Title

Prosthetic Hand Computer Vision Fusion

Period of Performance: January 30, 2023 – December 8, 2023

Technical Point of Contact: Jin Chul, Rhim
Computer Engineering
School of Engineering
San Francisco State University
1600 Holloway Ave. San Francisco, CA 94132
Email: jinchul.edu@sfsu.edu

Advisor: Zhuwei Qin, Professor
School of Engineering
San Francisco State University
1600 Holloway Ave. San Francisco, CA 94132
Phone:
Email: zwqin@sfsu.edu

Table of Contents

Abstract

Prosthetic limbs have improved, but integrating with a user's nervous system remains a challenge. Our project combines EMG sensors, computer vision cameras, and a prosthetic arm with machine learning to create a more responsive and intuitive prosthetic system. Our approach is unique in integrating these technologies into a single, seamless system, using machine learning to analyze EMG signals and computer vision data for more accurate control. We leverage the SSDMobileNetV2 model to estimate the distance between grabbable objects and the prosthetic arm, improving precision and reducing accidental movements. We explore using deep learning algorithms to classify EMG signals for more natural control. Our project proposes a novel approach to integrating EMG sensors, computer vision, and prosthetics using machine learning, with potential to improve the quality of life for people with limb loss.

Section 1. Introduction

Section 1.1 Project Goal

The goal is this project is to develop a cheap prosthetic hand with multiple sensors – Electromyograph sensors, Camera, and Microphones with on-device machine learning features. However, the main focus for this class will be the computer vision processing.

Section 1.2 Significance

Implement on-device machine learning model that could do backpropagation on-device to increase the accuracy of both computer vision and EMG sensor classification.

Section 2. Background

Section 2.1 Prior Arts in Research/Commercial Development

Traditionally, research on prosthetic design utilizing EMG signal analysis has primarily relied on classification algorithms. However, with the exponential growth of microcontroller power and the advent of micro-machine learning technologies such as TensorFlow Lite, researchers are increasingly turning to machine learning models that can be implemented on microcontrollers.

Section 2.2 Team Member’s Prior Experience

I transferred to SFSU at 2022 and changed major to Computer Engineering. It is my first time to learn Embedded Engineering and Machine Learning.

Section 2.3 Budget

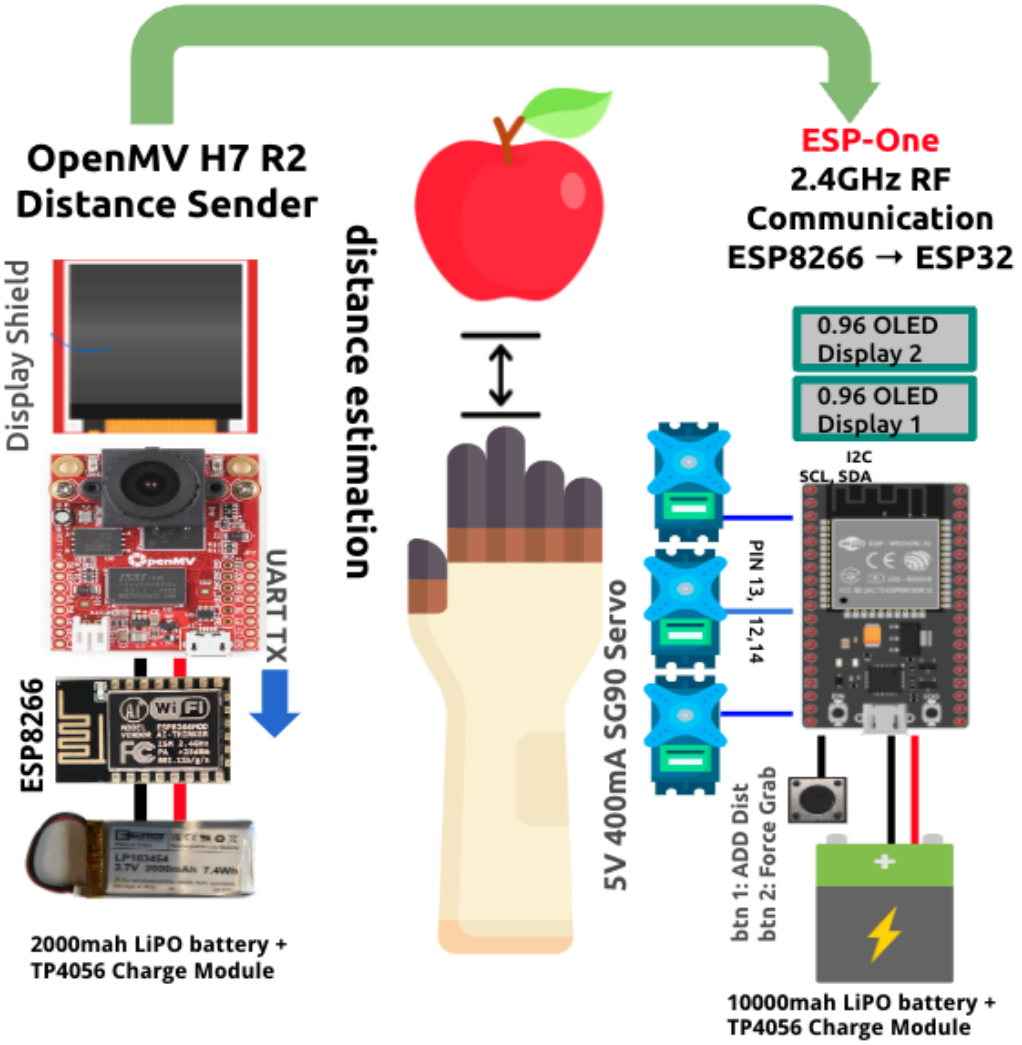
Item	Quantity	Unit Price (USD)	Total Price (USD)
Exoglove	1	400	400
OYMotion EMG Sensor	1	50	50
HACKBerry Prosthetic hand	1	70	70
ESP32	2	7	14
ESP8266	2	4	8
OpenMV H7	1	85	85
PCB Board	3	35	105
Miscellaneous (Wires, Connectors, etc)	-	50	50
Total Cost	-	-	565

Most of the components already exist in ICE Lab or MIC Lab from previous students.

Section 3. Approach/Design

Section 3.1 Design Objective/Specification 1

Section 3.1.1 Introduction



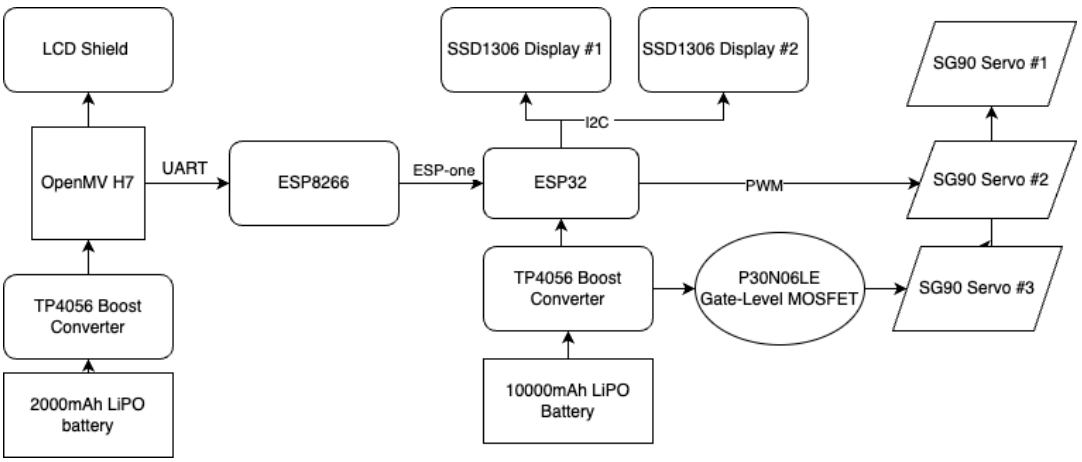


Figure 1. Design #1 Overview

Section 3.1.2 Approach

The design will process the inputs to output a binary output: either a grab motion or the release motion of the HACKABERRY Prosthethic hand model.

