearing car sounds vs other sounds of life
  - Fast fourier transformation: spectrograms, Mel-frequency cepstral coefficients

# 05

# Electrical

# General Wiring Diagram

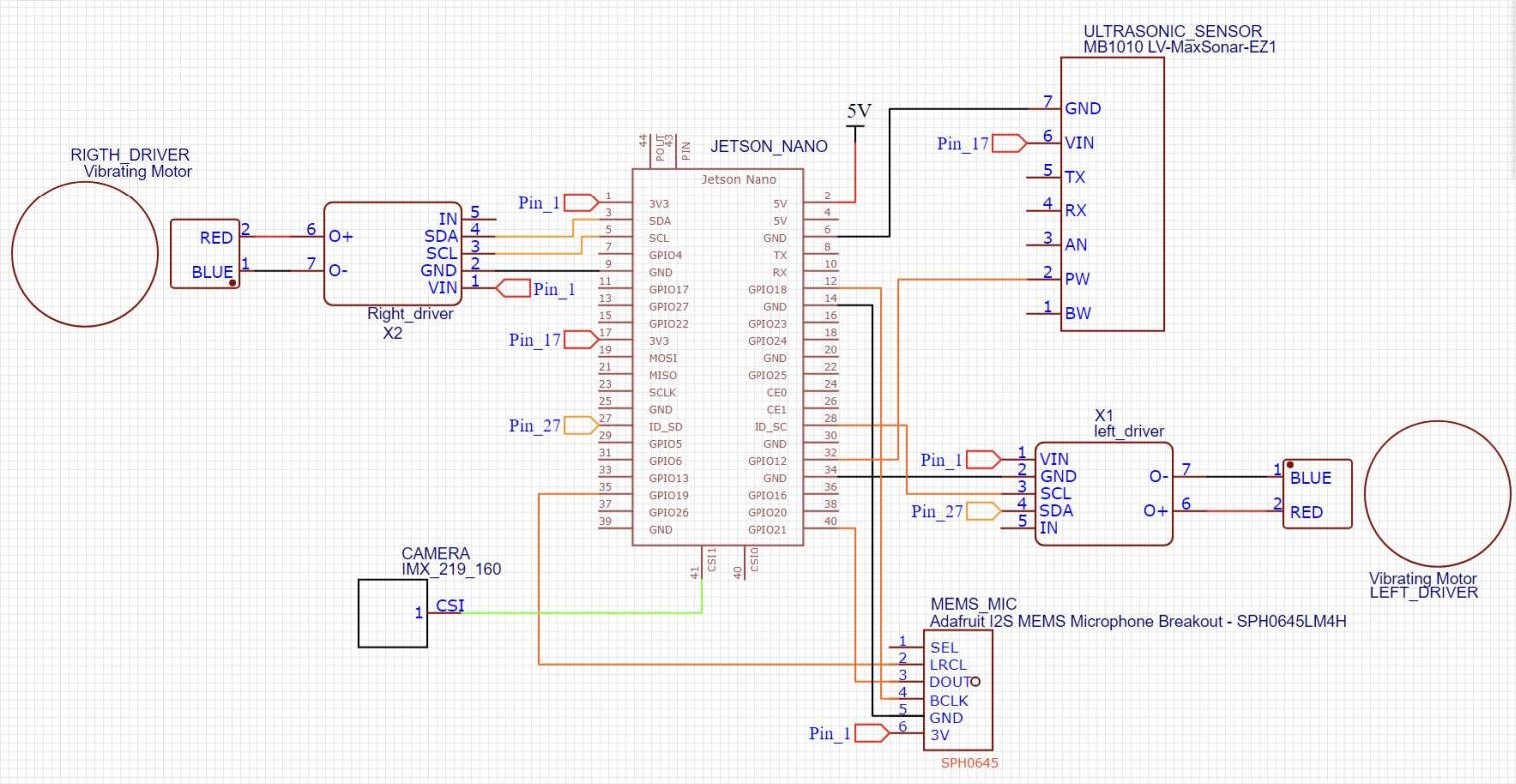



Figure 40: Jetson Nano Wiring Diagram

# Arduino Wiring Diagram

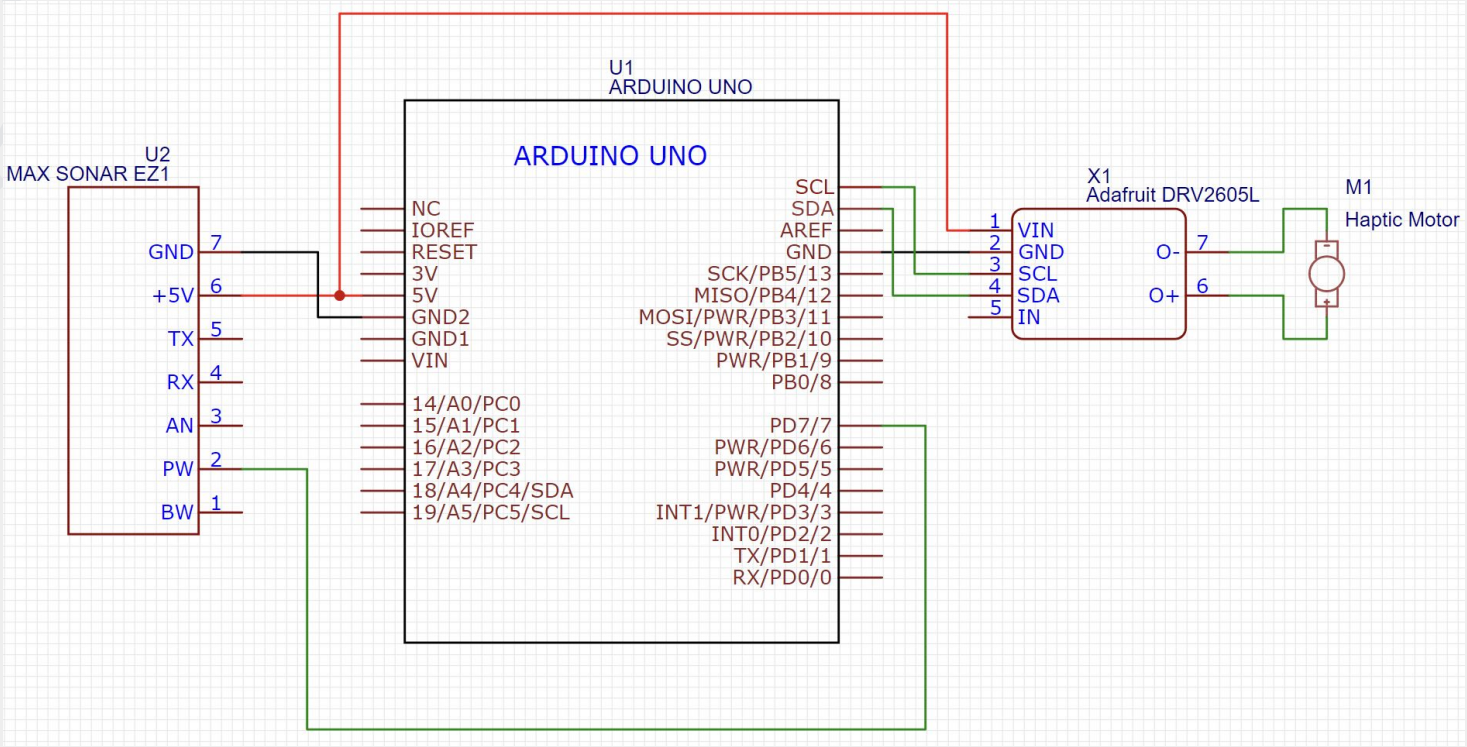



