Group Member Names: Dalton Ellis, Jason Ross, Justin Valentine, Ken Aoyagi

Course and Quarter: ENG114

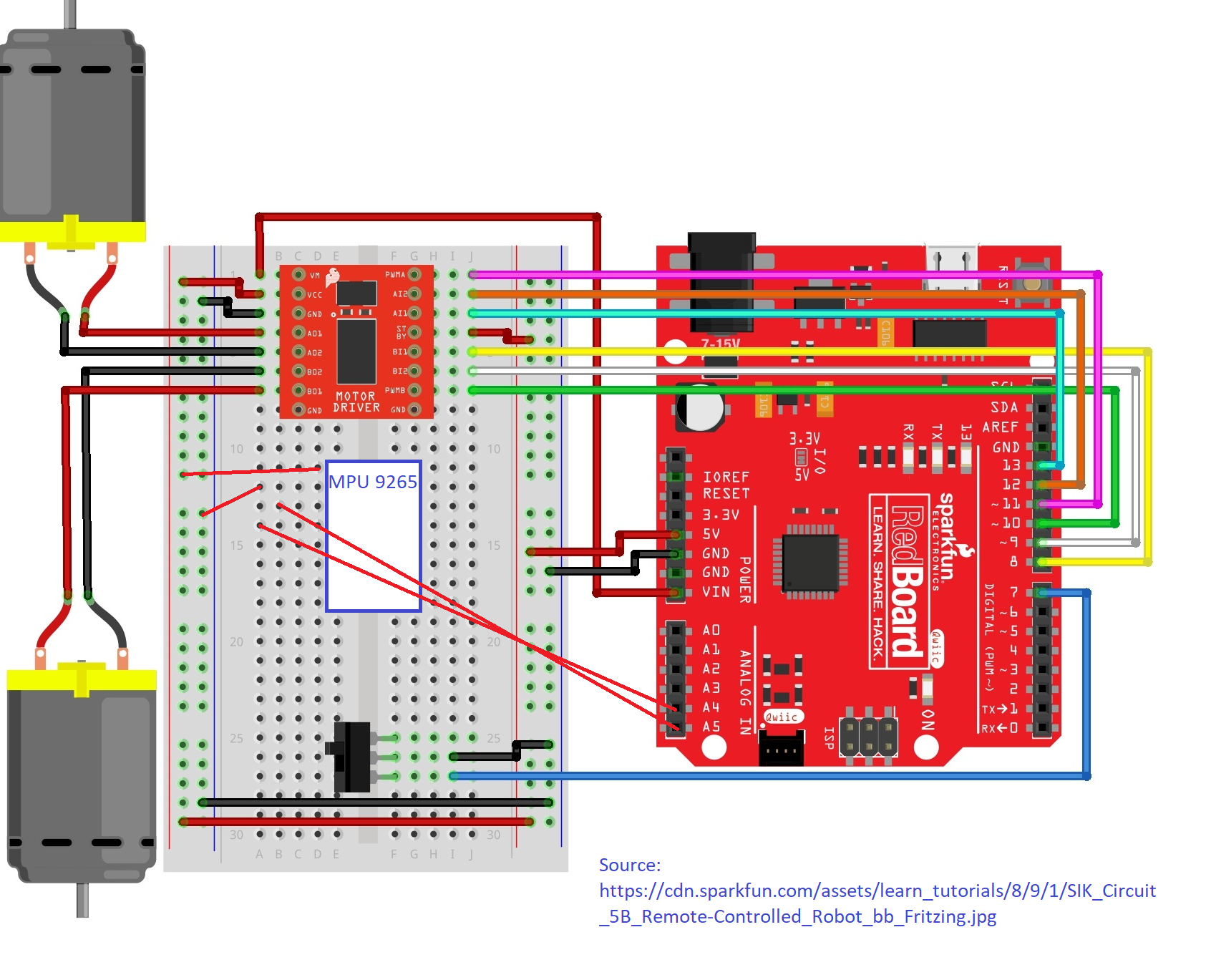
Date: 12/7/2019

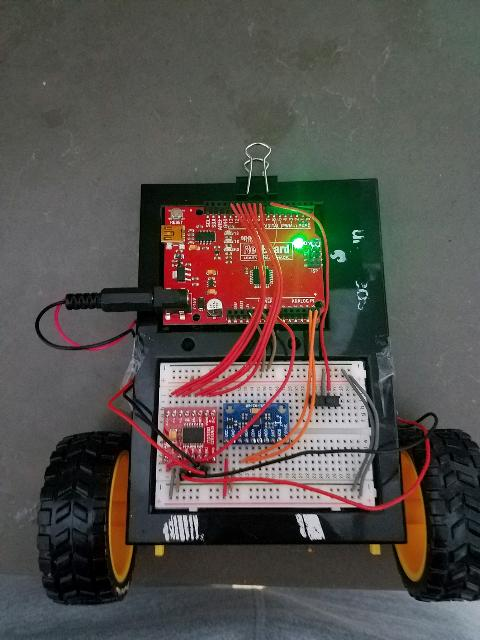
Revision Number:

## **Seesaw Car**

#### **Problem Statement:**

Our group chose to use an electric motor car to balance a Seesaw.