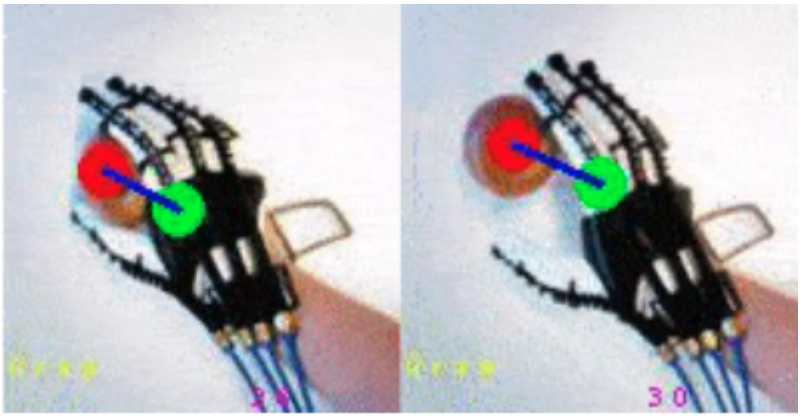


Figure 2: Design #1 openMV H7 Camera Input: 128X128 RGB Input

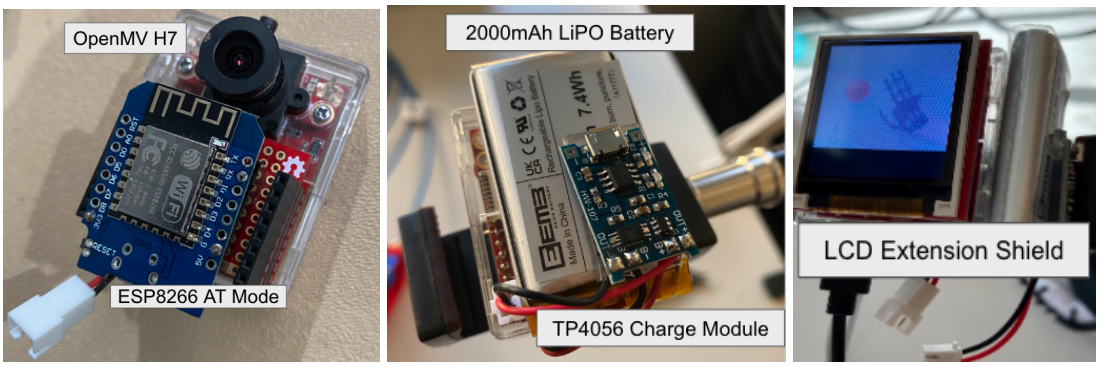


Figure 3: OpenMV H7 System Configuration

OpenMV H7 R2 will process the RGB images, convert them to array to be fed into pretrained tensorflow lite model. Tensorflow lite model will output the model index. If both the hand and apple's threshold exceeds 0.4, the OpenMV will calculate the distance between those two objects and store them into the array. In order to remove outliers and minimize latency caused by UART communication, the OpenMV will store 5 distance values, remove outliers, and then send the average of those 5 values to the UART communication. Once the UART distance is sent to the ESP8266 which is connected throught the external dongle, ESP8266 will act as a beacon with AT-command (Every uart information fed inside will be sent to ESP32 with ESP-one). If ESP8266 successfully finds the ESP32 with correct MAC address, it will stream the data.

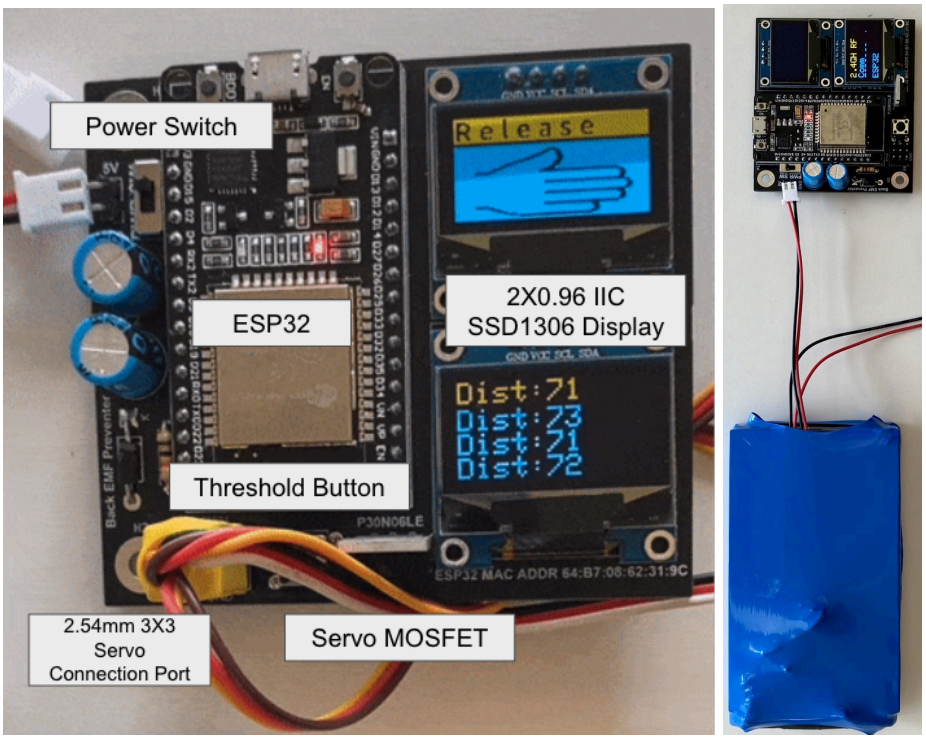


Figure 4: Design #1 ESP32 receiver board configuration with 10000mAh Battery Pack

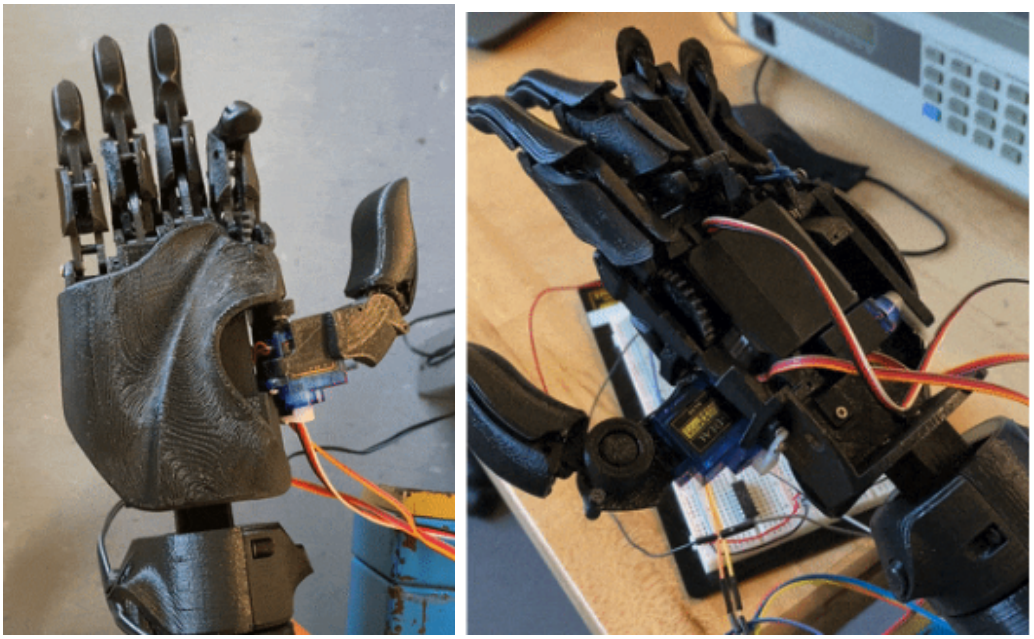


Figure 5. HACKABERRY Prosthetic hand model used to indicate grasping/releasing motion

ESP32 will autostart with esp-receiver mode. It has one button which user can adjust the distance threshold from 70 pixel to 250 pixel. Pressing button will increment threshold by 10 and loop back to 70 pixel if value is bigger than 250. In order to prevent SG90 servo from burning, the logic gate level mosfet was used to shut down any current going to the exoglove, as the servo commonly burns if it is stuck at certain angle or if internal potentiometer reads faulty vale. This approach was possible because there is no torque being applied to servo when grabbing or releasing.

If the distance value being sent is less than or above threshold, ESP32 will turn on the MOSFET, change PWM value, and turn off the MOSFET. However, in order to turn on or

off MOSFET, 10 microseconds of time is needed to fully turn on or off. However, this does not significantly affect the purpose of demoing the basic concepts of machine learning and wireless communication.

Section 3.1.3 Pitfalls

One of the biggest issue of the design is the miscalculation of the correct distance between exoglove and grabbable object. Despite the low threshold for detection and the static white background, sometimes the model does not detect the exoglove or the apple, outputting zero distance. In order to prevent this, the algorithm to remember the last 5 frames of coordinates of apple and exoglove was developed, but sometimes those coordinates causes ghost hand/apple problem where the hand had already moved but the coordinates stay the same, thus forcing the faulty grabbing/releasing motion.

This problem can be fixed by implementing more powerful and faster microcontroller to calculate higher fps. The current model is only optimized for static background and minimum amount of convolution, so the faulty inference cannot be avoided. Logics can be implemented to fix this issue, but it could cause other minor problems.

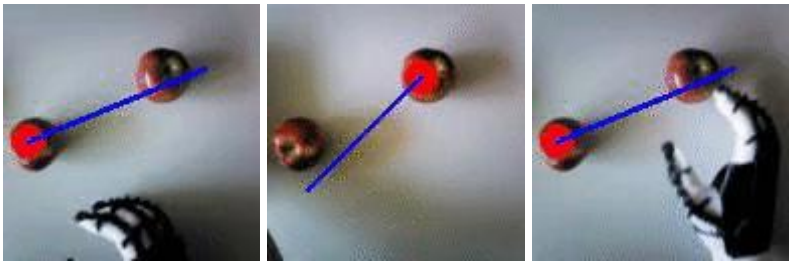


Figure 5. openMV faulty inference cases which failed to detect objects

Another issue of the design is the incorrect x,y, coordinate of hand model. Due to the exoglove dataset that covers wide area of exoglove, the exoglove detection inferece coordinates can vary a lot based on each frames. Due to this, the distance between hand and apple can vary a lot, and sometimes there can be multiple center coordinates for hand model. In order to fix this problem, the average pooling method was used to remove outlying data and to get the stable coordinates of both exoglove and hand.

However, this method significantly delays the system as the OpenMV does not have significantly high fps and those average pooling calculations are not fast in the python codes. In ~5fps environment, average pooling of 10 frames will causes system to output 1 distance out of every 2000ms. This causes significant delay in whole system.

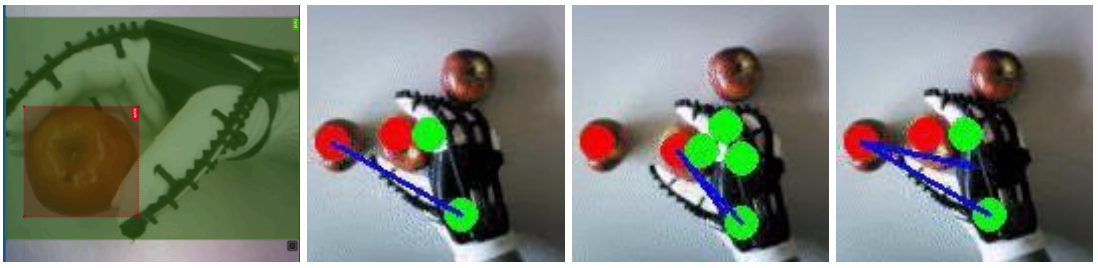


Figure 6. samplexoglove datasets (left). Cases of multiple exoglove coordinate calculation

The last issue is that the MOSFET takes too long time to turn on. Despite the theoretical turn on time which should take less than microseconds, the actual implementation on the circuit caused the significant latency.

