CoderDojo Arduino Robot Labs

Prerelease Version 0.2

Please send feedback to Dan McCreary

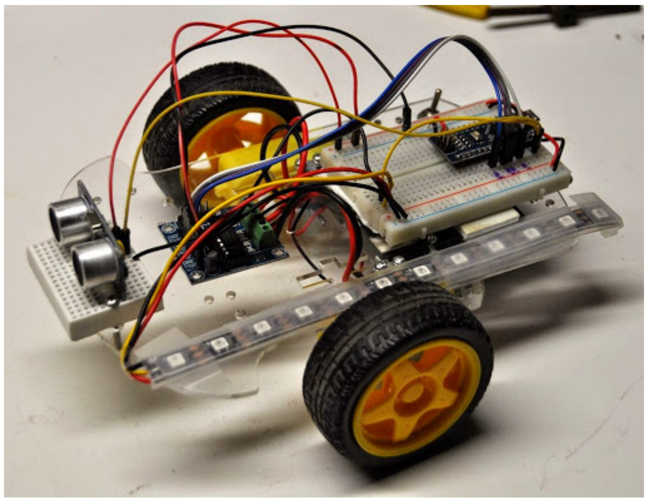
dan.mccreary@gmail.com

Open Source code at:

https://github.com/dmccreary/moving-rainbow

Build your own parts list:

http://datadictionary.blogspot.com/2015/01/30-coderdojo-robot-version-3.html



CoderDojo Robot Labs

Prerequisites

These labs assume you are already familiar with the Arduino system. You should have the Arduino development software installed and be familiar with how to write an Arduino blink program, read sensors and control analog and digital outputs.

Overview of Labs

1. **Motor Control -** Use a motor controller chip turn on and off a motor (H-Bridge)
2. **Change Motor Speed with PWM** – Us Pulse Wave Modulation (PWM) to change the speed of a motor.
3. **Forward and Reverse** – Make a motor go forward and reverse
4. **Moving and Turning** – Make two motors work together to move forward and turn
5. **Read the distance sensor** – Read the distance sensor to detect object in front of the robot
6. **Control the LED display** – display distance status on the LED strip
7. **Collision Avoidance** – move forward till an object is in front of us and then turn
8. **Slowing before turns** – slow down as you approach an object before a turn

