# Project Report on

**Intelligent Seat Counter with Audio**

**Alerts & Limit Configuration**

By:- **HARDIK SATHVARA, EC-086, 20ECUOG040;**

B.Tech - SEM VI

Faculty Supervisor

Mitesh J. Limachia

# ACKNOWLEDGEMENT

Firstly, I offer my sincere thanks to my guide Prof. Mitesh J. Limachia, Assistant Professor, Electronics and Communication Department, Dharmsinh Desai Institute of Technology for their invaluable support, guidance, and advice giventhrough this project and for helping to establish my direction. and also providesinvaluable support and given opportunities to learn and develop technical skills.And has been an unending source of inspiration to me. I am thankful to all the faculty members and institution staff for their consistent support and guidance.

## ABSTARCT

The project involves the creation of an intelligent seating solution that uses Arduino, two E18-D80NK infrared sensor modules, and a rotary encoder for bidirectional counting and limit configuration. The DFPlayer Mini has been integrated to control a speaker that provides audio output based on specific conditions, with all audio files stored in the memory card and played in mp3 format. The system starts with a greeting message and gives an alert message through the speaker when the counting exceeds the predetermined limit of total seats. Additionally, the project utilizes a 16*2 LCD to display the counts and other important information.

## INDEX

## 1.) INTRODUCTION:

The project is a testament to the power of modern technology, specifically the Arduino microcontroller, in creating innovative solutions for everyday problems. In this case, the problem at hand was the need for a reliable seating solution that could keep track of the number of people sitting in a given area. This requirement is particularly relevant in public spaces such as movie theaters, concert halls, and lecture halls, where the number of seats is limited, and there is a need to ensure that they are not over-occupied.

To address this need, the project utilizes a bidirectional counting system based on two E18-D80NK infrared sensor modules and a rotary encoder for limit configuration. This system is capable of tracking the number of people sitting in an area in real-time, making it an efficient and effective solution. To enhance the user experience, the project also integrates the DFPlayer Mini to control a speaker that provides audio output based on specific conditions. This feature allows for the provision of audible alerts when the counting exceeds the total number of seats, ensuring that safety protocols are observed.

Overall, the project provides a comprehensive solution that combines both hardware and software components to create a seamless and reliable seating system that is ideal for use in public spaces. It represents a significant advancement in the field of smart seating solutions, and its impact is sure to be felt in various industries where seating management is critical.

## 2.) BACKGROUND:

In today's world, technology has become an integral part of our daily lives, revolutionizing the way we work and interact with each other. One area where technology has made significant strides is in the field of automation, with the advent of microcontrollers and programmable logic controllers (PLCs). These technologies have enabled the development of intelligent systems that can perform tasks with greater accuracy and efficiency.

The Arduino microcontroller is one such technology that has gained popularity in recent years due to its ease of use and versatility. It has become a popular choice for hobbyists, students, and professionals alike, due to its affordability and ease of programming. The Arduino board can be programmed using the Arduino IDE (Integrated Development Environment), which provides a user-friendly interface for coding and uploading code to the board.

The project at hand is a demonstration of the capabilities of the Arduino microcontroller and its potential in creating intelligent systems. It is a smart seating solution that uses infrared sensors, a rotary encoder, and a DFPlayer Mini to create a reliable and efficient seating management system. The project leverages the capabilities of the Arduino board to create a system that is not only accurate but also user-friendly, making it ideal for use in various industries.

Overall, the project highlights the potential of microcontrollers such as Arduino in creating intelligent systems that can make our lives easier and more efficient. The technology continues to evolve, and with each passing day, we are witnessing the development of new and innovative solutions that are transforming the world around us.

## COMPONENTS REQUIRED:

a.) Arduino Uno
b.) E18-D80NK infrared sensor modules
c.) Connecting Wires
d.) Bread Board
e.) Rotary Encoder
f.) Speaker
g.) DF Player Mini
h.) Battery
i.) LCD 16*2 & I2C
j.) Buzzer
k.) SD Card
l.)

## COMPONENTS DESCRIPTION:

Arduino Uno:

Processor:

The Arduino Uno is based on the ATmega328P microcontroller, which has a 8-bit AVR architecture. It operates at 16 MHz clock speed and has 32 KB flash memory, 2 KB SRAM, and 1 KB EEPROM.

