ASSIGNMENT 2

based on

Autonomous Light Finder-follower Robotic Vehicle

Student ID: 1965776
Name: JAVID MALIK KUPPATIL
Course: 2022-23SEM2AACIS006-2
BSc artificial intelligence and robotics (hons)

Table of Contents

Table of Figures

I. Introduction

For many years, the idea of autonomous automobiles has fascinated people, and recent developments in robotics technology have made this idea a reality. These machines have the potential to revolutionize several industries, including transportation, surveillance, and exploration. They are capable of operating without the need for human interaction. The creation of robotic light follower-finder cars that can locate and follow light sources is one of the most interesting uses of autonomous vehicles. For this project, a wide range of abilities are needed, including mechanical design, programming, electronics, and control systems. The construction of an autonomous light follower-finder robotic vehicle will be the focus of this assignment, which will also examine the numerous design issues, difficulties, and potential applications of this technology.

There are various components which we used for this project, which is so exciting, the components used in an autonomous light follower-finder robotic vehicle may vary depending on the specific design and requirements of the project. However, some of the common components used in project are:

- The robotic vehicle's microprocessor, which directs its motions and handles the processing of sensory data, is its brain. The Arduino, Raspberry Pi, and PIC microcontrollers are frequently utilized in robotics projects.
- The wheels of the vehicle or other loco motional devices are moved by motors. I used the most popular kind of motor used in robotics projects which is a DC motor.
- Sensors: Sensors help a vehicle follow a source of light by detecting it. In light follower-finder robotic vehicles, photodiodes, phototransistors, and photoresistors are frequently utilized as sensor types

II. Project

Management has asked the department of robotics engineering to produce an autonomous light finder-follower robotic vehicle using the following main components:
1. An Arduino Bluno Nano Microcontroller or Arduino with ATmega328P processor (Uno/Nano) to integrate andcontrol all the electronic components and sensors.
2. Implement two gear down DC Motors to provide movement to the autonomous robot.
3. One dual H-Bridge driver module to control simultaneously both DC motors.
4. Two Infrared Speed Sensor Modules to calculate both motor's RPM, travelled distance and velocity.
5. One vehicle chassis with included accessories to mount all the electronic components on it.
6. Three small Breadboards to interconnect the microcontroller and all the electronic components.
7. One Ultrasonic sensor to detect the closest objects-obstacles in front of the robotic vehicle.
8. Two Light Dependent Resistors (LDRs) to detect and scan light at different levels within Robot's vicinity.
9. Two Light-Emitting Diodes (LEDs) to indicate the direction of the light source and its intensity.
10. One OLED Display to show relevant updated information and undergoing processes.
11. One Temperature and Humidity Sensor to detect and constantly retrieve these values.
12. One Potentiometer to calibrate the DC Motors and Temperature-Humidity sensor
13. One passive buzzer to indicate with sound relevant undergoing events.
14. One button to work as user input to control microcontroller processes. Several resistors with various values to safely implement LDRs, LEDs and button.
15. One Power Bank with two output ports to provide power to the Robotic Vehicle artefact.

You are a robotics engineer and have been asked to undertake the following tasks:

A. Build a Robotic Vehicle that integrates all previously listed components in such a manner that will fulfil the following design specifications:
   • The DC Motors and their corresponding IR speed encoder sensors should be positioned in the front part of the chassis; one on the Right side and the other on the Left side.
   • The Positioning of the Ultrasonic sensor at the front of the vehicle to efficiently detect upcoming obstacles and have an appropriate elevation to avoid ground reflections interfering with the sensor readings.
   • The LDRs must be on each side (Right/Left) of the vehicle, pointing forward and being able to efficiently channel incoming light.
   • The LEDs must be positioned at each side (Right/Left) without causing interference to the LDRs nor preventing the visibility of the OLED display.
   • The OLED display should be in an upright position, no other components should prevent the display to be seen clearly.
   • The Humidity-Temperature sensor should be in a position where the Robotic Vehicle's various components do not interfere with the readings.
   • The passive buzzer should be in a position where it can be easily heard.
   • The button, potentiometer and power bank switch should be easily accessible.

• Use the breadboards to neatly interconnect all necessary components and simultaneously provide the support necessary to securely place most of them on top of it.
• The Robotic Vehicle should have a sturdy construction.
• The overall design and construction of the Robotic Vehicle should be very thorough.

B. Design a program for the Bluno Nano microcontroller that integrates all components and executes the following functions:

• The Robotic Vehicle program should be able to display on the Serial Monitor (under Arduino IDE) all relevant readings and events in real-time such as: calibration procedures, light source scanning, velocity, sensor readings, errors, etc.
• The Display should indicate the main events or actions taking place in real-time including at least readings from the Ultrasonic Sensor (Distances to closes objects), readings of temperature/humidity sensor, and motors readings (velocity, distance travelled, RPM); more information could be added; it is up to the designer's discretion.
• The LEDs should be used to show the relative proximity distance and direction of the main light source, also they should be used as indicators of several events taking place such as: calibration procedures, light source scanning, end of the program, etc.
• A Light Source Scanning program subroutine should be implemented at the beginning of the program execution to make sure the Robotic Vehicle detects and analyses several light sources surrounding the vehicle and creates a light intensity reference threshold in the process.
• A DC Motors calibration program subroutine should be implemented to ensure the operation/speed consistency of each of the DC motors. This will guarantee the correct and predictable movements of the Robotic Vehicle.
• Due to the possible inaccuracy of the Temperature-Humidity sensor (±0.5°C and ±1%) it is necessary to develop a program subroutine that allows recalibration of the temperature and the humidity readings. The sensor should be recalibrated to match the values provided by external Temperature and Humidity sensors.
• The button should be used to access or not both calibration procedures.
• The potentiometer should be implemented to control the values of DC motors, temperature and humidity during the calibration procedures.
• The DC Motors should be programmed to operate based on the readings obtained from the LDRs, moving the robotic vehicle forward and constantly adjusting the direction it must follow to go towards the main light source.
• The two Infrared Speed sensor modules should be able to count each motor RPM and derive from these values the total distance travelled and the velocity (speed and direction) in real-time.
• The passive buzzer should emit various sound patterns representing each distinctive undergoing event.
• An Exit program subroutine should be implemented to ensure the Robotic Vehicle stops once it is 8cm or less in front of the main light source.

C. In summary, the basic behaviour and operation of the Robotic Vehicle should be the following:
• The Robotic Vehicle should be able to switch on and be fully operational.

• It should allow the user the chance to perform or not calibration procedures for the right DC motor, left DC motor, temperature and humidity sensor.
• Scan the light levels within the Robotic Vehicle surroundings and produce a light intensity threshold.
• Constantly take distance readings from the Ultrasonic sensor, temperature/humidity sensor readings and Infrared speed sensor readings and show them in real-time on the Display.
• The LEDs should indicate the light intensity and direction of the main light source.
• The Robotic Vehicle should move forwards and constantly change direction (when necessary) towards the main light source.
• When the main light source is reached the Robotic Vehicle should be able to stop and execute an exit routine.

