Arduino Line Follower Full Guide

This guide explains how to build an Arduino line-following car. Read through it to get a clear **understanding of the process**, as I've also included the challenges I faced and how I solved them. By the end, you'll have a working car and useful insights for overcoming **common issues.**

මෙම Document එකෙන් Arduino භාවිතා කරලා Line Follower කාර් එකක් හදන හැටි පැහැදිලි කරලා තියෙනවා. Line Follower එකක් හදන විට නිතරම මුහුණ දෙන්න වෙන සමාන්‍යය **ගැටළු කිහිපයක් සහා අදාල සියලුම Components ගැන විස්තරයක්** මේ Document එක සම්පුර්ණයෙන් කියවලා ලබාගන්න පුලුවන්.

Overview :

1. Components
   - Arduino Nano
   - TB6612FNG
   - Sensor Array
2. Circuit Diagrams
   - Main Circuit
   - Buck Converter
3. Source Codes

Components

To build a line-following car with Arduino you'll need a car chassis, wheels, motors, sensors, motor drivers, batteries and an Arduino board.

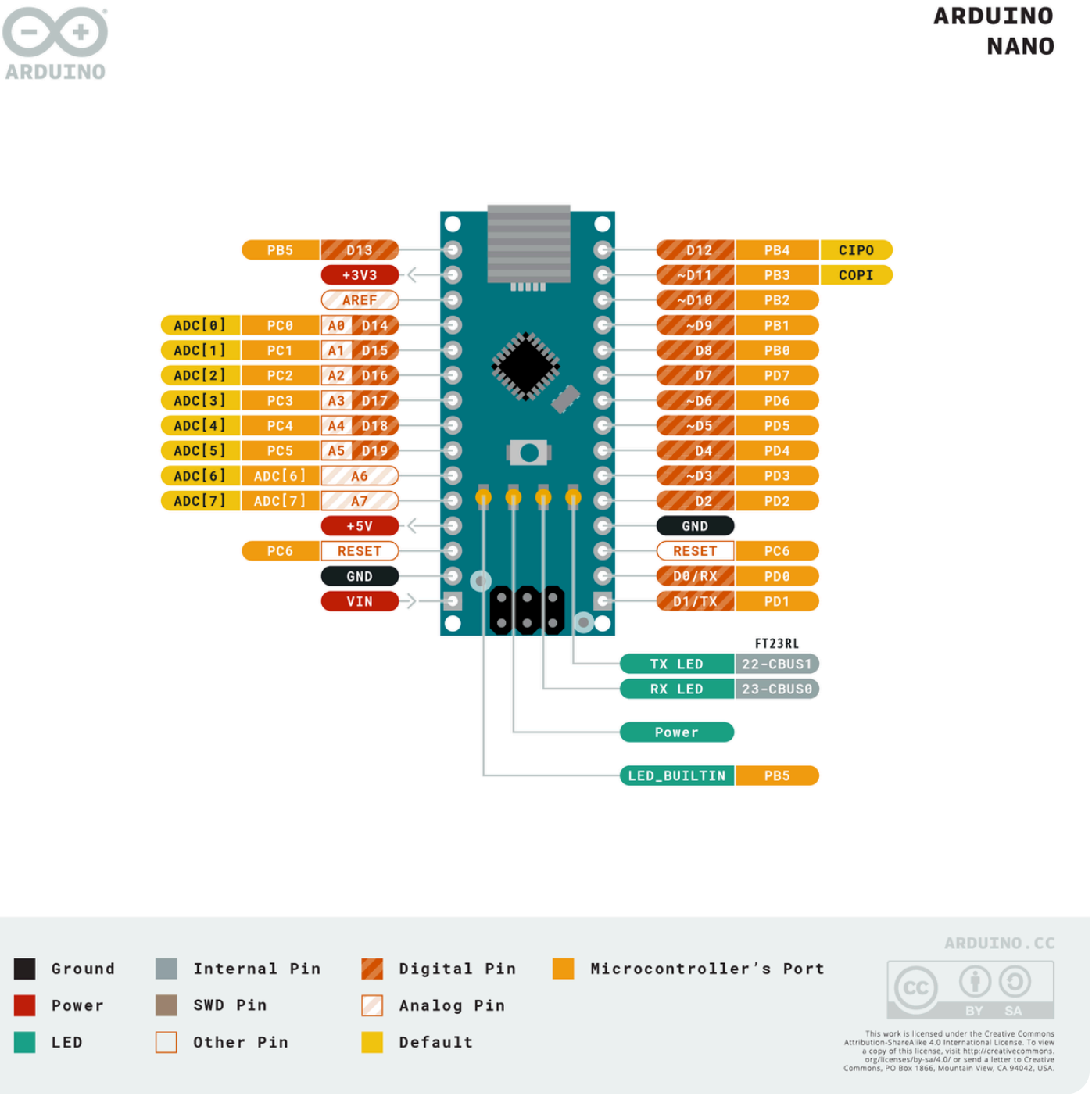
There are **several options** are available for each component, and the code and diagrams are generally the same regardless of your selection. Let's go over some key components.