Memory:

The flash memory of the Arduino Uno is where the program code is stored. It is non-volatile, which means that the code remains stored even when the power is turned off. The SRAM is used for storing variables and other data used during program execution. The EEPROM is a non-volatile memory that can be used to store data that needs to be retained even when the power is turned off.

GPIO:

The Arduino Uno has 14 digital input/output (I/O) pins, which can be used for controlling digital devices such as LEDs, motors, and switches. Of these pins, 6 can be used for PWM (Pulse Width Modulation) output, which allows for analog-like control of devices. The board also has 6 analog input pins, which can be used to read analog signals such as temperature and light level.

Other components:

The Arduino Uno also has a USB port, which can be used for programming and power. It also has a power jack for an external power source. The board has a reset button, which can be used to restart the program execution. There is also an LED connected to pin 13, which can be used for debugging and as an indicator.

These components, along with the Arduino programming language and the Arduino IDE, make the

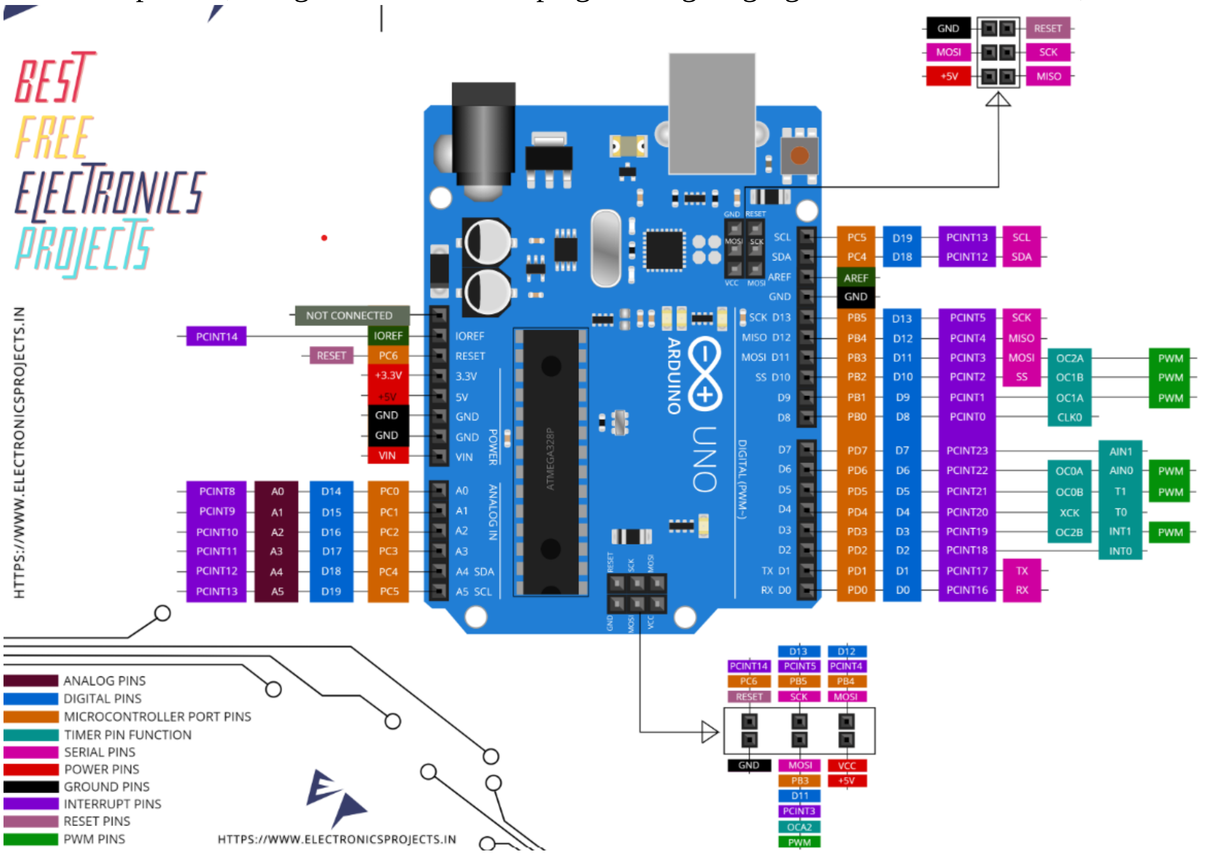
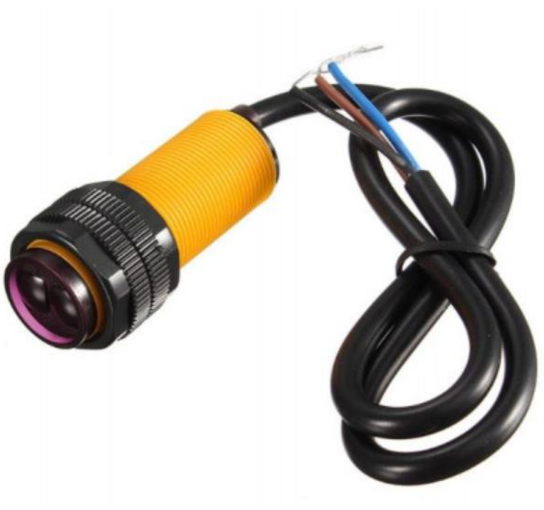

