PIPE (Pi for Parkinson's Effort)

By Eric Xiao, Jeromey Klein, Howard Wang

UCSD IEEE Quarterly Projects Team 19

Introduction

"If only you had $35 to change someone's life, how would you do it?"

That was the first question each of us in Team PIPE asked ourselves before started this project. The premise was simple; given a small budget, design a usable tool for patients of Parkinson's Disease that could potentially improve their quality of life, or be a life-changing invention. For our team, this was an opportunity to apply what we learn as students of UCSD in creating a genuinely useful and helpful project. For the participating patients, this was an opportunity to receive any amount of help they can get for performing simple, everyday tasks made harsh by their neurotic condition. As students, an opportunity such as this was rare, so as a team we were even more determined to create something amazing. However, in order to answer the question we asked ourselves at the beginning, we had to do some preliminary research.

Project Link

https://acsweb.ucsd.edu/~luw055

Code

https://github.com/hwanggit/Assistive-Spoon-for-Parkinsons

Preliminary Research

What causes Parkinson's?

PD is a type of movement disorder, happens when nerve cells in the brain don't produce enough of a brain chemical dopamine. Sometimes it is genetic, but most cases do not seem to run in families. Exposure to chemicals in the environment might play a role. [1]

What symptoms are caused by Parkinson's?

Trembling of hands, arms, legs, jaw, face. Stiffness/mild paralysis. Slowness of movement, poor balance and coordination. [1]

What daily/common struggles are caused by the symptoms of Parkinson's?

'Freezing' (also called motor block) is a difficulty that people with Parkinson's disease encounter often, making them unable to start a movement or continue repeat movements. It mostly affects walking but can also affect speech or writing. There are several tips that can help out when these episodes occur such as counting out loud or stepping over a specific target on the ground. For the last one a laser light pointer or a cane can be used. Suffering from Parkinson's disease can have an impact on one's driving. Thus, if you are a driver you have to inform the Driving and Vehicle Licensing Agency and your insurance company about your condition. [2]

What are existing technologies used to diagnose Parkinson's?

Parkinson's Voice Initiative:
Uses machine learning to record and train system to recognize parkinson's patient voices, they want to record over 10,000 voices. Tests population-wide, so may detect the disease before it worsens. There is no need for clinical trials and it will be cost effective, optimized for individual use, and for drug treatment especially. [3]

Other methods of diagnosis conducted by neurologist:
Medical history, review of symptoms, neurological/physical examination, specific single-photon emission computerized tomography SPECT scan called a dopamine transporter (DAT) scan Imaging tests — such as MRI, CT, ultrasound of the brain, and PET scans — may also be used to help rule out other disorders. Imaging tests aren't particularly helpful for diagnosing Parkinson's disease. In addition to your examination, your doctor may give you carbidopa-levodopa (Rytary, Sinemet, others), a Parkinson's disease medication. You must be given a sufficient dose to show the benefit, as low doses for a day or two aren't reliable. [4]

What technology already exists to assist Parkinson's patients?

Deep brain stimulation:
A surgical procedure that implants electrodes into part of the brain and connects them to a small electrical device implanted in the chest. Using a mild pulse of current, DBS delivers stimulation to specific areas of the brain responsible for PD symptoms, and effectively "shuts off" the characteristic tremor patients experience. The DBS patient programmer uses a wireless iOS mobile platform. In essence, your smart phone becomes a personal programmer that gives a patient or caregiver the unique opportunity to turn the system on and off, and to adjust setting within limits set by a physician. Early PD patients who received DBS therapy along with medication were less likely to develop new tremors than those patients treated with drug therapy alone. [5]

Project Emma:
The Emma Watch technology introduces a rhythmic vibration effect through small motors around the wrist. While the specific therapeutic mechanism is still unknown, one theory suggests that the ability to move is regulated by a sensorimotor feedback loop, involving the perception of movement and position of the body. The tremor symptom could arise from an erroneous feedback loop, where the brain is overcompensating for an initial movement error, resulting in a continuous tremor. The injection of vibration by the Emma Watch introduces white noise that short-circuits this erroneous feedback loop, stopping the brain from sensing the initial error and trying to overcompensate. [6]

Eatwell:
According to the inventor, Eatwell is a "tableware set with a very user-centered design that helps to increase food intake and maintain dignity for its users, while also helping to alleviate caring burdens by making the process of eating as easy as possible." The design won first place at the 2014 Stanford Design Challenge [7].

Dragon – The leading voice to text software solution. Voice-to-Text for everyday notes, memos, anything. [8]

Talking Book Library – This is the Library of Congress national library service for the blind and physically handicapped. They provide audio, braille and large print books to keep you connected to the stories you love to read. And they have libraries everywhere.

Path Finder - a shoe attachment that provides visual cues to help people with unsteady and irregular gait. This is particularly helpful to people with neurodegenerative diseases such as Parkinson's disease, who commonly suffer from Freezing of Gait (FoG), a symptom causing an individual to feel as if frozen to the ground. Path-Finder can also be used to help alleviate shuffling and we have begun testing for stroke rehabilitation as well as exploratory work into other various movement disorders. [9]

Our Inspiration

Liftware -

Liftware is an assistive electronic spoon that uses a microchip and sensors to detect the direction and force of a user's tremor, before motoring the spoon in the opposite direction to cancel out the movement as best it can. [10]

Our Idea

The initial sketches of our idea involved recreating Liftware, but this time with much less expensive, off-the-shelf materials. We were inspired by Liftware, but its pricing margin was too great for financially troubled patients and their families. Hence, we proposed a simpler solution using three simple electronics: Raspberry Pi CPU, servo motors, and an Inertial Mass Unit that can detect the user's axial motions in space. These were the initial points of consideration for our brainstorming:

- Assistive grip module that can be attached to objects handles and correct for tremors.
- Should not be limited to just tableware, can be attached to anything with a rod handle (forks, spoons, knives, pens).
- Main driver of the program will be the interfacing between RPi 0 W and the Microservo motors connected to its GPIO pins.
- Sensor data will be read from an IMU to correct for skewed or shaking motions.
- Device usage analytics will be sent to a live web app via RPi 0 W Wifi for analysis by Doctors, Caregivers, family members, etc.

