Electro-Mechanical Build March 21st - 27th

In contrast to the construction of the rover main body, the integration of the electromechanical systems has many issues and requires a major redesign of the scoop mechanism and general rover layout.

These stem almost entirely from trying to save on costs by purchasing a non-branded cheap linear actuator. The actuator sourced from (pictured below) is approximately five times cheaper than a brand name item. The main Issue is that it had no data sheet and the vendor could not obtain one. This meant when the CAD model was designed, we assumed actuator dimensions that are about fifty percent too small. The geometry of the scoop operation had to be redesigned with the actual actuator size.

To fit the large actuator, we moved the attachment of the scoop pivot from a metal rod between the scoop arms, to attached directly to the scoop. We also refabricated the floor and back of the scoop in sheet metal because it was flexing by 20mm under the load from the actuator. We also changed the diameter of the actuator pivot from 6mm diameter stainless steel rod to 8mm to eliminate some bending that was occurring during testing.

We also needed to remove the front brace and move the intended location of the Arduino because both was in the way of the actuator.

A few of these issues are also caused by the differences between the CAD models and real-world application. CAD models (or at least the simple model we used) have no friction, perfectly stiff materials, and zero efficiency loses.

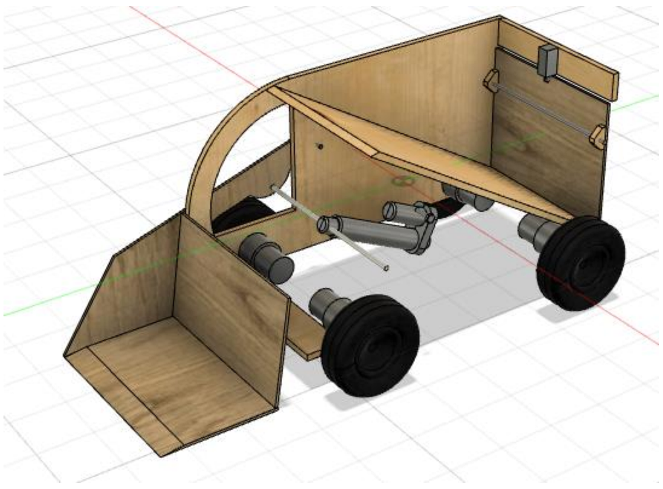
Our main mistake was to try and save money on such a critical part of the system. In particular, the lack of a datasheet meant a lot of the geometry had to be reconfigured when the main body was already built. It would have been better to speed the extra money on the actuator and recoup it on less critical parts.

On a positive note, the motors which we had datasheets for fitted perfectly into the design.

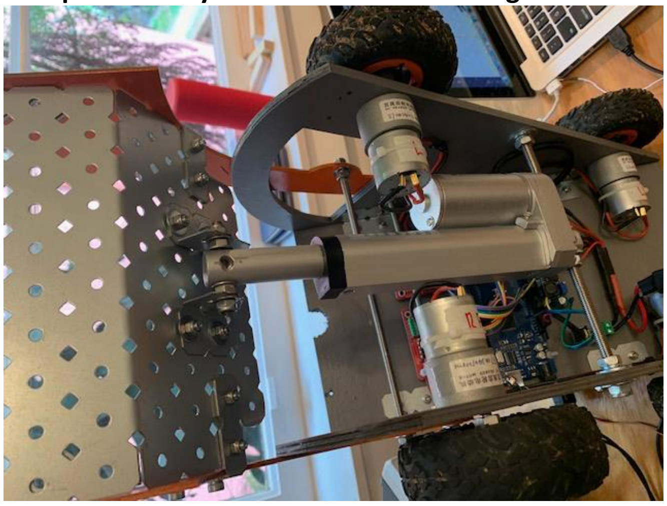
Apart from the extra weight of the actuator, the reconfiguration of the design brought some benefits. The parts count was lowered and attaching the actuator directly to the scoop made the overall structure much stiffer.

