Line Following Robot

Alfred Palacios
Daniel Delgado
Alberto Luna

Connections

Procedure:

Step 1: Connect the circuit as shown in the schematic.

Step 2: Use the Arduino IDE to write your own code.

Step 3: Upload your code to the Arduino and connect it to the batteries or you can even use a power bank to run the Arduino.

Step 4: Test it on a black line path.

Connections: Line Sensor

QRE113 Line Sensor: Each line sensor has an IR emitter-detector pair. The sensors are mounted so that the pair is close to the floor. Each sensor has three pin connections. The ground on the sensor is connected to ground on the Arduino pin, the sensor has a VCC pin that is connected to the 5V pin on the Arduino, the OUT sensor pin is connected to A0 and A2 pin on the Arduino board.

NOTE: The distance between the floor and the detector should be only about a quarter of an inch. Both sensors must be positioned closely. You may use a wire or paper clip to position the sensors correctly. If the robot does not respond correctly to the line, it may be due to the distance between the floor and the sensors.

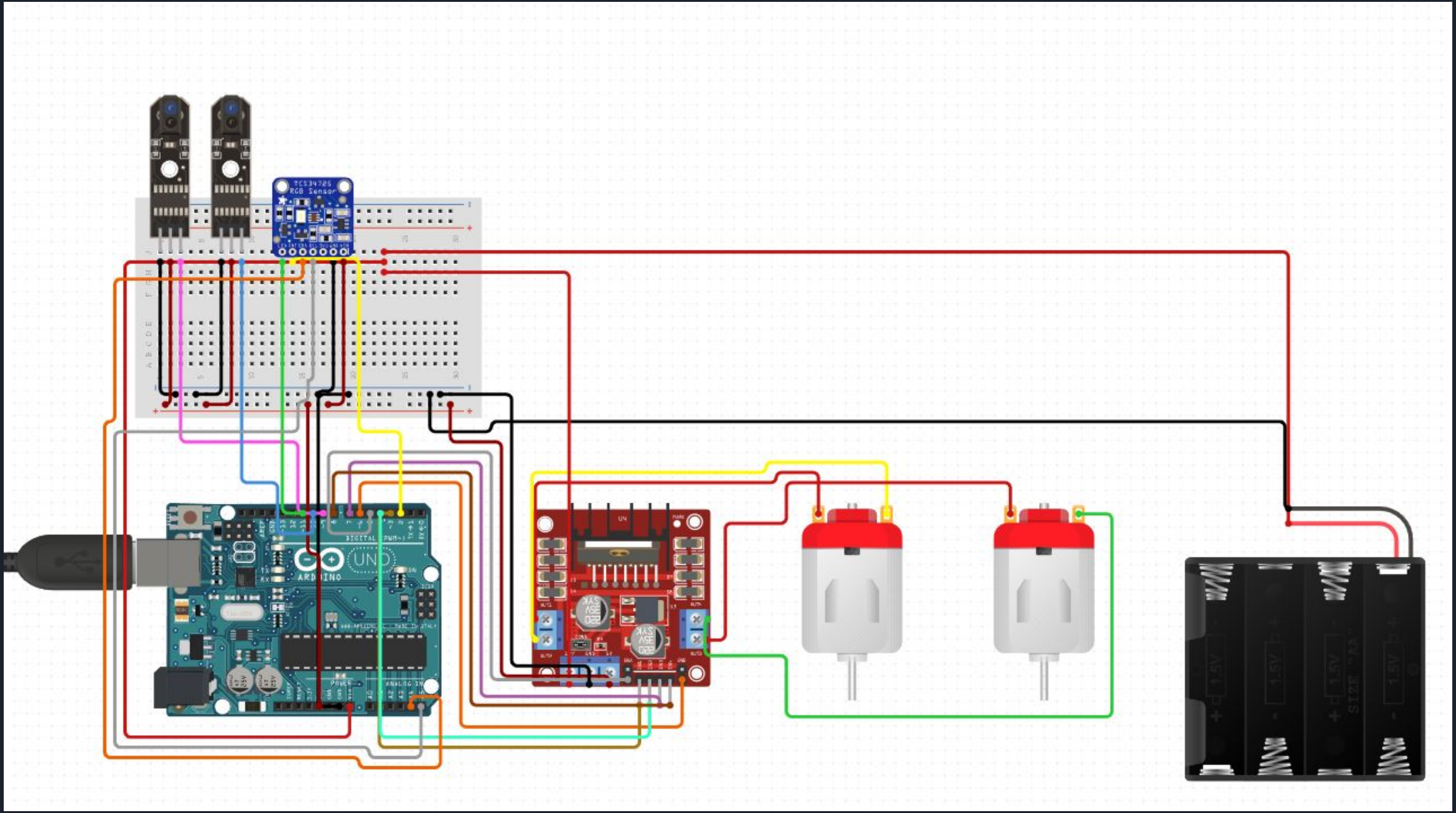