Line follower එකක් හදන්න සමාන්‍යයෙන් අපිට ඕන වෙන්නෙ car chassis එකක්, wheels, motors, sensors, motor drivers, batteries සහා Arduino board එකක්. මේ හැම component එකකටම වගේ **විවිධ options** තියෙනවා. ඒවා නම්,

1. Arduino Nano / Arduino Uno / ESP32

2. TB6612FNG Motor Driver / L298N
3. 5 bit Sensor Array / 8 bit Sensor Array / 16 bit Sensor Array
4. 12v Li-po Battery / 3.7v Li-ion Battey x4
5. Buck Converter (12v to 5V)
6. 10k Variable Resistor (optional)
7. Mini Breadboard x2
8. Switch & Jumper Wires

Now there are some important things you need to understand with each of these components. let's discuss about those,



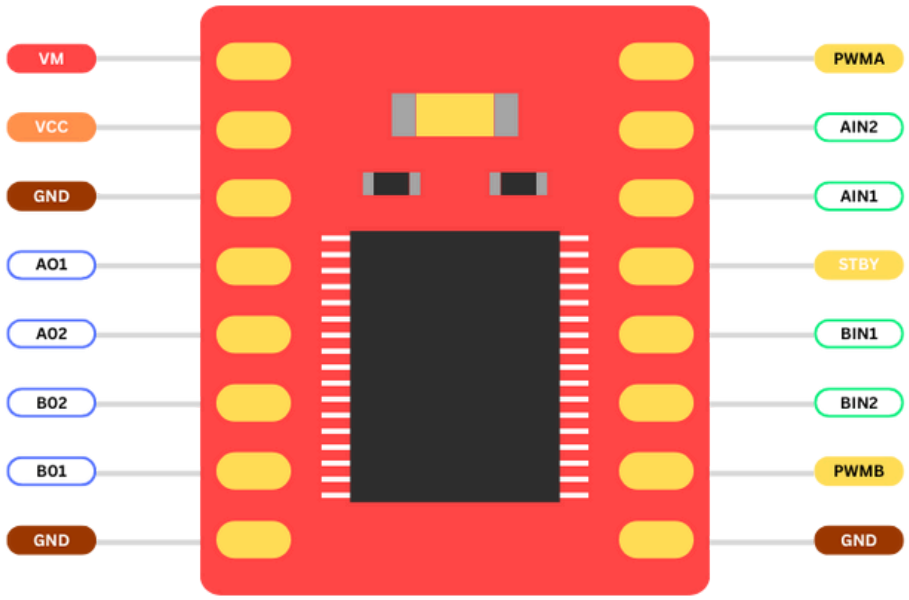
In the diagram, you'll notice a **(~)** symbol next to pins **D3, D5, D9, D10, and D11.** These represent **PWM** (Pulse Width Modulation) pins, which are crucial for controlling motor speed. PWM pins are a type of digital output pins on Arduino microcontrollers that allow you to **simulate analog outputs**.

