Date: 08 Dec. 2021

Project Name: Height based sorting station with integrated Line follower robot and Robotic arm

Project Description:

This is an object sorting mechanism where different objects incoming continuously on a conveyor belt are sorted and diverted to different boxes based on their heights. Also the number of objects of each different size category is counted and recorded.
This project provides a miniature working model of a whole sorting station where objects of multiple varying sizes are arriving on the conveyor belt. Different objects are having different uses based on their sizes, thus the sorting mechanism sorts them into category of small, medium and large.
A robotic arm depicts the model of an Industrial robot for pick and place operation that takes the large sized objects to the next workstation for further manufacturing operations.
Medium sized objects are dropped at 180 degree into another box by the sorting and dumper mechanism.
The small sized objects are dropped into the line follower robot depicting an AGV in a manufacturing facility that is programmed to follow a predefined path and take the objects from the conveyor to the destination location and then come back and repeat. This AGV (Line follower) receives a signal via Bluetooth as soon as it is filled to its full capacity, and then it starts the travel on its predefined path across the manufacturing facility. AGVs are very widely used in almost all large production and storage facilities across the world, and so does Robotic manipulator arm.
Thus this model simulates a whole automated sorting station integrated with an Automated Guiding Vehicle and a Industrial robot manipulator.

Working Operation:

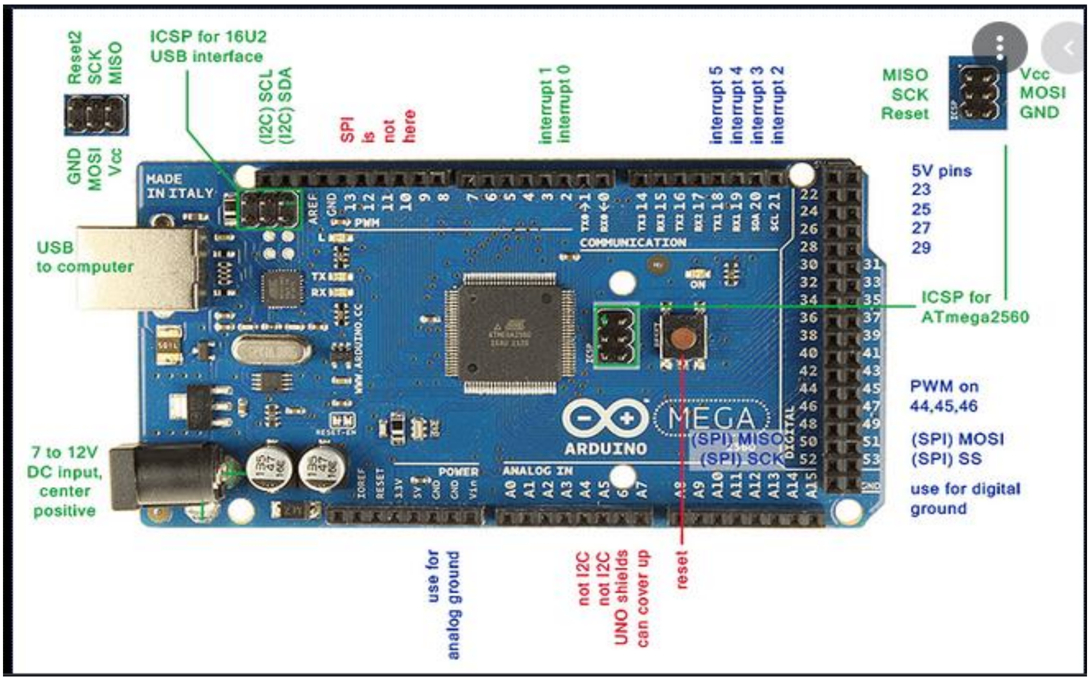
Two 200rpm geared DC motors are used to operate a conveyor belt on which differently sized objects are coming continuously in random manner. An ultrasonic sensor is used to sense the arrival of an object on the sorting station, and another ultrasonic sensor is used to measure the height of objects coming toward the sensor on conveyor belt. Different threshold values are set to identify the object as Small, Medium and Large and keep the count of them separately.
A LCD Display is used to display the number of object of each different size passing through the sensor.
After that a sorting mechanism with servo motor and a robotic arm will be used to sort the objects and divert them to different boxes based on their sizes respectively.

Sequence of operation:

- The conveyor belt is continuously operating, and objects of different sizes are coming on it in random order.
- As soon as an object reaches the sorting station, an Ultrasonic sensor will sense its presence and the conveyor belt will stop.
- Then another ultrasonic sensor places on top will measure its height, and sort it in the category of Small, Medium, Large based on the set threshold height value.
- After identifying the object as Small, Medium or Large, if it is Large, a robotic arm will pick the object and place in a different box.
- For small and medium, a servo motor will divert them to different channels at 90degree and 180 degree angles respectively.
- Then another 180degree servo motor will drop the object into the respective box, and the first servo will return to original position.
- Small object will be dropped in the container mounted on a line follower robot (Automated guided Vehicle). Medium size object will be dropped in another box.

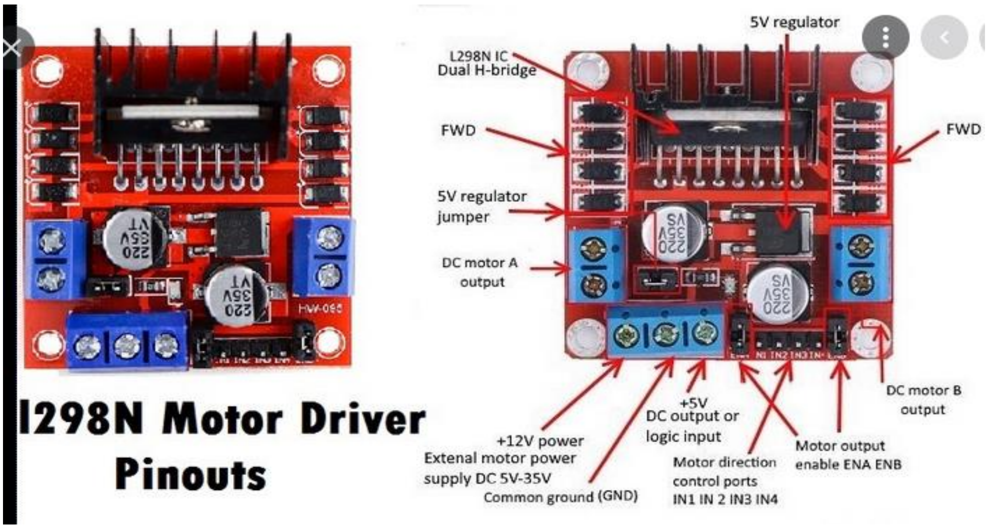
- After the servo reaches its original position, the conveyor will start moving again, and the Ultrasonic sensor will start checking for the presence of another object arriving on the sorting station.
- As soon as another object's presence in the sorting station is detected, the conveyor will stop, and the above cycle of sequence will start repeating.
- Meanwhile, an OLED display will keep showing the count of Small, Medium and Large sized objects that passed through the system, and also the absolute height of the object presently in the sorting station.
- As soon as the number of Small object reaches 5, ie when the container mounted on line follower is full, Arduino mega in the main sorting station will send a signal to line follower robot via Bluetooth module.
- Line follower will start to follow the pre-laid path, reach the destination, wait for 5 seconds for unloading, and start travelling on pre-laid path again to reach the previous position for further loading.
- After some time delay for line follower to complete the loop and return, the conveyor will start again.
- This whole cycle will keep on continuing until power is cut off.