III. Components and Their Models

- DFRobot Bluno Nano Bluetooth Improvement Board DFR0296: This can be a little board that incorporates a Bluetooth module and a microcontroller, based on the Arduino Nano. It can be utilized to create Bluetooth-enabled ventures. (Figure 1)

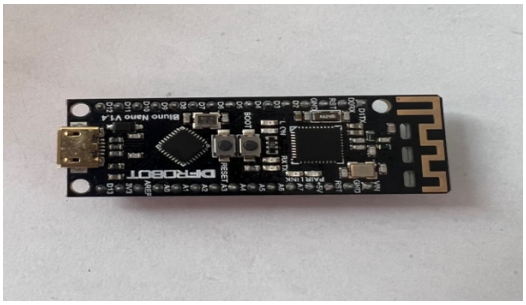


Figure 1

- 2WD Robot Savvy Car Chassis DIY Packs Intelligent Engine with Following Speed and Tacho Encoder 65x26mm Tire for Raspberry Pi (2WD): This is often a pack that incorporates a chassis and wheels for building a two-wheel drive robot car. It moreover incorporates an encoder for following the speed of the wheels. (Figure 2)

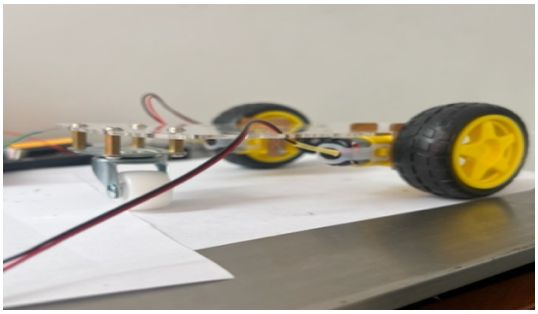


Figure 2

- ZHITING 5pcs LM393 Chip Engine Measuring Comparator Speed Sensor Module Opening Sort IR Optocoupler for MCU Arduino: Usually a sensor module that employments an IR optocoupler to measure the speed of an engine. It can be utilized with microcontrollers just like the Arduino. (Figure 3)

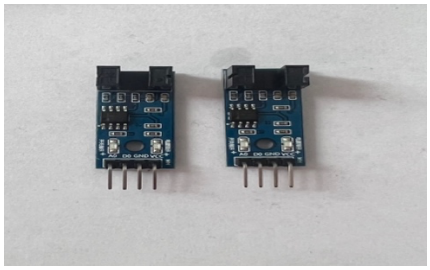


Figure 3

- HALJIA HC-SR04 Ultrasonic Sensor Remove Measuring Module Congruous with Arduino: Typically, an ultrasonic remove sensor module that can be utilized with the Arduino or other microcontrollers to degree separations. (Figure 4)

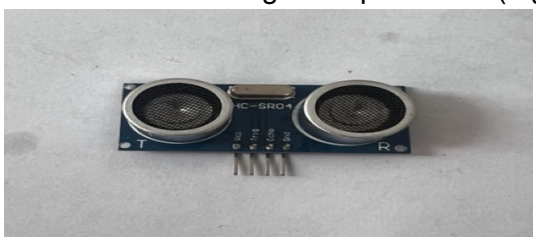


Figure 4

- DollaTek 5Pcs Double Channel L298N PWM Speed DC Engine Driver Board Dual H-Bridge Stepper Engine Module: This is often an engine driver board that can be utilized to control DC engines or stepper engines. It incorporates two H-bridge circuits and bolsters PWM speed control.

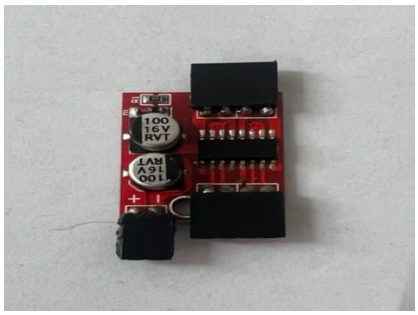


Figure 5

- TUOFENG 22 awg Strong Wire-Solid Wire Kit-6 Different Coloured 9 Meter spools 22 Gage Jumper Wire -Hook up Wire Unit: Usually a pack of strong centre wires in several colours, which can be used for wiring electronic circuits.

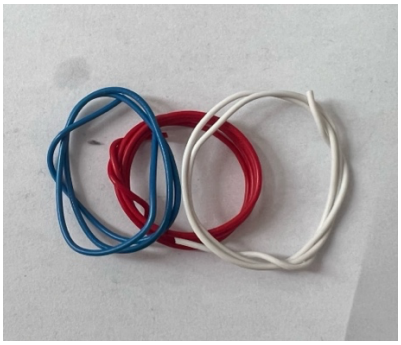


Figure 6

- MakerHawk OLED Show Module I2C IIC 128X64 Driven Display Module for Arduino UNO R3 STM 0.96 Inch and pcs Wires 20CM 40-Pin Female to Female: This is often a little show module with a 128x64 OLED screen, which can be controlled by microcontrollers just like the Arduino. The pack too incorporates female-to-female jumper wires.

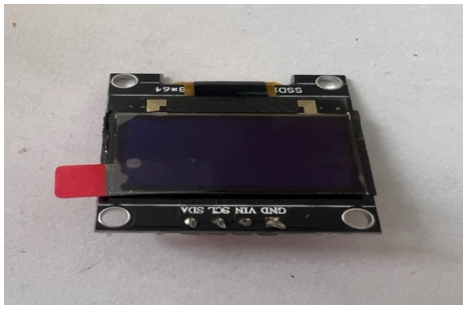


Figure 7

- Nanlaohu 6PCS Scaled down Breadboard Unit 170 tie-Points Scaled down Little Solderless Breadboard Consistent for Arduino Proto Shield: These are little breadboards with 170 tie focuses, which can be utilized for prototyping electronic circuits.

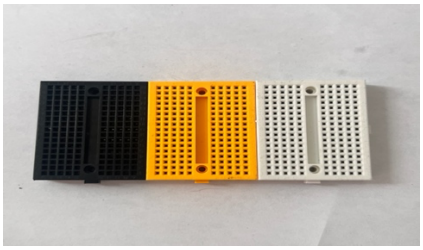


Figure 8

- ZKAPOR Small scale USB Cable Android Cable [1M/3.3FT] Nylon Braided USB Cable Quick USB Charger Charging Cables: Typically, a smaller scale USB cable for charging and interfacing gadgets like smartphones or microcontrollers

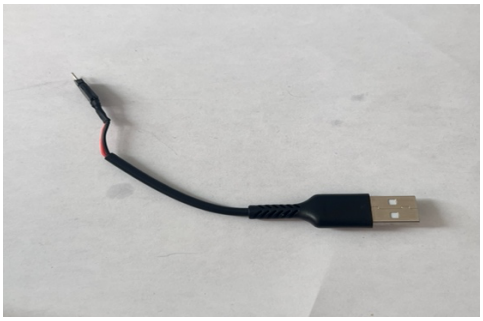


Figure 9

- AZ Delivery DHT22 AM2302 Temperature and Humidity **Sensor:** Another temperature and humidity sensor **module** compatible with Arduino and Raspberry Pi.

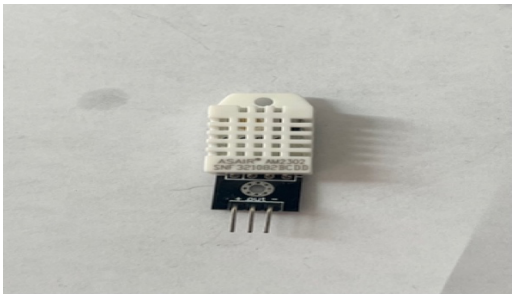


Figure 10

- AZ Delivery 5 x KY-006 Passive Piezo Buzzer Alarm Module: A module that produces **a** sound when voltage is applied to it.

