

**California State University, Long Beach**

**EE 444 – Microprocessor Based System Design**

**Lab 2 – Shaft encoders, rover, Matlab and fun!**



**Student: Walter Heth, Paul Zelaya**

**Professor: Gary Hill**

**2/27/2013**

## Table of Contents

- 1 Overview/Objective (Walter)
- 2 Reference Material (Paul)
- 3 Mission Checklist (Paul)
- 4 How it Works (Walter)
- 5 Instructions
  - 5.1 Software Downloads (Paul)
  - 5.2 Shaft Encoder Demo (Paul)
  - 5.3 Installing shaft encoders on Rover (Walter)
  - 5.4 Traveling a set distance (Walter)
  - 5.5 Using Matlab (Paul)
- 6 Make the Rover Go Straight (Walter)
- 8 Code (Walter)
  - 8.1 Calibration Code
  - 8.2 Go1Foot
  - 8.3 Matlab Script

## 1 Overview/Objective

The purpose of incorporating a shaft encoder is to have an accurate way of measuring how far the rover has traveled. In previous semesters the rovers were told to go forward for a certain amount of time, now we are able to tell it to go a certain number of inches.

## 2 Reference Material

1. Arduino IDE for Arduino UNO - [Link](#)
2. Adafruit Motor Shield library - [Link](#)
3. Alps EC11B15202AA rotary encoder cut sheet - [Link](#)

## 3 Mission Checklist

- Connect LED's to the shaft encoder on a breadboard. The LED's should visually show the type of output shaft encoder produces.
- Install the shaft encoder on the rover.
- Use shaft encoder to have rover travel a pre-determined distance. Monitor rover movement on Matlab.

## 4 How it Works

If you've ever seen the inside of a roller ball mouse you may have noticed the optical shaft encoder. The ball of the mouse would roll against the rim of the shaft encoder, causing it to turn. The computer would read how many pulses occurred and the mouse would move on the screen accordingly.

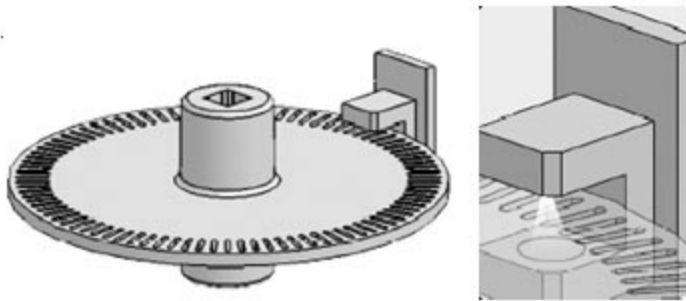


Figure 5.1

Our encoder behaves similarly. The input channels are configured with pull up resistors and by default have a logic of 1 or HIGH. When channel A hits its first “click” it will essentially close a switch between A and C. C is connected to ground causing our input to be read as logic 0 or LOW. The next click for channel A will open the switch and we will have our HIGH value again. The optical encoder only represents one of the channels of our shaft encoder, so why do we have two?

If we only use one channel and rotate the shaft clockwise we would see  
1010101010101....

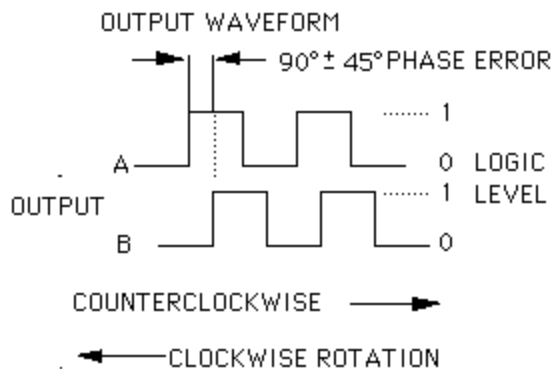
If we use one channel and rotate the shaft counter clockwise we would see  
101010101010....

With two channels we can count from 0 to 3 and tell if we are moving

clockwise: 10,11,01,00,10,11.... (2,3,1,0,2,3....)

counterCW: 01,11,10,00,01,11... (1,3,2,0,1,3...)