ඉහත Diagram එකේ **D3, D5, D9, D10, D11** Pins ළඟ **(~)** සංකේතයක් දක්වලා තියෙනව බලාගන්න පුලුවන්. මේ සංකේතය යොදා ගන්නේ **PWM** pins නිරූපණය කරන්න. PWM (Pulse Width Modulation) වලින් අපිට පුලුවන් වෙනවා **analog output එකක් (0-255) simulate කරගන්න**. ඉතින් මේක අපිට Motor Speed එක වෙනස් කරන්න, පාලනය කරගන්න ගොඩක් වැදගත් වෙනවා. Motor driver එක connect කරද්දි අපි අනිවාරෙන්ම **speed controlling වලට අදාල pins, Arduino PWM pins වලටම connect කළ යුතුයි.**

Analog pins (A0-A7) collect data from the sensor array. If you need to use more than five sensors and gather analog values, you'll want to opt for an Arduino Nano (8 analog inputs) or Mega (16 analog inputs). Alternatively, you can use digital pins for sensor input and modify only one line of code, as we'll be using the **SparkFun QTR library** for sensors.

Analog pins (A0-A7) Sensor Array එකෙන් අරන් එන values read කරගන්න පවිච්චි කරනවා. අපිට මේ වෙනුවට **digital pins පාවිච්චි කරන්නත් පුලුවන්** හැබැයි එතකොට අපිට read කරන්න වෙන්නෙත් **digital values** මයි. **Analog readings ගන්න නම්** අපි පවිච්චි කරන sensor array එකත් analog outputs දෙන්න පුලුවන් එකක් වෙන්න ඕනෙ. කොහොම නමුත් අපි **Sparkfun QTR library** එක පවිච්චි කරලා කෝඩ් කරන නිසා අපිට coding වලදි, මොන outputs පාවිච්චි කළත් පහසුවෙන් codes change කරගන්න පුලුවන්. 5 ට වැඩි sensor array එකක් තියනවානම් සහා **analog values ගන්න අවශ්යනම්** Arduino Nano හෝ Mega එකක් පාවිච්චි කරන්න ඕන. Nano එකේ analog input pins 8ක් වගේම Mega එකේ 16 බලාගන්න පුලුවන්.

Next TB6612FNG Motor Driver,



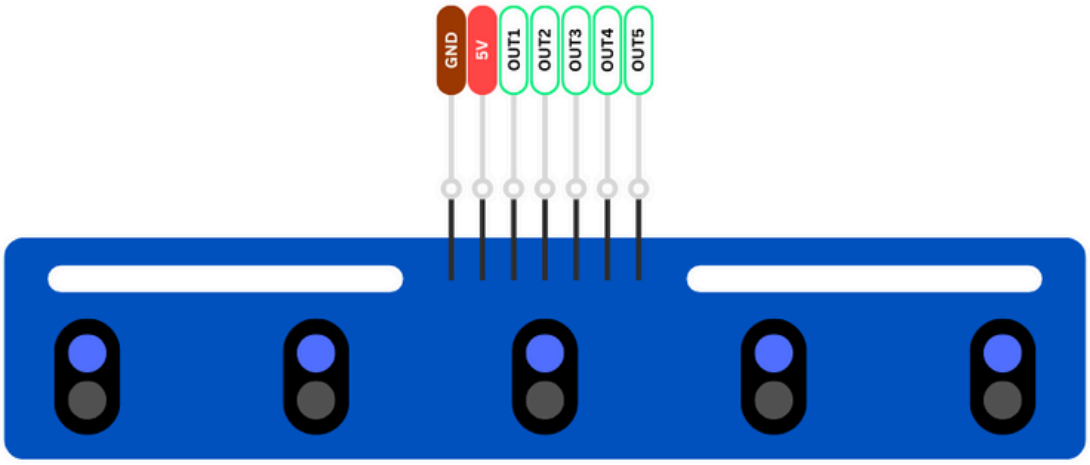
For the **TB6612FNG Motor Driver**, the **PWMA** and **PWMB** pins should connect to the PWM pins on the Arduino to control motor speeds. Pins **AIN1, AIN2, BIN1,** and **BIN2** control motor direction, while **AO1, AO2, B01,** and **B02** connect to the motors.

TB6612FNG Motor Driver එකේ PWMA & PWMB, Arduino board එකේ **PWM pins වලට අනිවාරෙන්ම connect වෙන්න ඕනෙ** මෝකද අපි Motor එකේ speed එකට 0-255 අතර value එකක් use කරන නිසා. එහෙම speeds control කරන්න නම් පිට සාමන්‍යය digital pin එකකින් high & low values දීලා හරියන්නේ නැහැ අපි ඒ වෙනුවට PWM pin එකකින් analog value එකක්ම දෙන්න ඕනෙ. AIN1, AIN2, BIN1, BIN2 කියන්නේ motor direction එක control කරන pins (inputs). AO1, AO2, BO1, BO2 pins connect වෙන්නේ අපේ motors වලට (outputs).