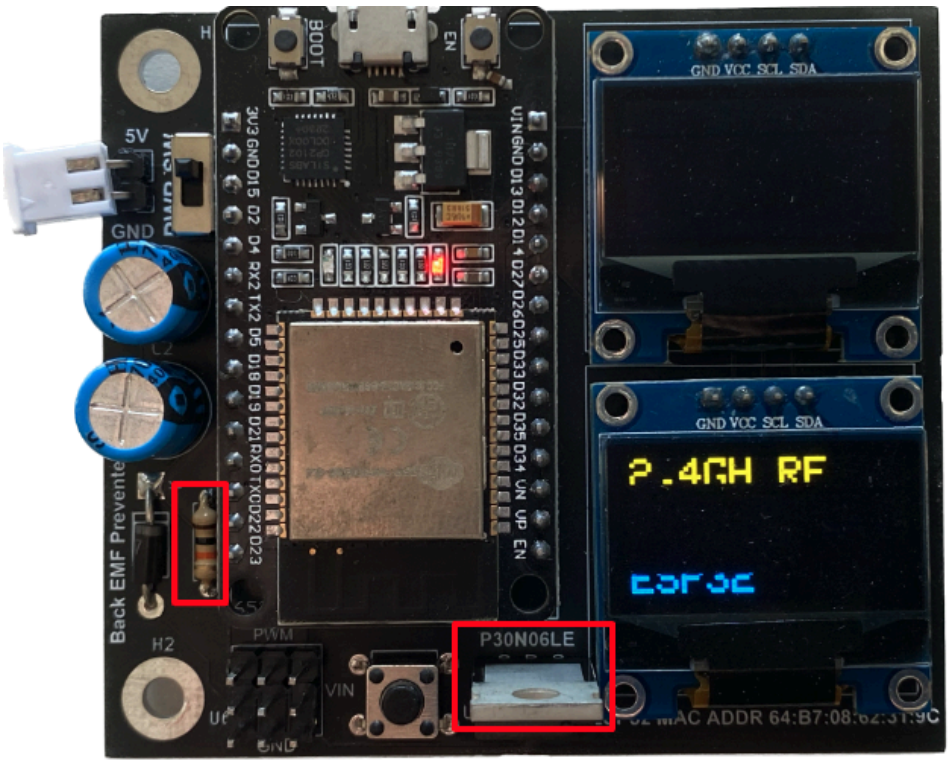


Figure 7. P30N06LE Gate-level MOSFET with 10Kohm pull-down resistor

Despite the expectation, without the delay the Servos did not properly turned on. Whenever the grabbing or releasing was happening, the mosfet is turned on, the servo changes angle based on PWM value, and then mosfet turns off. However, if there was no delay function, the servo did not properly moved. Because this design was for simple demo purposes, the problem was temporarily fixed by eusuring significant delay in the function. However, this problem should be properly debugged through looking at oscilloscope values.

```
    digitalWrite(25, HIGH); //turn on MOSFET
    delay(100); // 0.1 seconds delay
    servo.grab();
    delay(100);
    digitalWrite(25, LOW);` // turn off MOSFET
```

Figure 8. code snippet of delay logic in Servo function

Section 3.1 Design Objective/Specification 2

Section 3.2.1 Introduction

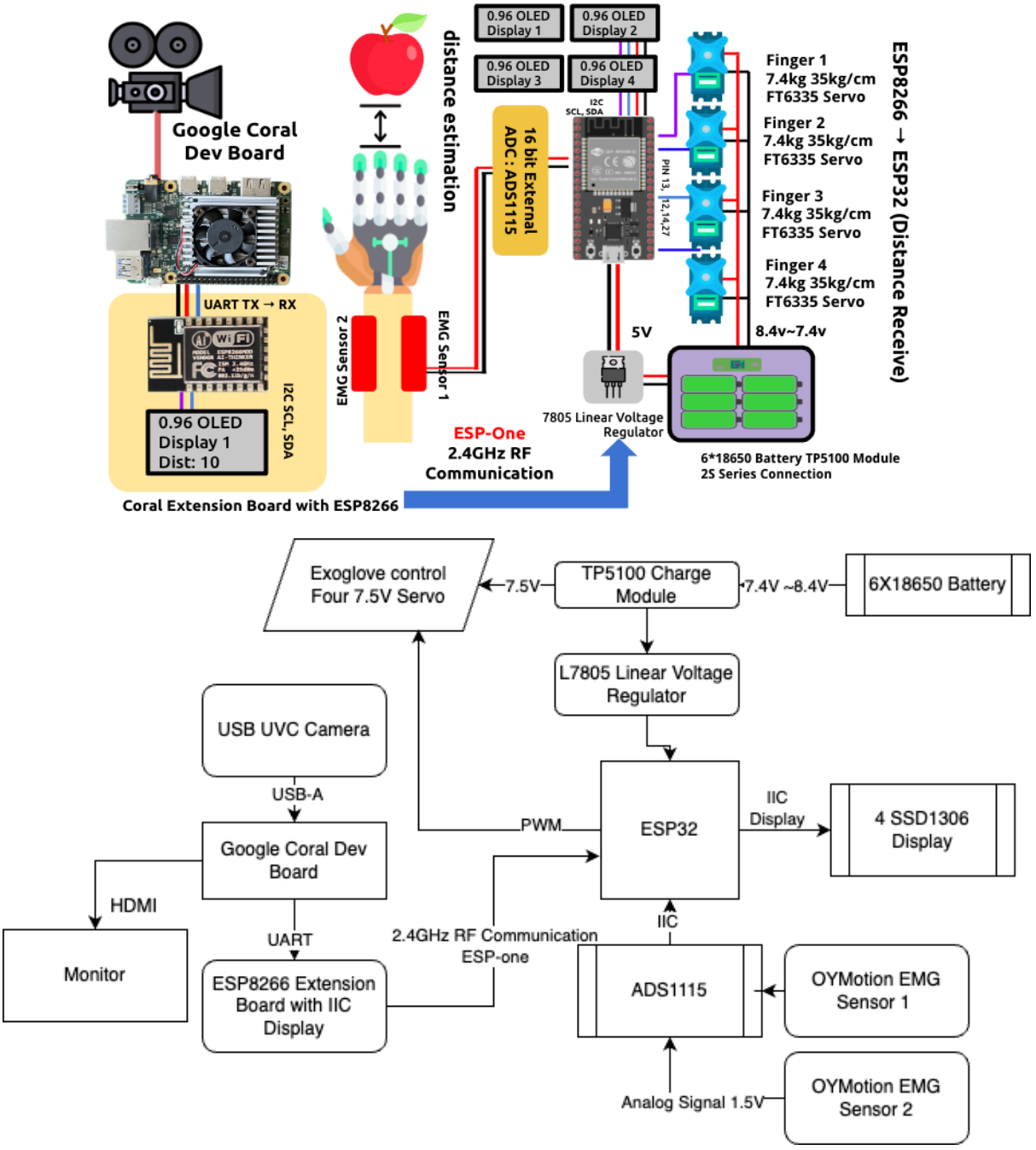


figure 9. Design #2 with Google Coral Dev Board

Section 3.2.2 Approach: Distance Estimation

The USB-A UVC camera inputs the 640X480 frames to the Google Coral Dev board. The dev board will run EfficientDetV2 in TPU chip, runs inference and stores the distance value between exoglove and apple. Once the distance value is saved, the coral dev board will sort and remove outlying values and outputs the average distance values of last 10 frames. The Google Coral Dev board was selected due to its ability to accelerate the tflite operands through tensor TPU with stable framerate.

The EfficientDet model was trained in local computer with input size of 224X224 with float16 quantization. The tensorflow model was quantized through EdgeTPU compiler in order to optimze the models with 94% testing dataset accuracy.

figure 10. Trained Efficientdet edgetpu models

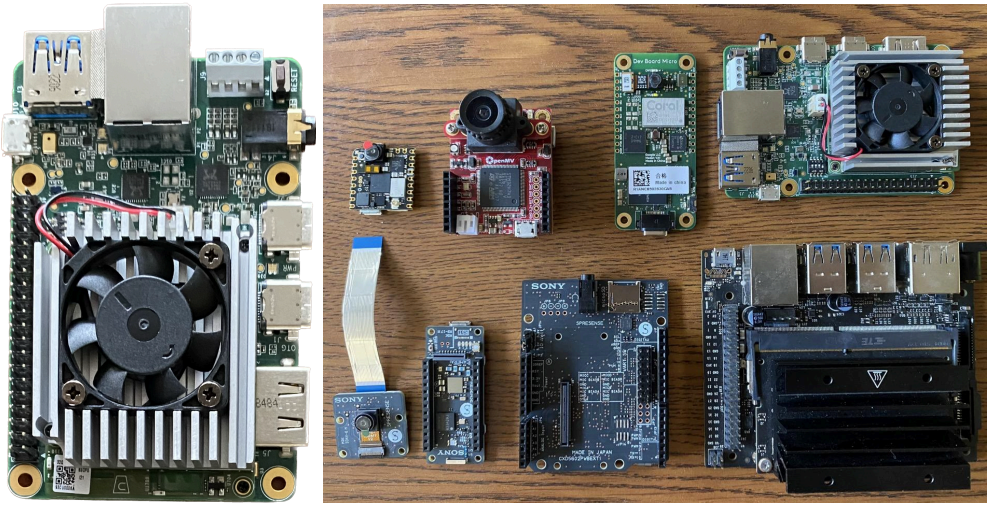


Figure 10. Google Coral Dev Board (left). Top row rightmost microcontroller is google coral dev board.

