

DIY IOT HARDWARE PROJECT

CS578

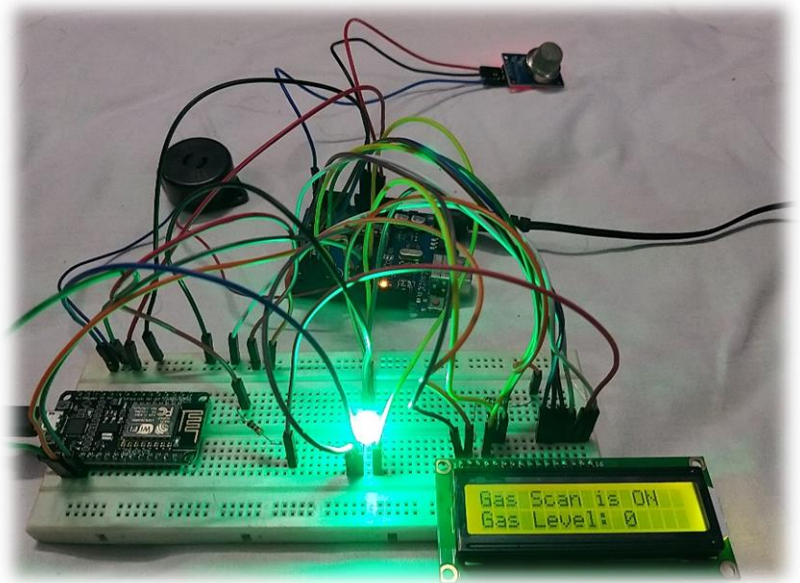
Name: Abhishek Saharia

ROLL: 180122002

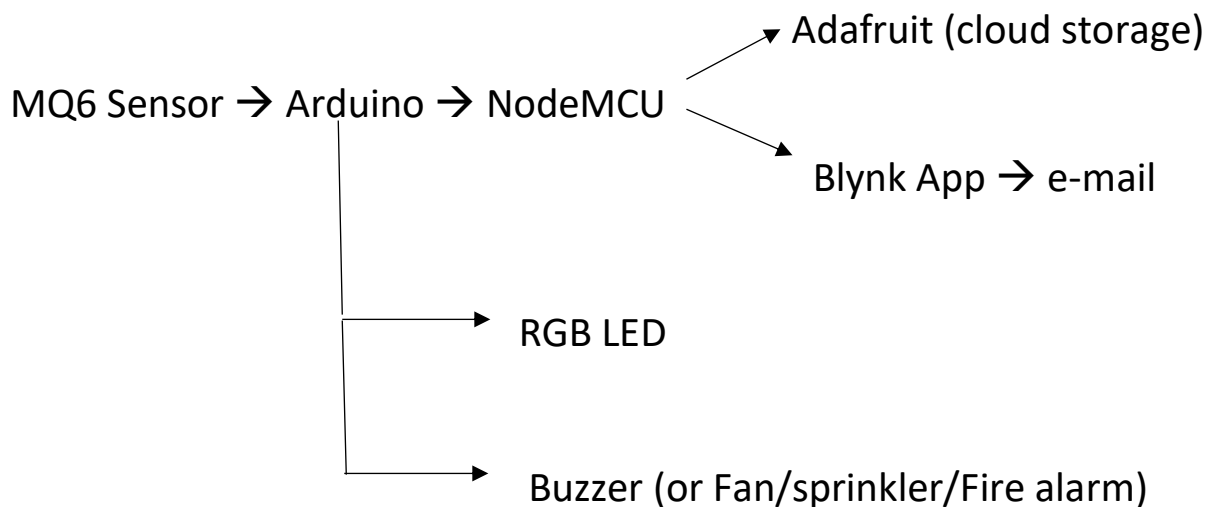
OBJECTIVE: To create an intelligent Gas Detector System using IoT

MATERIALS:

- Arduino UNO
- NodeMCU ESP8266
- MQ6 Gas Sensor
- 1k ohm resistance
- LCD Display (16x2)
- RGB LED
- Buzzer
- 12V Adapter
- Breadboard
- Smartphone
- Laptop/Desktop
- Cables



