Line Following Robot Project Documentation

Sonia Wong

December 2020

Contents

1 Build

1.1 Overall Build

All components are mounted to the acrylic chassis with a combination of foam, cardboard, hot glue, and hockey tape. Standoffs for the speaker are made from bolts with nuts that sandwich holes on the speaker.

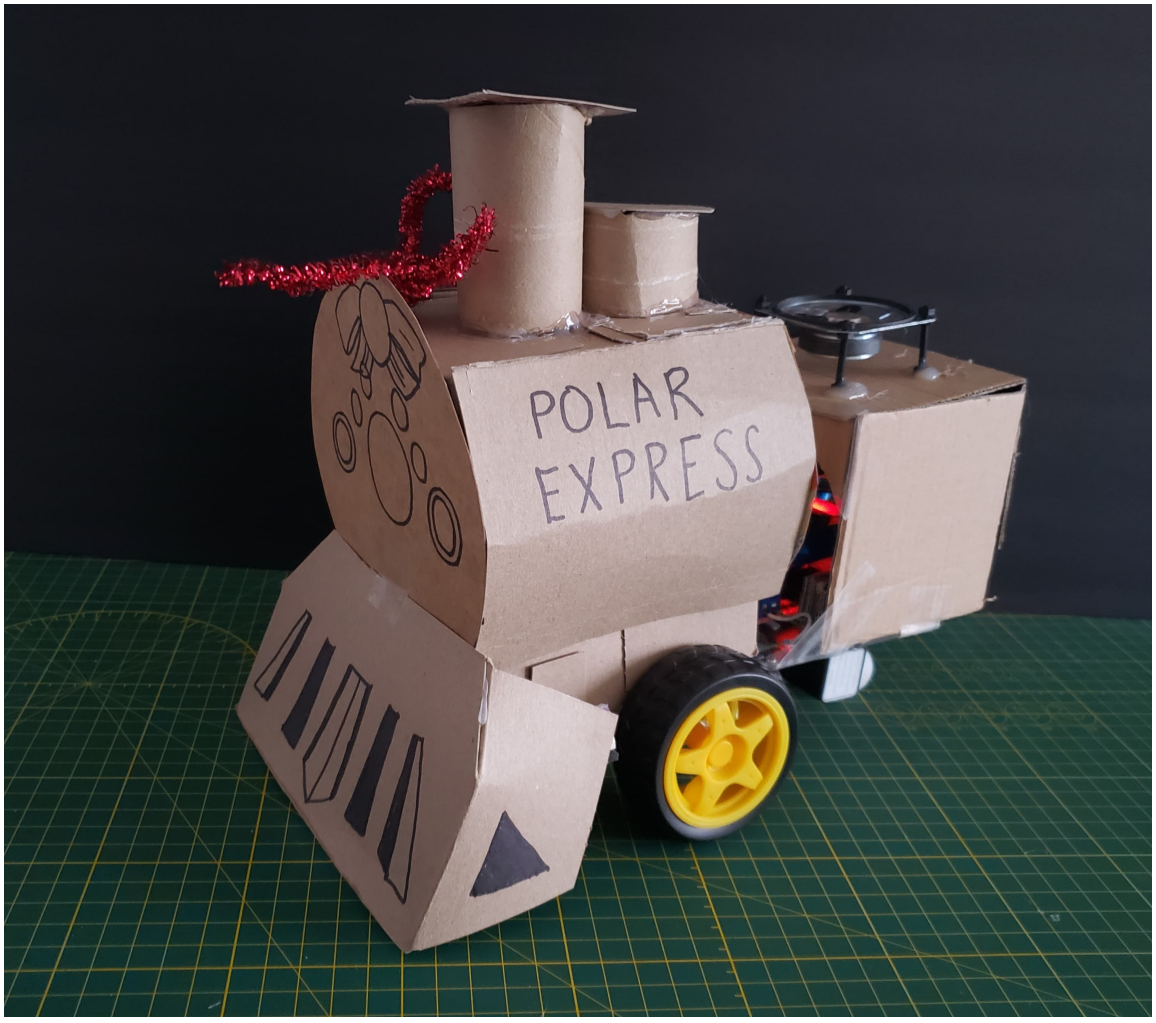

Figure 1: Final product.

