

James Sass

ELECTRONICS REPORT MYO RC CAR

user

Table of contents

Front cover	1
Table of contents	2
Introduction	3
Aim	3
Design	3
Equipment	4
Code	9
Building	13
Compilation	15
Conclusion	16
References	16

Introduction

What is an RC car – a RC car is a model vehicle that can be controlled from a distance using a specialised transmitter or remote. The RC can either mean 'radio-controlled' or 'remote-controlled'. In this project the transmitter will be a Myo Armband.

Aim

The aim of my project is to create a working RC car that will be controlled through the movements of a Myo Armband. The coding of the car will be created in Arduino, also including the downloadable library of the Myo Armband, Myoduino. The program must be able to detect specific movements of the hand through the use of the Myo Armband and make the correct wheels on the car react.

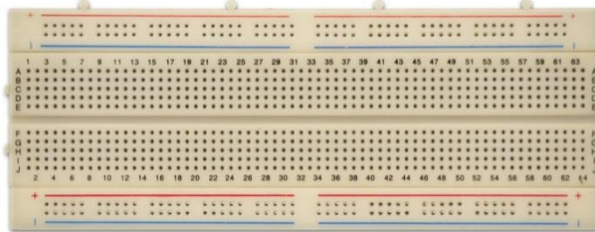
This report will contain information on the design of the project, the code and what each part does and images. Images will be used where appropriate, both drawn and photographed.

Design

My project will be created using Arduino coding to make it work. The case that will hold the wiring, Arduino uno board and DC motors will be 3D printed to the size of 190mm x 85mm x 82mm. the top of the case will also be 3D printed to match the size of the case. It will be 3mm thick. There will also be eight corner like piece being printed that will be used to lock the case top to the case and also hold the motors in position. The wiring will be done on a breadboard, connecting to the Arduino uno and the two motors. The motors will be for the two wheels that will go on the side of the case.

Equipment

- A breadboard - A breadboard is a solderless device which is used for creating temporary prototypes of circuits with electronics and is used for testing circuit designs. It is used by inserting leads or terminals into the holes and then making connections through wires where appropriate. This is where most of the projects wiring and components will go.



- Connecting wires - A jump wire is an electrical wire or group of them in a cable with a connector or pin at each end which is normally used to interconnect the components of a breadboard or other prototype or test circuit. These were simply used to wire up the needed parts to allow the circuit to be complete



- Arduino Uno - an Arduino Uno has fourteen digital Input/ output pins, six analog inputs. This project doesn't require that many however this is the smallest usable Arduino board available. The poses preformed while wearing the Myo band will each have their own designated output. The power and ground wires will also be connected to this.