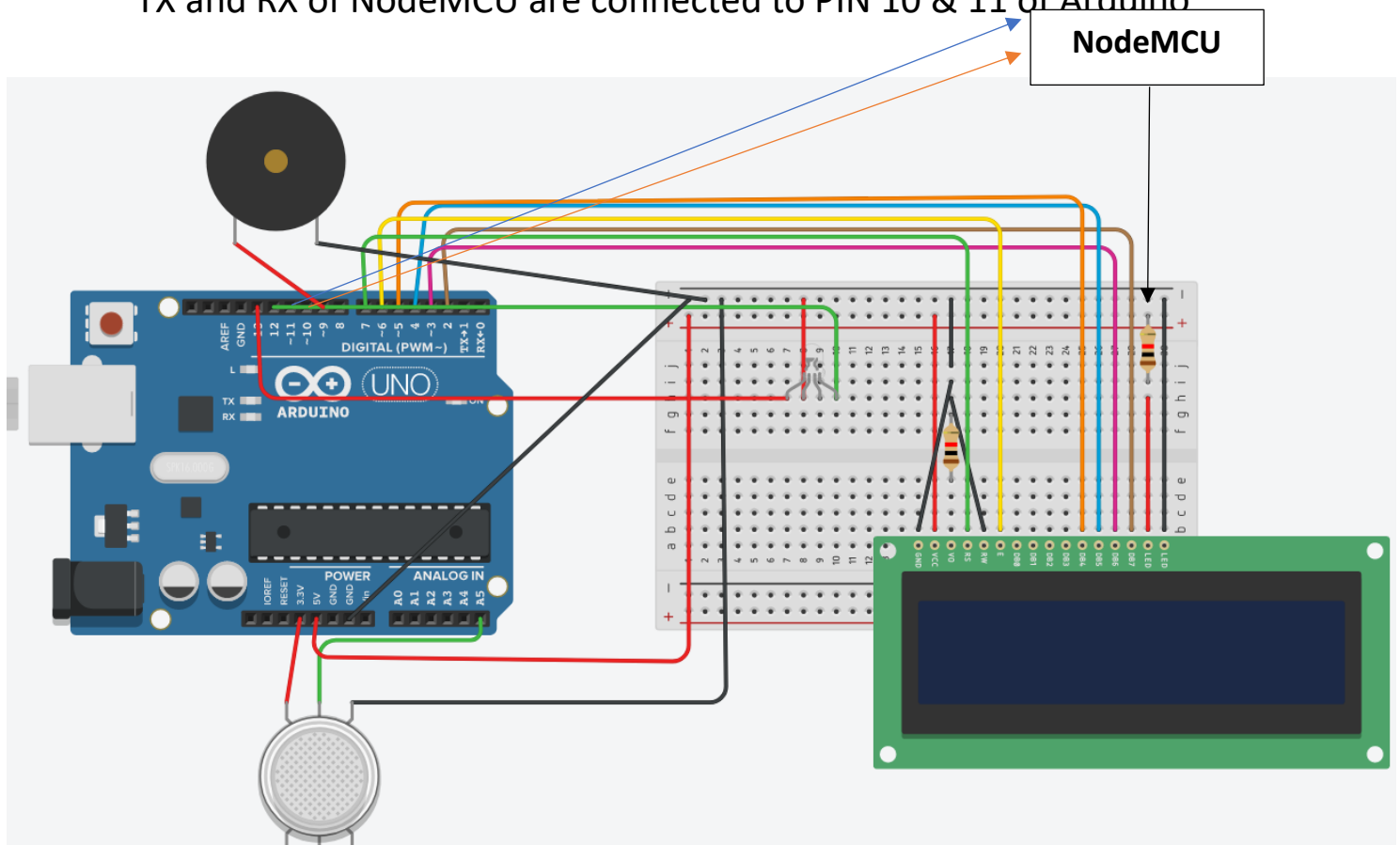
DATA FLOW:



CHARACTERISTICS: Sensor senses the Amount of Gas and sends analog signal every 5 seconds to Arduino. Arduino send the data to NodeMCU using Serial communication. The NodeMCU sends data to **adafruit cloud** through **MQTT** over **TCP/IP** and **blynk**. Blynk App gets updated at real time and the values can be monitored from the smartphone. An e-mail is sent if the data is above the threshold followed by buzzer and other actions. Data is also stored on the cloud using adafruit where data history is available for a month.