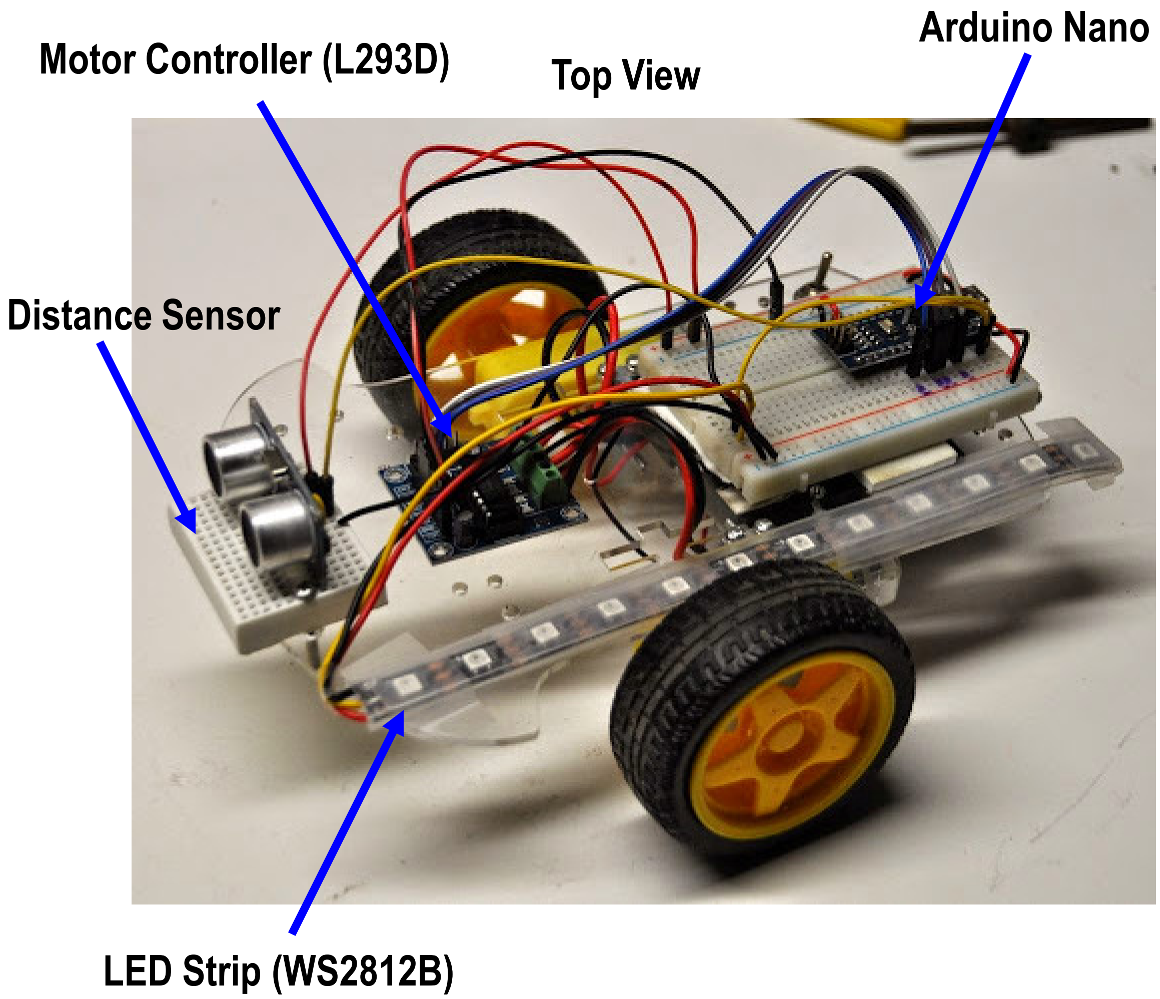
A Warning About Power

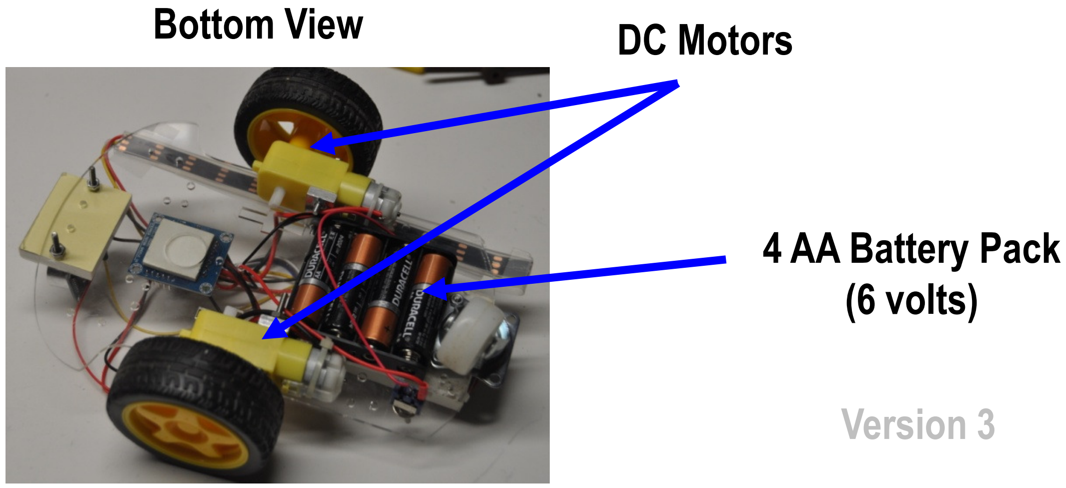
Our Arduino microcontrollers use 5 volts. Motors can often use 6 to 12 volts! Make sure you do not mix these two! Putting over 5 volts into the input of the Arduino or your laptop will damage it. Worse yet, you can easily destroy your laptop power system if you plug the power into the wrong area.

Make sure you **disconnect** the main robot battery power before you reprogram your Arduino on your computer.

If you are unsure of how to hook up the power, just ask your mentor.