BEST FREE ELECTRONICS PROJECTS
HTTPS://WWW.ELECTRONICSPROJECTS.IN
ARDUINO UNO
ATMEGA328P
ANALOG PINS
DIGITAL PINS
MICROCONTROLLER PORT PINS
TIMER PIN FUNCTION
SERIAL PINS
POWER PINS
GROUND PINS
INTERRUPT PINS
RESET PINS
PWM PINS
NOT CONNECTED
IOREF
RESET
3.3V
5V
GND
VIN
A0 A1 A2 A3 A4 SDA A5 SCL
PCINT8 PCINT9 PCINT10 PCINT11 PCINT12 PCINT13
D14 D15 D16 D17 D18 D19
PC0 PC1 PC2 PC3 PC4 PC5
SCL SDA AREF GND
D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
PB5 PB4 PB3 PB2 PB1 PB0 PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0
PCINT0–PCINT5, PCINT16–PCINT23
SCK MISO MOSI SS TX RX
OC2A OC1B OC1A CLK0 OC0A OC0B XCK OC2B
AIN1 AIN0 T1 T0 INT1 INT0
PWM

Fig 4.1 Pinout Diagram of Arduino Uno

b.) E18-D80NK infrared sensor modules

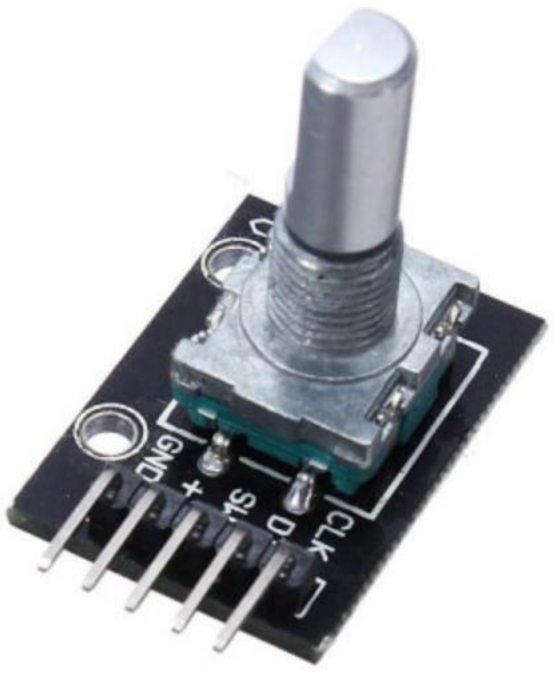


**Rotary Encoder:-**

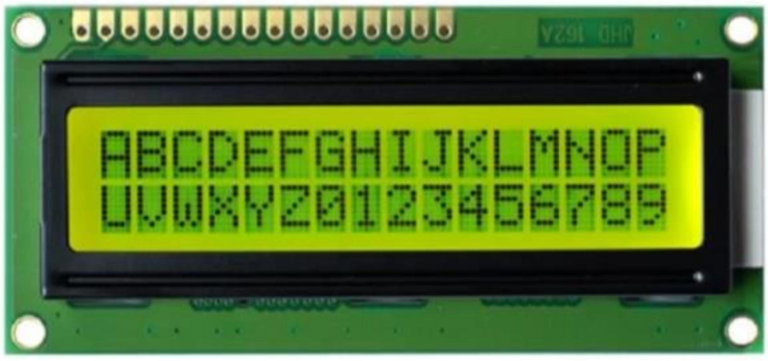
A rotary encoder is an electro-mechanical device that converts the angular position or motion of a shaft into digital or analog signals. It is commonly used in various applications where precise position, speed, or direction control is required, such as in robotics, automation, CNC machines, and audio equipment.