Connections: Color Sensor

TCS34725 Color Sensor: This color sensor will be used to allow the line following robot to stop when a piece of black tape is read by the sensor. The color sensor may be placed on either side of the line following robot. The color sensor should be capable of detecting Red, Green, and Blue. The color sensor needs to be as close to the floor as possible. A wire or paper clip may be used to attach the sensor to the chassis.

The color sensor has 4 pins that will be used. A Ground , Vin, SDA, and SCL pin. The Ground pin on the sensor will be connected to the Ground pin on the Arduino. The Vin on the sensor will be connected to the to the 5V pin on the Arduino board. The SDA pin on the sensor will be connected to the the SDA pin on the Arduino board. The SCL pin will be connected to the SCL pin on the Arduino board.

Note: You may need to download a library for the color sensor code in order for it to work properly.

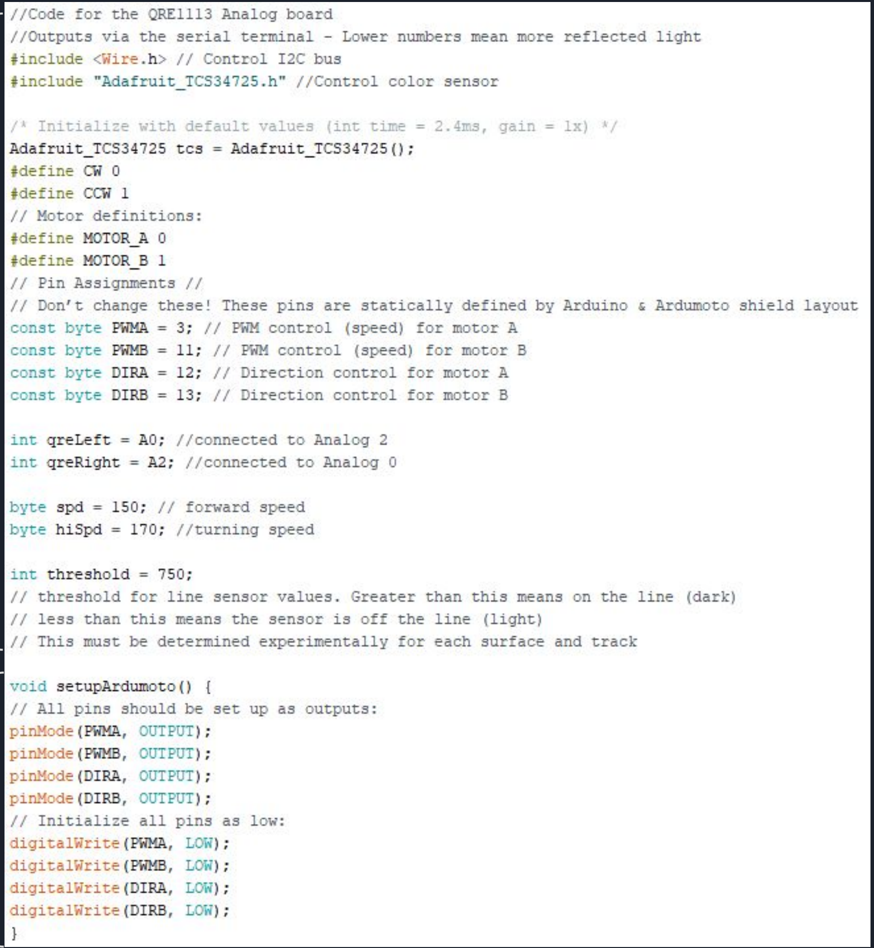
Schematics


TCS34725
RGB Sensor
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

Code

- Defined libraries and variable initializations.