However we do not need to use the sensor to find this information as we will be telling the rover weather we are going forwards or backwards. The only use for the shaft encoder is to accurately record how far we have gone. Therefore we will only need one channel



## 5 Instructions

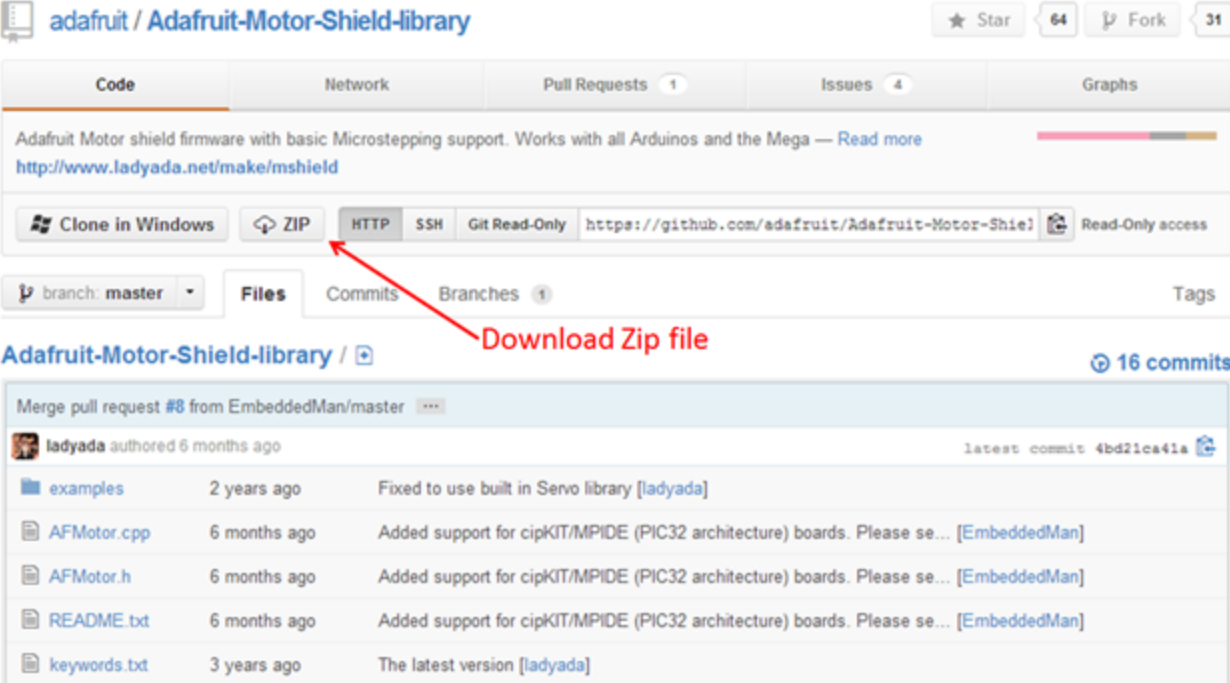
## 5.1 Software downloads

Download the Arduino IDE - [Link](#)

- The IDE can be launched from the downloaded folder, no need to run an installation routine. Make sure to preserve the folder structure.

Download the Adafruit Motor Shield Library - [Link](#)

- Download the zip file as shown below:



adafruit / **Adafruit-Motor-Shield-library** ★ Star 64 🍴 Fork 31

Code Network Pull Requests 1 Issues 4 Graphs

Adafruit Motor shield firmware with basic Microstepping support. Works with all Arduinos and the Mega — [Read more](#)  
<http://www.ladyada.net/make/mshield>