The module also has two voltage inputs: **VM** (to power the motors) and **VCC** (to power the module). For this guide, connect VM to a 7V power source (depending on the motors you are using, which can range between 2.5V and 13.5V) and VCC to a 5V source.

VM එක motors වලට අවශ්‍යය power එක දෙන්න use කරන අතර **VCC එක module එකට** අවශ්‍යය power එක දෙන්න use කරනවා.

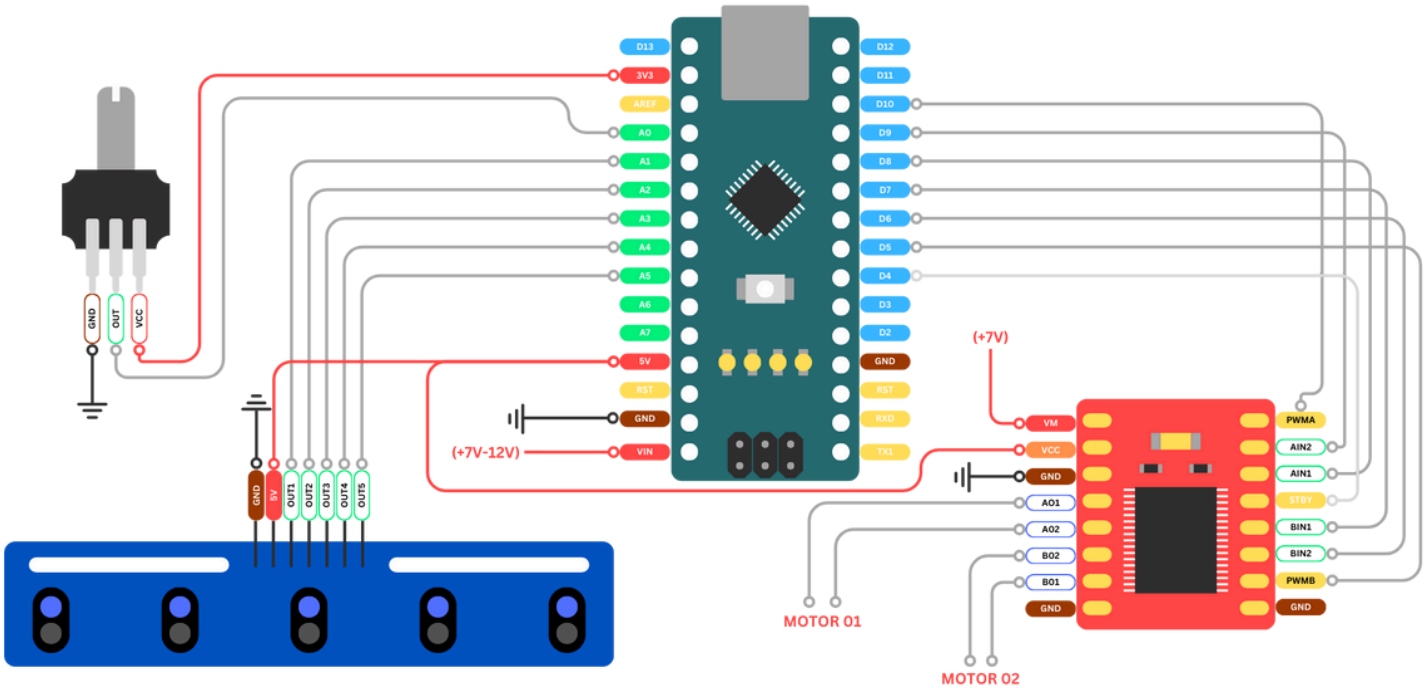
Next sensor array,



I recommend using the **QTR 8-bit Sensor Array** for optimal performance, though there are other options available. For best results, the sensor array should be placed approximately 3mm above the surface, perfectly leveled. So to avoid those practical errors, we'll use the **TCRT5000L 5-bit Sensor Array** In this guide.