Materials and Outline

Materials

Here was our list of materials, all sticking to our motto of "affordable yet highly functional", and an overall budget of $35.

- Raspberry Pi 0 W including 20x2 GPIO pins. Total Cost: $10.00. Vender: Amazon.
- 3x SG90 Micro-servo motor. Total Cost: $10.50. Vendor: Adafruit.
- Inertial Measurement Unit (IMU) (4 deg freedom). Total Cost: FREE. Vendor: IEEE QP Committee UCSD 2019.
- 4x Female to female wires. Total Cost: FREE. Vendor: IEEE QP Committee UCSD 2019.

- Monitor, Keyboard, Mouse, charger and 8GB MicroSD card for setting up RPi. Total Cost: FREE. Vendor: IEEE QP Committee UCSD 2019.
- 3-D Printed handle, hardware cases, and fake utensils. Total Cost: FREE. Vendor: UCSD DML 3-D Printing Lab.

Timeline

As UCSD Students, all of us were on a time budget, so we had to allocate time towards the project on a weekly basis. We met every Tuesday morning to update each other on our respective progress, and how well we met our quotas for the week, and we used IEEE work-a-thon hours to build as a group and actually use the provided materials to interface the hardware and software portions of our idea. Overall, this was our 9-week timeline that we stuck to:

- Week 1: Research the causes, symptoms, struggles of Parkinson's Patients, and create a human-centered design idea, a potential project.
- Week 2: Brainstorm designs for the assistive module, think of an overall plan of how to use each hardware component and research how to interface hardware components using software. Use work-a-thon hours to gather hardware and start building. Start by soldering pins onto the Raspberry Pi 0 W.
- Week 3: Start interfacing RPi, IMU and Microservos using Python (Try to make the motor move with the IMU). This required importing Python modules, figuring out how to wire the IMU to the RPi, and use it to control servos.
- Week 4: Start initial sketches for 3-D CAD designs using SolidWorks. Use SolidWorks servo and pi templates to get accurate sizing of casings.
- Week 5: Print initial prototype of servo arm casings, and servo casings. Fix sizes if needed. Test RPi to IMU interfacing with more than one motor, and correct jittery motions.
- Week 6: Print second casings prototypes, and print prototype two of handle, including model fake utensils.
- Week 7: Figure out how to link IMU data from RPi to live web app that tracks the stability of the spoon and gives patient medical information. Start working on GUI.
- Week 8: Print final prototype of casings and join all components together to get a working utensil. Finish GUI for web app, and link IMU data to Firebase (online database) via Python, so it can be transferred to the web app.
- Week 9: Fix any finer details of the project and start using it in actual eating tests. Clean up web app GUI and host onto web server so users can see live data. Prepare project for showcasing.

Contributions and Tasks

Overall, we delegated tasks to each of our team members effectively so it would play to each of our strengths:

- Eric Xiao (Python expert) - Having ample experience with object-oriented programming, project-building, and effective at brainstorming ideas, Eric was the one who narrowed our idea down to a single, buildable project. He was the one who wrote and tested the initial Python program that would be used to interface the RPi to the IMU and finally to the servo motors. Note: impeccable mathematics knowledge.
- Jeromey Klein (The Designer) - Starting with nothing but templates, Jeromey was instrumental in designing all of the hardware casings and designing the overall look of the Pi-embedded device. Having substantial knowledge of SolidWorks, the only person on our team who was capable of creating complex designs in a short few weeks was Jeromey.
- Howard Wang (The Technician) - As a web developer, and expert in hardware soldering, Howard Wang played a huge role in soldering the GPIO pins onto the RPi, configuring the RPi to be able to run the program without a monitor or keyboard, linking all of the data from the IMU to Firebase, and finally developing the web app that would provide a user interface to our project.

