

Brandeis University Robotics Club

Engineering Notebook for Zumo Automation Project

Contributors: Jacob Smith, Matthew Millendorf

Advisor: Tim Hebert

Abstract: This project's main goals are to create a user-friendly platform to program the ZUMO32U4 robots in, and to allow for automated testing of sumo-competition strategies

<b>Project Name</b>	<b>Page</b>	<b>Date</b>
Zumo bot setup	1	2/7/18
Getting Motors Running	3-5	2/14/18-2/28/18
Line Follower	6-7	3/7/18
LCD Startup	8	3/8/18
Drive Function Wrapping/Drive class	9-11	3/8/18-3/9/18
Line Class	12-13	3/13/18
Line Navigator Update	14	3/13/18
Proximity Sensor Test	15	3/15/18
Encoder, Timer, and Velocity	16-18	3/15/18
Proximity Detector	19	3/21/18
Accelerometer Setup	20-21	3/21/18-3/28/18
Accelerometer Max Value	22-23	3/28/18
Timer, Drive, Line Class	24	4/4/18
Timer, Drive, Line class documentation	25-27	4/10/18
Accelerometer Class	28-34	4/12/18-4/25/18
Connection Debugging	32-33	4/12/18
Encoder Class	34-35	4/25/18
Improved Github Documentation	36-42	4/26/18
Graphical output	43	4/29/18
Position Calibration Test	44-45	4/29/18
Velocity Calibration Test	46-51	5/1/18

2018.1.31 6:04 PM: Zumo bot

Zumo 32U4 Robot, Installing Driver Software

<<https://www.pololu.com/docs/0J63/5.1>>.

There is a Pololu ComPort

Downloading Arduino Integrated Development Environment

<<https://www.arduino.cc/en/Main/Donate>>.

Adding Additional Board Manager in Arduino Software--> File→ Preferences

Didn't work, installing Board through Tool→ Board Manager

Pololu A Star 32U4

Selecting Com Port 3

2018.3.8 12:14 PM Jacob Smith: Yesterday Matthey and I got the Zumo Bot to print a message to the lcd, and then print the line follower sensor readings to the LCD. This allowed us to compute the above list of line sensor readings on tape versus on carpet, which let us write a program to have the robot drive forward until it sees tape, when it backs up and turns.

```
/* Name:Jacob Smith and Matthew
Millendorf, Brandeis Robotics Club
Date: March 7 2018
Assignment: Line Follower
*/

#include <Zumo32U4LCD.h>
#include<ZUmo32U4Motors.h>
#include <Wire.h>
#include <Zumo32U4LineSensors.h>
Zumo32U4LCD lcd;
Zumo32U4LineSensors sense;
Zumo32U4Motors junior;
/*
Methods to use for LCD
clear() //clears contents
command() //sends an arbitrary
comand to LCD
display() //turns the display on
home() //resets the screen back to
default
init() //initializes the LCD
send() //sends data or commands to
LCD
send(uint8_t data, bool rsValue, bool
only4Bits)
write() ;
*/

//make an array of line sensor readings
#define NUM_SENSORS 5
uint16_t
lineSensorReadings[NUM_SENSORS];

void setup() {
// put your setup code here, to run
once:
    sense.initFiveSensors();
    lcd.print("Junior");
}

void loop() {
    sense.read(lineSensorReadings,true)
    ;
    lcd.clear();
    lcd.gotoXY(0, 1);
    for(int i = 0; i < 4; i+=2){
        lcd.print(lineSensorReadings[
            i]);
    }
    junior.setSpeeds(300, 300);
    if(lineSensorReadings[0] < 300 ||
    lineSensorReadings[2] < 200 ||
    lineSensorReadings[4] <390){
        junior.setSpeeds(-300, -300);
        delay(100);
        junior.setLeftSpeed(300);
        junior.setRightSpeed(-300);
        delay(500);
    }
}
```

2018.3.8 6:55PM Jacob Smith: Today I am working on making the current line follower program more readable and fixing the LCD output, which wasn't working. In order to fix the LCD output, I researched how to get the robot's time. Here is a sample program I wrote that prints the robot's current time to the lcd.

```
/* Name:Jacob Smith
 * Date: March 8 2018
 * Assignment: Time Printer
 * source
 * https://learn.adafruit.com/multi-tasking-the-arduino-part-1/using-millis-for-timing
 */
#include <Wire.h>
#include <Zumo32U4LCD.h>
Zumo32U4LCD lcd;

void setup() {
  lcd.clear();
  lcd.print("CurTime");
  lcd.gotoXY(0,1);
}

void loop() {
  long time=millis();
  lcd.print(time);
  lcd.gotoXY(0,1);
}
```

2018.3.9 5:08 PM Jacob Smith: Today, I used the specific output function of the Arduino integrated development environment to fix the syntax of the Drive class. The most significant change I made to allow it to compile was making the method instance methods instead of static methods. Here is the complete drive file

Drive body file .cpp

```
1  /*Written by Jacob Smith for Brandeis Robotics club
2  Provides readbale wrapper functions for Zumo32U4Motors.h*/
3
4  #include <Arduino.h>
5  #include <Wire.h>
6  #include <Zumo32U4Motors.h>
7  #include "Drive.h"
8
9  Drive::Drive() {
10
11  }
12
13  void Drive::driveForward (int t) {
14      driveForward();
15      delay(t);
16      stopDrive();
17  }
18
19
20  void Drive::driveBackward(int t) {
21      driveBackward();
22      delay(t);
23      stopDrive();
24  }
25
26  void Drive::turnRight(int t) {
27      turnRight();
28      delay(t);
29      stopDrive();
30  }
31
32  }
33
34  void Drive::stopDrive (int t) {
35      stopDrive();
36      delay(t);
37  }
38
39  void Drive::driveForward() {
40      drive.setSpeeds(300, 300);
41  }
42
43  void Drive::driveBackward() {
44      drive.setSpeeds(300, -300);
45  }
46
47  void Drive::turnRight() {
48      drive.setSpeeds(-300, -300);
49  }
50  }
```

2018.3.13 Jacob Smith 4:34 PM: I added a turnLeft function to the Drive class and wrote the Drive\_Example file to include examples of driveForward, driveBackward, turnRight, and turnLeft

7:15 PM I wrote a Line class, which provides wrapper functions for the Zumo Line Sensor. The functions are isOnLine, which returns true if any of the sensors are on the line, and printAllSensors, which prints the sensors with the lcd parameter. The Class and Header file are included on the left of the next page. Matthew is included in the credit comment because he and I wrote the Line reader program that I code from to make this class.

March 14, 2018 Jacob Smith 6:18 PM: I put blue tape over the arena to make it a consistent color, going to take 10 readings from on the tape and ten readings from surrounding black tape

Reading Number	Inside Blue Tape Sensor 0	Inside Blue Tape Sensor 1	Inside Blue Tape Sensor 2	Border Black Tape Sensor 0	Border Black Tape Sensor 1	Border Black Tape Sensor 2
1	388	172	328	1300	936	
2	368	156	328	1204	936	
3	316	176	328	1204	972	
4	368	216	368	1204	956	
5	332	236	392	1200	1020	
6	332	228	372	1240	1020	
7	276	196	376	1240	1020	
8	268	212	408	1240	1028	
9	288	192	368	1276	1044	
10	324	228	352	1288	1008	

Absolute Max

408

Absolute Min

1200

Average of Absolute Min and Max

804

7:06 PM: Daniel Kang had the idea to take the max and min of the tape values, and have the cutoff of the line sensors be the average, which is 804. I watched the program for 5 minutes, and it did not fail during that time.