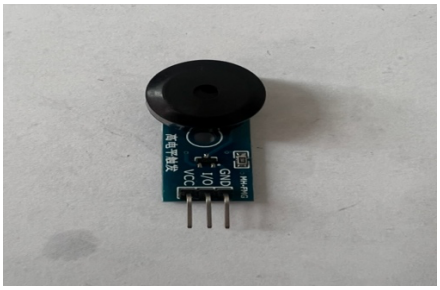


Figure 11

- High **quality 10pcs** 5MM GL5516 Light Dependent Resistor **Photo Resistor** LDR: **Light dependent** resistor that changes its resistance in response to changes in light intensity.

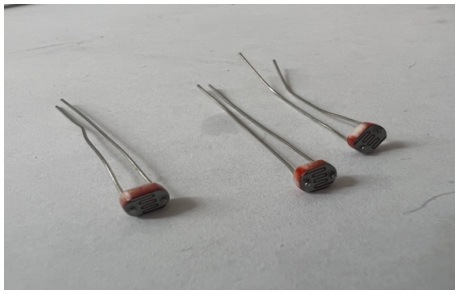


Figure 12

- Youmile **Pack** (5 **Colors** x **50 Pack)** 3mm LED Light Diode **Light Split Draw Set:** A **set** containing 3mm LEDs in five different colors.

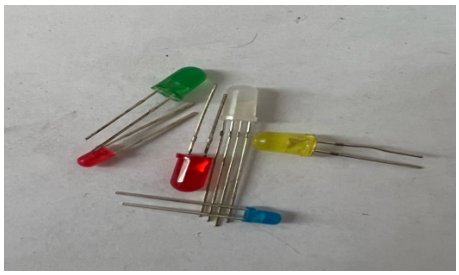


Figure 13

- Chanzon **100pcs** 5mm RGB LED Diode Lights Tricolor: A kit containing 5mm RGB LEDs that can emit different colors by **changing** the voltage applied to them.

Figure 14

- sourcing map **10pcs** 2 Position Straight 6P DPDT Micro Slide Switch Power Switch: A small switch that can be used to turn **a circuit** on or **off. acquisition card** Momentary Tact Tactile Push Button Switch: A small **push** button switch used for momentary connections.

Figure 15

- Youmile **100 Pack 6x6x5mm** Miniature Micro Momentary Tactile Push Button Switch: Another set of small **push** button switches that can be used for momentary connections.

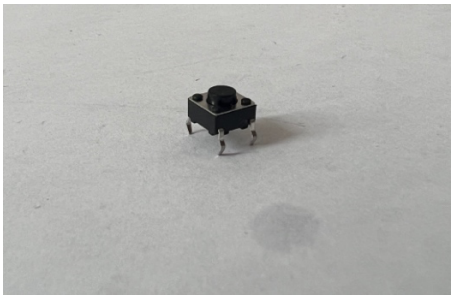
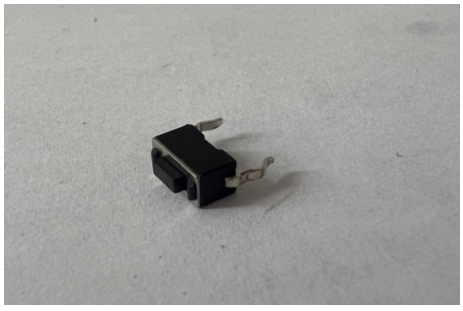


Figure 16



.

Figure 17

- HELLOYEE 10K Ohm Trimpot Rotary Breadboard Trim Potentiometer **with** Knob: A potentiometer that can be used to control **circuit voltage.**

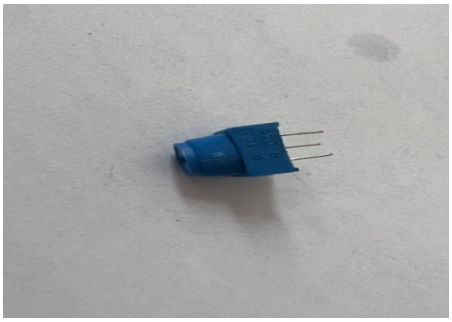


Figure 18

- Power Bank by YOBON 10000mAh: **a** portable battery that can be used **as a** power **source for electrical** devices.

.

Figure 19

- Resistor **Set:** A **set** containing 30 different **value resistors.** nuoshen **7.0" Precision** Wire Stripper: A tool used **to strip** insulation from wires.

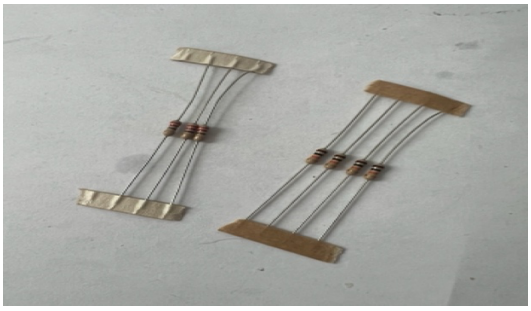


Figure 20

- Soldering **Kit, Soldering** Iron with Multimeter: A **set that includes** a soldering iron and **a** multimeter, as well as other soldering **irons** and accessories.

Figure 21

- FYSL 2.54mm Pin Header PCB Board Pin Male & Female Connector Kit: A kit that includes male and female headers that can be used for breadboard or circuit board prototyping.

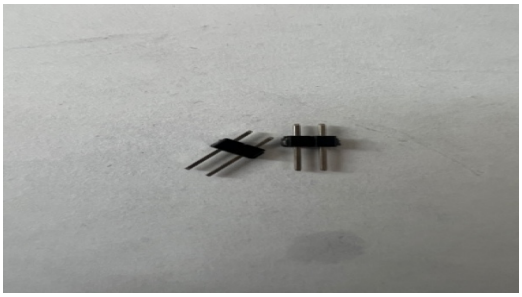


Figure 22

IV. Integration and design of all components

Practical part

I followed the instructions and did what I was supposed to and put the DC motors and their sensors in the front of the robot. I put one motor on the right and one on the left and made sure they were firmly held in place. I made a robot that works well and goes at the right speed by doing what the instructions said. I made my robot and put a special tool at the front of it called an ultrasonic sensor. I was very careful when I did this. I put it in a good place so I could find any obstacles easily. To ensure the sensor gives accurate readings,

I adjusted its height to prevent interference from the ground's reflection. My robot can now find and dodge things in front of it because of the things I did to it. I put light sensors on the right and left sides of the car. I made them look straight ahead so they could see the light coming in better. I changed some things to make sure the light was going in the right place. Now that I have placed the LDRs in the right place,

my robot can find out when the number of light changes on each side while it moves ahead. I carefully added little lights to the right and left sides of my robot while I was building it. I was careful not to move the LDRs that are on each side of the car. I made sure that the small lights did not block the view of the screen. I put some lights on my robot to make it brighter. These lights are called LEDs and they shine on both sides. This won't harm other parts of the robot such as the LDRs or the OLED display. I made sure to put the Humidity-Temperature sensor in a spot where it wouldn't be affected by the other parts of the car. I checked the place so that the measurements wouldn't be affected. My robot can check how hot or cold and how wet or dry the air is around it very accurately. This is possible because we put a special sensor in a spot where nothing gets in the way. We also put a buzzer in a place where noise can be heard easily. I thought about where to put it so that the sound wouldn't be blocked by other things. My robot can now make loud noises that people nearby can hear. I put the button, knob, and power switch in easy-to-reach places. I thought about where to put it so that it could be easily reached and used.