A rotary encoder typically consists of a rotating shaft with a disc or a ring attached to it, and a stationary sensor or multiple sensors that detect the movement of the disc or ring. There are two main types of rotary encoders: incremental and absolute.

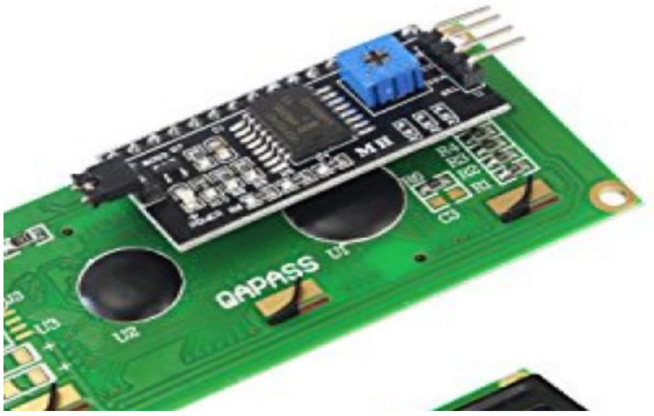
An incremental rotary encoder generates a series of pulses as the shaft rotates, which can be used to measure the relative position, speed, or direction of the shaft. It usually has two outputs, called A and B channels, which produce quadrature signals that have a phase shift of 90 degrees. By counting the pulses and monitoring the phase shift between the A and B channels, it is possible to determine the angular position, speed, and direction of the shaft.



LCD 16*2



I2C



## DFPlayer Mini

DF Player mini is a compact and low-cost MP3 player module that is widely used in various electronic projects such as DIY sound systems, talking toys, musical instruments, and voice prompts. It is designed to play MP3 and WAV audio files stored on a micro SD card with up to 32GB of storage capacity. The module is easy to integrate into any project and comes with a variety of control options including buttons, UART serial communication, and an infrared remote control. It also supports various playback modes such as single loop, all loop, random play, and more. Overall, the DF Player mini is a versatile and cost-effective solution for adding sound playback capabilities to your projects.