Robot Parts



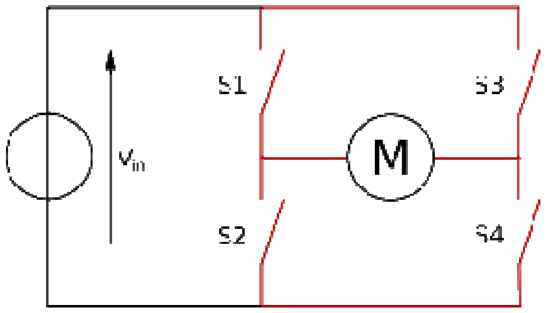


Version 3

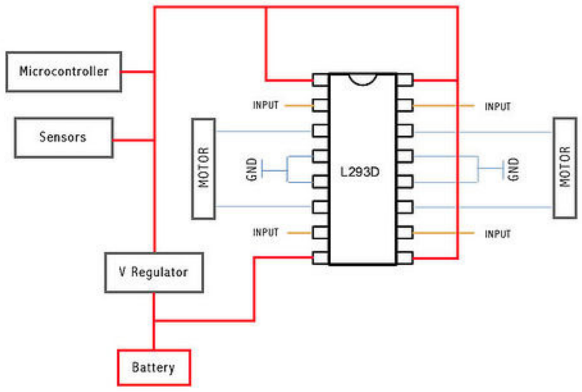
Lab 1: Motor Control

Background

In most of our Arduino labs so far we applied a voltage of 0 to +5 volts to make something change. However motors are different. We need to be able to reverse the voltage to the terminals of a motor to get it to change direction. We do this by using an "H-Bridge" circuit like this:



In the circuit above, the motor will go in one direction if switches 1 and 4 are closed and 2 and 3 are open. If switches 2 and 3 are closed and 1 and 4 are open, the motor will **reverse** directions. The job of reversing polarity is done by a device called a "motor controller" circuit. We will use the popular L283D chip in these labs. This single chip allows us to control two motors, forward and back, using output pins on our Arduino. Connections to the L292D chip are shown below:

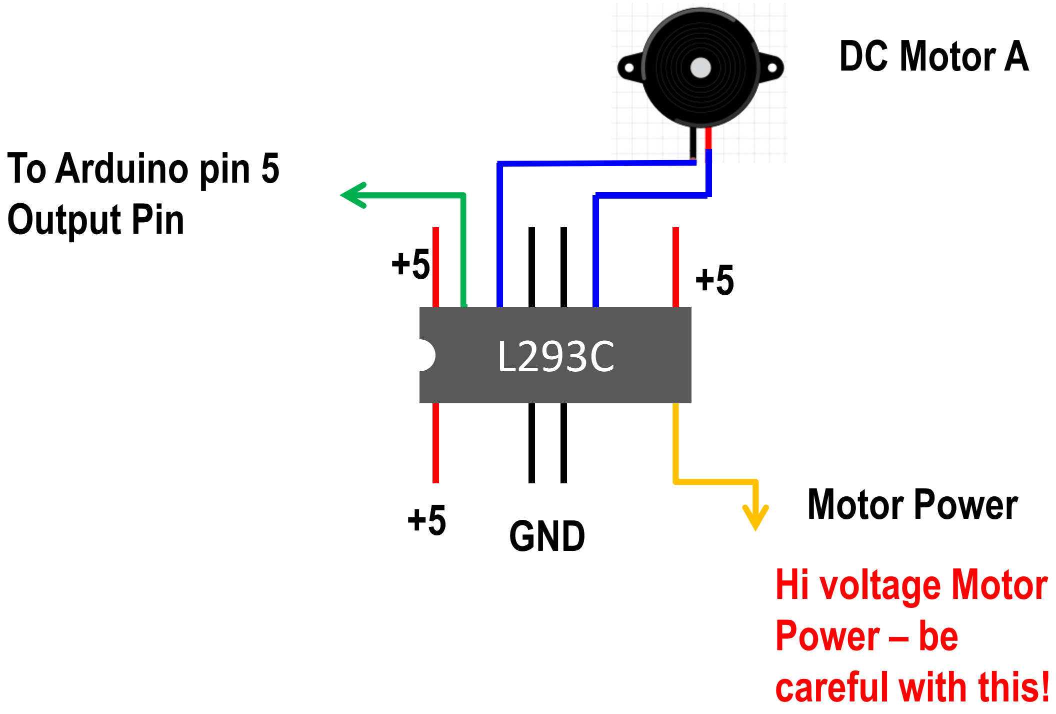


Note that we need two connections between the Arduino and the chip for each motor. Also note that the Battery power MUST go through the regular before it goes to the Arduino Microcontroller and sensors.