Major Components:

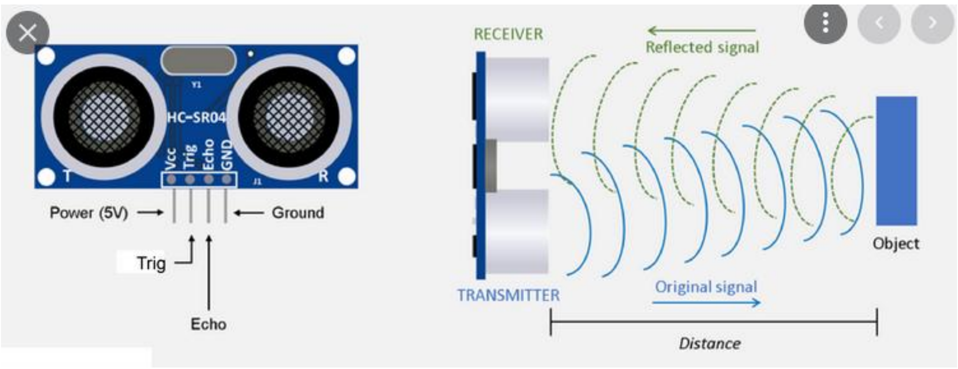


Arduino Mega

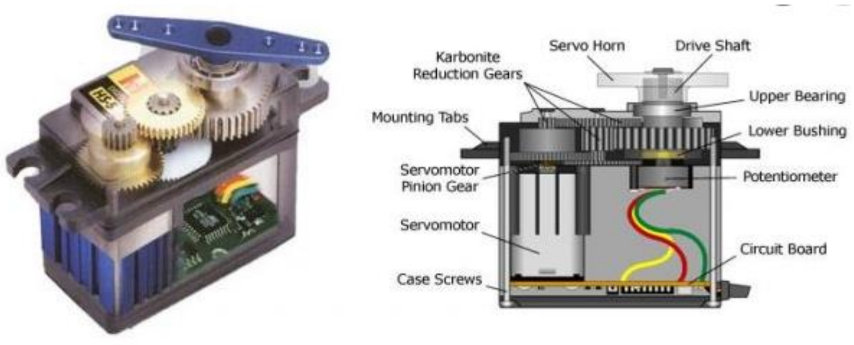
Date: 08 Dec. 2021


l298N Motor Driver Pinouts
5V regulator
L298N IC Dual H-bridge
FWD
5V regulator jumper
DC motor A output
FWD
DC motor B output
+12V power
Extenal motor power supply DC 5V-35V
Common ground (GND)
+5V DC output or logic input
Motor direction control ports IN1 IN 2 IN3 IN4
Motor output enable ENA ENB
HW-095

L298N motor driver

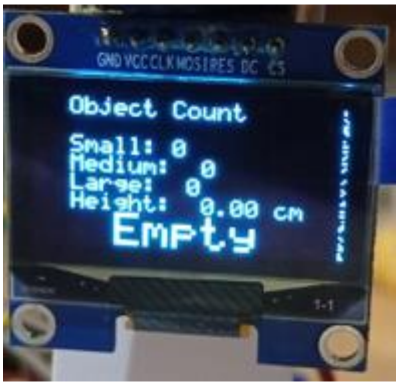


Ultrasonic Sensor HC-SR04



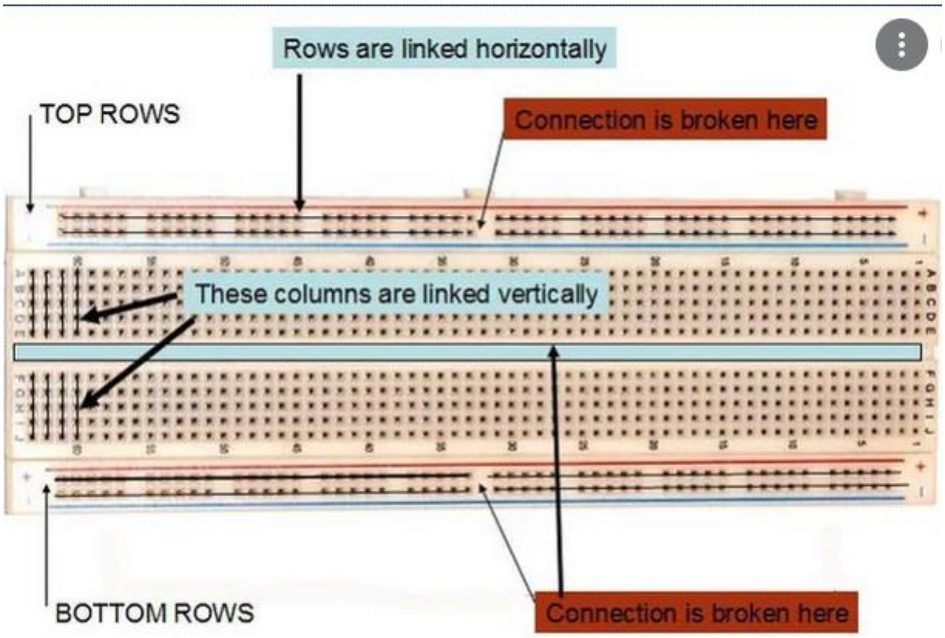
Metal gear Servo motor: 180 degree and 360 degree