## PROGRAMMING CODE:

```
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and
.                                                     //2 line display

SoftwareSerial mySoftwareSerial(10, 11); // RX, TX
DFRobotDFPlayerMini myDFPlayer;


int irPin1=A2;
int irPin2=A1;
int count=0;
int buzz = A3;

//Defining pins for rotary encoder
const int RotaryCLK = 2; //CLK pin on the rotary encoder
const int RotaryDT = 3; //DT pin on the rotary encoder
const int RotarySW = 4; //SW pin on the rotary encoder (Button function)


int total_seat ;

int RotateCounter = 0; //counts the rotation clicks
bool rotated = true; //info of the rotation
bool ButtonPressed =true ; //info of the

boolean state1 = true;
boolean state2 = true;
boolean insideState = false;
boolean outsideIr=false;
boolean isPeopleExiting=false;

int i=1;

//Statuses
int CLKNow;
int CLKPrevious;
int DTNow;
int DTPrevious;
```

```
// Timers
float TimeNow1;
float TimeNow2;//Statuses




void setup() {
 mySoftwareSerial.begin(9600);
  Serial.begin(115200);
  lcd.init();
  // Print a message to the LCD.
  lcd.backlight();

   if (!myDFPlayer.begin(mySoftwareSerial)) {  //Use softwareSerial to communicate
with mp3.

    while(true){
      delay(0); // Code to compatible with ESP8266 watch dog.
    }
  }
myDFPlayer.volume(30);  //Set volume value. From 0 to 30
  myDFPlayer.play(6);//Hi there, we're happy to have you on board.

   lcd.clear();
    lcd.print("WELCOME!!");
   delay(4000);
   lcd.clear();

   lcd.print("Set Limit: ");
   myDFPlayer.volume(30);
    myDFPlayer.play(4);   //Hello and Please Set the limit of total Seats.



     //setting up pins
   pinMode(2, INPUT_PULLUP);
   pinMode(3, INPUT_PULLUP);
   pinMode(4, INPUT_PULLUP);
  pinMode(irPin1, INPUT);
  pinMode(irPin2, INPUT);
  pinMode(buzz,OUTPUT);

  //Store states
  CLKPrevious = digitalRead(RotaryCLK);
  DTPrevious = digitalRead(RotaryDT);

  // Atach a CHANGE interrupt to PinB and exectute the update function when this
change occurs.
attachInterrupt(digitalPinToInterrupt(RotaryCLK), rotate, CHANGE);
  attachInterrupt(digitalPinToInterrupt(RotarySW), buttonPressed, FALLING);
//either falling or rising but never "change".

  TimeNow1 = millis(); //Start timer 1
}

void loop() {
 printLCD();
 ButtonPressed = digitalRead(RotarySW);

   // If the switch is pressed (LOW), print message
  if (ButtonPressed == LOW) {
      Serial.println("Switch pressed");
     total_seat = RotateCounter;
      lcd.clear();
      lcd.print("You have set ");
      lcd.setCursor(0,1);
      lcd.print("Limit: ");
      lcd.setCursor(7,1);
      lcd.print(total_seat);
      delay(2000);


  }

  if (!digitalRead(irPin1) && i==1 && state1){
     outsideIr=true;
     delay(100);
     i++;
     state1 = false;
  }

   if (!digitalRead(irPin2) && i==2 &&   state2){
     outsideIr=true;
     delay(100);
     i = 1 ;
     count++;
      digitalWrite(buzz,HIGH);
     delay(100);
     if(count>total_seat){
      lcd.clear();
      lcd.print("Sorry!");
      lcd.setCursor(0,1);
      lcd.print("Seats are FULL");

        myDFPlayer.volume(30);  //Set volume value. From 0 to 30
 myDFPlayer.play(1);// * 1.>>> Sorry! Seats are Packed.
     }
     else{
      lcd.clear();
      lcd.print("Person in bus: ");
      lcd.setCursor(14,0);
     lcd.print(count);

     state2 = false;

       myDFPlayer.volume(30);  //Set volume value. From 0 to 30
  myDFPlayer.play(5);  //Welcome, Please find a seat and enjoy the ride.
     }



  }
  else{
    digitalWrite(buzz,LOW);
  }

   if (!digitalRead(irPin2) && i==1 && state2 ){
     outsideIr=true;
     delay(100);
     i = 2 ;
     state2 = false;
  }

  if (!digitalRead(irPin1) && i==2 && state1 ){
     outsideIr=true;
     delay(100);
     count--;
     digitalWrite(buzz,HIGH);
     delay(100);
     if(count<0){
      count = 0;
     }

     lcd.clear();
     lcd.print("Person in bus: ");
       lcd.setCursor(14,0);
     lcd.print(count);
```

```
      myDFPlayer.volume(30);  //Set volume value. From 0 to 30
  myDFPlayer.play(8);  //Play the first mp3 "Farewell and have a pleasant rest of your
day!"
     i = 1;

     state1 = false;
  }
   else{
    digitalWrite(buzz,LOW);
  }

    if (digitalRead(irPin1)){
     state1 = true;
    }

     if (digitalRead(irPin2)){
     state2 = true;
    }
}
void buttonPressed()
{
  //This timer is a "software debounce". It is not the most effective solution, but it
works
  TimeNow2 = millis();
  if(TimeNow2 - TimeNow1 > 500)
  {
    ButtonPressed = false;
    total_seat = RotateCounter;

  }
  TimeNow1 = millis();  //"reset" timer; the next 500 ms is counted from this moment
}

void rotate()
{



  CLKNow = digitalRead(RotaryCLK); //Read the state of the CLK pin

  // If last and current state of CLK are different, then a pulse occurred
    if (CLKNow != CLKPrevious  && CLKNow == 1)
    {
    // If the DT state is different than the CLK state then
    // the encoder is rotating CCW so increase
      if (digitalRead(RotaryDT) != CLKNow)
      {
      RotateCounter++;
      }
      else
      {
      RotateCounter--;

      }
    }

  CLKPrevious = CLKNow;  // Store last CLK state
  rotated = true;

}


void printLCD()
{
    if(rotated == true) //refresh the CLK
    {
      lcd.setCursor(12,0);
      lcd.print(RotateCounter);
      rotated = false;
    }

}
```

## Code Explanation:

This code is written in C for an Arduino microcontroller. It uses various libraries such as LiquidCrystal, SoftwareSerial, and DFRobotDFPlayerMini to interface with various hardware components such as LCD displays, rotary encoders, IR sensors, and a DFPlayer mini MP3 module. The code is meant to simulate a bus ride with a limited number of seats, and it counts the number of people entering the bus and plays an MP3 file welcoming the passengers. If the number of passengers exceeds the set limit, it displays a message on the LCD display and plays an MP3 file indicating that the seats are full.