8:39 PM: Matthew, Aaron and I worked on trying to get the Proximity sensors to print their values to the screen, but we don't fully understand how they work yet.

March 15 2018 10:47 PM Jacob Smith: Today I ran a program that I wrote yesterday, which prints the robot's encoder values to the lcd. It functions normally.

Then, I wanted to use the encoders to have the robot know what direction it is going in. To accomplish this, I wrote a Timer class (left) which keeps track of the current Time by subtracting a reset point from the Arduino system's time. The example file that I wrote is on the next page, and it demonstrates the usage of the class. The most useful function is the interval function, which will return true for every say, 50 milliseconds, allowing a repeated task to be done without delay statements

```
/* Name: Jacob Smith
 *Email:jsmith2021@brandeis.edu
 * Date:March 14 2018
 * Assignment: Brandeis Robotics Club, prints encoder
readings
 */
#include<Zumo32U4LCD.h>
#include <Zumo32U4Encoders.h>
#include <Drive.h>

Zumo32U4Encoders sense;
Zumo32U4LCD lcd;
Drive motors;

void setup() {
  lcd.init();

}

void loop() {
  lcd.clear();
  lcd.print(sense.getCountsLeft());
  lcd.gotoXY(0,1);
  lcd.print(sense.getCountsRight());
  delay(100);

}
```

```
/*Written by Jacob Smith for Brandeis
Robotics club
Keeps track of current time
March 15 2018*/
```

```
#include <Arduino.h>
#include <Timer.h>
```

```
//Timer constructor
Timer::Timer(){
    initTime=millis();
}
```

```
//resets the initial time
long Timer::resetTime () {
    initTime=millis();
}
```

```
//returns the current time
long Timer::getTime(){
    return millis()-initTime;
}
```

```
//returns the current time and resets the
initial time
long Timer::getandResetTime(){
    long curTime=getTime();
    resetTime();
    return curTime;
}
```

```
//keeps track of a specified interval, so a
procedure can be performed every
second for example
bool Timer::interval(long interval){
    bool intPassed=false;
    if (getTime()>interval){
        intPassed=true;
        resetTime();
    }
    return intPassed;
}
```



March 21 2018 Jacob Smith 6:11 PM: Starting using the accelerometer. After looking at Zumo example file, I see that the accelerometer has public fields a.x and a.y and a.z for acceleration , and m for magnetometer

6:52 PM: Looking up converting sensor output to Gs <http://ozzmaker.com/accelerometer-to-g/>

March 28 2018: Last week, Matthew wrote this code to use the proximity sensors, but he concluded that the sensors aren't very reliable.

<pre>// // 3-21-18 Matthew Millendorf // This program intends to orient // the robot towards the direction of the other robot // results... //quite a frustrating day, sensors are not accurate enough for this type of algorithm // will revert to something much similar or try better algos/testing methods of sensors... or better sensors  #include &lt;Wire.h&gt; #include &lt;Zumo32U4.h&gt; #include &lt;Zumo32U4IRPulses.h&gt;  Zumo32U4LCD lcd; Zumo32U4Motors motors; Zumo32U4ProximitySensors proxSensors; Zumo32U4IRPulses pulses;  void setup() { proxSensors.initThreeSensors(); }</pre>	<pre>void loop() { // motors.setSpeeds(100, 100); //get the pulses going all the time pulses.start(1, 20, 10); pulses.start(0, 20, 10); proxSensors.read(); uint8_t A = proxSensors.countsRightWithRightLeds() + proxSensors.countsRightWithLeftLeds(); uint8_t B = proxSensors.countsRightWithLeftLeds() + proxSensors.countsFrontWithRightLeds(); uint8_t C = proxSensors.countsFrontWithLeftLeds() + proxSensors.countsFrontWithRightLeds(); uint8_t D = proxSensors.countsFrontWithLeftLeds() + proxSensors.countsLeftWithRightLeds(); uint8_t E = proxSensors.countsLeftWithRightLeds() + proxSensors.countsLeftWithRightLeds();  lcd.print("A:"); lcd.print(A); delay(1000); lcd.print("B: "); lcd.print(B); delay(1000); lcd.print("C:"); lcd.print(C); delay(1000);  //loops for sensing stuff if(((A + B)+1) &gt;= C) { // motors.setRightSpeed(200); // motors.setLeftSpeed(400); } if(E &gt;= C) { // motors.setRightSpeed(300); } }</pre>
---	---

```

/* Name: Jacob Smith
   Date: March 28 2018
   Email: jsmith2021@brandeis.edu
   Assignment: Finds Acceleratometer
   max and min values using an array of minimum,
   maximum, and current readings. The arrays are
   of length 3, for the x, y and z axes
*/

#include <LSM303.h>
#include <Wire.h>
#define NUM_AXES 3

LSM303 compass;
double mGPerLSB = .061;

double accelMin[NUM_AXES];
double accelMax[NUM_AXES];
double accelCur[NUM_AXES];

void setup() {
  Serial.begin(9600);//send data at 9600 bits per
  second, provided example value
  while (!Serial) {}
  Wire.begin();
  if (!compass.init()) {
    Serial.print("fail");
    delay(5000);
  }
  compass.enableDefault();
  Serial.println("Acceleration in milligs Brandeis
  Robotics Club");
  delay(2000);
}

void loop() {
  updateCurReadings();
  updateRange();
  if (millis()%100 == 0) {
    Serial.print("Current Acceleration Values:");
    printArr(accelCur);
    Serial.print("Maximum Acceleration Values:");
    printArr(accelMax);
    Serial.print("Minimum Acceleration Values:");
    printArr(accelMin);
    //while(1){}effectively stop program
  }
}

```

```

void updateRange() {
  for (int i = 0; i < NUM_AXES; i++) {
    if (accelCur[i] < accelMin[i]) {
      accelMin[i] = accelCur[i];
    }
    if (accelCur[i] > accelMax[i]) {
      accelMax[i] = accelCur[i];
    }
  }
}

void updateCurReadings() {
  compass.read();
  accelCur[0]=compass.a.x*mGPerLSB;
  accelCur[1]=compass.a.y*mGPerLSB;
  accelCur[2]=compass.a.z*mGPerLSB;
}

void printArr(double Arr []) {
  for (int i = 0; i < NUM_AXES; i++) {
    Serial.print(i);
    Serial.print(":");
    Serial.print(Arr[i]);
    Serial.print("\t");
  }
  Serial.println();
}

