Michael Jacques 12/1/2020

Line Follower Project Report ECE 111

Line Follower

Section #1: Project Summary

This report explains the circuit and software that controls a line following robot. This robot makes use of an Arduino microcontroller, two photoresistors to determine the light levels under the robot, as well as two DC motors attached to wheels that allow the robot to move. The robot works by placing a photoresistor on each side of the thick black line that will be followed. Then the motors start spinning the wheels. If either photoresistor detects a change in light level because it is over the black line, the motor on the same side of the robot as that photoresistor is turned off while the other motor continues turning. This allows the robot to turn in the direction of the line. Then once neither photoresistor is over the black line, both motors spin to move the robot forward.

The photoresistors are implemented as part of a voltage divider circuit which allow the Arduino to detect the change in light level as a change in voltage between the photoresistor and a 10 kΩ resistor. Once a light threshold has been determined, the software can detect the change in light levels and turn on and off the motors accordingly. The motors are connected making use of an NPN transistor as a low side switch. The Arduino sends current to the base of the transistor when the motor's corresponding light level is above the threshold. This allows the motors to be turned on and off as needed. The left and right side of the robot function as two separate circuits that both have the same configuration.

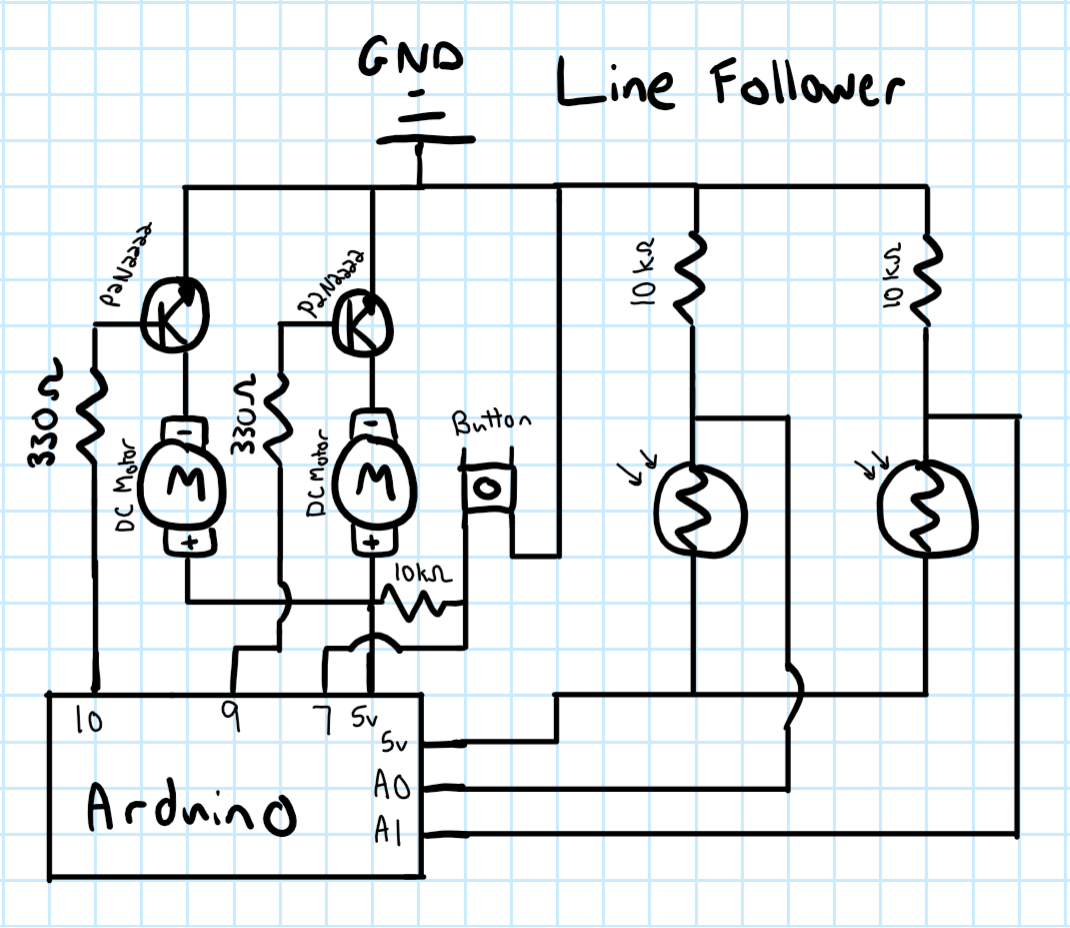
In order to improve the design, a button was implemented that could be pressed to determine new light thresholds. This allows the robot to work accurately in many different settings. This works by placing the robot on the track with both photoresistors over the white board. Then when the button is pressed, the light values for both photoresistors are stored and new thresholds are determined by subtracting from the stored light value. Then once the new thresholds are determined, the motors start spinning and the robot functions the same way, but with updated, more accurate thresholds. Before the button improvement was made, the robot was very inconsistent. Whenever the light levels in the room slightly changed, even from shadows, the robot would not function properly. Then the robot would have to be manually calibrated which is very time consuming. By adding the button functionality which automatically creates the thresholds, the robot works more accurately, and it does not take long to get it running.

