Github link :

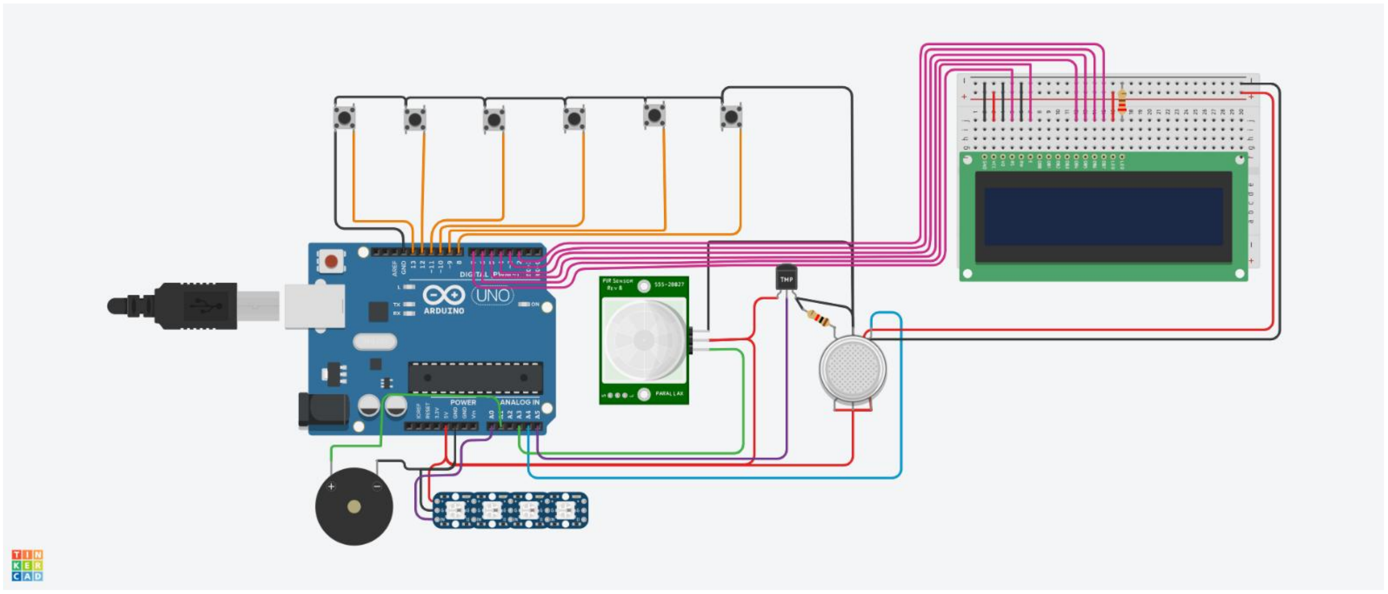
https://github.com/Ranjeet-Waghmode/pro1234

tinkercad simulation

https://www.tinkercad.com/things/gY7AZ48yXZQ/editel?returnTo=%2Fdashboard%2Fdesigns%2Fcircuits&sharecode=KopxZ0IeZOQJX739y-c9Cbg-0yVHGmhb3_toLmWuPjA

all related files are shared in the repo

.BRD, picture ,electronic component list and all ..