Date: 08 Dec. 2021



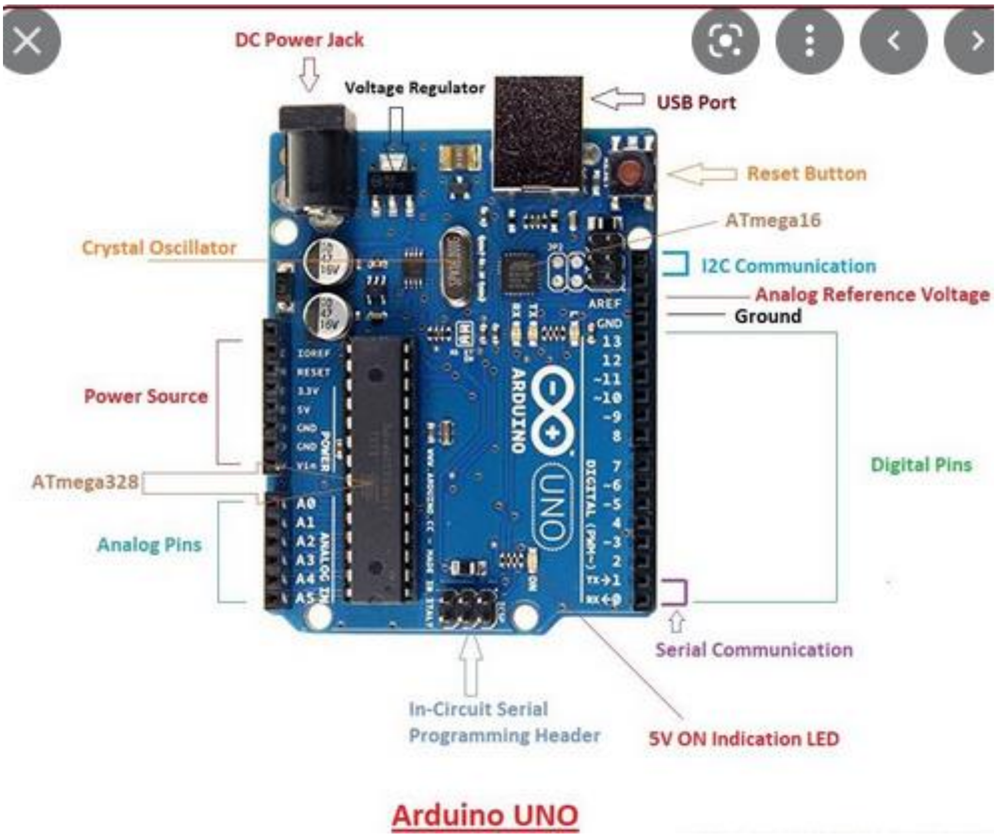
OLED Display

200 RPM DC geared motors and wheels

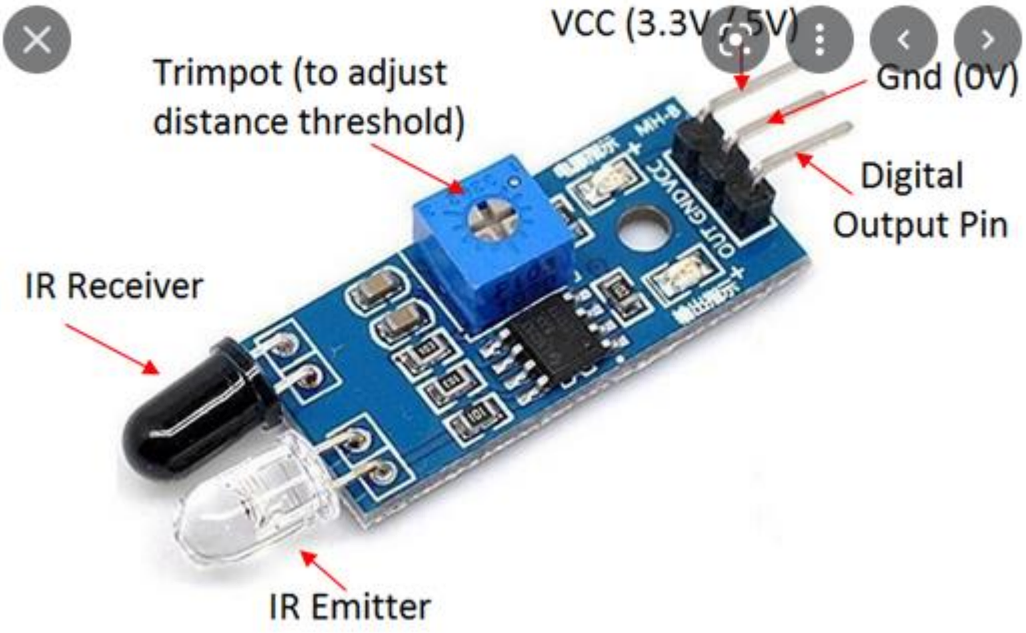

Rows are linked horizontally
TOP ROWS
Connection is broken here
These columns are linked vertically
BOTTOM ROWS
Connection is broken here

Bread Board

Date: 08 Dec. 2021

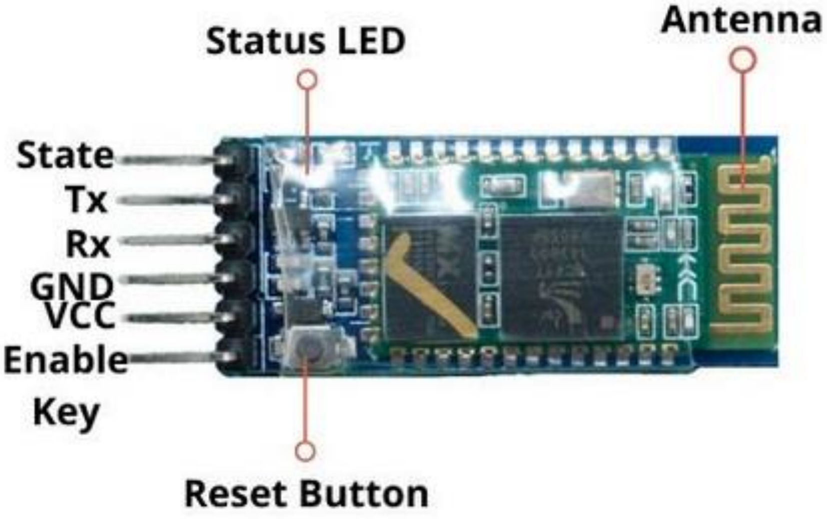


Arduino Uno



IR sensor module

Date: 08 Dec. 2021



HC-05 Bluetooth module

Date: 08 Dec. 2021

Connection Diagram:

Arduino mega on main sorting station:

Arduino Mega

PIN:	Connected to:
2	Motor driver (EN1)
3	Motor driver (EN2)
4	Sorter mechanism base servo (PWM)
5	Sorter mechanism dumper servo (PWM)
6	Arm base servo (PWM)
7	Arm link servo (PWM)
8	Arm gripper servo (PWM)
22	Proximity ultrasonic sensor (ECHO)
23	Proximity ultrasonic sensor (TRIG)
26	Height ultrasonic sensor (ECHO)
27	Height ultrasonic sensor (TRIG)
28	Motor driver (IN2)
29	Motor driver (IN1)
30	Motor driver (IN3)
31	Motor driver (IN4)
36	OLED (CLK)
37	OLED (MOSI)
38	OLED (DC)
39	OLED (CS)
40	OLED (RST)
44	Bluetooth (TX)
45	Bluetooth (RX)
5V output	Breadboard
GND	Breadboard