Figure 1: Eric Xiao working on Python code

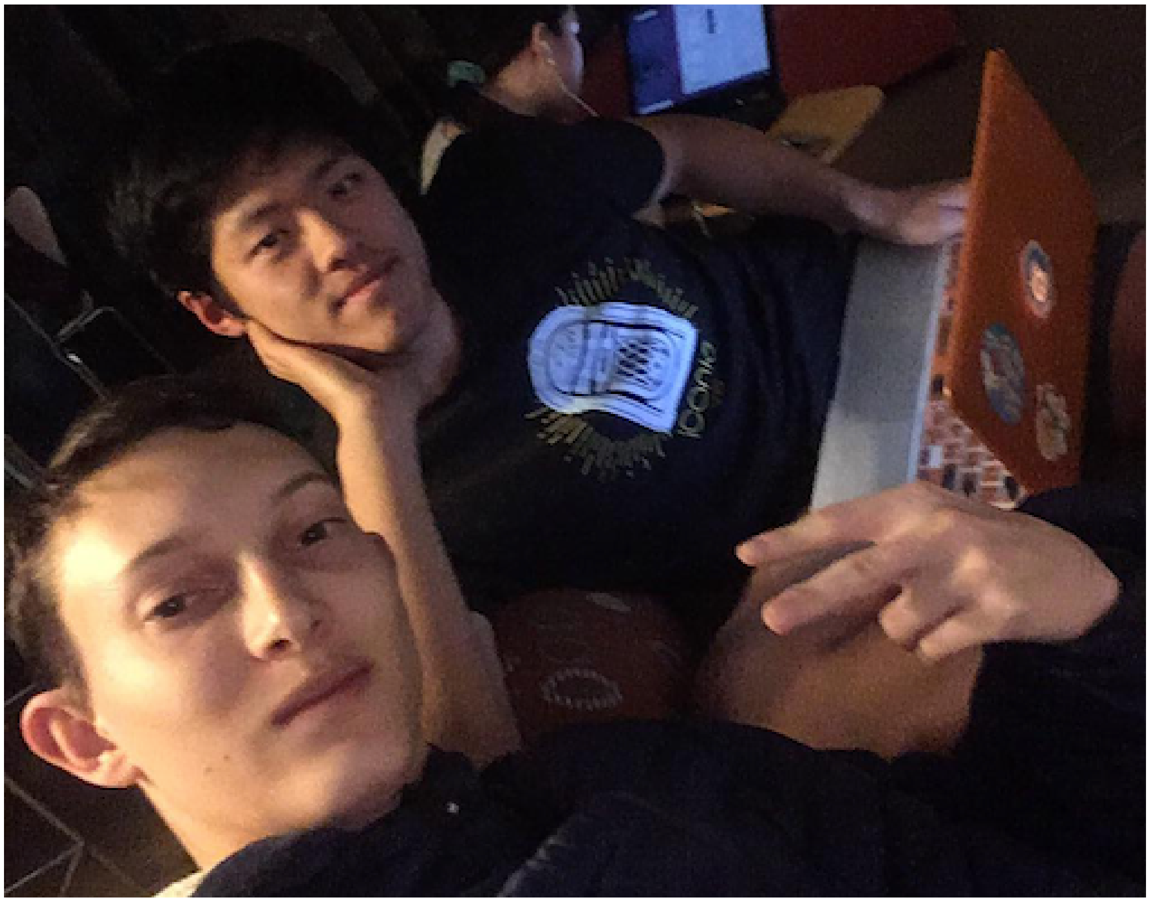


Figure 2: Jeromey (Left) and Howard (Right) preparing to film a short demo video

Guide: Our Process

Here is a step-by-step guide of the whole process we went through, from interfacing the IMU to the RPi, to designing the casings and the prototypes, and finally linking all the data to a database and web app. Enjoy!

Step 1: Set up the RPi

For this step, simply visit this link and follow the step by step instructions with the necessary materials you already have to setup a new Raspberry Pi 0 W. You will need a monitor, keyboard and mouse to program the RPi, and connect it to WiFi we recommend using a USB cable extension that has multiple USB inputs to connect the keyboard and mouse. For the monitor, use an HDMI to MiniHDMI cable. This is a preliminary process so we will not be going into much detail.

Link: https://thepi.io/how-to-install-raspbian-on-the-raspberry-pi [11]

Step 2: Solder GPIO pins onto the Pi and Wire the IMU

First, using a soldering iron, solder the 20x2 GPIO pins onto the RPi. Like so:

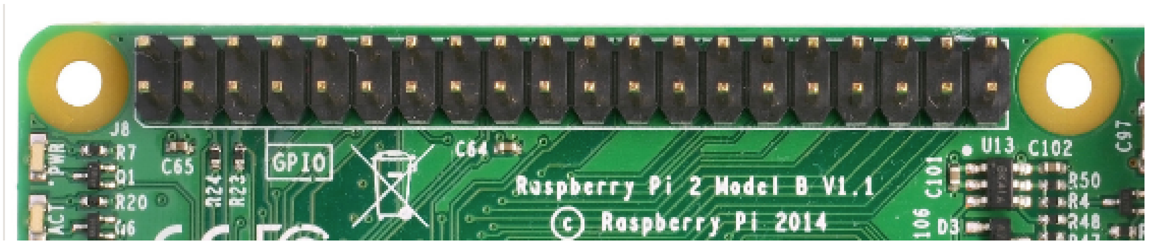


Figure 3: Raspberry Pi 0 W 20x2 GPIO Pins

Then using 4 Female to Female wires, connect the VDD, GND, SCL, SDA pins of the IMU to the correct GPIO pins on the Raspberry Pi. You can use this image from http://ozzmaker.com/berryimu [12] as a reference:

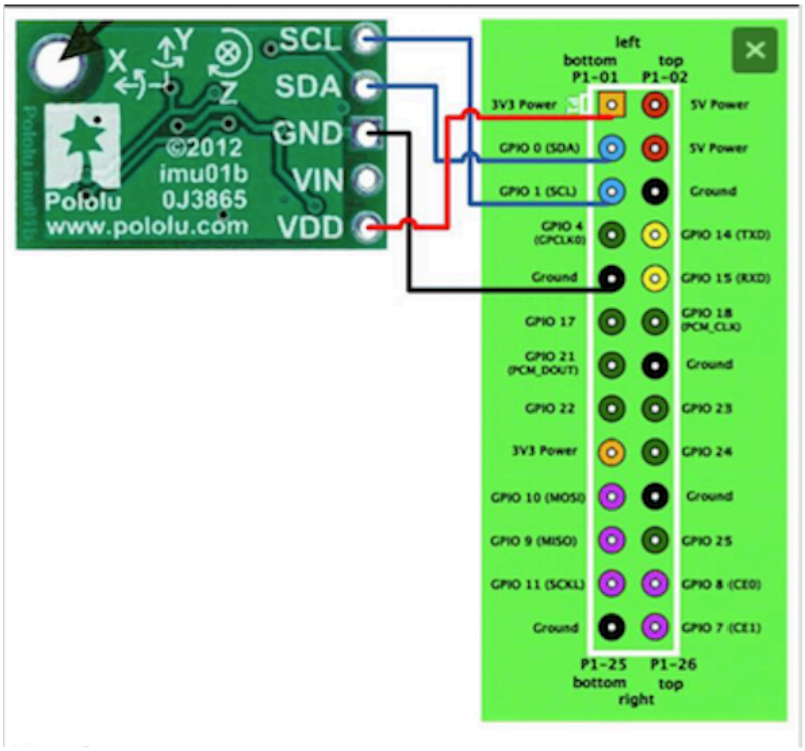


Figure 4: IMU to RPi Connections

Step 3: Boot up the Pi and check if IMU is connected properly

In order to check if your IMU is connected properly, you will need to boot up the RPi, and run:

```
sudo raspi-config
```

Then select advanced options, and enable I2C detection. After this, run:

```
i2cdetect -y 1
```

You should get the value of input registers connected to the RPi, and one address will be displayed, 0x68, which means your IMU is connected properly. If this is not the case, switch out your wiring, or switch out your IMU.