Lab 1: Motor Control (continued)

Background

1) Add power and ground connections to a L293D chip. I used red for 5v and black for ground. Note that motor power is in the lower right in orange.
2) Connect the green wires to Arduino. They can use any digital output pin, however we suggest using pins 3, 5, 6, 9, 10, and 11 since they also will allow us to control motor speed
3) Add blue wires to motor
4) Download the test program



http://en.wikipedia.org/wiki/H_bridge

Lab 1: Motor Control

Background

This lab is very similar to the "Blink" lab, but instead of an LED going on and off a motor will go on for one second and off for one second.

Steps

1. Disconnect the main robot battery power supply
2. Enter the code below into your Arduino Development Software (IDE).
3. Upload the code into your Arduino
4. Disconnect the Arduino from your PC
5. Connect the robot battery

```
int motorPin = 5;   // to one of the inputs on the motor controller (L293D)

void setup() {
  pinMode(motorPin, OUTPUT);
}

void loop() {
  digitalWrite(motorPin, HIGH);  // turn the motor on
  delay(1000); // wait 1 second
  digitalWrite(motorPin, LOW);  // turn the motor off
  delay(1000); // wait 1 second
}
```

Your motor should now go on for one second and then stop for one second.

Note that you might need to adjust the **motorPin** line if your setup does not connect a motor to the Arduino pin 5.

The digitalWrite() function has two parameters: the pin to use and the value (either HIGH or LOW).

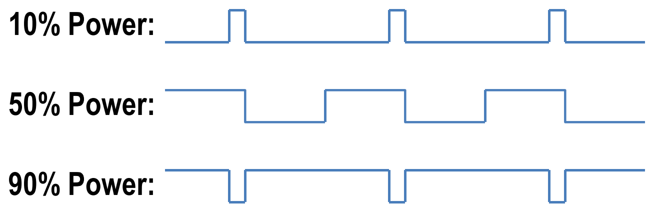
Continue to Explore

Change the delay time (in milliseconds) to other values such as 200 or 2000. What is the result?