1.2 Additional Features

The line follower is decorated as the Polar Express that plays "We Wish You A Merry Christmas" through a speaker as it follows the line (or track if you are feeling imaginative). A nice way to bring some Christmas cheer!

1.3 Sensor Bracket

An array of 5 sensors in a line, evenly spaced was chosen for the bracket design for ease of manufacturing and replication. The sensors are spaced around 7.5mm apart so that there is one sensor for the middle of the chassis, and two that border outside edge of the line. The outermost two were placed at the same distance interval, 7.5mm, for consistency. However, an improvement upon this design would be to place the outermost sensors further away to assist with sharp turns as this arrangement did not perform well on hard turns due to these limitations.

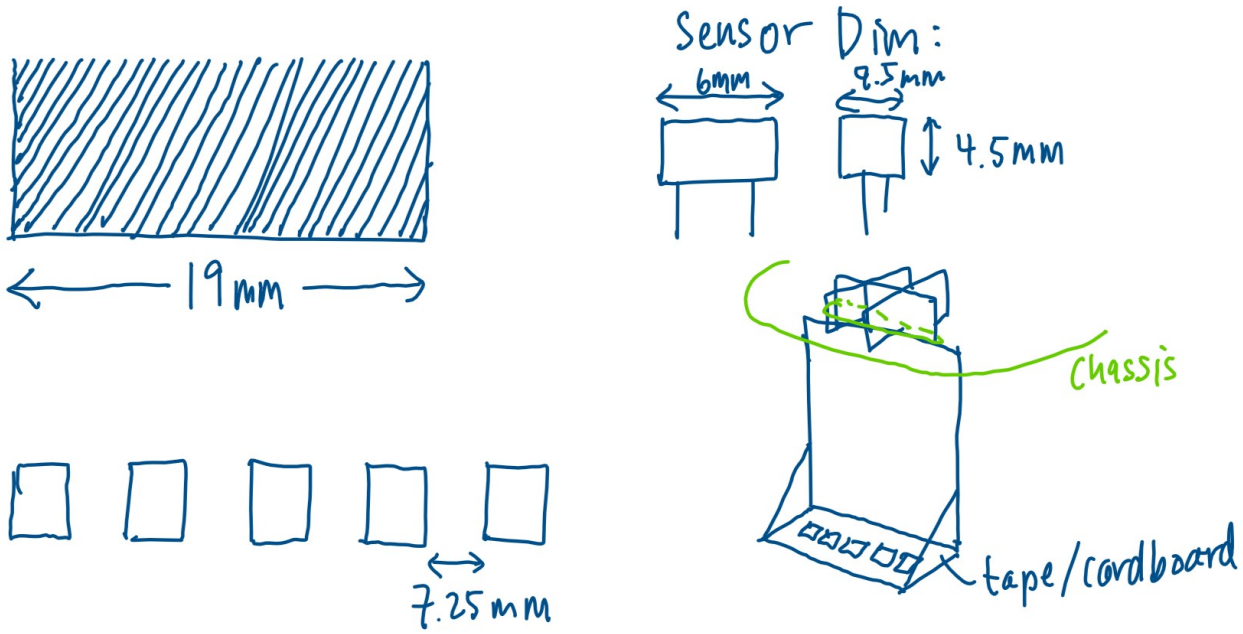

Figure 2: Bracket design sketch.

This bracket was designed to be made easily (was cut from a single sheet of cardboard and reinforced with hot glue). Adding a couple of reinforcing tabs made the bracket strong enough to rest the weight of the chassis on with no stiffness issues, when balanced on just the sensor bracket and caster wheel. Thin strips of cardboard were added bordering the sensors to help create a constant offset distance from the sensors to the paper, giving consistent readings. Finally, the sensors were fixed in place with hot glue to the cardboard bracket.

