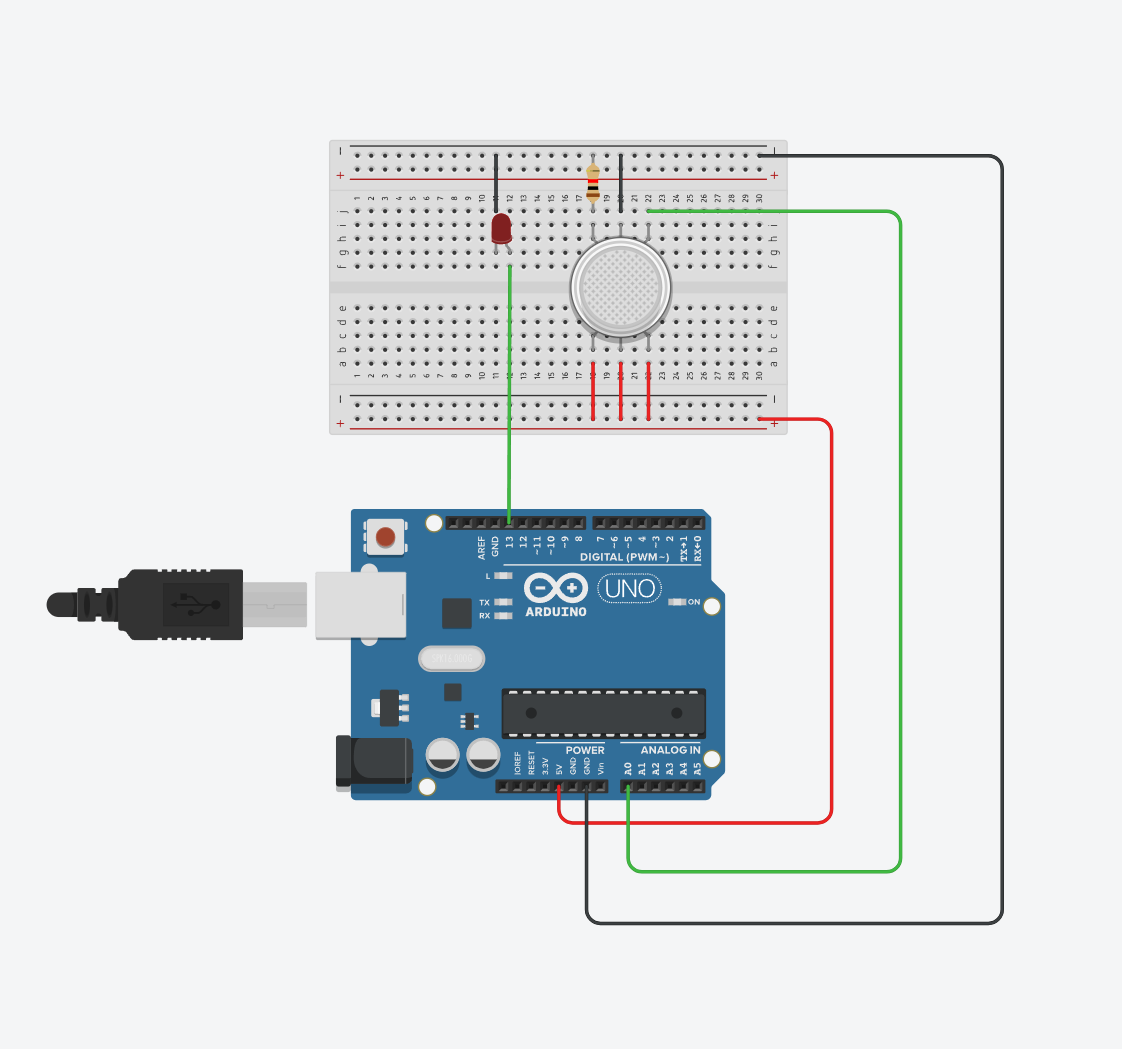
**SMART PUBLIC RESTROOM**

**IoT – PHASE – 3**

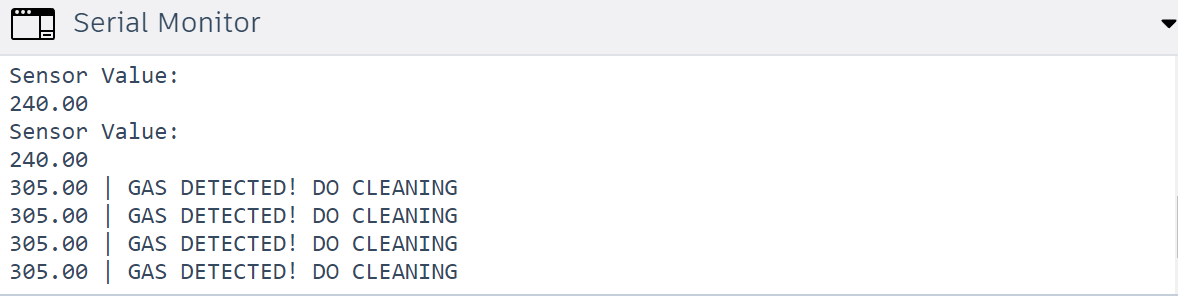
**DEVELOPMET – 1**

***Getting The Gas Sensor Setup***

|  |
| --- |
| **CODE**  int LED = 13;  int MQ2pin = A0;  void setup()  {  Serial.begin(9600);  }  void loop()  {  float sensorValue;  sensorValue = analogRead(MQ2pin);  if(sensorValue >= 250)  {  digitalWrite(LED, HIGH);  Serial.print(sensorValue);  Serial.println(" | GAS DETECTED! DO CLEANING");  }  else  {  digitalWrite(LED, LOW);  Serial.println("Sensor Value: ");  Serial.println(sensorValue);  }  delay(1000);  } |