I used breadboards to neatly connect all the components that were needed. The breadboards helped hold and keep electronic parts in place safely on top of them. I arranged and connected the parts on the breadboards carefully, so they work correctly. I made my robot carefully using a stable base called a breadboard. I was very careful and paid attention to every detail so that it would work well.

I double-checked everything and thought carefully about how all the parts fit together. We wanted to make sure that the robot was strong and worked perfectly in every way. I made a robot that works well because I paid close attention to how I built it. It's strong and doesn't break down.

Tinder cad part

Tinkercad is a highly valuable tool that can be employed in the process of designing and prototyping robotic project components. Tinkercad designs with other CAD software can be seamlessly achieved due to the capacity to import and export models.

The picture below shows the parts on the Tinker card that are the same as the real parts.

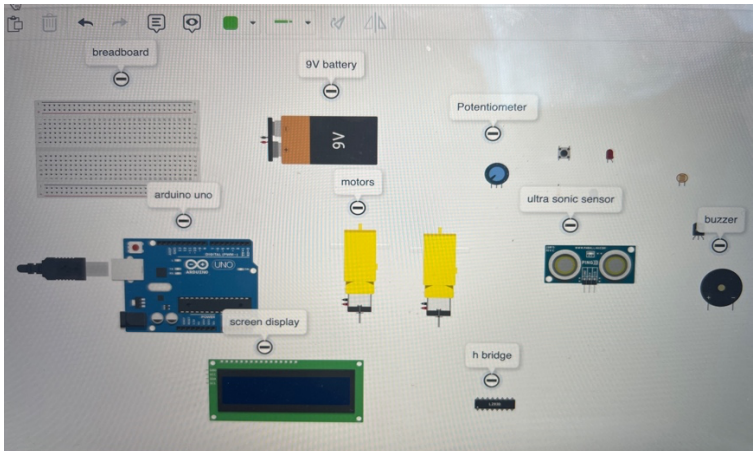


Figure 23

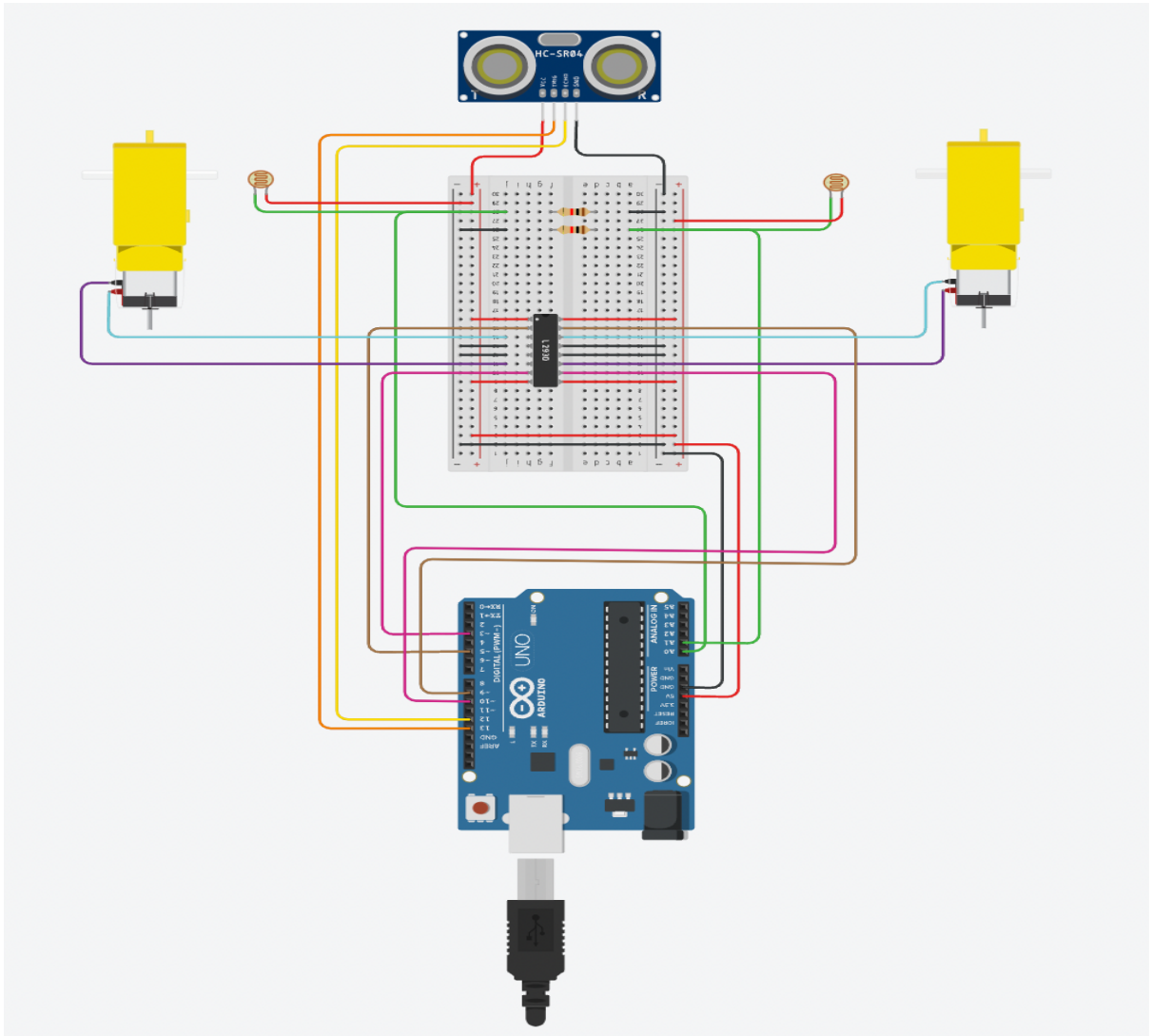


Figure 24

Figure 25 shows the Motor driver “L293D” connection for working the motors and ultra-sonic sensor.

The L293D is a type of H-bridge electronic component that allows voltage to be applied to a circuit in either direction. In our case, the L293D is used to power the Gearmotor in either direction.

