

Smart Car Parking System Based on Microcontroller

Joy Barai, Maleha Israt Chowdhury, Kamarum Monira Mow

CSE Department, East West University

Jahurul Islam City Gate, A/2 Jahurul Islam Avenue, Dhaka 1212, Bangladesh

Joy barai, Maleha Israt Chowdhury, Kamarum Monira Mow

Abstract- Nowadays it's a quite complex problem for our daily life that where the parking lot is, what is its condition, does it have a vacancy for parking another car now, etc. More or less the number of cars in the city is increasing and the parking lots are needed to be the more available cause of this scenario. People have to park their cars for different issues in different places. The main goal of a smart car parking system is to initialize a fully automated parking system that will provide the security of the car and can handle some emergency issues. A fully secured individual password will protect the whole system of the smart car parking lot and it can give warning about an incident like gas leakage.

Keywords- Microcontroller, Smart Parking system, Secured parking system, Password-protected system, Warning for gas leakage, IR sensor.

1. Introduction

Nowadays the increment of cars is one of the problems for traffic control and car parking. When a car is not parking in a perfect place it prompts traffic blockage. For the serious issue, we have to control the parking lots smartly as they can work time efficiently. [1]

The search for the parking space is a time-consuming process with affects social interactions and costs also. The network companies cannot provide necessary information about parking lots and parking facilities do not cooperate with those companies. As a result, complexity keeps increasing its probability. [2]

The main problem of traffic accidents is not any functional electronics problem. The driver has to take his stare at too many places. He has to control the car, the indicators, the eyesight, and many things. He has to use his brain for finding a parking lot when he needs to stop. If it is possible to reduce the stress level or responsibility of the driver by some automation, then it will be helpful for the people. [3]

The terrific thing is finding a perfect parking place for the car. Here and there are parking lots but people don't know where can be the possible vacancy and if the place is fit for their car. Like some parking lots are not available for long heights car and some parking lots are already filled with cars. Drivers have to find the perfect parking lots continuously and it's a reason for wasting the precious time of the people. The smart car parking system can maintain a time-efficient travel time. [2]

As the world is developing day by day its obvious that vehicle will increase on the road

by people's choice. But the problem is when people start using their cars, the road becomes busy and it will blockage the traffic. The main dealing issue with the problem is to maintain a digital smart parking lot that can offer a fully secured parking lot with some extra advantages. The report said that more than 30% of the usual total traffic jams are due to road construction and related traffic problems. European data says that searching for the parking lot is approximate 20 minutes of travel time. The important news is more than 24% of traffic accidents have the same cause in it and that is because of careless parking or related with parking.[4]

2. Related Works

It's important that a smart car parking system is tracing a car when it is in the parking lot. By the way, the microcontroller can count the cars and note down the data in and out. There are many projects those are working for developing a smart car parking system. [5]

Georgia Institute of Technology was working on the same project as this. Their goal was similar but the procedure was not. They were working with sensible technologies like satellite imaging, data acquisition, and future data prediction for dealing with different scenarios. They have dealt with sensor-based and FiberBragg grating sensors, which were placed at ground and pressure-sensitive. By the way, they found out any presence of a vehicle. [6]

There are other similar projects which were based on indoor, outdoor, and city management systems with the smart parking lot and that was done by GAK technology. They have used ultrasonic modules and cameras as sensors. They used RS485

communication cable for building the communication system outdoor and indoor. The whole system got information from ultrasonic motors and cameras and visualize the output as green or red signals. They used a similar system for the outside too. The difference was they were using the geometric sensor for detecting changes in magnetic fields and communicate the server wirelessly. [7]

3. Proposed Work

Normal parking lots are too complex to use when it's about to hurry in a busy city. A smart parking lot can solve the problem. It's more advantageous when the parking lot has security if any gas leakage or unwanted situation arrives. The smart solution can stop the entry for any emergency scenario and lives can be saved.

A. Features

- Displaying the password box and require a password to enter the parking lot to maintain security with a protected password-based system.
- Displaying the number of free spaces to show the available space in the parking lot at a time.
- Displaying the total number of vehicles currently present in the parking spot for giving the idea it's on the rush hour or not.
- Detecting gas by using sensors before any accidents occur to maintain safety measures.
- Providing emergency exits for emergencies.
- Because of safety purposes, it can block the entry system if gas is detected in the parking spot.