The code starts by setting up the necessary pins, initializing the various libraries, and playing an MP3 file welcoming the passengers. It then waits for the user to set the limit of the total number of seats on the LCD display. Once the user sets the limit, the program reads the number of seats set by the user using a rotary encoder.

The program then waits for passengers to enter the bus. It uses two IR sensors to detect the passengers entering the bus. When the first sensor detects a passenger, it waits for the second sensor to detect the passenger. Once the second sensor detects the passenger, it increments the passenger count and plays an MP3 file welcoming the passenger. If the passenger count exceeds the set limit, it displays a message on the LCD display and plays an MP3 file indicating that the seats are full.

The program also includes a buzzer that buzzes every time a passenger enters the bus. The LCD display shows the number of passengers in the bus. The program uses a timer to ensure that the program runs at regular intervals.

## WORKING/METHODOLOGY:

Data collection: The first step is to collect data related to the project topic. This data can be collected from various sources, such as online databases, research papers, and surveys.

Data pre-processing: Once the data is collected, it needs to be pre-processed to remove any inconsistencies, inaccuracies, and redundancies. This includes data cleaning, data transformation, and data integration.

Model development: After pre-processing the data, the next step is to develop a suitable model. This involves selecting an appropriate algorithm, setting model parameters, and training the model using the pre-processed data.

Model evaluation: Once the model is trained, it needs to be evaluated to determine its performance. This involves testing the model on a separate set of data and analyzing its accuracy, precision, recall, and F1 score.

Model refinement: Based on the results of the model evaluation, the model needs to be refined by tweaking the algorithm, adjusting parameters, or modifying the data. This process is repeated until the model's performance is optimized.

Deployment: Once the model is refined and optimized, it is deployed for use. This involves integrating the model into the existing system or creating a new system that utilizes the model to solve real-world problems.

## PRACTICAL RESULTS (APPLICATIONS):

The intelligent seating solution project has practical applications in various settings that require tracking the number of people occupying a space. Here are a few potential applications:

Conference rooms and auditoriums - the system can help keep track of the number of people attending a meeting or event and provide alerts when the maximum capacity is reached.

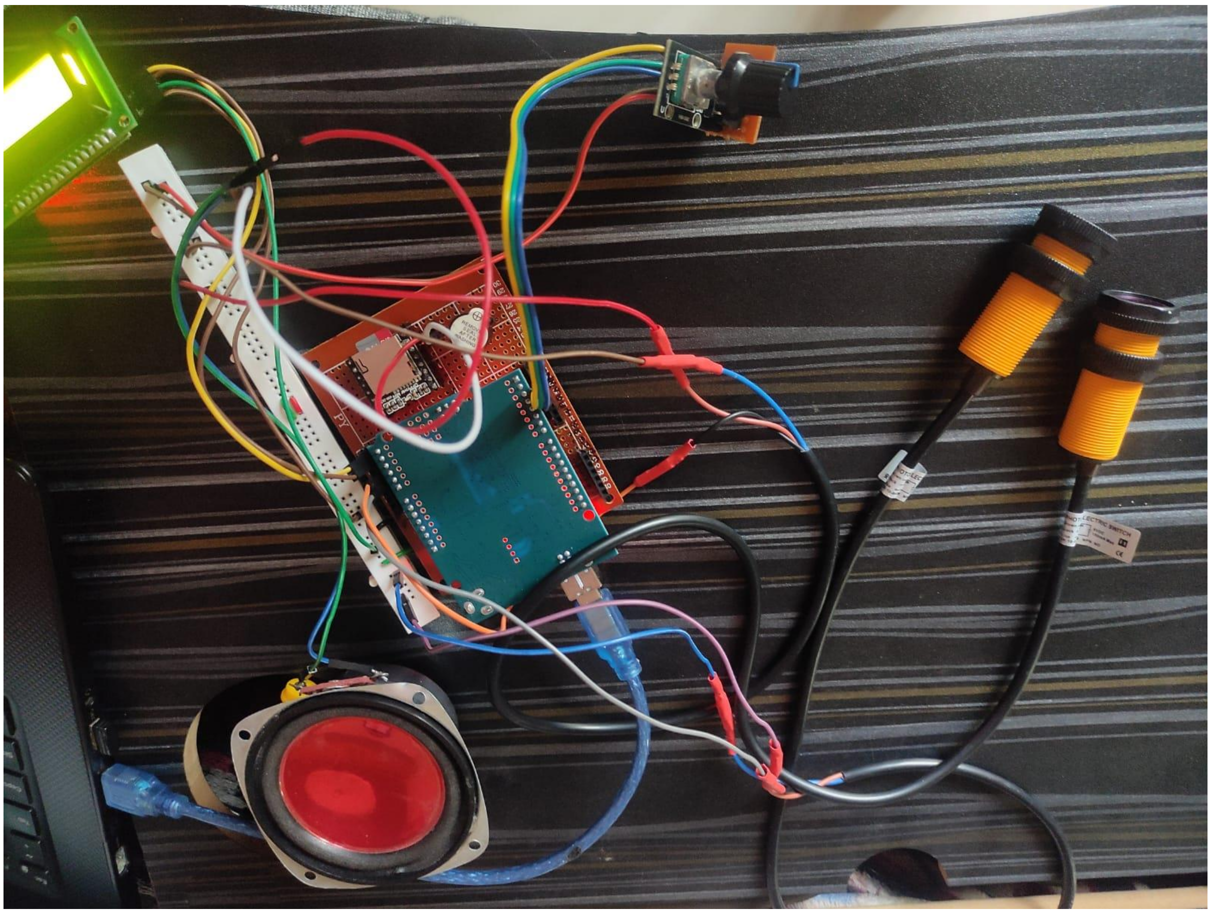
Public transportation - the system can be installed in buses and trains to track the number of passengers and alert the driver when the capacity limit is reached.

