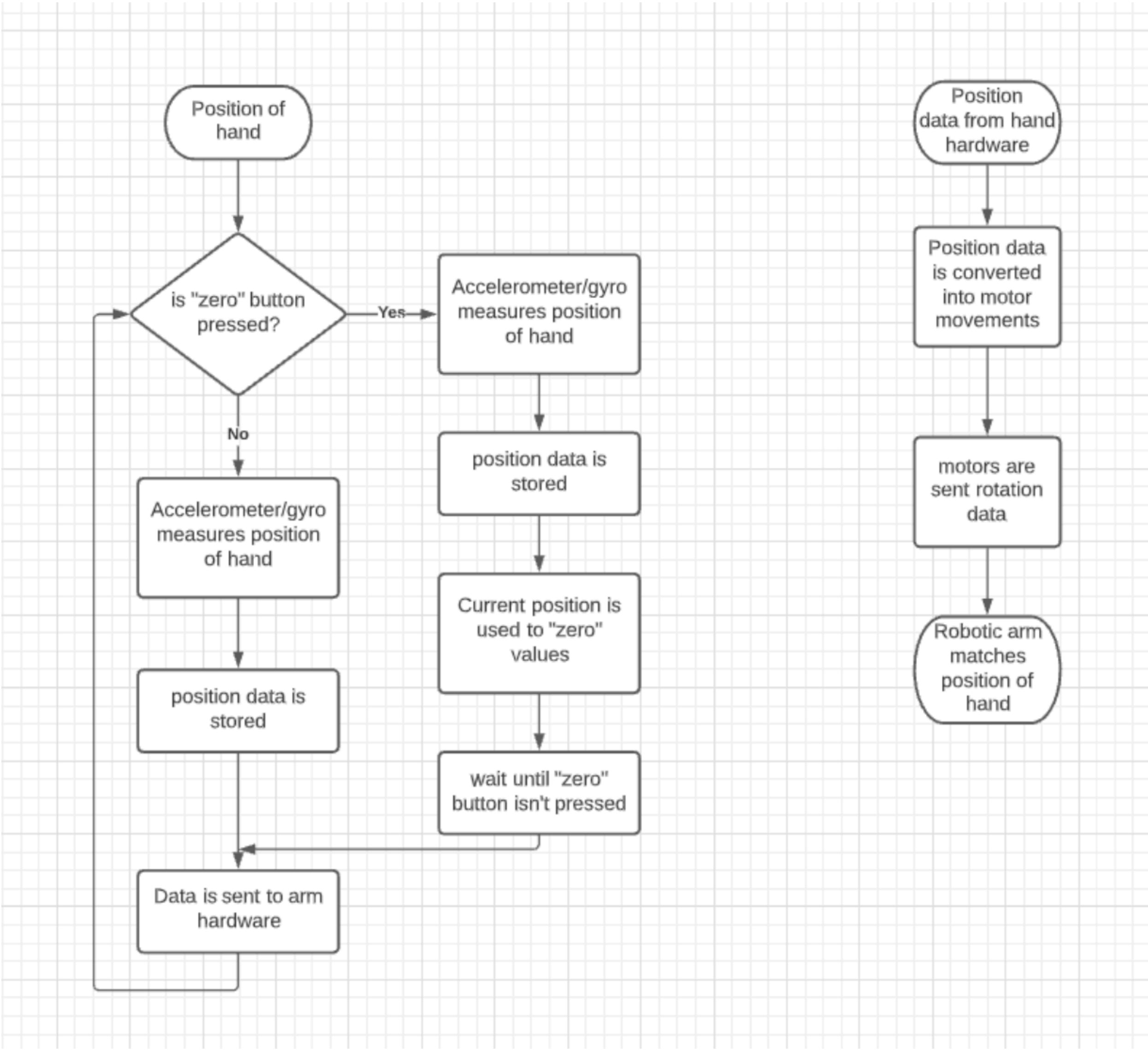
ECE 110/120 Honors Final Report

1. Introduction
    a. Statement of Purpose:

    This project intends to create a robotic arm, similar in size and scale to a real human arm. The primary goal is to develop a device which is able to minimize human injury by using a device that is both as accurate and capable as the human hand and arm. That is still the intended purpose, and the problem it is solving is to make activities that need to be done by hand precisely more accessible and possible to do with more power and strength than previously possible. It is unique as it hopefully allows intuitive control over this powerful machinery.