L298N motor driver

L298n motor driver

Pin	connected to
EN1	2
EN2	3
IN1	29
IN2	28
IN3	30
IN4	31
M1	motor1 +ve
M2	motor1 -ve
M3	motor2 +ve
M4	motor2 -ve
GND	GND
12Vin	+12V from 12V adapter

DC motor for conveyor

Pin	Pin on L298N
(motor 1)	
1	IN1
2	IN2
(motor 2)	
1	IN3
2	IN4

Conveyor Belt

DC motors for conveyor

Date: 08 Dec. 2021

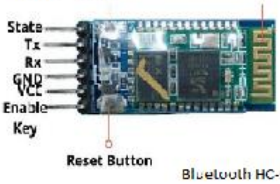
OLED Display

Pin	Connected to pin on arduino mega
CLK	36
MOSI	37
RES	40
DC	38
CS	39
VCC	+5V on breadboard
GND	GND on breadboard

OLED Display

Bluetooth

Pin	Pin on mega
TX	44
RX	45
VCC	+5V
GND	GND

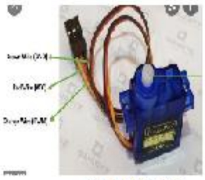


Bluetooth HC-05

Robotic arm

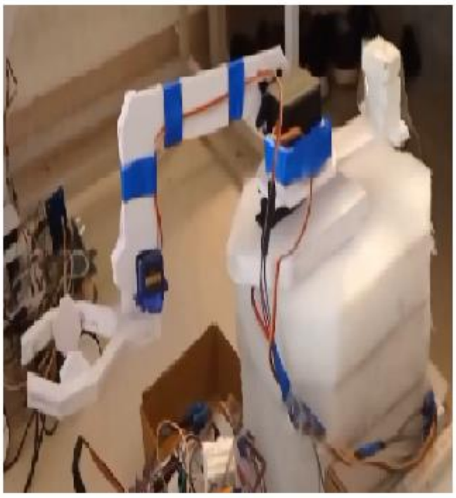
Pin	Pin on mega
(Gripper)	
PWM	8
VCC	+5V
GND	GND
(Link)	
PWM	7
VCC	+5V
GND	GND
(Base)	
PWM	6
VCC	+5V
GND	GND

Arm Gripper Servo



Arm Link Servo

Arm Base Servo



Robotic Arm

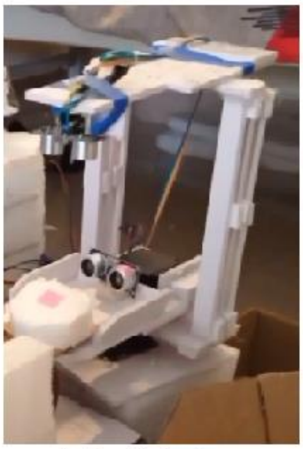
Sorting mechanism

(Ultrssonic sensors)	
(Proximity)	
ECHO	22
TRIG	23
VCC	+5V
GND	GND
(Height)	
ECHI	26
TRIG	27
VCC	+5V
GND	GND

Pin	Pin on mega
(Base Servo)	
PWM	4
VCC	+5V
GND	GND
(Dumper servo)	
PWM	5
VCC	+5V
GND	GND

Dumper servo

Base servo



Sorting Mechanism

Ultrasonic sensor (Height)

Ultrasonic sensor (Proximity)

Date: 08 Dec. 2021

Line follower robot connection diagram:

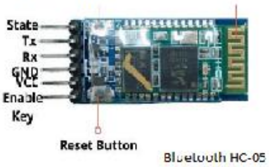
Arduino UNO

Arduino UNO

Pin	Connected to
2	IR sensor (middle) (OUT)
3	Bluetooth (RX)
4	Bluetooth (TX)
5	Motor driver (EN1)
6	Motor driver (IN2)
7	Motor driver (IN1)
8	Motor driver (IN3)
9	Motor driver (IN4)
10	Motor driver (EN2)
12	IR sensor (right) (OUT)
13	IR sensor (left) (OUT)
5V	To breadboard
GND	to breadboard

Bluetooth

Pin	Pin on mega
TX	4
RX	3
VCC	+5V
GND	GND



Bluetooth HC-05

IR sensors

Pin	Conneted to
(left)	
OUT	13
VCC	+5V
GND	GND
(Middle)	
OUT	2
VCC	+5V
GND	GND
(Right)	
OUT	12
VCC	+5V
GND	GND

Left IR sensor

Middle IR sensor

Right IR sensor

L298N motor driver

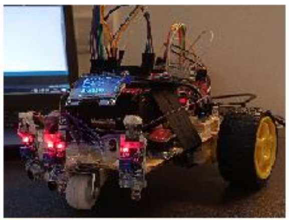
L298n motor driver

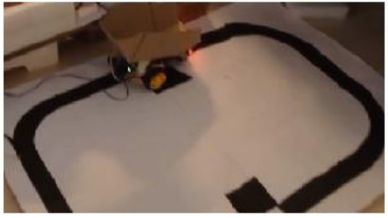
Pin	connected to
EN1	5
EN2	10
IN1	7
IN2	6
IN3	8
IN4	9
M1	motor1 +ve
M2	motor1 -ve
M3	motor2 +ve
M4	motor2 -ve
GND	GND
12Vin	+12V from 12V adapter