ARDUINO SETUP

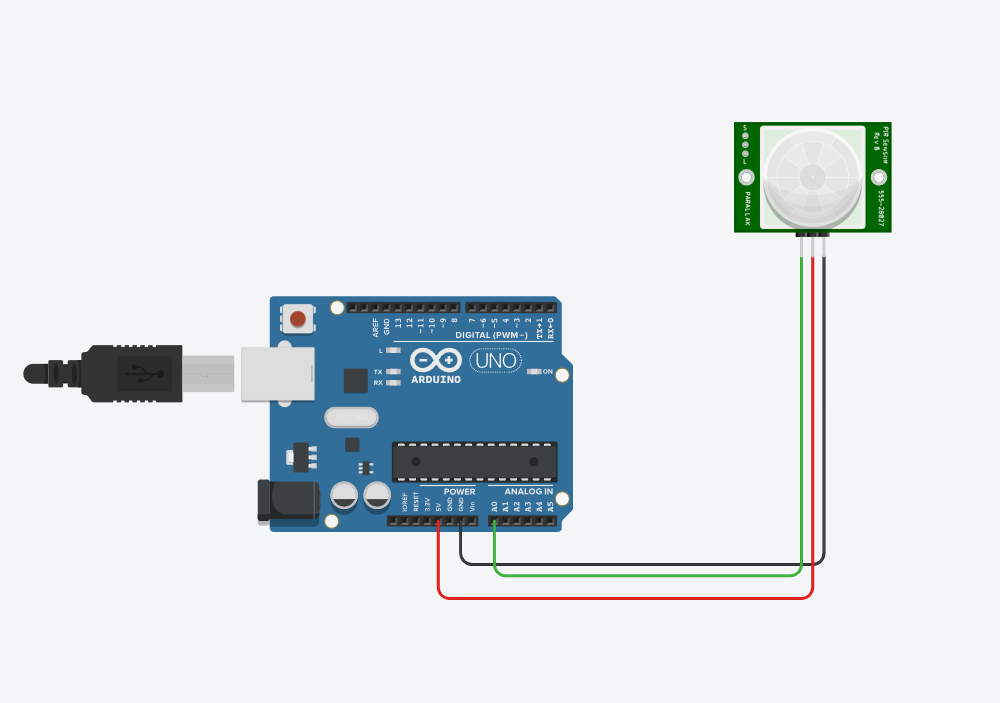


OUTPUT

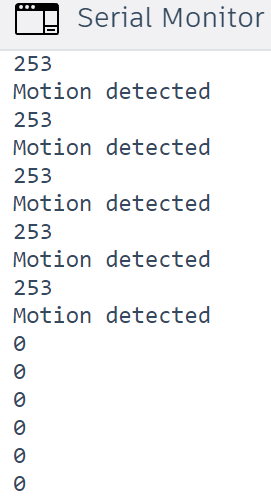
|  |
| --- |
| ***Python script for the same***  import RPi.GPIO as GPIO  import time  LED\_PIN = 13  MQ2\_PIN = 0 # This corresponds to A0 in Arduino nomenclature  def setup():  GPIO.setmode(GPIO.BCM)  GPIO.setup(LED\_PIN, GPIO.OUT)  GPIO.setup(MQ2\_PIN, GPIO.IN)  def loop():  while True:  sensor\_value = GPIO.input(MQ2\_PIN)  if sensor\_value == GPIO.HIGH:  GPIO.output(LED\_PIN, GPIO.HIGH)  print(f"{sensor\_value} | GAS DETECTED! DO CLEANING")  else:  GPIO.output(LED\_PIN, GPIO.LOW)  print("Sensor Value:")  print(sensor\_value)  time.sleep(1)  if \_\_name\_\_ == '\_\_main\_\_':  try:  setup()  loop()  except KeyboardInterrupt:  GPIO.cleanup() |

***Setting Up the PIR Sensor***

|  |
| --- |
| **CODE**  int a = 0;  int b = 0;  void setup()  {  Serial.begin(9600);  pinMode(13, OUTPUT);  }  void loop()  {  a = analogRead(A0);  b = map(a,0,1023,0,255);  Serial.println(b);  if(b>100)  {  Serial.println("Motion detected");  delay(100);  }  } |



ARDUINO SETUP



OUTPUT

|  |
| --- |
| ***Python code for the same***  import RPi.GPIO as GPIO  import time  SENSOR\_PIN = 0 # This corresponds to A0 in Arduino nomenclature  LED\_PIN = 13  def setup():  GPIO.setmode(GPIO.BCM)  GPIO.setup(LED\_PIN, GPIO.OUT)  def loop():  while True:  a = GPIO.input(SENSOR\_PIN)  b = int((a / 1023) \* 255)  print(b)  if b > 100:  print("Motion detected")  GPIO.output(LED\_PIN, GPIO.HIGH)  time.sleep(0.1)  else:  GPIO.output(LED\_PIN, GPIO.LOW)  time.sleep(0.1)  if \_\_name\_\_ == '\_\_main\_\_':  try:  setup()  loop()  except KeyboardInterrupt:  GPIO.cleanup() |

***To Setup a Real Time IoT Platform***

*The above setups just depict the simulation of the IoT devices used. For a real-time IoT device interpretation and an IoT platform to be set up, the following steps are required:*