**Hardware Setup**

****

**Materials:**

Bill of Materials:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part Name | Purpose | Item Name | URL | Price |
| Redboard | Microprocessor | Arduino Uno | <https://www.sparkfun.com/products/15123> | $20 |
| Arduino and  breadboard holder | Base for breadboard | Arduino and breadboard holder | [**https://www.sparkfun.com/products/11235**](https://www.sparkfun.com/products/11235) | 3.95 |
| Motor driver | Controls voltage levels to motors | **Dual TB6612FNG** | [**https://www.sparkfun.com/products/14450**](https://www.sparkfun.com/products/14450) | 5.45 |
| accelerometer | Sensing pitch level | MPU 92/65 | <https://www.banggood.com/buy/mpu-9265.html> | 12 |
| Hobby gearmotor | Moves vehicle | DG01D | <https://www.sparkfun.com/products/13302> | 4.95 |
| Wheels | Provide rotational inertia and minimal points of contact to board | Wheel | <https://www.sparkfun.com/products/13259> | 2.95 |
| Seesaw | Tilt car | Seesaw | HOMEMADE | N/A |
| Jumper wires | Facilitates flow of electrons with minimal resistance | Jumper wires | <https://www.sparkfun.com/products/11026> | 2.25 |
| Mini power switch | Turns off power | Mini power switch SPDT | <https://www.sparkfun.com/products/102> | 1.50 |

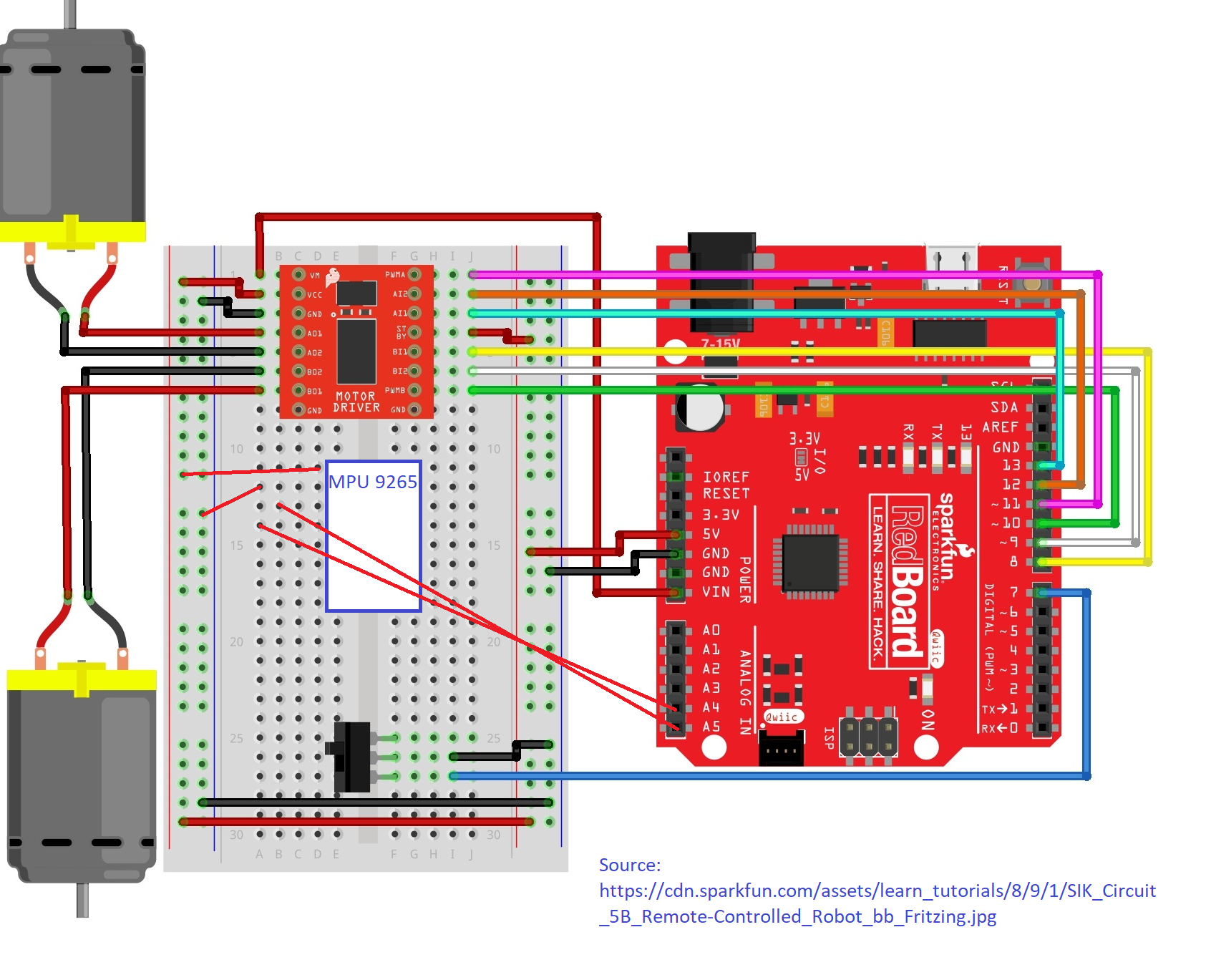
Hookup Guide:

|  |  |
| --- | --- |
| Redboard | Motor Driver |
| Digital 11 | PWMA |
| Digital 12 | A12 |
| Digital 13 | A11 |
| Digital 10 | PWMB |
| Digital 8 | B11 |
| VIN | VM |
| 5V | STBY |

|  |  |
| --- | --- |
| Left motor | Motor driver |
| Black wire | A02 |
| Red Wire | A01 |

|  |  |
| --- | --- |
| Right motor | Motor driver |
| Black wire | B02 |
| Red Wire | B01 |

|  |  |
| --- | --- |
| Accelerometer | Redboard |
| SCL | Analog A5 |
| SDA | Analog A4 |



#### 

#### **Arduino Code:**

/\*

SparkFun Inventor’s Kit

Circuit 5B - Remote Control Robot

Control a two wheeled robot by sending direction commands through the serial monitor.

This sketch was adapted from one of the activities in the SparkFun Guide to Arduino.

Check out the rest of the book at

https://www.sparkfun.com/products/14326

This sketch was written by SparkFun Electronics, with lots of help from the Arduino community.

This code is completely free for any use.

View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40

Download drawings and code at: https://github.com/sparkfun/SIK-Guide-Code

\*/

#include "MPU9250.h"

// an MPU9250 object with the MPU-9250 sensor on I2C bus 0 with address 0x68

MPU9250 IMU(Wire,0x68);

int status;

//the right motor will be controlled by the motor A pins on the motor driver

const int AIN1 = 13; //control pin 1 on the motor driver for the right motor

const int AIN2 = 12; //control pin 2 on the motor driver for the right motor

const int PWMA = 11; //speed control pin on the motor driver for the right motor

//the left motor will be controlled by the motor B pins on the motor driver

const int PWMB = 10; //speed control pin on the motor driver for the left motor

const int BIN2 = 9; //control pin 2 on the motor driver for the left motor

const int BIN1 = 8; //control pin 1 on the motor driver for the left motor

int switchPin = 7; //switch to turn the robot on and off

const int driveTime = 20; //this is the number of milliseconds that it takes the robot to drive 1 inch

//it is set so that if you tell the robot to drive forward 25 units, the robot drives about 25 inches

const int turnTime = 8; //this is the number of milliseconds that it takes to turn the robot 1 degree

//it is set so that if you tell the robot to turn right 90 units, the robot turns about 90 degrees

//Note: these numbers will vary a little bit based on how you mount your motors, the friction of the

//surface that your driving on, and fluctuations in the power to the motors.

//You can change the driveTime and turnTime to make them more accurate

String botDirection; //the direction that the robot will drive in (this change which direction the two motors spin in)

String distance; //the distance to travel in each direction

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

// serial to display data

Serial.begin(115200);

while(!Serial) {}

// start communication with IMU

status = IMU.begin();

if (status < 0) {

Serial.println("IMU initialization unsuccessful");

Serial.println("Check IMU wiring or try cycling power");

Serial.print("Status: ");

Serial.println(status);

while(1) {}

}

{

pinMode(switchPin, INPUT\_PULLUP); //set this as a pullup to sense whether the switch is flipped

//set the motor control pins as outputs

pinMode(AIN1, OUTPUT);

pinMode(AIN2, OUTPUT);

pinMode(PWMA, OUTPUT);

pinMode(BIN1, OUTPUT);

pinMode(BIN2, OUTPUT);

pinMode(PWMB, OUTPUT);

}

}

void rightMotor(int motorSpeed) //function for driving the right motor

{

if (motorSpeed > 0) //if the motor should drive forward (positive speed)

{

digitalWrite(AIN1, HIGH); //set pin 1 to high

digitalWrite(AIN2, LOW); //set pin 2 to low

}

else if (motorSpeed < 0) //if the motor should drive backward (negative speed)