Pin 1 is the enable pin of L293D, it was connected to the 5v power supply in the breadboard, connecting it to 5V sets the pin high and thus enables the left side of the L293D motor driver. Pin 2 is an input pin and is connected to pin 5 of the Arduino board which produces a pulse width modulated signal for motor speed control. Pin 3 is connected to the positive terminal of the Gearmotor as it is in an output pin. Both pins 4 and 5 are connected to the ground line in the breadboard to ground the motor driver and they also serve as a heat sink to the motor driver. Pin 6 is an output pin and is thus connected to the negative terminal of the Gearmotor. Pin 7 is an input pin and is thus connected to the pulse-modulated pin 5 on the Arduino board. When both pins 2 and 7 of the motor driver are set to HIGH or LOW the Gearmotor comes to a sudden stop. When pin 2 is set to HIGH while 7 is set to LOW the Gearmotor moves in the forward direction. When pin 2 is set to LOW and 7 is set to HIGH the Gearmotor moves in the backward direction.
Pin 8 of the motor driver is a power input pin thus connected to the power line in the breadboard, normally pin 8 is usually connected to a 9V external power supply to power the motors. Pin 9 is the enable pin of the right side and is hence connected to the 5v power line to activate the right side of the motor driver. Pins 10 and 14 are both input pins and are thus connected to the pulse width modulated Arduino pins 9 and 10. Pin 12 are 13 are ground pins and thus connected to the ground line in the breadboard, they also act as a heat sink. Pins 11 and 15 are output pins and are thus connected to the positive and the negative terminals of the Gearmotor.
Pin 16 is the input power pin for the motor driver and is thus connected to the 5V power line in the breadboard.

V. The process of programming the Bluno Nano microcontroller for the purpose of enabling robotic vehicle functions: -

My code for the autonomous light-finding-following robot is divided into several sections, each responsible for different functionalities. The main sections include setup, loop, shutdown, do Calibrate, calibrate, and make Threshold.

In the setup section, the microcontroller pins are configured for input and output. The screen library (Adafruit_SSD1306) and DHT sensor library (cactus_io_AM2302) are included, and the necessary objects for the screen and DHT sensor are initialized.

The loop section continuously executes the main logic of the robot. It reads and displays temperature and humidity values, reads LDR values, measures distance using the ultrasonic sensor, and performs light following based on the LDR values. Additionally, it checks for proximity to objects and initiates a shutdown if necessary.

The shutdown section stops all motors, outputs a shutdown message, and plays a tone to indicate the end of the program before stopping execution.

The doCalibrate section prompts the user to calibrate the robot. It allows the user to choose calibration options using the potentiometer and button, including wheel speed calibration, temperature offset calibration, humidity offset calibration, and threshold creation.

The calibrate section handles the calibration process for wheel speed, temperature offset, and humidity offset. It takes input from the potentiometer and saves the calibrated values for future use.

The makeThreshold section performs threshold creation, which is used for light detection. The robot is spun for a specified duration to measure the maximum LDR values, and the threshold is calculated based on these values

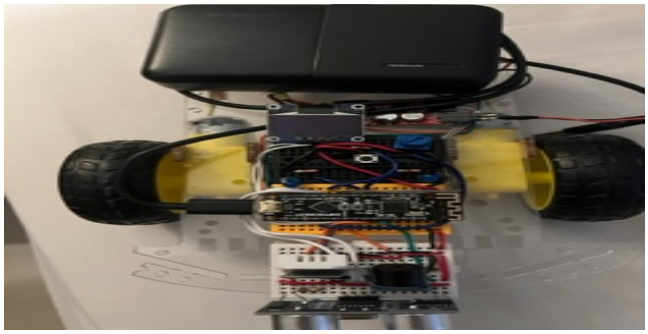


Figure 25

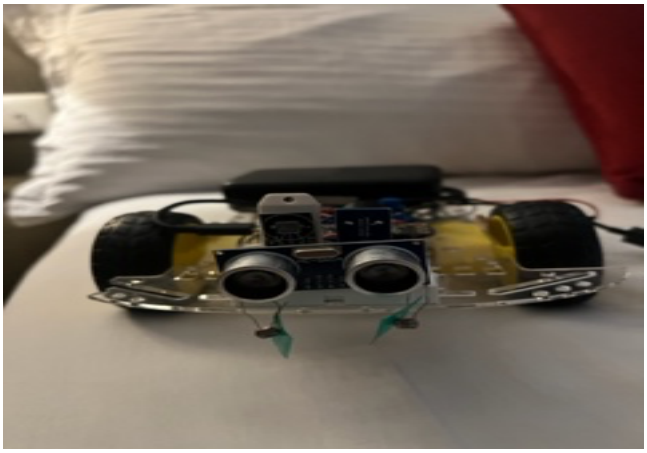


Figure 26

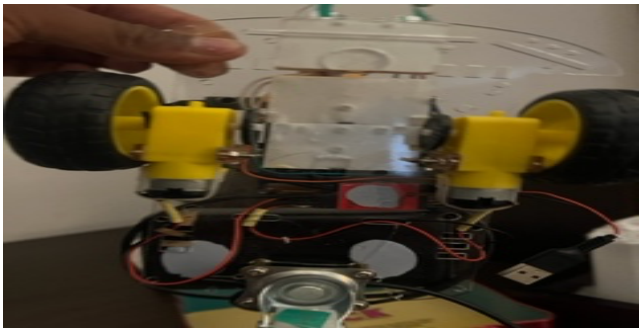


Figure 27

MATHEMATIC OPERATIONS;

I used range to calculate and take a value for the accuracy detecting of light in the room

The presentation of pertinent readings and occurrences on the Serial Monitor in Arduino IDE example code.

```
// Include necessary libraries

#include <Adafruit_SSD1306.h> // screen library
#include <cactus_io_AM2302.h> // dht sensor library

// analog
int Potentiometer = 1;
int Buzzer = 17;
int LeftLdr = 7;
int RightLdr = 6;

// digital
int LeftLed = 3;
int RightLed = 4;
int Button = 7;
int UltrasonicEcho = 9;
int UltrasonicTrigger = 8;
int RightMotorForward = 6;
int RightMotorBack = 5;
int LeftMotorForward = 11;
int LeftMotorBack = 10;

// calibration variables
int LeftWheelSpeed = 172;
int RightWheelSpeed = 172;
float TemperatureOffset = 0;
float HumidityOffset = 0;
int Threshold = 0;
```

Here I put all my values and **Serial.print()** and **Serial.println()** functions. This allows us to monitor the performance of the robot in real-time.

Calibration procedures: If our sensors or motors require calibration, we can display messages indicating the calibration procedures being performed.

```
void doCalibrate()
{
```

```
  delay(200); // wait for button to be released
  Serial.println(F("Starting Calibration"));
  while(true)
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Do Calibration?"));
    Display.setCursor(0, 20);
    Display.print(F("Choose using \npotentiometer \nThen press button")); // \n is new line
    Display.setCursor(0, 50);
    int potVal = analogRead(Potentiometer); // get current pot value
    if (potVal >= 512)  // choose based on pot value
      Display.print(F("Yes"));
    else if (potVal < 512)
      Display.print(F("No"));
    if (potVal >= 512 && digitalRead(Button) == HIGH)
    {   // do the calibration
      calibrate();
      break;  // leave the loop
    }
    else if (potVal < 512 && digitalRead(Button) == HIGH)
    {   // skip the calibration
      break;  // leave the loop
    }
    Display.display();
  }
  makeThreshold(); // threshold calibration needs to be done
}

void calibrate()
{
  delay(200); // wait for button to be released
  Serial.println("Starting Left Wheel Calibration");
  while(true) // start loop for left wheel
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Left Wheel Speed"));
    Display.setCursor(0, 20);
    Display.print(F("Choose wheel speed, \nthen press button"));
    Display.setCursor(0, 40); // move cursor below text
    Serial.println(); // print new line to serial monitor
```

```
  int potVal = map(analogRead(Potentiometer), 1, 1023, 0, 255); // Map pot value to be to motor speeds
  analogWrite(LeftMotorForward, potVal); // make motors go at selected speed
  Display.print(potVal);
  if (digitalRead(Button) == HIGH) // when the button is pressed, calibration is done
```

The current data derived from diverse sensors, including those that measure temperature, humidity, and distance can be displayed to enable the monitoring of these parameters' instantaneous values.