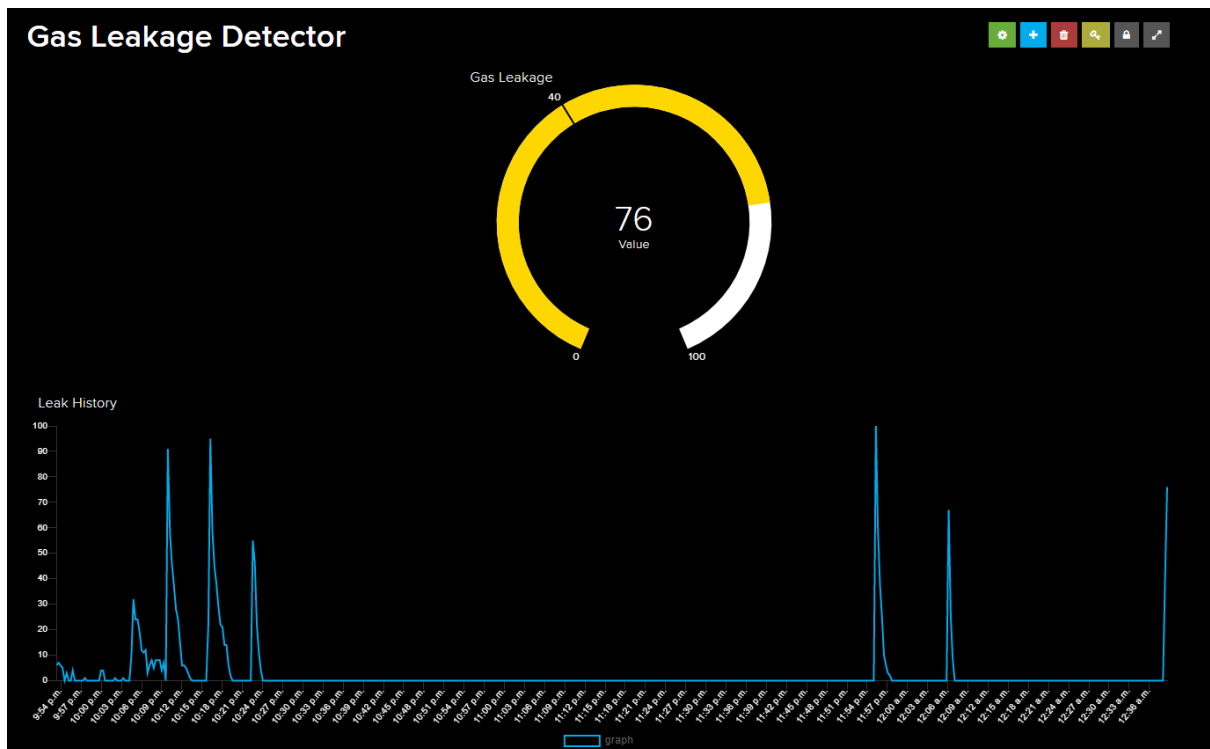
CONFIGURATION DIAGRAM (Without including the NodeMCU):