```

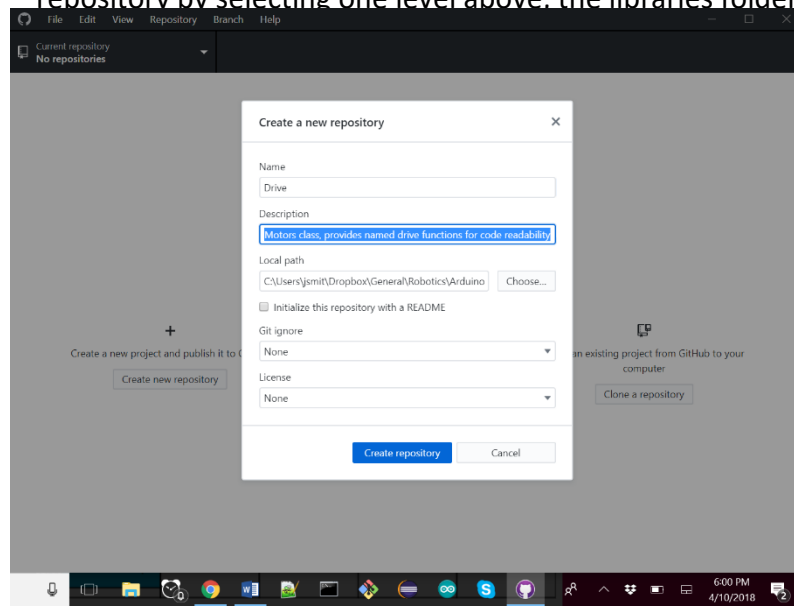
2018.3.30 12:12 AM: On Wednesday March 28 I expanded the min and max acceleration in one axis to a program that keeps track of the minimum and maximums of all of the axes with min, max, and current arrays. FUTURE PLANNING: Use this program to find the minimum and maximum values of the robot's resting state, to get an idea of the precision we can use. In addition, my plan is to use the printing programs on pages 16, 18 and 21 to print to the robot's position, velocity, and acceleration while moving and being rammed. The main goal right now is to use the robot's included sensors to give it an idea of its motion.

April 4 2018 8:23 PM Jacob Smith: I am documenting the Line, Drive, and Timer Libraries in preparation for their uploading to github. To clean up the Timer example file, I am trying to write a method to input a message and output to the lcd. A reference for that may be found here:<https://stackoverflow.com/questions/59670/how-to-get-rid-of-deprecated-conversion-from-string-constant-to-char-warnin>. Here is the updated timer example:

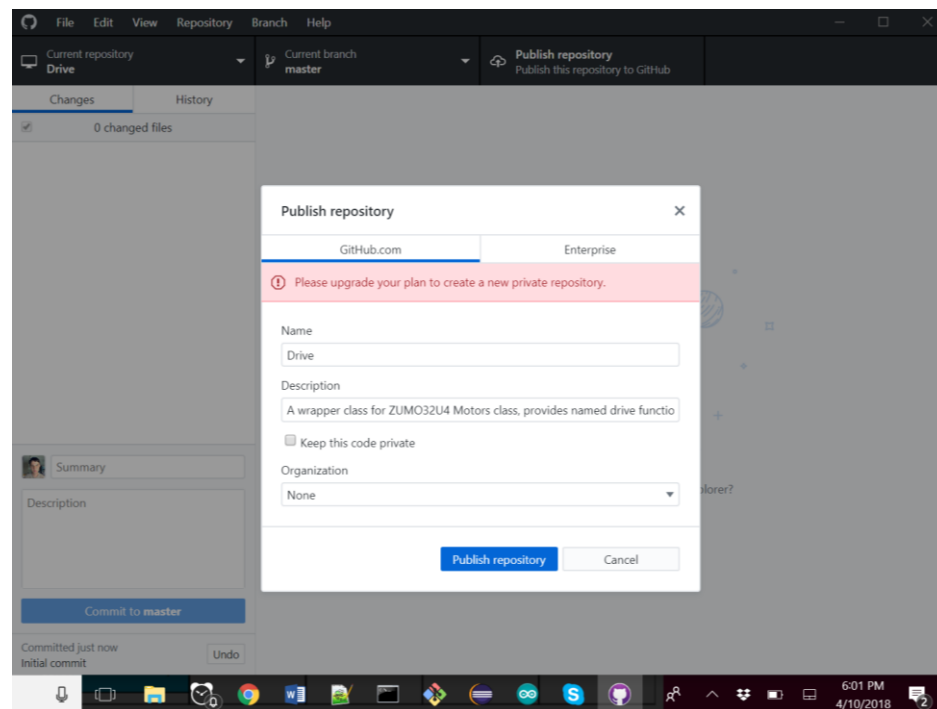
<pre> /* Written by Jacob Smith for Brandeis Robotics club.   Provides sample usage of the Timer class, which allows   a task to be repeated on an interval without delay statements, see the   loop. In the setup, the more basic functions of the Timer class are   demonstrated.   April 4 2018 */  //include the classes necessary to make this one work #include &lt;Timer.h&gt; #include &lt;Zumo32U4LCD.h&gt;  //declare robot display timer, and secondCount variable Zumo32U4LCD lcd; Timer timer; int secCount;  //setup is executed before main loop void setup() {    printNumber("getTime:", timer.getTime());   waitAndClear();    printNumber("Rsetime:", timer.resetTime());   waitAndClear();    printNumber("getReset", timer.getAndResetTime());   waitAndClear();    printNumber("getTime:", timer.getTime());   waitAndClear(); } </pre>	<pre> //main loop of execution //prints the time every second without usage of delay statements void loop() {   if (timer.interval(1000)) {     printNumber("Secs ",secCount);     secCount++;   } }  //wait for 2 seconds and clears the robot's display void waitAndClear() {   delay(2000);   lcd.clear(); }  //print a message and a number to the robot's display void printNumber (const char *message, long number) {   lcd.print(message);   lcd.gotoXY(0, 1);   lcd.print(number); } </pre>
---	--

April 4, 2018 Jacob Smith 5:33 PM: I am going to run the test files of all the libraries I've written so far before I upload them to GitHub.

- How to create a GitHub Repository (You need to be a maintainer to do this)
  - 1) I downloaded GitHub desktop: <https://desktop.github.com/>
  - 2) All of the files I want to be in GitHub repository are in Drive folder, I create a repository by selecting one level above, the libraries folder



- 3) Then I publish the repository



- 4) My repository shows up in my GitHub account

April 11 2018 7:20 PM Jacob Smith: After trying to get teams work in github, we decided that teams are too complicated. The new structure in github will consist of repositories in the Makerlab Page. This is simple to implement, highly collaborative, and is recommended for small groups. Finally, the repositories will be nested with folders.

<https://stackoverflow.com/questions/12258399/how-to-create-folder-in-github-repository>

I put the github project online manually, but I should really create git projects on my computer

<https://github.com/BrandeisMakerLab/Robotics-Zumo>

I tested the Timer and Line example files, both work, but Timer has a printing glitch

Future Planning: Get back to mapping robot's position, velocity, and acceleration. When I have a few more classes, I'll try to upload them with better workflow.

April 12 2018 2:46 PM: I cleared up the printing glitch from yesterday in the Timer example. I am updating the Accelerometer Min/Max array printing program from Page 23 to return acceleration in millimeters per second squared, which can better be compared to velocity and position

[https://en.wikipedia.org/wiki/Metre\\_per\\_second\\_squared](https://en.wikipedia.org/wiki/Metre_per_second_squared)

5:14 PM: I had the robot print its minimum/maximum and current accelerometer values over 10 minutes:

```
Acceleration in milligs Brandeis Robotics Club
Current Acceleration Values:0:208.18    1:-121.44    2:9838.09
Maximum Acceleration Values:0:415.15    1:71.78 2:10215.56
Minimum Acceleration Values:0:0.00      1:-332.00    2:0.00
```

0 is the x axis, 1 is the y axis, and 2 is the z axis. Note that the current reading on the z axis is 9838 millimeters per second squared, which is correct.