Step 4: Connect the servo motors to the RPi pins

Using the image you saw above as a reference, connect 2 servo motors to the RPi, first one on GPIO 17, and second on GPIO 23. Both servos should be connected to the 3V power pins while the IMU requires the 5V power pin. Basically, connect the brown wires to Ground Here is the final result:

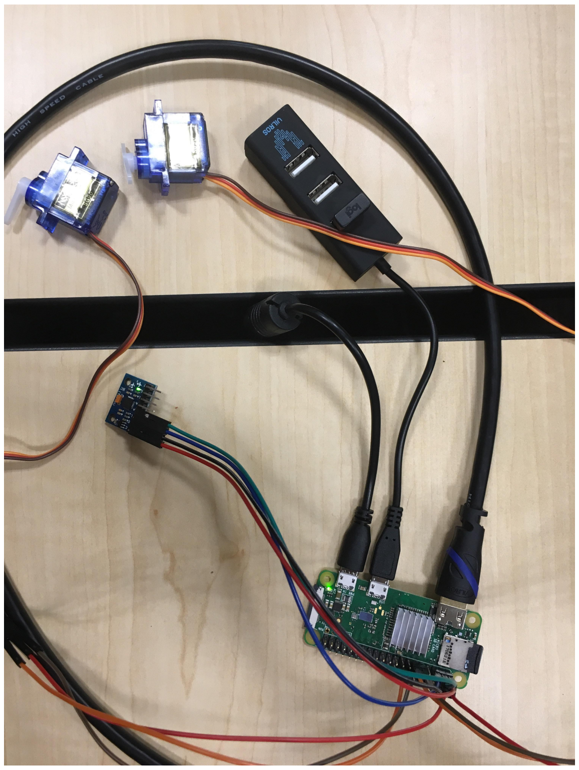


Figure 5: Entirely connected setup

Step 5: Download the control.py file from Github

From our team Github repository (https://github.com/hwanggit/Assistive-Spoon-for-Parkinsons) download the control.py file, and read all the requirements for Python modules necessary to run this program in requirements.txt. Example run:

```
git clone https://github.com/hwanggit/Assistive-Spoon-for-Parkinsons
```

Then open the file and put the control.py file in your RPi Home directory and then start installing the necessary Python modules listed in requirements.txt. For example, run:

```
pip3 install <NAME OF MODULE>
```

In your RPi Home directory for every required module.

Step 6: Run pigpio API and then control.py to test the program

This should be the easiest step. In your RPi home directory simply run:

```
sudo pigpiod

./control.py
```

NOTE: If you want the device to read and post realtime data values, first connect the RPi to local WiFi, setup a Firebase account, and replace the file's API credentials with your own, otherwise leave the lines with Firebase functions commented out.

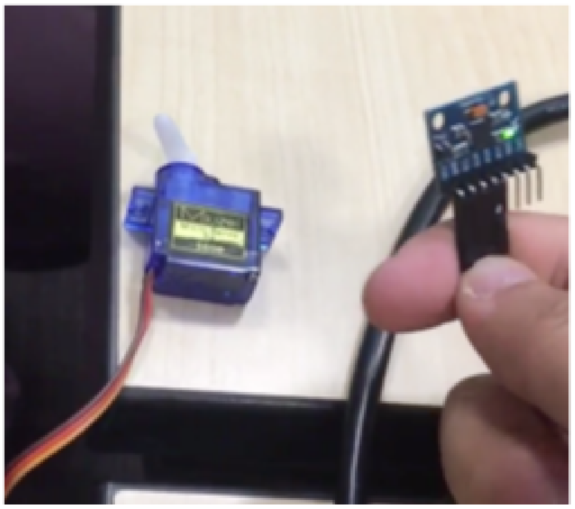


Figure 6: IMU interfacing with servos

The program should start reading values of servo1, and the servos should rotate in the opposite direction in which you move the IMU sensor.

Step 7: Design all of the 3-D casings and utensil holding

> *" We are trying to build something that is small and compact and the precision required to implement our design meant 3-D printing custom parts was necessary. "* - Jeromey Klein

We recommend using SolidWorks, because it provides you with templates of all the hardware we used so you can use those templates as a mold for your casings. Make sure you have 3-D printer available, and up to 8 hours of time to print. Here are the design steps we took, with descriptions of the process by Jeromey Klein:

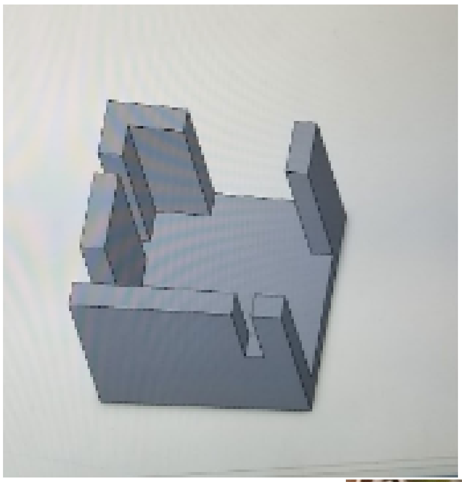
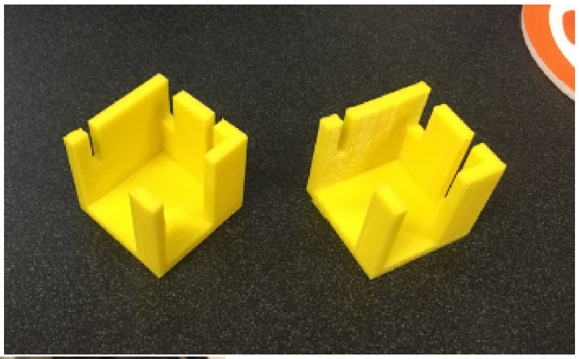
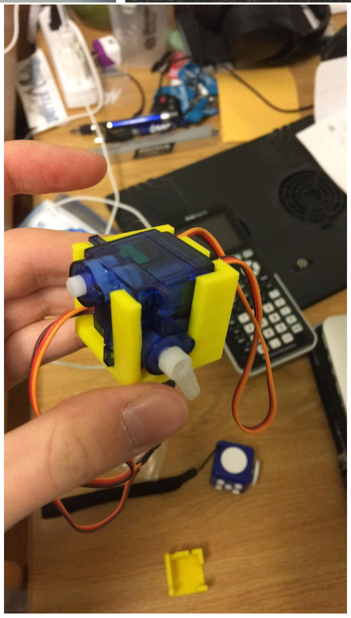




