Elder Care Wearable Device

Final Report

ECEN 4024 – Capstone Design

Lauren Brown

Scott Kincannon

Bill Liando

Elinor Rowe

Oklahoma State University

Abstract:

Elderly care is an important area of research as a high proportion of our population is reaching old age. The current care system can be very costly and hiring a personal aid is unrealistic for some. The ASCC lab at Oklahoma State University has developed a Personal Assistive Robot to interact with elderly people in their homes. This robot can monitor and communicate with the user, but it is restricted to one part of the home. In this lab report for the Final Project of ECEN 4024 Capstone Design, four undergraduate students were tasked with creating a wearable device for elderly users that is low-powered, compact, able to monitor daily activities, collect images, interact with the user and wirelessly communicate with a Personal Assistive Robot (ELSA) while the user is not near the robot. To complete this project, our team designed a wearable device with multiple sensors such as an accelerometer, microphone and camera that would track movement data. The device was also able to exchange messages with ELSA, closing the gap between ELSA and our device.

Table of Contents:

Team and team structure:

Bill Liando: I was the point of contact for the group. I was also in charge of programming, configuring the software and integrating the component's software.

Lauren Brown: I worked on programming, configuring the software and ensuring power efficiency.

Elinor Rowe: I worked on the designing the component's layout and 3D printing design and production.

Scott Kincannon: I was responsible for integrating the component's hardware.

Problem statement:

Our project objective was to design a wearable device to assist elderly users by communicating with a Personal Assistive Robot (ELSA). This device must monitor the activities of the user while the user is out of view of the robot, providing the robot with the information necessary for effectively aiding the user. The device was to be designed to be comfortable, easy to use, and power efficient.

Design Constraints:

- Minimal size and weight of device
- Low power
- Privacy/Consumer awareness of camera use
- Limited time available to complete project
- Parts delays due to inclement weather
- Safety of consumer and design engineers
- Precision of 3D printers
- Budget requirements

Initial Part Comparison:

1. Speaker
    a. Our original choices for the speaker were the CLS0231MA-1-L152, K 16-8 ohm, and the AS03008M R-R speakers. As all the speakers are 0.5 W and 8 ohm impedance, we chose to use the CLS0231MA-1-L152 for its size and its sound pressure level. This speaker was intended to be paired with the ReSpeaker 2-Mic Pi HAT v1.0, which would act as the speaker driver.
2. Microphone
    a. Our three preferred options for the microphone were the ReSpeaker 2-Mic Pi HAT v1.0, the POM-2738L-LW100-R, and the CMA-4544PF-W. We decided to use the ReSpeaker 2-Mic Pi HAT v1.0 because it included a microphone and microphone driver, but also provided a speaker driver. The size and weight was larger than our other options, but this device acted as an all in one for our audio needs.
3. Battery
    a. Our original design options for batteries were the Attom Tech Compact Battery Pack, the Lithium Ion Polymer Battery - 3.7v 2500mAh, and the TG90 External Battery Pack. The battery packs had the advantage of large capacity and simple connection to the device, but were larger and heavier, while the lithium ion battery had less capacity and required extra connection circuitry, but was smaller and lighter than the battery packs. Since minimizing size and weight was a major concern during our initial design, we chose to use the Lithium Ion Polymer Battery - 3.7v 2500mAh.
4. Accelerometer
    a. Our three options for accelerometers were MPU-6050, MMA8451 and ADXL345. These were selected because they were small and power efficient. We decided to go with the MMA8451, since it was the smallest of the three and consumed the second lowest amount of power.
5. Microcontroller
    a. For the microcontroller our three preferred options were the Raspberry Pi Zero W, The Sancloud BeagleBone Enhanced Wifi-1G, and the ESP32-wroom-32e. We decided to go with the Raspberry Pi Zero W for its size, weight, low power requirements, and module attachments. Since the goal of this project was to create the smallest device possible we went with the smallest option. Because this option also required the least power, it would allow us to reduce the size of the battery as well. We also chose this microprocessor because it is widely used. Since the Raspberry Pi is a common choice for small projects, there are plenty of available coding libraries and components specifically designed to work with the pi.

6. Camera
    a. The three initial design options for the camera were the Raspberry Pi Camera V2.0, the OV7670, and the ESP32-CAM. The ESP32-CAM was intended for use with a different microprocessor, and was not suitable once we chose to use the Raspberry Pi. While the Raspberry Pi Camera V2.0 had the highest operating current, it was also specifically designed for efficient use with the Raspberry Pi. We chose to use the Raspberry Pi Camera V2.0 because it let us effectively and easily connect to our microcontroller, allowing us to use our GPIO pins for other parts, and made programming easier and more efficient later.
7. Ambient Light Sensor
    a. We added the ambient light sensor to our initial design by request of Dr. Sheng. We selected this ambient light sensor because it was pre-packaged and easy to install. We did not include this sensor in our final product, because the lack of analog input pins on the Raspberry Pi Zero limited the sensor's usefulness.

Final Part Comparison:

1. Speaker
    a. Speaker-

    After getting the speaker driver to work properly with the Pi we soon realized that the 0.5 watt speaker was too quiet. Even at the Pi's maximum volume, we could only hear sound if we had the speaker pressed to our ear. To replace it, we chose a 3W speaker, which worked correctly and was much louder. However, the speaker's casing was too large and took up too much space in our case. Finally, we received a 2 watt speaker from Dr. Sheng and his graduate team. The overall sound quality was roughly the same as the 3 watt speaker, but the size profile was much smaller and allowed us to fit the speaker into a smaller case.

    b. Speaker Driver-

    The use of a speaker driver is required for a speaker to work properly and receive enough power to play sound. This means that an amplifier is also needed to boost, or drive, the speaker. Another requirement is that the driver device needs to act as a DAC(digital-to-analog converter), as our microprocessor only has digital GPIO pins. Our initial design contained the ReSpeaker 2-mic Pi HAT v1.0 as our speaker driver and DAC, but this solution was found to require connection to all 40 GPIO pins on our microcontroller, so we had to look for other solutions. Our next design involved designing and building a PCB speaker driver that would use the Raspberry Pi's PCM pins. Unfortunately, the Raspberry Pi was unable to recognize the PCB speaker driver as a valid sound output, so we could not send sound to the speaker. We then connected the speaker to the Raspberry Pi's data micro-USB port using several adapters and a driver. This setup worked correctly, but was very large and heavy. In addition, the driver did not put out very much

power to the speaker, making it too quiet. After talking with Dr. Sheng, he mentioned researching Adafruit for a solution, and that led us to the Adafruit I2S 3W Class D Amplifier Breakout - MAX98357A. This board includes the required driver, amplifier, and DAC. We were able to use this board to connect the speaker to the Raspberry Pi's PCM pins, which were reconfigured to act as I2S. This configuration was able to provide louder speaker output while adding much less size and weight to the device.

2. Microphone
    a. Our original plan was to use the ReSpeaker 2-mic Pi HAT v1.0 as our microphone. We unfortunately had to replace the Pi HAT as the component used all 40 pins on the Raspberry Pi Zero. With this device, we would have been unable to attach any of our additional components to the Pi. Instead, we moved on to a micro-USB connection for our microphone. We had a mini microphone with a headphone jack that led to an adapter that plugged into the Raspberry Pi. Unfortunately we never were able to get the Raspberry Pi to recognize the microphone as an audio option. Our final solution was moving to the Electret Microphone Amplifier - MAX4466 with Adjustable Gain paired with the MCP3008 ADC. This microphone worked with our ADC, and we were able to read volume. Our biggest issue with this solution is that we were never able to record an audio file, we could only read the noise volume in the area. This could potentially be fixed by recording the noise levels from the microphone and then processing that data into a wav file using the resolution and sampling rate of the ADC. At one point we considered solving the issue by using PCM pins for the microphone and ADC, as we did with the speaker. Unfortunately, we do not have enough PCM pins or channels on the Raspberry Pi for both the speaker and microphone to run.
3. Vibration motor
    a. Vibrating Motor

        There were a number of commercial mini vibrating motors for us to use. We eventually chose the DZS Elec 1027-L35-4 mini vibrating motor for its high RPM and its 3V rating. Similar motors were the same size and weight, but the DZS motor was able to share a voltage supply bus with the accelerometer. This was convenient as we already needed to bus the SCL and SDA connected to the accelerometer in order to add the haptic motor controller.
    b. Motor Driver

        The Motor driver was connected to the Raspberry Pi's I2C pins (see Circuit Schematic in Design Details Section), which required the construction of a bus, since the I2C pins are also used for the accelerometer. It worked as intended, and was not redesigned for our final product since we encountered no issues with it.