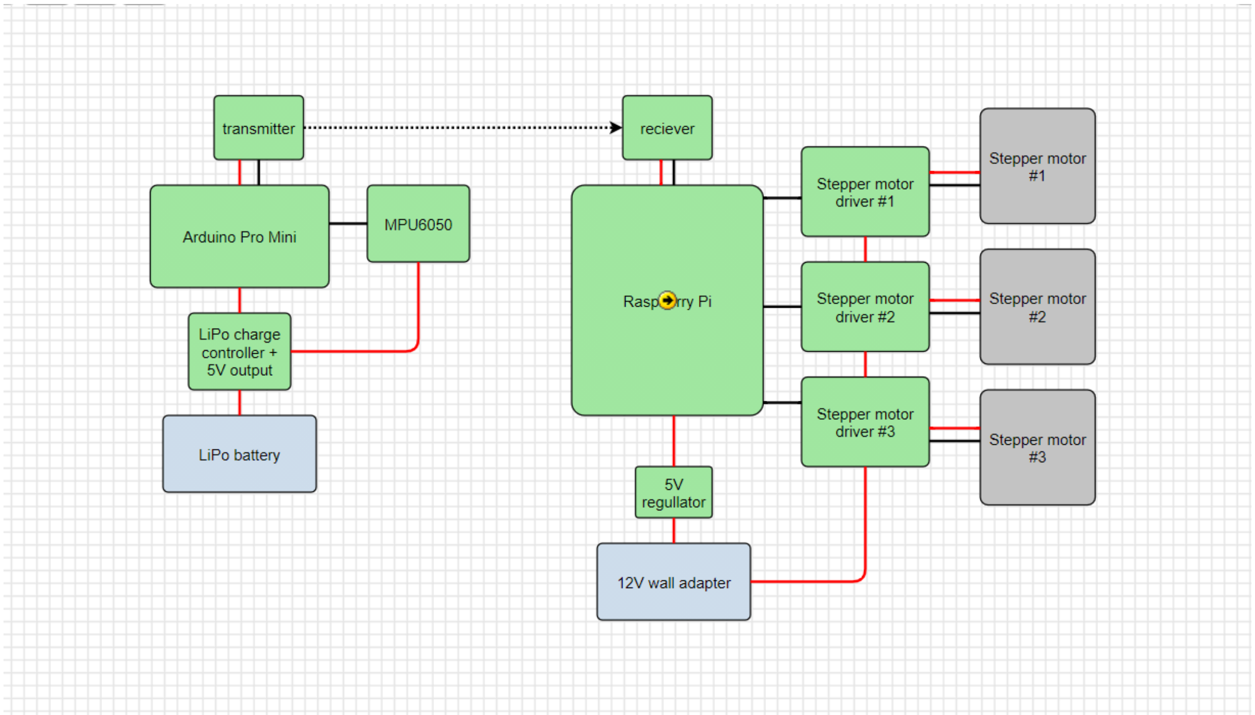
    b. Features and Benefits
        i. Precise and intuitive control from hand movement
        ii. More mechanical strength than human arm
        iii. Uses gyroscope for rotation
        iv. Increase production capacity at a factory
2. Design:
    a. System Overview:

    The robotic arm is designed to match the movements of the operator's arm and hand. This is measured using a 3 axis accelerometer/gyroscope, which communicates the data through an arduino and over radio to a Raspberry Pi. This is an open-loop system, so challenges can arise in the long-term accuracy of the arm, due to imperfections in measurement.