Another issue with the linear actuator is that it has no feedback mechanism to tell the Arduino what position the scoop is in. We wanted to use a rotary encoder as one of the scoop arm pivot shafts to provide this feedback. Unfortunately, the components were out of stock and will not arrive in time to be included. We intend to control the position of the scoop by a time reference in the Arduino code. It remains to be seen how successful this will be.

Initial CAD design of scoop and actuator assembly



Scoop as actually constructed after testing



Vendor photo of actuator. There is no size reference.

Electronic System 28th March to 1st April

07 April 2020
14:05

Twenty years ago, the best information was found in books and peer reviewed journals. This is still true. In researching the electronic system, we relied on two excellent resources.

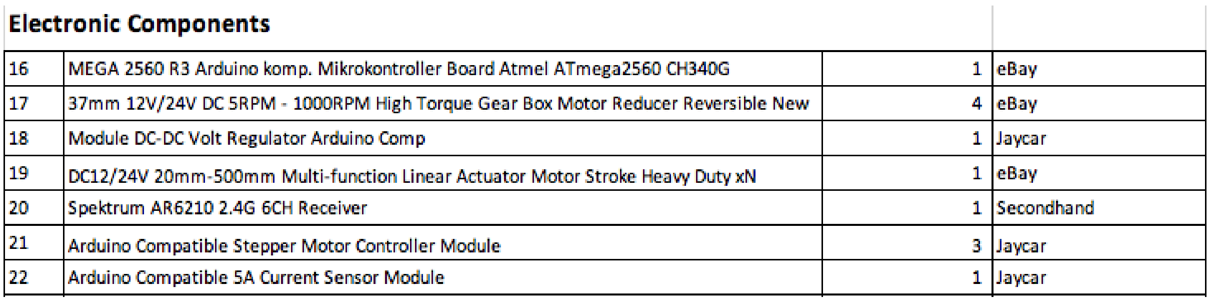
Monk, S, & Scherz, P, 2016, Practical Electronics for Inventors, Fourth Edition, McGraw-Hill Education for general electronics information and layout ideas.

Warren, J, Adams, J, & Molle, H, 2011, Arduino Robotics, Apress for Arduino specific research and code development.

We had a few options when decided how to implement the electronic control system. As per our design principle of simplicity we had chosen an Arduino controlling four 24 volt motors and a linear actuator via three motor controllers. A 24 V solenoid was initially specified to secure and release the rear gate. Unfortunately, the eBay vendor cancelled the order so we replaced it with two servo's we had available.

Due to the COVID-19 related problems shipping electronics from china we had to source many of the electronic components from within Australia leading to budget overruns.

Purchasing list of main electrical components



Electronic Components

16	MEGA 2560 R3 Arduino komp. Mikrokontroller Board Atmel ATmega2560 CH340G	1	eBay
17	37mm 12V/24V DC 5RPM - 1000RPM High Torque Gear Box Motor Reducer Reversible New	4	eBay
18	Module DC-DC Volt Regulator Arduino Comp	1	Jaycar
19	DC12/24V 20mm-500mm Multi-function Linear Actuator Motor Stroke Heavy Duty xN	1	eBay
20	Spektrum AR6210 2.4G 6CH Receiver	1	Secondhand
21	Arduino Compatible Stepper Motor Controller Module	3	Jaycar
22	Arduino Compatible 5A Current Sensor Module	1	Jaycar

For the electronics layout we considered two options:

1. Pre-constructed Arduino compatible modules wired together.
   Pros – simplest option, quick to implement, easy to test and you know it's (probably) going to work.
   Cons – takes up lots of space and requires more wiring, parts of some modules are not used.

