Introduction:

My goal was to create a myoelectric prosthetic. I did this through 3D printing and a myo gesture band that picks up an EMG (Electromyogram, the electrical signal that control the muscle) signal from the wearer and sends it to an arduino which then talks to the servos so that it may recreate the gesture.

Establishing the Bluetooth Connection:

I created a sync from the Myoband to the HM-11 Bluetooth chip with help from online forums where people attempted to achieve the same thing. This sync needed to be accomplished because there does not seem to be a way for the myo gesture band to talk to the arduino board without a computer in between directly so this bridged the gap. To get the bluetooth chip to connect to the myoband I had to upload custom firmware to the chip. To flash the chip a CC DeBugger is needed. One must be careful in soldering the Bluetooth chip so that none of the connections touch each other. The connections will be shown below.

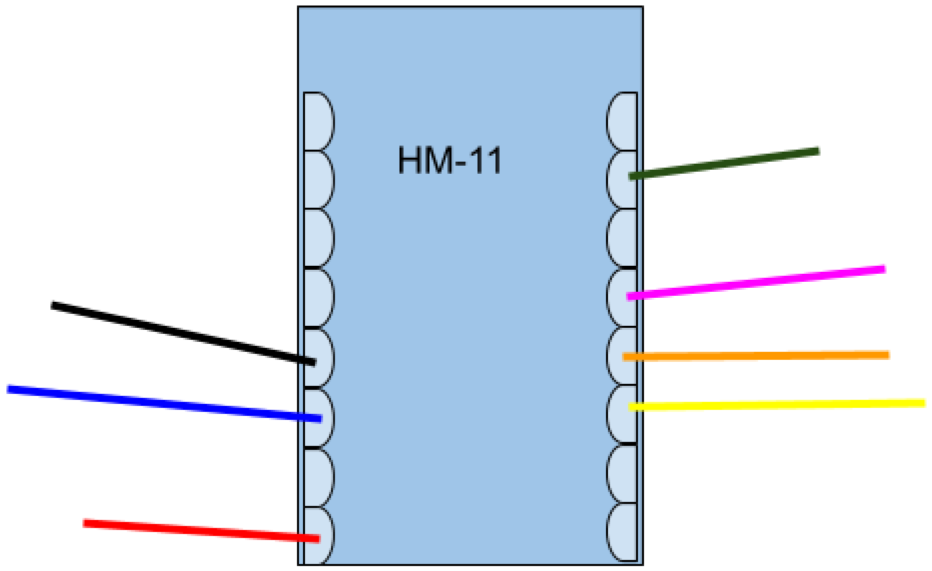


Figure 1. Bluetooth chip wires soldered to correct ports. Black-GND, Blue-Reset, Red-3.3V, Green-Target Voltage, Pink-RX, Orange-Debug Data, Yellow-Debug Clock.

To attach to the debugger the connections between the HM-11 bluetooth chip and DeBugger need to be made correctly in order to successfully flash the code onto the chip.