DC motor for conveyor

Pin	Pin on L298N
(motor 1)	
1	IN1
2	IN2
(motor 2)	
1	IN3
2	IN4

DC motors



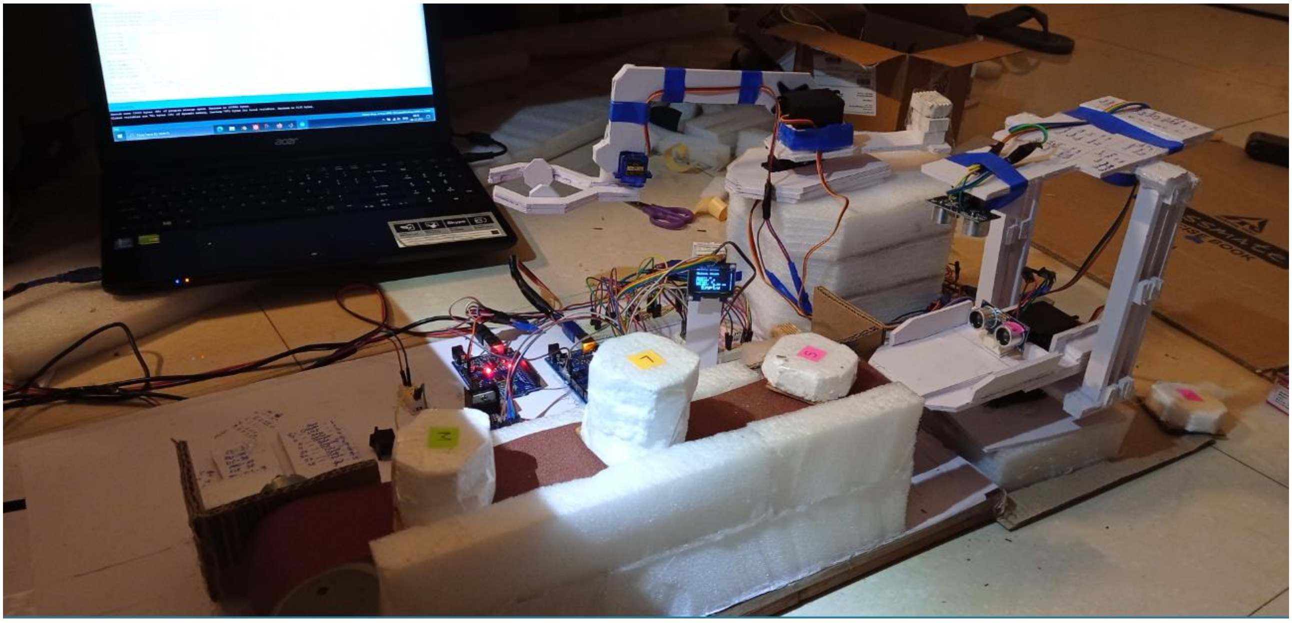


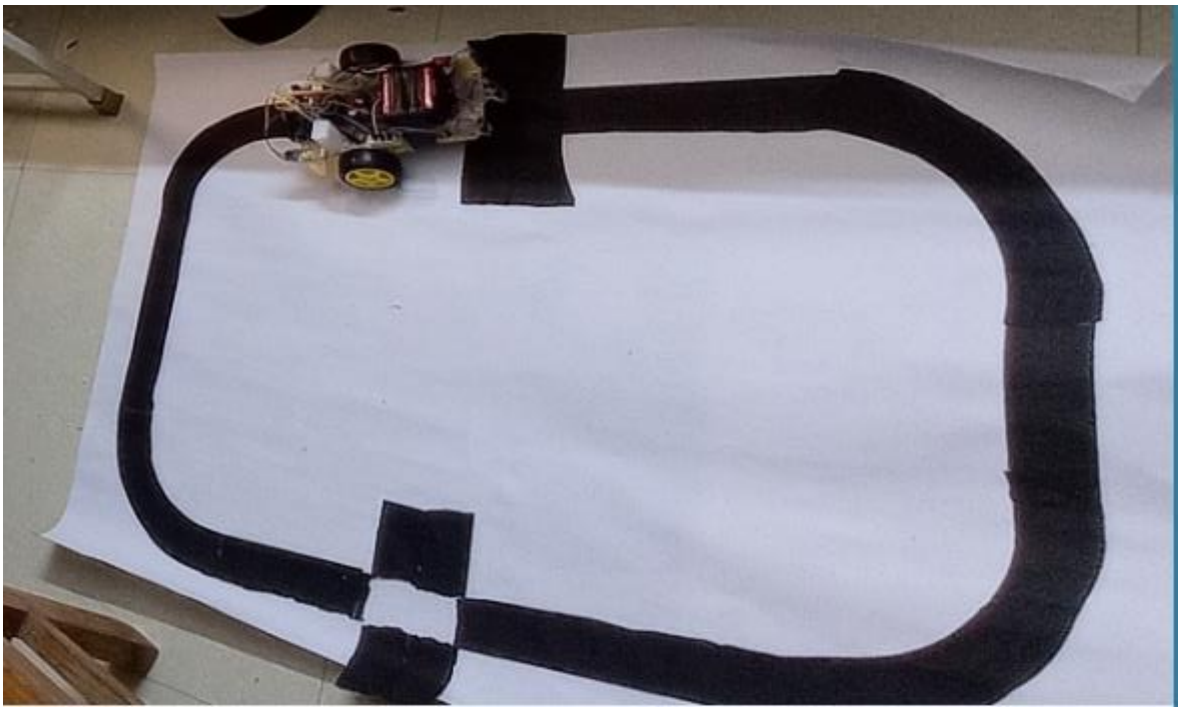
Line follower

Date: 08 Dec. 2021

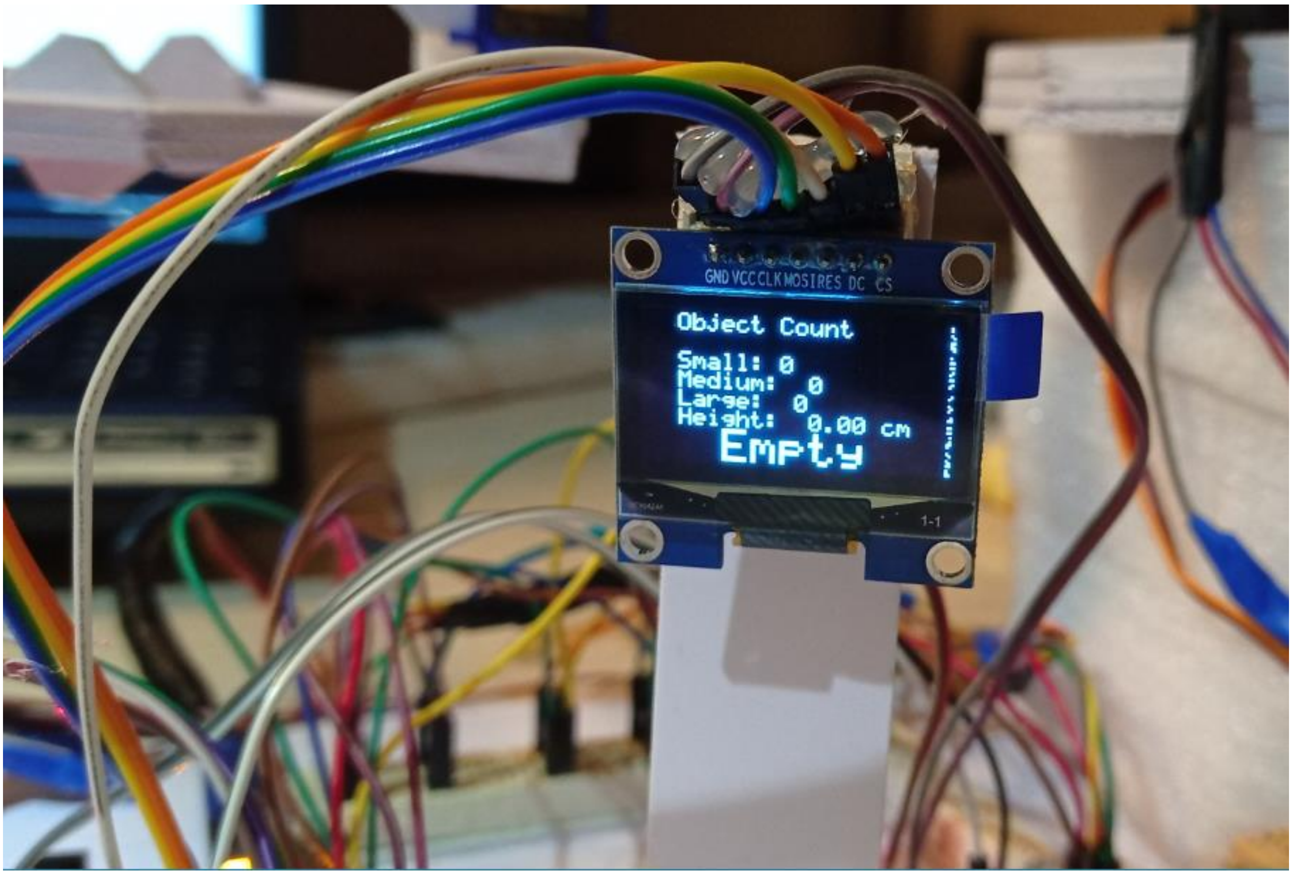
Project Model:

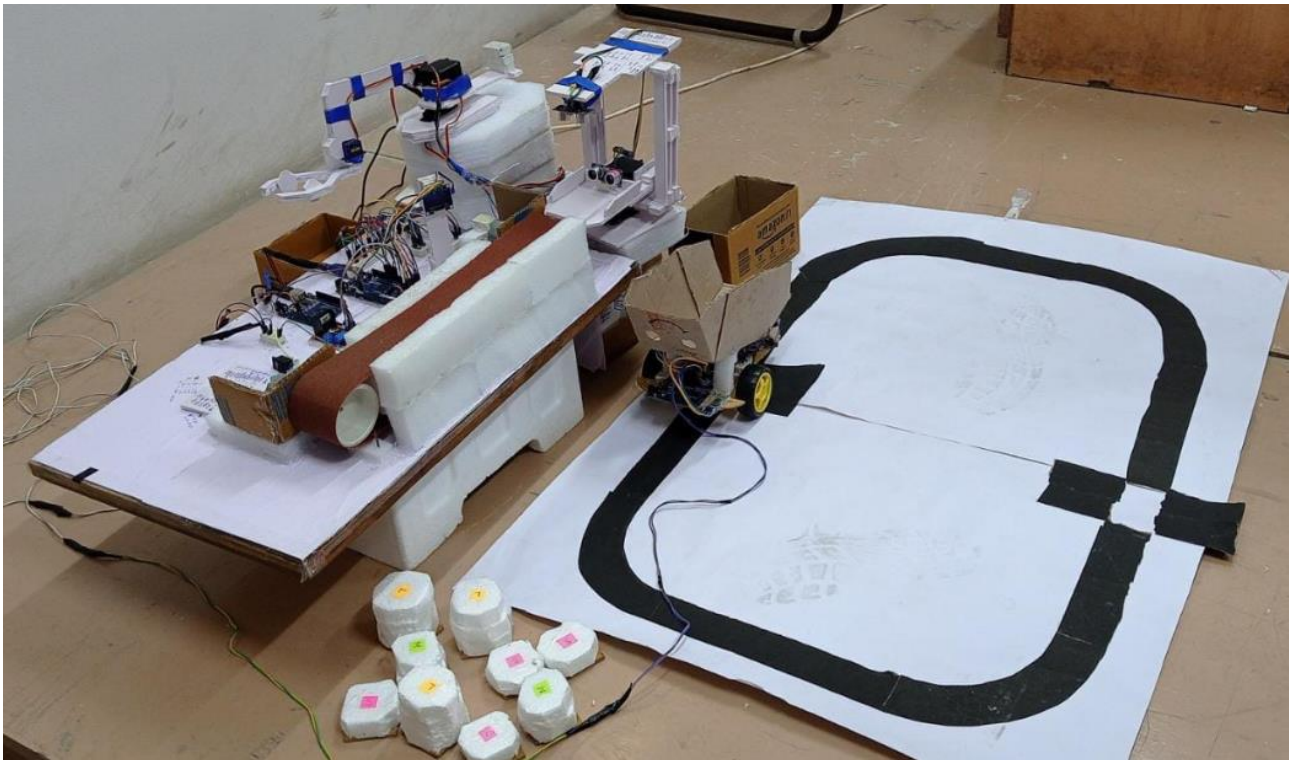
Final Project:





Date: 08 Dec. 2021





Date: 08 Dec. 2021

Codes used in the project:

Sorting station Arduino code for Arduino mega:

```
// Simple software SPI test for ebay 128x64 oled.

#include "SSD1306Ascii.h"
#include "SSD1306AsciiSoftSpi.h"

#include <Servo.h>

#include <SoftwareSerial.h>
SoftwareSerial Bluetooth(44, 45); // RX, TX  connect Bluetooth RX to 45, TX to 44

//defining ultrasonc sensor pins
#define echoPin_proximity 22
#define trigPin_proximity 23
#define echoPin_height 26
#define trigPin_height 27

// pin definitions OLED
#define CS_PIN    39
#define RST_PIN   40
#define DC_PIN    38
#define MOSI_PIN 37
#define CLK_PIN  36

SSD1306AsciiSoftSpi oled;

//for Line Following Robot using 2IR sensors
int lm1 = 29; //left motor output 1
int lm2 = 28; //left motor output 2
int rm1 = 30; //right motor output 1
int rm2 = 31; //right motor output 2
int sprm = 3; //motor speed
int splm = 2;
int defspeed = 120; //default speed
int mspeed = defspeed;
int conveyor_motor_delay = 12;
int conveyor_motor_stop_delay = 10;

// defines variables for ultrasonic sensor
long duration_proximity; // variable for the duration of sound wave travel
float distance_proximity; // variable for the distance measurement
long duration_height;
float distance_height;

//defining thresholds for ultrasonic
float object_on_conveyor = 42;
float object_detected = 4.5;
float small = 2.85;
float medium = 5.5;
float large = 7.5;
float base_height = 11.2;
float object_height = 0;

//define servo motors
Servo servo_base;
Servo servo_dumper;
Servo robo_arm_base;
```