2. Design a separate PCB with Arduino, DC power stepdown, motor controllers on one board.
   Pros – compact and uses less power, only has the components needed, you know exactly how it works.
   Cons – much more work (I.e more time) to implement (designing, fabricating & testing PCB)

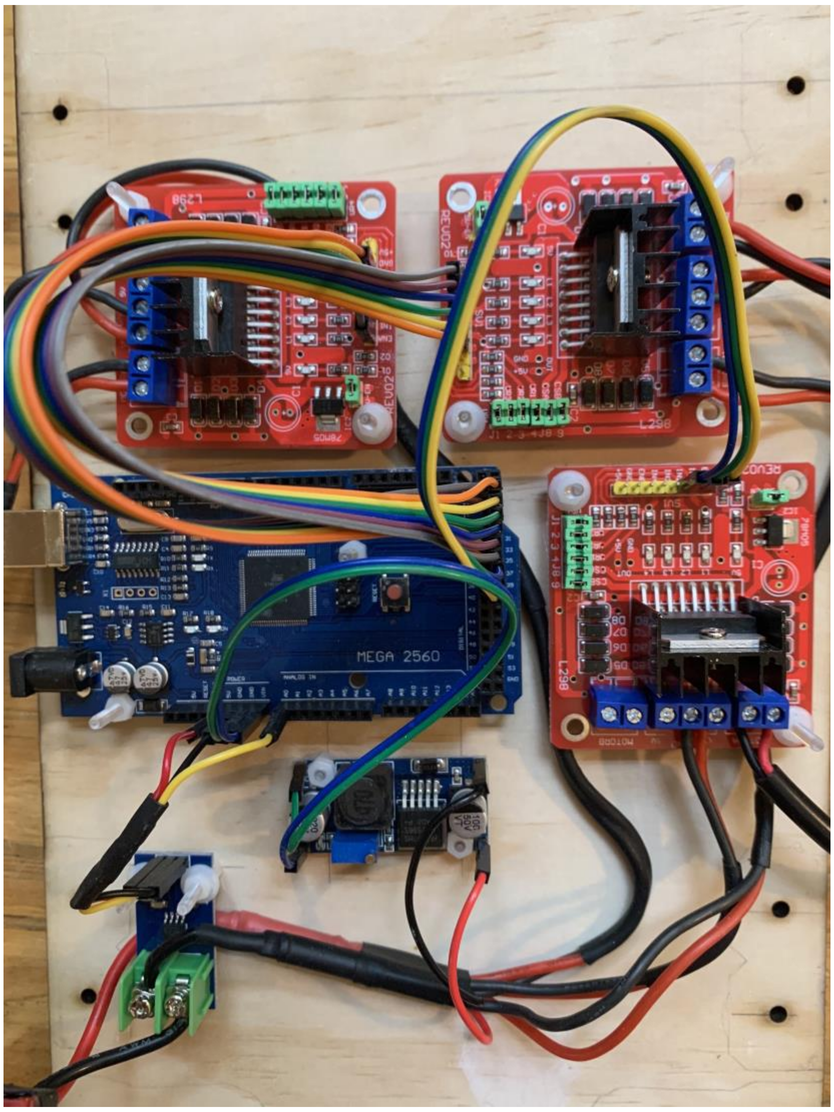
We chose option one due to project time constraints and lack of access to PCB manufacturing facilities. If we continue to develop the design in the future we would like to design a custom PCB to make more space available for other sensors.

The system has seven components.