Throughout the design process, some problems did occur. One of the problems was that the two photoresistors that were implemented in the robot did not show exactly the same light readings under the same conditions. This meant that if just one threshold was used for both photoresistors, there was a chance that one side of the circuit would work perfectly, while the other one would either never turn on or never turn off. To solve this problem, two separate thresholds were implemented, one for each photoresistor. This means that even if the photoresistors do not come up with exactly the same readings, they will still work properly in the circuit because the threshold is tailored to that photoresistors readings. Another problem was the placement of the photoresistors. The robot functioned very differently based on how far away the photoresistors were from the track. If they were

too close, the circuit wouldn't function properly because the shadows of the photoresistors on the board would skew the readings. If the photoresistors were too far away, they had a hard time detecting the black line. Through some testing it was determined that the robot was most accurate with the photoresistors about 1.5 cm above the track.

Once the robot was complete it was tested on three different tracks of varying difficulty. The results of these tests are described and analyzed in Section #6.

Section #2: Circuit Schematic

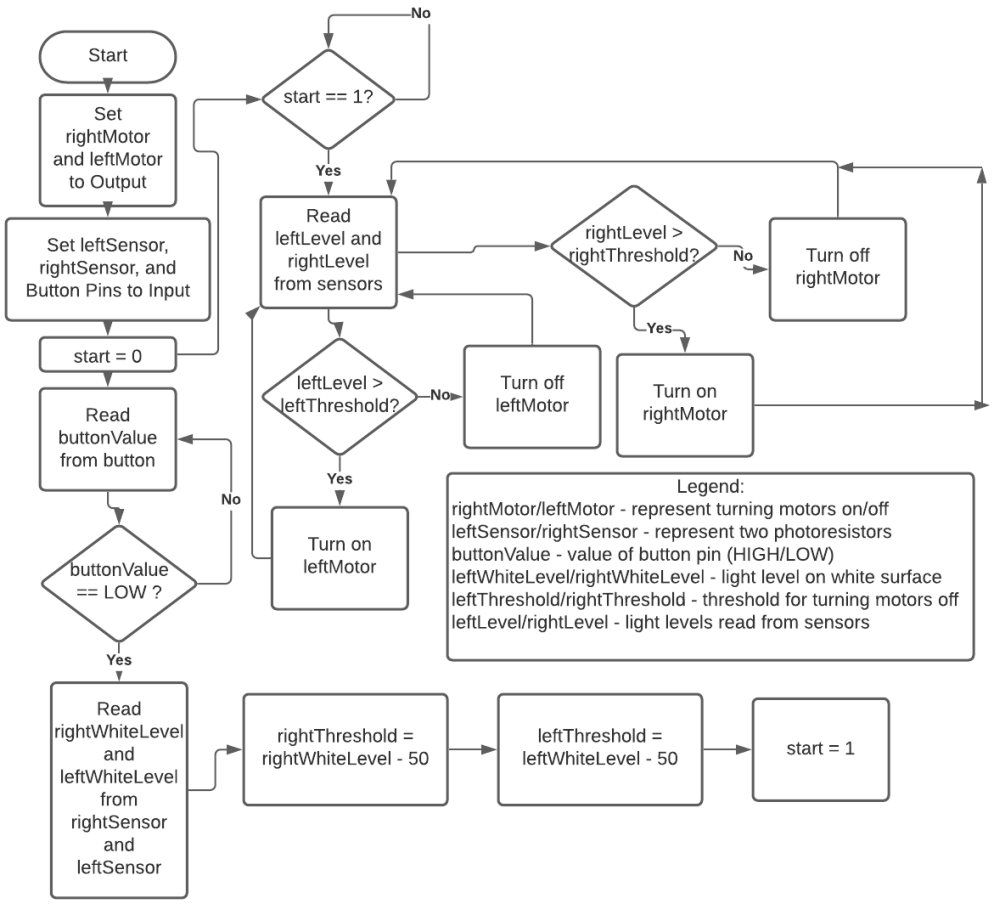


Section #3: Circuit Description

This circuit shown above uses two photoresistors and two DC motors to control a line following robot. For the DC motors, 5 Volts is connected to the positive terminals of both motors, and the negative terminals are connected to the collectors of two separate P2N2222 transistors. The base of these transistors are connected to two separate digital pins on the Arduino (in this case pins 9 and 10). Then the emitters of both transistors are connected to ground. This allows the transistors to act like a switch and an amplifier, only driving the motors when there is current running into the base terminal (which is controlled by the Arduino). The photoresistors are connected as part of a voltage divider, which allows its changing resistance to be recorded easily. For both photoresistors, 5 Volts is connected to one terminal of the photoresistor, and the other terminal is connected to a 10 kΩ resistor, which is then

connected to ground. Analog pins from the Arduino are connected in between the photoresistor and the 10 kΩ resistor, which allows us to use the voltage divider to read the voltage drop associated with the change in resistance of the photoresistor (this is given as a value between 0 and 1023, with 1023 representing 5 Volts). The values read by the analog pins are used to turn on and off the DC motors which allows the robot to follow a line. The last part of this circuit is a button which is used in the software to read the initial photoresistor values, as well as starting the loop that drives the motors. 