Restaurants and cafes - the system can help keep track of the number of customers occupying tables and provide alerts when the maximum capacity is reached.

Libraries and study spaces - the system can be used to monitor the number of people occupying study rooms and provide alerts when the maximum capacity is reached.

Overall, this project demonstrates the potential for using Arduino and other components to create intelligent solutions that can help solve real-world problems.

## 3.) PRACTICAL HARDWARE SETUP:



## ADVANTAGES, LIMITATIONS AND FUTURE USES:

Advantages:

- Educational: This project can be used to educate beginners about the basics of Arduino programming and how to interface different hardware components.
- Real-world simulation: The simulation of a bus ride with limited seats can help in understanding real-world scenarios where constraints exist, and managing them is necessary.
- Interactive: The use of MP3 files and LCD display makes the project interactive, and it can be used as an attention-grabbing exhibit for children or beginners.
- Customizable: The user can set the number of seats and adjust the code to fit their needs, making it customizable for different scenarios.

Limitations:

- Limited functionality: The project has a limited functionality as it only counts the number of passengers and plays an MP3 file.
- Hardware restrictions: The project is limited to the hardware components available and may not work with other hardware components.
- Complex setup: The setup process can be complex, and beginners may require assistance to set up the hardware and software.

Future uses:

- Improved functionality: The project can be improved by adding more features such as a payment system, route selection, and GPS tracking.
- Educational purposes: This project can be used in educational institutions to teach programming and electronics.
- Real-world applications: The project can be modified and applied to real-world scenarios such as a ticketing system for public transportation or events.

## CONCLUSION:

In conclusion, the Arduino-based bus simulation project provides a fun and interactive way to learn the basics of programming and electronics. The use of various hardware components such as IR sensors, rotary encoders, and LCD displays makes the project interesting and engaging. This project can be used for educational purposes in schools, colleges, and other institutions to teach students about electronics, programming, and real-world applications.

The project's functionality is limited as it only counts the number of passengers and plays an MP3 file. However, it can be modified to add more features such as a payment system, route selection, and GPS tracking to make it more functional. This project can be a starting point for more advanced projects with improved functionality and more extensive applications.

The project has some limitations, including hardware restrictions and a complex setup process. However, with proper guidance, beginners can overcome these limitations and enjoy the learning experience. The project's hardware restrictions also limit its compatibility with other hardware components, which may be a drawback for some users.

In summary, the Arduino-based bus simulation project is a fun and interactive way to learn programming and electronics. With its customizable features and real-world applications, this project can be a starting point for more advanced projects with improved functionality. Despite its limitations, the project provides an excellent learning experience for beginners and can be used for educational purposes in institutions..