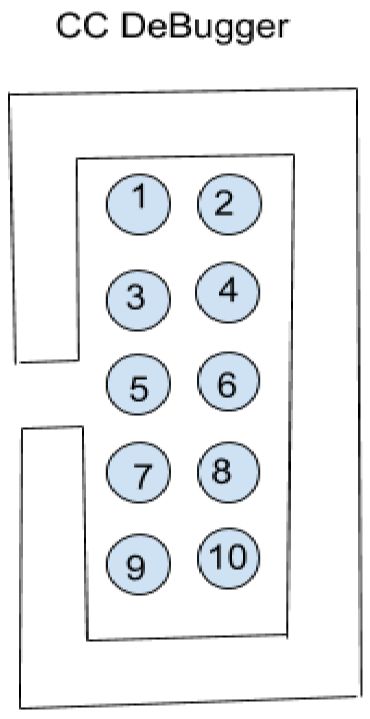
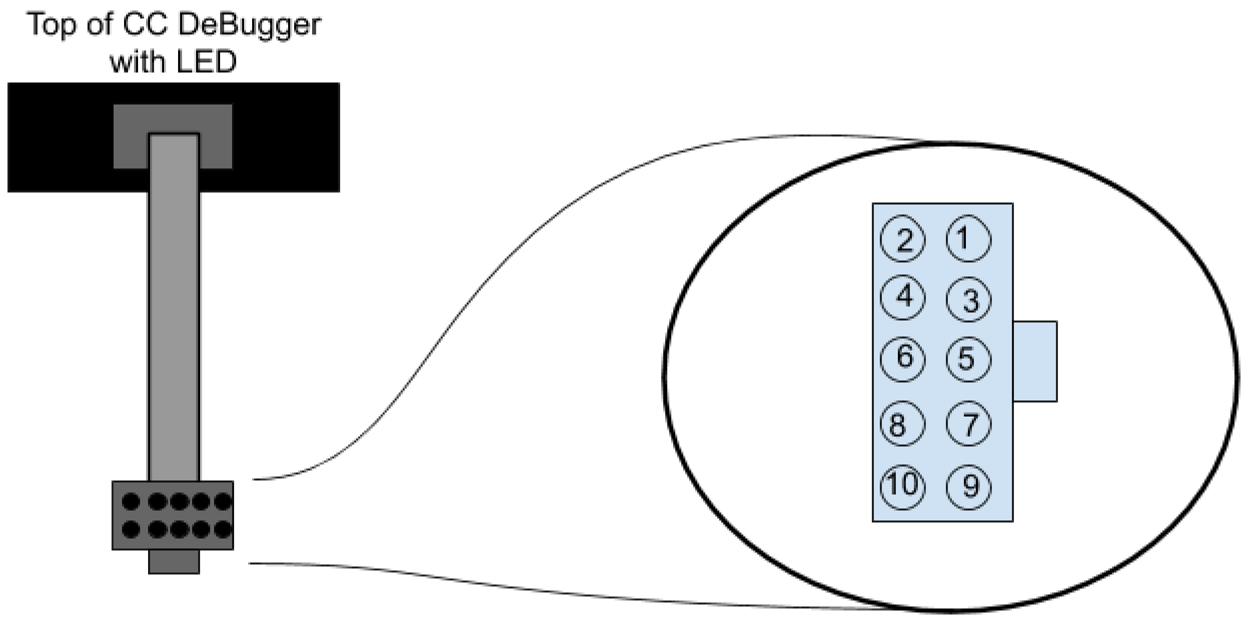


Figure 2. Shows the CC Debugger and numbers the ports with the reference point that faces the top of the Debugger.

Using the large adapter that creates female ports for the bluetooth chip to connect into the Debugger. This will create something like the image below.



**The adapter reverses the ports and so are relabeled above for convenience.

Connections Bluetooth Chip to Debugger:

Now connect wires from HM-11 Bluetooth chip to the female ports of the CC Debugger adapter. Connections shown in table below.

Black Wire	Port 1 on adapter
Blue Wire	Port 7 on adapter
Red Wire	Port 10 on adapter
Green Wire	Port 2 on adapter
Orange Wire	Port 4 on adapter
Yellow Wire	Port 3 on adapter
Pink Wire	Port 9 on adapter

Connecting the Chip and Debugger to the computer:

Once connections are made between Bluetooth Chip and CC Debugger you can begin to flash the device.

Plug in CC Debugger to the WINDOWS (This step must be down on a windows computer due to the flashing firmware does not work on a Mac) computer via the USB Cable that came with the Debugger..

-The light should be red. Press Reset on the top. If the chip is detected the light will turn green.

*If light doe
s not turn green check and make sure all wires are in the correct place on the DeBugger Adapter and on the bluetooth chip make sure they are in the correct position and soldered well.

Flashing the Firmware to the Bluetooth chip:

Download Myobridge firmware (link directly below)
http://github.com/vroland/MyoBridge/blob/master/myobridgI'm
e_firmware/Hex/MyoBridge_CC2541.hex

-Right click "Raw", Save as…

Now to get flash program ready to go follow the link www.ti.com/tool/flash-programmer

*Use version 1

Follow the SmartRF flash programmer and make sure the CC DeBugger is showing and highlighted in blue.