Component list file name is

component.csv

Arduino Code :

```
#include <Adafruit_NeoPixel.h>

#include <LiquidCrystal.h>


// Pin Definitions

const int frontLeftDoorPin = 13;

const int frontRightDoorPin = 12;

const int backLeftDoorPin = 11;

const int backRightDoorPin = 10;

const int frontBonnetPin = 9;

const int backTrunkPin = 8;


// Temperature sensor pin (LM35) - Now at A5

const int tempPin = A5;


// Gas sensor pin (analog input)

const int gasPin = A4;  // Gas sensor connected to A4 pin


// PIR sensor pin

const int pirPin = A3;  // PIR sensor to detect motion


// Debounce-related variables

unsigned long debounceDelay = 50;                    // 50ms debounce delay

unsigned long lastDebounceTime[6];                   // Store debounce time for 6 switches
```

```
int lastButtonState[6] = { HIGH, HIGH, HIGH, HIGH, HIGH, HIGH };  // Last state of switches

int buttonState[6] = { HIGH, HIGH, HIGH, HIGH, HIGH, HIGH };      // Current state of switches


LiquidCrystal lcd(7, 6, 5, 4, 3, 2);


// ====== Buzzer for Sound Effects ======

// Variables for controlling buzzer timing

unsigned long previousToneMillis = 0;   // Timer to avoid blocking delays for tone.

unsigned long previousPauseMillis = 0;  // Timer to avoid blocking delays for pause.

bool isTonePlaying = false;            // Flag to indicate if tone is playing.

bool isPauseActive = false;            // Flag to indicate if pause is active.




#define PINBUZZ A1  // input pin Buzz is attached to


#define PIN A0  // input pin Neopixel is attached to


#define NUMPIXELS 4  // number of neopixels in Ring


Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);


int delayval = 100;  // timing delay

int redColor = 0;

int greenColor = 0;

int blueColor = 0;
```

```
// setColor()
// picks random values to set for RGB
void show_lights(int red = 0, int green = 0, int blue = 0) {
  redColor = red;
  greenColor = green;
  blueColor = blue;
  // Serial.print("red: ");
  Serial.println(redColor);
  // Serial.print("green: ");
  Serial.println(greenColor);
  // Serial.print("blue: ");
  Serial.println(blueColor);

  for (int i = 0; i < NUMPIXELS; i++) {

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));  // Moderately bright
green color.

    pixels.show();  // This sends the updated pixel color to the hardware.
    beep();
    delay(delayval);  // Delay for a period of time (in milliseconds).

    // Serial.println(i);
    if (i == NUMPIXELS) {
      i = 0;  // start all over again!
    }
  }
}



void beep() {
  unsigned long currentMillis = millis();
  if (true) {
    // Tone ON for 100 ms
    if (!isTonePlaying && !isPauseActive && currentMillis - previousToneMillis >= 300) {
      tone(PINBUZZ, 800);
      isTonePlaying = true;
      previousToneMillis = currentMillis;
    }
    // Tone OFF after 100 ms
    if (isTonePlaying && currentMillis - previousToneMillis >= 100) {
      noTone(PINBUZZ);
      isTonePlaying = false;
      isPauseActive = true;
      previousPauseMillis = currentMillis;
    }

    // Pause for 300 ms after the tone
    if (isPauseActive && currentMillis - previousPauseMillis >= 300) {
      isPauseActive = false;  // Reset for the next tone
    }
  } else noTone(PINBUZZ);
}

void setup() {
  // Initialize serial communication
  Serial.begin(9600);  // Serial to PC
  pixels.begin();      // Initializes the NeoPixel library.
  lcd.begin(16, 2);

  // Initialize limit switch pins as inputs with internal pull-ups
  pinMode(frontLeftDoorPin, INPUT_PULLUP);
  pinMode(frontRightDoorPin, INPUT_PULLUP);
  pinMode(backLeftDoorPin, INPUT_PULLUP);
  pinMode(backRightDoorPin, INPUT_PULLUP);
  pinMode(frontBonnetPin, INPUT_PULLUP);
  pinMode(backTrunkPin, INPUT_PULLUP);

  // Initialize sensor pins
  pinMode(tempPin, INPUT);  // Temperature sensor pin
  pinMode(gasPin, INPUT);   // Gas sensor pin
  pinMode(pirPin, INPUT);   // PIR sensor pin

  // BOT started !!!! NOW READY
  tone(PINBUZZ, 300);  // BEEP AGAIN FOR SUCCESS
```

```
delay(200);

noTone(PINBUZZ);

delay(200);

tone(PINBUZZ, 300);

delay(200);

noTone(PINBUZZ);

delay(200);


show_lights(0, 255, 0);

lcd.setCursor(0, 0);

lcd.print("  Rolls Royce ");

lcd.setCursor(2, 1);

lcd.print("  Loading ... ");

delay(700);


show_lights();

lcd.setCursor(0, 0);

lcd.print(" Status : ok ");

lcd.setCursor(2, 1);

lcd.print("                        ");


// BOT started !!!! NOW READY

tone(PINBUZZ, 300);  // BEEP AGAIN FOR SUCCESS

delay(200);

noTone(PINBUZZ);

delay(200);
```

```
  tone(PINBUZZ, 300);
  delay(200);
  noTone(PINBUZZ);
  delay(200);

  delay(500);
}

void loop() {

  // Print the button states on the second row (line 2)
  lcd.setCursor(2, 1);  // Move the cursor back to the start of line 2
  for (int i = 0; i < 6; i++) {
    lcd.print(!buttonState[i]);
    lcd.print(" ");  // Space between values