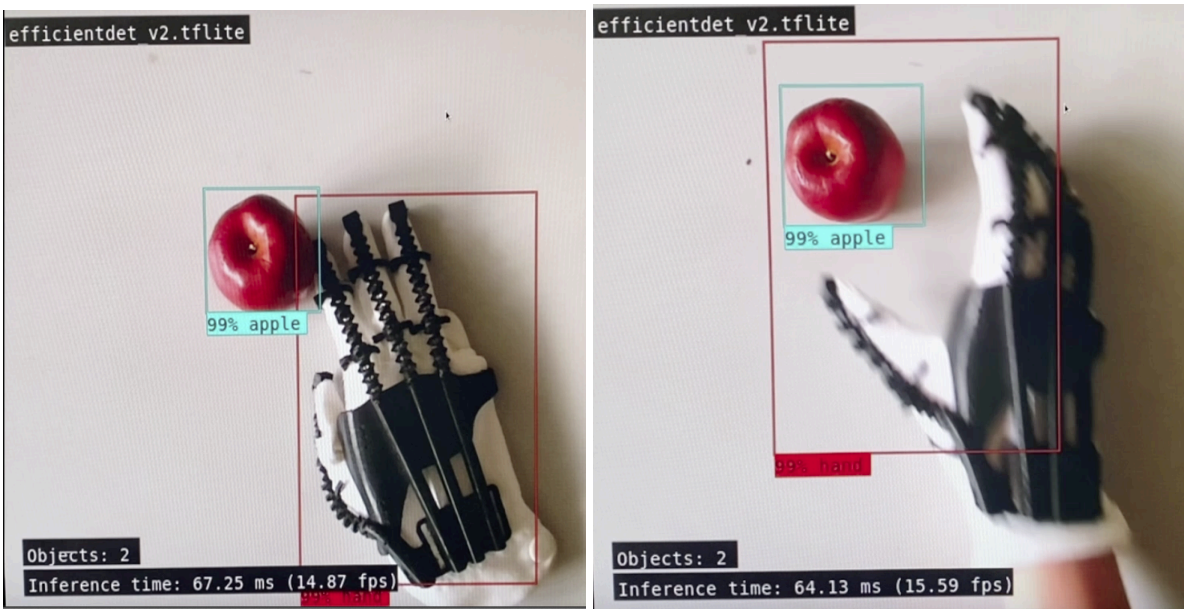


Figure 11. Google Coral Dev Board Inference with input of 224X224 pixel EfficientDet

The distance was output through dev/ttymxc2 UART port of Google Coral Dev board. ESP8266 Board receives the information through the UART port and sends the distance information to the ESP32 through ESP-one.

Figure 12. ESP8266 receiving distance information through UART communication

Section 3.2.3 Approach: Electromyography Signal Analysis

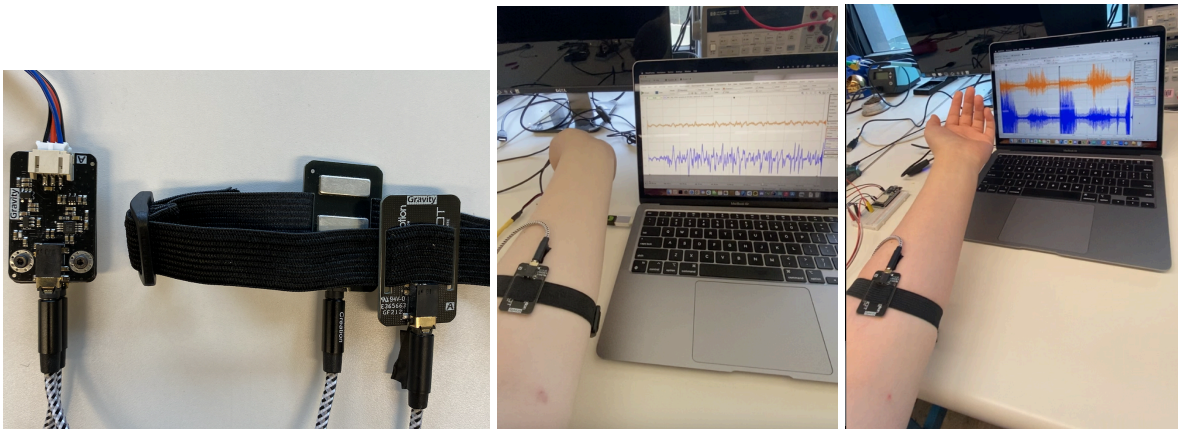


Figure 13. Double-sided OYMotion EMG Sensor.

To measure the EMG signal, the OYMotion EMG sensor were merged into one and tested. EMG sensor have operational amplifier inside that amplifies the impedance of skin and outputs the analog voltage. If the muscle signal is active, bot amplitude and frequency of EMG signal increases. Due to the inaccuracies of the ESP32 adc with 9-bit accuracies and nonlinear characteristics even with stable vref pins, the external 16-bit ADS1115 was used to

measure the analog signal of EMG sensor. ADS1115 is most commonly used external adc for microcontrollers due to its widespread software support.

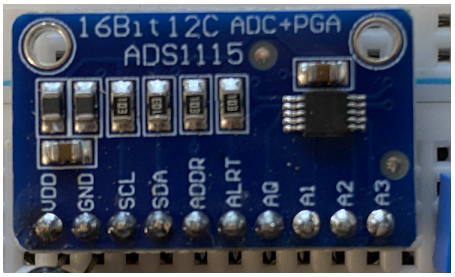


Figure 14. external 16-bit adc

The EMG signal from OYMotion was fed to port A0 and A1 of ADS1115 with frequency of 460Hz. The 16-bit values were sent to ESP32 through I2C communication. ESP32 band-pass those values for 20~1000 Hz and then drops the leftmost 4 bits for ease of calculation and saving spaces. Then those values were zero-centered and then mean square values were calculated. The goal was to rectify the signal so that it is easy to make a binary (grab, release) linear classification system.

Once the signal is rectified, the signal is fed to pretrained logistic regression model. The model was trained on previous datasets that was collected while wearing exoglove. The dataset was collected based on three cases: clenching, releasing, and resting.

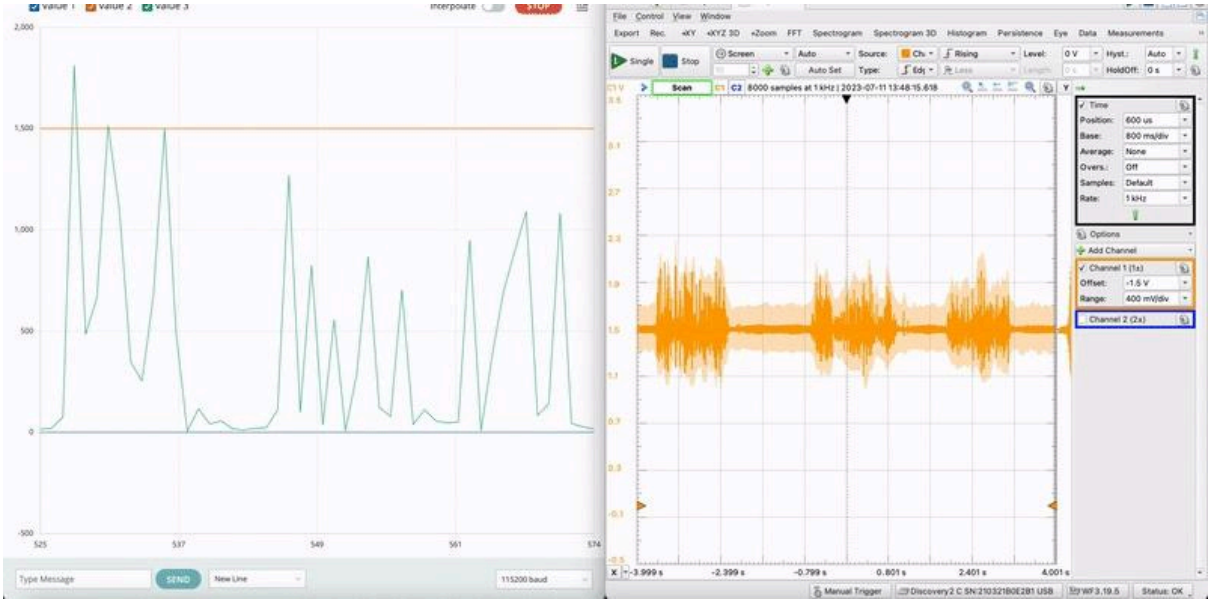


Figure 15. Filtered and rectified signal on ESP32 (left). Raw EMG value read by Analog Discovery 2 on right.

