Notes for the 2025 Summer Intern

Luke Boyd

If you need any further help or information please contact me at [lboyd@mines.edu](mailto:lboyd@mines.edu)

For extra EMG Pads use this link: <https://bio-medical.com/covidien-kendall-disposable-surface-emg-ecg-ekg-electrodes-1-24mm-50pkg.html?srsltid=AfmBOoo-Sdy-UEVerChJYRAGt198krXs1igWDr_WBAscLMPrXQQHZpk6>

[Myoware.com](https://myoware.com/)

The overall goal of this project is to control a simple robotic hand using inputs from electromyographic sensors. It uses an ESP32 microcontroller to control the hand and 3 servos. The hand must be powered by a 9-12V battery connected to the black PCB with the two wires coming out of it, I recommend one of the 12V batteries sitting around the lab. Make sure to also press the button on the black PCB to turn on the power. Along with power from this battery it must also be connected via USB-C to a laptop to run the scripts that can either be run through Python or the Arduino IDE. Currently in this folder there are Arduino scripts to test the robotic hand positions and take EMG data as well as python scripts that control the hand via machine learning algorithms.

The main body of code that is being used for this project is written in Python 3.13 and uses multiple python libraries that will need to be installed before use. I have written a list below of some but if I forgot to include on I’m sure your IDE will let you know.

Libraries Required (some might not be totally required but are helpful):

* colorama
* contourpy
* cycler
* fonttools
* joblib
* kiwisolver
* matlplotlib
* numpy
* packaging
* pandas
* pillow
* pip
* pyparsing
* pyserail
* dateutil
* pytz
* scikit-learn
* scipy
* seaborn
* six
* threadpool
* tinyml
* tqdm
* tzdata

The main body of the code is broken up into 4 scripts, CreateClassifier, TestClassifier, extract\_features and IntegrationTest. Additionly scipts Demonstration and SerialTest are also used. This is a brief explanation of each but a more indepth review of machine learning protocols would be useful if you are not familiar. These are all in the .venv folder

CreateClassifier

* This code creates the machine learning classifier that will determines which grasp type is presented.
* It used the extract\_features code to find specific features from the training data provided
* The training data is from the open source dataset called [NinaPro](https://ninapro.hevs.ch/instructions/DB1.html). We are using dataset one, exercise C(3) data because it is the closest to the real world grasp types we want to explore in this project.
* When you look at the exercise 3 grasp you will notice that there are 23, yet we are classifying to 5 grasp types (0-4). These 23 were truncated down to 5 styles (Rest(0), Cylinder(1), Pinch(2), Ball(3), Inverse Pinch (4)). This was done using the MATLAB script reclassify.m in the archive folder. The data was also rebalanced using the rebalance.m MATLAB script also in the archive folder because there was a lot more data for Rest (0) then the other four positions combined.
* The data we used was also pulled from subjects 23, 24 and 25 because they were physically similar and it is hoped that this gave a more consistent signal while taking data from multiple subjects.

TestClassifier

* TestClassifier is the next step after CreateClassifier, it simulates real world use by giving the classifier discrete sections of proportioned data from a different test subject, usually subject 1. This allows us to see the accuracy of the classifier on different people

extract\_features

* Extract features takes the features that we use to classify the grasp types into different groups
* We take the average, RMS and mean absolute value over a chosen time step among other features for each sensor used
* More about this can be found in the EMG protocol documentation

SerialTest

* SerialTest verifies that the python script can accurately connect to the ESP32 microcontroller that we are using to control the hand
* **The Arduino IDE cannot be open when connecting over serail, it will override the signal and not let it go through**
* **This also cannot be run through a python IDE like pycharm or visual studio, it must be run through command prompt** 
  + Protocol to run this through cmd can be found in the IntegrationTest file, commented in the top of the script. Make sure to change the file location to where it is stored on your computer.

IntegrationTest

* IntegrationTest is simply a combination of SerialTest and TestClassifier, being that it runs the test classifier script but over serial and sends the output to the ESP32.

Demonstration

* Demonstration is a similar code to IntegrationTest but gives the user more control over the EMG data sent to the classifier.
* The code will create a popup that displays what hand position the robotic hand is trying to replicate, if it is replicating it correctly, a graph of the data being sent.
* There is a popdown box that allows you to select which grasp position data is being sent to the classifier.
* This also like other serial based scripts must be run through cmd.

**EMG Collection Protocol**

Preparation

* Make sure whichever arm you are using is shaved, even if you don’t have much hair on your forearm, it should still be shaved every few days so that the hair does not interfere with the sensors
* Wash the forearm with isopropyl alcohol if available and if not wash with soap and water and dry to remove any oils or debris from the forearm
* Make sure that you never use the sensor pads more than twice (only remove and replace them once), before replacing the pads. The pads can stay on your arm for as long as needed but should not be used more than 2 times otherwise the data collected is degraded

Placement

* The sensors are labeled A0, A1 and A2, make sure that A0 is placed on the blue region, A1 is placed on the green region and A2 is placed on the red region shown below

A person's arm with multiple fidget pins

Description automatically generated

Collection Prep

* Use the EMGDataCollection.py script to collect data
* Run this through command prompt
* Make sure that the COM# in the script is set correctly, this can be checked through the Arduino IDE to check which COM the Arduino uno is connected to
* Check the band rate as well but it is set at 9600 which should work correctly

Collection

1. Set the grasp that you want to collect data for, there are 5 options: Cylinder, Pinch Ball, Inverse Pinch and All Transitions. Cylinder, Pinch Ball and Inverse Pinch will all have you change between a rest state of and open hand and the active grasps selected. All Transitions will have you change your hand position between each of the grasps to allow us to collect data of the change between every grasps position.
2. Set the amount of data points to gather for both the rest and activated state of the grasp being collected, by default each is set to 100 data points but this could be increased if seen necessary
3. Set the number of cycles you want the test to take, a cycle is a full collection of the rest and active state of a grasp
4. Choose the file path that you want the data to be saved at, this requires both specifying the file path and the file name
   1. Naming convention for the file will be as follows, GX\_R##\_A##\_C#\_S.csv
      1. GX will indicate the grasp type collected, this can be G1 for cylinder, G2 for pinch, G3 for ball, G4 for inverse pinch and AT for all transitions
      2. R## is the number of rest data points, so for 100 rest data points it would be R100
      3. A## is the same for the number of active data points, so A1000 would be 1000 active data points
         1. For any number of rest or active data points over 9999, use a #k notion for thousand, so 100,000 would be 100k although this amount of data points is probably way to high
      4. C# is the number of cycles collected, so C20 would be 20 cycles
      5. S stands for the speed, this can be set to S, N or F for slow, natural or fast. This is the speed that you should be changing between grasps, slow would take around a second, natural is the natural rate the you change grasps and fast would be faster than natural maybe 2 time faster.