Clone in Windows ZIP HTTP SSH Git Read-Only <https://github.com/adafruit/Adafruit-Motor-Shield-library> Read-Only access

branch: master Files Commits Branches 1 Tags

**Adafruit-Motor-Shield-library** / [+](#) 📄 16 commits

Merge pull request #8 from EmbeddedMan/master

File	Time ago	Commit message
examples	2 years ago	Fixed to use built in Servo library [ladyada]
AFMotor.cpp	6 months ago	Added support for cipKIT/MPIDE (PIC32 architecture) boards. Please see... [EmbeddedMan]
AFMotor.h	6 months ago	Added support for cipKIT/MPIDE (PIC32 architecture) boards. Please see... [EmbeddedMan]
README.txt	6 months ago	Added support for cipKIT/MPIDE (PIC32 architecture) boards. Please see... [EmbeddedMan]
keywords.txt	3 years ago	The latest version [ladyada]

- Inside the zip file there will be a header file and a c++ source file. Both these files must be in the same folder as the sketch included at the end of this lab.

## 5.2 Shaft Encoder Demo

The rotary encoder used in this lab is a quadrature encoder. Refer to the encoder documentation. The encoder has a ground pin (pin C) and two signal pins (pins A and B). The pins not only give the number of revolutions the encoder has turned through, but the pins can also determine which way the encoder has turned. LEDs will be used in order to demonstrate this point. Wire a Arduino / breadboard as shown in Figure 1.