  }

  // Print to Serial Monitor as well for debugging
  Serial.print("Button States: ");
  for (int i = 0; i < 6; i++) {
    Serial.print(!buttonState[i]);
    Serial.print(" ");
  }
  Serial.println();

  // Read the temperature sensor (LM35 at A5)
  int tempValue = analogRead(tempPin);

  float temperatureC = (tempValue * 5.0 * 100.0) / 1024.0;  // LM35 in Celsius


  // Read the gas sensor (analog reading from A4)

  int gasValue = analogRead(gasPin);

  bool gasDetected = (gasValue > 100);  // Threshold for detecting harmful gas, adjust as needed


  // Read the PIR sensor (detect motion)

  bool motionDetected = digitalRead(pirPin);  // HIGH means motion detected


  // Limit switch reading with debounce logic

  unsigned long currentMillis = millis();


  // Read each limit switch and debounce

  buttonState[0] = digitalRead(frontLeftDoorPin);

  if (buttonState[0] != lastButtonState[0] && (currentMillis - lastDebounceTime[0]) >
debounceDelay) {

    lastDebounceTime[0] = currentMillis;

    lastButtonState[0] = buttonState[0];

  }


  buttonState[1] = digitalRead(frontRightDoorPin);

  if (buttonState[1] != lastButtonState[1] && (currentMillis - lastDebounceTime[1]) >
debounceDelay) {

    lastDebounceTime[1] = currentMillis;

    lastButtonState[1] = buttonState[1];

  }
  buttonState[2] = digitalRead(backLeftDoorPin);
  if (buttonState[2] != lastButtonState[2] && (currentMillis - lastDebounceTime[2]) >
debounceDelay) {
    lastDebounceTime[2] = currentMillis;
    lastButtonState[2] = buttonState[2];
  }

  buttonState[3] = digitalRead(backRightDoorPin);
  if (buttonState[3] != lastButtonState[3] && (currentMillis - lastDebounceTime[3]) >
debounceDelay) {
    lastDebounceTime[3] = currentMillis;
    lastButtonState[3] = buttonState[3];
  }

  buttonState[4] = digitalRead(frontBonnetPin);
  if (buttonState[4] != lastButtonState[4] && (currentMillis - lastDebounceTime[4]) >
debounceDelay) {
    lastDebounceTime[4] = currentMillis;
    lastButtonState[4] = buttonState[4];
  }

  buttonState[5] = digitalRead(backTrunkPin);
  if (buttonState[5] != lastButtonState[5] && (currentMillis - lastDebounceTime[5]) >
debounceDelay) {
    lastDebounceTime[5] = currentMillis;
    lastButtonState[5] = buttonState[5];
```

```
}

// Read limit switch states (debounced)
int frontLeftDoor = !buttonState[0];
int frontRightDoor = !buttonState[1];
int backLeftDoor = !buttonState[2];
int backRightDoor = !buttonState[3];
int frontBonnet = !buttonState[4];
int backBonnet = !buttonState[5];

// If gas is detected, send a warning


if (gasDetected || motionDetected) {
  beep();
  if (gasDetected && motionDetected) {
    Serial.println("WARNING: Harmful gas detected! Please check.");
    Serial.println("ALERT: Motion detected near the car.");
    show_lights(255);
    lcd.setCursor(0, 0);
    lcd.print("WARNING: Harmful gas detected");
    lcd.setCursor(2, 1);
    lcd.print("ALERT: Motion detected");
  } else if (gasDetected) {
    Serial.println("WARNING: Harmful gas detected! Please check.");
    show_lights(255);
    lcd.setCursor(0, 0);
    lcd.print("WARNING: gas");
  }

  // If motion is detected, send an alert
  else if (motionDetected) {
    Serial.println("ALERT: Motion detec");
    show_lights(255);
    lcd.setCursor(0, 0);
    lcd.print("ALERT: Motion dete");
  }
}

else {
  lcd.setCursor(0, 0);
  lcd.print(" Status : ok ");
  show_lights(0, 0, 0);
}



// Wait for a request from Python
if (Serial.available() > 0) {
  char request = Serial.read();  // Read the incoming request

  if (request == 'R') {  // If 'R' is received, send status of all data
```

```
    // Send the states back to Python (limit switches, temp, gas, motion)
    Serial.print(frontLeftDoor);
    Serial.print(",");
    Serial.print(frontRightDoor);
    Serial.print(",");
    Serial.print(backLeftDoor);
    Serial.print(",");
    Serial.print(backRightDoor);
    Serial.print(",");
    Serial.print(frontBonnet);
    Serial.print(",");
    Serial.print(backBonnet);
    Serial.print(",");
    Serial.print(temperatureC);  // Send temperature
    Serial.print(",");
    Serial.print(gasDetected);  // Send gas detection status (true/false)
    Serial.print(",");
    Serial.println(motionDetected);  // Send PIR motion detection status (true/false)

  }

  else if (request == 'O') {  // If 'O' is received, perform some operation based on Python's
command
    // Perform a sample operation like turning on/off an LED (example: you could control other
actuators based on Python's command)
    digitalWrite(LED_BUILTIN, HIGH);  // Turn on the onboard LED (just an example)
```

```
      Serial.println("Commmand received ");
      show_lights(0, 0, 255);
      lcd.setCursor(0, 0);
      lcd.print("Automatic System says ");
      lcd.setCursor(2, 1);
      lcd.print("From Python to arduino !!");
    }
  }
}
```