

# IOT SMART KITCHEN

## TEAM MEMBERS:

- 1.Sai Charan (1602-20-735-152)
  2. Sanjana (1602-20-735-157)
  3. Snehith (1602-20-735-313)
-

# INTRODUCTION

The kitchen is one of the important places in a house. The safety factor is the main aspect that must be taken into account during the activity in the kitchen. The existence of gas leakage, uncontrolled fire, excessive temperatures & a moist environment must be quickly identified and addressed. Apart from this, it is necessary to monitor & control Kitchen Appliances like lights, fridge, oven, etc. remotely.

The main motto of this project is to make a prototype of an IoT Based Smart Kitchen using the Internet of Things. This can be useful to turn on exhaust when gas starts leaking when we are not at home. The system uses multiple sensors, relays & NodeMCU ESP8266 Board. We can monitor all the sensor data on Blynk Applications. We can also send the command from Blynk App to control Kitchen Appliances for applications like pre heating microwave oven.

# PLAN OF ACTION

WEEK 1: Procuring information and required components

WEEK 2: Preparing the literature survey

WEEK 3: Hardware assembly

WEEK 4: Software installation and configuration

WEEK 5: Integrating the hardware and software

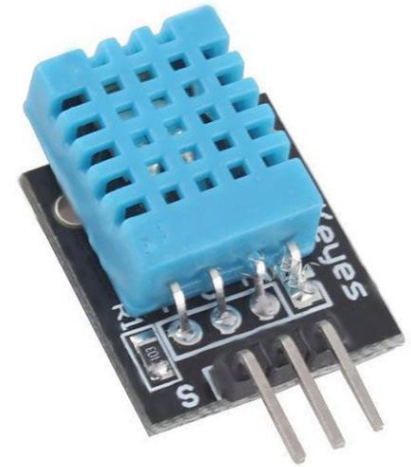
WEEK 6: Testing the prototype

WEEK 7: Final testing and debugging

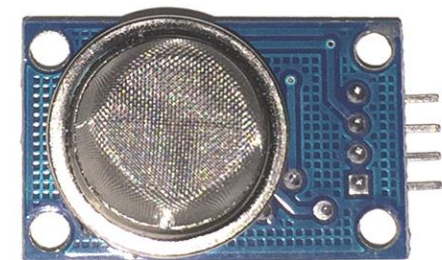
# SENSORS USED

The **DHT11** is a commonly used **Temperature and humidity sensor**. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of  $\pm 1^\circ\text{C}$  and  $\pm 1\%$ .