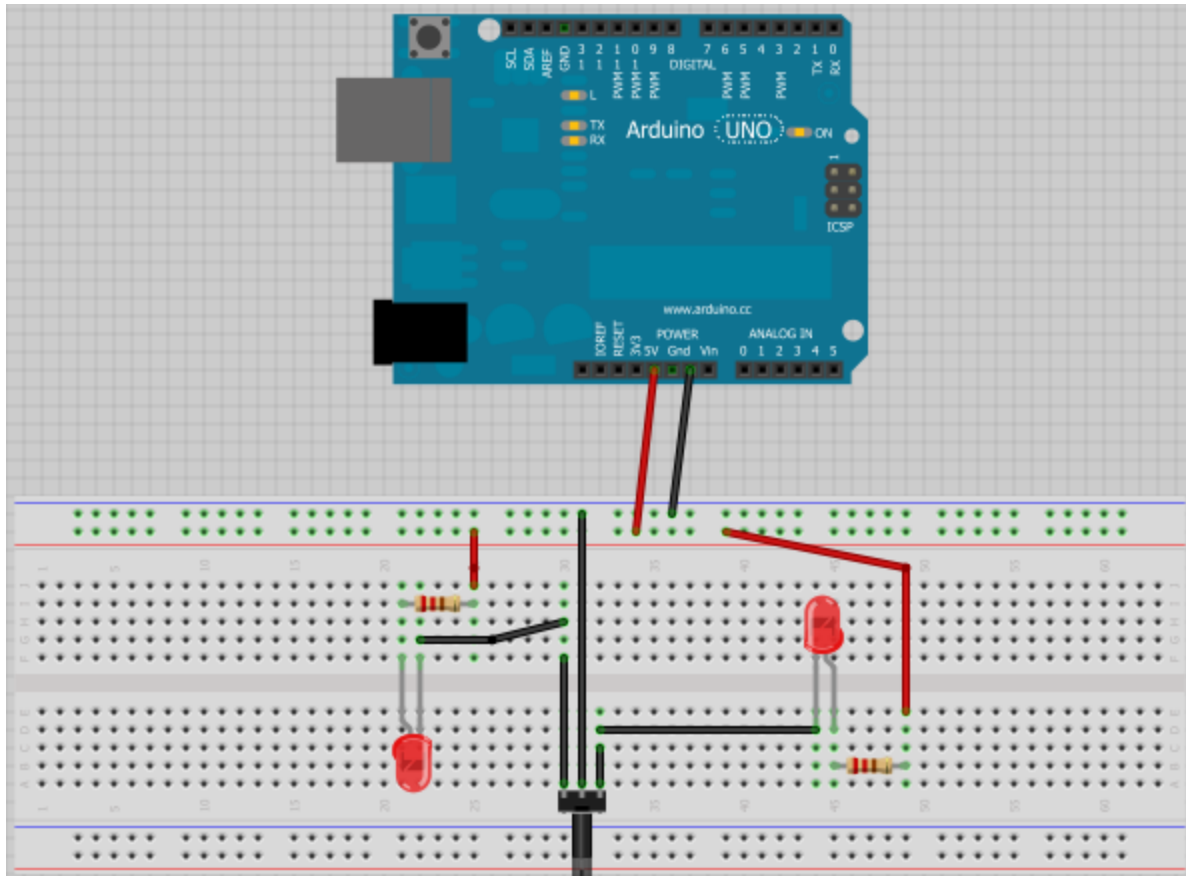


Figure 1. How to wire the shaft encoder to LED's (Image developed using Fritzing)

As the encoder turns, it will establish a ground that will turn the LEDs on or off. When the encoder is turned the LEDs light off in a pattern that is indicative of the direction the encoder is turning.

### 5.3 Installing shaft encoders on Rover

1. Solder 6-8 inch wires on each of the 3 pins or preferably crimp together a female connector that will go over the 3 pins. These pins break off easily, try to minimize any bending. Do not panic if one of the outside pins break off as you will only need one of them. If the middle pin or both the outside pins break off then get a new encoder.
2. Remove track and back wheels from the rover.
3. Remove the original shaft that comes with the chassis kit.
4. Insert shaft encoder in place of the original shaft, make sure pins are pointing towards the arduino. Figure 4.1 has the pins facing the wrong way.
5. Slide the nut over the shaft and tighten it down so that the side with pins can not rotate. The shaft should still be able to rotate. Check Figure 4.1.
6. Re-installing the wheels we can see that they stick out a bit more due to the shaft encoder. You can see this in Figure 4.2. Try flipping the wheels around and installing them the opposite direction. This should put them back in line.
7. Now for the fun part, the wheel needs to turn the shaft encoder. The best method would be to drill a hole in the wheel that would allow a small bolt to come in facing the flat side of the shaft and pinch it in place. It is well advised not to glue the wheel to the shaft encoder. If the pins of your shaft encoder break off you will need to replace it.
8. Run the wires through the large hole visible in Figure 4.2.
9. For the sake of making things simple, connect channel A or B of the shaft encoders to I/O pin 2 and Analog pin 0. Connect both C pins to ground. Check Figure 4.3.



Figure 4.1

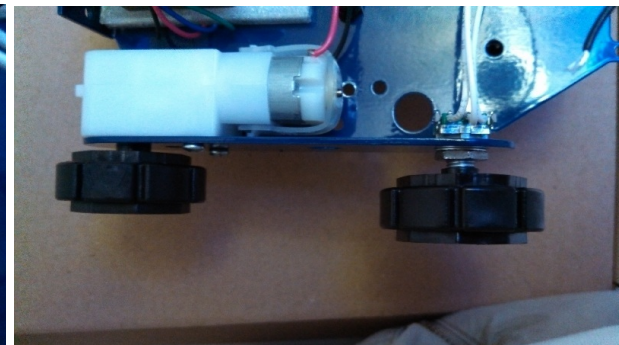


Figure 4.2

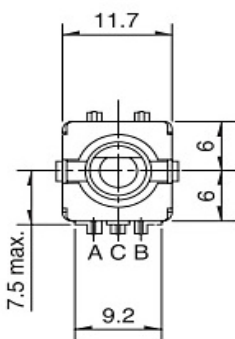


Figure 4.3

## 5.4 Traveling a set distance

Read through the code at the end of this lab named Go1Foot. It has very detailed comments that explain exactly what is going on. For a general understanding, the code tells each motor to go forward until one of the motors reaches a certain number of pulses. Once one of the encoders reaches this number both motors will turn off so that the rover will not turn when it reaches its destination. It is important to make sure the rover has both motors going at the same speed. You can set both motors to 200 and manually increase the speed of the slower motor until they both reach the same speed. You can also check out section 6 on the Calibration code if you feel like having the arduino try to do this for you. Once you have it going straight you should be able to tell the rover the exact distance to travel using the conversion of (80 pulses/1foot). We reached this number by telling the arduino to go 50 clicks which caused the arduino to go 7.5 inches. If we wanted 12 inches that would be  $12\text{in} \cdot (50 \text{ click}/7.5\text{in})=80$  clicks per foot. Try getting your rover to go 5 feet to see how accurate this is.