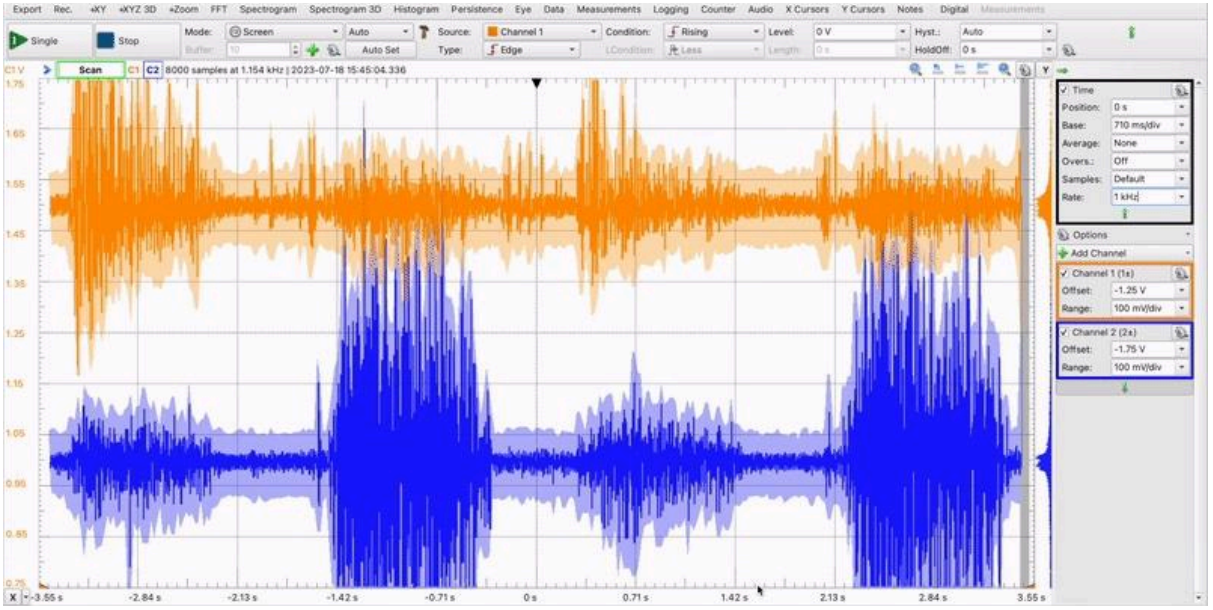


Figure 16. Typical clenching and releasing EMG signal captured by Analog Discovery 2. Higher amplitude indicates higher muscle activity.

Accuracy: 1.0
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	73
1	1.00	1.00	1.00	64
2	1.00	1.00	1.00	63
accuracy			1.00	200
macro avg	1.00	1.00	1.00	200
weighted avg	1.00	1.00	1.00	200

Table 1. Pretrained logistic regression model based on rectified EMG data

Based on extracted float16 weights and bias, the logistic regression classifier library was made so that zero-centered value of 16 emg values is fed and the output of classifier is either 0,1, or 2.

```
// Prediction function implementation
int LogisticRegressionClassifier::predict(int x1, int x2) {
 // Use the provided weights and bias for prediction
 // Replace weights and bias with actual values from your model
 double weights[3][2] = {{-0.03380946, 0.03126931},
                         {-1.13246563, 1.130974},
                         {1.16627509, -1.16224331}};
 double bias[3] = {77.84127544, -33.53217802, -44.30909742};
...
...
//main function
supine_emg_val = EMG.getVal(A0);
prone_emg_val = EMG.getVal(A1);
current_emg_position = static_cast<EMG_Position>(classifier.predict(supine_emg_val,
prone_emg_val));
```

Section 3.2.3 Approach: Whole Board Logic

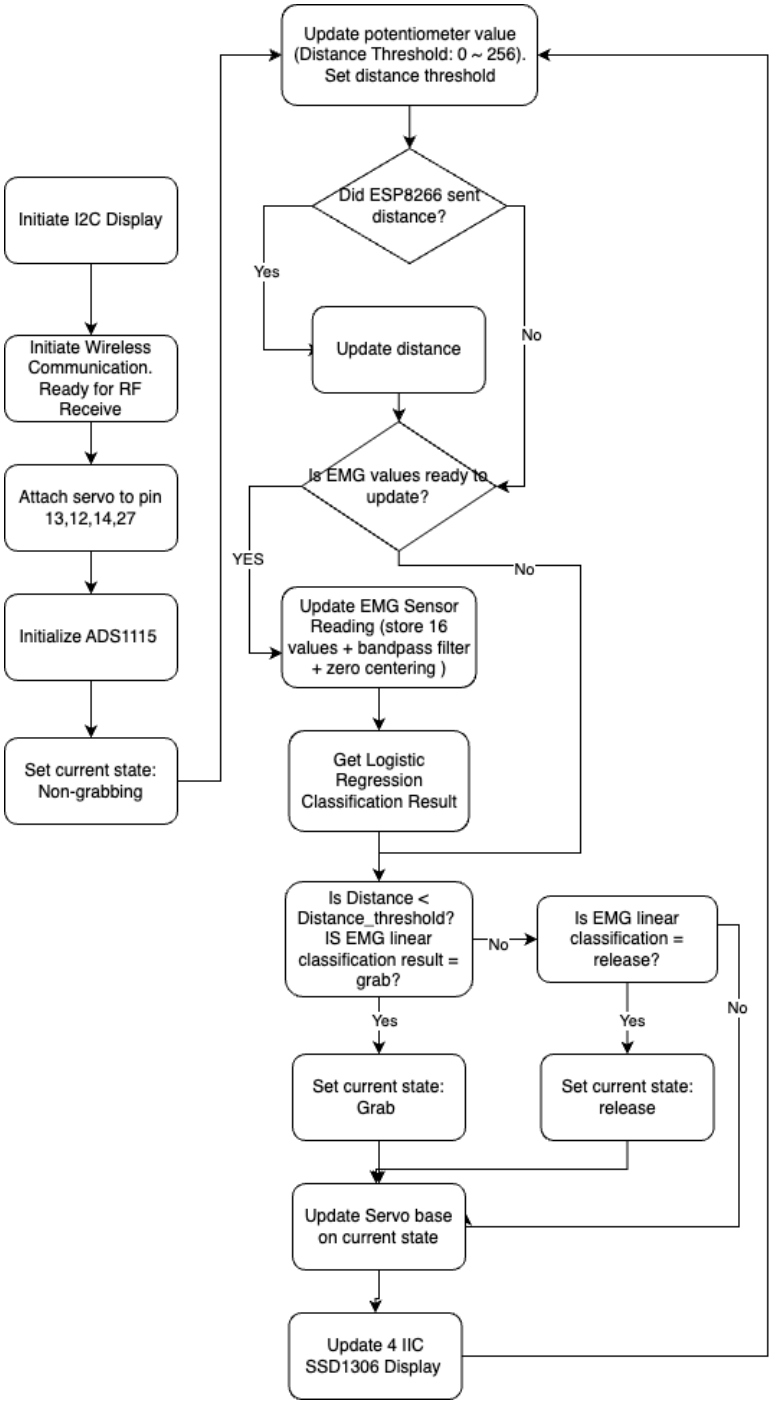
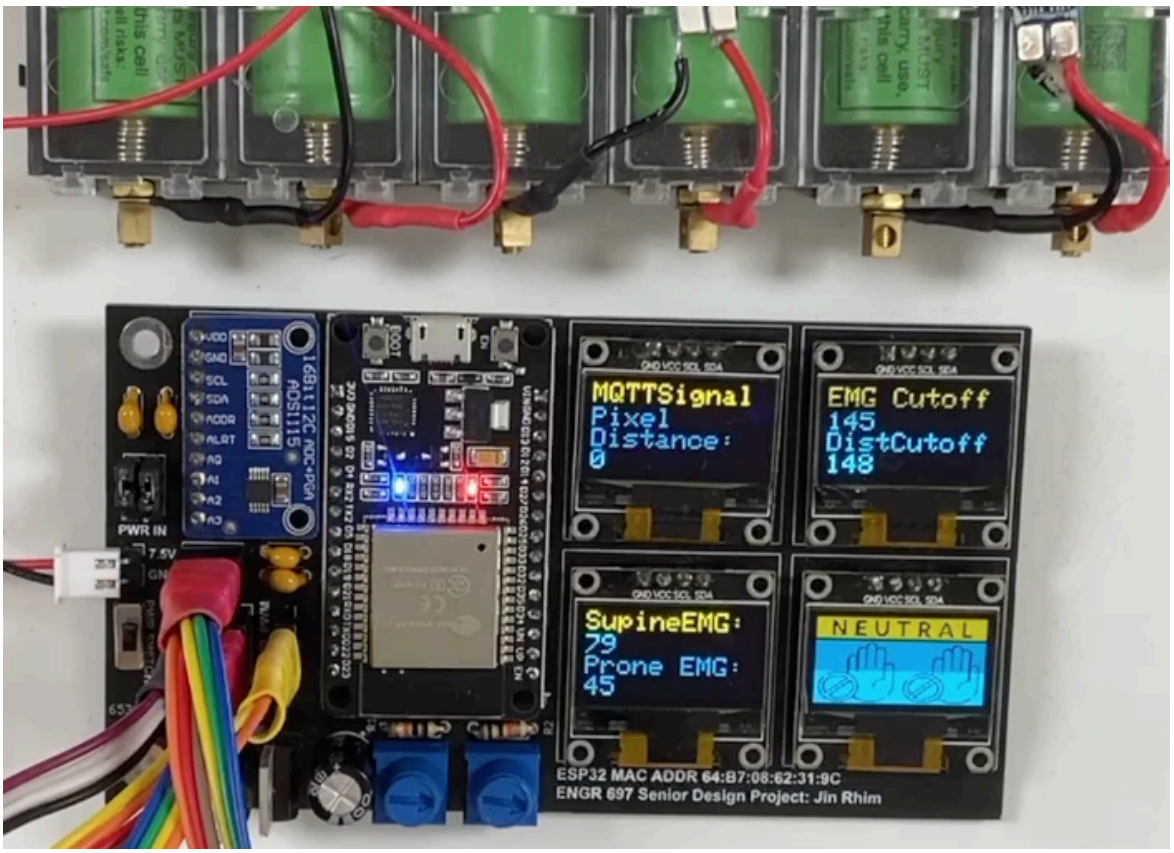
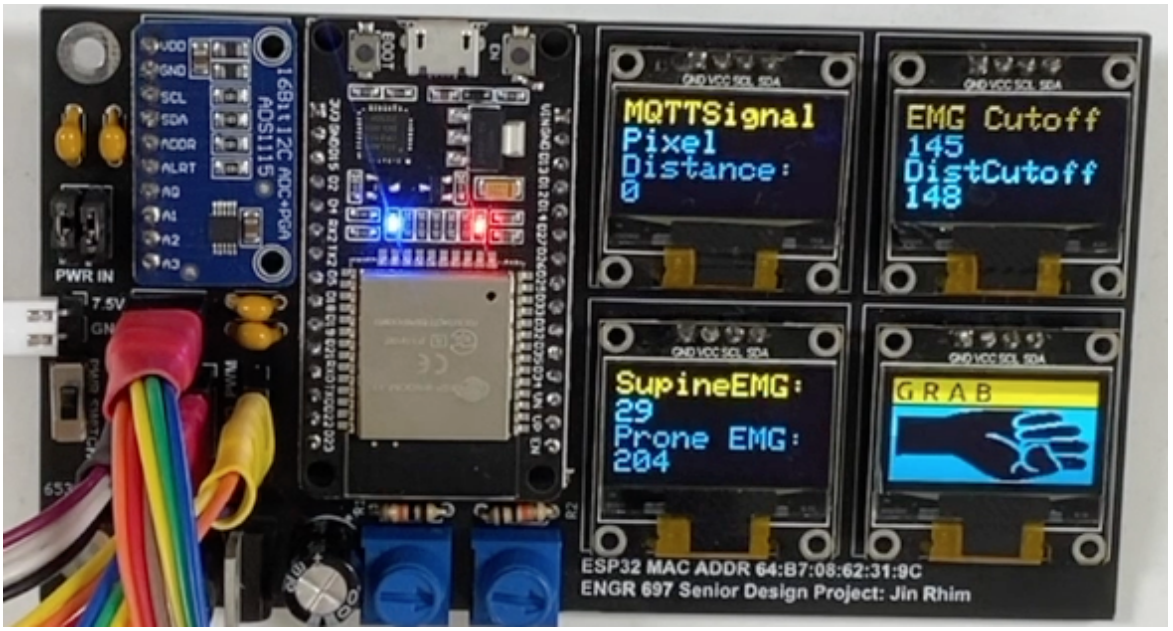