- A buzzer will keep beeping and the red LED will start blinking if the sensor detects gas on the spot, otherwise, the LED will turn on and show the red light.






B. Challenges

- Create a secured system for every vehicle.
- Protecting by a password.
- Handle and nonstop counting on vehicles entering and exiting.
- Counting the total vehicles parked in a 24-hour cyclic day.

Table 1

LIST OF COMPONENTS FOR THE SMART CAR PARKING SYSTEM.

Components	Diagram	Working
Arduino Uno		Programming of the display level, sensor level, input level, and motor level components
Servo Motor		Open and close the barrier for vehicles entry and exit
IR sensor		Detect the vehicles during entry and exit.
LCD		Display messages in 16X2 display

Keypad		Use for input the password
Gas sensor		Detect the gas for maintaining emergencies.
LED		It will blink when the gas will detect.
Buzzer		When the sensor detects gas, the buzzer will keep beeping for warning.
4 Digit 7 segment display		Count the total number of vehicles for a whole day.

C. Software

- Proteus 8 Professional
- Arduino IDE 1.8.15

D. System Overview

The system is always ready to read any enter or exit. It is required to enter the password for any actions like entering or exit. It will count the entry and exit because it is necessary for counting the free spaces inside the parking lot. The display is ready to show the available space of the parking lot.

Besides, the system detects gas if any gas leakage occurs. It immediately starts beeping with the buzzer itself and blocks the entrance of the parking lot.

It opens the exit immediately and the emergency exit is not declared to allow any entrance, it's only for exit immediately. The emergency LED light starts to blink if any gas leakage detects.

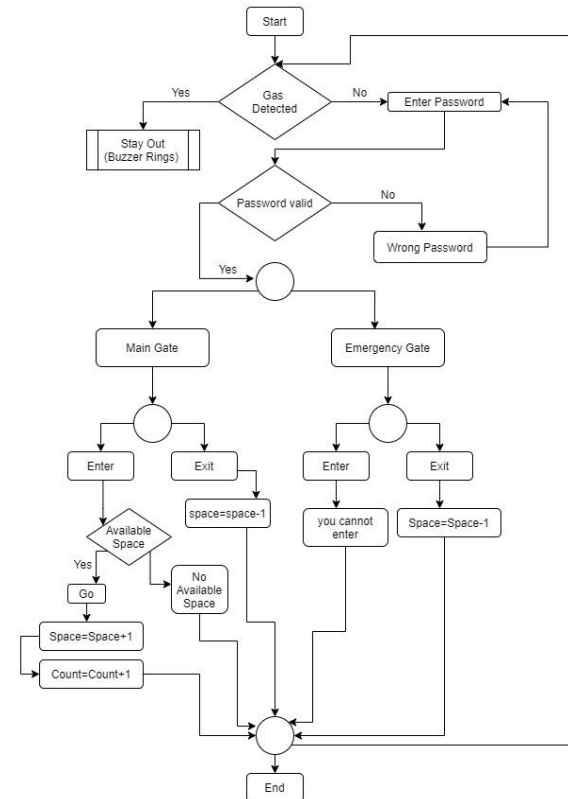


Fig 1: The diagram for smart car parking system.

E. Schematic Diagram

There is a diagram of the smart car parking system –

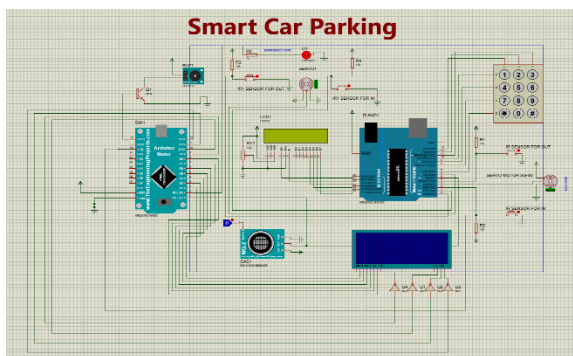


Fig 2: The schematic diagram for smart car parking system.