## 5.5 Using MATLAB to track rover movement

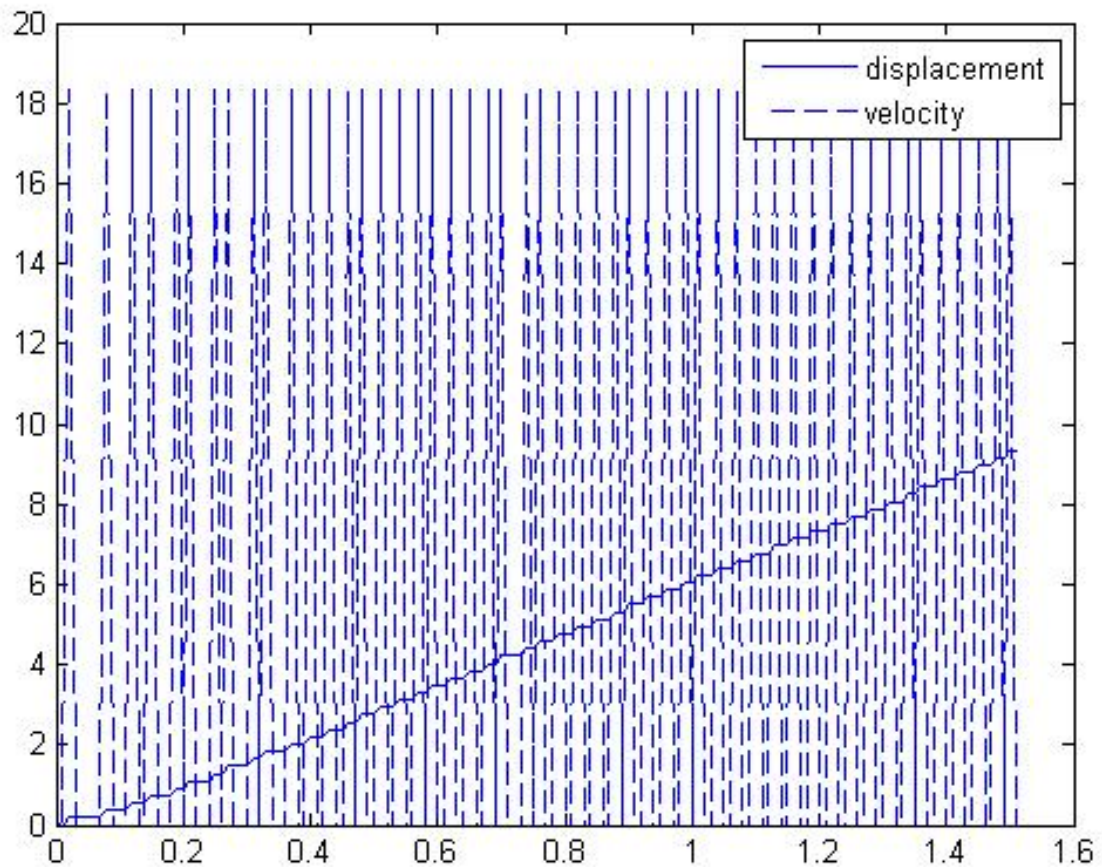
Matlab can be used to graph out rover displacement. The program reads a serial output from the Arduino and displays the data on a plot. In order to start the program, a serial object has to be created in Matlab. This object contains all the parameters used to establish communication.