4. Battery
    a. To effectively power the device, it was necessary to select a battery with enough capacity to last at least sixteen hours, while minimizing the size and weight. In addition, the complexity of connecting the battery to the Raspberry Pi, the size and weight of the materials necessary for that connection, and the space efficiency of the battery when combined with other components had to be considered. Our initial design choice, the flat lithium ion battery, required several extra connections to both safety circuitry and a 3.7V to 5V converter. This converter would require a two-wire connection to the Raspberry Pi's power pins, which would require removing the power micro-usb port from the Raspberry Pi. In addition, while it was small and light, it was so much wider than the Raspberry Pi that it was making it difficult to design an efficient case without wasting extra space. For these reasons, we chose to switch to a cylindrical battery with built-in safety circuitry and converter that allowed for easy connection to the Raspberry Pi. Its smaller width would also allow for more efficient use of space in case design. While this battery was more efficient than the initial one, the placement of the circuitry on the end of the battery caused more issues with efficient use of space. For this reason, we switched to a similar cylindrical battery that had circuitry placed on the side instead of the end of the battery. This allowed for both efficient use of space and easy, reliable connection to the Raspberry Pi. All batteries tested had a similar capacity, calculated to be enough to power the device for 16 hours (see Power Consumption Analysis chart in Design Details Section). The battery was connected to the power micro-USB port on the Raspberry Pi using a standard micro-USB to USB adapter.
5. Accelerometer
    a. The MMA8451 was connected to the Raspberry Pi's I2C pins (see Circuit Schematic in Design Details Section) according to our initial design plan. It worked as intended, and was not redesigned for our final product since we encountered no issues with it.
6. Camera
    a. The Raspberry Pi Camera v2 was connected to the Raspberry Pi's using the Raspberry Pi's dedicated camera interface port, according to our initial design plan. It worked as intended, and was not redesigned for our final product since we encountered no issues with it.
7. Microcontroller
    a. Raspberry Pi OS was installed on the Raspberry Pi Zero W by downloading the operating system onto the SD card. The Raspberry Pi's configuration files had to be adjusted to allow for connection with the campus wifi, as it has extra security measures not present in most wifi systems. With wifi connection established, it was possible to connect to the Raspberry Pi using ssh, in order to write code and

adjust configurations more easily. Because the campus wifi involves use of dynamic IP addresses, the IP addresses used to ssh into the Raspberry Pi and those present in the code had to be changed on occasion.

8. LEDs
    a. Two LEDs, a green LED to indicate when the device is on and a red LED to indicate when the camera is on. Connected to small resistors that attach to the raspberry pi. We always planned on adding LEDs to the project but held off until the end once the casing was completed. The coding and soldering needed were simple additions to the project.

Software Design Strategy:

Our initial strategy was to utilize pre-existing libraries and examples for each individual component for testing, and then integrate the code for each component into our client/server program. We would then run the device with each individual component until they were all working properly. If one device did not work well with what we currently had, we would take the component out to reduce the weight and size of our total device since that sizing and weight was a main criteria of our project.

The first software issue we encountered was getting the best operating system for our project's purpose downloaded onto the Raspberry Pi. There are many different operating systems that are compatible with Raspberry Pi Zero W's, but we needed one that would be universal with the components we would be using in our project. We decided to use "Raspberry Pi OS" since it came directly from Raspberry Pi, and most of the packaging libraries for our individual components were programmed to be compatible with the operating system.

The second thing we did was to configure the Raspberry Pi to use OSU's "eduroam" wifi. This network has a special protocol that the Raspberry Pi cannot automatically connect to unless we configure the "Wi-Fi Protected Access" (WPA) files. With the help from Dr. Sheng's graduate student Zhidong and some research online [4], we were able to set up the Raspberry Pi to automatically connect to the eduroam wifi. On the Raspberry Pi command line, we would run "sudo nano /etc/wpa_supplicant/wpa_supplicant.conf" to edit the "wpa_supplicant.conf" file. The file is easy to edit and your own login information can be used to log into the eduroam wifi.

After we configured the Raspberry Pi we implemented the components as we received them. The first components we received were the Accelerometer and Ambient Light Sensor. With pre-existing libraries for each, it was easy to test the functionality of both sensors and to add it to our device. When testing the ambient light sensor we concluded that it would not make a significant impact on our design since we were only able to receive digital values on the Raspberry Pi. The accelerometer did not have any issues so we used it in our final project. To determine which acceleration values indicated a fall, we secured the sensor to a student's shirt and we had the student walk around and replicate a fall. We then used the corresponding values as criteria to signal that a fall had occurred. After testing, we concluded that the values reached when a fall occurred were greater than $14 m/s^2$ in the X-axis, greater than $14 m/s^2$ in the Y-axis,

and less than $3m/s^2$ or greater than $14m/s^2$ in the Z-axis. The Z-axis had two parameters since the MMA8451 has a $9.81m/s^2$ acceleration built in due to gravity. The two parameters ensured a fall would be recorded on the Raspberry Pi. Other components we received later include the vibration motor, which required us to set up an effect number to determine what vibration pattern it would use.

To initialize the camera we followed the instructions from Raspberry Pi's website and we were able to get it to work without a problem. When implementing the camera into our device we designed it to take multiple pictures when a fall was detected and be turned off otherwise. We had the camera take pictures when any of the accelerometer's threshold values was reached. We placed the capture command in a for loop, causing the camera to take 10 pictures, one every 50 milliseconds.

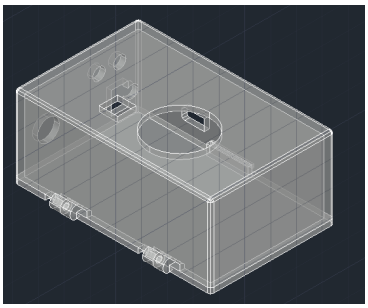
The next component we included in our device was a speaker. The speaker went through many different iterations as mentioned earlier, but the programming process remained the same for most of them. We would construct a speaker and format our audio configuration files accordingly. The command "sudo nano /boot/config.txt" was used to edit the "config.txt" file which determined which audio output the Raspberry Pi would use. The initial speaker setup using the PCB board had us convert GPIO pins 13 and 18 to PWM pins using the command "dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4". When we edited the configuration file to update the pins, the Raspberry Pi did not detect any audio output source so we were unable to output any audio. This was because the Pi requires its audio devices to have a power audio driver, which was not present in this design. The next speaker we attempted to implement was a USB-speaker driver. Once we edited the audio configuration file using the command "dtoverlay= USB Audio [USB Audio]", the Raspberry Pi detected the Speaker and we were able to play audio. However, the audio was very faint and would be difficult to hear especially for an elderly person, so we had to consider other solutions. Our final speaker used converted PCM pins to act as I2s pins. We edited the audio configuration file by adding "dtoverlay=hifiberry-dac" and "dtoverlay=i2s-mmap". The Raspberry Pi recognized the audio and we were able to successfully play audio at different volumes that would be easily detectable. The speaker driver had to be initialized with a sampling frequency, number of bits per sample, and channel number corresponding to that of the sound file it needed to play. We also used an online text-to-speech software to create sample audio messages to interact with the user. This allowed us to play a specific message if a fall was detected.

Our first attempt to implement communication on our device required us to use a TCP socket to create a client/server connection between the device and the server. We chose to use another Raspberry Pi to act in place of the server for this project. While we successfully programmed a TCP socket that allowed communication between the client and server, it became apparent that the server socket could not effectively receive the pictures and data sent to it by the client. The server attempted to receive data from the socket in chunks, but was unable to detect when all chunks had been received, creating an endless loop as the server searched for data that was no longer being sent. We attempted to fix this problem by setting a certain number of bytes

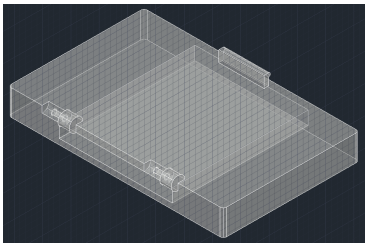
or amount of time for the server to search for data, but the variability in file sizes being received made this difficult. We also tried to add end-markers to the files being sent which could be detected by the server, but searching through all of the picture data that was being sent in byte format was time consuming. Eventually, we moved away from the TCP socket as our communication protocol, and chose to use an SCP socket protocol instead.

With the SCP socket, we were able to connect the client to the server through "SSH". This allowed the client to send whole files directly to the server and have them stored, avoiding the issues encountered with the receiving loop. Using SCP protocol allowed for successful communication between the client and server, and we were easily able to send files back and forth. The working code for the individual components was then integrated with the SCP communication program to create the device's logic.