Sensor array එකට තියෙන හොදම option එකක් තමයි **QTR 8bit sensor array** එක. නමුත් මේක use කරද්දි මේ module එක අපි පොළොව මට්ටමට ගොඩක් ළඟින් module එකත් පොළොවත් අතර ගොඩක් අඩු gap එකකින් (3mm) සහා පොළොව මට්ටමට හරියටම වගේ සමාන්තරව සවි කරගන්න ඕනෙ. මේ වගේ ප්‍රායෝගික ගටලු නිසා මේ example එකේදි අපි use කරන්නෙ **TCRT5000L 5bit sensor array** eka.

Circuit Diagrams



If your battery provides more than 7V (I recommend 12V), a **buck converter** will help reduce the voltage to 5V. This not only ensures stable voltage but also improves PID performance.

මෙතනදි **7v වලට වඩා** වැඩි voltage එකක් සහිත battery එකක් පාවිච්චි කරනවනම් buck converter කෙක් පාවිච්චි කරන්න ඕනෙ. **12V battery එකක් buck converters එක්ක පාවිච්චි කරන එක වඩාත් හොඳයි** මොකද buck converter එක මගින් ගොඩක් stable voltage එකක් ලැබෙන නිසා PID values පාවිච්චි කරද්දි battery එකේ සිදු වෙන discharge එකෙන් ඒ values වලට වෙන බලපෑම මගහැර ගන්න පුලුවන් වෙනවා.

Source Codes

The source code can be customized based on your hardware setup. If you're using an **Arduino Nano, TB6612FNG motor driver, and a 5-bit sensor array**, no major code changes are necessary. However, you will need to adjust the PID values to fine-tune your car's performance.

මේ codes ලියලා තියෙන්නෙ **Arduino nano, TB6612FNG, 5bit sensor array එකත් එක්ක ඉහත circuit එකට ගලපෙන විදිහට** නමුත් එහත සදහන් කරපු වෙනත් modules පාවිච්චි කළත් codes වල ලොකු වෙනස්කම් කරන්න අවශ්යය වෙන්නේ නෑ. codes වල add කරලා තියෙන **comments කියවීමෙන්** අවිශ්යය changes පහසුවෙන්ම කරගන්න පුලුවන්.

Start by adjusting the **Kp** value. The car should follow the line but may oscillate. If it does, then adjust the **Ki** and **Kd** values for smoother movement.