Figure 7: Sketch and Design of servo casing

To create the servo casing we took measurements of the 3D servo models and made the cuts accordingly. We built around the extrusions on the side of the servos that allow it to attach to another part. That is why there are strategically placed cuts in the sides of the cube. One servo was slightly larger than the other, which complicated the design. We ended up having to print another design that was slightly larger.

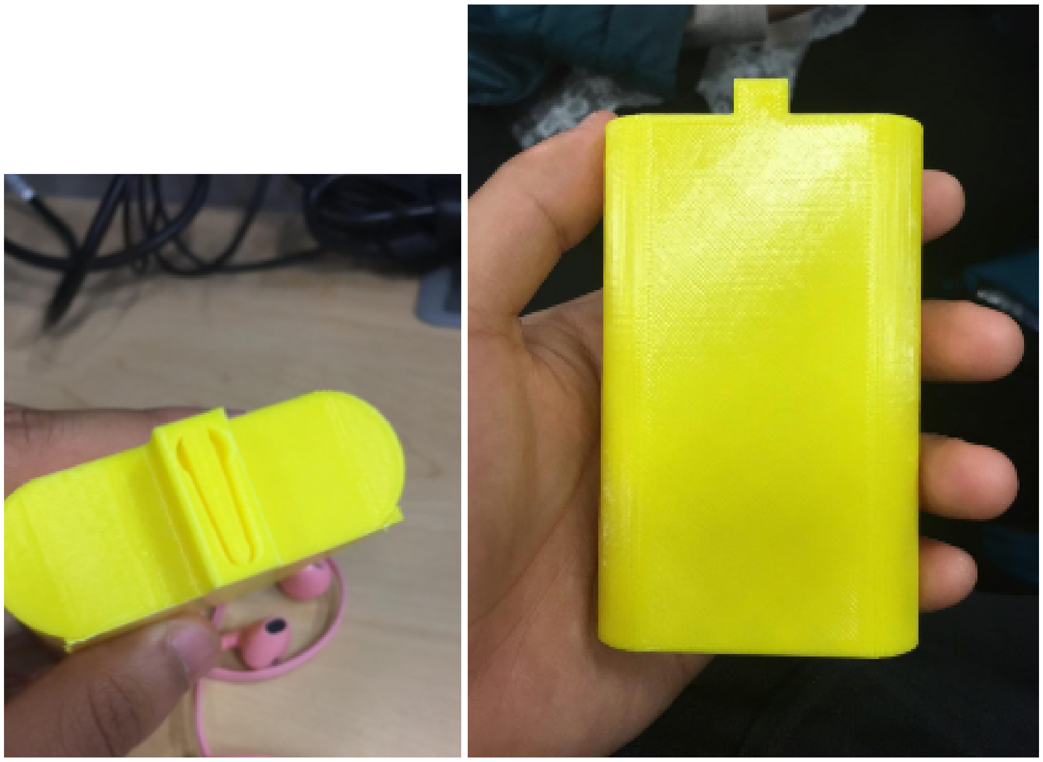


Figure 8: Sketch and Design of servo casing

The first iteration of our raspberry pi case was unsuccessful because the propeller adapter was far too small. So we decided to just print the adaptive part so we could be certain that it would fit.

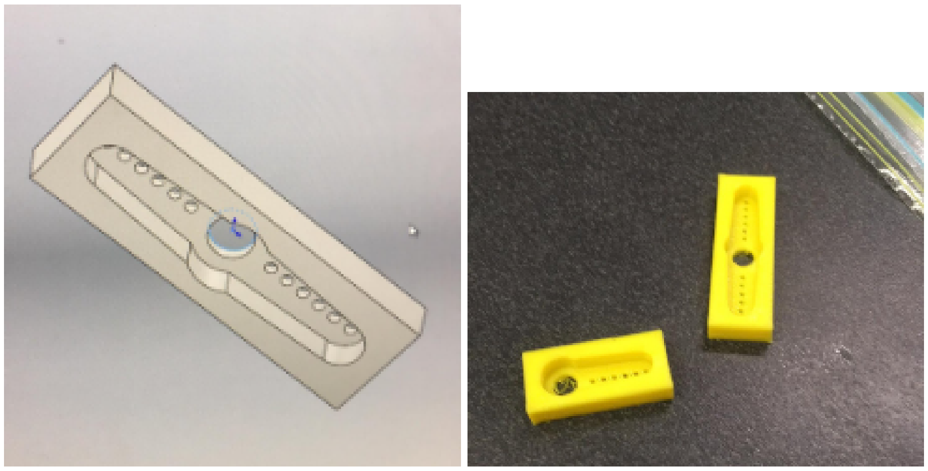


Figure 9: Servo Arm Casings

The spoon connector is meant to attach the spoon to the propellor the servo that rotates on the yaw axis. The cut is strategically designed for the attachment of any piece of silverware or more generally any utensil such as a pen that requires one to grip it with their hand.