4. Result Analysis

A. Simulation Analysis

We simulate the whole project on Proteus 8 Professional and here are the results of the simulation.

The whole system is protected by password and instead of the correct password, anything else will not work there. When the correct password will take place from the keypad, the door will unlock.

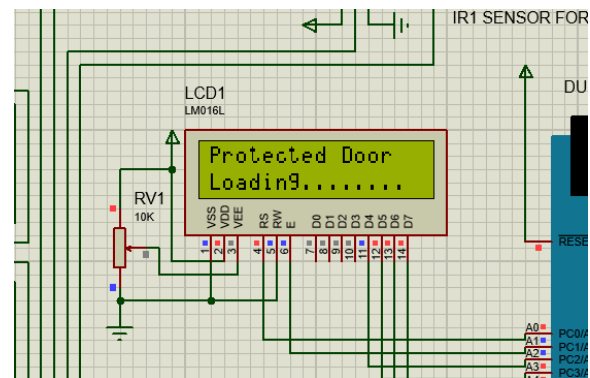


Fig 3: the protected door.

The password should take place from the analog keypad we place there and the screen will show the message to input the password.

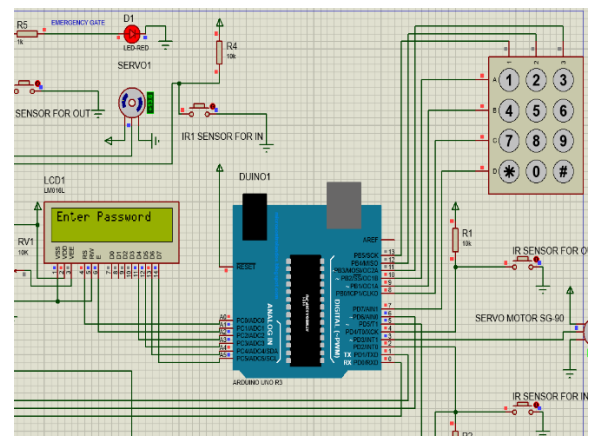


Fig 4: the password validation system for entering a vehicle.

If anyone inputs the wrong password the system will show the message on the screen.

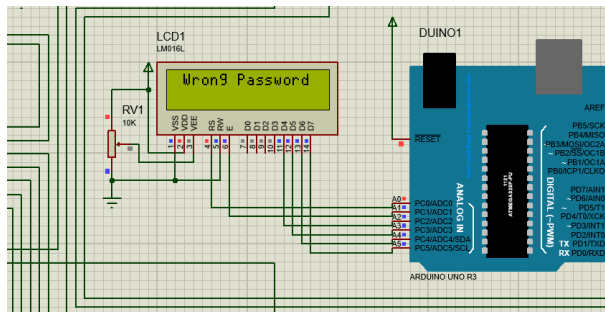


Fig 5: showing message if someone put the wrong password.

If the right password is given the door will open and the space available data will show.

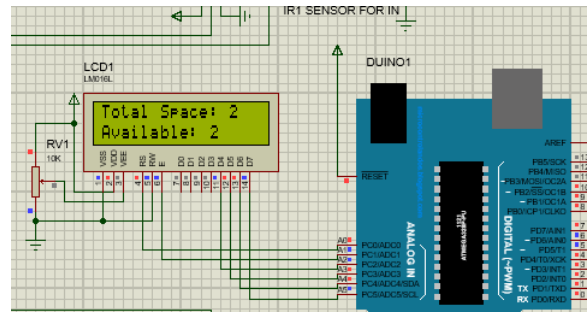


Fig 8: showing vacancy information if someone put the right password.

If anyone inputs the right password the system will show the message on the screen and the door will open itself.

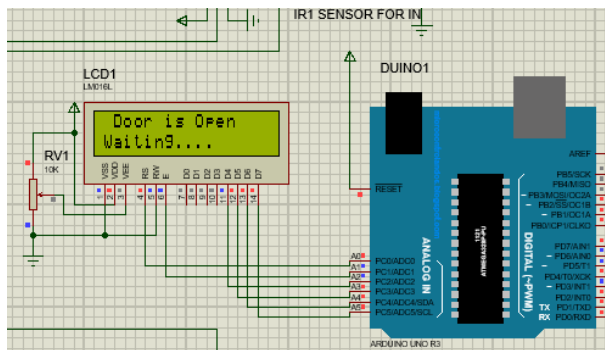


Fig 6: showing message if someone put the right password.