5 Volts is connected to one side of the button with a 10 kΩ resistor, and an Arduino digital pin (in this case pin 7) is connected to the same terminal. The other button terminal is connected directly to ground. With this setup, the button is normally closed, and the circuit opens when the button is pressed. All of these components work together with the software to allow the robot to follow a line. The circuit works by having the motors stay on when the photoresistor values are above a set light threshold, but as soon as the photoresistor value drops below the threshold, it turns that motor off. Here is an example of the circuit working before it was connected to the robot chassis. Here is a video of the completed robot following a curvy line.

Section #4: Software Flow Chart



Section #5: Flow Chart Description

This flowchart represents the logic of controlling a line following robot with a button, two DC motors, and two photoresistors. First, all variables are declared. The motor variables, rightMotor and leftMotor, are set to be output pins that can either be set to HIGH or LOW (on/off). The photoresistor variables, leftSensor and rightSensor, are set to be input pins that read light values as an integer value. The button pin is also set to be an input pin. Then an integer variable called start is declared and set to "0". This variable is used to control when the motors start turning. Once all variables are declared, a loop starts. First, the value of the button pin is assigned to the variable buttonValue, which will either be HIGH or LOW. A HIGH value means the button is not being pressed and a LOW value means the button is being pressed. Then there is a decision statement that looks at whether the button has been pressed. If buttonValue is HIGH, it loops back to the start and reads buttonValue again. If buttonValue is LOW, the program continues executing. First, two integer values, rightWhiteLevel and leftWhiteLevel are read from the sensors. These values represent the light levels read from the sensors while they are over a white board. Then, the rightThreshold and leftThreshold values are created by subtracting 50 from the values of the rightWhiteLevel and leftWhiteLevel. The threshold values are used to determine if the sensors are over the black line. This program assumes that the black line will have a light level that is more than 50 values less than the white background, and it also assumes that disturbances like shadows will not affect the light level by more than 50. Once the thresholds are determined, the start variable is set to "1" which starts the motor loop. The second loop of this program only begins when the start variable is set to "1". Once this condition is true, the current light levels from both sensors are assigned to leftLevel and rightLevel. Then there are two separated conditional statements, one for each motor. Both of them compare the current light level to the light threshold associated with that sensor. If the current value is less than the threshold, the sensor is over the black line and it turns that motor off. For example, if leftLevel < leftThreshold, then leftMotor would be set to LOW (off). These two conditions are continuously looping and continue to turn off the motors when the light levels are below the thresholds and turning them back on when the light levels are above the threshold. This allows the robot to drive forward while following the black line in between the two photoresistors.