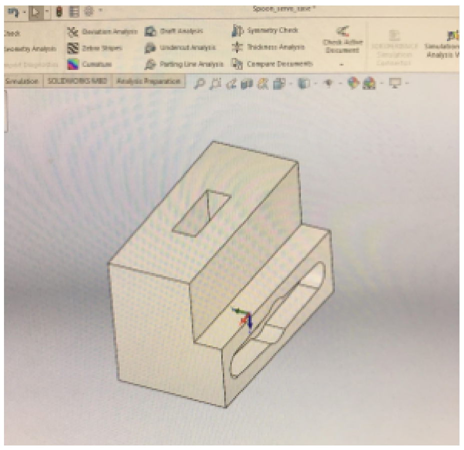


Figure 10: Utensil Holder

The casing is meant to hold the raspberry pi as well as the battery for the device. The propeller attachment is built-in to the case to make the device more compact and the sides are rounded for comfort for the user. The cut below the propellor is for the wires that power and control the servos. One face of the container is left open for display purposes but, in the actual implementation of the device, would most likely remain concealed.

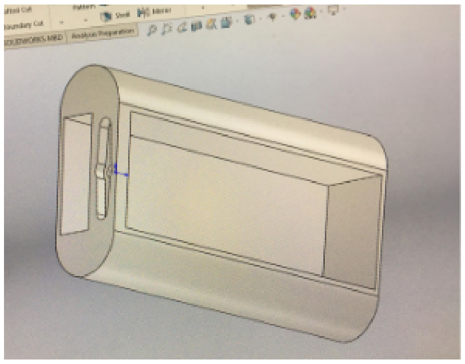


Figure 11: Prototype 2 Pi Casing

After Designing and 3-D printing all the parts, the final step is to hot glue all the components together, or use plastic binder. The key is to make the device look like an efficient, small and compact embedded electronic, so make it neat, and clean. Here is how our assembly turned out:

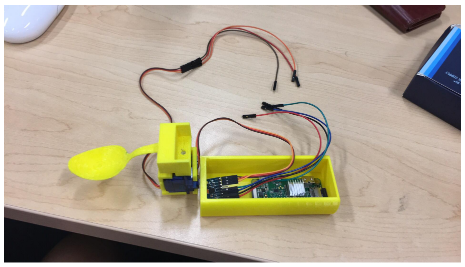


Figure 12: Prototype 2 Assembled

This is optional, but in order for the IMU to be highly stable, you have to glue it onto the device firmly so that it has no change of wobbling. This will not only give you more accurate values, but also make your design look nice as a whole.

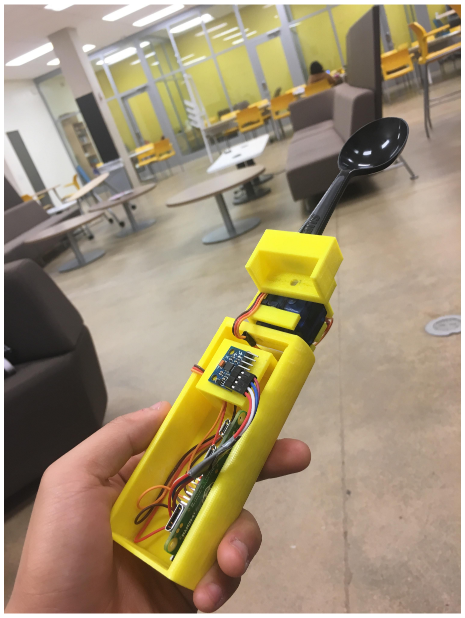


Figure 13: Final Compacted Design

Step 8: Enable SSH on your RPi so the device can be run through USB connection

Great! Once all the designs are completed, the device is almost complete. Just allow the device to be accessed from your personal laptop so you can run the python program from anywhere (headless setup for RPi 0 W). To do this, boot up your RPi like normal, and run:

```
sudo raspi-config
```

Then, in Interfacing Options, enable SSH-ing. Now quit, and attach a microUSB cable to the RPi, and plug the USB end into your laptop. If you have a MAC, like our team did, you can simply open up Terminal Window, and type:

```
ping raspberrypi.local
```

Which will give you the current IP address reading of your RPi (NOTE: this is different every time you switch locations). Using this IP, SSH into your Pi by running:

```
ssh pi@<IP>
```

Now replace IP with the IP address you saw after the Ping session. From there, you should be given access to the RPi's command line, where you will be able to call

```
./control.py
```

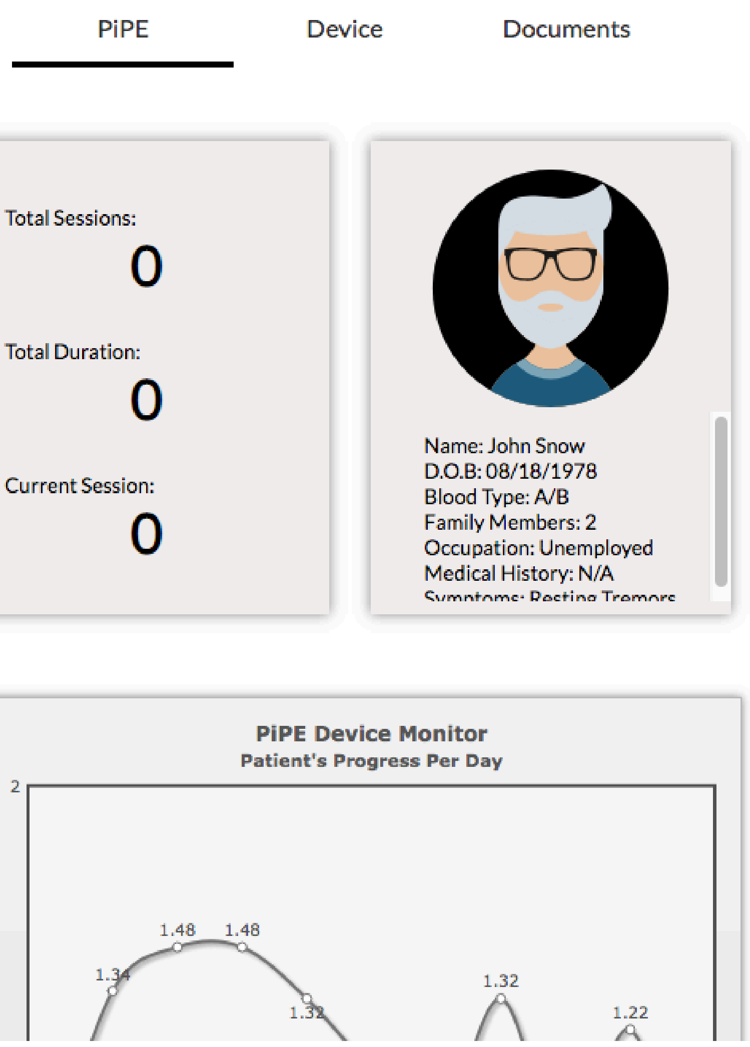
to run the program. Do this, and you can officially test out your full-fledged Parkinson's aide device!