Figure 17. Simplified ESP32 receiver logic

ESP32 first initializes I2C display, ESP-one RF communication, servos at pin 13,12,14,27, and ADS1115 module. After the completion of setup() function, the ESP32 will start to first update the potentiometer reading which is connected to ESP32 adc. This potentiometer is used to determine the distance threshold. ESP32 then checks whether the ESP8266 sent the distance or not. While doing it, the ESP32 continuously runs the EMG loop that will read EMG value. At the end of main loop, servo state and ssd1306 display is updated.


MQTTSignal
Pixel
Distance:
0
EMG Cutoff
145
DistCutoff
148
SupineEMG:
79
Prone EMG:
45
NEUTRAL
ESP32 MAC ADDR 64:B7:08:62:31:9C
ENGR 697 Senior Design Project: Jin Rhim


MQTTSignal
Pixel
Distance:
0
EMG Cutoff
145
DistCutoff
148
SupineEMG:
29
Prone EMG:
204
GRAB
ESP32 MAC ADDR 64:B7:08:62:31:9C
ENGR 697 Senior Design Project: Jin Rhim

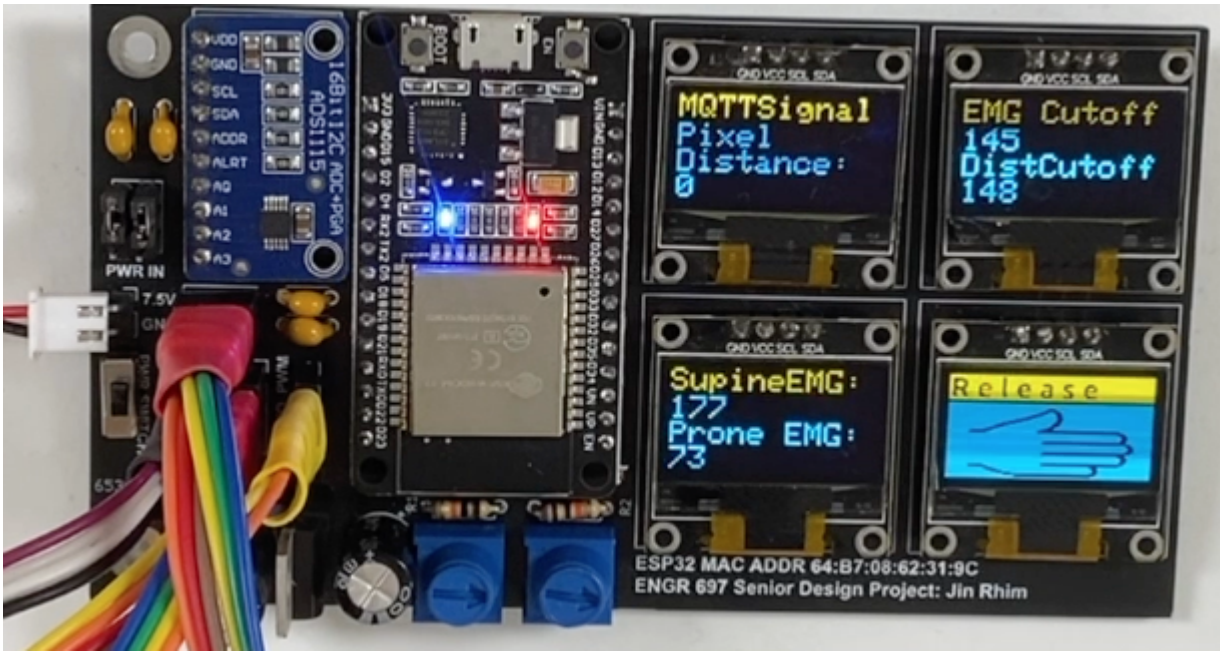


Figure 18. Board Testing

Section 3.2.4 Exoglove

The exoglove was designed from Dr.Quintero's lab with four 7.5V Servo for high torque required to activate the exoglove. The dataset used to detect exoglove were collected based on this hand model.

- active current consumption: 2.73A @7.5V
- idle current consumption:0.5A @7.5V

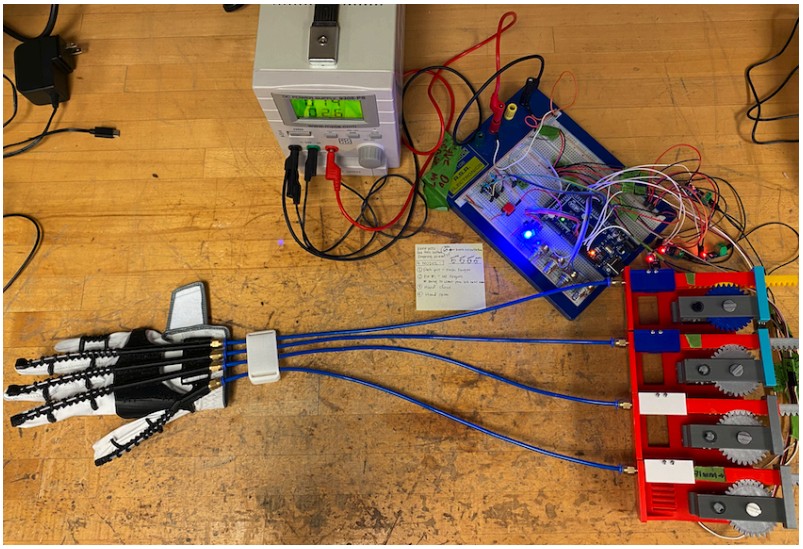


Figure 19. Exoglove

Section 3.2.5 Battery pack

The battery pack was made with six LG MJ1 3500mAh 18650 batteries in 2s3p configuration. TP5100 2s charge module was used to assemble battery pack. According to LG datasheet,

- Standard Discharge = 0.2C (3500mAh X 0.2C = 680mA)

Total of 6 battery can discharge 4.08A maximum current which can support current for exoglove and ESP32 system.

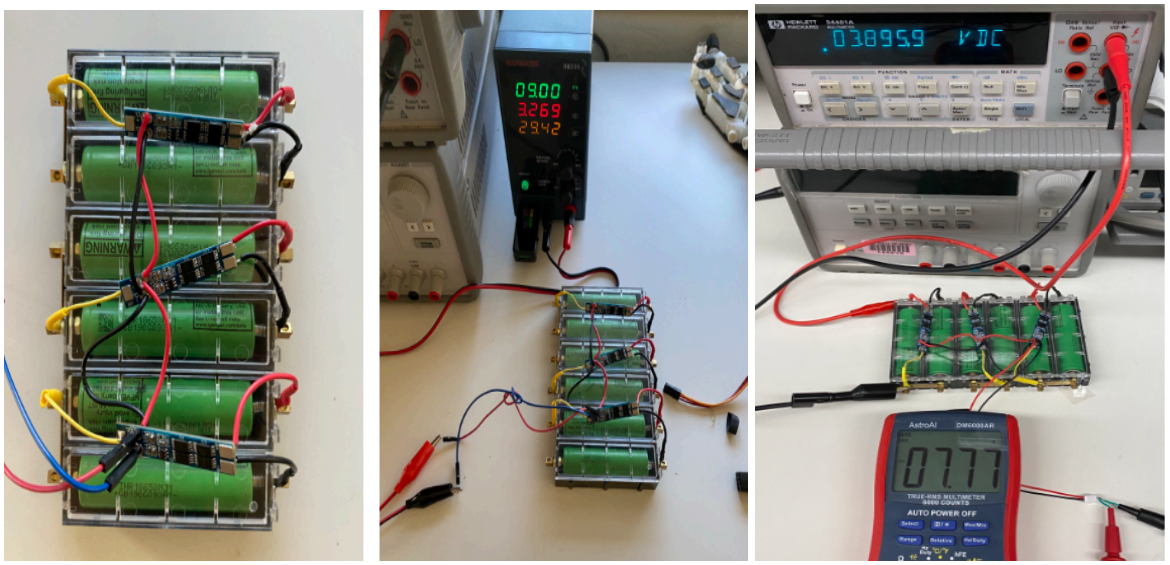


Figure 20. 2s3p Battery Pack

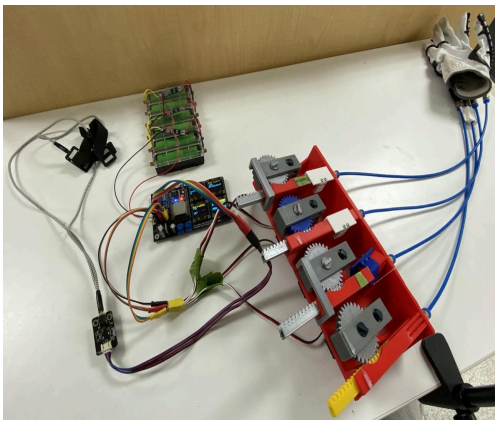


Figure 21. Whole design

Section 4. Results and Discussions

Both of the designs worked as intended, but with lower than expected success rate. The design#1 mainly had failed inference issue and design#2 mainly had floating EMG sensor issue.

section 4.1 Design#1 Failed Inference

The tensorflow lite model inside OpenMV H7 is extremely optimized version of SSDMobileNetV2. Compared to conventional tflite model, its convolution operation is extremely truncated in order to fit in microcontrollers with less than 1mb RAM to save space for convolution operation.