{

digitalWrite(AIN1, LOW); //set pin 1 to low

digitalWrite(AIN2, HIGH); //set pin 2 to high

}

else //if the motor should stop

{

digitalWrite(AIN1, LOW); //set pin 1 to low

digitalWrite(AIN2, LOW); //set pin 2 to low

}

analogWrite(PWMA, abs(motorSpeed)); //now that the motor direction is set, drive it at the entered speed

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void leftMotor(int motorSpeed) //function for driving the left motor

{

if (motorSpeed > 0) //if the motor should drive forward (positive speed)

{

digitalWrite(BIN1, HIGH); //set pin 1 to high

digitalWrite(BIN2, LOW); //set pin 2 to low

}

else if (motorSpeed < 0) //if the motor should drive backward (negative speed)

{

digitalWrite(BIN1, LOW); //set pin 1 to low

digitalWrite(BIN2, HIGH); //set pin 2 to high

}

else //if the motor should stop

{

digitalWrite(BIN1, LOW); //set pin 1 to low

digitalWrite(BIN2, LOW); //set pin 2 to low

}

analogWrite(PWMB, abs(motorSpeed)); //now that the motor direction is set, drive it at the entered speed

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop(){

if(digitalRead(7) == LOW){

IMU.readSensor(); // read the sensor

float y = IMU.getAccelY\_mss();

while (y!=-2.501){

IMU.readSensor(); // read the sensor

Serial.print("AccelY: ");

Serial.print(IMU.getAccelY\_mss(),6);

Serial.print(" ");

Serial.println();

delay(100);

float y = IMU.getAccelY\_mss();

Serial.print(y);

Serial.print(" ");

Serial.println();

/\*The following section of code is designed to create a running average of values from the accelerometer. This was necessary to eliminate spikes in torque that created problems that caused t

\*the car to jerk back and forth in a way that did not allow for proper balance.

\*/

float prev\_y1 = y;

delay(10);

float prev\_y2 = y;

delay(10);

float prev\_y3 = y;

delay(10);

float prev\_y4 = y;

delay(10);

float prev\_y5 = y;

delay(10);

float prev\_y6 = y;

delay(10);

float prev\_y7 = y;

delay(10);

float prev\_y8 = y;

delay(10);

float prev\_y9 = y;

delay(10);

float prev\_y10 = y;

delay(10);

float prev\_y11 = y;

delay(10);

float prev\_y12 = y;

delay(10);

float prev\_y13 = y;

delay(10);

float prev\_y14 = y;

delay(10);

float prev\_y15 = y;

delay(10);

float prev\_y16 = y;

delay(10);

float prev\_y17 = y;

delay(10);

float prev\_y18 = y;

float avg\_y = (prev\_y1 + prev\_y2 + prev\_y3 + prev\_y4 + prev\_y5 + prev\_y6 + prev\_y7 + prev\_y8 + prev\_y9 + prev\_y10 + prev\_y11 + prev\_y12 + prev\_y13 + prev\_y14 + prev\_y15 + prev\_y16 + prev\_y17 + prev\_y18) / 18;

if (avg\_y >= -2.8 && avg\_y <= -2.2) // if statement to decide direction, seems to not recieve data from the accelerometer

{

rightMotor(0); // drive the right wheel forward

leftMotor(0);

Serial.println("motorstuff"); // drive the left wheel forward

// delay(driveTime \* distance.toInt()); // drive the motors long enough travel the entered distance

// rightMotor(0); // turn the right motor off

// leftMotor(0);

}

else if (avg\_y > -2)

{

rightMotor(80); // drive the right wheel forward

leftMotor(80); // drive the left wheel forward

}

else if (avg\_y < -3)

{

rightMotor(-80); // drive the right wheel backward

leftMotor(-80); // drive the left wheel backward

}

else

{

continue;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#### **Results:**

Our result was that after some experimenting with the mechanics of the seesaw and the code we were able to successfully make our car automatically move to balance the seesaw.



#### 

#### 

#### **Future Work:**

If a future group wanted to improve on this project they could do it in a number of ways. Our group was able to make the seesaw function as planned but we did have to include some rubber bands to create tension under the board that makes the seesaw move slower. A future group could improve on our project by gradually increasing the speed of each motor immediately after a direction change so that the wheels don’t peel out and the car could theoretically balance the motor faster.

They could also incorporate a certain amount of machine learning so that the car could learn on it’s own when to correct direction.

They could also incorporate code to correct for motor directionality so that the car could stay straight without human correction.

#### **License**

Electric Car for Balancing a Seesaw

Copyright (C) 2019 J. Valentine, K. Aoyagi, D. Ellis, J. Ross

This program is free software: you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program. If not, see <<https://www.gnu.org/licenses/>>.