Position of hand
is "zero" button pressed?
Yes
No
Accelerometer/gyro measures position of hand
position data is stored
Current position is used to "zero" values
wait until "zero" button isn't pressed
Accelerometer/gyro measures position of hand
position data is stored
Data is sent to arm hardware
Position data from hand hardware
Position data is converted into motor movements
motors are sent rotation data
Robotic arm matches position of hand

b. Design Details:

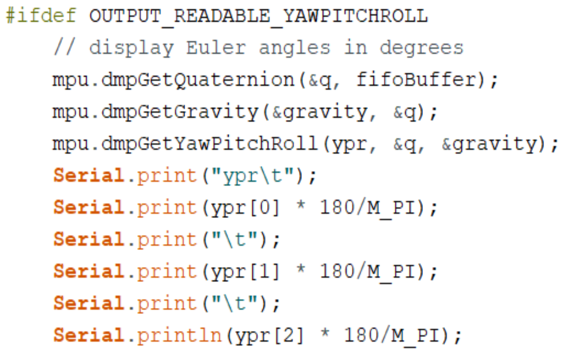
Lithium Polymer battery is what provides power to the transmitting portion of circuit, which includes handheld section that gets data for hand position. Power is sent from battery to power controller, which distributes power to Arduino Pro Mini, the accelerometer, and the transmitter through the Arduino Pro Mini. Accelerometer data is constantly sent to the arduino to consistently get the position of the hand, and the Arduino sends it to the transmitter where the data is received by the receiver and then sent in to the Raspberry Pi, where it is parsed and used. The entire receiver circuit is powered from a 12 Volt wall adapter, as the arm is stationary and does not need to be mobile. That power is passed through a 5 Volt regulator to power the Raspberry Pi, which also powers the receiver. The 12 Volts are also passed directly into the 3 stepper motor drivers to power the 3 stepper motors, one for each axis to give the arm full 3 dimensional movement. Data from the Raspberry Pi is taken from the receiver and parsed, and then sends the correct movement needed for each of the three motors to the motor driver which then sends the correct amount of power to the motor.



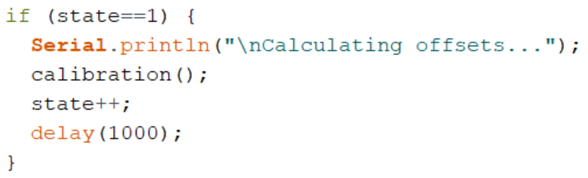
Red is Power   Black is Data   Dotted Line is wireless transmission over radio

3. Results:

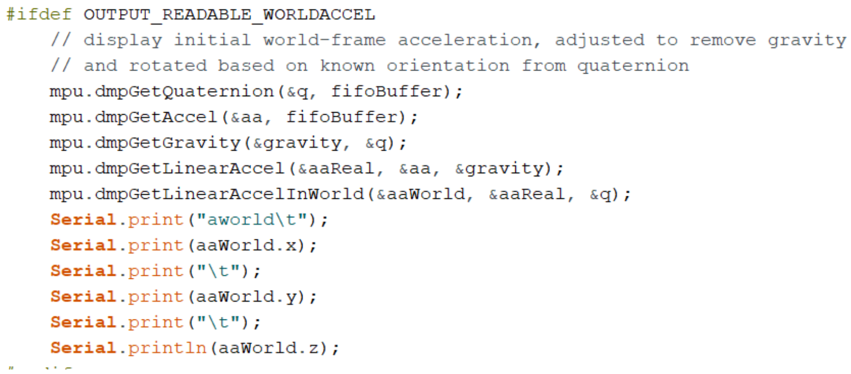
Our main input sensor was our MPU6050 gyrometer/accelerometer to get the position of our hand, it was fairly accurate but we were very aware that over time, it would start drifting, and the hand position would become out of sync with the position of the arm. Our only actuator aside from on and off buttons would be the button to zero the hand gyro so we can set a baseline and get accurate readings, as well as correct if it drifts too much. We used an online library to get gyrometer displacement and linear acceleration. We converted the angle displacement to degrees (top-left picture) and printed it then we used another function in the library to get linear acceleration (bottom picture) and the units already came in meters per second. To combat the drift, we used a separate online library to correctly calibrate the MPU6050 (top-right). After doing this, the drift was significantly less and well within our margin of error.



```
#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    Serial.print("ypr\t");
    Serial.print(ypr[0] * 180/M_PI);
    Serial.print("\t");
    Serial.print(ypr[1] * 180/M_PI);
    Serial.print("\t");
    Serial.println(ypr[2] * 180/M_PI);
```



```
if (state==1) {
  Serial.println("\nCalculating offsets...");
  calibration();
  state++;
  delay(1000);
}
```



```
#ifdef OUTPUT_READABLE_WORLDACCEL
    // display initial world-frame acceleration, adjusted to remove gravity
    // and rotated based on known orientation from quaternion
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
    Serial.print("aworld\t");
    Serial.print(aaWorld.x);
    Serial.print("\t");
    Serial.print(aaWorld.y);
    Serial.print("\t");
    Serial.println(aaWorld.z);
```