In the flash programmer click the "..." next to the "flash image bar" within the smart setup add the Myobridge firmware that was just downloaded (this should be .hex)

In the lower left under “actions” select “Erase, program and verify”

Then click “perform actions” bar at the bottom - this starts the flashing of the chip

Wait until the bar is fully loaded and it says “Erase, program, and verify OK”

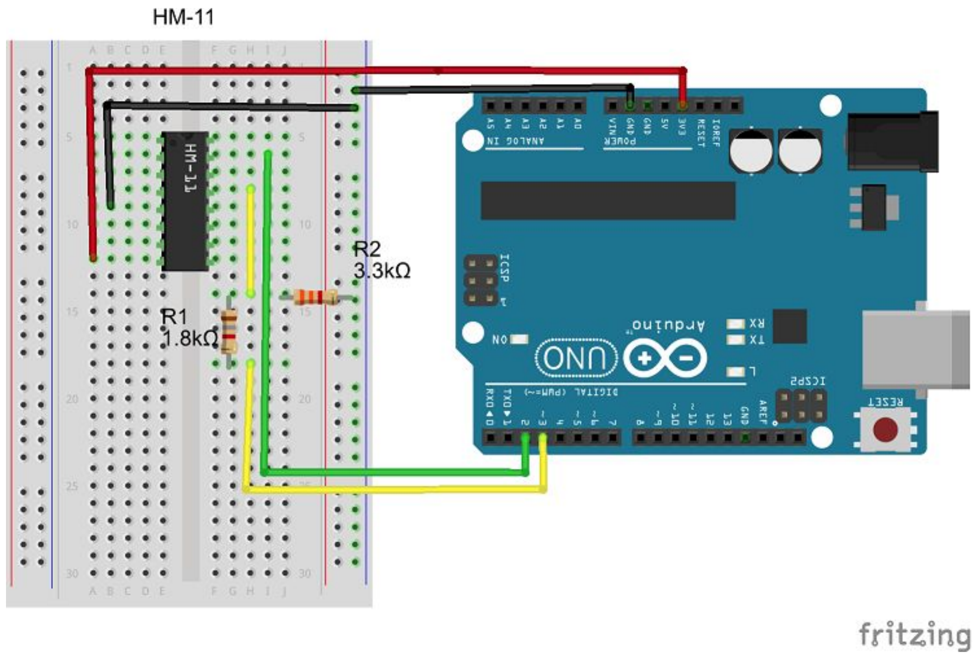
*Once verified the bluetooth chip is ready to go as long as the CC Debugger light is green (if not double check the download)
**You may now unplug and set aside the DeBugger it will not be needed again but keep the Bluetooth Chip and do not remove any of its wires.

Connecting Bluetooth Chip to Arduino Uno:

Wires from Chip	**Port on Arduino Uno**
Red Wire	3.3V
Black Wire	GND(the one directly above Vin)
White Wire (via a voltage divider)	Port 3
Green Wire	Port 2

*the rest of the wires on the bluetooth chip are not necessary now but it doesn’t hurt to keep them

Image of the connections are below and from
https://github.com/vroland/MyoBridge/wiki/Getting-Started-with-MyoBridge-Library



Testing the connection with the Myo Gesture Band

-Plug the Arduino Uno into the computer (Mac or Windows)

-Download the Arduino IDE

https://www.arduino.cc/en/Main/Software

-Now you have to add the MyoBridge library to the Arduino for the code to work
https://github.com/vroland/MyoBridge

-Click clone or download

-click download zip

-Go into zip folder

-> Arduino

->libraries

Here there will be a folder titled MyoBridge Create a new folder with only the contents of this folder inside and then convert it to a Zip file (right click and there should be an option to Compress or zip the folder, do this)

-Open the Arduino IDE

-click on sketch in the top left

-select “include libraries”

-click ‘add zip library’

-browse and insert the myobridge zip that was made.