If the parking lot is full the display will show that message.

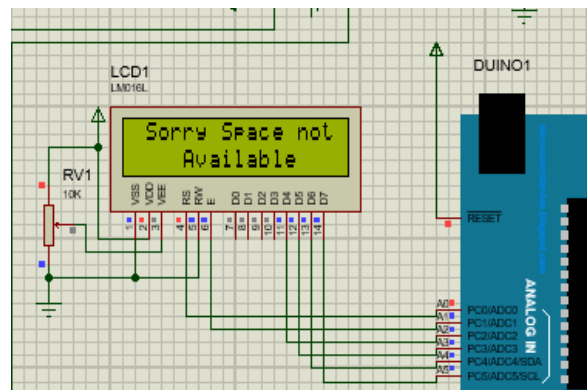


Fig 9: showing message if someone parking lot is full.

If gas is detected by the sensor, it will block the entry and the display will show the message.

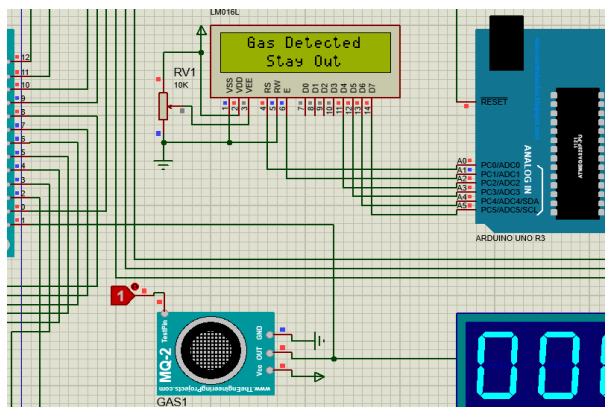


Fig 7: gas detection in the system.

If a vehicle enters the parking lot the available space will decrease automatically.

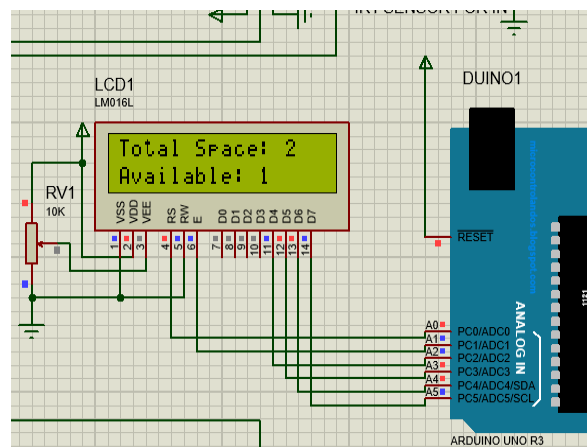


Fig 10: Space decreases after a vehicle entry.

If a vehicle exits, the parking lot increases its space on the display.

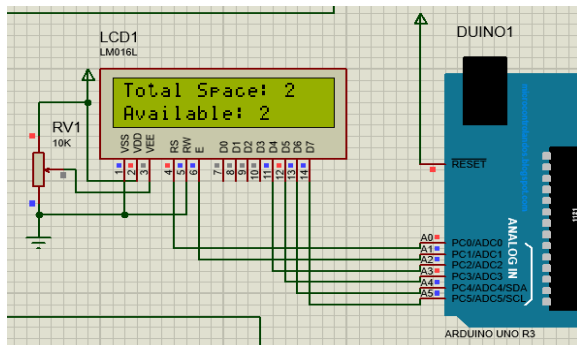


Fig 11: Space increase if a vehicle leaves.

Here we also count the total number of vehicles entering in a whole working day because if we want to work with this project further, this will help us.