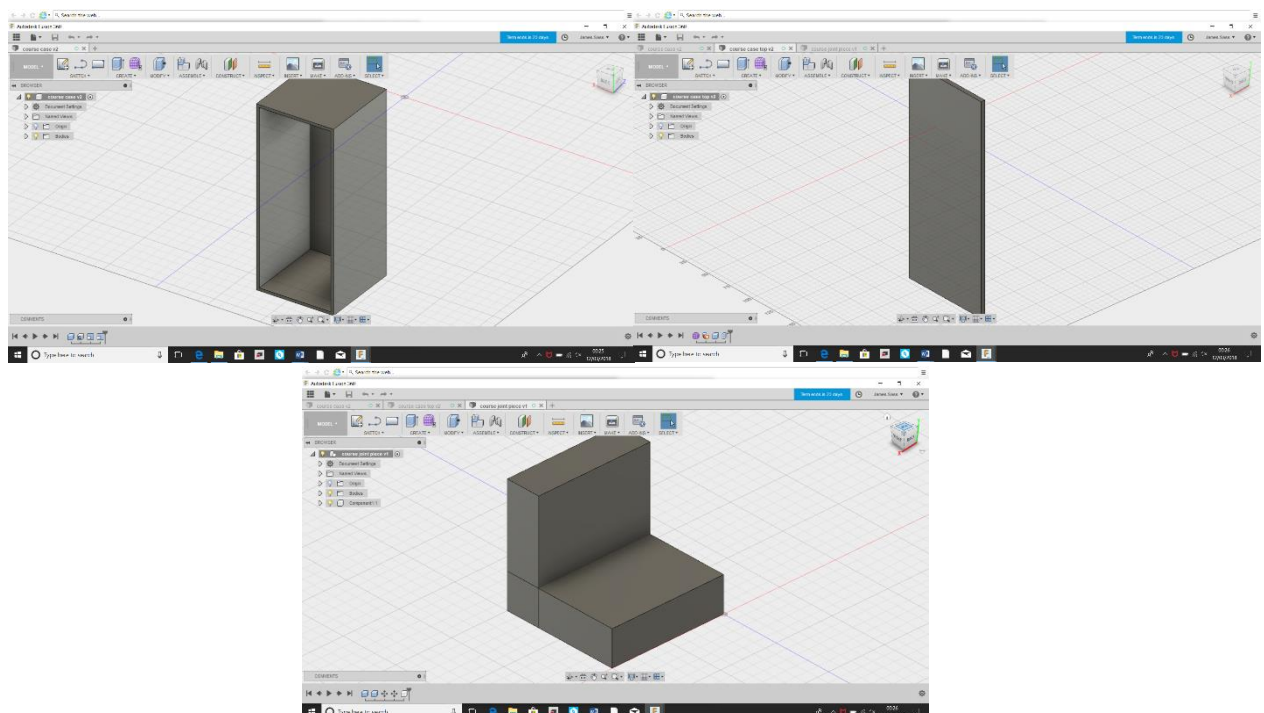
- Myo gesture control armband - manufactured by Thalmic Labs, the Myo band allows the wearer to control technology wirelessly through the use of hand motions. When worn on the forearm, it senses electrical activity in the muscles through the use of EMG sensors and can recognize gestures with a gyroscope, magnetometer and accelerometer. This will be used to direct the RC car based on certain gestures. For example, when a wave in motion is made by the hand, the RC car will start turning right.



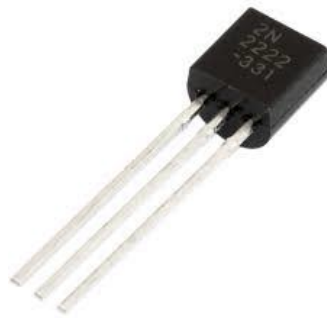
- DC motor - A rotary electrical machine which convert a direct current of electrical energy into mechanical energy, usually used to turn something. There are two of these, each one connected to a wheel and held in place on either side of the case(inner). These will be used to turn the wheels when the right gesture is picked up by the Myo Band.



- Autodesk Fusion 360 - Software which can be used to design objects to be 3D printed. This program was used to design the case, case to and joint parts of my project. Shown below are the designs in Autodesk Fusion 360. The top left shows the case that will hold the circuitry, Arduino board and motors. It measures 190 X 85 X 82 mm with 3mm inner thickness. The top right shows the lid of the case left of it. It measures 190 X 85 X 3mm. The bottom shows the piece that there will be multiple of; the is piece will be used to hold the motors and the lid in place. The largest sides both measure at 10mm across.



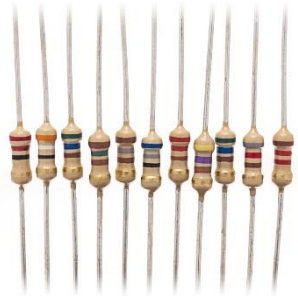
- Transistor - Transistors are semi-conductors that are commonly used to either amplify or switch electrical power. They have three terminals, the Collector, the Base and the Emitter. Usually a transistor requires around 0.7v at the Base for the electrical power at the Collector can be transferred through it and be emitted from the Emitter. There are two transistors in this system, one for each motor.



- Diode - Diodes are electronic components that are used to allow current to flow in one direction while blocking it from going in the reverse direction. They have low resistance in one direction and high resistance in the opposite. Diodes have two pins, the Anode and the Cathode; the Cathode is indicated on the component with a silver edge (shown below). There are two diodes in this system, one for each motor.



- Three resistors - A resistor is a passive two terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages. Their use in this project will be to reduce the current flow to the DC motors. One of the resistors measures at 220Ω and is used in one motor while the other two both measure at 120Ω and are implemented in series for the other motor.