9:30 PM: Actually, I found a bug in my max and min program, the arrays were initialized to 0, which especially in the case of the z axis, would be the lowest value. Repeating the 10 minutes trial, I got these results. These can now be references for the natural variation of the accelerometer

```
Acceleration in millimeters be second squared Brandeis Robotics Club
Curs:0:269.19    1:-45.46    2:9926.62
Mins:0:10.17     1:-152.54    2:9559.92
Maxs0:456.43     1:138.19     2:10243.67
```

April 14 2018 3:34 AM: I wrote the Accel class, which wraps what I learned in making the min/max printing program into a class that takes care of initializing the accelerometer in its constructor, and returns the acceleration of the robot in the units of millimeters per second squared. Some notes on getting this to compile include no having semicolon after #define macro and having double colons to define class methods in class cpp file

```
/*Written by Jacob Smith for Brandeis Robotics club
   A wrapper class for the pololu LSM303 class, which interfaces with the accelerometer on
   the ZUMO32U4 robot
   Allows for simple initialization of the accelerometer in the class constructor and provides
   getter methods to
   get the robot's x, y, and z axis acceleration, as well as print it to the Serial Monitor
*/
//includes the libraries of code necessary to make this one work
#include <LSM303.h>
#include <Wire.h>
#include <Arduino.h>
#include "Accel.h"
//constants necessary to convert the accelerometer's raw values into millimeters per second
squared
#define MG_PER_LSB .061
#define MSS_PER_G 9.80665
}
```

2018.4.18 7:07 PM Jacob Smith: I have spent a while trying to connect the ZUMO32u4 robot to my computer, I uninstalled Arduino to try to fix it

I can show hidden devices in device manager to see com ports

Also, I was able to download a program to the Arduino Uno

April 19 2018 Jacob Smith: Today Tim found this link to connect the robot

<https://www.pololu.com/docs/0J61/10.1>. What you do is download a program to the robot twice, and while the computer is trying to connect to the ZUMO 32U4, press the reset button twice to set it into bootloader mode.


I also was having trouble recognizing the Zumo robot in Arduino, this fixed it

<https://forum.arduino.cc/index.php?topic=365367.0>, deleting old versions of ZUMO32u4 board

April 23, 2018 Jacob Smith: Still trying to resolve bootloader connection issues. 8:34 PM: Reinstalled Zumo driver, robot connects now. <https://www.pololu.com/docs/0J61/6.1>

10:40 PM: I worked out that my Accel class stops the robot from doing anything, but without it the robot can print to the serial.

11:19 PM: I rewrote the Accel class to not do anything in its constructor, and I moved the initialization of Serial, Wire, and the accelerometer to the setup method: My current hypothesis is that running these setup functions in the constructor of an object reference with global scope caused a problem, including my connection issues. Output is shown:

 COM14 (Pololu A-Star 32U4)

```
X Accel 55.03   Y Accel -113.66 Z Accel 9936.20
X Accel 67.60   Y Accel -139.38 Z Accel 9930.21
X Accel 66.40   Y Accel -108.87 Z Accel 9866.80
X Accel 67.60   Y Accel -119.04 Z Accel 9915.86
X Accel 68.79   Y Accel -111.27 Z Accel 9917.05
X Accel 85.54   Y Accel -113.66 Z Accel 9911.67
X Accel 71.19   Y Accel -121.44 Z Accel 9914.66
X Accel 67.60   Y Accel -122.03 Z Accel 9942.18
```

/\*Written by Jacob Smith for Brandeis Robotics Club

contains an example program to use the Accel class.

Prints Accelerations to the console in millimeters per second squared

April 23 2018

\*/

//includes the header file of the library and to Zumo library

#include <Accel.h>

#include <Wire.h>

//creates a global reference to an Accel object

```
/occurs before the program enters its main loop
void setup() {
  //set up USB communication with computer
  (only necessary if you want to print results to
  screen)
```

```
  Serial.begin(9600); //send data at 9600 bits per
  second, provided example value
```

```
  while (!Serial) {}
```

```
  //initialize accelerometer
```

```
  Wire.begin();
```

```
  accelerometer.initializeCompass();
```

```
  //print introduction
```

```
  Serial.println("Welcome to Brandeis University
  Acceleration demo");
```

```
  Serial.println("Results are in millimeters per
  second squared");
```

```
  delay(2000);
```

```
}
```

```
//the main loop of the robot
```

```
//prints the robot's accelerations on x y and z
  axes every tenth of a second
```

```
void loop() {
```

```
  Serial.print("X Accel\t");
```

```
  Serial.print(accelerometer.getX());
```

```
  Serial.print("\tY Accel\t");
```

```
  Serial.print(accelerometer.getY());
```

```
  Serial.print("\tZ Accel\t");
```

```
  Serial.println(accelerometer.getZ());
```

```
  delay(100);
```

```
}
```



April 24 2018 Jacob Smith 5:53 PM: I modified the accel class to return acceleration in centimeters per second squared, this should be closer to the distance of a zumo ring. I am still having connection issues with the Zumo bot, but they are solved by pressing reset button on robot twice and selecting com port in Arduino IDE→Tools→Ports

6:38 PM: Turning Encoder code on page 16 and 17 into a class. This source shows how to convert ticks into revolutions: <https://www.pololu.com/docs/0J63/3.4>.

6:51 PM: I am writing the Encoder class to only work for the left encoder, and once that works in an example file, I will write it for both sides.

7:09 PM: The Encoder Class now prints the number of rotations of the left side of robot. I need to convert rotations to distance.  $\text{circumference} = 2 \pi \text{ radius}$ . The radius of the wheel should be half of the height of the robot <https://www.pololu.com/docs/0J63/3.13>,  $39 \text{ mm}/2=19.5 \text{ mm}$

7:44PM, the test program now prints the robot's position in centimeters

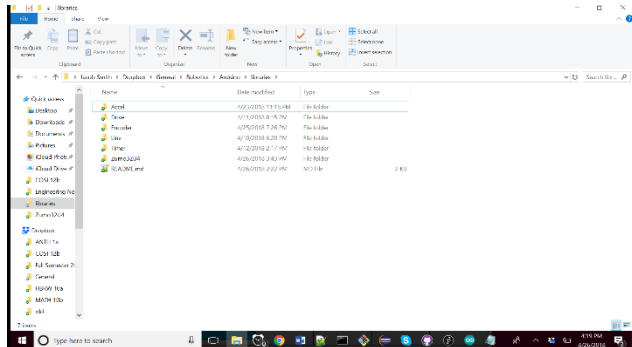
7:53 PM: I tested the robot next to a 1 meter tape measure, it resulted in about 97.7 centimeters.

FUTURE PLANNING: Find a more precise measurement of the radius of the wheel, add functionality for left and right sides, finish documenting Encoder class.

6:43 PM: I copied the variables for the left side of the robot for the right side of the robot, and added getRightPos and getRightVel functions. A key benefit of this change is that the getDirection method can check for turning

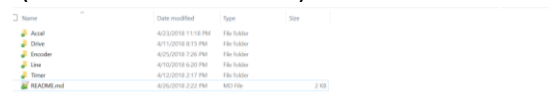
4.26.2018 Jacob Smith 2:28PM: I am going to properly create a GitHub repository instead of doing everything in the online editor.