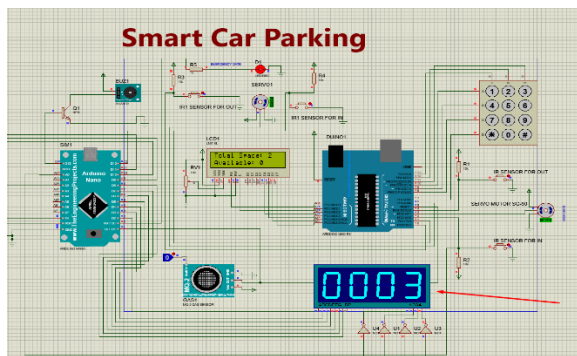


Fig 12: showing the total number of vehicles entering in a full day.

Here we also add an emergency exit so if any emergency arrived this gate can help vehicles in fast leaving purpose.

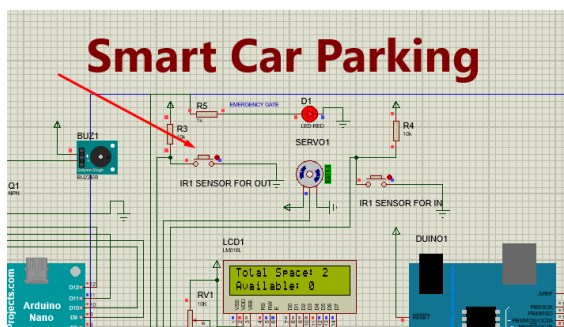


Fig 13: This is the emergency gate for emergencies.

5. Future Work Scopes

Here we are using simulators for simulating the whole project because of the Covid situation it is not nearly possible to arrange all the hardware and work with those. Meanwhile, the whole project can be converted into Machine Learning and in real-life simulation, we can find out more scopes of this project. As future work, we can add LED on parking slots to know whether it is available or not from far, and by using IOT we can use the location-based map to avoid hassle to find a place.

6. Conclusion

A country like us, which is rapidly increasing its vehicles numbers and the traffics are affected by this, we need the solution of smart car parking. This whole project is based on a microcontroller. With updating all the things the parking lot have to be updated meanwhile all the vehicles need to be placed on the right way. Developed countries have already adapted those features and they are getting advantages in daily life.

References

- [1] R. Atiqur, "Smart car parking system model for urban areas," [Online]. Available: https://www.researchgate.net/publication/349882122_Smart_car_parking_system_model_for_urban_areas.
- [2] Faheem, "A Survey of Intelligent Car Parking System," [Online]. Available: <https://reader.elsevier.com/reader/sd/pii/S1665642313715803?token=416AB9A0D26ED8CF95720F6D892A5881C8A4552A7A069EA2C1C2CC6126FA8CB4AE4A882EEC8933AD13625D650C2C1C45&originRegion=euro-west-1&originCreation=20210908161600>.
- [3] M. Pasquier, "Self-Trained Automated Parking System," [Online]. Available: <https://scihub.se/https://ieeexplore.ieee.org/document/1468981>.
- [4] M. Tampucci, "Real-Time Smart Parking Systems Integration in Distributed ITS for Smart Cities," [Online]. Available: https://www.researchgate.net/publication/328427016_Real-Time_Smart_Parking_Systems_Integration_in_Distributed_ITS_for_Smart_Cities.
- [5] F. Nait-Abdesselam, "Towards a Smart Parking Management System for Smart Cities," [Online]. Available: <https://scihub.se/https://ieeexplore.ieee.org/document/9071740>.
- [6] J. Tianb, "Research into a Wireless Smart Parking System," [Online]. Available: <https://www.aidic.it/cet/15/46/041.pdf>.
- [7] M. Zennaro, "On the Design of Smart Parking Networks in the Smart Cities: An Optimal Sensor Placement Model," [Online]. Available: https://www.researchgate.net/publication/281213642_On_the_Design_of_Smart_Parking_Networks_in_the_Smart_Cities_An_Optimal_Sensor_Placement_Model.