Code

The first thing to do is to include the Myo controller library so all myoduino commands are recognised and understood by the system.

```
#include <MyoController.h>
```

next is to give names to values we will be using throughout the program and assigning pins for them. In this case we are using the `'#define'` command to create the values that will react when certain gestures are picked up by the Myo Band. For example, when a fist is detected being made, the `'FIST_PIN'` will be used. `'FINGERSSPREAD_PIN'` has two due to it affecting both motors, as such they have been given `'L'` and `'R'` at the end of their names to indicated left and right.

Pin 4 is connected to both motors as it effects both motors as the `'fist'` pose will be used as a brak for the RC car.

Pins 5 and 7 will be connected to the left motor while pins 6 and 8 will be connected to the right. This is because the `'waveIn'` pose will be to make the RC car go left and the `'waveOut'` pose will make it go right. The `'fingersSpread'` pose will make the RC car go forward, as such in must be connected to both motors.

```
#define WAVEIN_PIN 5
```

```
#define WAVEOUT_PIN 6
```

```
#define FINGERSSPREAD_PINL 7
```

```
#define FINGERSSPREAD_PINR 8
```

Below is a command being pulled from the `'myocontroller'` libraries CPP file and changed while in this arduino code. what it is doing is instead of writing `'Myocontroller'`, `'myo'` can be used instead. This use is displayed multiple times below.

```
MyoController myo = MyoController();
```

```
void setup() {
```

Below, the pin modes of all the defined variables at the top of the code are being set. In this, the pins are all being set to outputs, as there is no need of inputs with them.

```
pinMode(WAVEIN_PIN, OUTPUT);
```

```
pinMode(WAVEOUT_PIN, OUTPUT);
```

```
pinMode(FINGERSSPREAD_PINL, OUTPUT);
```

```
pinMode(FINGERSSPREAD_PINR, OUTPUT);
```

Calling for the `'initmyo'` command from the `'myocontroller'` libraries CPP file using the previously defined `'myo'` to define where it

should be coming from. This command means to start calling for information being sent from the Myo Band.

```
myo.initMyo();  
}
```

```
void loop()
```

```
{
```

Prints 'HI' to the screen of the computer.

```
  //Serial.println("HI");
```

Calling for the 'updatepose' command from the 'myocontroller' library. This command looks at what is being sent from the Myo Band to 'initmyo' and puts each identifiable pose into a string.

```
  myo.updatePose();
```

The switch below uses what is placed within the string variable from 'updatemyo' and changes to which case below it matches with it.

```
  switch ( myo.getCurrentPose() ) {
```

The 'rest' pose is called whenever the other poses aren't detected, as such all the variables outputs go 'LOW' as no action should be taken during this pose as there is no purposeful input being done.

```
    case rest:
```

```
      digitalWrite(WAVEIN_PIN, LOW);
```

```
      digitalWrite(WAVEOUT_PIN, LOW);
```

```
      digitalWrite(FINGERSSPREAD_PINL, LOW);
```

```
      digitalWrite(FINGERSSPREAD_PINR, LOW);
```

```
      break;
```

The 'fist' pose was chosen to act as the breaks of the RC car, as such all the variables outputs go low to make sure the motors don't activate/ stay active.

```
    case fist:
```

```
      digitalWrite(WAVEIN_PIN, LOW);
```

```
      digitalWrite(WAVEOUT_PIN, LOW);
```

```
      digitalWrite(FINGERSSPREAD_PINL, LOW);
```

```
      digitalWrite(FINGERSSPREAD_PINR, LOW);
```

```
      break;
```

As it is more common for the Myo Armband to be worn on the right arm, The 'waveIn' pose was chosen to make the RC car go left, as

waving in from the right arm points left. To get this effect, the only variable to go 'HIGH' is 'waveIn'.

```
case waveIn:

    digitalWrite(WAVEIN_PIN,HIGH);

    break;
```

As it is more common for the Myo Armband to be worn on the right arm, The 'waveOut' pose was chosen to make the RC car go right, as waving out from the right arm points right. To get this effect, the only variable to go 'HIGH' is 'waveOut'.

```
case waveOut:

    digitalWrite(WAVEOUT_PIN,HIGH);

    break;
```