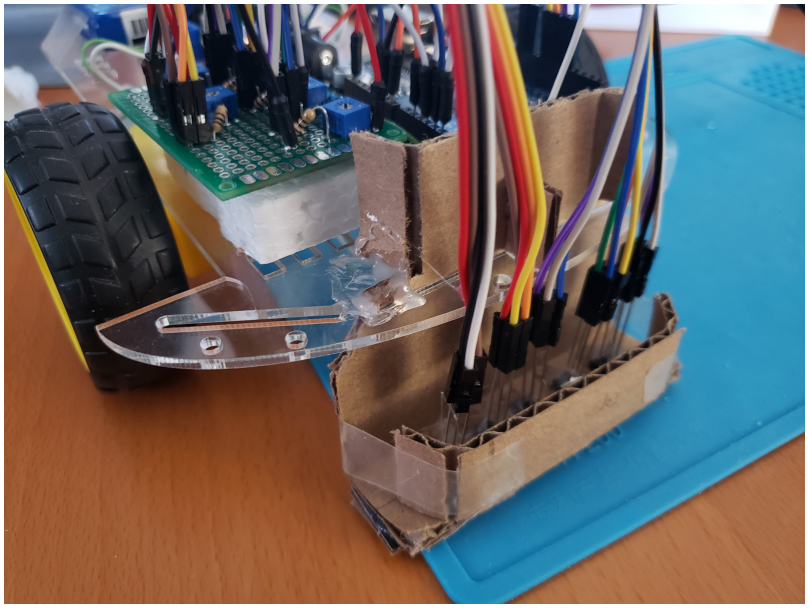

Figure 3: Bracket isometric view.