The rover uses one encoder channel to determine movement. The encoder has 30 transitions per revolution. As discussed before, the encoder is the center of the rover rear wheel. Rover displacement per transition can be given by the following expression:

$$\begin{aligned} r &= \text{Distance from wheel center to track contact} \\ s &= \text{Rover displacement} \\ N &= \text{transition or dent} \\ s &= \frac{2\pi r N}{30} \end{aligned}$$

Upload the Matlab Code into the Arduino. This code outputs the number of times the encoder has changed states. This data is then outputted through the serial pins. A matlab script takes this data and manipulates it in order to get distance traveled and velocity. The matlab script can be run at any time. Once rover is moving, run the script. A graph should appear that shows displacement and velocity plotted on the same graph. Refer to the Matlab Script at the end of this report.





*Figure 5.5*

## 6 Make the Rover Go Straight

There is a high possibility that your rover will not go straight. You can try to guess the two motor speeds that will get your rover to go straight. You can also try the calibration code given at the end of this lab. It tells both motors to go forward and counts 200 steps or clicks on each shaft encoder. If one motor reaches the 200 steps first it will speed up the other motor and run through the calibration again. There is a small error margin that is acceptable, and if it falls into that margin 5 times in a row it will give you the new motor speed values that you should use in your code. Try it out, start at motor speed 200 and when you get your values program your rover to go forward next to a wall and do some fine tuning.

## Calibration Code

```
#include <AFMotor.h>

AF_DCMotor motor1(1, MOTOR12_64KHZ); // create motor #1, 64KHz pwm
AF_DCMotor motor2(2, MOTOR12_64KHZ); // create motor #2, 64KHz pwm
const int pin2 = 2;
const int pinA0 = A0;
int turnEncoder1 = 0;
int turnEncoder2 = 0;
boolean lastStateA1;
boolean lastStateA2;
int NumbPulse=200;
int M1speed=200; //222,248
int M2speed=200;
int counter= 0;
int x=0;

void setup()
{
  Serial.begin(115200);
  Serial.println("Motor setup");

  pinMode(pin2, INPUT); //Define this pin as a input
  digitalWrite(pin2, HIGH); //Get this pull up resistor fired up
  pinMode(pinA0, INPUT); //Define this pin as a input
  digitalWrite(pinA0, HIGH); //Get this pull up resistor fired up

  motor1.setSpeed(M1speed); // set the speed to 200/255
  motor2.setSpeed(M2speed);
}

void loop()
{
  turnEncoder1=0;
  turnEncoder2=0;

  motor1.setSpeed(M1speed); // set new speed
  motor2.setSpeed(M2speed);

  motor1.run(FORWARD); // turn it on going forward
  motor2.run(FORWARD);

  while(turnEncoder1<NumbPulse || turnEncoder2<NumbPulse)
  {
    boolean stateA1 = digitalRead(pin2);
    boolean stateA2 = digitalRead(pinA0);

    if((stateA1==lastStateA1) && (stateA2==lastStateA2))
    {
      Serial.println("No Change");
    }
  }
}
```

```

    if (stateA1 != lastStateA1)
    {
        lastStateA1 = stateA1;
        turnEncoder1++;
    }
    if(stateA2 != lastStateA2) //((stateA2 == HIGH && lastStateA2 == LOW) || (stateA2 ==
LOW && lastStateA2 == HIGH))
    {
        lastStateA2 = stateA2;
        turnEncoder2++;
    }

    Serial.print("Left:");
    Serial.print(turnEncoder1);
    Serial.print("steps, Right:");
    Serial.print(turnEncoder2);
    Serial.println("steps");
}

motor1.run(RELEASE); // stopped
motor2.run(RELEASE);
Serial.println("Both motors stopped");
delay(1000);

x=0;

if (turnEncoder1>turnEncoder2)
{
    x=turnEncoder1-turnEncoder2;
    if (x>2 && x<5)
    {
        M2speed++;
    }
    if (x>4)
    {
        M2speed += 2;
    }
}
if (turnEncoder2>turnEncoder1)
{
    x=turnEncoder2-turnEncoder1;
    if (x>2 && x<5)
    {
        M1speed++;
    }
    if (x>4)
    {
        M1speed += 2;
    }
}
if (x<3)