Velocity can be determined by analyzing sensor readings and displaying the corresponding calculated value. This procedure can prove to be advantageous in tracking the motion or pace of an object.

If errors or malfunctions are identified within the code, it is recommended to display error messages for the purpose of notifying the user and/or prompting necessary actions for resolution.

Scanning light

```
/ get ldr values
  int LeftLdrVal = analogRead(LeftLdr);
  int RightLdrVal = analogRead(RightLdr);
  // output ldr values to serial monitor
  Serial.print(F("L LDR: "));
  Serial.print(LeftLdrVal);
  Serial.print(F(" R LDR: "));
  Serial.print(RightLdrVal);
  // output ldr values to screen
  Display.print(F("L LDR "));
  Display.print(LeftLdrVal);
  Display.print(F(" R LDR "));
  Display.print(RightLdrVal);

  Display.setCursor(0, 32); // move cursor

  // get and output distance
  long duration, distance;
  digitalWrite(UltrasonicTrigger, LOW);
  delayMicroseconds(2);
  digitalWrite(UltrasonicTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(UltrasonicTrigger, LOW);
  duration = pulseIn(UltrasonicEcho, HIGH);
  distance = duration * 0.034 / 2;
```

```
 Serial.print(F(" Distance: "));
 Serial.print(distance);
 Display.print(F("Distance: "));
 Display.print(distance);

 Display.setCursor(0, 48); // move cursor

 // set all motors low
 analogWrite(LeftMotorForward, 0);
 analogWrite(LeftMotorBack, 0);
 analogWrite(RightMotorForward, 0);
 analogWrite(RightMotorBack, 0);
 Display.setCursor(0, 56);

 // check threshold values to light
 if(LeftLdrVal < Threshold && RightLdrVal < Threshold) // both less than the threshold
 { // go left
   analogWrite(LeftMotorBack, LeftWheelSpeed);
   analogWrite(RightMotorForward, RightWheelSpeed);
   digitalWrite(LeftLed, LOW);
   digitalWrite(RightLed, LOW);
   Display.print(F("Both less, left"));
 }
 else if(LeftLdrVal < Threshold && RightLdrVal > Threshold) // right side greather than the
threshold
 { // go right
   analogWrite(LeftMotorForward, LeftWheelSpeed);
   analogWrite(RightMotorBack, RightWheelSpeed);
   digitalWrite(LeftLed, LOW);
   digitalWrite(RightLed, HIGH);
   Display.print(F("Going Right"));
 }
 else if(LeftLdrVal > Threshold && RightLdrVal < Threshold) // left side greater than the
threshold
 { // go left
   analogWrite(LeftMotorBack, LeftWheelSpeed);
   analogWrite(RightMotorForward, RightWheelSpeed);
   digitalWrite(LeftLed, HIGH);
   digitalWrite(RightLed, LOW);
   Display.print(F("Going Left"));
 }
```

LDR sensor that is connected to a specific pin on the robot. The LDR sensor helps the robot read how bright the light is in the surrounding area. The program calculates the average brightness over a period of one second. While scanning, a light connected to pin 13 turns on to show that it's working. After checking everything with the scanner,

the light on the scanner turns off, and the normal reading from the sensor is shown on the Serial Monitor to see if there are any technical problems. The code waits for a little bit before scanning again.

DC MOTOR calibration

This code uses a potentiometer connected to pin A0 to set the speed of a DC motor connected to pin 9. The potentiometer reading is mapped to the range of motor speeds from 0 to 255 using the **map ()** function. The motor speed is set using the **analogWrite ()** function, and the current potentiometer value and motor speed are printed to the Serial Monitor for debugging purposes. The code then waits for a moment before taking the next reading.
To calibrate the motor, the user can adjust the potentiometer until the desired motor speed is achieved and note the corresponding potentiometer value. This value can then be used in the main program to set the motor speed based on user input or sensor readings.

DC motors control and Infrared speed sensors

The robot's movements were directed using a special sensor that could "see" where the brig htest light was. This allowed the robot to move towards the light source. The sensor used was called a LightDependent Resistor. Also, we used special sensors called Infrared (IR) speedsensorstomeasurehow fast the electric motors were spinning, how far they moved, and how fast they were going at that very moment. Here's how it was done using Arduino.

DC motor control on LDR

Infrared Speed Sensors for RPM, Distance, and Velocity Measurement

Passive buzzer

In order to enable the passive buzzer to generate diverse audio patterns
that correspond to individual events, the tone ()
function could be utilized within Arduino programming. The following is an illustrative code excerpt that showcases the process of generating a sequence of auditory signal s with a passive buzzer.

Exit program.

To implement the Exit program subroutine, you can use the distance measurement from the Ultrasonic sensor. Once the Robotic Vehicle is 8cm or less from the main light source, the

program should stop all motor movement and emit a sound pattern from the passive buzzer indicating the end of the program.

V. CONCLUSION

The Robotic Vehicle can follow a bright light by Light Dependent Resistors (LDRs). These sensors control the motors in the vehicle and help it move towards the light. We use Infrared Speed sensors to find out how fast each motor is going and how far it has traveled at that speed. The program can tell where there are lights around the vehicle and make a chart to show how bright they are. The program teaches how to adjust the DC motors and measure the temperature and humidity accurately. The Ultrasonic sensor measures how far things are, and the Display shows how fast things are moving and what the temperature and humidity are. The little lights show where the main light is and how far away it is. They also let you know when some things are happening, like when the system is being set up or when the program is finished. The buzzer makes different sounds for different things that happen. The robot car stops when it gets very close to a bright light and then does another task this is my project, I enjoyed it and completed a little bit modification can be done.

REFERENCES

1. Arduin, M., & Dudek, G. (2018). Mobile robot navigation and exploration using an infrared spotlight. Robotics and Autonomous Systems, 103, 39-49.
2. Babu, S. G., & Sankar, N. S. (2016). An autonomous light-following robot using fuzzy control for agricultural applications. International Journal of Computer Science and Mobile Computing, 5(11), 92-99.
3. Khan, S. A., & Lee, H. S. (2020). Autonomous line follower robot using Arduino with PID control. International Journal of Advanced Computer Science and Applications, 11(1), 266-271.
4. Li, Y., & Bai, S. (2020). Research on the control strategy of autonomous robot for light seeking. Journal of Physics: Conference Series, 1572(3), 032032.
5. Mahapatra, A., & Patra, A. (2019). Autonomous light following robot using Arduino. International Journal of Engineering Science and Computing, 9(9), 22248-22252.
6. Mahdavian, H., Amin, R. A., & Mahmood, N. H. (2019). Design and implementation of an autonomous light-seeking robot using PID control. Journal of Robotics, 2019, 1-10.
7. Mazumdar, A., Nag, B., & Hati, A. (2020). Development of an autonomous line follower robot using PID controller for industrial applications. International Journal of Innovative Technology and Exploring Engineering, 9(3), 472-477.