TX and RX of NodeMCU are connected to PIN 10 & 11 of Arduino

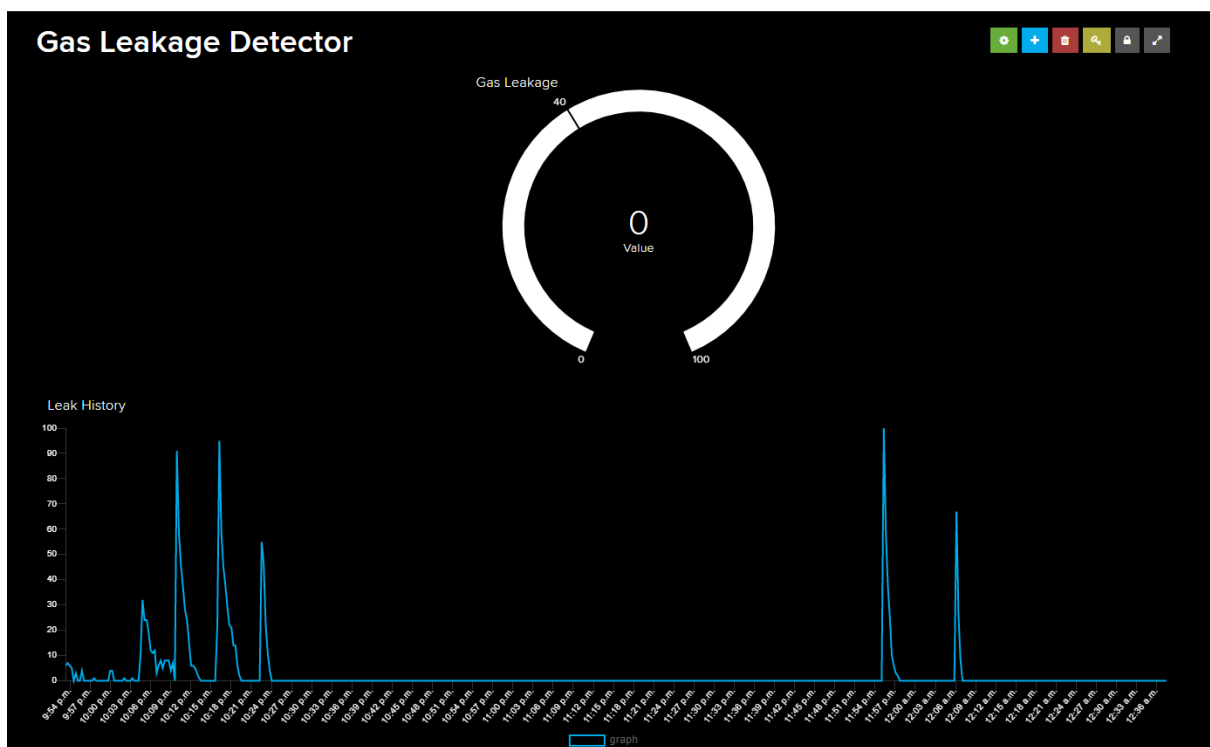


SAMPLE OUTPUTS (on adafruit):

When above Threshold:

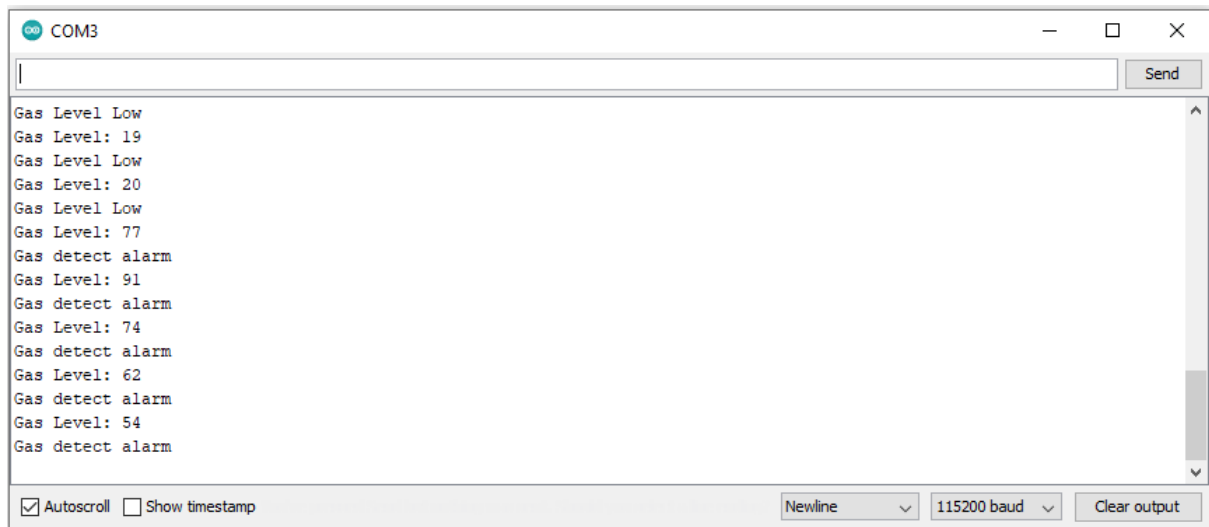


When below Threshold:



SAMPLE OUTPUTS (contd.):

a. Arduino Serial Monitor:



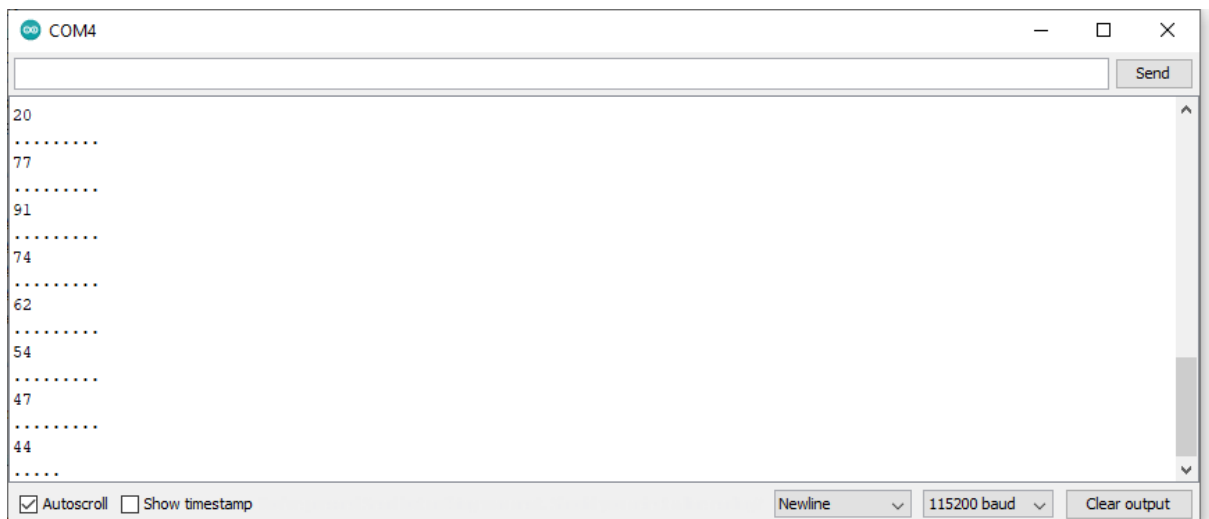
The screenshot shows the Arduino Serial Monitor window titled "COM3". The main text area displays the following data:

```
Gas Level Low
Gas Level: 19
Gas Level Low
Gas Level: 20
Gas Level Low
Gas Level: 77
Gas detect alarm
Gas Level: 91
Gas detect alarm
Gas Level: 74
Gas detect alarm
Gas Level: 62
Gas detect alarm
Gas Level: 54
Gas detect alarm
```

The bottom control bar includes the following options:

- ☒ Autoscroll
- ☐ Show timestamp
- Newline (dropdown menu)
- 115200 baud (dropdown menu)
- Clear output (button)

b. NodeMCU Serial Monitor (After Data is sent by the Arduino):



The screenshot shows the NodeMCU Serial Monitor window titled "COM4". The main text area displays the following data:

```
20
.....
77
.....
91
.....
74
.....
62
.....
54
.....
47
.....
44
.....
```

The bottom control bar includes the following options:

- ☒ Autoscroll
- ☐ Show timestamp
- Newline (dropdown menu)
- 115200 baud (dropdown menu)
- Clear output (button)