Physical Appearance/Dimensions:

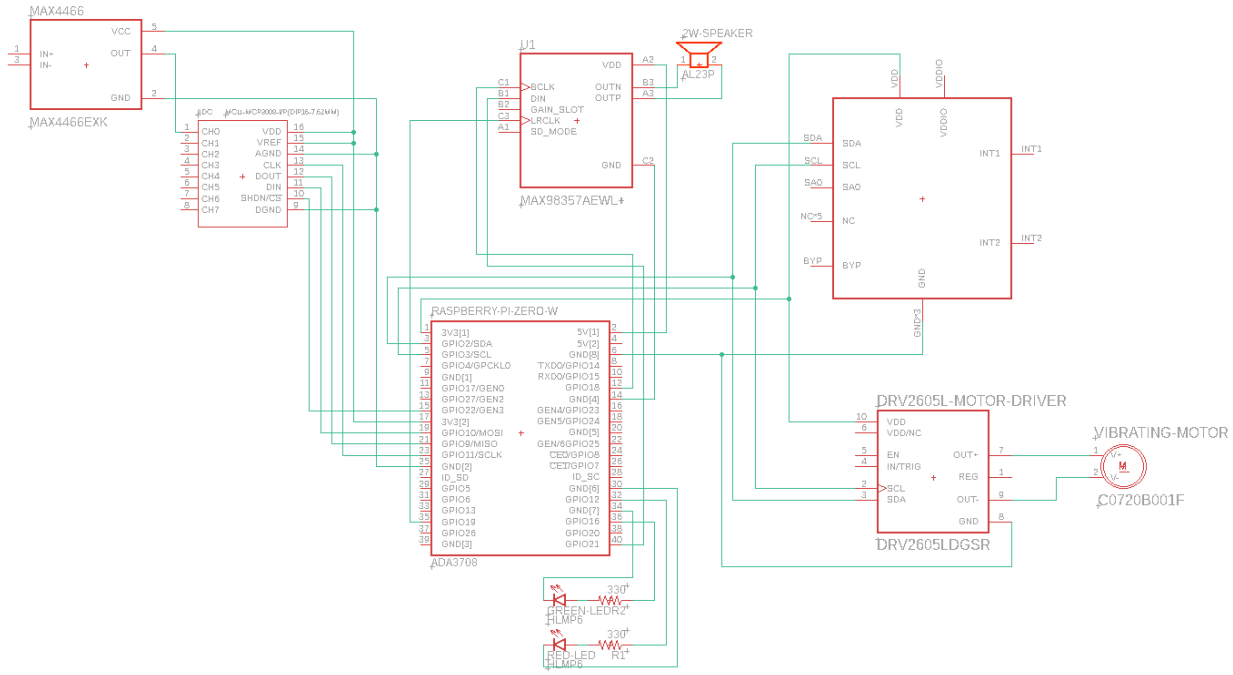


The component layout and case were designed to be small but still be functional. The microphone and the LED lights face upwards towards the user. This allows for the microphone to pick up a better quality of sound from the user, as well as allowing them to see when the device is running and when the camera is on. The camera faces outwards to allow for pictures to be taken of the surroundings in case the user falls outside the view of ELSA. The speaker also faces away from the body, since its surface area did not allow it to be on the top of the box. The power switch is located on the side of the box so it does not accidentally get pressed. The layout of the rest of the pieces inside is designed to help keep the overall dimensions small.



The final weight of the component parts and connections alone are 3.7oz. With the added weight of the plastic case, the device comes to 5.6oz. The strap attachment and corresponding backing added another 1.2 oz, for a final weight 6.8oz. The dimensions of the case with the plain backing is 95mm x 63mm x 40mm. With the backing that has the pocket clip, the dimensions increase to 95mm x 63mm x 44mm. The backing that fits the strap increases the dimensions to 95mm x 63mm x 49mm, but this does not include the dimensions of the strap.

Schematic for Raspberry Pi GPIO Pin Connections:



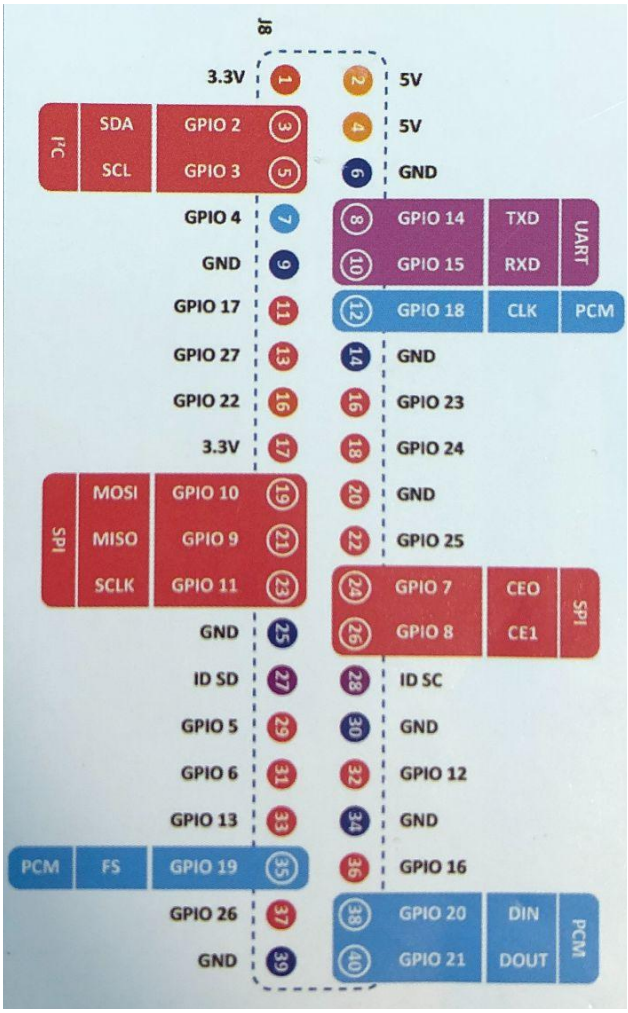
GPIO Pin Connections:

Accelerometer-
SDA - Pin 3(I2C SDA)
SCL - Pin 5(I2C SCL)
VDD - Pin 1(3.3V)
GND - Pin 6(GND)

Haptic Motor Driver-
SDA - Pin 3(I2C SDA)
SCL - Pin 5(I2C SCL)
VDD - Pin 1(3.3V)
GND - Pin 6(GND)

Speaker Driver-
VDD - Pin 2(5V)
GND - Pin 14(GND)
BCLK - Pin 12(PCM CLK)
DIN - Pin 40(PCM DOUT)
LRCLK - Pin 35(PCM FS)

ADC-

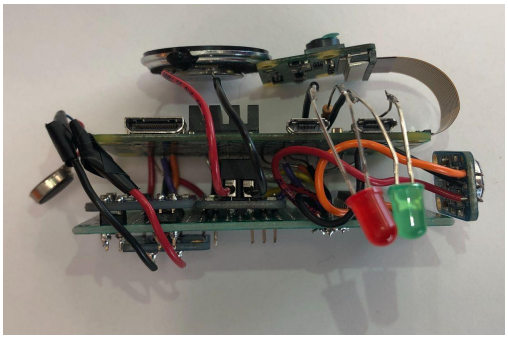


VDD - Pin 17(3.3V)
VREF - Pin 17(3.3V)
AGND - Pin 25(GND)
DGND - Pin 25(GND)
CLK - Pin 23(SPI SCLK)
DOUT - Pin 21(SPI MISO)
DIN - Pin 19(SPI MOSI)
SHDN/CS - Pin 15(GPIO 22)

Microphone-
VDD - Pin 17(3.3V)
GND - Pin 25(GND)

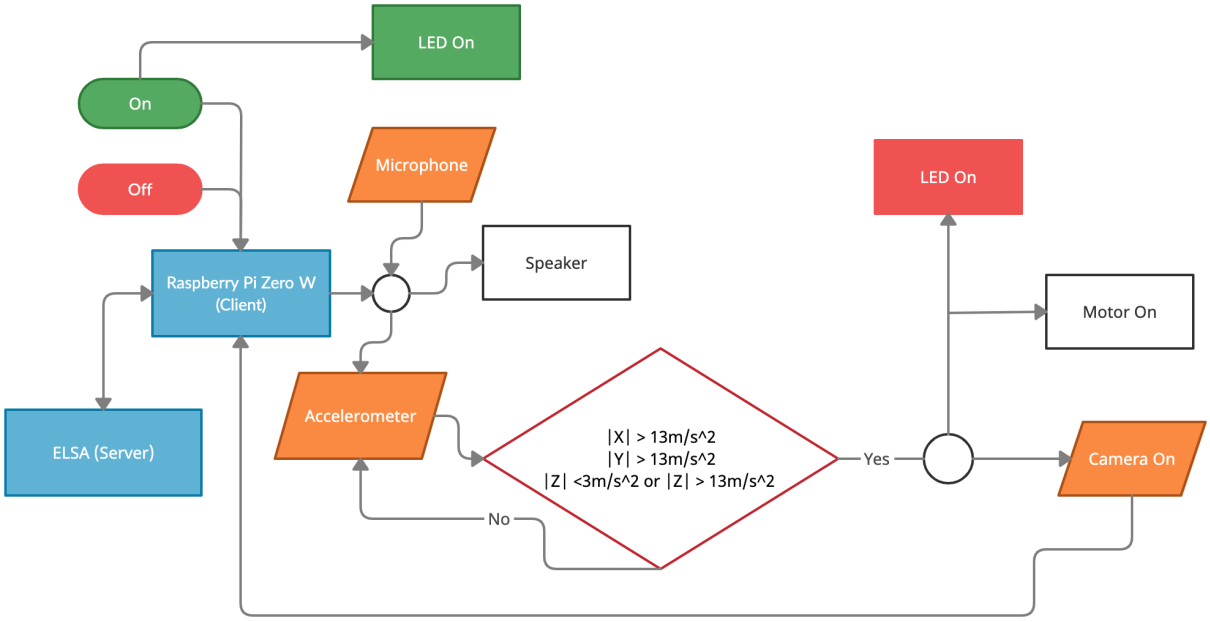
Green LED-
VDD - Pin 36(GPIO 16)
GND - Pin 34(GND)

Red LED-
VDD - Pin 32(GPIO 12)
GND - Pin 30(GND)



With the use of a perf board All the devices were attached together and then soldered to the raspberry pi. The accelerometer and haptic motor were soldered to the perf board with the VDD, GND, SCL, and SDA pins being bused to the same raspberry pi pins. The vibrating motor is soldered to the haptic motor and is attached to the bottom side of the case. The Speaker driver was soldered to the perf board with connecting wires between the raspberry pi and the perf board. The 2 watt speaker that attaches to the speaker driver is directed to face outward on the front of the device. The microphone ADC is soldered to the perf board and has connecting wires to the microphone and to the raspberry pi. The microphone itself is wired to the ADC for output with the VDD and GND being bused with the VDD and GND of the ADC. The microphone is directed to the top of the device and faces outward. The Camera is attached via a ribbon cable. It has its own connection on the raspberry pi and points outward on the front of the device. The two LEDs are connected to 330ohm resistors that are soldered to the raspberry pi. The LEDs are positioned to sit on top of the device and point upward from the device. The Device is powered by a separate battery that connects via the micro usb port.