Step 9 (Optional): Live Data Web App

There is one final, and optional step to this project, and that is:

- Connect the RPi to WiFi
- Set up a Firebase account and type your credentials into the control.py file
- Design a website, or web app using simple HTML and JQuery to post Firebase data

In order to accomplish this, you must first develop the front-end GUI for a live data web app. Our version of this was a personal dashboard that tracks the stability, and usage statistics of the PiPE device. For example,



Total Sessions: 0

Total Duration: 0

Current Session: 0

Name: John Snow
D.O.B: 08/18/1978
Blood Type: A/B
Family Members: 2
Occupation: Unemployed
Medical History: N/A
Symptoms: Resting Tremors

Figure 14: Live patient dashboard

Overall, you can create this sort of GUI by planning out relevant data you want to display, and then using basic web development skills, especially front-end Javascript and CSS. As for the live charts, use a framework like Chart.js to render JSON data into a graph for you, all you need to do is format the data to be read in Firebase. These were the software frameworks used in this project:

- Node.js
- HTML/CSS
- JQuery
- Chart.js
- Firebase
- Pyrebase (for Python)

Control.py

After following the guide, you were probably confused about a lot of the code, so in this section we will demystify some of the code.

First, we are defining the python program as a simple executable file using the first line of code. Then, we import all the necessary python modules, and configure Firebase (This is where you replace your Firebase credentials). Note that we define a lot of global variables for the IMU. These are all addresses that the Pi uses to read data from the IMU. Finally, pigpio.pi() is called as an API that has library functions to manipulate and read GPIO data. This is useful for moving and programming the servo motors.

```
#!/usr/bin/env python3

print("Importing modules (please be patient)...")
# import necessary python modules
# import pyrebase

#from firebase import firebase
import requests
import smbus
import time
import math
from time import sleep
import pigpio

# Define firebase variables
config = {
   "apiKey": INSERT YOUR KEY,
   "authDomain": INSERT YOUR DOMAIN,
   "databaseURL": INSERT YOUR DATABASE URL,
   "storageBucket": YOURDATABASE.appspot.com
}
print("Connecting to firebase..")

#firebase = firebase.FirebaseApplication('https://pipe-a7b56.firebaseio.com')
#firebase = pyrebase.initialize_app(config)
#db = firebase.database()

# Define IMU global variables
```

```
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG = 0x1A
GYRO_CONFIG = 0x1B
INT_ENABLE = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H = 0x43
GYRO_YOUT_H = 0x45
GYRO_ZOUT_H = 0x47

# Create pigpio API reference
pi = pigpio.pi()

# Name servos
servo1 = 17
servo2 = 23
```

These next lines of code define functions that we will be calling in a loop. The first is init() which is called once at the start of the program and reads in all IMU values and initializes all IMU global variables. The commented out section is where the session number and duration of the program is being initialized in Firebase. The second method is storeJSON(), which formats the global variables in this program to strings that can be pushed onto Firebase. First it converts each data variable to JSON data, and then pushes the result into Firebase.

```
PROG_TIME = 0
key = 0
value = 0
value2 = 0

# Initialize IMU values and database values
def init():
   print("Initializing program...")
   bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)
   bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)
   bus.write_byte_data(Device_Address, CONFIG, 0)
   bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)
   bus.write_byte_data(Device_Address, INT_ENABLE, 1)

    # Read firebase values and initialize
#  numSessions = db.child("numSessions").get()
#  numSessions = numSessions.val() + 1

   # print current session
#  print("Current Session = " + str(numSessions))

#  jsonNS = {"numSessions":numSessions}
#  db.update(jsonNS)

   print("Set program time to 0")
   jsonCurr = {"current":PROG_TIME}
#  db.update(jsonCurr)

# Store global json data
def storeJSON():

   # Increment program timer
   print(" ")
   print("Program time (s) = " + str(PROG_TIME))
```

```
  jsonCurr2 = {"current":PROG_TIME}
  db.update(jsonCurr2)
  print("Storing data plot: " + str(key))

  # Increment duration by PROGRAM TIME
  duration = db.child("total").get()
  duration = duration.val() + 1
  print("Total Duration of program: " + str(duration))
  jsonTotal = {"total":duration}
  db.update(jsonTotal)
```

The next part of the code calculates the stability Factor of program using readings from the IMU. The algorithm is: find the percentage of the total range of values for each servo, and then add them together to get a stability Factor, which will be high when the device is not moving a all. The storeStat function simply stores these variables back onto Firebase.

```
# Store stability factor
def storeStat(var1, var2, var3):

   # Post stability factor to firebase
  var1 = (1 - ((var1 + 60) / 120))
  var2 = (1 - ((var2 + 60) / 120))

  stabilityF = var1 + var2
  print("Stability Factor: " + str(stabilityF))
  data = str(stabilityF)
  label = str(PROG_TIME)

  jsonPlot = {"label":label, "value":data}

  index = str(var3)

  db.child("data").child(index).set(jsonPlot)
```