- Motor A and Motor B setup as void function.

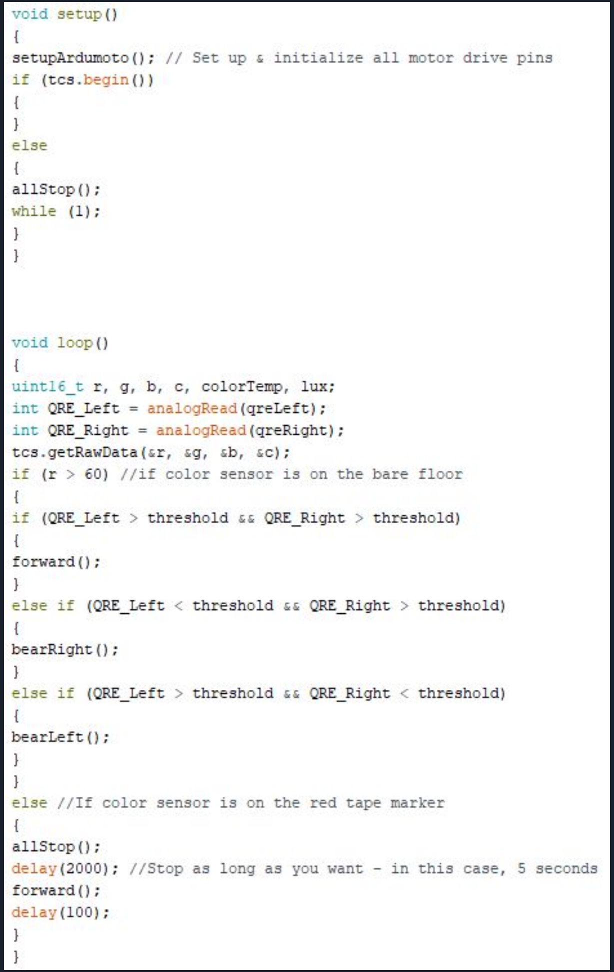


```
//Code for the QRE1113 Analog board
//Outputs via the serial terminal - Lower numbers mean more reflected light
#include <Wire.h> // Control I2C bus
#include "Adafruit_TCS34725.h" //Control color sensor

/* Initialize with default values (int time = 2.4ms, gain = 1x) */
Adafruit_TCS34725 tcs = Adafruit_TCS34725();
#define CW 0
#define CCW 1
// Motor definitions:
#define MOTOR_A 0
#define MOTOR_B 1
// Pin Assignments //
// Don't change these! These pins are statically defined by Arduino & Ardumoto shield layout
const byte PWMA = 3; // PWM control (speed) for motor A
const byte PWMB = 11; // PWM control (speed) for motor B
const byte DIRA = 12; // Direction control for motor A
const byte DIRB = 13; // Direction control for motor B

int qreLeft = A0; //connected to Analog 2
int qreRight = A2; //connected to Analog 0

byte spd = 150; // forward speed
byte hiSpd = 170; //turning speed

int threshold = 750;
// threshold for line sensor values. Greater than this means on the line (dark)
// less than this means the sensor is off the line (light)
// This must be determined experimentally for each surface and track

void setupArdumoto() {
// All pins should be set up as outputs:
pinMode(PWMA, OUTPUT);
pinMode(PWMB, OUTPUT);
pinMode(DIRA, OUTPUT);
pinMode(DIRB, OUTPUT);
// Initialize all pins as low:
digitalWrite(PWMA, LOW);
digitalWrite(PWMB, LOW);
digitalWrite(DIRA, LOW);
digitalWrite(DIRB, LOW);
}
```

Code

- Setup which initializes motor pins and code.