Programming Flowchart:



Power Consumption Analysis:

Item	Worst Case (mA)	Expected Case (mA)	Best Case (mA)
Microcontroller	150	100	80
Camera	225	15	12
Speaker	2.85	1	0.40
Motor	3.5	3	2.3
Accelerometer	0.165	0.165	0.165
Mic	0.060	0.060	0.024
Total	380.375	124.299	96.089
Operational Time	**5.26 Hours**	**16.10 Hours**	**20.81 Hours**

Operating Instructions:

Put the desired backing on the device, and attach to the user, with an upright initial position. Press the power button to start the device, and wait a few minutes for wifi connection to be established. Then use wifi to connect to the device through an ssh terminal on another computer. Begin functionality by running the eldercare.py program on the device. The device should then function correctly, detecting falls and sending data to the server. When finished using the device, stop the program from running in the ssh terminal, and press the power button to turn the device off. Charge the device by plugging the micro-USB charging port into a standard wall socket.

Test Code for Individual Parts:

Accelerometer

```
import time

import board
import busio

import adafruit_mma8451

i2c = busio.I2C(board.SCL, board.SDA)

sensor = adafruit_mma8451.MMA8451(i2c)

while True:
    z, x, y = sensor.acceleration
    print('Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2' .format( x, y,z))
    orientation = sensor.orientation

    print('Orientation: ', end='')
    if orientation == adafruit_mma8451.PL_PUF:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_PUB:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_PDF:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_PDB:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_LRF:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_LRB:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_LLF:
```

```
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_LLB:
        print('Portrait, up, front')
    time.sleep(1.0)
```

Camera

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Ambient Light Sensor

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(23,GPIO.IN)

for i in range(0,5):
    #print(GPIO.input(14))
    x =GPIO.input(23)
    if x == 1:
        print("It is dark")
    else:
        print("There is light")
    time.sleep(3)
```

Vibration Motor

```
# SPDX-FileCopyrightText: 2017 Tony DiCola for Adafruit Industries
# SPDX-License-Identifier: MIT

# Simple demo of the DRV2605 haptic feedback motor driver.
# Will play all 123 effects in order for about a half second each.
import time
```

```
import board
import busio

import adafruit_drv2605


# Initialize I2C bus and DRV2605 module.
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)

# Main loop runs forever trying each effect (1-123).
# See table 11.2 in the datasheet for a list of all the effect names and IDs.
#   http://www.ti.com/lit/ds/symlink/drv2605.pdf
effect_id = 1
while True:
    print("Playing effect #{0}".format(effect_id))
    drv.sequence[0] = adafruit_drv2605.Effect(effect_id)  # Set the effect on slot 0.
    # You can assign effects to up to 7 different slots to combine
    # them in interesting ways. Index the sequence property with a
    # slot number 0 to 6.
    # Optionally, you can assign a pause to a slot. E.g.
    # drv.sequence[1] = adafruit_drv2605.Pause(0.5)  # Pause for half a second
    drv.play()  # play the effect
    time.sleep(0.5)  # for 0.5 seconds
    drv.stop()  # and then stop (if it's still running)
    # Increment effect ID and wrap back around to 1.
    effect_id += 1
    if effect_id > 123:
        effect_id = 1
```

Speaker

```
from pygame import mixer

# Initialize pygame mixer
mixer.init(frequency=44100, size=16, channels=1)
#mixer.init()
sound = mixer.Sound('okay.wav')
while True:
            sound.play()
            #sound.stop()
```

Microphone

```
import os
import time
import busio
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
from time import sleep
# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D22)

# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
chan0 = AnalogIn(mcp, MCP.P0)
#use wav file, chan0.value
#75ksps,10bit resolution
#wav header
print('Raw ADC Value: ', chan0.value)
print('ADC Voltage: ' + str(chan0.voltage) + 'V')

last_read = 0       # this keeps track of the last potentiometer value
tolerance = 250     # to keep from being jittery we'll only change
                    # volume when the pot has moved a significant amount
                    # on a 16-bit ADC
fs = 44100
freq = 440

for i in range(10000):
    print("raw audio:" , chan0.value)
    sleep(.5)

def remap_range(value, left_min, left_max, right_min, right_max):
    # this remaps a value from original (left) range to new (right) range
    # Figure out how 'wide' each range is
    left_span = left_max - left_min
    right_span = right_max - right_min

    # Convert the left range into a 0-1 range (int)
    valueScaled = int(value - left_min) / int(left_span)
```

```
    # Convert the 0-1 range into a value in the right range.
    return int(right_min + (valueScaled * right_span))

while True:
    # we'll assume that the pot didn't move
    trim_pot_changed = False

    # read the analog pin
    trim_pot = chan0.value

    # how much has it changed since the last read?
    pot_adjust = abs(trim_pot - last_read)

    if pot_adjust > tolerance:
        trim_pot_changed = True

    if trim_pot_changed:
        # convert 16bit adc0 (0-65535) trim pot read into 0-100 volume level
        set_volume = remap_range(trim_pot, 0, 65535, 0, 100)

        # set OS volume playback volume
        print('Volume = {volume}%' .format(volume = set_volume))
        set_vol_cmd = 'sudo amixer cset numid=1 -- {volume}% > /dev/null' \
        .format(volume = set_volume)
        os.system(set_vol_cmd)
#         print("raw audio:" + chan0.value)
        # save the potentiometer reading for the next loop
        last_read = trim_pot

    # hang out and do nothing for a half second
    time.sleep(0.5)
```

LED

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(26,GPIO.OUT)
print( "LED on")
GPIO.output(26,GPIO.HIGH)
time.sleep(1)
print( "LED off")
```

```
GPIO.output(26,GPIO.LOW)
```

File Transfer using SCP

```
from paramiko import SSHClient
from scp import SCPClient

ssh = SSHClient()
ssh.load_system_host_keys()
ssh.connect(hostname='10.227.13.49',
        username='pi',
        password='elder')

# SCPCLient takes a paramiko transport as its only argument
scp = SCPClient(ssh.get_transport())

scp.put('/home/pi/Desktop/acc_image/imagez5.jpg', '/home/pi/Desktop/imagez5.jpg')
#scp.get('/home/pi/Desktop/imagex4.jpg', 'file_path_on_local_machine')

scp.close()
```