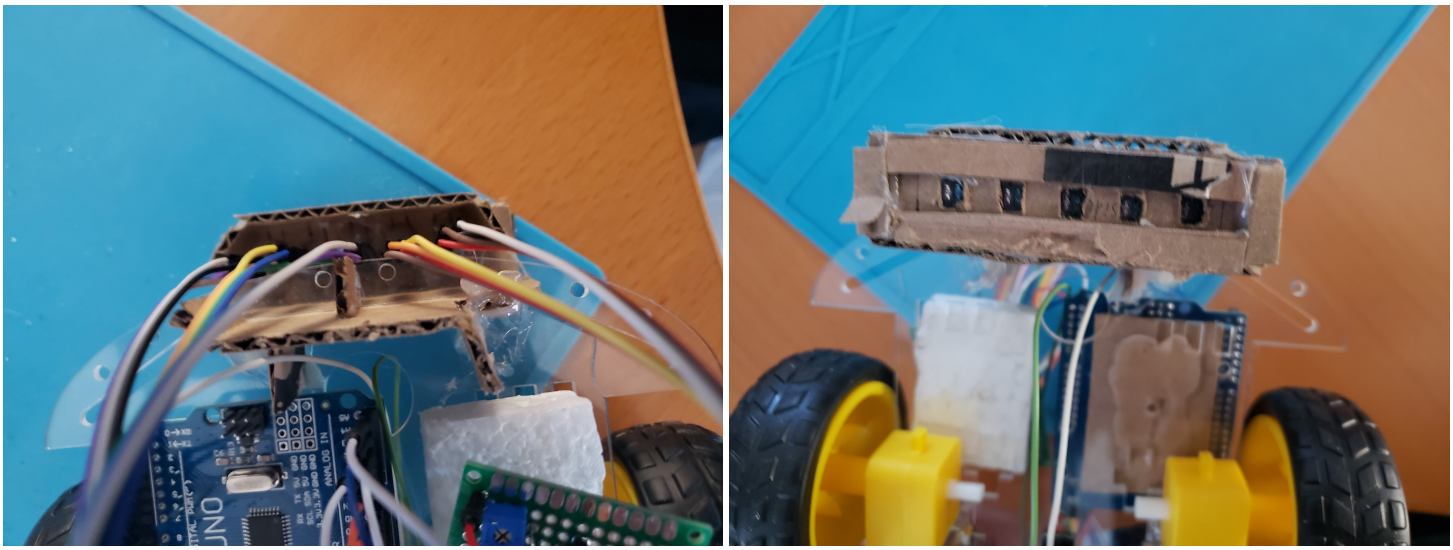

Figure 4: Bracket top and underside.

1.4 Wiring

Electrical components:

- 5x 5.1kΩ resistors, 5x 100Ω resistors
- 5x trimpots
- 1x protoboard
- Solid core wire
- 26x male-female jumper wires, 8x male-male jumper wires
- 2x DC motor
- 5x QRD1104 sensor

- 1x Arduino Uno
- 1x L298N
- 1x 2s LiPo, Voltage Checker

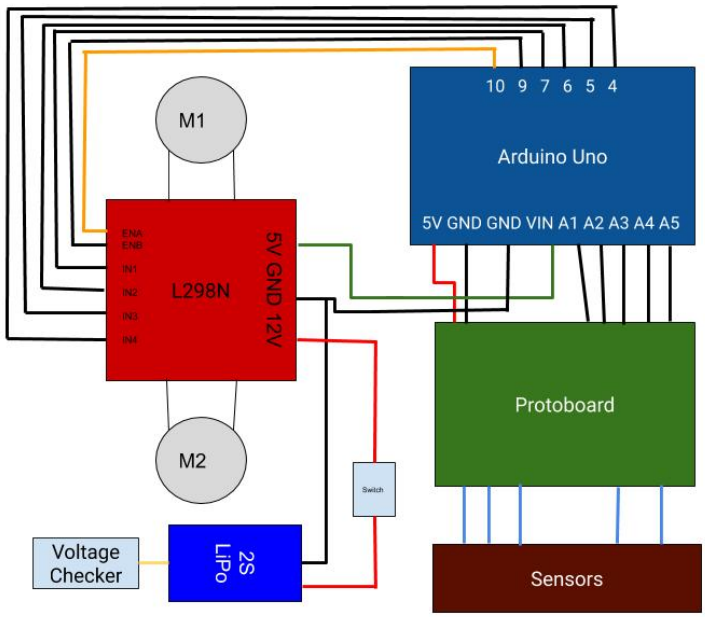

Figure 5: Simplified Block Wiring Diagram.