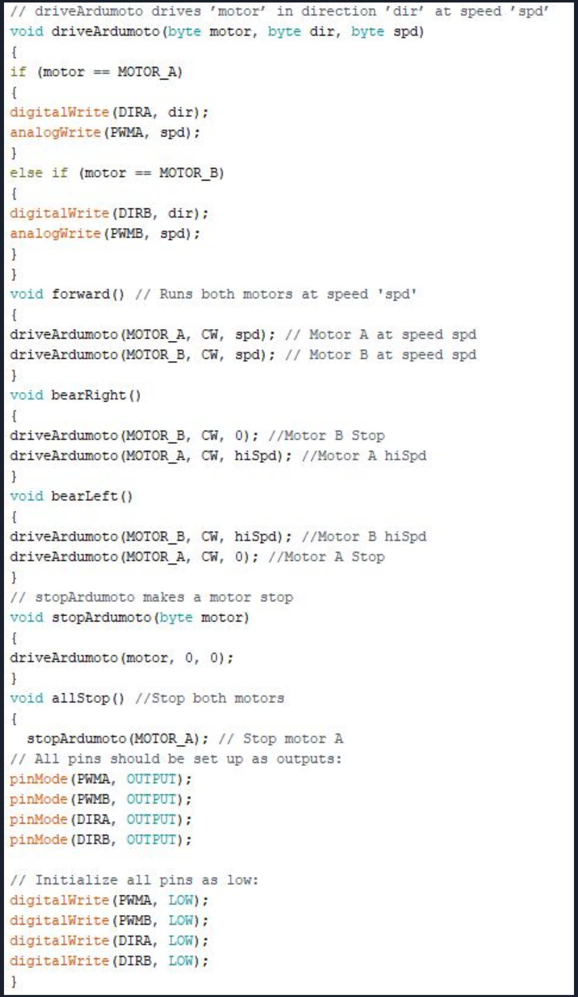
- Looped code to make robot follow black line and stop on thin black line.



```
void setup()
{
setupArdumoto(); // Set up & initialize all motor drive pins
if (tcs.begin())
{
}
else
{
allStop();
while (1);
}
}


void loop()
{
uint16_t r, g, b, c, colorTemp, lux;
int QRE_Left = analogRead(qreLeft);
int QRE_Right = analogRead(qreRight);
tcs.getRawData(&r, &g, &b, &c);
if (r > 60) //if color sensor is on the bare floor
{
if (QRE_Left > threshold && QRE_Right > threshold)
{
forward();
}
else if (QRE_Left < threshold && QRE_Right > threshold)
{
bearRight();
}
else if (QRE_Left > threshold && QRE_Right < threshold)
{
bearLeft();
}
}
else //If color sensor is on the red tape marker
{
allStop();
delay(2000); //Stop as long as you want - in this case, 5 seconds
forward();
delay(100);
}
}
```

Code

- Void function to set speed and direction of Motor A and Motor B.
- Void functions to steer robot forward, right, or left.
- Void functions to stop robot car.



```
// driveArdumoto drives 'motor' in direction 'dir' at speed 'spd'
void driveArdumoto(byte motor, byte dir, byte spd)
{
if (motor == MOTOR_A)
{
digitalWrite(DIRA, dir);
analogWrite(PWMA, spd);
}
else if (motor == MOTOR_B)
{
digitalWrite(DIRB, dir);
analogWrite(PWMB, spd);
}
}
void forward() // Runs both motors at speed 'spd'
{
driveArdumoto(MOTOR_A, CW, spd); // Motor A at speed spd
driveArdumoto(MOTOR_B, CW, spd); // Motor B at speed spd
}
void bearRight()
{
driveArdumoto(MOTOR_B, CW, 0); //Motor B Stop
driveArdumoto(MOTOR_A, CW, hiSpd); //Motor A hiSpd
}
void bearLeft()
{
driveArdumoto(MOTOR_B, CW, hiSpd); //Motor B hiSpd
driveArdumoto(MOTOR_A, CW, 0); //Motor A Stop
}
// stopArdumoto makes a motor stop
void stopArdumoto(byte motor)
{
driveArdumoto(motor, 0, 0);
}
void allStop() //Stop both motors
{
  stopArdumoto(MOTOR_A); // Stop motor A
// All pins should be set up as outputs:
pinMode(PWMA, OUTPUT);
pinMode(PWMB, OUTPUT);
pinMode(DIRA, OUTPUT);
pinMode(DIRB, OUTPUT);

// Initialize all pins as low:
digitalWrite(PWMA, LOW);
digitalWrite(PWMB, LOW);
digitalWrite(DIRA, LOW);
digitalWrite(DIRB, LOW);
}
```