Initial Client

```
# Main Driver Code for Elder Care Wearable Device

# Simple demo of reading the MMA8451 orientation every second.
from picamera import PiCamera
import RPi.GPIO as GPIO
camera = PiCamera()
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(26,GPIO.OUT)
import board
import busio
import adafruit_drv2605

import adafruit_mma8451

# socket code

import socket
```

```
host = '10.227.13.49'
port = 5560

def setupSocket():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    print("server connected")
    return s

# Initialize I2C bus & motor driver.
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)
# Initialize MMA8451 module.
sensor = adafruit_mma8451.MMA8451(i2c)

#initialize socket
s = setupSocket()

End = 0xFDE23BC2379

#set motor strength
effect_id = 16
drv.sequence[0] = adafruit_drv2605.Effect(effect_id)

def sendPic(s, filePathC, filePathS):
    print(filePathC)
    print(filePathS)
    pic = open(filePathC, 'rb')
    chunk = pic.read()
    s.send(str.encode("STORE " + filePathS))
    s.send(chunk)
    #while chunk:
       # print("Sending Picture")
       # s.send(chunk)
        #chunk = pic.read(5120)
    pic.close()
    print("Done sending")
    #s.close()
    return "Done sending"

# Optionally change the address if it's not the default:
# sensor = adafruit_mma8451.MMA8451(i2c, address=0x1C)

# Optionally change the range from its default of +/-4G:
# sensor.range = adafruit_mma8451.RANGE_2G  # +/- 2G
```

```
# sensor.range = adafruitrom picamera import PiCamera
from time import sleep

#camera = PiCamera()_mma8451.RANGE_4G  # +/- 4G (default)
#sensor.range = adafruit_mma8451.RANGE_8G  # +/- 8G

# Optionally change the data rate from its default of 800hz:
# sensor.data_rate = adafruit_mma8451.DATARATE_800HZ  #  800Hz (default)
# sensor.data_rate = adafruit_mma8451.DATARATE_400HZ  #  400Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_200HZ  #  200Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_100HZ  #  100Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_50HZ   #   50Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_12_5HZ # 12.5Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_6_25HZ # 6.25Hz
# sensor.data_rate = adafruit_mma8451.DATARATE_1_56HZ # 1.56Hz

# Main loop to print the acceleration and orientation every second.
while True:
    z, x, y = sensor.acceleration
    print(
        "Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2".format(x, y, z)
    )
    print("before if")

    if( abs(x) >= 12):
        print("inside if")
        GPIO.output(26,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        start_time = time.time()
        for i in range(10):
            sleep(0.05)
            camera.capture('/home/pi/Desktop/acc_image/imagex%s.jpg' % i )
        for i in range(10):
            sleep(0.05)
                                    sendPic(s,    "/home/pi/Desktop/acc_image/imagex%s.jpg"    %
i,"/home/pi/Desktop/imagex%s.jpg" % i)
        camera.stop_preview()
        print("picture taken x")
        drv.stop()
        GPIO.output(26,GPIO.LOW)
        end_time = time.time()
        total_time = end_time - start_time
        print(total_time)
    elif( abs(y) >= 12):
        print("inside if")
```

```
        GPIO.output(26,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        start_time = time.time()
        for i in range(10):
            sleep(.05)
            camera.capture('/home/pi/Desktop/acc_image/imagey%s.jpg' % i )
        for i in range(10):
            sleep(0.05)
                                        sendPic(s,    "/home/pi/Desktop/acc_image/imagey%s.jpg"    %
i,"/home/pi/Desktop/imagey%s.jpg" % i)
        camera.stop_preview()
        print("picture taken y")
        GPIO.output(26,GPIO.LOW)
        drv.stop()
        end_time = time.time()
        total_time = end_time - start_time
        print(total_time)
    elif( abs(z) >= 12 or abs(z) <= 7 ):
        print("inside if")
        GPIO.output(26,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        start_time = time.time()
        for i in range(10):
            sleep(.05)
            camera.capture('/home/pi/Desktop/acc_image/imagez%s.jpg' % i )
        for i in range(10):
            sleep(0.05)
                                        sendPic(s,    "/home/pi/Desktop/acc_image/imagez%s.jpg"    %
i,"/home/pi/Desktop/imagez%s.jpg" % i)
        camera.stop_preview()
        print("picture taken z")
        drv.stop()
        GPIO.output(26,GPIO.LOW)
        end_time = time.time()
        total_time = end_time - start_time
        print(total_time)
    else:
        print("all good")
    orientation = sensor.orientation
    # Orientation is one of these values:
    #  - PL_PUF: Portrait, up, front
    #  - PL_PUB: Portrait, up, back
    #  - PL_PDF: Portrait, down, front
    #  - PL_PDB: Portrait, down, back
```

```
    #  - PL_LRF: Landscape, right, front
    #  - PL_LRB: Landscape, right, back
    #  - PL_LLF: Landscape, left, front
    #  - PL_LLB: Landscape, left, back
    print("Orientation: ", end="")
    if orientation == adafruit_mma8451.PL_PUF:
        print("Portrait, up, front")
    elif orientation == adafruit_mma8451.PL_PUB:
        print("Portrait, up, back")
    elif orientation == adafruit_mma8451.PL_PDF:
        print("Portrait, down, front")
    elif orientation == adafruit_mma8451.PL_PDB:
        print("Portrait, down, back")
    elif orientation == adafruit_mma8451.PL_LRF:
        print("Landscape, right, front")
    elif orientation == adafruit_mma8451.PL_LRB:
        print("Landscape, right, back")
    elif orientation == adafruit_mma8451.PL_LLF:
        print("Landscape, left, front")
    elif orientation == adafruit_mma8451.PL_LLB:
        print("Landscape, left, back")
    time.sleep(0.5)
```

Initial Server

```
import socket
#from cookieLED import callLED

host = '10.227.9.34'
port = 5560

def setupServer():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Socket created.")
    try:
        s.bind((host, port))
    except socket.error as msg:
        print(msg)
    print("Socket bind complete.")
    return s

def setupConnection():
    s.listen(1) # Allows one connection at a time.
    conn, address = s.accept()
```

```
    print("Connected to: " + address[0] + ":" + str(address[1]))
    return conn

def storeFile(filePath):
    picFile = open(filePath, 'wb')
    print("Opened the file.")
    pic = conn.recv(1024)
    while pic:
        picFile.write(pic)
        pic = conn.recv(1024)
        continue
    print("Done Receiving")
    picFile.close()

def dataTransfer(conn):
    # A big loop that sends/receives data until told not to.
    while True:
        print("Receiving data")
        # Receive the data
        data = conn.recv(1024) # receive the data
        data = data.decode('utf-8')
        # Split the data such that you separate the command
        # from the rest of the data.
        dataMessage = data.split(' ', 1)
        command = dataMessage[0]
        if command == 'GET':
            reply = GET()
        elif command == 'REPEAT':
            reply = REPEAT(dataMessage)
        elif command == 'STORE':
            print("Store command received. Time to save a picture")
#           print(dataMessage[1])
            storeFile(dataMessage[1])
            print("Storage Complete")
            reply = "File stored."
        elif command == 'LED_ON':
            callLED()
            reply = 'LED was on'
        elif command == 'EXIT':
            print("Our client has left us :(")
            break
        elif command == 'KILL':
            print("Our server is shutting down.")
            s.close()
            break
        else:
```

```
            reply = 'Unknown Command'
        # Send the reply back to the client
        conn.sendall(str.encode(reply))
        print("Data has been sent!")
    conn.close()

s = setupServer()

while True:
    try:
        conn = setupConnection()
        dataTransfer(conn)
    except:
        break
```

Final Source Code for Device:

Final Driver Program

```
# Main Driver Code for Elder Care Wearable Device

# Libraries
from picamera import PiCamera
import RPi.GPIO as GPIO
import time
import board
import busio
import adafruit_drv2605
import adafruit_mma8451
from paramiko import SSHClient
from scp import SCPClient
from time import sleep
from datetime import datetime
from pygame import mixer

#Initialize SCP/SSH
ssh = SSHClient()
ssh.load_system_host_keys()
ssh.connect(hostname='10.227.13.49', username='pi', password='elder') #update to new server

# SCPCLient takes a paramiko transport as its only argument
scp = SCPClient(ssh.get_transport())

#initialize GPIO Pins
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(16,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)

# Initialize I2C bus & motor driver.
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)

# Initialize MMA8451 module.
sensor = adafruit_mma8451.MMA8451(i2c)

#Initialize Camera
camera = PiCamera()

#set motor strength, effect_id = (1-123)
effect_id = 16
drv.sequence[0] = adafruit_drv2605.Effect(effect_id)

#Initialize Audio
mixer.init(frequency=44100, size=16, channels=1)
sound = mixer.Sound('okay.wav') #Add your own audio files

# Main loop to print the acceleration and orientation every half second.
# Sends pictures to server and turns on indicators

while True:
    GPIO.output(16,GPIO.HIGH) #"On" indicator
    z, x, y = sensor.acceleration
    #Prints out accelerometer data to command line
    print("Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2".format(x, y, z))

    #X-axis
    if( abs(x) >= 13):
        GPIO.output(12,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        for i in range(10): #loop 10 times when a fall is detected in the X-axis
            sleep(0.05)
            camera.capture('/home/pi/Desktop/acc_image/imagex%s.jpg' % i )
        camera.stop_preview()
        print("picture taken x")
        drv.stop()
        GPIO.output(12,GPIO.LOW)
        sound.play()
        for i in range(10): #send 10 pictures to server using SCP
```

```
            sleep(.5)
            scp.put('/home/pi/Desktop/acc_image/imagex%s.jpg' % i ,
'/home/pi/Desktop/fall/imagex%s.jpg' % i)

    #Y-axis
    elif( abs(y) >= 13):
        GPIO.output(12,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        for i in range(10):
            sleep(.05)
            camera.capture('/home/pi/Desktop/acc_image/imagey%s.jpg' % i )
        camera.stop_preview()
        print("picture taken y")
        GPIO.output(12,GPIO.LOW)
        drv.stop()
        sound.play()
        for i in range(10):
            sleep(.5)
            scp.put('/home/pi/Desktop/acc_image/imagey%s.jpg' % i ,
'/home/pi/Desktop/fall/imagey%s.jpg' % i)

    #Z-axis
    elif( abs(z) >= 13 or abs(z) <= 3 ):
        GPIO.output(12,GPIO.HIGH)
        drv.play()
        camera.start_preview()
        for i in range(10):
            sleep(.05)
            camera.capture('/home/pi/Desktop/acc_image/imagez%s.jpg' % i )
        camera.stop_preview()
        print("picture taken z")
        drv.stop()
        GPIO.output(12,GPIO.LOW)
        sound.play()
        for i in range(10):
            sleep(.5)
            scp.put('/home/pi/Desktop/acc_image/imagez%s.jpg' % i ,
'/home/pi/Desktop/fall/imagez%s.jpg' % i)

    else:
        print("all good")
#orientation definitions
    orientation = sensor.orientation
    print("Orientation: ", end="")
    if orientation == adafruit_mma8451.PL_PUF:
```

```
        print("Portrait, up, front")
    elif orientation == adafruit_mma8451.PL_PUB:
        print("Portrait, up, back")
    elif orientation == adafruit_mma8451.PL_PDF:
        print("Portrait, down, front")
    elif orientation == adafruit_mma8451.PL_PDB:
        print("Portrait, down, back")
    elif orientation == adafruit_mma8451.PL_LRF:
        print("Landscape, right, front")
    elif orientation == adafruit_mma8451.PL_LRB:
        print("Landscape, right, back")
    elif orientation == adafruit_mma8451.PL_LLF:
        print("Landscape, left, front")
    elif orientation == adafruit_mma8451.PL_LLB:
        print("Landscape, left, back")
    time.sleep(0.5)
```