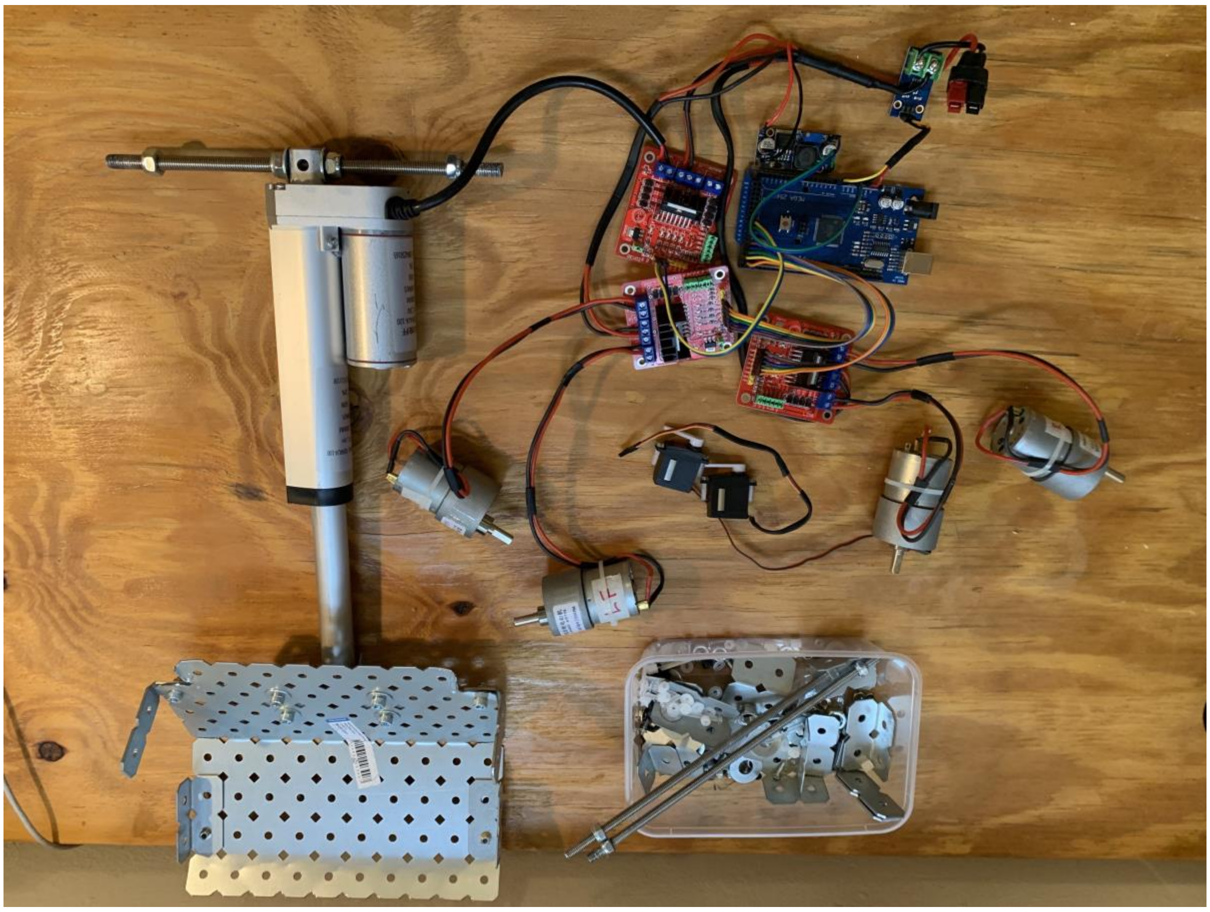
1. A receiver to receive (obviously) signals from the transmitter.
2. Arduino Mega microprocessor to process inputs and send outputs.
3. Three L298N Dual Motor Drivers – Left and right motors have one each and one to drive the linear actuator. The actuator driver has one redundant input/output.
4. A DC –DC convertor to provide 12V to the Arduino from the battery.
5. An ACS712 Hall-Effect Current Sensor to measure the system current draw.

The operation of Arduino, DC MOSFET motor controllers and PWM signals is well documented in many resources so we will not re-explain it here. A detailed explanation can be found in Monk et al. (2016) in chapters 13 and 15.

Electronics Layout (receiver not pictured)



Electro-mechanical System removed from main body



Most of the work on the electronic system involved constructing the wiring loom. A lot of time was spent making sure it would be reliable and not compromised by vibration or allow shorts to occur. This was done by heat shrink insulation around vulnerable areas and using nylon fastenings to secure the PCBs to the board.