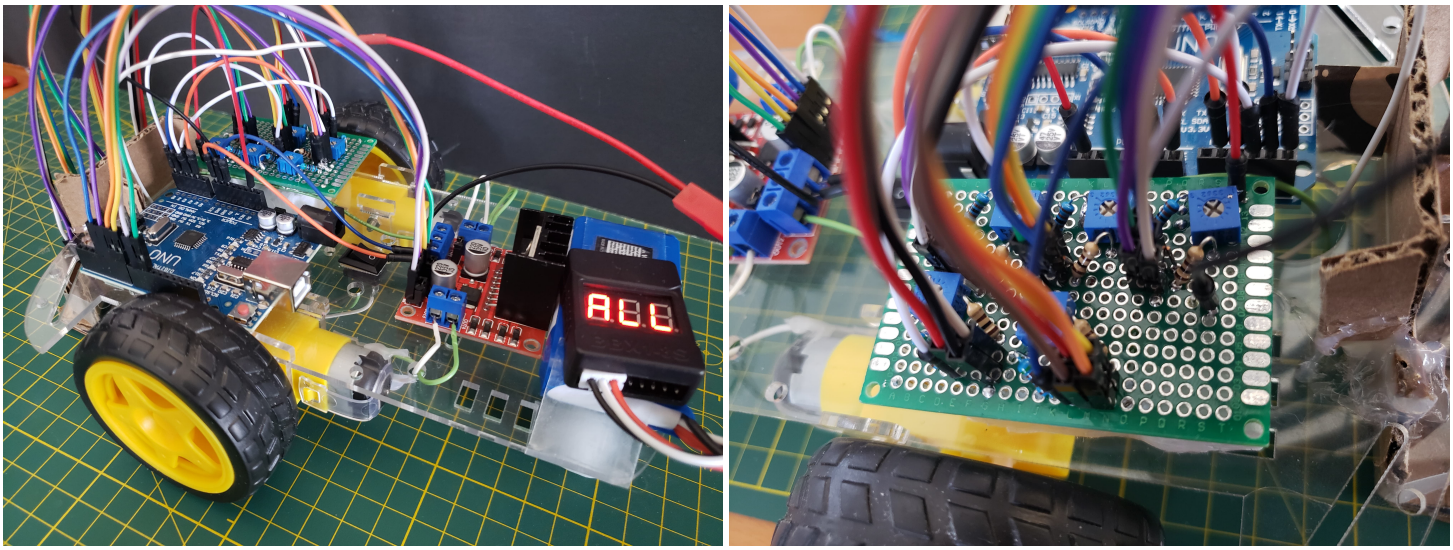

Figure 6: Modules fixed to the chassis.

The wiring was split into three modules: an arduino, the L298N motor driver, and a soldered protoboard sensor shield, each interfaced with as few jumper wires as possible. The sensor shield was soldered to reduce the amount of jumper wires, with jumpers connecting to the actual sensors in case a sensor failed, needing to be easily swapped without desoldering components. The sensor shield interfaces with the arduino through jumpers, so that the shield could be used with other boards if needed. Solid core wire and solder bridges on the underside of the board create the connections between the components. The images above shows how the components were physically arranged.

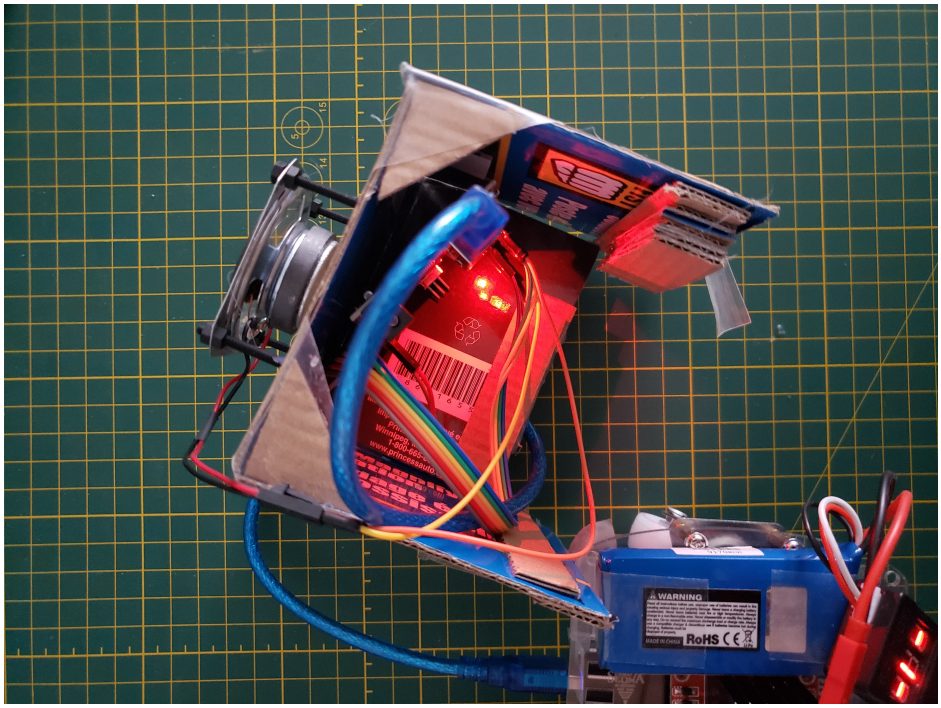

Figure 7: Wiring of the speaker and SD card.