Source Code for Raspberry Pi Configuration Files:

sudo nano /boot/config.txt

```
# For more options and information see
# http://rpf.io/configtxt
# Some settings may impact device functionality. See link above for details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720
```

```
# uncomment if hdmi display is not detected and composite is being output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
#dtparam=i2s=on
dtparam=spi=on

# Uncomment this to enable infrared communication.
#dtoverlay=gpio-ir,gpio_pin=17
#dtoverlay=gpio-ir-tx,gpio_pin=18

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap

[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
#dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
dtoverlay=vc4-fkms-v3d
enable_uart=1
start_x=1
```

```
gpu_mem=128

#changes made to GPIO 13 and 18
#dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4
#dtoverlay= USB Audio [USB Audio]
#dtoverlay=pwm-audio-pi-zero
```

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
	ssid="eduroam"
	key_mgmt=WPA-EAP
	auth_alg=OPEN
	eap=PEAP
	identity="bliando@okstate.edu"
	password="K@m1nsk4"
	phase2="auth=MSCHAPV2"
	priority=999
	proactive_key_caching=1
}
```

Initial and Final Schedule:

The table below is the initial action item chart that we made at the beginning of the semester. The chart lists out the task, who is responsible for each part, the start date, and when it needs to be completed.

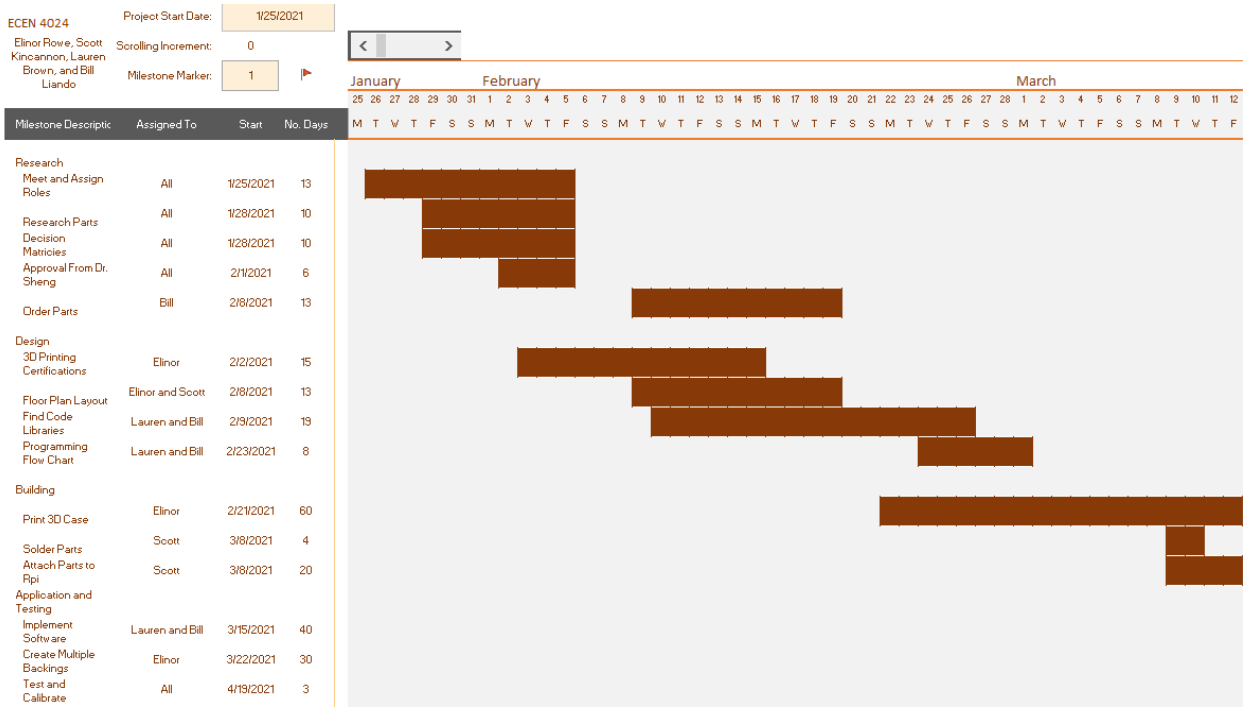
Task	Team Member	Start Date	End Date
Research Parts	All	1/28/21	2/5/21
Submit Design Proposal	All	2/1/21	2/5/21
3D Printing Certification	All	2/2/21	2/15/21
Order Parts	Bill	2/8/21	2/19/21
Floor Plan Layout	Elinor & Scott	2/8/21	2/19/21
Research Parts	All	1/28/21	2/5/21
Find Libraries for Each Component	Lauren & Bill	2/9/21	2/26/21
Create Programming Flowchart	Lauren & Bill	2/23/21	3/1/21
Debugging Programming	Lauren & Bill	3/1/21	3/15/21
3D Print Various Cases	Elinor & Scott	2/21/21	3/1/21

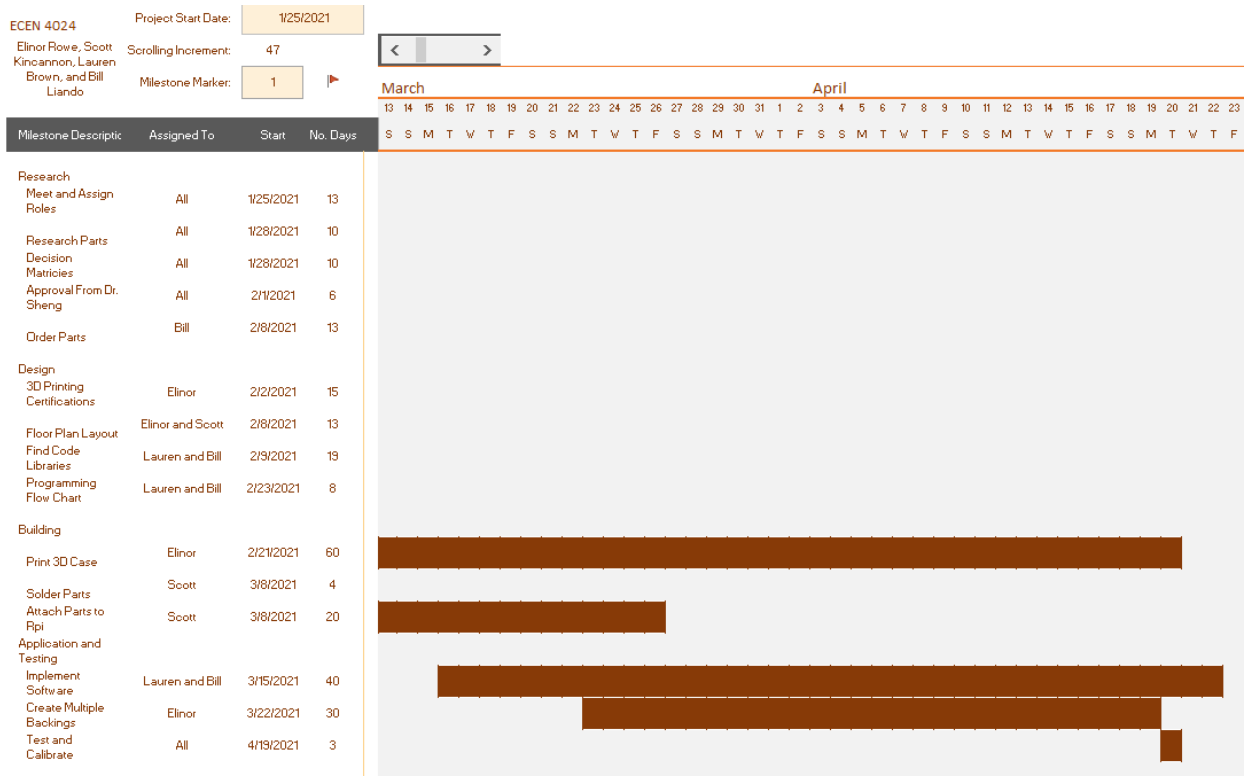
The table below is the final updated action item chart for the semester. Again, the chart lists out the task, who is responsible for each part, the start date, and when it needs to be completed. This chart's dates start after the mid-semester meeting and continue to the end of the project.

Task	Team Member	Start Date	End Date
Research Parts	All	1/28/21	2/5/21
Submit Design Proposal	All	2/1/21	2/5/21
3D Printing Certification	All	2/2/21	2/15/21
Order Parts	Bill	2/8/21	2/19/21
Test Best Location	All	2/8/21	2/12/21
Floor Plan Layout	Elinor & Scott	2/8/21	2/19/21
Find Libraries for Each Component	Lauren & Bill	2/9/21	2/26/21
Create Programming Flowchart	Lauren & Bill	2/23/21	3/1/21
Attach Accelerometer to	Scott	3/1/21	3/5/21

RPi			
Implement Accelerometer Program	Lauren & Bill	3/1/21	3/5/21
Add Ambient Light Sensor and Camera	Scott	3/8/21	3/12/21
Research Parts	All	1/28/21	2/5/21
Submit Design Proposal	All	2/1/21	2/5/21
3D Printing Certification	All	2/2/21	2/15/21
Order Parts	Bill	2/8/21	2/19/21
Test Best Location	All	2/8/21	2/12/21
3D Print Various Cases	Elinor & Scott	2/21/21	3/1/21
Connect and test all parts	Scott & Bill	2/19/21	4/23/21
Solder speaker driver and speakers	Scott	3/8/21	3/12/21
Implement software for Speaker driver	Lauren & Bill	3/15/21	3/26/21
Attach sensors to box	Scott	3/8/21	4/23/21
Test all code while inside box	Lauren & Bill	3/15/21	3/19/21
Finish First Prototype	All	3/22/21	3/26/21
Design Case Backings	Elinor	3/22/21	4/20/21
Testing Device	All	4/19/21	4/21/21
3D Print Various Cases	Elinor & Scott	2/21/21	3/1/21
Connect and test all parts	Scott & Bill	2/19/21	4/23/21
Solder speaker driver and speakers	Scott	3/8/21	3/12/21
Implement software for Speaker driver	Lauren & Bill	3/15/21	3/26/21
Attach sensors to box	Scott	3/8/21	4/23/21
Test all code while inside box	Lauren & Bill	3/15/21	3/19/21
Finish First Prototype	All	3/22/21	3/26/21
Design Case Backings	Elinor	3/22/21	4/20/21
Testing Device	All	4/19/21	4/21/21

The figure below is the Gnatt chart for the entire project. The chart lists out the milestone descriptions, who was responsible, when the start date was, and how many days it took to complete each part.





Budget summary:

The table below includes the final parts for the project and their prices. The cost of the parts used in the finished project came out to $135.87. This price can be reduced by selecting the parts from different suppliers where they would be less expensive. The department gave us a budget of $200. Since ordering through the university took longer than desired, the majority of the final parts were paid for by the undergraduate students. Only $131.77 was spent from the budget, while $232.05 was spent by the students. This differs from the cost of the final project because many parts were replaced as the design evolved.

	Quantity	Price	Description
Raspberry Pi Zero W Camera pack	1	$61.81	Kit that includes microcontroller and camera
MMA 8451	1	$7.95	Accelerometer
Micro SD Card	1	$16.34	Micro SD Card for Raspberry Pi Zero W
2W 8Ohm Speaker	1	$3.73	2w speaker
DZS Elec 1027-L35-4	1	$1.38	vibration motor
DRV2605L	1	$16.79	motor driver
MAX98357A	1	$8.59	Speaker Driver
MCP3008	1	$3.15	ADC
MAX4466	1	$4.20	Microphone Driver
FST18650·2000mAh	1	$6.20	Battery
Micro USB Cable	1	$4.99	Micro USB Cable
LED	2	$0.74	LED's
Total		**$135.87**	

Things to Change and Future Ideas:

What we would have done differently:

- Choose a microcontroller with I2S pins or with two PCM channels to allow for easier implementation of components
- Figure out a way to attach the parts without USB since the cords caused issues with space in the case design.
- Use a 3350mah cylindrical battery with a similar 3.3V-5V converter and charging PCB to allow the device to run for longer

Future Ideas:

- Design our own drivers and surface mount components to reduce space
- Create a server powerful enough to process image dataDesign our own drivers and surface mount components to reduce space
- Have the server include a display interface with the user (i.e. TV)
- Wireless charging

- Use the Arduino Nano 33 IoT
- Move to a USB microphone that includes its own driver
- Remove micro usb connection used for power and solder wires to cut down on space from cables

References:

[1] Ada, Lady. "Adafruit DRV2605L Haptic Controller Breakout." *Adafruit Learning System*, learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout/python-circuitpython.

[2] Ada, Lady. "Adafruit MMA8451 Accelerometer Breakout." *Adafruit Learning System*, 30 June 2014, learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/python-circuitpython.

[3] "Getting Started with the Camera Module." *Projects.raspberrypi.org*, projects.raspberrypi.org/en/projects/getting-started-with-picamera/5.