Testing and Final Assembly April 2nd - April 8th

Once the electronic system was working and integrated with the electro-mechanical system, some testing was done on the regolith collection and dispatch system.

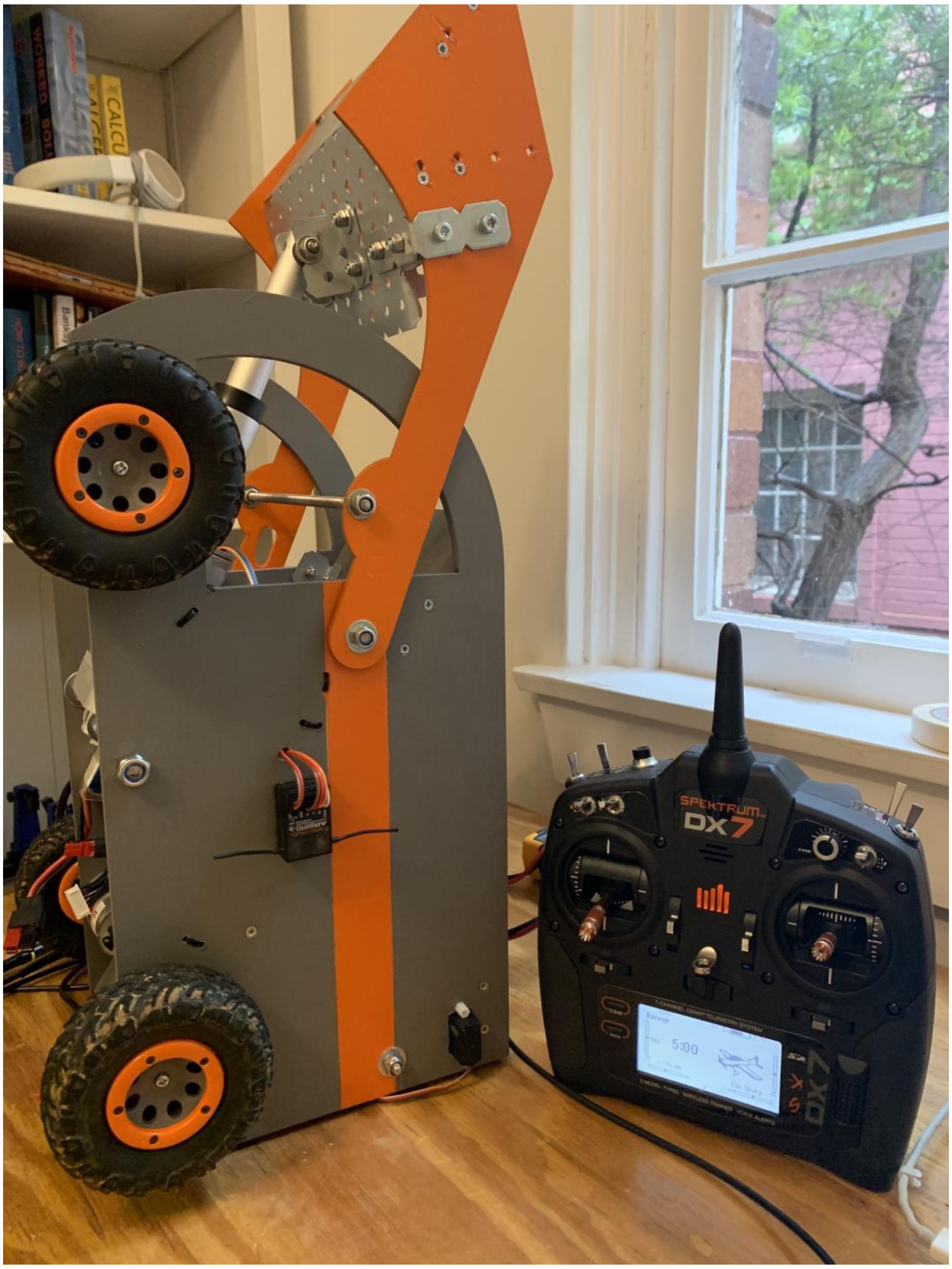
The regolith provided is angular and has a lot of flat surfaces. It was a high angle of repose and does not slide easily. This is difficult for gravity fed systems to handle and we had to adjust the angle of the scoop back to allow it to slide. To lower the friction co-efficient we constructed a coreflute inner on the scoop and regolith container to allow it to slide more easily. In initial testing this worked reasonably well. It is hoped that the vibration and inertia from the moving rover should facilitate this process.