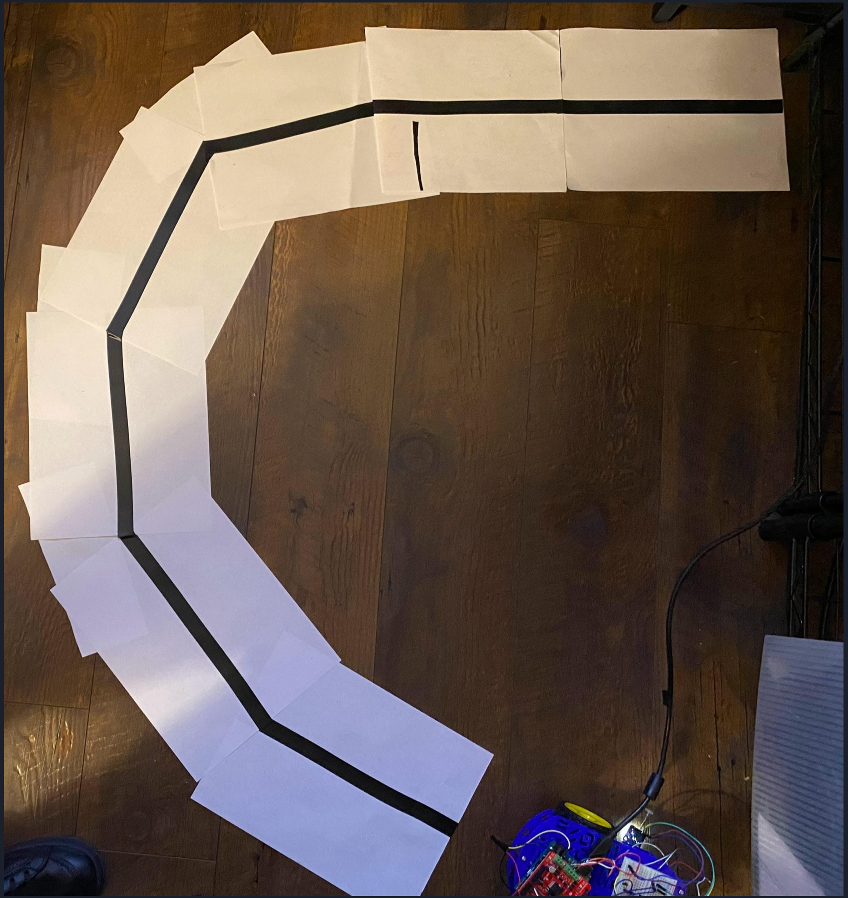
Troubleshooting

- Components such as the line sensor were very finicky
- Motor B was stronger than Motor A causing one wheel to spin faster than the other.
- Coding was a nightmare since each component was extremely fickle.
- Arduino would sometimes work properly sometimes not.
- Powering the robot was also difficult.

Most of these issues were resolved with trial and error. The arduino would still sometimes ignore the coding uploaded and the power required to run the motors on batteries alone didn’t seem to work.

Track Setup

- The robot car must follow the black line and stop at the small black line.



Video Demo

How it works

- The arduino runs the code that will move the car forward
- Periodically checking for the black line
    - If the line is no longer detected by the LEFT sensor the arduino will move the car a bit to the right
    - If the line is no longer detected by the RIGHT sensor the arduino will move the car a bit to the left
- When the car detects the thin black line with the color sensor, it will stop

Conclusion

- The robot sensors were difficult to get to work properly.
- The sensors did not want to properly pick up the right data and interpret it correctly
- The motors did not have enough power to fully move the car forward, we did manage to get it to move itself faster later.

- We currently have a few ideas to add to the final version.