Section #6: Testing

This line following robot was tested on three different tracks with varying shapes. The first track was a straight line, the second track zig zagged back and forth very slightly, and the third track had two large smooth curves making an "S" shape. The robot was run on each of the tracks 10 times, and the results of the tests are documented below.

Track	Number of Tests	Average % of Track Completed
Track #1 – Straight Line	10	100%
Track #2 – Zig Zag	10	100%
Track #3 – "S" curve	10	70%

This robot performed much better than expected. On both the straight-line track and the zig zag track, the robot had no problems following the line. The robot did have trouble on the "S" track, but the curves were so big that this was expected. On the "S" track, the robot had no trouble getting through the first curve, but when the track switched directions, the robot did not consistently turn enough to stay on the track. The robot did successfully complete the "S" track two times, but in the rest of the trials it usually failed at around 50% of the track (right where the track switches directions). I believe

that this robot could be improved by making the motors turn slower, so that the photoresistors have more time to get a reading and turn the motors off. This could be done by adding gears to the design between the motor and the wheels. By slowing the robot down, it could more accurately follow very curvy lines because there would be much more time for the robot to react to the turns. Overall, this robot performed extremely well and did follow lines as expected.

Appendix: Code

```
// This program controls a line following robot circuit
// which contains two DC motors and two photoresistors.
// A light threshold is determined between the dark line
// and the white board, and when a photoresistor senses
// the dark line, its motor is turned off to keep the robot
// on the track.

// Define Pin Mappings
const int leftMotorPin = 9;
const int leftSensorPin = A1;
const int rightMotorPin = 10;
const int rightSensorPin = A0;
const int buttonPin = 7;

// thresholds will be determined later
int rightlightThreshold = 0;
int leftlightThreshold = 0;

// the photoresistor values when both photoresistors are directly
// over the white board.
int rightWhiteLevel = 0;
int leftWhiteLevel = 0;

// start variable is used to start the motors turning
```

```
int start = 0;


void setup()
{
  //Set Baud Rate, Required to use Serial Monitor
  Serial.begin(9600);
  pinMode(leftMotorPin, OUTPUT);
  pinMode(leftSensorPin, INPUT);
  pinMode(rightMotorPin, OUTPUT);
  pinMode(rightSensorPin, INPUT);
  // button is used to determine light thresholds
  // and start the robot
  pinMode(buttonPin, INPUT);

}

void loop()
{
  // read the value of the button (HIGH/LOW) (Default is HIGH)
  int buttonValue = digitalRead(buttonPin);

  // only determine thresholds and start robot if the button
  // was pressed
  if (buttonValue == LOW) {
    // Read sensor values for white light level.
    rightWhiteLevel = analogRead(rightSensorPin);
    leftWhiteLevel = analogRead(leftSensorPin);

    // create thresholds by subtracting 50 from the white
```

```
  // light level (this step predicts that the black line will
  // be more than 50 light values less than the white board).
  rightlightThreshold = rightWhiteLevel - 50;
  leftlightThreshold = leftWhiteLevel - 50;

  // print the values of the thresholds (used for testing)
  Serial.println("Left: " + leftlightThreshold);
  Serial.println("Right: " + rightlightThreshold);

  // set start variable to 1 to start the motor logic and delay for
  // half a second to give time to move hand away from button.
  start = 1;
  delay(500);
}

// start variable is 0 until the button is clicked,
// so the robot won't start until then.
while (start == 1) {
  //Read Sensor Values
  int leftLightLevel = analogRead(leftSensorPin);
  int rightLightLevel = analogRead(rightSensorPin);

  // If left light sensor is greater than threshold
  // then turn left Motor on, else turn left Motor off
  if (leftLightLevel > leftlightThreshold)
    digitalWrite(leftMotorPin, HIGH);
  else
    digitalWrite(leftMotorPin, LOW);
```

```
    // If right light sensor is greater than threshold
    // then turn left Motor on, else turn left Motor off
    if (rightLightLevel > rightlightThreshold)
      digitalWrite(rightMotorPin, HIGH);
    else
      digitalWrite(rightMotorPin, LOW);
  }
}
```