4:18 PM: The goal is to create a GitHub repository from the libraries section of my Arduino folder:



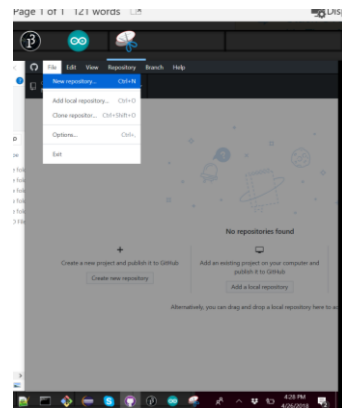
This way, the Arduino Integrated Development Environment will be able to recognize the libraries I write, but I can also easily update the GitHub page online. The main challenges to doing this is that there may be libraries that I don't want to include in my GitHub project, and also I want to accomplish this whole process without writing any code.

1. So first, I move any libraries that I don't want to be in my Arduino project to a temporary location (current libraries folder)



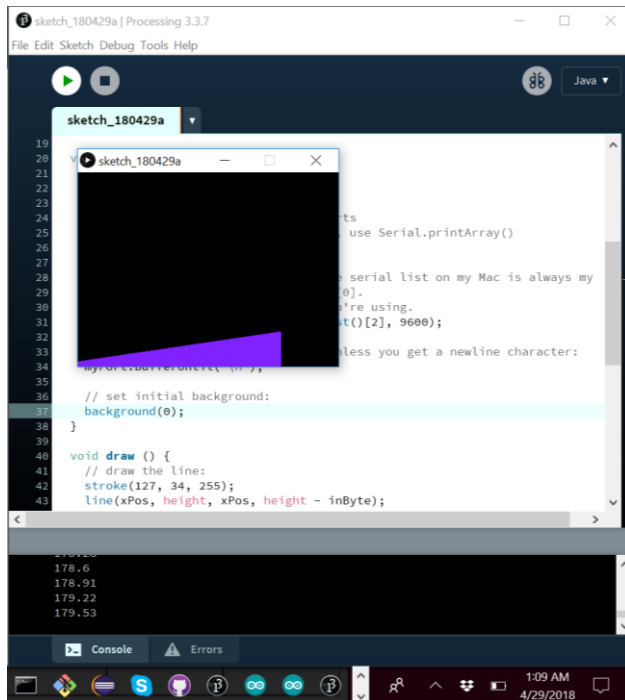
- a.
2. Then I create a new Repository using GitHub Desktop, which is a graphical user interface for GitHub that should allow us to create GitHub repositories without code. So I click "new repository" in file menu, and set the name to the libraries folder. This means that Instead of creating a whole new repository, git will load the contents of the libraries folder.

- a. <https://desktop.github.com/>
- b.



- c.

April 29 2018 12:41 AM Jacob Smith: I am looking into graphing the robot's motion, which should make programming sumo game responses easier. For example, knowing when the robot crosses a line will be simpler when I can visually see what the robot knows itself. The provided Arduino Graph example is designed to graph an analog value to Processing. The program works, but Processing doesn't include axes on its output. Here is the output I get when printing the robot's left position to the serial port and reading it with the provided processing code.



```

/*Written by Jacob Smith on April 29 2018
 Prints the position of the robot's left side
 to the serial port, for graphing testing in
 Processing.*/

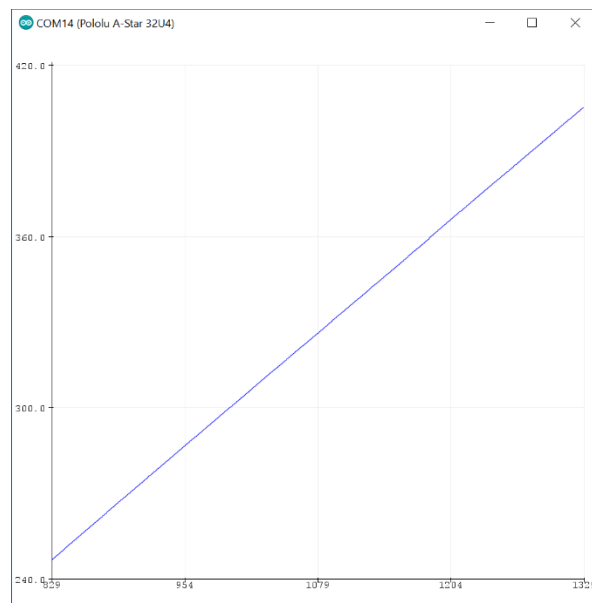
#include <Drive.h>
#include <Encoder.h>
Drive d;
int count = 0;
Encoder enc;
void setup() {
  Serial.begin(9600);
}

void loop() {
  d.driveForward();
  if (count < 1000) {
    Serial.println(enc.getLeftPos());
    count++;
  } else {
    count = 0;
  }
  delay(10);
}

```

Future planning: While this is promising, I should consider that graphing output isn't the highest priority of the project, and I can stick to printing positions to the Serial Monitor.

1:16 AM: Actually, I can get just as good results with the serial plotter. Future Planning: Generate a graph of the robot's position, velocity, and acceleration using the Serial Monitor for simplicity, although the serial plotter would serve as a good demonstration display.



April 29 2018 4:09 PM Jacob Smith: Today I will have the robot output its motion to the serial monitor and graph the results. This will allow an assessment of the accuracy of the robot's sensory information, because it will drive over a known time and distance at constant acceleration.

4: 55PM: I wrote average Position and averageVelocity functions in the encoder class so I can take advantage of both encoders instead of just using or printing one side.

```
//calls getPos helper method to find left and right positions, and averages them
//returns absolute value of position if one side is negative
double Encoder::getAvgPos(){
    double leftPos=getLeftPos();
    double rightPos=getRightPos();
    if (leftPos<0 && rightPos>0){
        leftPos=fabs(leftPos);
    }else if (leftPos>0 && rightPos<0){
        rightPos=fabs(rightPos);
    }
    double avgPos=(leftPos+rightPos)/2;
    return avgPos;
}

//calls getPos helper method to find left and right positions, and averages them
//if one velocity is negative, the absolute value of position is returned
double Encoder::getAvgVel(){
    double leftVel=getLeftVel();
    double rightVel=getRightVel();
    if (getDirection()=="TURNRIGHT"){
        rightVel=fabs(rightVel);
    }else if (getDirection()=="TURNLEFT"){
        rightVel*=-1;
    }
    double avgVel=(leftVel+rightVel)/2;
    return avgVel;
}
```

6:04 PM: Here is the program to print the robot's motion to the serial monitor. It prints the robot's motion every tenth of a second for 3 seconds. Then, I had the robot drive across a tape measure to see how reliable the motion information is. The position was about two centimeters off, which is reasonable, but the velocity is completely incorrect. While the results could be graphed, for the sake of time I leave them in table form

2018.5.1 5:04 PM Jacob Smith: Yesterday I tried to convert the units of velocity to centimeters per second, the constant MS\_PER\_S is 1000

```
//calculates velocity of left side of the robot, using a timer interval to find the change in position over a time interval
double Encoder::getLeftVel(){
    if (leftTimer.interval(TIME_INTERVAL)){
        finalLeft=getLeftPos();
        dLeft=(finalLeft-initLeft)/TIME_INTERVAL*MS_PER_S;
        initLeft=finalLeft;
    }
    return dLeft;
}
```

However, the velocities are still wrong, I traced the problem to the interval function of the timer not properly working. In addition, Yesterday I wrote one combined getVel function to save code between getting left and getting right velocities, but I will show that once I can get getLeftVel to work properly.