```

```
{
  counter++;

  if (counter== 5)
  {
    Serial.println("*****");
    Serial.print("set M1 speed to ");
    Serial.println(M1speed);
    Serial.print("set M2 speed to ");
    Serial.println(M2speed);
    Serial.println("*****");
    delay(20000);
  }
}
else
{
  counter=0;
}
Serial.print("counter=");
Serial.println(counter);
Serial.print("M1speed=");
Serial.println(M1speed);
Serial.print("M2speed=");
Serial.println(M2speed);
delay(5000);
}
```

## Go1Foot

```
#include <AFMotor.h>

AF_DCMotor motor1(1, MOTOR12_64KHZ); // create motor #1, 64KHz pwm
AF_DCMotor motor2(2, MOTOR12_64KHZ); // create motor #2, 64KHz pwm
const int pin2 = 2; // Reads shaft encoder 1
const int pinA0 = A0; // Reads shaft encoder 2
int turnEncoder1 = 0; // use these variables to keep track how many steps
int turnEncoder2 = 0; // the rover has taken
boolean lastStateA1; // Channel A of shaft encoder 1
boolean lastStateA2; // Channel A of shaft encoder 2
int NumbPulse=80; //12in (1click/0.15in)=80 clicks 2ft=160clicks etc
int x=0;
int y=0;
int z=0;
int M1speed=222; //Left motor 222,239
int M2speed=248; //right motor

void setup()
{
  Serial.begin(115200); // set up Serial library at 115200 bps

  pinMode(pin2, INPUT); //Define this pin as a input
  digitalWrite(pin2, HIGH); //Get this pull up resistor fired up
  pinMode(pinA0, INPUT); //Define this pin as a input
  digitalWrite(pinA0, HIGH); //Get this pull up resistor fired up

  motor1.setSpeed(M1speed); // set the speed to 200/255
  motor2.setSpeed(M2speed);
  Serial.println("Motor setup complete");
}

void loop()
{
  turnEncoder1=0; //Reset counters if making another travel
  turnEncoder2=0;

  motor1.setSpeed(M1speed); // set Motor1 speed (Left)
  motor2.setSpeed(M2speed); // set Motor2 speed (Right)
  x=NumbPulse; // Tell left motor how many steps it needs to take
  y=NumbPulse; // Tell right motor to take the same amount of steps
  z=0;

  motor1.run(FORWARD); // turn it on going forward
  motor2.run(FORWARD);

  // while neither motor has reached their last step, continue to run forward
  // and keep track of what step you are on
  while(turnEncoder1<NumbPulse && turnEncoder2<NumbPulse)
  {
    boolean stateA1 = digitalRead(pin2); //Read shaft encoders and see if it has
```

```

boolean stateA2 = digitalRead(pinA0);    //changed from low to high or vice versa

//If both states are the same then there is no change
if((stateA1==lastStateA1) && (stateA2==lastStateA2))
{
    Serial.println("No Change");
    z++;
}

//if Shaft encoder 1 has changed then decrement x
if (stateA1 != lastStateA1)
{
    lastStateA1 = stateA1;
    turnEncoder1++;
    x--;
}

//if Shaft encoder 2 has changed then decrement y
if(stateA2 != lastStateA2)
{
    lastStateA2 = stateA2;
    turnEncoder2++;
    y--;
}

//While walking, tell us how many steps you have left to take
Serial.print(y);
Serial.print("Left:");
Serial.print(turnEncoder1);
Serial.print("steps, Right:");
Serial.print(turnEncoder2);
Serial.println("steps");
}

// One of the motors have reached the final step
// Stop both motors
motor1.run(RELEASE);    // stopped
motor2.run(RELEASE);
Serial.println("Both motors stopped");

// How many steps does each motor have remaining?
// How many times do we have no change read?
// We want a high # of no change to know that we are checking for
// changes faster than we are changing.
Serial.print(x);
Serial.println(" steps left on the left motor");
Serial.print(y);
Serial.println(" steps left on the right motor");
Serial.print(z);
Serial.println("x No Changes");
delay(5000);
}