The pose 'fingersSpread' was chosen to make the RC car go forward, as the only two poses built into the 'myocontroller' library left at this point is 'fingersSpread' and 'doubleTap' and double tapping makes less sense since the spreading your fingers to make something go forward. Also spreading your fingers is far easier to do consistently than double tapping, as is required by this code and double tapping isn't picked up by the Myo Band as easily as 'fingersSpread'. Since we want the RC car to go forward, that means both motors must be going simultaneously, therefore both L and R 'fingerSpread' pins go 'HIGH'.

```
case fingersSpread:

    digitalWrite(FINGERSSPREAD_PINL,HIGH);

    digitalWrite(FINGERSSPREAD_PINR,HIGH);

    break;
```

The 'doubleTap' is used to make the RC car turn around, as the motors can't go in reverse in this circuit. The original plan was to have one motor spin in one direction while the other goes in the reverse direction, however it was discovered that to make a motor spin one way and another in a different way required a special chip that was not accessible at this time.

This code causes the 'waveout' pin to go high when a doubletap between fingers and thumb is detected for 2 second and then stop.

This part of the code, as such, isn't strictly necessary as holding either the 'waveIn' or 'waveOut' for the right amount of time will cause the same effect, however I chose to add this as in the future if the ability to make the motors go in reverse could make this section of code simpler.

```
case doubleTap:

    digitalWrite(WAVEOUT_PIN,HIGH);
```

```
    delay(2000);  
    digitalWrite(WAVEOUT_PIN, LOW);  
    break;  
}
```

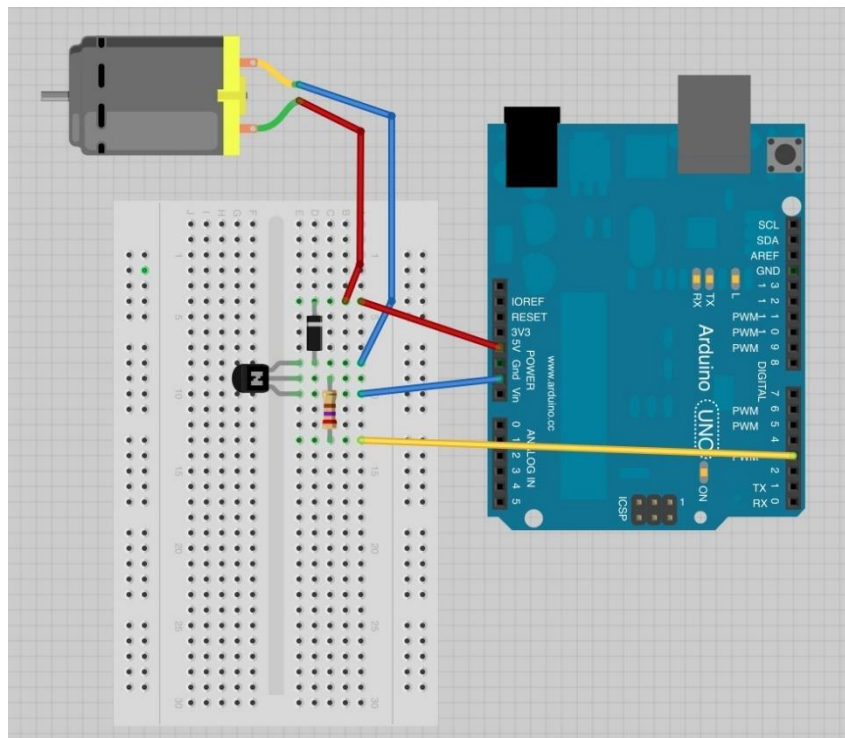
Causes a 100-millisecond delay between pose commands being switched around.

```
    delay(100);  
}
```

Building

The building of the circuit was done very quickly due to the fact that there was very little to build. The reason for this is because most of the circuit's capabilities is done through the Arduino code.

The first thing that was done was to set up the two DC motor systems individually, to check if their design worked before connecting up the Arduino / Myo systems. Following the design below for building a simple motor Arduino system, it was built and tested and worked. Then I added two on the same board as this project requires two and that also worked. The original design for the motors I found called for a 260 Ω resistor to be used, however there were none available, so I substituted with a 220 Ω resistor for one motor and two 120 Ω resistor for the other motor.



The next thing to do was to install the Myo Connect software along with downloading the Myo controller library. The Myo Connect software is required to connect a Myo Band to a computer and the library is necessary for it to interact with Arduino.