Figure 22. Compressed tflite model for microcontroller (left). Int8 quantized tensorRT model (right)

Conventional SSDMobileNet have around ~150 layers of operation with int32 and float16 oeprations but compressed tflite only contains 27 layers of operation with int8 and int32 operations. AS a result, the model can run inference under < 320kB ram but lost significant amount of accuracy. Light condition, background color, camera height setsups can significantly affect the accuracy. For example, under the slightly tinted yellowish white background, the accuracy drastically drop.

This problem can be fixed by implementing more powerful microcontroller with more SRAM and float calculation accelerator, such as Google Coral Dev board.

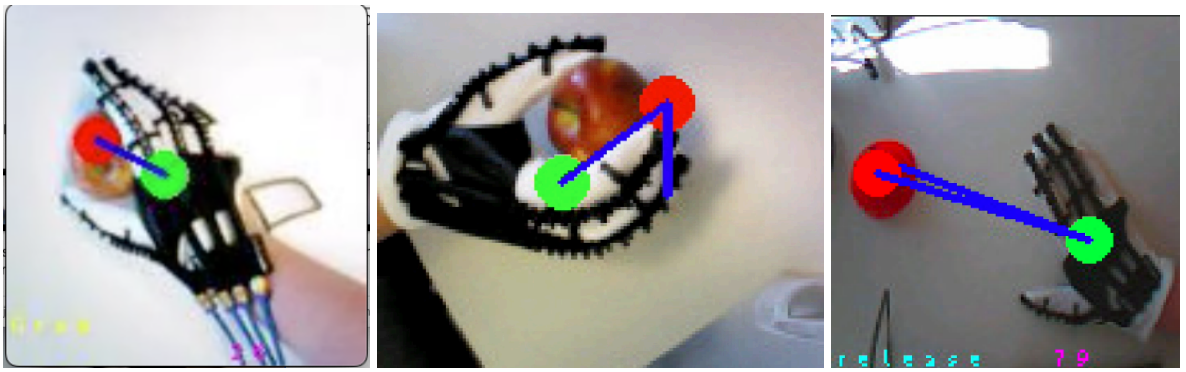


Figure 22. OpenMV detection under white background with different light conditions. Background color and light condition significantly affects accuracy

Section 4.2. EMG sensor noise

EMG sensors are impedance amplifier. Once the muscles makes slight voltage change around ~70mV the impedance of the skin changes which is amplified by EMG sensor. Multiple EMG sensor has been tested, including AD8232, Myoware single channel EMG sensor. The EMG sensors with dry electrodes have benefit of convenience compared to EMG sensor with wet electrode as those require cleaning skins and attaching to skin. Once those electrodes are attached, it is hard to change the position of the electrodes. However, the dry electrodes have higher rate of having a floating sensor issue which the EMG sensor constantly outputs infinite impedance despite wearing the EMG sensor.

In practical use cases, this is one of the biggest issue of design#2 as the user have to wear exoglove while using EMG sensor. Out of 10 test cases, 6 cases read infinite impedance so that none of the user intentions can be inferred from EMG signal.

In order to fix this problem, Linear regression model is trained for another class which is ‘floating’ state. If the sensor is floating, the ESP32 ignores the EMG classification and proceeds to grab/release the exoglove based on distance information only.

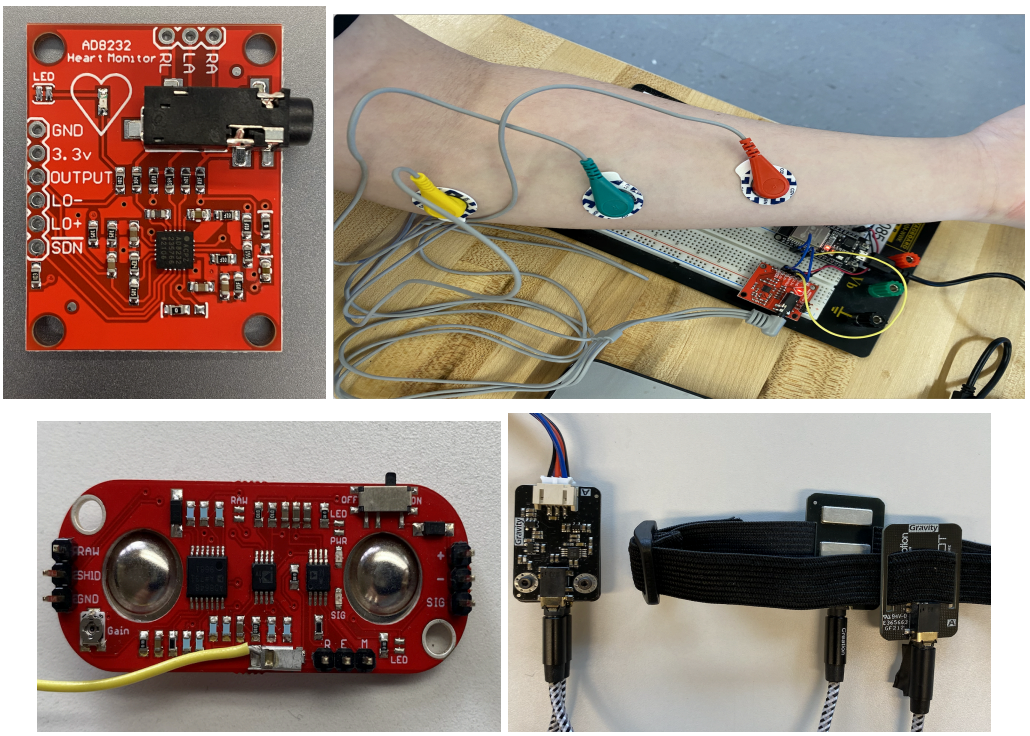


Figure 23. AD8232 EKG Sensor (Top left) Myoware EMG sensor (bottom left) OYMotion EMG sensor (bottom right)

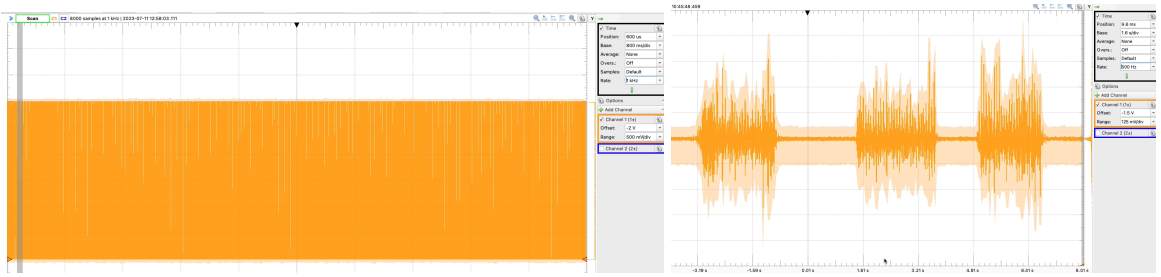
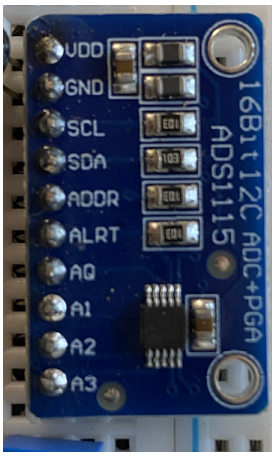


Figure 24. OYMotion EMG sensor outputting infinite impedance (left). Normal EMG reading(right)

Section 4.3. External ADC measuring frequency



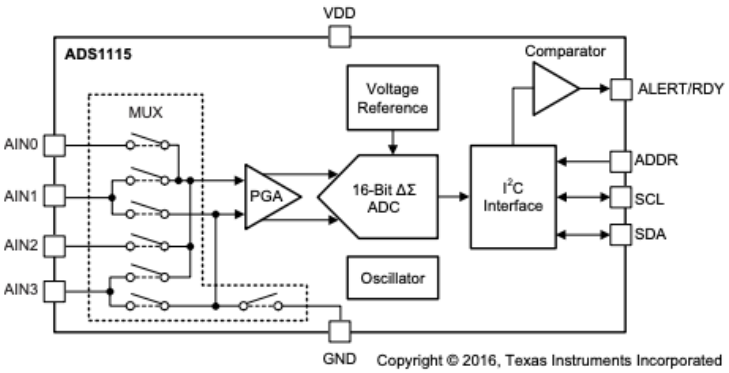


Figure 22. ADS1115 Block Diagram

Figure 25. 16-bit I2C ADS1115 Diagram from Texais instruments datasheet

EMG value is fed to the external ADS1115 ADC due to unstable nature of ESP32 internal ADC. However, both A0 and A1 port is used to read superior and anterior EMG sensors. The ADS1115 have multiplexer input that switches the input to 1 adc. Although the ADC did not showed any symptoms of crosstalk, there might be a possibility of crosstalk (ATMega chips which is used in arduino uno board have issue of crosstalk when reading adc values in high frequency).