For the speakers I had initially wired the sd card reader and speaker to the same arduino, however this required me to change the motor driver output pins to 5 and 6 which operate at 980Hz. This made the arduino much louder when giving pwm outputs (louder than the speaker) and caused a bit of noise/interference with the speaker, so I opted to separate the power supply and speaker/SD electronics to a different board. This worked much better as the small LiPo battery I am using, nominally rated for 7.4V, was struggling to give the motors the power needed with the power draw of the rest of the electronics.

2 Code and Implementation

2.1 Motors and Movement

```
int enA = 10;
int in1 = 7;
int in2 = 6;
int in3 = 5;
int in4 = 4;
int enB = 9;
int farLeft = 0, left = 0, center = 0, right = 0, farRight = 0;

void setup() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

//Move forward, with specified speed of left and right motors as int from 0-255
//turnDirection = 0 for left pivot, 1 for right pivot, 2 for full forward
void drive(int turnDirection, int leftMotor, int rightMotor) {
  switch(turnDirection) {
    case 0:
      digitalWrite(in1, HIGH);
      digitalWrite(in2, LOW);
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
      break;
    case 1:
      digitalWrite(in1, LOW);
      digitalWrite(in2, HIGH);
      digitalWrite(in3, HIGH);
      digitalWrite(in4, LOW);
      break;
    case 2:
      digitalWrite(in1, HIGH);
      digitalWrite(in2, LOW);
      digitalWrite(in3, HIGH);
      digitalWrite(in4, LOW);
      break;
  }
  analogWrite(enA, rightMotor);
  analogWrite(enB, leftMotor);
}

void loop() {
  farLeft = analogRead(A1);
  left = analogRead(A2);
  center = analogRead(A5);
  right = analogRead(A4);
```

```
50    farRight = analogRead(A3);
51
52    if((center>100 && left<100 && right<100) || (farLeft>100 && farRight>100)){
53      drive(2,70,70);
54    } else if( center>100 && left>100 && right<100 && farRight<100 && farLeft<100) {
55      drive(2,0,100);
56    } else if(center>100 && right>100 && left<100 && farRight<100 && farLeft<100) {
57      drive(2,100,0);
58    } else if(farLeft>100 && left>100 && center>100 && right<100 && farRight<100) {
59          //90 degree pivot turn left when center to far left sensors see a line
60      for(int i=0; i < 30; i++) {
61        drive(0,50,100);
62      }
63    } else if(farRight>100 && right>100 && center>100 && left<100 && farLeft<100) {
64          //90 degree pivot turn right when center to far right sensors see a line
65      for(int i=0; i < 30; i++) {
66        drive(1,100,50);
67      }
68    } else if((center<100 && left<100 && right>100 && farRight<100 && farLeft<100) ||
69   (farRight>100 && right>100 && center<100)) {
70      drive(2,115,0);
71    } else if((center<100 && left>100 && right<100 && farRight<100 && farLeft<100) ||
72   (farLeft>100 && left>100 && center<100)) {
73      drive(2,0,115);
74    } else {
75      drive(2,70,70);
76    }
77  }
```

2.2 Speaker and SD

```
1  #include <pcmConfig.h>
2  #include <pcmRF.h>
3  #include <TMRpcm.h>
4  #include "SD.h"
5  #include "SPI.h"
6  #define SDchipselect 2
7
8  TMRpcm tmrpcm;
9
10 void setup() {
11   tmrpcm.speakerPin = 10;
12   if(!SD.begin(SDchipselect)) {
13     return;
14   }
15   delay(5000);
16   tmrpcm.setVolume(5);
17   tmrpcm.play("merry.wav");
18 }
19
20 void loop() {}
```