8. Raman, S., & Nain, S. (2018). Design and implementation of autonomous line follower robot with obstacle detection using Arduino. International Journal of Advanced Research in Computer Science and Software Engineering, 8(6), 350-355.
9. Reddy, R. S., & Swain, R. K. (2021). Design and implementation of autonomous light following robot using Arduino Uno. International Journal of Scientific Research and Reviews, 10(3), 4374-4382.
10. Sahoo, A. K., Ghosh, A., & Singh, R. (2017). Autonomous line follower robot using PID controller. International Journal of Electronics, Electrical and Computational System, 6(3), 2473-2476.

Appendix

```
/*
  l / r motors - d11 / d10 / d6 / d5 (PWM)
  ultrasonic sensor - d8 trigger / d9 echo
  screen - scl A5, sda A4
  l / r ldr - a7 / a6
  l / r leds - d3 / d4
  potentiometer - a1
  dht - d16 / a2
  button - d7
  buzzer - d17 / a3
*/

#include <Adafruit_SSD1306.h> // screen library
#include <cactus_io_AM2302.h> // dht sensor library

// analog
int Potentiometer = 1;
int Buzzer = 17;
int LeftLdr = 7;
int RightLdr = 6;

// digital
int LeftLed = 3;
int RightLed = 4;
int Button = 7;
int UltrasonicEcho = 9;
int UltrasonicTrigger = 8;
int RightMotorForward = 6;
int RightMotorBack = 5;
int LeftMotorForward = 11;
int LeftMotorBack = 10;
```

```
// calibration variables
int LeftWheelSpeed = 172;
int RightWheelSpeed = 172;
float TemperatureOffset = 0;
float HumidityOffset = 0;
int Threshold = 0;

// objects for screen, dht sensor
Adafruit_SSD1306 Display(128, 64, &Wire, -1);
AM2302 Dht(16);

void setup() {
  // start serial monitor
  Serial.begin(115200);
  // start screen
  if(!Display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  // set pin modes
  pinMode(Potentiometer, INPUT);
  pinMode(Button, INPUT);
  pinMode(LeftLdr, INPUT);
  pinMode(RightLdr, INPUT);
  pinMode(UltrasonicEcho, INPUT);
  pinMode(UltrasonicTrigger, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(LeftLed, OUTPUT);
  pinMode(RightLed, OUTPUT);
  pinMode(LeftMotorBack, OUTPUT);
  pinMode(LeftMotorForward, OUTPUT);
  pinMode(RightMotorBack, OUTPUT);
  pinMode(RightMotorForward, OUTPUT);

  // start temperature / humidity sensor
  Dht.begin();

  // set up the display
  Display.setTextSize(1);
  Display.setTextColor(WHITE);
  Display.setCursor(0, 0);
  Display.clearDisplay();

  // use a tone to show startup has finished
  Serial.println(F("Startup Procedure Finished"));
  tone(Buzzer, 440, 400);
  delay(400);
```

```
  noTone(Buzzer);

  // pause for 1 second after done
  delay(1000);
  // start calibration
  doCalibrate();
}

void loop() {
  // clear display before output
  Display.clearDisplay();
  Display.setCursor(0, 0);

  // write new line on serial monitor
  Serial.println();

  // read temperature and humidity with offsets
  Dht.readHumidity();
  Dht.readTemperature();
  int Temperature = Dht.temperature_C + (TemperatureOffset);
  int Humidity = Dht.humidity + (HumidityOffset);
  // print temperature and humidity to serial monitor
  Serial.print(F(" Temperature: "));
  Serial.print(Temperature);
  Serial.print(F(" Humidity: "));
  Serial.print(Humidity);
  // output temperature and humidity to screen
  Display.print(F("T: "));
  Display.print(Temperature);
  Display.print(F(" C "));
  Display.print(F("H: "));
  Display.print(Humidity);
  Display.print(F("%"));

  Display.setCursor(0, 16); // move cursor

  // get ldr values
  int LeftLdrVal = analogRead(LeftLdr);
  int RightLdrVal = analogRead(RightLdr);
  // output ldr values to serial monitor
  Serial.print(F("L LDR: "));
  Serial.print(LeftLdrVal);
  Serial.print(F(" R LDR: "));
  Serial.print(RightLdrVal);
  // output ldr values to screen
  Display.print(F("L LDR "));
  Display.print(LeftLdrVal);
  Display.print(F(" R LDR "));
```

```
  Display.print(RightLdrVal);

  Display.setCursor(0, 32); // move cursor

  // get and output distance
  long duration, distance;
  digitalWrite(UltrasonicTrigger, LOW);
  delayMicroseconds(2);
  digitalWrite(UltrasonicTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(UltrasonicTrigger, LOW);
  duration = pulseIn(UltrasonicEcho, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print(F(" Distance: "));
  Serial.print(distance);
  Display.print(F("Distance: "));
  Display.print(distance);

  Display.setCursor(0, 48); // move cursor

  // set all motors low
  analogWrite(LeftMotorForward, 0);
  analogWrite(LeftMotorBack, 0);
  analogWrite(RightMotorForward, 0);
  analogWrite(RightMotorBack, 0);
  Display.setCursor(0, 56);

  // check threshold values to light
  if(LeftLdrVal < Threshold && RightLdrVal < Threshold) // both less than the threshold
  { // go left
    analogWrite(LeftMotorBack, LeftWheelSpeed);
    analogWrite(RightMotorForward, RightWheelSpeed);
    digitalWrite(LeftLed, LOW);
    digitalWrite(RightLed, LOW);
    Display.print(F("Both less, left"));
  }
  else if(LeftLdrVal < Threshold && RightLdrVal > Threshold) // right side greather than the
threshold
  { // go right
    analogWrite(LeftMotorForward, LeftWheelSpeed);
    analogWrite(RightMotorBack, RightWheelSpeed);
    digitalWrite(LeftLed, LOW);
    digitalWrite(RightLed, HIGH);
    Display.print(F("Going Right"));
  }
  else if(LeftLdrVal > Threshold && RightLdrVal < Threshold) // left side greater than the
threshold
  { // go left
```

```
    analogWrite(LeftMotorBack, LeftWheelSpeed);
    analogWrite(RightMotorForward, RightWheelSpeed);
    digitalWrite(LeftLed, HIGH);
    digitalWrite(RightLed, LOW);
    Display.print(F("Going Left"));
  }
  else if(LeftLdrVal > Threshold && RightLdrVal > Threshold) // both sides greater than the
threshold
  { // go forward
    analogWrite(LeftMotorForward, LeftWheelSpeed);
    analogWrite(RightMotorForward, RightWheelSpeed);
    digitalWrite(LeftLed, HIGH);
    digitalWrite(RightLed, HIGH);
    Display.print(F("Going Forward"));
    // if going forward, check if less than 8 cm away
    if(distance < 8 && distance > 0)
    {
      Serial.println(); // write new line to serial monitor
      shutDown(); // start to shut down
    }
  }
  // output everything to the display
  Display.display();
}

void shutDown()
{ // shutting down
  // turn off all motors
  analogWrite(LeftMotorForward, 0);
  analogWrite(LeftMotorBack, 0);
  analogWrite(RightMotorForward, 0);
  analogWrite(RightMotorBack, 0);
  // output to serial monitor
  Serial.println(F("Shutdown initiated"));
  delay(100);
  tone(Buzzer, 293, 500); // Play a tone to signify the end of the program
  delay(500);
  noTone(Buzzer);
  delay(50);
  for(;;); // stop
}

void doCalibrate()
{
  delay(200); // wait for button to be released
  Serial.println(F("Starting Calibration"));
  while(true)
  {
```

```
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Do Calibration?"));
    Display.setCursor(0, 20);
    Display.print(F("Choose using \npotentiometer \nThen press button")); // \n is new line