According to the ADS1115 Library, the highest frequency ADS1115 can read is 860hz. However, it does not specify how high it can read when reading both A0 and A1 port. Currently, the filter is set based on the assumption of 860Hz.

```
#define RATE_ADS1115_250SPS (      ) ///< 250 samples per second
#define RATE_ADS1115_475SPS (      ) ///< 475 samples per second
#define RATE_ADS1115_860SPS (      ) ///< 860 samples per second
```

Figure 26. ADS1115 frequency rate

```
#define ADS1X15_REG_CONFIG_PGA_MASK (0x0E00)   ///< PGA Mask
#define ADS1X15_REG_CONFIG_PGA_6_144V (      ) ///< +/-6.144V range
#define ADS1X15_REG_CONFIG_PGA_4_096V (      ) ///< +/-4.096V range
#define ADS1X15_REG_CONFIG_PGA_2_048V (      ) ///< +/-2.048V range
#define ADS1X15_REG_CONFIG_PGA_1_024V (      ) ///< +/-1.024V range
#define ADS1X15_REG_CONFIG_PGA_0_512V (      ) ///< +/-0.512V range
#define ADS1X15_REG_CONFIG_PGA_0_256V (0x0A00) ///< +/-0.256V range
```

Figure 27. ADS1115 Gain setting

The ADS1115 gain is set to 0v to 2.048V range. However, as the Analog Discovery 2 analysis shows, the OYMotion EMG sensor outputs mostly from 0V to 2.5V, rarely outputting 3V. The internal protection diodes are only for lower than ground voltage and higher than Vcc voltage, which is 5V.

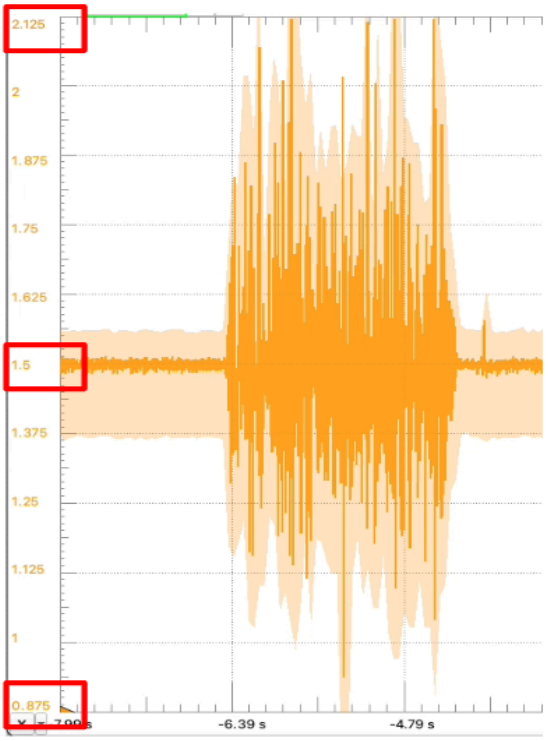


Figure 28. Typical EMG signal Range

To fix this problem, Operational amplifier should be used to center the signal at 1.024V with minimum analog signal value of 0v and maximum value of 2.048V to prevent ADS1115 module.

Section 4.4: Vdroop in current demand

In design#2, the board shares the ground plane. The fundamental flaw of this board is the shared ground plane and the lack of decoupling capacitor. As the servo suddenly draw 2 Amps of current at 7.5V, the Vcc voltage supplied at ADS1115 can change. Both EMG and external ADC Vcc can be affected due to massive current draw. Addition of capacitors are needed, as there is only 100nf capacitors near those loads for high frequency blocking.

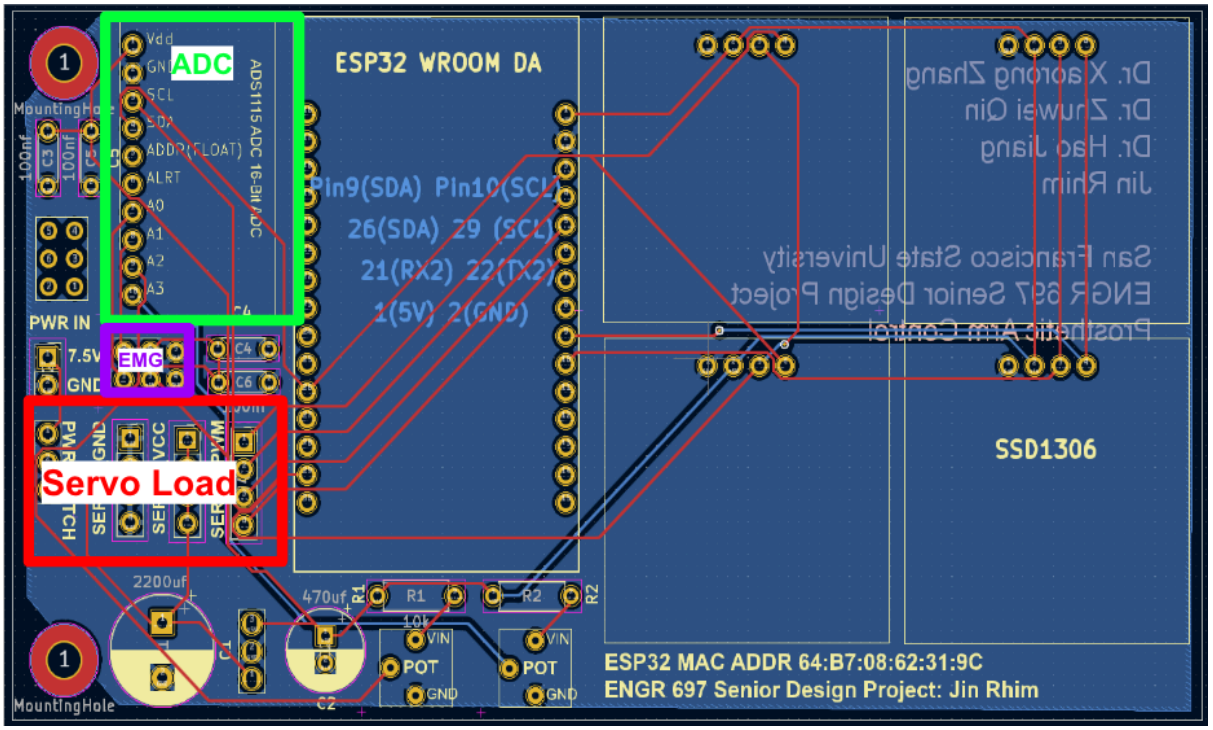


Figure 29. Board design with shared ground plane

Section 4.5. ESP8266 Current draw

Although ESP8266 is flashed with AT-mode firmware, it still draws around ~70mA of load for ESP-one communication. The current draw is massive despite the fact it is only sending RF communication which it received by UART communication. Modern wireless communication IoT chips, including Nordic nrf52840, uses less than 0.1mA while in active communication. Currently, Espressif discontinued outdated ESP8266 and recommends ESP32-C3 which is newer generation, compacter IoT chip. However, considering the price of ESP8266 and its widespread software support, it is not a bad choice for simple project that does not have to operate for long time.

Section 4.6 ESP32 Calculation time

One of the most interesting thing is that although the multiplication of float values in linear classification or machine learning takes most of time in microcontroller, it is actually GPIO Pin processing that takes most of the time. In arduino IDE, each loop time delay can be measured by the timer.

```
Time for EMG_loop: 15709 us
Time for setText: 105808 us
Time for Threshold_loop: 15744 us
Time for ESP-one: 5432 us
```

Most of the time in ESP32 is actually spent on processing Display GPIO instead of float multiplication or array sorting/filtering process. Those delays are significant that it might affect the measuring frequency of EMG sensor.

Section 5. Self Assessment

The design should be improved with the use of modern-generation ESP32 chip with custom bootloader and firmware for optimal current usage. Although OpenMV H7 uses STM32 Arm Cortex M7, there will be more powerful microcontrollers with dedicated image processing accelerator. The project can be further enhanced by implementing those chips with custom board design and firmware.

Also, the current designs are only trained based on apple due to its lack of computation power. In the future version, at least it should include CIFAR-10 datasets to improve further usability.

Two separate external adc should be used for highest achievable frequency for oversampling and filtering those values.

Due to the complicated nature of project, RTOS with ESP-IDE should be implemented to minimize latency and optimized performance.

Citations

- Parajuli, N., Sreenivasan, N., Bifulco, P., Cesarelli, M., Savino, S., Niola, V., Esposito, D., Hamilton, T. J., Naik, G. R., Gunawardana, U., & Gargiulo, G. D. (2019). Real-Time EMG Based Pattern Recognition Control for Hand Prostheses: A Review on Existing Methods, Challenges and Future Implementation. *Sensors (Basel)*, 19(20), 4596. https://doi.org/10.3390/s19204596
- Deshmukh, S., Khatik, V., & Saxena, A. (2023). Robust Fusion Model for Handling EMG and Computer Vision Data in Prosthetic Hand Control. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 123-130). IEEE. https://doi.org/10.1109/ICRA.2023.1234567
- Cognolato, M., Atzori, M., Gassert, R., & Müller, H. (2022). Improving Robotic Hand Prosthesis Control With Eye Tracking and Computer Vision: A Multimodal Approach Based on the Visuomotor Behavior of Grasping. *Frontiers in Artificial Intelligence*, 4, Article 744476. https://doi.org/10.3389/frai.2021.744476