NAME : S.AKSHAYA

IBM NAAN MUDHALVAN

REGISTER NO: 312821106003

IOT PHASE 4 project

SMART PUBLIC RESTROOM

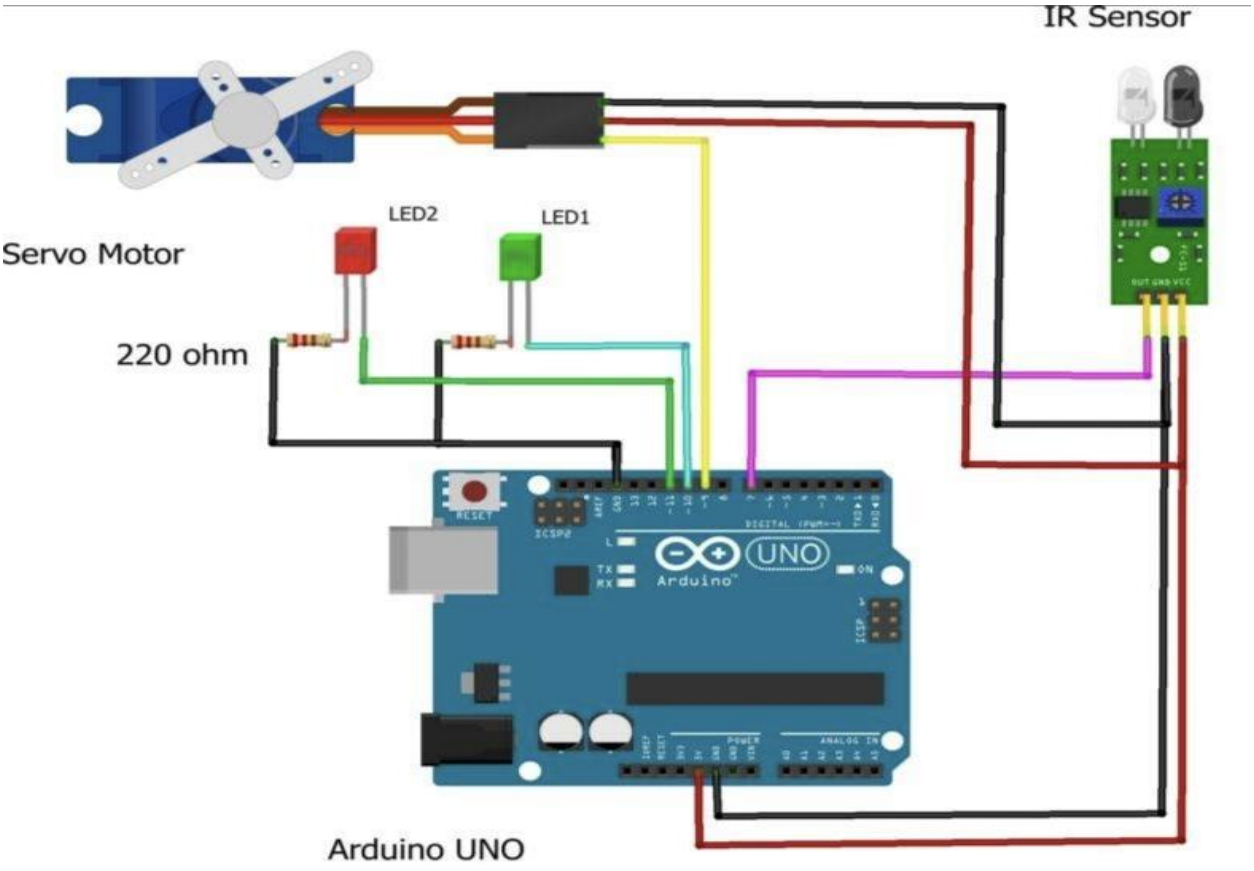
PROJECT : SMART PUBLIC RESTROOMS

AUTOMATIC DOOR SYSTEM

Components:

- Arduino UNO
- Servo motor
- IR sensor
- Red and green LEDs
- 220 ohm resistor
- Jumper wires and a breadboard
- USB cable for uploading the code

Circuit:



Code:

```
#include <Servo.h>
Servo s1;
Int val = 0 ;
Void setup()
{
   Serial.begin(9600); // sensor buart rate
   S1.attach(3);
   pinMode(2,INPUT);
   pinMode(5,OUTPUT);  // led green pin
   pinMode(6,OUTPUT);  // led red pin
}
Void loop()
{
 Val = digitalRead(2);  // IR sensor output pin connected
 Serial.println(val);   // see the value in serial mpnitor in Arduino IDE
 Delay(1);

 If(val == 1 )
 {
  digitalWrite(5,HIGH);  // LED ON
  digitalWrite(6,LOW);   // LED OFF
  s1.write(90);
  delay(2000);

 }
 Else
```

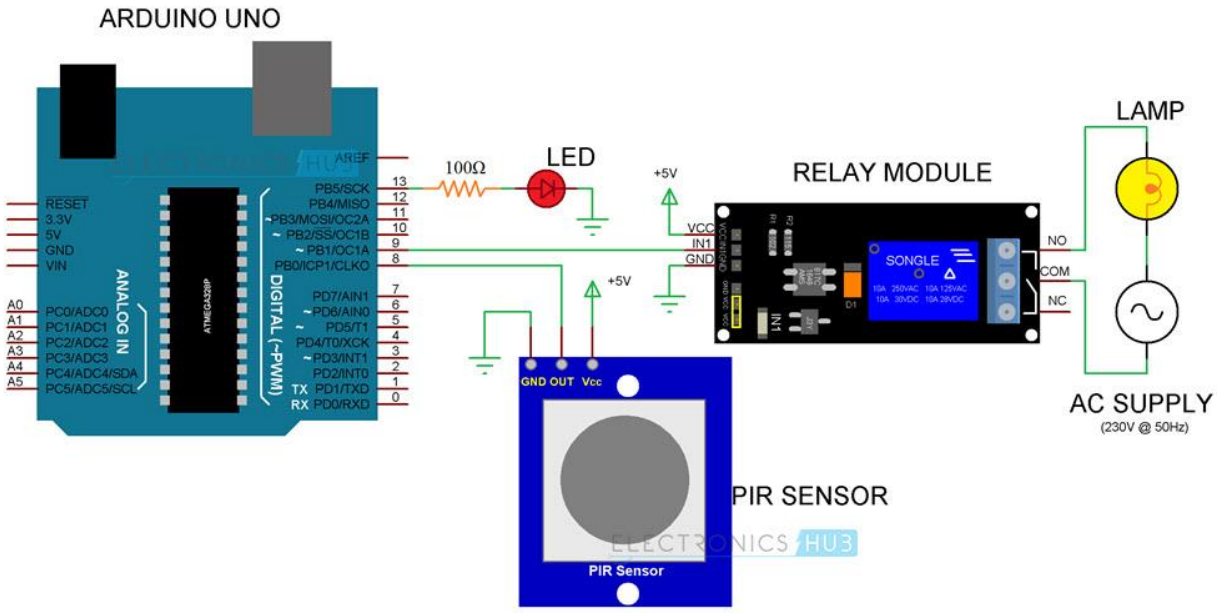
```
{
  digitalWrite(5,LOW);   // LED OFF
  digitalWrite(6,HIGH);  // LED ON
  s1.write(0);

}
}
```

AUTOMATIC ROOM LIGHTING SYSTEM

Components:

- Arduino
- Arduino UNO [Buy Here]
- PIR Sensor
- 5V Relay Module (Relay Board)
- LED
- 100Ω Resistor (1/4 Watt)
- Connecting Wires
- Breadboard
- Power Supply

Circuit:

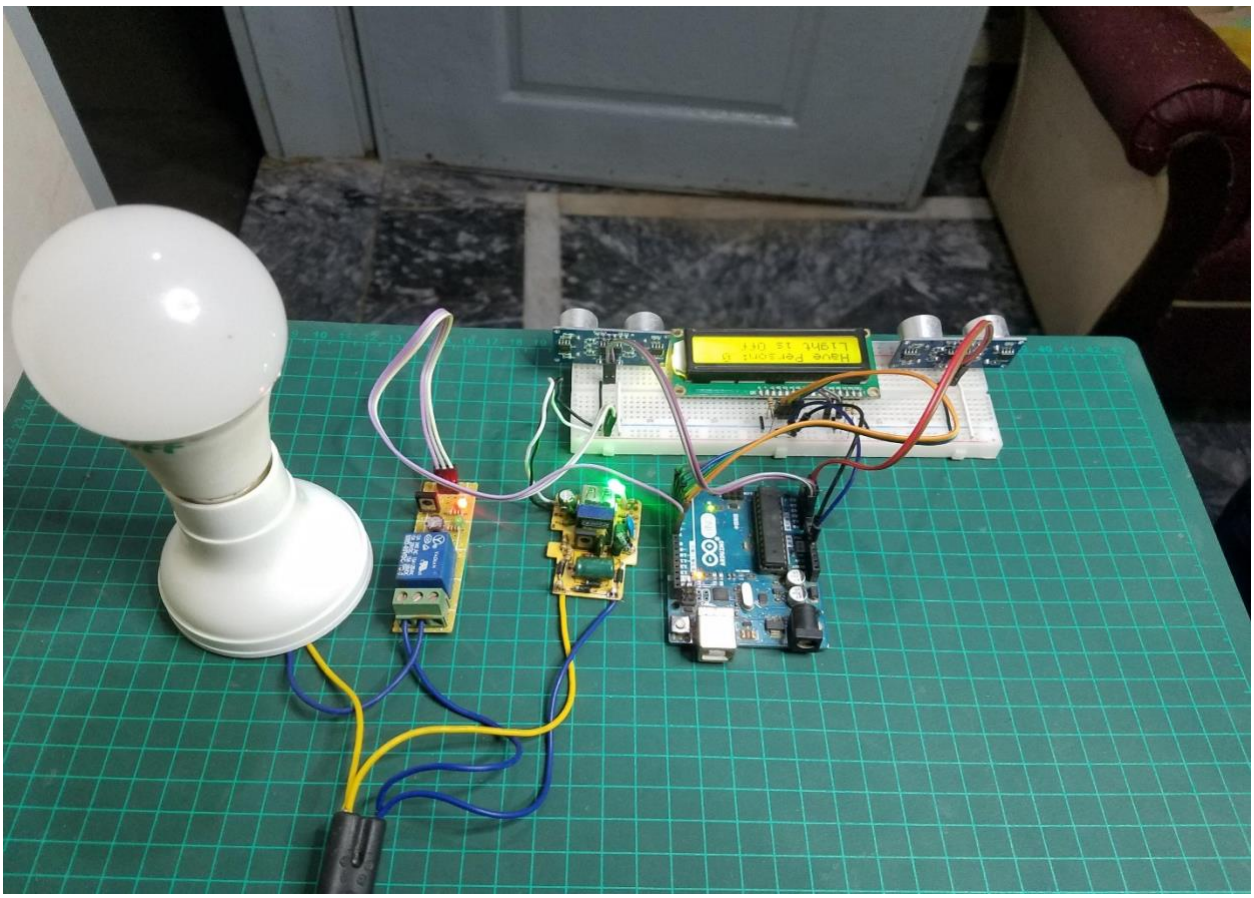


Code:

```
Int in1 = 9;

Int sensor = 8;

Int led = 13;

Unsigned long t=0;


Void setup()

{

  Serial.begin(9600);

  pinMode(in1, OUTPUT);

  pinMode(sensor, INPUT);

  pinMode(led, OUTPUT);


  digitalWrite(in1,HIGH);

  digitalWrite(led,LOW);
```

```
  while(millis()<13000)
  {
    digitalWrite(led,HIGH);
    delay(50);
    digitalWrite(led,LOW);
    delay(50);
  }
  digitalWrite(led,LOW);

}



Void loop()
{
  digitalWrite(in1,HIGH);
  digitalWrite(led,LOW);
  if(digitalRead(sensor)==HIGH)
  {
   T=millis();
   While(millis()<(t+5000))
   {
   digitalWrite(in1,LOW);
   digitalWrite(led,HIGH);
     if((millis()>(t+2300))&&(digitalRead(sensor)==HIGH))
      {
       T=millis();
      }
   }
  }
```