Figure 41: Arduino Wiring Diagram for Ultrasonic and Haptic Motor

# Wiring Diagram - PCB

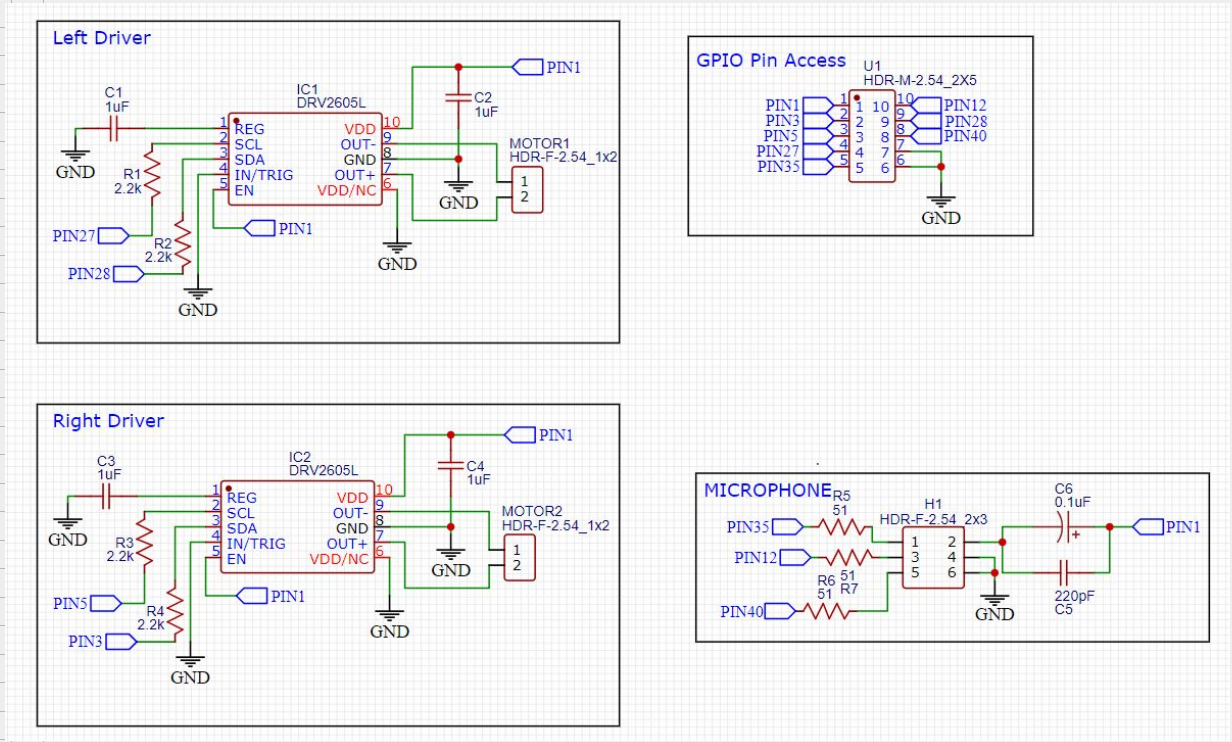



Figure 42: Circuit Diagram of Drivers and Mic Components

- Housed on separate PCB

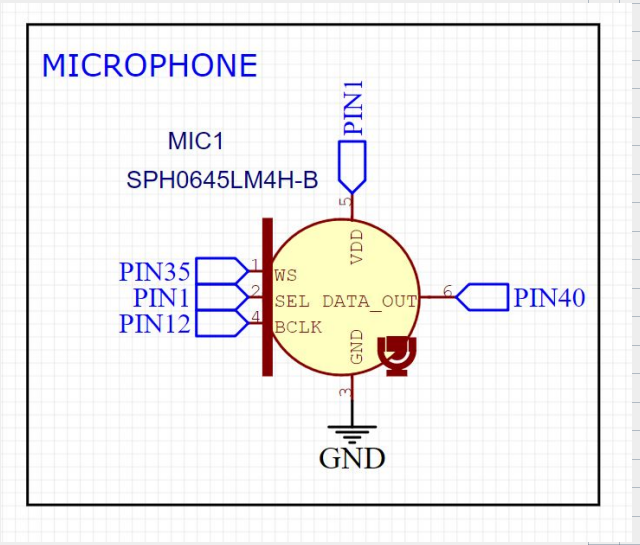



Figure 43: Circuit Diagram of Mic

- MEMS Mic housed on temples