Date: 08 Dec. 2021

```
Servo robo_arm_link;
Servo robo_arm_gripper;
int pos = 0;
int servo_speed_delay = 5; //control speed of sorting servo
int arm_speed_delay = 8; //control speed of arm


//inventory of detected objects
int num_small = 0;
int num_medium = 0;
int num_large = 0;

int truck_capacity = 3;
int truck_busy = 0;
char truck_task_status = 'x'; //set to z when line follower completes action

void setup()
{
  pinMode(trigPin_proximity, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin_proximity, INPUT); // Sets the echoPin as an INPUT
  pinMode(trigPin_height, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin_height, INPUT); // Sets the echoPin as an INPUT

  pinMode(lm1, OUTPUT); //conveyor motor pins
  pinMode(lm2, OUTPUT);
  pinMode(rm1, OUTPUT);
  pinMode(rm2, OUTPUT);
  pinMode(splm, OUTPUT);
  pinMode(sprm, OUTPUT);

  servo_base.attach(4);
  servo_dumper.attach(5);
  robo_arm_base.attach(6);
  robo_arm_link.attach(7);
  robo_arm_gripper.attach(8);

  Serial.begin(9600); // Serial Communication is starting with 9600 of baudrate speed
  //Serial1.begin(9600); //initiate bluetooth
  Bluetooth.begin(9600);
  delay(200);
  conveyor_stop();
  delay(500);
  initiate_servo();
  delay(750);


  //Initiate OLED display
  // Use next line if no RST_PIN or reset is not required.
  // oled.begin(&Adafruit128x64, CS_PIN, DC_PIN, CLK_PIN, MOSI_PIN);
  oled.begin(&Adafruit128x64, CS_PIN, DC_PIN, CLK_PIN, MOSI_PIN, RST_PIN);
  oled.setFont(System5x7);
  uint32_t m = micros();
  oled_display();
  oled.set2X();
  oled.println("   Empty");

  delay(400);
}


void loop()
{
```

```
check_proximity();
initiate_conveyor_speed();

if (distance_proximity > object_detected)
{
  conveyor_forward();
  delay(conveyor_motor_delay);
  conveyor_stop();
  delay(conveyor_motor_stop_delay);
}

if (distance_proximity <= object_detected)
{
  conveyor_stop();
  check_height();
  delay(2000);

  if (object_height <= small)
  {
    num_small = num_small + 1;
    oled_display();
    oled.set2X();
    oled.println("   Small");
    servo_90();
    delay(1500);
    servo_dumping_action();
    delay(1500);
    oled_display();
    oled.set2X();
    oled.println("   Empty");
    servo_90_return();
    delay(1500);
    if (num_small > 0  &&  num_small % truck_capacity == 0)
    {
      Bluetooth.write("i");
      //truck_busy = 1;
      conveyor_stop();
      oled_display();
      oled.set2X();
      oled.println("   BUSY");
      delay(20000);//waiting fro line follower to finish
      oled_display();
      oled.set2X();
      oled.println("   Empty");
      conveyor_forward();
    }
  }
  if (object_height > medium)
  {
    num_large = num_large + 1;
    oled_display();
    oled.set2X();
    oled.println("   Large");
    delay(500);
    robotic_arm_action();
    oled_display();
    oled.set2X();
    oled.println("   Empty");
    delay(500);
    conveyor_forward();
  }
  if (object_height > small && object_height <= medium)
  {
      num_medium = num_medium + 1;
      oled_display();
      oled.set2X();
      oled.println("   Medium");
      servo_180();
      delay(2000);
      servo_dumping_action();
      delay(2000);
      oled_display();
      oled.set2X();
      oled.println("   Empty");
      servo_180_return();
      delay(2000);
      conveyor_forward();
    }
  }
}

void check_height()
{
  digitalWrite(trigPin_height, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin_height, LOW);
  duration_height = pulseIn(echoPin_height, HIGH);
  distance_height = duration_height * 0.034 / 2;
  object_height = base_height - distance_height;
  Serial.print("  |  Object_height: ");
  Serial.print(distance_height);
  Serial.print("     ");
  Serial.print(object_height);
  Serial.print(" cm   ");
}

void check_proximity()
{
  // Clears the trigPin condition
  digitalWrite(trigPin_proximity, LOW);
  digitalWrite(trigPin_height, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin_proximity, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin_proximity, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration_proximity = pulseIn(echoPin_proximity, HIGH);
  // Calculating the distance
  distance_proximity = duration_proximity * 0.034 / 2;// Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance_proximity: ");
  Serial.print(distance_proximity);
  Serial.print(" cm   ");
  Serial.println();
}

void initiate_conveyor_speed()
{
  // set speed
  analogWrite(splm, defspeed); //full speed at 255
  analogWrite(sprm, defspeed);
}

void conveyor_stop()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, LOW);
}
void conveyor_forward()
{
  digitalWrite(lm1, HIGH);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, HIGH);
  digitalWrite(rm2, LOW);
}
void conveyor_backward()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, HIGH);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, HIGH);
}
void initiate_servo()
{
  servo_base.write(0);
  servo_dumper.write(0);
  robo_arm_link.write(15);
  robo_arm_base.write(90);
  robo_arm_gripper.write(180); //gripper close
}
void servo_90()
{
  for (pos = 0; pos <= 90; pos += 1)
  {
    servo_base.write(pos);
    delay(servo_speed_delay);
  }
  pos = 0;
}
void servo_180()
{
  for (pos = 0; pos <= 180; pos += 1)
  {
    servo_base.write(pos);
    delay(servo_speed_delay);
  }
  pos = 0;
}
void servo_180_return()
{
  for (pos = 180; pos >= 0; pos -= 1)
  {
    servo_base.write(pos);
    delay(servo_speed_delay);
  }
  pos = 0;
}
void servo_90_return()
{
  for (pos = 90; pos >= 0; pos -= 1)
  {
    servo_base.write(pos);
    delay(servo_speed_delay);
  }
  pos = 0;
}
```