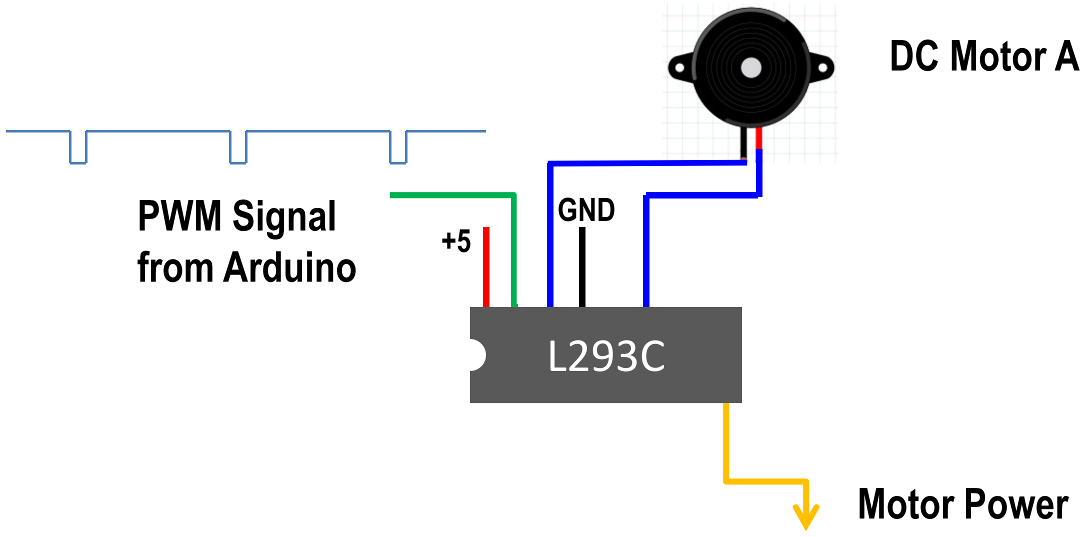
Lab 2: Changing Motor Speed

Background

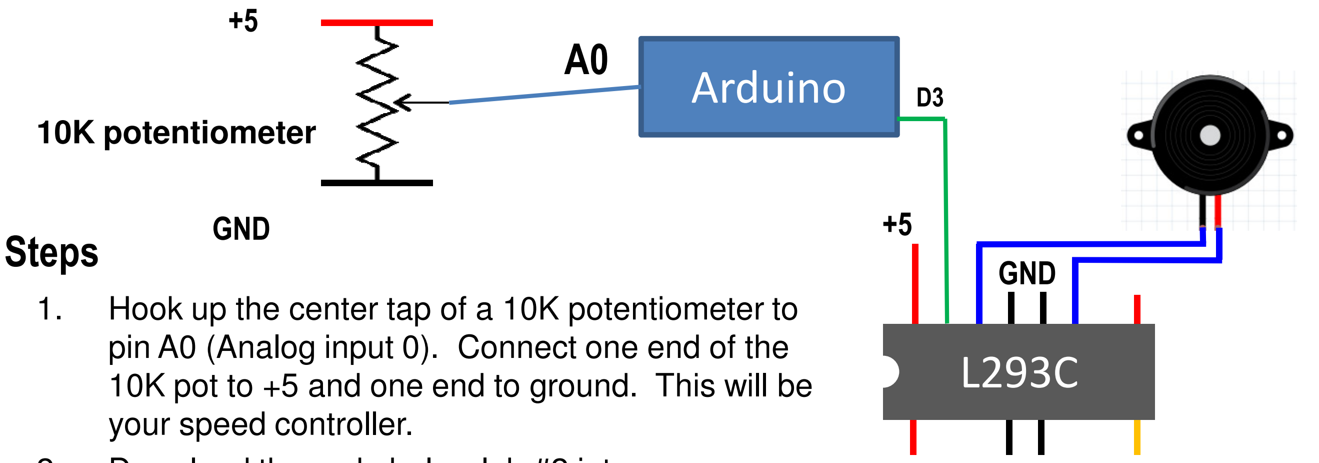
Arduino's have two ways of controlling output: **Digital** and **Analog**. Digital Output is always HIGH (5V) or LOW (0V) without any middle value. Analog output on the Arduino can vary between any value from 0 t 255. However they don't change the voltage to a specific value, they only change the width of pluses on the output. This is done doing using a technique called **Pulse Wave Modulation** or PWM. The shape of the voltage out of the pin will be similar to the waveforms below:



There are several pins on most Arduinos that are configured for PWM (3, 5, 6, 9, 10, and 11). In this lab we will just use pin 3. We can change the motor speed by using the AnalogWrite function which we give a value from 0 (off) to fully on (255).