නමුත් බොහෝවිට **PID values ගැලපෙන විදිහට වෙනස් කරගන්න** සිදුවෙයි. PID values fine tune කරගද්දි **මුලින්ම Kp විතරක්** පාවිච්චි කරලා values adjust කරමින් test කරන්න ඕනෙ. එහෙම test කරලා car එක පොඩි oscillation එකක් එක්ක හරි **line eka follow කරන මට්ටමට** tune කරගත්තට පසුව **Ki** සහා **Kd** values adjust කරලා car එකේ movement එක ගොඩක් smooth කරගන්න පුලුවන්.

```
#include <QTRSensors.h>

#include <SparkFun_TB6612.h>


// Defines Motor Controller Pins

#define AIN1 8

#define BIN1 7

#define AIN2 9

#define BIN2 6

#define PWMA 10

#define PWMB 5

#define STBY 4


const int offsetA = 1;

const int offsetB = 1;


Motor motor1 = Motor(AIN1, AIN2, PWMA, offsetA, STBY);

Motor motor2 = Motor(BIN1, BIN2, PWMB, offsetB, STBY);


// Setting Up Sensor Array

QTRSensors qtr;

uint16_t position;

const uint8_t sensorCount = 5; // Number of Sensors Used (5, 8, 16)

uint16_t sensorValues[sensorCount];


int treshold = 500;


int L = 0;  // Left Motor

int R = 0;  // Right Motor

int error = 0;
```

```
int add = 0;


int P;

int I;

int D;

int lastError = 0;


float kp = 0.13;  // 0.0729

float ki = 0;    // 0.005

float kd = 0;    // 1


int speed = 225;

int maxSpeed = 255;

int minSpeed = 0;

int turnSpeed = 255;


int resistor;  // Variable Resistor

int z;


int b_in;  // Bluetooth Module



void setup() {


 Serial.begin(9600);


 qtr.setTypeAnalog(); // qtr.setTypeRC for Digital Inputs

  qtr.setSensorPins((const uint8_t[]){ A5, A4, A3, A2, A1 }, sensorCount); // Input Pins for Sensors

 // qtr.setEmitterPin(13);
```

```
  // Callibrating Sensors...
  for (uint16_t i = 0; i < 200; i++) {
    qtr.calibrate();
    Serial.println("calibrating...");
  }
}

void loop() {

  // b_control();
  pid();    // PID and Controlling Motor Speeds
  print();  // Serial Printing Values
}

void pid() {
  resistor = map(analogRead(A0), 0, 1023, 0, 255);
   position  =  qtr.readLineWhite(sensorValues);   // qtr.readLineBlack(sensorValues for Black Line
  error = 2000 - position;  // 3500 - position for 8bit Sensor Array

  P = error;
  I = I + error;
  D = error - lastError;
  lastError = error;

  add = P * kp + I * ki + D * kd;  // Adds To Initial Motor Speeds

  L = speed + add;  // New Speed Of Left Motor
  R = speed - add;  // New Speed Of Right Motor
```

```
  if (L > maxSpeed) {
    L = maxSpeed;
  }
  if (R > maxSpeed) {
    R = maxSpeed;
  }
  if (L < minSpeed) {
    L = minSpeed;
  }
  if (R < minSpeed) {
    R = minSpeed;
  }
  drive(L, R);  // Drive Function
}

void print() {

  // Printing Sensor Values, Mortor Speeds And Resistor Values...
  Serial.print("QTR::");
  Serial.print(position);
  Serial.print("\t");

  Serial.print("Error::");
  Serial.print(error);
  Serial.print("\t");

  Serial.print("Add::");
  Serial.print(add);
  Serial.print("\t");
```

```
  for (uint16_t x = 0; x < sensorCount; x++) {
    Serial.print(sensorValues[x]);
    Serial.print("\t");
  }

  Serial.print("Resis::");
  Serial.println(resistor);
}

void drive(int L, int R) {

  // When Resistors At Max Taking Right Turns... (Shortcut-1)
  if (resistor > 245) {
    if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {
      left();
      delay(500);
    } else if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] > treshold) {
      right();
      delay(400);
    } else if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] > treshold && sensorValues[4] > treshold) {
      right();
      delay(400);
    } else if (sensorValues[0] > treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {
      left();
      delay(400);
```

```
    } else if (sensorValues[0] > treshold && sensorValues[1] > treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {

    left();

    delay(400);

   } else {

    motor1.drive(L);

    motor2.drive(R);

   }


   // When Resistors At Lowest Taking Right Turns... (Shortcut-2)

  } else if (resistor < 10) {

   if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {

    right();

    delay(500);

    } else if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] > treshold) {

    right();

    delay(400);

    } else if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] > treshold && sensorValues[4] > treshold) {

    right();

    delay(400);

    } else if (sensorValues[0] > treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {

    left();

    delay(400);

    } else if (sensorValues[0] > treshold && sensorValues[1] > treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {

    left();
```

```
    delay(400);
   } else {
    motor1.drive(L);
    motor2.drive(R);
   }

   // When Resistors At Neither Max Nor Lowest Drives Forward...
  } else {
     // if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] > treshold) {
   //  right();
   //  delay(400);
    // } else if (sensorValues[0] < treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] > treshold && sensorValues[4] > treshold) {
   //  right();
   //  delay(400);
    // } else if (sensorValues[0] > treshold && sensorValues[1] < treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {
   //  left();
   //  delay(400);
    // } else if (sensorValues[0] > treshold && sensorValues[1] > treshold && sensorValues[2] < treshold && sensorValues[3] < treshold && sensorValues[4] < treshold) {
   //  left();
   //  delay(400);
   // } else {
   motor1.drive(L);
   motor2.drive(R);
   // }
  }
```

```
}

void left() {
  // Turns Left...
  motor1.drive(0);
  motor2.drive(turnSpeed);
}

void right() {
  // Turns Right...
  motor1.drive(turnSpeed);
  motor2.drive(0);
}

void brake() {
  //Brake...
```