CODES:

a. Arduino Code:

```
1. #include <LiquidCrystal.h>
2. LiquidCrystal lcd(7,6,5,4,3,2);
3. #include<SoftwareSerial.h>
4.
5. SoftwareSerial ArduinoUno(10,11);
6.
7. //int gas = A0;
8. //int data = 0;
9. void setup() {
10.    // put your setup code here, to run once:
11.    //randomSeed(analogRead(0));
12.    ArduinoUno.begin(115200);
13.    Serial.begin(115200);
14.    lcd.begin(16,2);
15.    pinMode(A5,INPUT);
16.    lcd.print(" Gas Leakage ");
17.    lcd.setCursor(0,1);
18.    lcd.print("Detection System");
19.    delay(3000);
20.    lcd.clear();
21.    pinMode(13, OUTPUT);
22.    pinMode(12,OUTPUT);
23.    pinMode(9, OUTPUT);
24.    pinMode(10,INPUT);
25.    pinMode(11,OUTPUT);
26. }
27.
28. void loop() {
29.    // put your main code here, to run repeatedly:
30.    int data = analogRead(A5)-285;
31.    if(data<=0)
32.        data=0;
33.    if(data>=100)
34.        data=100;
35.    Serial.print("Gas Level: ");
36.    Serial.println(data);
37.    lcd.print ("Gas Scan is ON");
38.    lcd.setCursor(0,1);
39.    lcd.print("Gas Level: ");
40.    lcd.print(data);
41.    ArduinoUno.print(data);
42.    //ArduinoUno.println("\n");
43.    delay(1000);
44.
45.    if ( data > 40) //
46.    {
47.        //SendMessage();
48.        Serial.println("Gas detect alarm");
49.        lcd.clear();
50.        lcd.setCursor(0,0);
51.        lcd.print("Gas Level Exceed");
52.        delay(5000);
53.        digitalWrite(12, HIGH);
54.        digitalWrite(13, LOW);
55.        digitalWrite(9, HIGH);
56.    }
57.    else
58.    {
59.        Serial.println("Gas Level Low");
60.        lcd.clear();
```

```

61. lcd.setCursor(0,0);
62. lcd.print("Gas Level Normal");
63. delay(5000);
64. digitalWrite(13, HIGH);
65. digitalWrite(12, LOW);
66. digitalWrite(9, LOW);
67. }
68.
69. lcd.clear();
70. }

```

b. NodeMCU Code:

```

1. #include <SoftwareSerial.h>
2. #include <ESP8266WiFi.h>
3. #include "Adafruit_MQTT.h"
4. #include "Adafruit_MQTT_Client.h"
5. #include <BlynkSimpleEsp8266.h>
6. #include <math.h>
7. int data=0;
8. int ctr=0;
9. int chk=1;
10. int tme=0;
11. /***** Sketch Code *****/
12. #define R_SSID      "Home Wifi"
13. #define PASS        //I removed the password from here before putting on the Project
    Report
14.
15. /***** Adafruit.io Setup *****/
16.
17. #define AIO_SERVER    "io.adafruit.com"
18. #define AIO_SERVERPORT 1883           // use 8883 for SSL
19. #define AIO_USERNAME  "Sherlock149"
20. #define AIO_KEY        //I removed the password from here before putting on the Pr
    oject Report
21. #define auth          //I removed the password from here before putting on the Pr
    oject Report
22.
23. /***** Global State *****/
24.
25. WiFiClient client;
26. // or... use WiFiClientSecure for SSL
27. //WiFiClientSecure client;
28.
29. Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
30.
31. /***** Feeds *****/
32.
33.
34. Adafruit_MQTT_Publish Leakage = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/Leak");
35.
36. Adafruit_MQTT_Publish Graph = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/graph");
37. void setup(){
38.     Serial.begin(115200);
39.     //NodeMCU.begin(9600);
40.     //pinMode(D3,INPUT);