    Display.setCursor(0, 50);
    int potVal = analogRead(Potentiometer); // get current pot value
    if (potVal >= 512)  // choose based on pot value
      Display.print(F("Yes"));
    else if (potVal < 512)
      Display.print(F("No"));
    if (potVal >= 512 && digitalRead(Button) == HIGH)
    {   // do the calibration
      calibrate();
      break;  // leave the loop
    }
    else if (potVal < 512 && digitalRead(Button) == HIGH)
    {   // skip the calibration
      break;  // leave the loop
    }
    Display.display();
  }
  makeThreshold(); // threshold calibration needs to be done
}

void calibrate()
{
  delay(200); // wait for button to be released
  Serial.println("Starting Left Wheel Calibration");
  while(true) // start loop for left wheel
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Left Wheel Speed"));
    Display.setCursor(0, 20);
    Display.print(F("Choose wheel speed, \nthen press button"));
    Display.setCursor(0, 40); // move cursor below text
    Serial.println(); // print new line to serial monitor
    int potVal = map(analogRead(Potentiometer), 1, 1023, 0, 255); // Map pot value to be to
motor speeds
    analogWrite(LeftMotorForward, potVal); // make motors go at selected speed
    Display.print(potVal);
    if (digitalRead(Button) == HIGH) // when the button is pressed, calibration is done
    {
      LeftWheelSpeed = potVal; // set the wheel speed
      Serial.print(F("Final left wheel speed: "));
```

```
      Serial.println(LeftWheelSpeed);
      analogWrite(LeftMotorForward, 0); // turn wheel off again after finishing
      break;  // leave the loop
    }
    Display.display();
  }
  delay(200); // wait for button to be released
  Serial.println("Starting Right Wheel Calibration");
  while(true) // start loop for right wheel
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Right Wheel Speed"));
    Display.setCursor(0, 20);
    Display.print(F("Choose wheel speed, \nthen press button"));
    Display.setCursor(0, 50); // move cursor below text
    Serial.println(); // print new line to serial monitor
    // Map pot value to be to motor speeds
    int potVal = map(analogRead(Potentiometer), 1, 1023, 0, 255); // Map pot value to be to
motor speeds
    analogWrite(RightMotorForward, potVal); // make motors go at selected speed
    Display.print(potVal);
    if (digitalRead(Button) == HIGH) // when the button is pressed, calibration is done
    {
      RightWheelSpeed = potVal;
      Serial.print(F("Final right wheel speed: "));
      Serial.println(RightWheelSpeed);
      analogWrite(RightMotorForward, 0); // turn wheel off again after finishing
      break;  // leave the loop
    }
    Display.display();
  }
  delay(200); // wait for button to be released
  Serial.println(F("Starting Temperature Calibration"));
  while(true) // start loop for temperature calibration
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Temperature Offset"));
    Display.setCursor(0, 20);
    Display.print(F("Choose temp \noffset, then press button")); // \n is new line
    Display.setCursor(0, 50); // move cursor below text
    float potVal = map(analogRead(Potentiometer), 1, 1023, -20, 20); // 2 degrees positive /
negative
    Serial.print(F("Current temperature offset: ")); // print to serial monitor
    Serial.println(potVal / 10); // divide value by 10 to make decimal number
```

```
    Display.print(potVal / 10);
    Display.print(F("      ")); // print a gap to make output more readable
    Dht.readTemperature(); // read current temperatue from sensor
    Display.print(Dht.temperature_C + (potVal / 10)); // output value with current offset
    if (digitalRead(Button) == HIGH) // when the button is pressed, calibration is done
    {   // Temperature is calibrated
      TemperatureOffset = potVal / 10;
      Serial.print(F("Final temperature offset: "));
      Serial.println(TemperatureOffset);
      break;  // leave the loop
    }
    Display.display();
  }
  delay(200); // wait for button to be released
  Serial.println(F("Starting Humidity Calibration"));
  while(true) // start loop for humidity calibration
  {
    // output to display
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Humidity Offset"));
    Display.setCursor(0, 20);
    Display.print(F("Choose humidity \noffset, then press button")); // \n is new line
    Display.setCursor(0, 50); // move cursor below text
    float potVal = map(analogRead(Potentiometer), 1, 1023, -20, 20); // 2% positive and
negative
    Serial.print(F("Current humidity offset: ")); // print to serial monitor
    Serial.println(potVal / 10);// divide value by 10 to make decimal number
    Display.print(potVal / 10);
    Display.print(F("      ")); // print a gap to make output more readable
    Dht.readHumidity(); // read current temperatue from sensor
    Display.print(Dht.humidity + (potVal / 10)); // output value with current offset
    if (digitalRead(Button) == HIGH) // when the button is pressed, calibration is done
    {
      HumidityOffset = potVal / 10;
      Serial.print(F("Final humidity offset: "));
      Serial.println(HumidityOffset);
      break;  // leave the loop
    }
    Display.display();
  }
}

void makeThreshold()
{
  delay(200); // wait for button to be released
  Serial.println(F("Starting Threshold Creation"));
  while(true)
```

```
  {
    Display.clearDisplay();
    Display.setCursor(0, 0);
    Display.print(F("Threshold Creation"));
    Display.setCursor(0, 20);
    Display.print(F("Place robot on the ground, \nthen press button"));
    if (digitalRead(Button) == HIGH) // wait for button to be pressed
    {
      Display.clearDisplay();
      break;  // leave the loop
    }
    Display.display();
  }
  long currTime = 0;
  currTime = millis(); // get the current time
  long checkTime = currTime;
  int LeftLdrMax = 1; // set to minimum value
  int RightLdrMax = 1;
  while(checkTime <= currTime + 4000) // check if 4 seconds have passed
  {
    checkTime = millis(); // get a new value to check the time against
    int LeftLdrVal = analogRead(LeftLdr); // get ldr values
    int RightLdrVal = analogRead(RightLdr);
    analogWrite(RightMotorForward, 255); // spin robot at full speed
    analogWrite(LeftMotorBack, 255);
    if (LeftLdrVal > LeftLdrMax) // save ldr value if greater than current max
      LeftLdrMax = LeftLdrVal;
    if (RightLdrVal > LeftLdrMax)
      LeftLdrMax = LeftLdrVal;
  }
  analogWrite(LeftMotorForward, 0); // stop spinning
  analogWrite(RightMotorBack, 0);
  // output highest values to serial monitor
  Serial.print(F("L LDR Highest: "));
  Serial.print(LeftLdrMax);
  Serial.print(F(" R LDR Highest: "));
  Serial.println(RightLdrMax);

  // calculate threshold
  Threshold = (LeftLdrMax + RightLdrMax / 2) * 0.8; // average the two maxes, then make a
bit smaller
  delay(200); // wait a moment after finishing
}
```