Next was to set up all of the connections between the Arduino board and the breadboard motors. The first connections were easily done, connecting the 'WAVEIN_PIN' and 'WAVEOUT_PIN' each to a motor. The harder part was the connection that required both the motors to be connected. Initially the code only had 'FINGERSSPREAD_PIN', no left or right; when trying to connect this pin to both motors and testing the connection to the Myo Band, it either didn't work at all, or locked one of the motors into a never-ending 'HIGH', unable to stop without disconnecting the Arduino board. As such it was split into left and right, each pin going to its own motor. The same problem

didn't happen with 'FIST_PIN', most likely because its purpose is to make all pins go 'LOW' unlike 'FINGERSPREAD_PIN'.

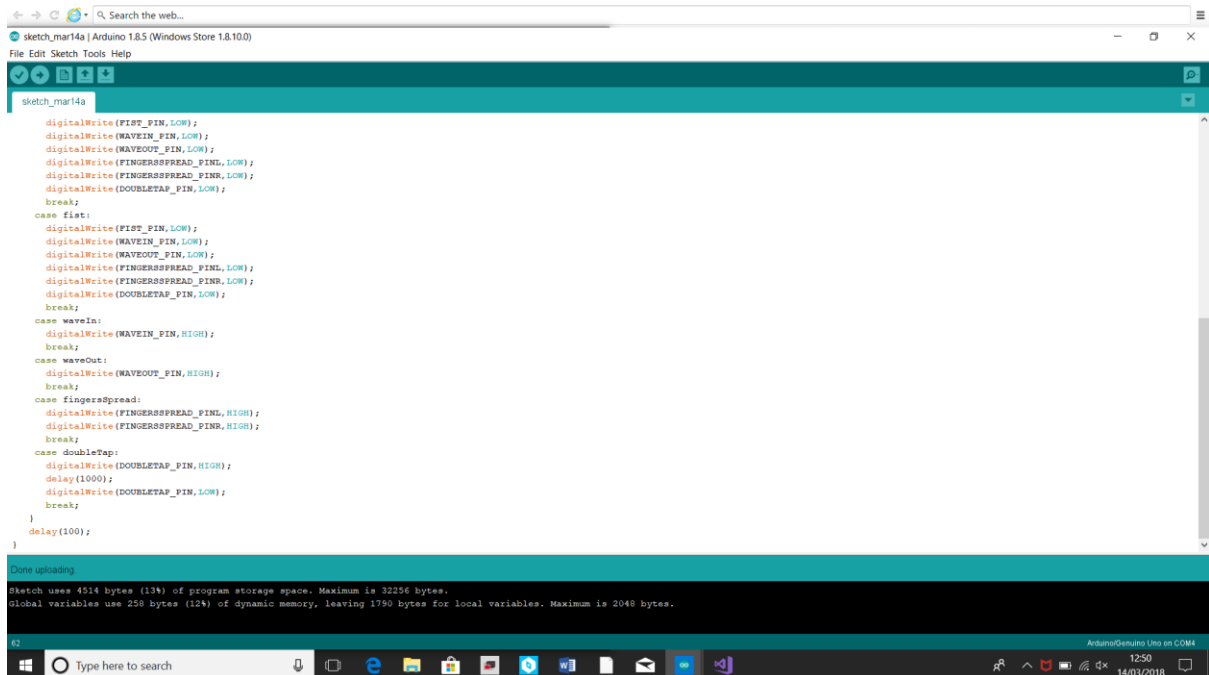
After setting up all of the wiring was the testing the connection of the Myo Band to the computer, to the Arduino board. Setting it up, the problem with 'FINGERSPREAD_PIN' was made apparent and corrected as mentioned above. During the retest, the motor react as they should with the correct gesture. No problems with connecting the Myo Band to the computer was found.

The next step, and honestly the one that turned up the most issues, was to design and 3D print the case. Initially attempted on 3D Studios Max, all the designs mentioned in the equipment's page were done, the converted into STL files (the file type 3D printers take) and sent. However, a problem came up in that the designs shrunk down to less than a centimetre in volume when converted. Recreating the designs in a new program called Autodesk Fusion 360, saving as STL and sending them along seems to have fixed this problem as I was sent a message saying that the 3D printers have accepted them and they would be finished printing within two days. Nearly a week later I had not received any message saying they finished despite asking for one, so I sent a message asking if they had been finished the email I received back said that the printers are no longer accepting the files and I would have to go somewhere else to get them printed.