5:15 PM: After inserting a print command into the get velocity function, I see that the timer is working properly:

```
//calculates velocity of left side of the robot, using a timer interval
double Encoder::getLeftVel(){
    if (leftTimer.interval(TIME_INTERVAL)){
        Serial.println("Interval Reached");
        finalLeft=getLeftPos();
        dLeft=(finalLeft-initLeft)/TIME_INTERVAL*MS_PER_S;
        initLeft=finalLeft;
    }
    return dLeft;
}
```

When the time interval is half a second, and the loop in the Encoder motion print example file repeats every tenth of a second

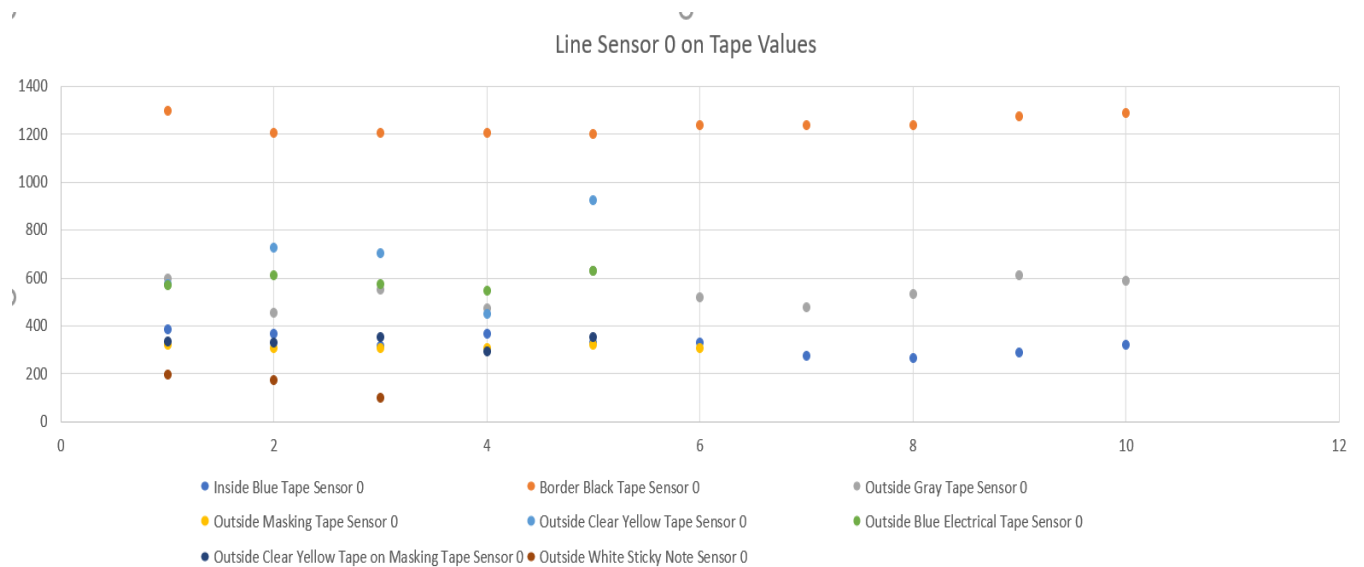
```
//main loop, prints the robot's position, velocity, and direction every half second
void loop() {
    printNumber("LEFTPOS:", enc.getLeftPos());
    printNumber("RIGHTPOS:", enc.getRightPos());
    printNumber("AVGPOS:", enc.getAvgPos());
    printNumber("LEFTVEL:", enc.getLeftVel());
    printNumber("RIGHTVEL:", enc.getRightVel());
    printNumber("AVGVEL:", enc.getAvgVel());
    Serial.print("\t");
    Serial.println(enc.getDirection());
    delay(100);
}
```

Then the interval should be reached once out of every five lines in the Serial monitor, which is what happens:

LEFTVEL:0.00	RIGHTVEL:0.00	AVGVEL:0.00	STOPPED
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	Interval Reached
LEFTVEL:0.00	RIGHTVEL:0.00	AVGVEL:0.00	STOPPED
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	LEFTVEL:0.00 RIGH
LEFTPOS:0.00	RIGHTPOS:0.00	AVGPOS:0.00	Interval Reached
LEFTVEL:0.00	RIGHTVEL:0.00	AVGVEL:0.00	STOPPED

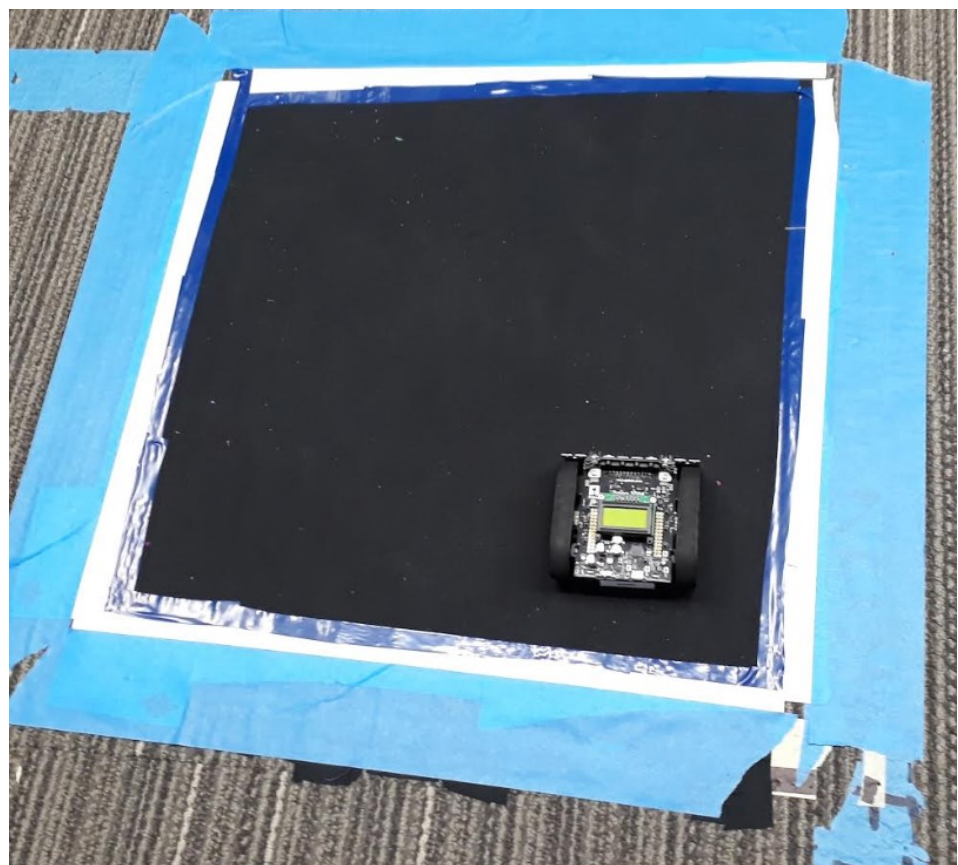
2018.5.2 Jacob Smith 5:52 PM: I will put a line of tape around the Zumo ring, so the robot can use its line sensor to know which robot won. This solution is much simpler than using the robot's velocity.