```

```

41. //pinMode(D2,OUTPUT);
42. delay(10);
43.
44. // Connect to WiFi access point.
45. Serial.println(); Serial.println();
46. Serial.print("Connecting to ");
47. Serial.println(R_SSID);
48.
49. WiFi.begin(R_SSID, PASS);
50. int counter=10;
51. while (WiFi.status() != WL_CONNECTED) {
52.     delay(500);
53.     Serial.print(".");
54.     counter--;
55.     if(counter==0){
56.         // reset me
57.         ESP.reset();
58.     }
59. }
60. Serial.println();
61.
62. Serial.println("WiFi connected");
63. Serial.println("IP address: "); Serial.println(WiFi.localIP());
64. Blynk.begin(auth, R_SSID, PASS);
65. }
66.
67. void loop(){
68.     //while(NodeMCU.available()>0){
69.     MQTT_connect();
70.     Blynk.run();
71.     int val = Serial.read()-48;
72.     if(val>=0 && ctr==0)
73.     {
74.         chk=0;
75.         data=0;
76.         Serial.println("");
77.     }
78.     if(val>=0)
79.     {
80.         if(ctr==0)
81.             data = data+int(val)*100;
82.         if(ctr==1)
83.             data = data+int(val)*10;
84.         if(ctr==2)
85.             data = data+int(val);
86.
87.         //Serial.print(val);
88.         //val= -1;
89.         //Serial.print(ctr);
90.
91.         ctr++;
92.     }
93.     else if(chk==0)
94.     {
95.         if(ctr==2)
96.         {
97.             data = data/10;
98.         }
99.         else if(ctr==1)
100.        {
101.            data = data/100;
102.        }
103.        Serial.println(data);
104.        Blynk.virtualWrite(V2, data);
105.        if(data>40)
106.            Blynk.notify("Gas Leak !");

```

```

107.         if(tme>=30)
108.         {
109.             if (! Leakage.publish(data)) {
110.                 Serial.println(F("Upload Failed"));
111.             }
112.             if (! Graph.publish(data)) {
113.                 Serial.println(F("Upload Failed"));
114.             }
115.             tme = 0;
116.         }
117.         ctr=0;
118.         chk=1;
119.     }
120.     else
121.     {
122.         Serial.print(".");
123.         ctr=0;
124.     }
125.     delay(500);
126.     tme++;
127. }
128.
129. void MQTT_connect() {
130.     int8_t ret;
131.
132.     // Stop if already connected.
133.     if (mqtt.connected()) {
134.         return;
135.     }
136.
137.     Serial.print("Connecting to MQTT... ");
138.
139.     uint8_t retries = 5;
140.     while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connect
ed
141.         Serial.println(mqtt.connectErrorString(ret));
142.         Serial.println("Retrying MQTT connection in 5 seconds...");
143.         mqtt.disconnect();
144.         delay(5000); // wait 5 seconds
145.         retries--;
146.         if (retries == 0) {
147.             // reset me
148.             ESP.reset();
149.         }
150.     }
151.     Serial.println("MQTT Connected!");
152. }

```


USER MANUAL:

1. Connect the circuit as shown in the diagram
2. 1k resistors are used everywhere
3. In some RGB LEDs the long terminal should be connected to ground while in others to Vcc. So, make sure it is correct.
4. 12V adapter should be used so that all components get required power.
5. Create an account at <https://io.adafruit.com/> and follow the steps to create a dashboard. Name the feed as graph and leak for the graph and gauge respectively.
6. Put the adafruit authentication code at AIO_key in NodeMCU code
7. Download Blynk App on smartphone and put the authentication code to auth in NodeMCU code
8. Put the gauge on virtual pin V2. Send e-mail instructions using the eventer.
9. Connect Arduino and NodeMCU to PC and launch Arduino IDE. Make sure all needed libraries are included.
10. After confirming that the ports and board are correct, upload the code to both of them separately.
11. While uploading, make sure the TX and RX pin of NodeMCU are not connected.
12. After uploading, connect the TX and RX pin and check the output on Serial monitor. Any portable lighter can be used to mimic gas leakage, check the same on Blynk app and Adafruit.

*The gas sensor takes around 10mins to calibrate properly when turned on for the first time.