[4] Khalid, Muhammad Yasoob Ullah. "Yasoob Khalid." *Connecting Raspberry Pi to Eduroam Wifi*, 14 Apr. 2019, yasoob.me/posts/raspberry-pi-eduroam-wifi/#:~:text=If%20you%20are%20not%20familiar,automatically%20out%20of%20the%20box.&text=Raspberry%20Pi%20uses%20wpa%2Dsupplicant,wpa%2Dsupplicant%20has%20a%20wpa_supplicant.

[5] Linuxize. "How to Use SCP Command to Securely Transfer Files." *Linuxize*, Linuxize, 30 May 2020, linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/.

[6] Morganti, Alessio. "Raspberry Pi Zero, Audio Output via I2S." *Lucadentella.it*, 26 Apr. 2017, www.lucadentella.it/en/2017/04/26/raspberry-pi-zero-audio-output-via-i2s/.

[7] *Projects.raspberrypi.org*, projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2.

[8] *Python Programming Tutorial: Getting Started with the Raspberry Pi*, learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-2-play-sounds.

[9] "Tutorial 30: Transfer & Backup a File Using Socket File Transfer." *The Zan Show*, thezanshow.com/electronics-tutorials/raspberry-pi/tutorial-30.

[10] "Using Light Sensor Module with Raspberry Pi." *UUGear*, www.uugear.com/portfolio/using-light-sensor-module-with-raspberry-pi/.

Appendix A:

Weekly Progress Reports

- **1/25/20**
  https://ascc.okstate.edu/content/projects.html

- **1/26/2021**

Weekly Advisor Meeting

Attendance:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon
- Dr. Weihua Sheng

Meeting Minutes:
This meet was an introduction for Dr. Sheng and the Wearable Device team. We covered the requirements of the Elder Care Device. The device is required to have:

1. Low power microprocessor
2. Camera to detect predetermined activities such as eating
3. Mic/Speaker to facilitate interaction between user and the Personal Assistive Robot

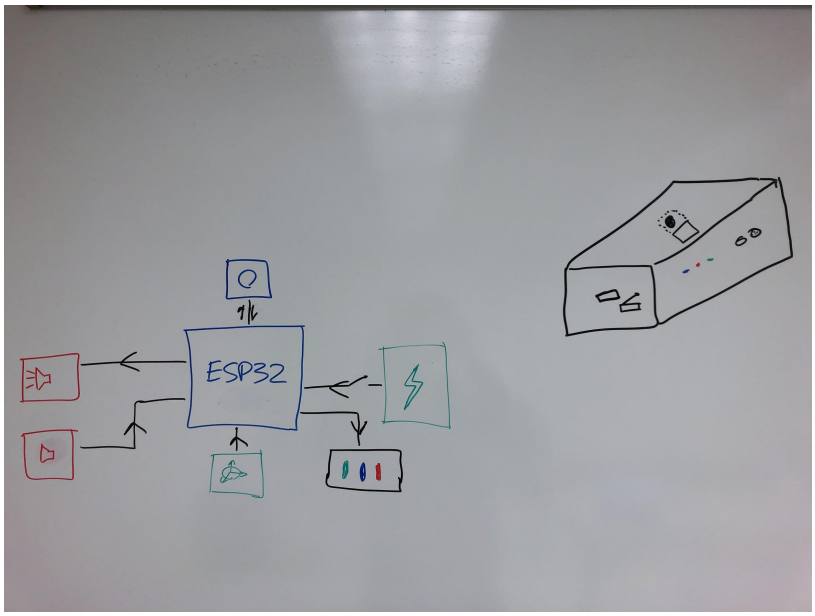
The current prototype includes these features already. The Wearable Device team will need to also include other features such as:

1. Wifi/Bluetooth option to facilitate communication between our device and the PAR(Personal Assistive Robot)
2. Risk Analysis- Determining where the device will be located
    a. Headset
    b. Hat clip
    c. Necklace
    d. Chest clip
3. Rechargeable battery with a 16 hour lifespan
4. LEDs with coded device states
5. Case to house the device electronics
6. Accelerometer to detect falls
7. Programming- Serial Link/Check in connection

- **1/28/2021**

Discussed each part that would be added to the device. Covered microcontroller, camera, speaker, microphone, accelerometer, battery pack, and LEDs. Started work on our design rough draft to be submitted the next week. Reference Design Submittal Rough Draft for what was accomplished during the meeting.

Rough design of circuit and storage compartment



- **2/1/2021**

Finished our design rough draft, reference Design Submittal Rough Draft. Created a broad list of action items for our team to accomplish this semester. Submitted the design rough draft to Dr. Sheng for review.

- **2/2/2021**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon
- Dr. Weihua Sheng
- Ricky Hernandez

Meeting Minutes:
We discussed our initial plan for our design proposal and received feedback on how the report should be structured as well as our plans up until our next deadline. We agreed to have our final design proposal submitted by Friday February 5th, 2021.

Feedback from Dr. Sheng and Ricky:
- Dr. Sheng and Ricky think it is redundant to have the Esp-cam as well as the Esp-wroom. They believe that we should add a camera to the wroom for less power and less complexity.
- Asked about other microcontrollers that we looked up, didn't seem convinced that the esp32 should be automatically used since the current prototype uses this microcontroller (computational power, power efficient, small, more integration)
- Battery chosen is too big, needs to think more about the weight, smaller the better but still needs to power everything, maybe if we only used one microcontroller we would use less power. Size of the battery should be flat like chocolate
- Have more students researching everything, more group research on main components
- Need backups if main fails

Notes about further developing Device from past versions:
- Arm is more power efficient
- Adding Ambient light sensor to detect if person is near a window
- How to drive the speaker.
- Reminder for wearing in the morning, sound, light, vibration
- Lightweight device
- How to wear it

Proposal Requirements:
- Need a proposal, with diagrams, interface, need light sensor.
- Need to know what each student is going to do, have a list of each tasks
- Calculate power consumption

- **2/8/21 Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Meeting Minutes:
- OSU Snow Day, push 3D Printing Training to Wednesday and Monday
- Next steps is Ordering parts and determining best location for device
- Battery
    - Current one is too large
    - Prototype will current one, find smaller one if needed later for final prototype
- Speakers
    - Use current drivers, can use alternative speaker with an adapter
- Created Parts list, will fill out more after meeting with Dr. Sheng

- **2/9/21 Advising Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon
- Dr. Weihua Sheng

Meeting Minutes:
We discussed our Design Proposal and proposed new battery ideas.

Device Components:
- What type of charger for battery
- Plans for voltage upconverter (3.7V to 5V)
- Plan for the speaker driver weight, can be too heavy
- Ambient light sensor
    - Is it able to interface with Raspberry Pi?
        - Analog/Digital
- Vibration for alert system
- Power Supply, How long can the battery last
    - Longest lifespan with lightest power consumption mode
    - Shortest lifespan with highest power consumption mode
- Order 1200mAh battery for backup
- LED lights
- Find existing libraries for each component to interface programming with Raspberry Pi

- **2/11/21 Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Meeting Minutes:
- Update on ordering parts
- PCB Board if needed
- 3D Printing Certification update due to bad weather
    - Work on parts before certification
- Location for device - Armband, belt clips
- Vibration alert - coin motor to vibrate
    - https://www.mouser.com/ProductDetail/Adafruit/1201?qs=sGAEpiMZZMu%252BmKbOcEVhFUbhUd4riIKNQ8QK%252BzDNq2UMFL5y%252B9Z6cA%3D%3D

- **2/15/21 Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Meeting Minutes:
- Present new vibration motor to Dr. Sheng
- Parts are in transit

- **2/17/21 Advising Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon
- Dr. Weihua Sheng

Meeting Minutes:

- Vibration motor, needs a lot of current