Coding for the Arduino

The code takes the information received on the Bluetooth chip it gets from the armband and translates it into instructions that the servos can follow.

*Ideally linear actuators would be used in this prototype and a slight change in the code would need to be made. The changes would need to be in the degrees of the servo and since the use of the Ada hand is in place you can separate the index_middle servo and ring_pinkie servo and make them their own. This would need to be done throughout the code to prevent error. However I only had normal servos on hand to use so I had to modify my arm design and kept the same code from poparaguay on github found below.

Create a new blank sketch within the IDE and copy and paste poparaguay the code. Modify if needed then click upload.
https://github.com/poparaguay/mypo/blob/master/Myo_Arduino_3servos.ino

Code:

```
#include <Servo.h>
#include <MyoBridge.h>
#include <SoftwareSerial.h>

//SoftwareSerial connection to MyoBridge
SoftwareSerial bridgeSerial(2,3);

//initialize MyoBridge object with software serial connection
MyoBridge bridge(bridgeSerial);

//declare a function to handle pose data
void handlePoseData(MyoPoseData& data) {

  //convert pose data to MyoPose
  MyoPose pose;
  pose = (MyoPose)data.pose;

  //print the pose
  Serial.println(bridge.poseToString(pose));

  move_hand(pose);

   //delay(15);
}

//a function to inform us about the current state and the progess of the connection to the
Myo.
void printConnectionStatus(MyoConnectionStatus status) {

  //print the status constant as string
  Serial.println(bridge.connectionStatusToString(status));
}


Servo servo_thumb;  // servo object thumb
  #define PIN_SERVO (7) //Pin 7
Servo servo_index;  // servo object index
  #define PIN_SERVO (6) //Pin 6
Servo servo_middle; //servo object middle
  #define PIN_SERVO (5) //Pin 5
Servo servo_ring;  // servo object ring
```

```
  #define PIN_SERVO (8)  // pin 8
Servo servo_pinky; // servo object pinky
  #define PIN_SERVO (A0, 9) //Pin 9

void SetStrokePerc(float strokePercentage)
{
  if ( strokePercentage >= 1.0 && strokePercentage <= 99.0 )
  {
    int usec = 1000 + strokePercentage * ( 2000 - 1000 ) / 100.0 ;
  }
}
void SetStrokeMM(int strokeReq,int strokeMax)
{
  SetStrokePerc( ((float)strokeReq) / strokeMax );
}

void setup() {

  //initialize both serial connections
  Serial.begin(115200);
  bridgeSerial.begin(115200);

  //wait until MyoBridge has found Myo and is connected. Make sure Myo is not
connected to anything else and not in standby!
  Serial.println("Searching for Myo...");
  for (int i=0; i<10;i++){digitalWrite(13, HIGH);delay(50);
  digitalWrite(13, LOW);delay(50);}
 //initiate the connection with the status callback function
  bridge.begin(printConnectionStatus);
  digitalWrite(13, HIGH);
  delay(15);
  digitalWrite(13, LOW);
  Serial.println("connected!");
  //declare a storage array for the hardware address
  byte address[6];
  //get the address and store it in our array
  bridge.getHardwareAddress(address);
  //print the hardware address in HEX format
  Serial.print("Hardware Address: 0x");
  for (int i=0;i<6;i++) {
    Serial.print(address[i], HEX);
  }
  Serial.println();
  //get the current battery level and print it
  byte batteryLevel = bridge.getBatteryLevel();
  Serial.print("Battery Level: ");
  Serial.println(batteryLevel);

  //short vibration to show we are ready