Lab 2: Speed Control



Steps

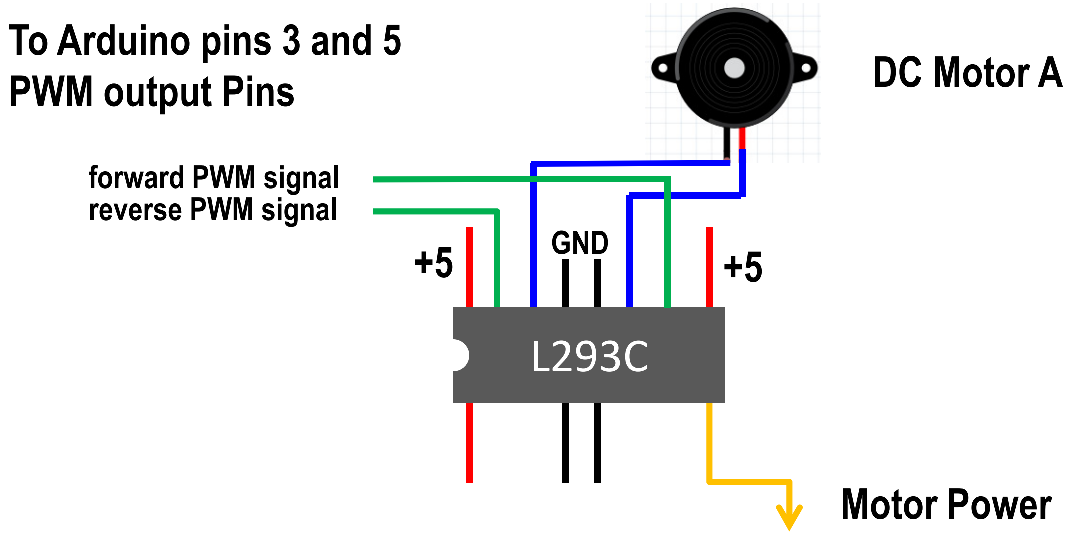
1. Hook up the center tap of a 10K potentiometer to pin A0 (Analog input 0). Connect one end of the 10K pot to +5 and one end to ground. This will be your speed controller.
2. Download the code below lab #2 into your Arduino.
3. As you turn the pot, the motor will change speed.

```
int potPin = A0;    // select the input pin for the potentiometer
int motorPin = 3;   // The pin for the motor
int potValue = 0;   // variable to store the value from the pot

void setup() {
  pinMode(potPin, INPUT);
  pinMode(motorPin, OUTPUT);
}

void loop() {
  potValue = analogRead(potPin); // get the value from the pot
  speedValue = map(potValue, 0, 1023, 0, 255) // convert into proper range
  analogWrite(motorPin, speedValue); // set the speed
  delay(100); // wait 1/10th of a second
}
```

Reversing Directions

Background

How can our robot move around? One way is to have our robot turn by moving one wheel forward and one in reverse. To do this we will now hook up two **different** PWM signals to the top part of our Motor Controller, each one controlled by a different pin of the Arduino microcontroller. When we want to go forward we apply a PWM signal of 0 to 255 to one pin. When we want to go in the opposite direction, we apply a PWM to the other pin.



http://arduino.cc/en/Reference/analogWrite

Direction Control

Controlling Direction

This code uses not just one, but two signals between the Arduino and the motor controller. The program checks to see if the value of the potentiometer is in the lower half (0 to 511). If it is it will move the motor in one direction. If the value is in the upper half (512 to 1023) then the motor will increase the speed in the opposite direction. The motor will not be moving in the middle range of numbers.

```
int potPin = A0; // the pin for the potentiometer
int motorAForwardPin = 3;  // Pin to make the motor go forward
int motorABackwardPin = 5; // Pin to make the motor go backward
int potValue = 128;  // variable to store the value coming from the sensor
int speedValue = 0; // variable to store the value going to the motor

void setup() {
  pinMode(potPin, INPUT);
  pinMode(motorAForwardPin, OUTPUT);
  pinMode(motorABackwardPin, OUTPUT);
}

void loop() {
  potValue = analogRead(potPin); // get the value from the Potentiometer from 0 to 1023
  Serial.print("pot=");
 Serial.print(potValue);
  if (potValue < 512) { // we will go backward if the value us under this
    speedValue = map(potValue, 0, 511, 255, 0);  // convert to the right range
    analogWrite(motorABackwardPin, speedValue); // go back this speed
    analogWrite(motorAForwardPin, 0);
 } else {
  speedValue = map(potValue, 512, 1023, 0, 255);
  analogWrite(motorAForwardPin, speedValue);  // go forward this speed
  analogWrite(motorABackwardPin, 0);
 }
delay(100); // sample ever 1/10th of a second
}
```

Distance Sensor

```
void loop() {

  // get the distance from the ping sensor in CM
  dist_in_cm = get_distance_cm();

}
```

```
int get_distance_cm()
{
  long duration, cm=0;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the
sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);
  // convert the time into a distance
  cm = microsecondsToCentimeters(duration);
  Serial.print(" cm=");
  Serial.println(cm);
  return cm;
}

long microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per
centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

LED Strip

```
// ping sensor
const int pingPin = 12;
int dist_in_cm = 120;