# PCB Design - Main

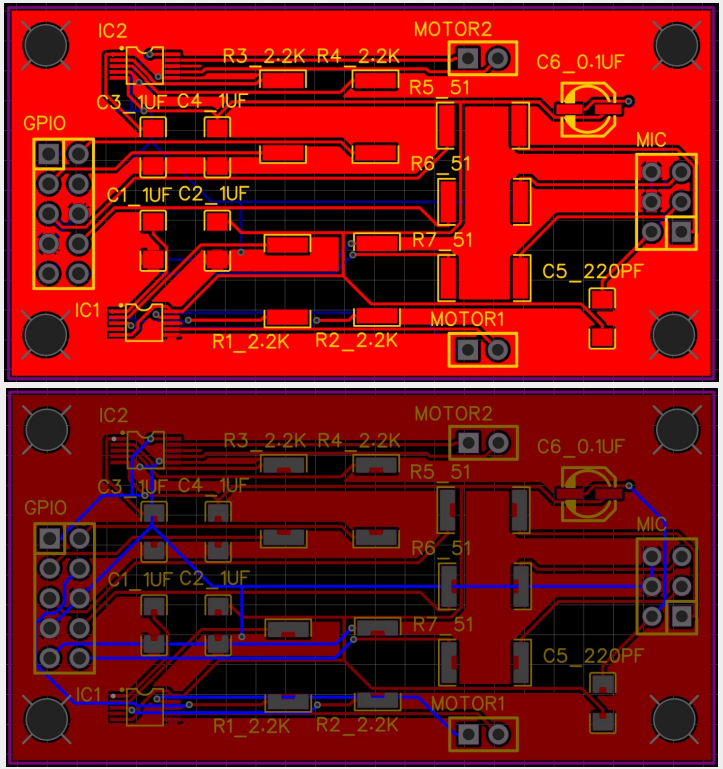



Figure 44: PCB 3D View [1,7]

- 2.36x0.86 inches
  - (60mm x 22mm)
- Resistors - 2512
- Capacitors - 1812
- M3 holes (3.5mm)
- Ground Plane
- Labeled components with values
- Male Pin Header - visualization purpose

# MEMS Microphone PCB

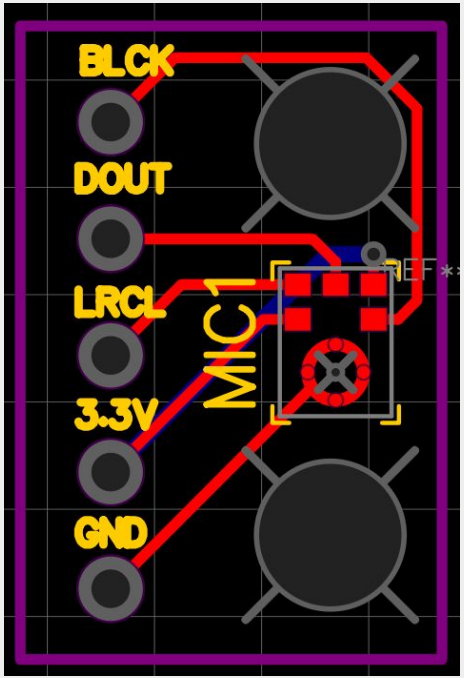