***Bluetooth Communication (Optional):*** If you want to connect your mobile application directly to the Arduino, you can use Bluetooth communication. You'll need a Bluetooth module (e.g., HC-05) on the Arduino and a Bluetooth library in your mobile app. When gas is detected, the Arduino can send a Bluetooth message, and the mobile app can receive and display it.

***Internet Communication:*** Alternatively, you can use the internet to send data from your Arduino to a server, and the mobile app can periodically check the server for updates. This approach allows your Arduino to connect to the internet through Wi-Fi or Ethernet shields.

***Set up an IoT platform or a web server that can receive and store messages from your Arduino.***

Modify the Arduino code to send an HTTP POST request to your server when gas is detected. You can use libraries like Ethernet or ESP8266WiFi for internet connectivity.

In your mobile app, use HTTP requests (e.g., HttpURLConnection for Android or URLSession for iOS) to periodically check the server for gas detection updates.

Push Notifications: To make the system more user-friendly, you can implement push notifications in your mobile app. When gas is detected (based on data received from the server), your app can send a push notification to the user's device.

In your mobile app, use a push notification service like Firebase Cloud Messaging (FCM) for Android or Apple Push Notification Service (APNs) for iOS.

When the app detects gas via the server's data, it can send a push notification to the user's device. The user will receive a real-time notification on their mobile device.

***User Interface:*** In your mobile app, create a user-friendly interface that displays the gas detection status and notifications. This can include alerts, notifications, and logs of gas detection events.

***Testing and Deployment:*** Test your mobile app and Arduino setup thoroughly. Once you are satisfied with its performance, you can deploy it for real-world use.