```

## Matlab Code

```
#include <AFMotor.h>

AF_DCMotor motor1(1, MOTOR12_64KHZ); // create motor #1, 64KHz pwm
AF_DCMotor motor2(2, MOTOR12_64KHZ); // create motor #2, 64KHz pwm
const int pin2 = 2; // Reads shaft encoder 1
const int pinA0 = A0; // Reads shaft encoder 2
int turnEncoder1 = 0; // use these variables to keep track how many steps
int turnEncoder2 = 0; // the rover has taken
boolean lastStateA1; // Channel A of shaft encoder 1
boolean lastStateA2; // Channel A of shaft encoder 2
int NumbPulse=400; //12in (1click/0.15in)=80 clicks 2ft=160clicks etc
int AvgStep=0;
int M1speed=222; //Left motor 222,239
int M2speed=248; //right motor

void setup()
{
  Serial.begin(9600); // set up Serial library at 9600 bps

  pinMode(pin2, INPUT); //Define this pin as a input
  digitalWrite(pin2, HIGH); //Get this pull up resistor fired up
  pinMode(pinA0, INPUT); //Define this pin as a input
  digitalWrite(pinA0, HIGH); //Get this pull up resistor fired up

  motor1.setSpeed(M1speed); // set the speed to 200/255
  motor2.setSpeed(M2speed);
}

void loop()
{
  turnEncoder1=0; //Reset counters if making another travel
  turnEncoder2=0;
  AvgStep=0;

  motor1.setSpeed(M1speed); // set Motor1 speed (Left)
  motor2.setSpeed(M2speed); // set Motor2 speed (Right)

  motor1.run(FORWARD); // turn it on going forward
  motor2.run(FORWARD);

  // while neither motor has reached their last step, continue to run forward
  // and keep track of what step you are on
  while(turnEncoder1<NumbPulse && turnEncoder2<NumbPulse)
  {
    boolean stateA1 = digitalRead(pin2); //Read shaft encoders and see if it has
    boolean stateA2 = digitalRead(pinA0); //changed from low to high or vice versa

    //if Shaft encoder 1 has changed then decrement x
    if (stateA1 != lastStateA1)
    {
```

```

        lastStateA1 = stateA1;
        turnEncoder1++;
    }

    //if Shaft encoder 2 has changed then decrement y
    if(stateA2 != lastStateA2)
    {
        lastStateA2 = stateA2;
        turnEncoder2++;
    }

    //Find the average step taken between both shafts and send to MATLAB
    AvgStep=((turnEncoder1+turnEncoder2)/2);
    Serial.println(AvgStep, DEC);
    delay(10);
}

// One of the motors have reached the final step
// Stop both motors
motor1.run(RELEASE);    // stopped
motor2.run(RELEASE);
delay(5000);
}

```

## Matlab Script

```

%Clear all the variables and clear command window

clear all
clc

%Define Serial object
s = serial('COM7','BaudRate',9600)
fopen(s) %<----- enable serial communication

interv = 100; %Defines the number of samples we want
pass = 1; %Simple counter
t = 1; %Column vector counter
x = 0; %Array that holds transition data
dis = 0; %Displacement
delay = 50; %Delay defined in Arduino sketch
r = 7/8; %Distance from wheel center to table surface
treal(1,1) = 0;
tstep = delay/1000; %in seconds

while(t<interv)
    b = fscanf(s,'%d'); %Reads what is outputed from Arduino
    if(t==1) %have to perform this if,else loop to ensure treal can be used in plot
        treal(t,1) = 0;
    else
        treal(t,1) = treal(t-1,1)+tstep;
    end
    end
    x(t,1) = b;

```



```
dis(t,1) = (2*pi*r*b)/(30);
if(t>2)
    v(t,1) = (dis(t,1)-dis(t-1,1))/(treal(t,1)-treal(t-1,1));
plot(treal,dis);
hold on
plot(treal,v,'--');
legend('displacement','velocity');
end
%axis([-10,interv,-50,1050]);
t = t+pass;
drawnow
end
fclose(s)
```