2018.5.3 3:10 AM: Over the night, I tested Line sensor readings on various surfaces:



From those readings, I selected the white sticky note, blue electrical tape, and outside marking tape for the best contrast.

Then, I updated the Zumo ring with those colors, it is very roughly a .5 m square. Starting from the inside, the black tape is the inside of the ring, the blue tape represents the in bounds border that the robot should drive away from, and the white sticky notes represent the out of bounds region. The blue tape on the very outside holds the white sticky notes down.



2018.5.4 12:44 AM Jacob Smith: I am rewriting the Line class to allow for three regions of tape. To use a nondefault constructor in Arduino, I have this link: <https://www.arduino.cc/en/Hacking/LibraryTutorial>, and to use the this keyword in c++<https://stackoverflow.com/questions/6905598/c-equivalent-to-java-this..>

3:37 AM: I added a constructor to the Line class which allows the threshold values to be modified without touching the actual Line class

```
//assigns line sensor threshold values
//edge are leftMost and Rightmost line sensors
//middle is middle Line Sensor
Line::Line(int edgeLow,int edgeHigh,int middleLow,int middleHigh) {
    lineReader.initThreeSensors();
    this->edgeLow=edgeLow;
    this->edgeHigh=edgeHigh;
    this->middleLow=middleLow;
    this->middleHigh=middleHigh;
}
```

With these thresholds, I wrote the getRegion function, which returns where the robot is in the sumo ring. The logic of this method took some troubleshooting, but now it works so that being in bounds is the default case, and if any of the sensor values is in the edge or out of bounds range, then the robot is in that range.

```
//returns a string representing where the robot is in the zumo ring, using three colors of tape going from dark on inside to blue on ec
//INRING,EDGE,OUTBOUNDS
const char* Line::getRegion(){
    lineReader.read(reflections,true);
    if(within(reflections[0],edgeLow,edgeHigh)||within(reflections[1],middleLow,middleHigh)||within(reflections[2],edgeLow,edgeHigh)){
        return "EDGE";
    }else if(reflections[0]<edgeLow || reflections[1]<middleLow || reflections[2]<edgeHigh){
        return "OUTBOUNDS";
    }else{
        return "INRING";
    }
}
```

And this is the within helper method to make the getRegion more readable:

```
//returns whether the first arument is within the second and third
bool Line::within(int num, int min,int max){
    return num>min && num<max;
}
```



2018.5.6 10:04 PM Jacob Smith: I rewrote the Line Class to have three methods, `isInRing`, `isOnEdge`, and `isOutOfBounds`. These methods are less clunky to use in logic, and will be easier to use as pointers.

```
//first takes line sensor readings to see where the robot is in the sumo ring, and
//returns true if all of the sensors are darker than the edge tape, false otherwise
bool Line::isInRing(){
    lineReader.read(reflections,true);
    if (reflections[0]>=edgeHigh && reflections[1]>=middleHigh && reflections[2]>=edgeHigh){
        return true;
    }
    return false;
}

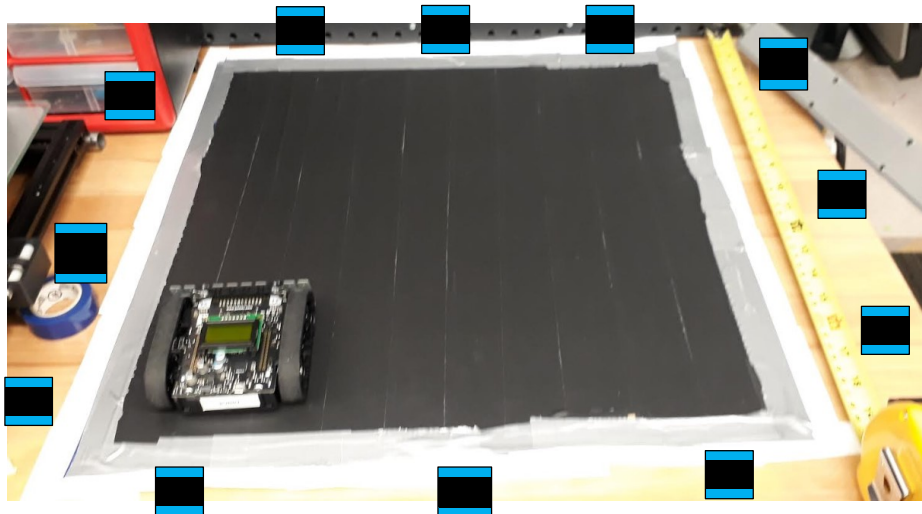
//first takes line sensor readings to see where the robot is in the sumo ring, and
//returns true if any of the line sensor reflections are bright enough to be the out of bounds tape
bool Line::isOutOfBounds(){
    lineReader.read(reflections,true);
    if(reflections[0]<edgeLow || reflections[1]<middleLow || reflections[2]<edgeLow){
        return true;
    }
    return false;
}

//first takes line sensor readings to see where the robot is in the sumo ring, and
//returns true if any of the sensors are brighter than the inside tape but darker than the out of bounds tape
bool Line::isOnEdge(){
    lineReader.read(reflections,true);
    if(within(reflections[0],edgeLow,edgeHigh)||within(reflections[1],middleLow,middleHigh)||within(reflections[2],edgeLow,edgeHigh)){
        return true;
    }
    return false;
}
```

Then, I use these simple methods in the Complex Line Reader, leading to cleaner code.

```
//allows the robot to decide where to drive
void navigate() {
    robot.driveForward();
    if (line.isOnEdge()) {
        turnBack();
    } else if (line.isOutOfBounds()) {
        numLosses++;
        robot.stopDrive();
        speaker.playNote(NOTE_D(6), 500, 15);
        delay(2000);
        turnBack();
    }
}
```

Also today, I made another sumo ring on a desktop, so it isn't on the floor anymore, the old one was getting roughed out. However, when testing out line readings, I was getting brighter blue electrical tape readings. This makes sense, because the blue tape is translucent, so the wood of the table makes for a brighter reflection. This caused me to switch to duct tape as the edge, which is opaque





2-18.5.7 3:01 AM Jacob Smith: I have been redesigning the Drive class to avoid repeated code. This is more necessary now, because the Drive class is going to become more complicated, and now I will only have to make changes in one place. To accomplish this, I am writing a private move function, which does the real work of moving the robot. The driveForward, driveBackward, turnRight, and turnLeft public functions then simply have to call this move function. To accomplish this, I took an example from this source:

[https://www.ibm.com/support/knowledgecenter/en/SSLTBW\\_2.3.0/com.ibm.zos.v2r3.cbclx01/cplr242.htm](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.cbclx01/cplr242.htm) and modified it to work in Arduino with void methods.

---

```
/* Based on example code from IBM,
 *  shows how method references work in arduino
 *  Jacob Smith 2018.5.7
 */
void g() {}
void bar (void f()) {
    f();
}

void setup() {

}

void loop() {
    bar(g);
}
```

2018.5.7 7:02 PM Jacob Smith: I am rewriting the Drive class so that DriveForward,driveBackward, turnLeft, and turnRight functions all point to the same master move function. This will save coind and centralize it, especially as the Drive class becomes more complicated. To do this, I found an online sample of function pointers: <http://www.giannistsakiris.com/2012/09/07/calling-a-member-function-pointer/>. Then, I rewrote the drive class to reference the move function.