  bridge.vibrate(1);

  //set the function that handles pose events
  bridge.setPoseEventCallBack(handlePoseData);
  //tell the Myo we want Pose data
  bridge.enablePoseData();
  //make sure Myo is unlocked
  bridge.unlockMyo();

  //You have to perform the sync gesture to receive Pose data!

  servo_thumb.attach(7);  // attaches the servo on pin 7 to the servo object
  servo_index.attach(6);  // attaches the servo on pin 6 to the servo object
  servo_middle.attach(5); // attaches the servo on pin 5 to the servo object
  servo_ring.attach(8);  // attaches the servo on pin 8 to the servo object
  servo_pinky. attach(9); // attaches the servo on pin 9 to the servo object
}
void loop()
{
  //update the connection to MyoBridge
  bridge.update();

  //MyoPose pose;
  //pose = (MyoPose)data.pose;
  //char a = bridge.poseToString(pose);
}
void move_hand(int pose)
{
  int d = 10;
  int delayMS = 15;
  int i = 1000;
  for ( i = 1000; i < 300; i += d )
  {
    SetStrokePerc(i);
    delay(delayMS);
  }
```

```
 for ( i = 300; i > 1000;  i -= d )
 {
   SetStrokePerc(i);
   delay(delayMS);
 }
#define POS_THUMB_OPEN 1000
#define POS_THUMB_CLOSE 300
#define POS_INDEX_OPEN 1000 // It is done
#define POS_INDEX_CLOSE 300 // It is done
#define POS_MIDDLE_OPEN 1000
#define POS_MIDDLE_CLOSE 300
#define POS_RING_OPEN 1000
#define POS_RING_CLOSE 300
#define POS_PINKY_OPEN 1000
#define POS_PINKY_CLOSE 300

int pos_thumb=300;
int pos_index=300;
int pos_middle=300;
int pos_ring=300;
int pos_pinky=300;
int flag_openhand=300;

 switch (pose) {
      case 0://MYO_POSE_REST
    pos_thumb = POS_THUMB_OPEN;
    pos_index = POS_INDEX_OPEN;
    pos_middle = POS_MIDDLE_OPEN;
    pos_ring = POS_RING_OPEN;
    pos_pinky = POS_PINKY_OPEN;
    flag_openhand=1;
    LED_blink();
    break;
   case 1://MYO_POSE_FIST
    pos_thumb = POS_THUMB_CLOSE;
    pos_index = POS_INDEX_CLOSE;
    pos_middle = POS_MIDDLE_CLOSE;
    pos_ring = POS_RING_CLOSE;
    pos_pinky = POS_PINKY_CLOSE;
    LED_blink();
    break;
   case 2://MYO_POSE_WAVE_IN
    pos_thumb = POS_THUMB_OPEN;
```

```
          pos_index = POS_INDEX_CLOSE;
          pos_middle = POS_MIDDLE_CLOSE;
          pos_ring = POS_RING_CLOSE;
          pos_pinky = POS_PINKY_OPEN;
          LED_blink();
          break;
        case 3://MYO_POSE_WAVE_OUT
          pos_thumb = POS_THUMB_OPEN;
          pos_index = POS_INDEX_OPEN;
          pos_middle = POS_MIDDLE_OPEN;
          pos_ring = POS_RING_OPEN;
          pos_pinky = POS_PINKY_OPEN;
          flag_openhand=1;
          LED_blink();
          break;
        case 4: //MYO_POSE_FINGERSPREAD
          pos_thumb = POS_THUMB_CLOSE;
          pos_index = POS_INDEX_OPEN;
          pos_middle = POS_MIDDLE_OPEN;
          pos_pinky = POS_PINKY_CLOSE;
          pos_ring = POS_RING_CLOSE;
          LED_blink();
          break;
        case 5://MYO_POSE_DOUBLE_TAP
          LED_blink();
          break;
      }
     servo_thumb.write(pos_thumb);
     servo_index.write(pos_index);
     servo_middle.write(pos_middle);
     servo_ring.write(pos_ring);
     servo_pinky.write(pos_pinky);
    }
    void LED_blink ()
    {
        digitalWrite(13, HIGH);delay(50);
        digitalWrite(13, LOW);delay(50);
    }
```

I used the Ada Hand from Open Bionics printed with Ninja Flex (**https://www.openbionics.com/obtutorials/ada-v1-assembly**). I designed the arm that would be used as the Arduino/battery holder