UNO Code

```
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Servo.h>

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
Servo myservo1;
Servo myservo2;

#define Password_Length 4

int ir_s1 = 2;
int ir_s2 = 4;

int ir_s3 = 1;
int ir_s4 = 0;

int Total = 3;
int Space ;

int flag1 = 0;
int flag2 = 0;

int Gas = 5;

char Data[Password_Length];
char Master[Password_Length] = "789";
byte data_count = 0, master_count = 0;

bool Pass_is_good;
bool door = false;
char customKey;

const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};

byte rowPins[ROWS] = {10, 9, 8, 7};
byte colPins[COLS] = {13, 12, 11};
```



```
Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
void setup() {
```

```
  pinMode(ir_s1, INPUT);
```

```
  pinMode(ir_s2, INPUT);
```

```
  pinMode(ir_s3, INPUT);
```

```
  pinMode(ir_s4, INPUT);
```

```
  pinMode(Gas, INPUT);
```

```
  myservo1.attach(3);
```

```
  myservo1.write(100);
```

```
  myservo2.attach(6);
```

```
  myservo2.write(100);
```

```
  lcd.begin(16, 2);
```

```
  lcd.print("Protected Door");
```

```
  loading("Loading");
```

```
  lcd.clear();
```

```
  Space=Total;
```

```
}
```

```
void loop()
```

```
{
```

```
  if (digitalRead (Gas) == HIGH)
```

```
  {
```

```
    lcd.setCursor (0, 0);
```

```
    lcd.print(" Gas Detected ");
```

```
    lcd.setCursor (0, 1);
```

```
    lcd.print(" Stay Out ");
```

```
  }
```

```
  else
```

```
  {
```

```
    if (door == true)
```

```
    {
```

```
      customKey = customKeypad.getKey();
```

```
      EntryExit();
```

```
      EmergencyExit();
```

```
      if (customKey == '#')
```

```

    {
        lcd.clear();
        lcd.print("Door is closed");
        delay(3000);
        door = false;
    }
}
else
    Open();
}
}

```

```

void EntryExit()

```

```

{
    if (digitalRead (ir_s1) == LOW && flag1 == 0)
    {
        if (Space > 0)
        {
            flag1 = 1;
            if (flag2 == 0)
            {
                myservo1.write(0);
                Space = Space - 1;
            }
        }
    }
    else
    {
        lcd.setCursor (0, 0);
        lcd.print(" Sorry Space not ");
        lcd.setCursor (0, 1);
        lcd.print(" Available ");
        delay (1000);
        lcd.clear();
    }
}

if (digitalRead (ir_s2) == LOW && flag2 == 0 && Space < Total)
{
    flag2 = 1;
    if (flag1 == 0 )
    {
        myservo1.write(0);
        Space = Space + 1;
    }
}
}

```

```
if (flag1 == 1 && flag2 == 1)
{
    delay (1000);
    myservo1.write(100);
    flag1 = 0, flag2 = 0;
    door = false;
}
```

```
lcd.setCursor (0, 0);
lcd.print("Total Space: ");
lcd.print(Total);
```

```
lcd.setCursor (0, 1);
lcd.print("Available: ");
lcd.print(Space);
}
```

```
void EmergencyExit()
{
    if (digitalRead (ir_s3) == LOW && flag1 == 0)
    {
        flag1 = 1;
        if (flag2 == 0)
        {
            lcd.setCursor (0, 0);
            lcd.print("    Sorry    ");
            lcd.setCursor (0, 1);
            lcd.print("You cannot Enter");
            delay (1000);
            lcd.clear();
            flag1 = 0;
        }
    }
}
```

```
if (digitalRead (ir_s4) == LOW && flag2 == 0 && Space < Total)
{
    flag2 = 1;
    if (flag1 == 0)
    {
        myservo2.write(0);
        Space = Space + 1;
    }
}
```

```
if (flag1 == 1 && flag2 == 1)
{
    delay (1000);
    myservo2.write(100);
    flag1 = 0, flag2 = 0;
    door = false;
}
```

```
lcd.setCursor (0, 0);
lcd.print("Total Space: ");
lcd.print(Total);
```

```
lcd.setCursor (0, 1);
lcd.print("Available: ");
lcd.print(Space);
}
```

```
void loading (char msg[])
{
    lcd.setCursor(0, 1);
    lcd.print(msg);

    for (int i = 0; i < 9; i++)
    {
        delay(200);
        lcd.print(".");
    }
}
```

```
void clearData()
{
    while (data_count != 0)
        Data[data_count--] = 0;
    return;
}
```

```
void Open()
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Enter Password");
```

```
customKey = customKeypad.getKey();
if (customKey)
{
```

```

    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);
    data_count++;
}

if (data_count == Password_Length - 1)
{
    if (!strcmp(Data, Master))
    {
        lcd.clear();
        lcd.print(" Door is Open ");
        door = true;
        delay(500);
        loading("Waiting");
        lcd.clear();

        lcd.setCursor(0, 0);
        lcd.print(" Car Parking ");
        lcd.setCursor(0, 1);
        lcd.print(" System ");
        delay(500);
    }
    else
    {
        lcd.clear();
        lcd.print(" Wrong Password ");
        door = false;
    }

    delay(1000);
    lcd.clear();
    clearData();
}
}