It is important to note that, even though this strucutre for the Drive class is more complicated and centralized, the public methods haven't changed. So the end user can still use the class just as easily as before.

10:57 PM: The robot is detecting being on the edge while it is in the ring. I think this is because the line sensors are desensitized to light. Future Planning: fix the line sensors, then add an abort in the move function if the robot is on the edge or out of bounds.

```
//the main driving method, performs one
//sets the robot to perform the basic ac
void Drive::move(void (Drive::*command)
    driveTimer.resetTime();
    while (driveTimer.getTime()<time){
        (*this.*command)();
    }
    stopDrive();
}

//public methods to command the robot to
//refer to private move helper method
void Drive::driveForward(int time) {
    move(&Drive::driveForward,time);
}

void Drive::driveBackward(int time) {
    move(&Drive::driveBackward,time);
}

void Drive::turnRight(int time) {
    move(&Drive::turnRight,time);
}

void Drive::turnLeft(int time){
    move(&Drive::turnLeft,time);
}

void Drive::stopDrive(int time) {
    move(&Drive::stopDrive,time);
}
```

2018.8.14 Jacob Smith 8:38 PM: I have been researching how to build a proper mini-sumo ring, here are some helpful links

<https://maker.tufts.edu/projects/sumobots>

2018.8.15 Jacob Smith 1:33 PM I also have some links for sensors that Deis Robotics may want

**MyoWare Muscle Sensor:** <https://www.pololu.com/product/2732>

**Electrodes for MyoWare Muscle Sensors (6-Pack)** <https://www.pololu.com/product/2733>

**Pololu IR Beacon Transceiver Pair** <https://www.pololu.com/product/702>

**Parallax ColorPAL** <https://www.pololu.com/product/1608>

So the options for construction are:

## Mini Sumo Ring Options:

Type of Ring	Link	Pros	Cons
Ring Poster	<a href="https://engineerdog-webstore.fwscart.com/SimpleSumoCompetitionPoster/p497040917776128.aspx">https://engineerdog-webstore.fwscart.com/SimpleSumoCompetitionPoster/p497040917776128.aspx</a>	Ready to order ring	Colors aren't official regulations
Pre Cut Wooden Ring	<a href="https://www.instructables.com/id/How-to-Make-a-Mini-Sumo-Ring/">https://www.instructables.com/id/How-to-Make-a-Mini-Sumo-Ring/</a> Option 2  List of supplies: <a href="https://www.amazon.com/hz/wishlist/ls/28AAHJ3MS18PQ">https://www.amazon.com/hz/wishlist/ls/28AAHJ3MS18PQ</a>	Less work than cutting ring ourselves	Ring can't be recessed
Self Cut Wooden Ring	<a href="http://www.robotroom.com/SumoCircleMini.html">http://www.robotroom.com/SumoCircleMini.html</a>	Would let ring be either raised or recessed	Takes a lot of physical work

Rules of Mini Sumo: <http://robogames.net/rules/all-sumo.php>

The other consideration as we think about building the mini sumo ring is how the robot should detect that it has gone over the edge, especially if that involved an extra color of paint as an out of bounds marker for the infrared sensors

September 20 2018 4:10 PM Jacob Smith: I am preparing a demonstration for the robotics club meeting, using this tape I got from Tim: [https://www.amazon.com/dp/B01NANPNZC?ref=yo\\_pop\\_ma\\_swf](https://www.amazon.com/dp/B01NANPNZC?ref=yo_pop_ma_swf)

Line Sensor Printing Values

Black Tape: 1160,900,1016, avg=1025

White Table: 160, 100, 100 avg =120

September 27 2018 5:18 PM Jacob Smith: TEACHING: So Last week the students used my Drive class to get the robot to push another robot out of the ring. The idea is to introduce students in to how to actually get the robot to drive.

---

```

/*Written by Brandeis Robotics Club
The starting drive program for the club, the rob
The program is documented for a general audience
September 27 2018

500 milliseconds is 19.5 inches
300 milliseconds is 90 degree turn

The robot starts at end of ring facing center, tr
*/

//includes the header file of the library
#include <Drive.h>

//creates a global reference to a Drive object
Drive robot;

//occurs before the program enters its main loop
void setup() {
  //commands the robot to stop for 2 seconds
  robot.stopDrive(2000);

  teamASequence();
  //teamBSequence();
}

//the main loop of the robot
void loop() {
  }
}

void teamASequence() {
  //move to center
  robot.driveForward(500);
  //turn left 45 degrees
  robot.turnLeft(150);
  //move back all the way
  robot.driveBackward(500);
  //move forward & ram other robot
  robot.driveForward(1000);
}

void teamBSequence() {
  //Turn left to face robot
  robot.turnLeft(70);
  //move forward to push the robot out of the arena
  robot.driveForward(1500);
}

```

POSTERS: Also, Tim bought two posters with the official color layout, which the poster on the previous page doesn't have. <https://www.parallax.com/product/27404>

---

<http://www.sumorobotleague.com/purchasekits/competition-ring>

10.4.2018 Jacob Smith 3:55 PM: Last week, I showed the members how loops repeat code over and over again. This is the difference between the loop and setup methods in adruino, setup doesn't repeat and loop does. I asked them to use this idea to write a program for the robot to drive in a square.

```

//the main loop of the robot
void loop() {
  //move forward
  robot.driveForward(500);
  robot.turnLeft(300);
}

```

They came up with the left solution, which shows that driving in a square is really a repeated process.

Then, I showed them how to use the isOnLine function to determine if it is on the edge of the ring, and how they can write a program that does one thing when the robot is on the edge, and another when the robot is inside the ring.

```

/* Name: Jacob Smith and Matthew Millendorf, Brandeis Robotics Club
Date: April 4 2018
Assignment: Line Follower, designed to keep the robot on a line. It is designed
by turning left whenever it sees a line. It is designed to follow a line.
See the documentation of the Brandeis Robotics Club

*/
//include all the libraries that this program needs to
#include <Wire.h>
#include <Drive.h>
#include <Line.h>
#include <Zumo32U4LCD.h>

//declare objects to control the robot's display, motor
Zumo32U4LCD lcd;
Drive motors;
Line lineReader(527);

//this method starts before the program enters the main loop
//the robot will display its program name, then display
//finally it will drive forward for 100 milliseconds, so
void setup() {
  //start robot on line, facing inward
  lineReader.printAllSensors(lcd);
}

//the main loop of the program, the robot will drive forward
//unless it sees a line, in which case it will turn left
void loop() {
  if (lineReader.isOnEdge()) {
    motors.stopDrive(500);
    motors.turnLeft(450);
  } else {
    motors.driveForward();
  }
}

```

The program they filled in to the loop was the correct idea of how to make sure the robot stays in the ring, but in practice the line detection didn't work very well.

October 11 2018, Last week, the students worked on using a provided proximity sensor program to think about how it could help the robot in the ring. This fits in to the theme of giving the robot more information on the world around it. Here is the example program, which is a lot more effective than the one I tried on page 15. <<https://www.youtube.com/watch?v=ddPo6HQvxzQ>>. Today, I am going to build the example program into a class so the students can use it in a strategy.

I wrote proximity sensor class, and it has a nice lcd printing trick. Previously, I was clearing the LCD every time and having a delay. But the example code just resets the cursor instead of clearing, so no delay is necessary.