When we were satisfied that the systems were working we disassembled and painted the rover main body components. This was done to increase the aesthetic appeal of the rover as it will be assessed by video. We chose an orange and grey colour scheme as a reference to 'The Martian'. Final assembly was carried out using Loctite on all nuts and bolts (except Nylocs) to reduce the chance of vibration induced failures.

Colour scheme Inspiration

Painting components

99% Complete



Code Development April 9th - ongoing

The Arduino code development was done in two main steps.

First, the motor controllers were tested with a simple motor demo code that did not have any real time input. This allowed us to test the controller and Arduino were working correctly and become familiar with the outputs required.

One issue with the linear actuator motor controller was identified. The controller had power and the Arduino was switching the inputs correctly, however the controller was not supplying power to the motor output. After a few hours of testing everything independently we discovered the cable to the enable input of the controller had an intermittent fault that tested fine, but when put back in the system was not sending the PWM signal to the controller. This did not show up on the controller as the switching LED's do not required the enable signal to illuminate. We should have used the scope to test this signal instead of relying on the LED's.

Demo Code Below
****

```
// connect motor controller pins to Arduino digital pins
// Linear Actuator - A

int LIN_enB = 46;
int LIN_in3 = 50;
int LIN_in4 = 48;

// Left front - Motor - A
int LF_enA = 2;
int LF_in1 = 3;
int LF_in2 = 4;

// Left Back- Motor B
int LB_in3 = 5;
int LB_in4 = 6;
int LB_enB = 7;

// Right Front- Motor B
int RF_enB = 8;
int RF_in4 = 9;
int RF_in3 = 10;

// Right Back - Motor A
int RB_in2 = 11;
int RB_in1 = 12;
int RB_enA = 13;
```

```
void setup()
{
  // set all the motor control pins to outputs
  pinMode(LIN_enB, OUTPUT);
  pinMode(LIN_in3, OUTPUT);
  pinMode(LIN_in4, OUTPUT);

  pinMode(RB_enA, OUTPUT);
  pinMode(RB_in1, OUTPUT);
  pinMode(RB_in2, OUTPUT);

  pinMode(RF_in3, OUTPUT);
  pinMode(RF_in4, OUTPUT);
  pinMode(RF_enB, OUTPUT);

  pinMode(LF_enA, OUTPUT);
  pinMode(LF_in1, OUTPUT);
  pinMode(LF_in2, OUTPUT);

  pinMode(LB_in3, OUTPUT);
  pinMode(LB_in4, OUTPUT);
  pinMode(LB_enB, OUTPUT);
}

void demoOne()
{
  // this function will run the motors in both directions at a fixed speed
  // turn on Linear Actuator for 6 seconds
  digitalWrite(LIN_in3, LOW);
  digitalWrite(LIN_in4, HIGH);
  // set speed to 200 out of possible range 0~255
  analogWrite(LIN_enB, 255);

  delay(6000);
// Reverse linear actuator for 6 seconds
  digitalWrite(LIN_in3, HIGH);
  digitalWrite(LIN_in4, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(LIN_enB, 255);

  delay(6000);



// Set Motors to forward for 6 seconds
//Right Back
  digitalWrite(RB_in1, LOW);
  digitalWrite(RB_in2, HIGH);
  // set speed to 200 out of possible range 0~255
  analogWrite(RB_enA, 255);

//Right Front
  digitalWrite(RF_in3, HIGH);
  digitalWrite(RF_in4, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(RF_enB, 255);

//Left Front
  digitalWrite(LF_in1, LOW);
  digitalWrite(LF_in2, HIGH);
  // set speed to 200 out of possible range 0~255
  analogWrite(LF_enA, 255);

//Left Back
  digitalWrite(LB_in3, HIGH);
  digitalWrite(LB_in4, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(LB_enB, 255);

  delay(6000);

// Set Motors backwards for 6 seconds
//Right Back
  digitalWrite(RB_in1, HIGH);
  digitalWrite(RB_in2, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(RB_enA, 255);

//Right Front
  digitalWrite(RF_in3, LOW);
  digitalWrite(RF_in4, HIGH);
  // set speed to 200 out of possible range 0~255
  analogWrite(RF_enB, 255);

//Left Front
  digitalWrite(LF_in1, HIGH);
  digitalWrite(LF_in2, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(LF_enA, 255);

//Left Back
  digitalWrite(LB_in3, LOW);
  digitalWrite(LB_in4, HIGH);
  // set speed to 200 out of possible range 0~255
  analogWrite(LB_enB, 255);

  delay(6000);

}

void loop()
{
  demoOne();
  delay(1000);
}
```