```

NANO Code

```
#define aPin 2 //
#define bPin 3 //
#define cPin 4 // | A |
#define dPin 5 // F | | B
#define ePin 6 // | G |
#define fPin 7 // E | | C
#define gPin 8 // D O dot

#define c1Pin 9 // CommLOW pin for digit 1
#define c2Pin 10 // Common pin for digit 2
#define c3Pin 11 // Common pin for digit 3
#define c4Pin 12 // Common pin for digit 4

#define bt_up A0

long Counter=0;
byte current_digit;
int flag1=0, flag2=0, timer=0;

int Gas = 1;
int redLed = 0;
int Buzz = 13;

int Common = 1; // <Common=1; for Common anode> <Common=0; for Common cathode>
int On, Off, Off_C;
int DTime = 15; // Display timer

void setup() {
  pinMode(bt_up, INPUT);
  pinMode(aPin, OUTPUT);
  pinMode(bPin, OUTPUT);
  pinMode(cPin, OUTPUT);
  pinMode(dPin, OUTPUT);
  pinMode(ePin, OUTPUT);
  pinMode(fPin, OUTPUT);
  pinMode(gPin, OUTPUT);

  pinMode(c1Pin, OUTPUT);
  pinMode(c2Pin, OUTPUT);
  pinMode(c3Pin, OUTPUT);
  pinMode(c4Pin, OUTPUT);
  disp_off();
  TCCR1A = 0;
```

```

TCCR1B = 1; // enable Timer1 with prescaler = 1 ( 16 ticks each 1 µs)
TCNT1 = 0; // set Timer1 preload value to 0 (reset)
TIMSK1 = 1; // enable Timer1 overflow interrupt
pinMode(Gas, INPUT);
pinMode (13 , OUTPUT);
}
ISR(TIMER1_OVF_vect) // Timer1 interrupt service routine (ISR)
{
    disp_off(); // turn off the display

    switch (current_digit)
    {
        case 1:
            showNumber((Counter/1000)%10);

digitalWrite(c1Pin, LOW);
            break;

        case 2:
            showNumber((Counter/100)%10);

digitalWrite(c2Pin, LOW);
            break;

        case 3:
            showNumber((Counter/10)%10);

digitalWrite(c3Pin, LOW);
            break;

        case 4:
            showNumber(Counter%10);

digitalWrite(c4Pin, LOW);
    }

    current_digit = (current_digit % 4) + 1;
}

void loop() {

    if(digitalRead(Gas) == HIGH){

        //" Gas Detected "
        digitalWrite(13, HIGH);
    }
}

```

```

    digitalWrite(0 , HIGH);
    delay(80);
    digitalWrite(0 , LOW);
    delay(80);

}
else{

    digitalWrite(13 , LOW);
    digitalWrite(0 ,HIGH);
    delay(500);
}

if (flag1 == 1 && flag2 == 1)
{

    flag1 = 0, flag2 = 0;

}
if(digitalRead (bt_up) == 0){
Counter++; // increment 'count' by 1
    if(Counter>=9999)
        Counter = 0;
    delay(200); // wait 200 milliseconds
}
}

void showNumber(int x){

    switch(x){
        case 1: one(); break;
        case 2: two(); break;
        case 3: three(); break;
        case 4: four(); break;
        case 5: five(); break;
        case 6: six(); break;
        case 7: seven(); break;
        case 8: eight(); break;
        case 9: nine(); break;
        default: zero(); break;
    }
}