}

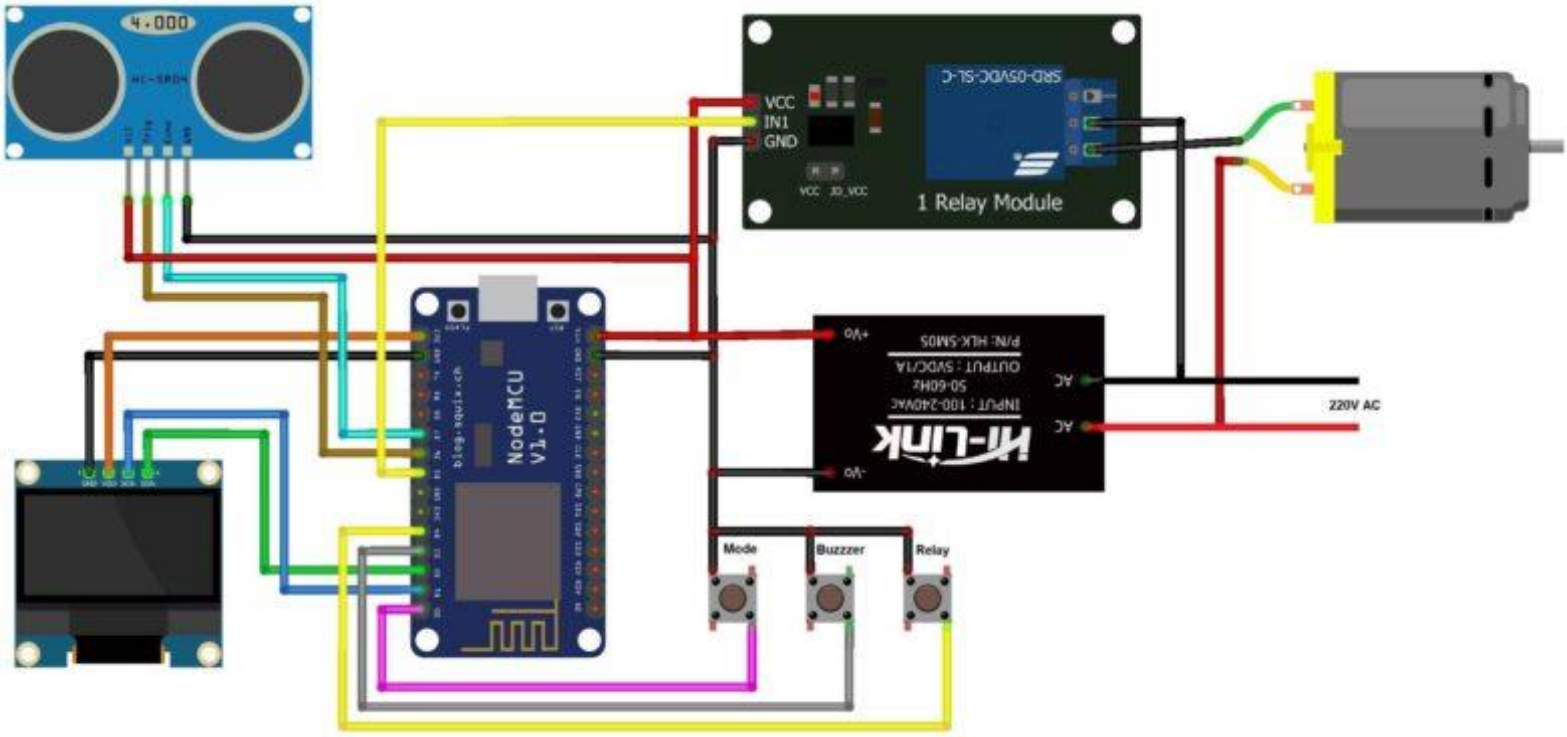

Have Person is
Light is Off

WATER LEVEL MONITORING AND CONTROL SYSTEM:

Components:

- NodeMCU ESP8266 WiFi Module
- JSN-SR04T Ultrasonic Sensor
- 0.96” I2C OLED Display
- Hi-Link 220V AC to 5V DC Converter
- Push Button Switch
- 5V Relay Module
- AC Water Pump
- Pipes 2 meter or more
- Zero PCB Board

Circuit:



Code:

A cloud storage is used at blynk.cloud

The follow is the complete code for IoT ESP8266 Based Ultrasonic Water Level Monitoring System. The code is written in Arduino IDE.

There are few changes that you need to make before uploading this code to the Esp8266 Board.

From the following lines change the Blynk Authentication Token.