    - Need transistor to boost current
- How to attach a device.
    - Shoulder Mount
    - Chest Clip
    - Rectangular prism to have multiple attach points
- Raspberry Pi Operating System
    - Raspbian vs Raspberry Pi OS vs Ubuntu

Weekly Summary:
- Ordered Parts
- Still waiting on 3D Printing Certification (Snow Days)

- **2/18/21 Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Meeting Minutes:

- Bill ordered personal Raspberry Pi Zero W
- Layout presented to Dr. Sheng on Tuesday
- Have Programming Libraries and wiring schematics for Tuesday

- **2/23/21 Advising Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon
- Dr. Weihua Sheng
- Ricky Hernandez

Meeting Minutes:

- Schematic Box Completed
    - Currently Printing
    - 138x66x35mm
    - Add room for future additions
        - Wiring
    - Board Mount
- Pi Zero W specs
    - 512MB RAM
    - Using Raspberry PI OS
- Programing
    - Flowchart
    - Server/Client

Weekly Summary:
- Everyone has been 3D Printing Certified at Endeavor
- We have ordered a personal Pi Zero W to practice programming on and to use to compile programs while the hardware is being installed
- Program Libraries for all the parts have been found
- Floor plan design completed. Began printing our first casing prototype

- **2/25/21 Meeting Notes**

Attendance:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Meeting Minutes:

- Delay first Prototype?
    - Use personal Pi and ambient light sensor
- new case design
    - 80x71x48mm
- Pi HAT V1.0
    - May not work because it uses all the pins
    - Create our own speaker driver (PCB)
- March 2nd, prototype demonstration
    - Use ambient light sensor and LED Indicators
    - Program in Python

Week 6 Progress Report (2/22 - 2/28)

Team Members:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary
- Created a presentation
- Parts have been delivered
- Begin assembly and uploading code
- Wifi working at home but not on school's wifi (eduroam)
    - Contacted Zhidong and got help on connectivity and will schedule a meeting to troubleshoot
- Accelerometer and Ambient Light Sensor have been installed
- Replaced Re-Speaker with a PCB Speaker
    - Designed and ordered PCB

- **Week 6 Progress Report (3/1 - 3/5)**

Team Members:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary

- Wifi issue has been resolved
- PCB for speaker has been delivered and now ready for assembly
- The camera module and the accelerometer has been installed
    - Camera is alerted when accelerometer reaches a certain threshold in all three directions
- Microphone has been researched, have three options that we are deciding between
- Alternate power supply
    - Portable battery
        - 0.8 oz extra weight
        - 3.7” x 0.9” x 0.9” dimensions
        - More power capacity: 2500mAh increased to 3350mAh
        - Simpler due to fewer parts
        - Safer due to built in safety measures in the power bank
            - Less exposed elements
        - More compact case design
        - Easily replaceable due to parts already built

- **Week 8 Progress Report (3/8 - 3/12)**

Team Members:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary
- Microphone ordered
- 2 batteries ordered
    - Opened one, integrating into case
    - Contacting Anker for internal specifications
- Camera takes 10 pictures at 50ms intervals, around 20fps
- Vibration Motor
    - https://learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout/python-circuitpython
    - DRV2605L
        - Driver Only
- Speaker
    - Connected to Raspberry Pi
    - Problems with initializing Raspberry Pi Configuration
        - Only audio option is HDMI output
- Begin working on creating a Server/Client connection between the two Raspberry Pi's.

- **Week 9 Progress Report (3/15 - 3/19)**

Team Members:
- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary
- Socket created between the RPi
- RPi able to send pictures between each other
    - Working on integrating feature with main program
- Battery casing has been removed and still functioning
- Microphone has been delivered
- Designing multiple case backings

- **Week 10 Progress Report (3/22 - 3/26)**

Team Members:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary

- Parts Ordered:
    - 3 Watt, 8 Ohm 2 pin speaker
    - Motor
    - Micro-usb to usb cable
- Parts Delivered:
    - Motor Controller
- Task Completed:
    - Able to output from speaker, not loud enough(0.5 watts)
        - Ordered a 3 watt speaker
    - Ambient Light Sensor is working now
        - binary output, can modify the sensitivity
- Task in Progress:
    - Microphone is functional
        - Cannot verify it works with current speaker
    - Client camera uploading multiple images to Server
    - Case Design
        - New speaker

- **Week 11 Progress Report (3/29 - 4/2)**

Team Members:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary

- Parts Ordered:
    - ADC board
- Parts Delivered:
    - 3 Watt, 8 Ohm 2 pin speaker
    - Motor
    - Micro-usb to usb cable
- Task Completed:
    - LED indicator light
        - Lights up when camera is on
    - 3W speaker works but it needs to be louder
        - Replace with full micro-usb speaker
        - Audio amplifier circuit
    - Client camera sending multiple images to Server
- Task in Progress:
    - Designing audio amplifier for speaker
        - Constraint, amplifier needs to much supply voltage
    - Microphone is functional
        - Cannot verify it works with current speaker
        - ADC is on its way
    - Case Design
        - New speaker
    - Vibrating Motor
        - Parts arrived and now we're building circuits
    - Battery
        - Avoid soldering Lithium battery

- **Week 12 Progress Report (4/5 - 4/9)**

Team Members:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary

- Task Completed:
    - Cases
        - New case design with hinges and latch system
        - 3D printed prototype
    - Vibration Motor
        - Installed and integrated into program
    - Speaker
        - Works with micro-usb and PCM pins (I2S)
            - PCM has smaller size than usb set-up
    - Microphone
        - Only measures volume

- Task in Progress:
    - Battery
        - Looking at different variations
    - Final Case
        - Get all parts soldered together first
    - Heartbeat connection server/client

- **Week 13 Progress Report (4/12 - 4/16)**

Team Members:

- Elinor Rowe
- Bill Liando
- Lauren Brown
- Scott Kincannon

Weekly Summary

- Task Completed:
    - Assembled all the components into one piece
    - Presentation for final demonstration

- Task in Progress:
    - 3D Printing final case
    - Prototyping testing
        - Getting values and fine tuning project