4. Problems and Challenges:

One problem we faced was being able to effectively use our time, especially with limited resources and with one member being remote. Covid as a while made it difficult to work together as much and as effectively as we would have wanted. We also had trouble acquiring the necessary parts for the project. One technical challenge we faced was having to deal with drift, in our accelerometer slowly getting out of sync with the arm, but we were able to provide workarounds for that and our end code worked pretty well. We were aiming for precision and accuracy, and we did the best with our situation and the tools we were given.

One challenge that we faced in designing the system was that of minimizing position drift over time. Due to the nature of calculating position from an acceleration value, a small error in the acceleration will cause an error in position that increases with time. Another challenge we faced was in the mechanical design of the arm. Being electrical and computer engineers, we were less savvy with mechanical design and manufacturing, which made it difficult for us to know how to best manufacture a powered robotic arm. However, we gave it our best effort and made a 3D model of what we had in mind for the arm.

5. Future Plans:

We may decide to continue to work on this project over this coming summer to make sure all of the separate elements that we have created effectively combine well together, as well as try to test it much more thoroughly on both the software and hardware sides. We hope to make it more precise and work smoother and more intuitively with the input from the hand, as well as hope to minimize the delay and error.

Final Video Link:

https://drive.google.com/file/d/1qzlyhqOrWyFQGqHH1lBHaX9jWXOjQfD8/view

6. References:

1. R. Satheeshkumar, "IEEE Transactions on Computational Social Systems society information", IEEE Transactions on Computational Social Systems, vol. 2, no. 4, pp. C3-C3, 2015. Available: 10.1109/tcss.2016.2527186 [Accessed 16 February 2021].
2. V. Patidar and R. Tiwari, "Survey of robotic arm and parameters," 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2016, pp. 1-6, doi: 10.1109/ICCCI.2016.7479938.
3. K. Kruthika, B. M. Kiran Kumar and S. Lakshminarayanan, "Design and development of a robotic arm," 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), Bangalore, 2016, pp. 1-4, doi: 10.1109/CIMCA.2016.8053274.
4. Anisur Rahman, Md & Khan, Alimul & Ahmed, Tofayel & Sajjad, Md. (2013). Design, Analysis and Implementation of a Robotic Arm-The Animator. International Journal of Engineering Research. 02. 298.
5. "Trainable Robotic Arm", Adafruit Learning System, 2021. [Online]. Available: https://learn.adafruit.com/trainable-robotic-arm/required-parts. [Accessed: 18- Feb-2021].

Arduino Libraries:

6. Arduino. (2020). *Serial Peripheral Interface Library.* v2.7.4. [Library]. https://github.com/esp8266/Arduino/blob/master/libraries/SPI/SPI.h

7. Arduino. (2020). *Radio Driver Library.* v1.4.0. [Library].

   https://github.com/nRF24/RF24

8. Arduino. (2020). *Servo Library.* v1.1.7. [Library].

   https://github.com/arduino-libraries/Servo

Appendix

Arduino Code:

https://github.com/ronitk2/Honors-Lab-Spring-2021.git