// This LED strip is used for distance feedback
// The closer we get to an object in front of us, the further up the blue pixel is on
#include <Adafruit_NeoPixel.h>
#define LEDPIN 11 // connect the Data from the strip to this pin on the Arduino
#define NUMBER_PIEXELS 12 // the number of pixels in your LED strip
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMBER_PIEXELS, LEDPIN, NEO_GRB +
NEO_KHZ800);

int old_strip_index = 0;
int new_strip_index = 0;
int power_turn_level = 150; /* full power on turns */

void setup() {

  Serial.begin(9600);

  strip.begin();
  strip.setPixelColor(0, 10, 0, 0);
  strip.show();
}

void loop() {


  // don't go over the max
  if (new_strip_index > (NUMBER_PIEXELS - 1)) {
    new_strip_index = 11;
  }

  // only draw if there is a change
  if ( old_strip_index != new_strip_index) {
    // erase the old strip
     for (int i=0; i < NUMBER_PIEXELS; i++)
        strip.setPixelColor(i, 0, 0, 0);
    // turn on new value to a light blue
     strip.setPixelColor(new_strip_index, 0, 0, 30);
     strip.show();
  };
```

Obstacle Avoidance

Page 1

```
// ping sensor
const int pingPin = 12;
int dist_in_cm = 120;

// This LED strip is used for distance feedback
// The closer we get to an object in front of us, the further up the blue pixel is on
#include <Adafruit_NeoPixel.h>
#define LEDPIN 11 // connect the Data from the strip to this pin on the Arduino
#define NUMBER_PIEXELS 12 // the number of pixels in your LED strip
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMBER_PIEXELS, LEDPIN, NEO_GRB +
NEO_KHZ800);

int old_strip_index = 0;
int new_strip_index = 0;
int power_turn_level = 150; /* full power on turns */

// adjust these till the robot goes streight to compensate for motor differences
int power_forward_right = 180; /* half power on turns */
int power_forward_left = 180; /* half power on turns */

// motor pins.  Note that only pins 2,5,6, 9 and 10 can be used for pwm
int right_forward = 5;
int right_reverse = 3;
int left_forward = 6;
int left_reverse = 9;

// try this time to make a right turn just above 90 degrees
int delay_time_ninty_turn = 200;
// if we are under this distance, make a turn
int cm_for_turn = 10;
int delay_time_forward = 100;
```

Obstacle Avoidance Setup

Page 2 the setup() function

```
void setup() {

  Serial.begin(9600);

  strip.begin();
  strip.setPixelColor(0, 10, 0, 0);
  strip.show();

  pinMode(pingPin, INPUT);

  pinMode(right_forward, OUTPUT);
  pinMode(right_reverse, OUTPUT);
  pinMode(left_forward, OUTPUT);
  pinMode(left_reverse, OUTPUT);

  // turn all off
  digitalWrite(right_forward, LOW);
  digitalWrite(right_reverse, LOW);
  digitalWrite(left_forward, LOW);
  digitalWrite(left_reverse, LOW);
  // for debugging
  // Serial.println('Start');
  delay(1000);
}
```

Main Loop

The loop() function

```
void loop() {

  delay(100);

  // get the distance from the ping sensor in CM
  dist_in_cm = get_distance_cm();
  new_strip_index = dist_in_cm / 5;

  // don't go over the max
  if (new_strip_index > (NUMBER_PIEXELS - 1)) {
    new_strip_index = 11;
  }

  // only draw if there is a change
  if ( old_strip_index != new_strip_index) {
    // erase the old strip
     for (int i=0; i < NUMBER_PIEXELS; i++)
        strip.setPixelColor(i, 0, 0, 0);
    // turn on new value to a light blue
     strip.setPixelColor(new_strip_index, 0, 0, 30);
     strip.show();
  };

  // if there is something in front, turn right
  if (dist_in_cm < cm_for_turn) {
      turn_right();
    } else {
      move_forward();
    }

  // Serial.print(" new=");
  // Serial.print(new_strip_index);

}
```

Turn Right And Forward

```
void turn_right() {
  Serial.println("turning right");
  analogWrite(right_forward, LOW);
  analogWrite(right_reverse, power_turn_level);
  analogWrite(left_forward, power_turn_level);
  analogWrite(left_reverse, LOW);
  delay(delay_time_ninty_turn);
}

void move_forward() {
  Serial.println("moving forward");
  analogWrite(right_forward, power_forward_right);
  analogWrite(right_reverse, LOW);
  analogWrite(left_forward, power_forward_left);
  analogWrite(left_reverse, LOW);
  delay(delay_time_forward);
}
```