After using the website 3DHubs to find local 3D printers in the area my product was ordered and came 2 days later. Below shows an image of the circuit boards size being checked to make sure it fits into the case along with an image of the case top alongside the case (empty).



compilation

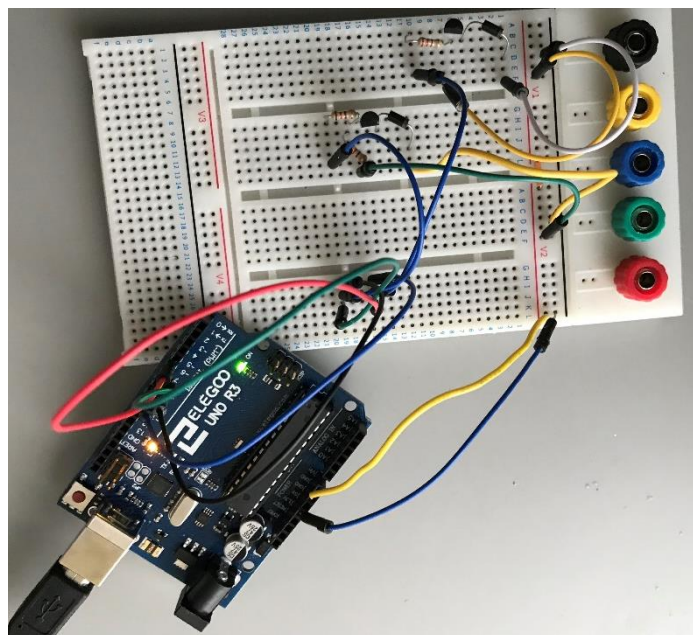


```
sketch_mar14a

digitalWrite(FIST_PIN, LOW);
digitalWrite(WAVEIN_PIN, LOW);
digitalWrite(WAVEOUT_PIN, LOW);
digitalWrite(FINGERSSPREAD_PINL, LOW);
digitalWrite(FINGERSSPREAD_PINR, LOW);
digitalWrite(DOUBLETAP_PIN, LOW);
break;
case fist:
digitalWrite(FIST_PIN, LOW);
digitalWrite(WAVEIN_PIN, LOW);
digitalWrite(WAVEOUT_PIN, LOW);
digitalWrite(FINGERSSPREAD_PINL, LOW);
digitalWrite(FINGERSSPREAD_PINR, LOW);
digitalWrite(DOUBLETAP_PIN, LOW);
break;
case waveIn:
digitalWrite(WAVEIN_PIN, HIGH);
break;
case waveOut:
digitalWrite(WAVEOUT_PIN, HIGH);
break;
case fingersSpread:
digitalWrite(FINGERSSPREAD_PINL, HIGH);
digitalWrite(FINGERSSPREAD_PINR, HIGH);
break;
case doubleTap:
digitalWrite(DOUBLETAP_PIN, HIGH);
delay(1000);
digitalWrite(DOUBLETAP_PIN, LOW);
break;
}
delay(100);
}
```

Done uploading.
Sketch uses 4514 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 258 bytes (12%) of dynamic memory, leaving 1750 bytes for local variables. Maximum is 2048 bytes.

Here is where the Arduino code is both written and tested for any potential errors. If there are any errors when compiling the code, the program will alert the user with what it believes is the problem and stop compiling. The figure above presents the full compilation of the Arduino code for the RC car which was compiled successfully. Below shows the code being uploaded to the completed circuit successfully.



Conclusion

There were a few problems encountered during the building of this project, the main one being explained above involving 3D printing.

The second biggest problem was that it became difficult to work with the Myo Band in university due to admin access blocking the download of necessary software in university computers, this was dealt with by just bringing my laptop to university to work on my project as it had all the necessary software already installed.

All goals of this project were achieved, despite the problems encountered. The code was constructed and uploaded to the Arduino mega. The circuit was built using a breadboard and connected to the Arduino mega through the use of jumper wires.



References

Simon Monk. (17/12/12). *Arduino Lesson 13. DC Motors*. Available: <https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors/parts>. Last accessed 10/03/18.

Jake Chapeskie. (2013). *MyoDuino*. Available: <https://market.myo.com/app/54bd7403e4b00db53ad527a2/myoduino->. Last accessed 10/03/18.