Robotic Car Navigation

Scott Lee 204 955 724
Jennie Zheng 304 806 663

Introduction

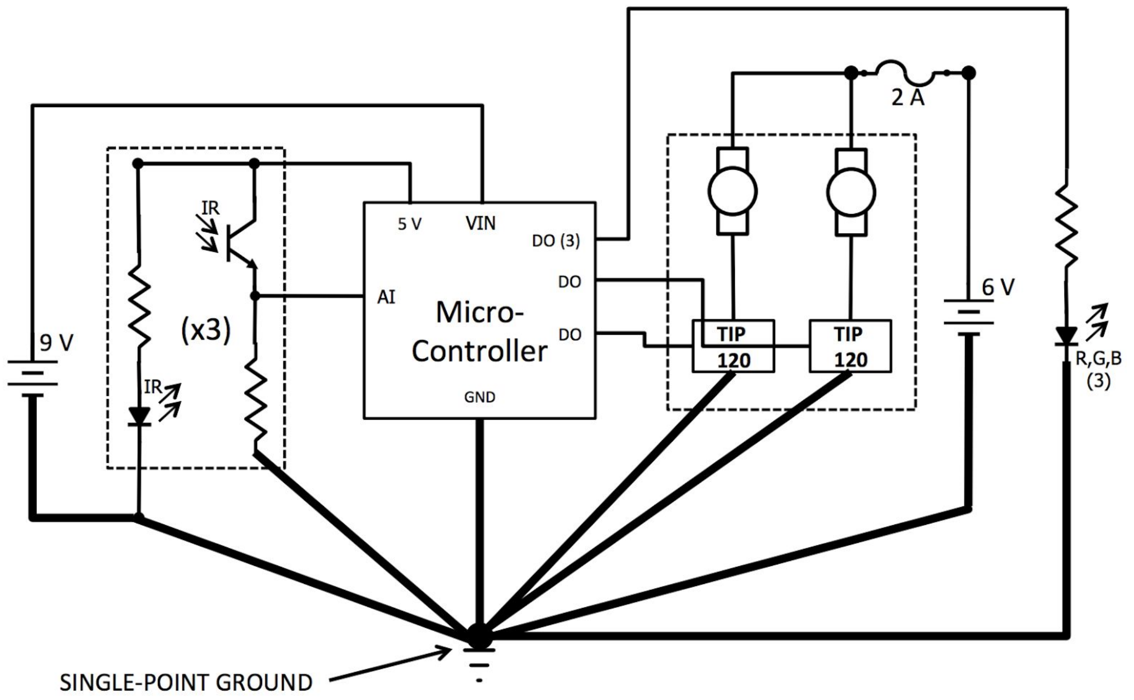
- Goal and Design

The motivation behind the project is to understand and apply the fundamental concepts of electrical engineering and computer science. Furthermore, the project goal is to build a robotic car which can follow a path outlined with black tape. We were supplied with an Arduino Nano microcontroller, infrared LEDs and sensors, regular LEDs, direct current motors, breadboards, resistors, wires, batteries. We used the infrared LEDS and sensors to figure out where the car was relative to the track, and we used proportional integral derivative controller logic to control the car.

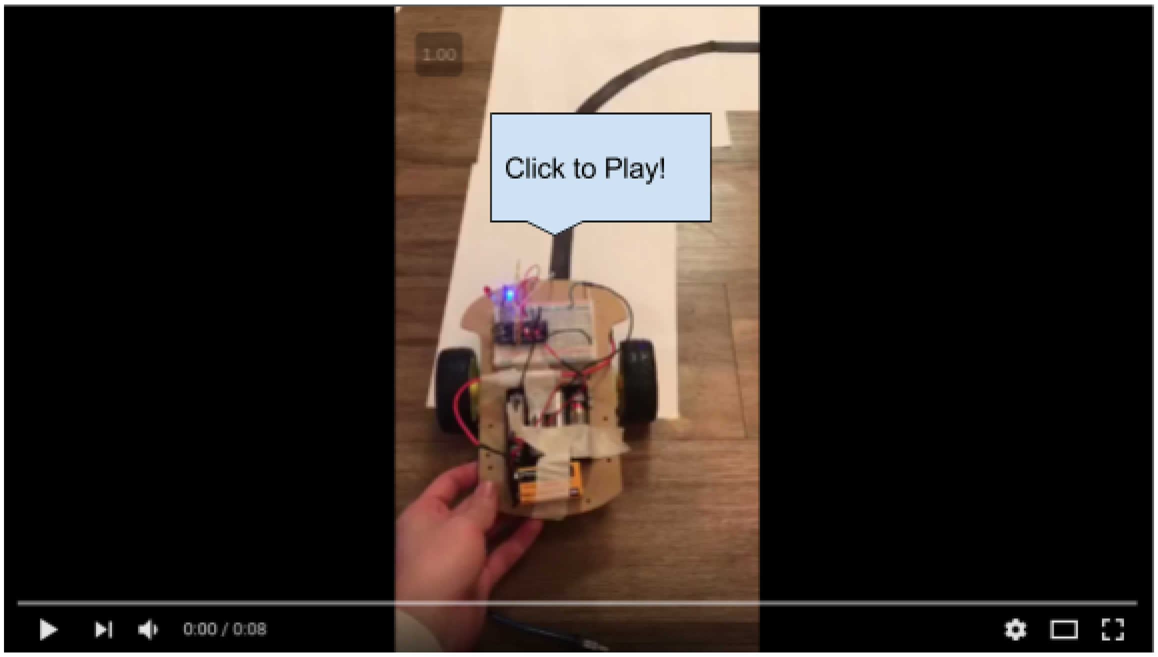
The overall design:

The System

- Block Diagram



(Project EE3 Fall 17 - Block Diagram of the System)

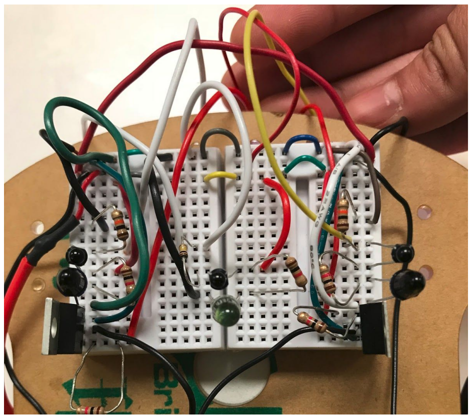


(URL: https://www.youtube.com/watch?v=lRObLrlhMAU&feature=youtu.be)

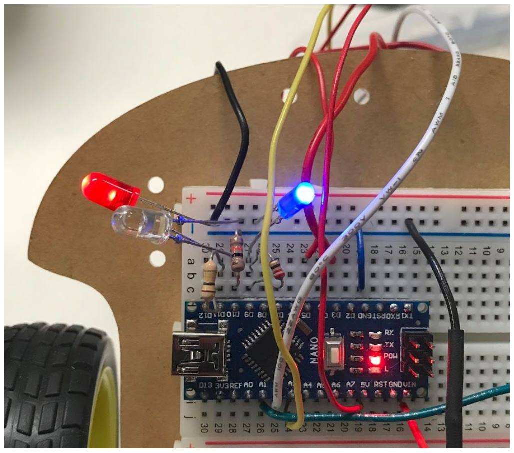
- Theory

  To understand the project, we need to understand infrared sensors/LEDs, Arduino, and PID.

1. Infrared sensors and LEDs – Each infrared LED sent light to its respective infrared sensor. However, the infrared LED sends less light to the sensor when near the light-absorbing black tape, because black absorbs light more than white. Thus, the infrared LED/sensor pairs effectively detected how close they were to the black line. These LED pairs should be placed on the left and right side of the car to figure out whether the black line is closer to the left or to the right of the car. A picture of our final infrared LED and sensor configuration is below. Note that we accidentally switched the order of the LED and sensor. The LED should be on top, before the sensor.



2. Arduino – The Arduino microcontroller was programmed to control the car through its analog and digital read/write pins. [1] Several digital pins controlled the green, blue, and red LED lights by specifying whether they should be on or off, while other analog pins read the values of the infrared sensors and wrote the values of the left and right motors. Data was written to the Arduino microcontroller using the Arduino IDE. A picture of our final Arduino nano wiring is below.



3. PID - The car's turning logic utilized a simplified version of PID. The PID is a controller that uses negative feedback from the infrared sensor to adjust the speed of the left and right motors and keep the car on the track. Our controller found the error, or how far the car was from its ideal position on the track, and then incremented the right wheel speed and decremented the left wheel speed by a constant proportion of the error. This kept the car very stable on the track. The main PID code is below.

```
//reading sensor data
int  val_r = analogRead(IR_r_Pin);
int  val_l = analogRead(IR_l_Pin);

//applying simplified PID logic
double errorIR = val_r - val_l;
double KpIR = .12;
double correction = KpIR * error;

//cap on speed
  if ((leftSpeed + correction) < 60){
    // turn left
    leftSpeed = 80;
    rightSpeed = 115;
  }
//other cap on speed
  else if ((rightSpeed - correction) < 60){
    // turn right
    rightSpeed = 80;
    leftSpeed = 115;
  }
//default behavior speed
  else{
    rightSpeed = rightSpeed + correction;
    leftSpeed = leftSpeed - correction;
  }
```

Testing Methodology : Path Sensing Subsystem

<u>How We Designed the Test</u>

We configured the infrared sensors and LEDs in four different ways and then measured the values sensed by the left, center, and right infrared sensors at the left, center, and right of two different tracks to find the ideal detection system for the car.
Configurations:

1. Each LED a centimeter away from its respective sensor, 100K resistor in sensor circuit
2. Each LED touching its respective sensor, 100K resistor in sensor circuit
3. Each LED a centimeter away from its respective sensor, 10K resistor in sensor circuit
4. Each LED touching its respective sensor, 10K resistor in sensor circuit

Tracks:

1. Clean track on single piece of paper
2. Dirty track on large poster paper

<u>How We Conducted the Test</u>

We wrote the following code into our Arduino:

```
//reading sensor data
  val_r = analogRead(IR_r_Pin);
  val_l = analogRead(IR_l_Pin);
  val_c = analogRead(IR_c_Pin);

//printing sensor data
  Serial.print("Right Sensor: ");
  Serial.print(val_r);
  Serial.print(", Left Sensor: ");
  Serial.print(val_l);
  Serial.print(", Center Sensor: ");
  Serial.println(val_c);
```

Then, we monitored the Arduino’s print statements for each configuration, track, and car position on the track.

How We Analyzed the Test Data

We analyzed the data using tables. Below are our results for the four different configurations, on a clean track.

LED Centimeter From Sensor, 10K Resistor

	Left Sensor	Center Sensor	Right Sensor
Line to Left of Car	315	961	971
Line in Center of Car	992	262	961
Line to Right of Car	336	310	363
Stop Line	294	263	341

LED Touching Sensor, 100K Resistor

	Left Sensor	Center Sensor	Right Sensor
Line to Left of Car	945	984	983
Line in Center of Car	993	910	923
Line to Right of Car	954	932	981
Stop Line	939	927	925

LED Centimeter from Sensor, 10K Resistor

	Left Sensor	Center Sensor	Right Sensor
Line to Left of Car	201	535	562
Line in Center of Car	513	324	351
Line to Right of Car	481	521	231
Stop Line	159	213	124

LED Touching Sensor, 10K Resistor

	Left Sensor	Center Sensor	Right Sensor
Line to Left of Car	201	744	736
Line in Center of Car	739	586	570
Line to Right of Car	738	738	155
Stop Line	159	477	124

How We interpreted the Data

We were looking for a configuration which stably detected whether the car was at the left, right, or center of the track. We were searching for a configuration in which the car could flexibly detect whether the black line was at the center, center right, right, center left, or left.

Results and Discussion

Test Discussion

First and foremost, we found out that our sensors in any configuration worked significantly better on a clean track than on a dirty track full of footprints. This difference was severe enough that on race day our car it consistently failed the dirty test track but succeeded in the clean final track.

Furthermore, we realized from our tests that the 100K resistors did not work in the sensor circuit. This is because the 100K resistors made the infrared sensors too sensitive to infrared light. That's why the the 100K resistor with LED touching its sensor reached the max value of around 1000 in all cases, even when it was right above the black line. Furthermore, the 100K resistor with the LED a centimeter away from its sensor was not much better, because it reached the max value of 1000 in almost all cases, unless it was absolutely perfect aligned with the black line. Thus, we switched the 100K resistor for the 10K resistor.

Lastly, we found that the 10K resistor configuration was more sensitive to changes when the the LED was very close or touching the sensor, so we decided to use the 10K and touching configuration.

As a side note, we realized the morning of race day that our infrared LEDs were supposed to be in front of our LED sensors. Instead, our car had our LEDs behind our sensors, which made it less effective our car less effective at detecting the how the path was changing in front of the car. This is likely the reason why our car's sensor system was less effective than that of most other teams. We did not have time to switch the circuit, but noted to ourselves that in future path navigation projects, we should reach the directions more carefully and then place our LEDs in front of our sensors.

Race Day Results

On race day, our car tested the track two times. The first time, our car randomly swerved off course in the middle of the second straight part of the track. We weren't sure of the cause of this strange failure, but noted that another team's car also malfunctioned at the exact same place as ours. We suspect that the large footprint at the side of that particular part of the track was a severe obstacle for the car. After our first trial, we adjusted the proportion error of the car from .16 to .12. This made the car less sensitive to small errors. Then, we immediately tested the car again and was able to successfully clear the path and then stop. Although our car was not fast (around 35 seconds), it ran very smoothly due to our well adjusted PID logic.

Conclusions and Future Work

The project successfully taught us to understand and apply the fundamental concepts of electrical engineering and computer science. We learned about the fundamentals of path navigation with the help of infrared LEDs and sensors, motors, as well as an Arduino microcontroller. If we had more time, we would switch our infrared LED and sensors so that our LEDs were in front of our sensors and better detected the path in front of it. Furthermore, we would speed up our car so that it cleared the track faster.

In addition, while working on this project, we wanted the car to be controlled remotely by us. Therefore, we planned to make more advance IR sensing car. The car is controlled by a typical TV remote controller. For example, the channel-up button is for go-straight command; volume-down button is for go-right command; etc. Then, there must be a IR receiver that detects the IR signals from the remote controller. Based on the signals, we would adjust the wheel speed.

References

[1] Arduino Reference - https://www.arduino.cc/reference/en/