Figure 45: MEMS Mic PCB 3D Top View [7]

- 0.39x0.59 inches
  - (10mm x 15mm)
- THT to Mic F-HDR
- Rectangular board outline
- 2 copper layer
- M3 holes (3.5mm)
- No ground plane

- Intention - to reduce space taken on glasses temples

# Final Wiring

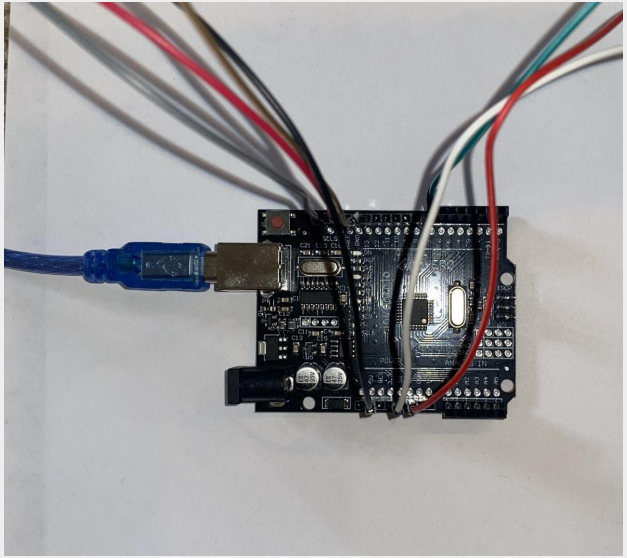


Figure 46: Arduino Uno [8]