Video of [Motor demo test]

The second part of the coding process is developing the working code to allow real time RC control of the rover.

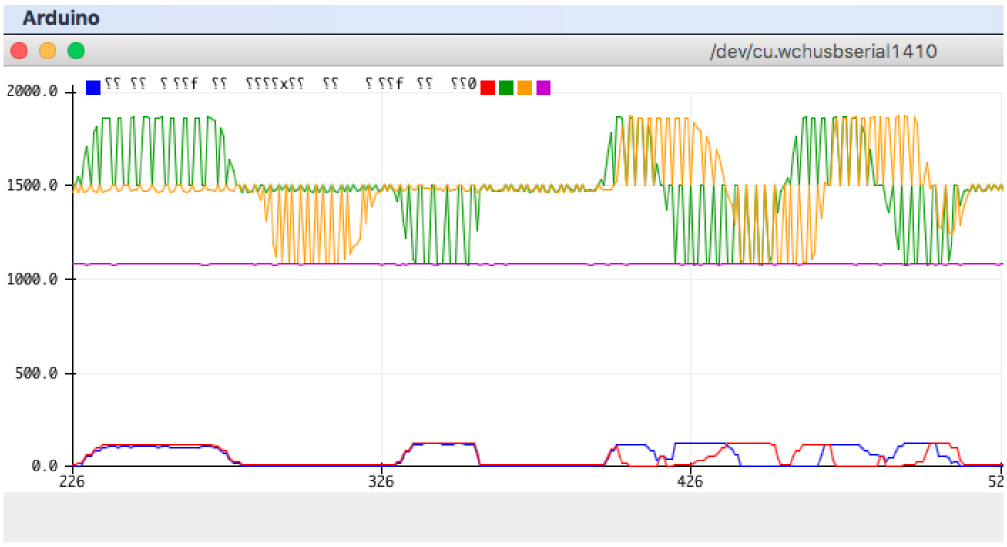
This process is quite complex for novice coders and is currently in development.

Below is a video of oscilloscope analysis of the PWM input signals from the receiver.

Video of [Analysis of reciever signal]

We can use the serial plotter to analysis the signal and calibrate the inputs to the Arduino.

The green is forward/reverse and the yellow is steering. The overlay is where the channels are mixed.


Arduino
/dev/cu.wchusbserial1410