This stores the device address from i2cdetect -y 1.

```
# Store device address
bus = smbus.SMBus(1)
Device_Address = 0x68
```

In this main loop, there are multiple events occurring. First, the loop reads the values of the IMU into value and value2. Then, it converts each into an integer, and converts the IMU readings range from [0, 60], [190, 255] to [-60, 60] and finally, [0, 120]. Finally, the values from [0, 120] are converted into servo pulsewidth values ranging from 500 to 2500. This is where pigpio API is helpful, in reading GPIO data securely and with stability. In order to further stability these readings, the program is checking if the current reading differs more than +/- 5 from the previous reading. If not, the programs continues the iteration of the while function, skipping over the rest of the function.

```
# Initialize all global variables
init()
initial = 0
prevVal = 0
prevTemp = 0
```

```
while True:

  key += 1
  key %= 12

  PROG_TIME = PROG_TIME + 1

  #storeJSON()

  # Remove first child of data
  #db.child("data").limit_to_first(1).remove()

  # Read in IMU readings for both servos
  value = bus.read_byte_data(Device_Address, ACCEL_XOUT_H)
  value2 = bus.read_byte_data(Device_Address, ACCEL_YOUT_H)

   # Convert to int for more stable values
  value = math.floor(value)
  value2 = math.floor(value2)

   # Set ranges to [-60, 60]
  if 189 <= value <= 256:
    value = value - 255

  if 189 <= value2 <= 256:
    value2 = value2 - 255

  if prevVal - 2 < value < prevVal + 2:
    value = prevVal

  # Set prevVal
  prevVal = value

   # Round to nearest 5 to increase stability
  value2 = value2 - (value2%5)

   # Convert value to servo pulse width
  if -60 <= value <= 60:
    temp = value + 60
    temp = temp / -120 * 1300 + 2200
    temp = math.floor(temp)
    if prevTemp - 10 < temp < prevTemp + 10:
      temp = prevTemp
    prevTemp = temp
    print("Servo1 Position: " + str(temp))
    pi.set_servo_pulsewidth(servo1, temp)
    \\
    \\
```

This part of the program is to expose a new functionality of the PiPE device: that it can toggle the direction the spoon is facing if the device is tilted up more or down than 50 degrees. To toggle the device direction, tilt the handle up 50 degrees or more and the handle will loop through 10 directions. To stop the loop, simply tilt your device back down.

```
   # Use servo two to calibrate utensil direction
  while -60 <= value2 <= -50:
    value2 = bus.read_byte_data(Device_Address, ACCEL_YOUT_H)
    initial += 100
    initial %= 2500
    if 500<=initial<=2500:
        print("Servo2 Position: " + str(initial))
        pi.set_servo_pulsewidth(servo2, initial)
    sleep(0.1)

  # storeStat(value, value2, key)
   # delay read
  sleep(0.12)
```

Evaluation and Conclusion

Evaluation

As a team we have made this quarter's project experience an indelible one. Each member played to their strengths over the course of the project and we all filled in for each other's weaknesses.

Overall, we are extremely satisfied with the outcome of our project. We are especially pleased with the usefulness of the product that we built. Tremors are a very common symptom that nearly all Parkinson's patients suffer from. We are also proud of our unique, human-centered design that is highly compact and personalized. Finally we are proud that we were able to set up a method for displaying the data we collect on a website in real time. This will give doctors, caregivers, and family members accurate readings of the state of their symptoms.

While we have accomplished a lot in these past ten weeks, there were some areas where our final product did not quite meet our expectations. For instance, we were very displeased to find out that the IMU we are using did not support a third axis. The IMU used gravity to measure the pitch and roll, but we would need some sort of compass to measure the yaw. Another miscalculation was the sensitivity of the IMU. Even the slightest movement would set off the servos, which is not very ideal for our purposes of keeping the utensil steady. The project is still far from professional industry grade, but we definitely stuck to our initial motivations, that we would create something affordable yet highly functional and unique. Overall, we wish to make our project applicable to real life, and be used as a legitimate way to alleviate Parkinson's tremors.

If we had more time and resources to complete this project, there would be more modifications we would make, including:

- Adding yaw correction functionality (a third axis)
- Further humanizing and customizing our nuanced design so that it looks even more professional and industry-standard
- Improving our software, which means not only making the device smoother and less jittery overall, but also making our web app more sophisticated, and add more user-oriented features
- Test our product with actual patients to determine if they actually need this kind of device, and seek to modify our design based on their feedback

Conclusion

In evaluation, all three of us have learned a lot about topics that we would never have been exposed to without this experience. Computer-assisted design, micro controllers, IMUs, software development, and complex API programming. We have also learned a lot about the engineering process in general and the need to test elements before we expand on them. We are confident that our project will potentially have a positive impact on the lives of Parkinson's patients and serve as an example for future engineers to replicate. It was a lot of hard work, but we all enjoyed these past ten weeks and we hope to engage in more projects such as these in the future!

References

Citations

[1] https://medlineplus.gov/parkinsonsdisease.html

[2] https://www.bluebirdcare.co.uk/greenwich/useful-information/how-parkinson-s-disease-effects-every-day-life

[3] http://www.parkinsonsvoice.org

[4] https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/diagnosis-treatment/drc-20376062

[5] https://www.forbes.com/sites/johnnosta/2018/07/27/high-tech-parkinsons-therapy-with-a-powerful-consumer-touch/3d28d0a74fdc

[6] https://www.microsoft.com/en-us/research/project/project-emma

[7] https://www.davisphinneyfoundation.org/blog/treatments-tools-technology-to-help-you-live-well-with-parkinsons

[8] https://www.nuance.com/dragon.html

[9] https://www.walkwithpath.com

[10] https://www.theverge.com/2013/9/23/4761182/liftware-spoon-helps-parkinsons-sufferers-control-their-tremor

[11] https://thepi.io/how-to-install-raspbian-on-the-raspberry-pi

[12] http://ozzmaker.com/berryimu