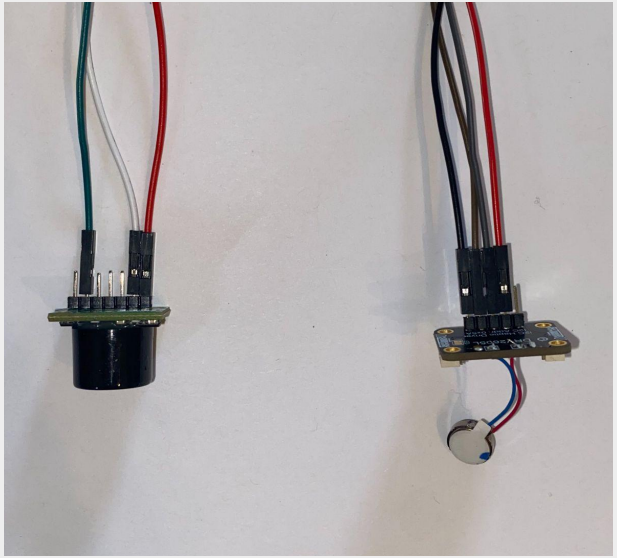


Figure 47: Ultrasonic sensor and Haptic Driver with Motor [3] [5]

# Future Improvements

- Integrate all components with the Jetson Nano instead of using an Arduino
- Utilize the PCB instead of wiring everything directly to the microcontroller
- Improve upon PCB design
    - Decrease surface area by using smaller components for ergonomics
    - Add ultrasonic PCB onto main PCB to condense temple components
- Test multiple alternative components to determine the most effective choice
- Upgrade battery to increase run time

# 06

# Final Bill of Materials

# Bill Of Materials

| | Guidance Glass Bill of Material (Breadboarding) | | | | | | |
|---|---|---|---|---|---|---|---|
| Manufacturer | Item Description | Supplier | Supplier Part Number | Link | Quantity | Unit Cost | Total |
| Adafruit | Adafruit I2S MEMS Microphone Breakout | Adafruit | B06XNL2GBW | https://www.amazon.com/Adafruit-I2S-MEMS-Microphone-Breakout/dp/B06XNL2GBW | 1 | 10.03$ | 10.03$ |
| Waveshare | 8-Megapixels Camera Module 3280 × 2464 Resolution 160 Degree Wide Angle of View with IMX219 Sensor | Waveshare | IMX219-160 Camera | https://www.waveshare.com/imx219-160-camera.htm | 1 | 26.59$ | 26.59$ |
| Arducam | Arducam Sensor Extension Cable 300MM/1FT Cable to Extend IMX219 Work with V2 Camera on Jetson Nano | Amazon | B07TYHKXS5 | https://www.amazon.com/pcs-Raspberry-Camera-Cable-60cm/dp/B07S72KKBT | 1 | 12.99$ | 12.99$ |
| MAXBOTIX, INC. | Ultrasonic Range Finder | Amazon | B00A7YGVJI | https://www.amazon.com/Maxbotix-MB1010-LV-MaxSonar-EZ1-Ultrasonic-Finder/dp/B00A7YGVJI | 1 | 29.99$ | 29.99$ |
| Vybronics Inc | VIBRATION LRA MOTOR 170 HZ 2G | Digikey | VG1040003D | https://www.digikey.com/en/products/detail/vybronics-inc/VG1040003D/10285886 | 2 | 3.71$ | 7.42$ |
| Adafruit | Adafruit DRV2605L Haptic Motor Controller - STEMMA QT / Qwiic | Amazon | 2305 | https://www.adafruit.com/product/2305 | 2 | 7.95$ | 15.90$ |
| Waveshare | Camera Cable FFC 60cm | Amazon | B07S72KKBT | https://www.amazon.com/pcs-Raspberry-Camera-Cable-60cm/dp/B07S72KKBT | 1 | 6.99$ | 6.99$ |
| Waveshare | 5V Cooling Fan Brushless Fan 4PIN | Amazon | B07VS8LJJW | https://www.amazon.com/Dedicated-Developer-Adjustment-Reverse-Proof-40mm%C3%9740mm%C3%9720mm/dp/B07VS8LJJW | 1 | 8.35$ | 8.35$ |
| Adafruit | Adafruit ADXL345 - Triple-Axis Accelerometer (+-2g/4g/8g/16g) w/ I2C/SPI [ADA1231] | Amazon | ADXL345 | https://www.amazon.com/dp/B01BT4N9BC?psc=1&ref=ppx_yo2ov_dt_b_product_details | 1 | 17.58$ | 17.58$ |
| N/A | 3D Printing Right and Left Temple(acrylonitrile butadiene styrene) | N/A | N/A | N/A | | N/A | N/A |
| JLCPCB | Main PCB and Mic PCB | N/A | | https://jlcpcb.com/ | 5 | 13.38$ | 66.88$ |
| N/A | 3D Printing Glass Part(acrylonitrile butadiene styrene) | N/A | N/A | N/A | | N/A | N/A |
| | Additional Parts for Testing | | | | | | |
| Adafruit | Vibrating Mini Motor Disc | | 1201 | Vibrating Mini Motor Disc : ID 1201 : $1.95 : Adafruit Industries, Unique & fun DIY electronics and kits | 3 | 1.95$ | 5.85$ |
| | | | | Total Cost | | 139.51$ | **208.57$** |