Date: 08 Dec. 2021

```
void servo_dumping_action()
{
  for (pos = 0; pos <= 27; pos += 1)
  {
    servo_dumper.write(pos);
    delay(5);
  }
  delay(1500);
  for (pos = 27; pos >= 0; pos -= 1)
  {
    servo_dumper.write(pos);
    delay(5);
  }
  pos = 0;
}
void robotic_arm_action()
{
  delay(400);
  robo_arm_gripper.write(140); //gripper open
  delay(500);
  for (pos = 15; pos <= 50; pos += 1)  //robo link move up
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  for (pos = 90; pos <= 158; pos += 1) //robo base move half toward pick-up
  {
    robo_arm_base.write(pos);
    delay(arm_speed_delay);
  }
  delay(750);
  for (pos = 50; pos >= 15; pos -= 1) //robo link half down
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  for (pos = 158; pos <= 180; pos += 1) //robo base reach pick-up
  {
    robo_arm_base.write(pos);
    delay(arm_speed_delay);
  }
  delay(750);
  for (pos = 15; pos >= 0; pos -= 1) //robo link reach object
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  robo_arm_gripper.write(180); //gripper close
  delay(750);
  for (pos = 0; pos <= 15; pos += 1) //robo link move half up
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  for (pos = 180; pos >= 158; pos -= 1) //robo base leave pick-up
  {
    robo_arm_base.write(pos);
    delay(arm_speed_delay);
  }
  delay(750);
  for (pos = 15; pos <= 50; pos += 1) //robo link move up
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  for (pos = 158; pos >= 0; pos -= 1) //robo base reaches drop
  {
    robo_arm_base.write(pos);
    delay(arm_speed_delay);
  }
  delay(1500);
  for (pos = 50; pos >= 0; pos -= 1) //robo link move down
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  robo_arm_gripper.write(140); //gripper open
  delay(750);
  for (pos = 0; pos <= 50; pos += 1) //robo link move up
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  delay(500);
  for (pos = 0; pos <= 90; pos += 1) //robo base reaches initial
  {
    robo_arm_base.write(pos);
    delay(arm_speed_delay);
  }
  delay(750);
  for (pos = 50; pos >= 15; pos -= 1) //robo link reaches initial
  {
    robo_arm_link.write(pos);
    delay(arm_speed_delay);
  }
  robo_arm_gripper.write(180); //gripper close
  delay(400);
  pos = 0;
}
void oled_display()
{
  oled.clear();
  oled.set1X();
  oled.println("   Object Count\n");
  oled.print("   Small: ");
  oled.println(num_small);
  oled.print("   Medium:  ");
  oled.println(num_medium);
  oled.print("   Large:  ");
  oled.println(num_large);
  oled.print("   Height:  ");
  oled.print(object_height);
  oled.println(" cm");
}
```

Date: 08 Dec. 2021

Line follower robot code:

```
#include <SoftwareSerial.h>
//for Line Following Robot using 2IR sensors
int lm1 = 7; //left motor output 1
int lm2 = 6; //left motor output 2
int rm1 = 8; //right motor output 1
int rm2 = 9; //right motor output 2
int sl = 13;  //sensor 1 input (left)
int sr = 12;  //sensor 2 input (right)
int smid = 2;  //sensor middle input (middle)
int SlV = 0;
int SrV = 0;
int SmidV = 0;
int led = A0;
int sprm = 10; //motor speed
int splm = 5;
int bt = 0; //trigger to initiate line follower
int defspeed = 150; //default speed
int mspeed = defspeed;
int line_follower_delay = 10;
int line_follower_stop_delay = 5;
SoftwareSerial Bluetooth(4, 3); // RX, TX  connect Bluetooth RX to 3, TX to 4

//send 'z' when action complete
void setup()
{
  pinMode(lm1, OUTPUT);
  pinMode(lm2, OUTPUT);
  pinMode(rm1, OUTPUT);
  pinMode(rm2, OUTPUT);
  pinMode(splm, OUTPUT);
  pinMode(sprm, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(sl, INPUT);
  pinMode(sr, INPUT);
  pinMode(smid, INPUT);
  Serial.begin(9600);
  Bluetooth.begin(9600);
  sTOP();
  read();
}
void loop()
{
  // set speed
  analogWrite(splm, defspeed); //full speed at 255
  analogWrite(sprm, defspeed);

  char c;

  if (Bluetooth.available())
  {
    c = Bluetooth.read();
    switch (c)
    {
      case 'w':   //move forward
        ForWard();
        break;

      case 's': //move Backward
        BackWard();
        break;
      case 'a': //move left
        slowLeft();
        break;

      case 'd': //move right
        slowRight();
        break;

      case 'q': //move left sharp
        Left();
        break;

      case 'e': //move right sharp
        Right();
        break;

      case 'o': //STOP
        sTOP();
        bt = 0;
        break;

      case 'n': //reduce speed
        if (mspeed > 5)
        {
          mspeed = mspeed - 5;
        }
        break;

      case 'm': //increase speed
        if (mspeed < 220)
        {
          mspeed = mspeed + 5;
        }
        break;

      case 'l': //reset speed to default 100
        mspeed = defspeed;
        break;

      case 'i': //IR Initiate
        bt = 1;
        break;
    }
  }
  if (bt == 1)
  {
    delay(line_follower_stop_delay);
    read();
    if (SrV == 0 && SlV == 0)
    {
      line_follower_forward();
    }
    if (SrV == 1 && SlV == 1)
    {
      if (SmidV == 0)
      {
        line_follower_halt();
      }
      if (SmidV == 1)
      {
        line_follower_stop_reset();
      }
    }
```

Date: 08 Dec. 2021

```
    if (SrV == 1 && SlV == 0)
    {
      line_follower_right();
    }
    if (SrV == 0 && SlV == 1)
    {
      line_follower_left();
    }
    if (SrV == 0 && SlV == 0 && SmidV == 0)
    {
      sTOP();
    }
  }
}
void ForWard()
{
  digitalWrite(lm1, HIGH);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, HIGH);
  digitalWrite(rm2, LOW);
}
void BackWard()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, HIGH);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, HIGH);
}
void Left()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, HIGH);
  digitalWrite(rm1, HIGH);
  digitalWrite(rm2, LOW);
}
void Right()
{
  digitalWrite(lm1, HIGH);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, HIGH);
}
void slowLeft()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, HIGH);
  digitalWrite(rm2, LOW);
}
void slowRight()
{
  digitalWrite(lm1, HIGH);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, LOW);
}
void sTOP()
{
  digitalWrite(lm1, LOW);
  digitalWrite(lm2, LOW);
  digitalWrite(rm1, LOW);
  digitalWrite(rm2, LOW);
}
```

Date: 08 Dec. 2021

```
void read()
{
  SlV = digitalRead(sl);
  SrV = digitalRead(sr);
  SmidV = digitalRead(smid);
}
void line_follower_forward()
{
  ForWard();
  delay(line_follower_delay);
  sTOP();
}
void line_follower_right()
{
  Right();
  delay(line_follower_delay);
  sTOP();
}
void line_follower_left()
{
  Left();
  delay(line_follower_delay);
  sTOP();
}
void line_follower_halt()
{
  sTOP();
  delay(5000);
  ForWard();
  delay(100);
  sTOP();
}
void line_follower_stop_reset()
{
  ForWard();
  delay(100);
  sTOP();
  bt = 0;
}
```