#define BLYNK_TEMPLATE_ID “**********”

#define BLYNK_TEMPLATE_NAME “**********”

#define BLYNK_AUTH_TOKEN “**********”

Replace the WiFi SSID and Password from the following lines.

Char ssid[] = “**********”;

Char pass[] = “**********”;

After making these changes, you can now upload the code to the ESP8266 Board.

```
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <AceButton.h>

#define BLYNK_TEMPLATE_ID “**********”
#define BLYNK_TEMPLATE_NAME “**********”
#define BLYNK_AUTH_TOKEN “**********”

Char ssid[] = “**********”;
Char pass[] = “**********”;

Int emptyTankDistance = 160;
Int fullTankDistance = 20;
Int triggerPer = 20;

Using namespace ace_button;

#define TRIG 12             //D6
#define ECHO 13             //D7
#define Relay 14          //D5
#define BP1 2             //D0
```

```
#define BP2 13          //D3
#define BP3 15          //D4

#define V_B_1 V1
#define V_B_3 V3
#define V_B_4 V4

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

Float duration;
Float distance;
Int waterLevelPer;

Bool toggleRelay = false;
Bool modeFlag = true;
String currMode;

Char auth[] = BLYNK_AUTH_TOKEN;

ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);
ButtonConfig config3;
AceButton button3(&config3);
```

```
Void handleEvent1(AceButton*, uint8_t, uint8_t);
Void handleEvent2(AceButton*, uint8_t, uint8_t);
Void handleEvent3(AceButton*, uint8_t, uint8_t);

BlynkTimer timer;

Void checkBlynkStatus() {

  Bool isconnected = Blynk.connected();
  If (isconnected == false) {
  }
  If (isconnected == true) {
  }
}

BLYNK_WRITE(VPIN_BUTTON_3) {
  modeFlag = param.asInt();
  if (!modeFlag && toggleRelay) {
    digitalWrite(Relay, LOW);
    toggleRelay = false;
  }
  currMode = modeFlag ? “AUTO” : “MANUAL”;
}

BLYNK_WRITE(VPIN_BUTTON_4) {
  If (!modeFlag) {
    toggleRelay = param.asInt();
    digitalWrite(Relay, toggleRelay);
```

```
  } else {
    Blynk.virtualWrite(V_B_4, toggleRelay);
  }
}

BLYNK_CONNECTED() {
  Blynk.syncVirtual(V_B_1);
  Blynk.virtualWrite(V_B_3, modeFlag);
  Blynk.virtualWrite(V_B_4, toggleRelay);
}

Void displayData() {
  Display.clearDisplay();
  Display.setTextSize(3);
  Display.setCursor(30, 0);
  Display.print(waterLevelPer);
  Display.print(“ “);
  Display.print(“%”);
  Display.setTextSize(1);
  Display.setCursor(20, 25);
  Display.print(currMode);
  Display.setCursor(95, 25);
  Display.print(toggleRelay ? “ON” : “OFF”);
  Display.display();
}

Void measureDistance() {

  digitalWrite(TRIG, LOW);
```

```
delayMicroseconds(2);
digitalWrite(TRIG, HIGH);
delayMicroseconds(20);
digitalWrite(TRIG, LOW);
duration = pulseIn(ECHO, HIGH);
distance = ((duration / 2) * 0.343) / 10;
if (distance > (fullTankDistance – 15) && distance < emptyTankDistance) {
  waterLevelPer = map((int)distance, emptyTankDistance, fullTankDistance, 0, 100);
  Blynk.virtualWrite(V_B_1, waterLevelPer);
  If (waterLevelPer < triggerPer) {
    If (modeFlag) {
      If (!toggleRelay) {
        digitalWrite(Relay, HIGH);
        toggleRelay = true;
        Blynk.virtualWrite(V_B_4, toggleRelay);
      }
    }
  }
  If (distance < fullTankDistance) {
    If (modeFlag) {
      If (toggleRelay) {
        digitalWrite(Relay, LOW);
        toggleRelay = false;
        Blynk.virtualWrite(V_B_4, toggleRelay);
      }
    }
  }
}
displayData();
```

```
  delay(100);
}

Void setup() {
  Serial.begin(9600);
  pinMode(ECHO, INPUT);
  pinMode(TRIG, OUTPUT);
  pinMode(Relay, OUTPUT);

  pinMode(BP1, INPUT_PULLUP);
  pinMode(BP2, INPUT_PULLUP);
  pinMode(BP3, INPUT_PULLUP);

  digitalWrite(Relay, HIGH);

  config1.setEventHandler(button1Handler);
  config2.setEventHandler(button2Handler);
  config3.setEventHandler(button3Handler);

  button1.init(BP1);
  button2.init(BP2);
  button3.init(BP3);

  currMode = modeFlag ? “AUTO” : “MANUAL”;

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F(“SSD1306 allocation failed”));
    For (;;)
      ;
```

```
  }
  Delay(1000);
  Display.setTextSize(1);
  Display.setTextColor(WHITE);
  Display.clearDisplay();

  WiFi.begin(ssid, pass);
  Timer.setInterval(2000L, checkBlynkStatus);
  Timer.setInterval(1000L, measureDistance);
  Blynk.config(auth);
  Delay(1000);

  Blynk.virtualWrite(V_B_3, modeFlag);
  Blynk.virtualWrite(V_B_4, toggleRelay);

  Delay(500);
}

Void loop() {
  Blynk.run();
  Timer.run();
  Button1.check();
  Button3.check();

  If (!modeFlag) {
    Button2.check();
  }
}
```

```
Void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println(“EVENT1”);
  Switch (eventType) {
    Case AceButton::kEventReleased:
      If (modeFlag && toggleRelay) {
        digitalWrite(Relay, LOW);
        toggleRelay = false;
      }
      modeFlag = !modeFlag;
      currMode = modeFlag ? “AUTO” : “MANUAL”;
      Blynk.virtualWrite(V_B_3, modeFlag);
      Break;
  }
}

Void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println(“EVENT2”);
  Switch (eventType) {
    Case AceButton::kEventReleased:
      If (toggleRelay) {
        digitalWrite(Relay, LOW);
        toggleRelay = false;
      } else {
        digitalWrite(Relay, HIGH);
        toggleRelay = true;
      }
      Blynk.virtualWrite(V_B_4, toggleRelay);
      Delay(1000);
      Break;
```

```
  }
}

Void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT3");
  Switch (eventType) {
    Case AceButton::kEventReleased:
      Break;
  }
}
```