**Total Cost: $208.57**

Figure 48 : Bill of Materials

# 07

# Live Demo

# Thank you for listening

**CREDITS:** This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

# Citations


[1] Texas Instruments. "DRV2605L Datasheet." May 2022. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv2605l.pdf

[2] Adafruit Industries, “Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H,” www.adafruit.com. https://www.adafruit.com/product/3421

[3] Vibronics Inc., “LRA Coin Vibration Motor - VG1040003D,” Vybronics, May 06, 2019. https://www.vybronics.com/coin-vibration-motors/lra/v-g1040003d

[4] MaxBotix Inc., “LV-MaxSonar-EZ Datasheet,” MaxBotix. https://maxbotix.com/pages/lv-maxsonar-ez-datasheet

[5] Adafruit Industries, “Adafruit DRV2605L Haptic Controller Breakout,” Adafruit Learning System. https://learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout/overview

# Citations

[6]“NVIDIA Jetson Nano Specs,” TechPowerUp. https://www.techpowerup.com/gpu-specs/jetson-nano.c3643

[7] “SPH0645LM4H-B,” digikey. https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPH0645LM4H-B.pdf

[8]G. Staples, “Arduino Power, Current, and Voltage Limitations.” https://www.electricrcaircraftguy.com/2014/02/arduino-power-current-and-voltage.html

# Citations

[9] Dwyer, B., Nelson, J. (2022). Roboflow (Version 1.0) [Software]. Available from https://roboflow.com. computer vision.

[10] Pedestrian Signal Images Dataset. UCI Senior Project (2023). Roboflow Universe (Version 1.0) [Software]. Available from https://universe.roboflow.com/uci-senior-project/pedestrian-signals/dataset/9. https://roboflow.com/ computer vision.

[11] HAWC_AI Computer Vision Project. Kolattukudy, J. (2023). Roboflow Universe (Version 1.0) [Software]. Available from https://universe.roboflow.com/joseph-kolattukudy/hawc_ai/dataset/1/images/?split=train&numImages=100. https://roboflow.com/ computer vision.

[12] July 29th Annotations, Kolattukudy, J. (2023). Roboflow Universe (Version 1.0) [Software]. Available https://universe.roboflow.com/joseph-kolattukudy/july-29th-annotations. https://roboflow.com/ computer vision.

[13] VehicleCount Computer Vision Project, Fyp (2023). Roboflow Universe (Version 1.0) [Software]. Available https://universe.roboflow.com/fyp-5ctjf/vehiclecount. https://roboflow.com/ computer vision.

[14] PedestriansDetection Computer Vision Project, Reg, V (2023). Roboflow Universe (Version 1.0) [Software]. Available from https://universe.roboflow.com/victor-reg/pedestriansdetection computer vision.