```



```

void one(){
    digitalWrite( aPin, HIGH); //
    digitalWrite( bPin,LOW); // |
    digitalWrite( cPin,LOW); // |
    digitalWrite( dPin,HIGH); // |
    digitalWrite( ePin,HIGH); // |
    digitalWrite( fPin, HIGH);
    digitalWrite( gPin, HIGH);

}

```

```

void two(){
    digitalWrite( aPin,LOW); // ____
    digitalWrite( bPin,LOW); //      |
    digitalWrite( cPin,HIGH); // ____|
    digitalWrite( dPin,LOW); // |
    digitalWrite( ePin,LOW); // |____
    digitalWrite( fPin, HIGH);
    digitalWrite( gPin, On);

}

```

```

void three(){
    digitalWrite( aPin,LOW); // ____
    digitalWrite( bPin,LOW); //      |
    digitalWrite( cPin,LOW); // ____|
    digitalWrite( dPin,LOW); //      |
    digitalWrite( ePin,HIGH); // ____|
    digitalWrite( fPin,HIGH);
    digitalWrite( gPin,LOW);

}

```

```

void four(){
    digitalWrite( aPin, HIGH); //
    digitalWrite( bPin, LOW); // | |
    digitalWrite( cPin, LOW); // |____|
    digitalWrite( dPin, HIGH); //      |
    digitalWrite( ePin, HIGH); //      |
    digitalWrite( fPin, LOW);
    digitalWrite( gPin, LOW);
}

```

```
}
```

```
void five(){  
    digitalWrite( aPin,LOW); // ____  
    digitalWrite( bPin,HIGH); // |  
    digitalWrite( cPin,LOW); // ____  
    digitalWrite( dPin,LOW); // ____|  
    digitalWrite( ePin,HIGH); // ____|  
    digitalWrite( fPin, LOW);  
    digitalWrite( gPin,LOW);  
  
}
```

```
void six(){  
    digitalWrite( aPin,LOW); // ____  
    digitalWrite( bPin,HIGH); // |  
    digitalWrite( cPin,LOW); // ____  
    digitalWrite( dPin,LOW); // | ____|  
    digitalWrite( ePin,LOW); // ____|  
    digitalWrite( fPin,LOW);  
    digitalWrite( gPin,LOW);  
  
}
```

```
void seven(){  
    digitalWrite( aPin,LOW); // ____  
    digitalWrite( bPin,LOW); // ____|  
    digitalWrite( cPin,LOW); // ____|  
    digitalWrite( dPin, HIGH); // ____|  
    digitalWrite( ePin, HIGH); // ____|  
    digitalWrite( fPin, HIGH);  
    digitalWrite( gPin, HIGH);  
  
}
```

```
void eight(){  
    digitalWrite( aPin, LOW); // ____  
    digitalWrite( bPin, LOW); // | ____|  
    digitalWrite( cPin, LOW); // ____|  
    digitalWrite( dPin, LOW); // | ____|  
    digitalWrite( ePin, LOW); // ____|  
    digitalWrite( fPin, LOW);  
    digitalWrite( gPin, LOW);
```

```
}
```

```
void nine(){  
    digitalWrite( aPin, LOW); //  ____  
    digitalWrite( bPin, LOW); // |   |  
    digitalWrite( cPin, LOW); // |____|  
    digitalWrite( dPin, LOW); //      |  
    digitalWrite( ePin, HIGH); //  ____|  
    digitalWrite( fPin, LOW);  
    digitalWrite( gPin, LOW);  
  
}
```

```
void zero(){  
    digitalWrite( aPin, LOW); //  ____  
    digitalWrite( bPin, LOW); // |   |  
    digitalWrite( cPin, LOW); // |   |  
    digitalWrite( dPin, LOW); // |   |  
    digitalWrite( ePin, LOW); // |____|  
    digitalWrite( fPin, LOW);  
    digitalWrite( gPin, HIGH);  
  
}
```

```
void disp_off()  
{  
    digitalWrite(c1Pin, HIGH);  
    digitalWrite(c2Pin, HIGH);  
    digitalWrite(c3Pin, HIGH);  
    digitalWrite(c4Pin, HIGH);  
}
```