Arduino Code (Upload to Arduino Uno):

```
Const int triggerPin = 2;
Const int echoPin = 3;
Const int ledPin = 13;

Void setup() {
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

Void loop() {
  digitalWrite(triggerPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2;

  if (distance < 50) {
    digitalWrite(ledPin, HIGH); // Occupied
    Serial.println("Restroom is occupied.");
  } else {
    digitalWrite(ledPin, LOW); // Available
    Serial.println("Restroom is available.");
  }

  Delay(1000);
}
```

Arduino Code Output:

The Arduino code continuously measures the distance from an ultrasonic sensor.

***If the measured distance is less than 50 cm (you can adjust this threshold in the code), it considers the restroom as "occupied" and turns on the built-in LED on pin 13, which indicates that the restroom is in use.**

***If the distance is greater than or equal to 50 cm, it considers the restroom as “available” and turns off the LED.**

***The Arduino code also sends occupancy status messages over the serial port, which the Python program will read and display.**

Python Code:

```
Import serial
Import time

# Define the Arduino’s serial port
Ser = serial.Serial(‘COM3’, 9600)  # Replace ‘COM3’ with your Arduino’s port

While True:
   Try:
      # Read data from the Arduino
      Arduino_data = ser.readline().decode(‘utf-8’).strip()

      # Check occupancy status received from Arduino
      If “occupied” in arduino_data:
         Print(“Restroom is occupied.”)
      Else:
         Pr Restroom int(“Restroom is available.”)

      Time.sleep(2)
```

```
    Except KeyboardInterrupt:
        Ser.close()
        Break
```

Python Code Output:

Restroom is available.

Restroom is occupied.

Restroom is available.

Restroom is available.

The Python code establishes a serial connection with the Arduino using the specified COM port and baud rate.

***It continuously reads data from the serial port, expecting to receive occupancy status messages.**

***When it receives data, it checks if the received message contains “occupied” or “available” and then prints a corresponding message to the console.**

***Here's an example of the kind of output you might see in the Python console:**