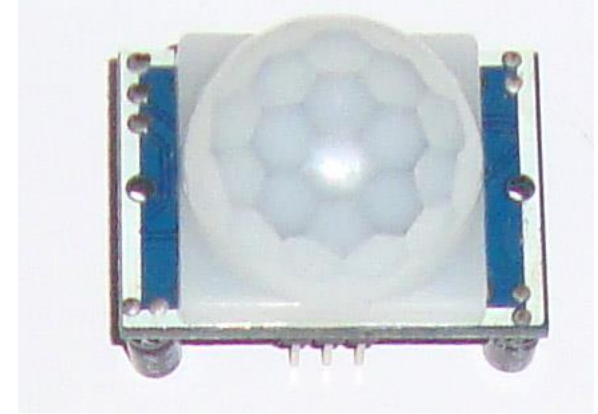


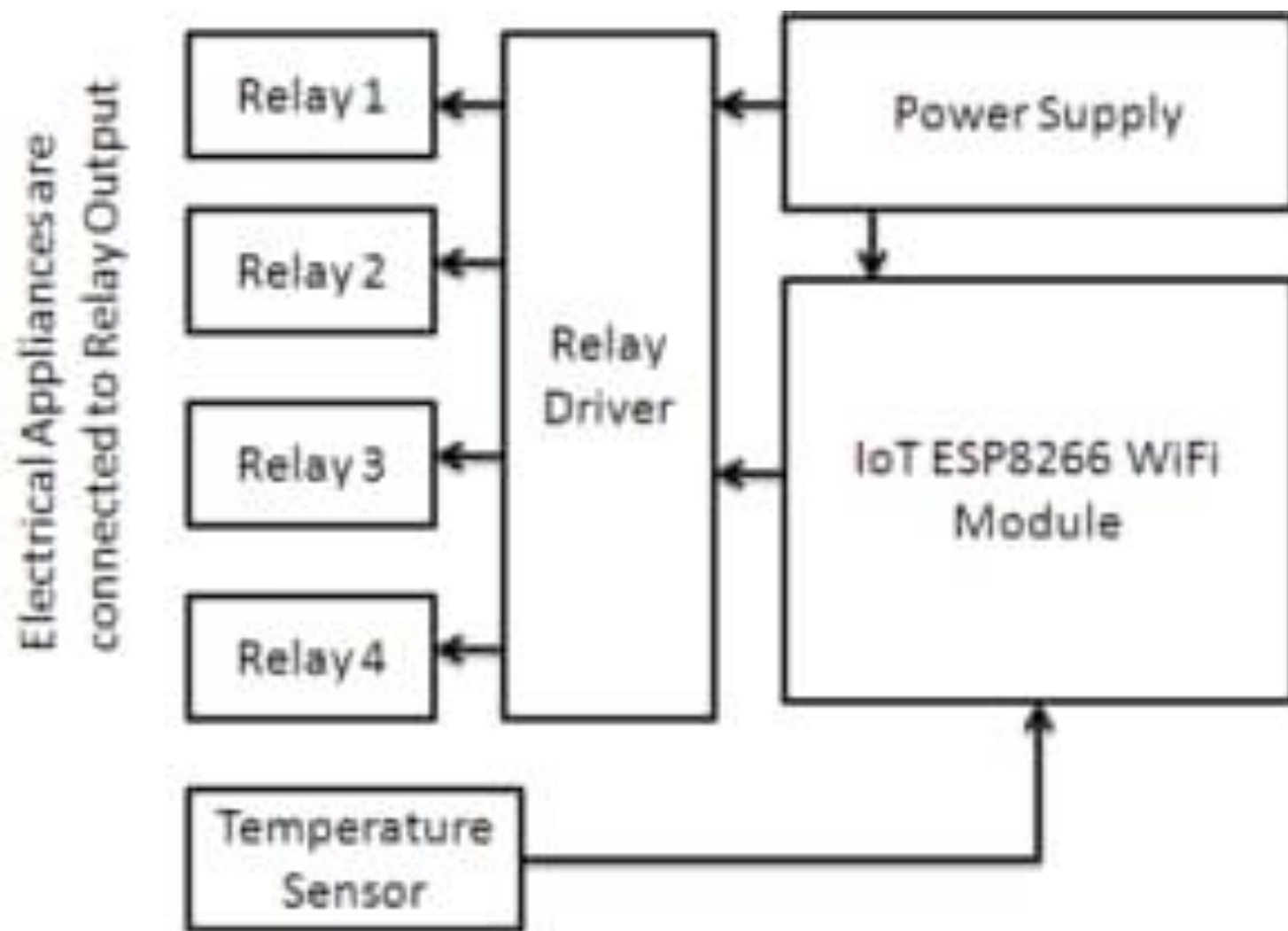
The **MQ-135 Gas Sensors** are used in air quality control devices and are suitable for detecting or measuring of **NH<sub>3</sub>, NO<sub>x</sub>, Alcohol, Benzene, Smoke, CO<sub>2</sub>**. The MQ-135 sensor module comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. If you need to measure the gases in PPM, the analog pin need to be used. The analog pin is TTL driven and works on 5V and so can be used with most common microcontrollers. It can detect or measure common air quality gases such as CO<sub>2</sub>, Smoke, NH<sub>3</sub>, NO<sub>x</sub>, Alcohol, Benzene.

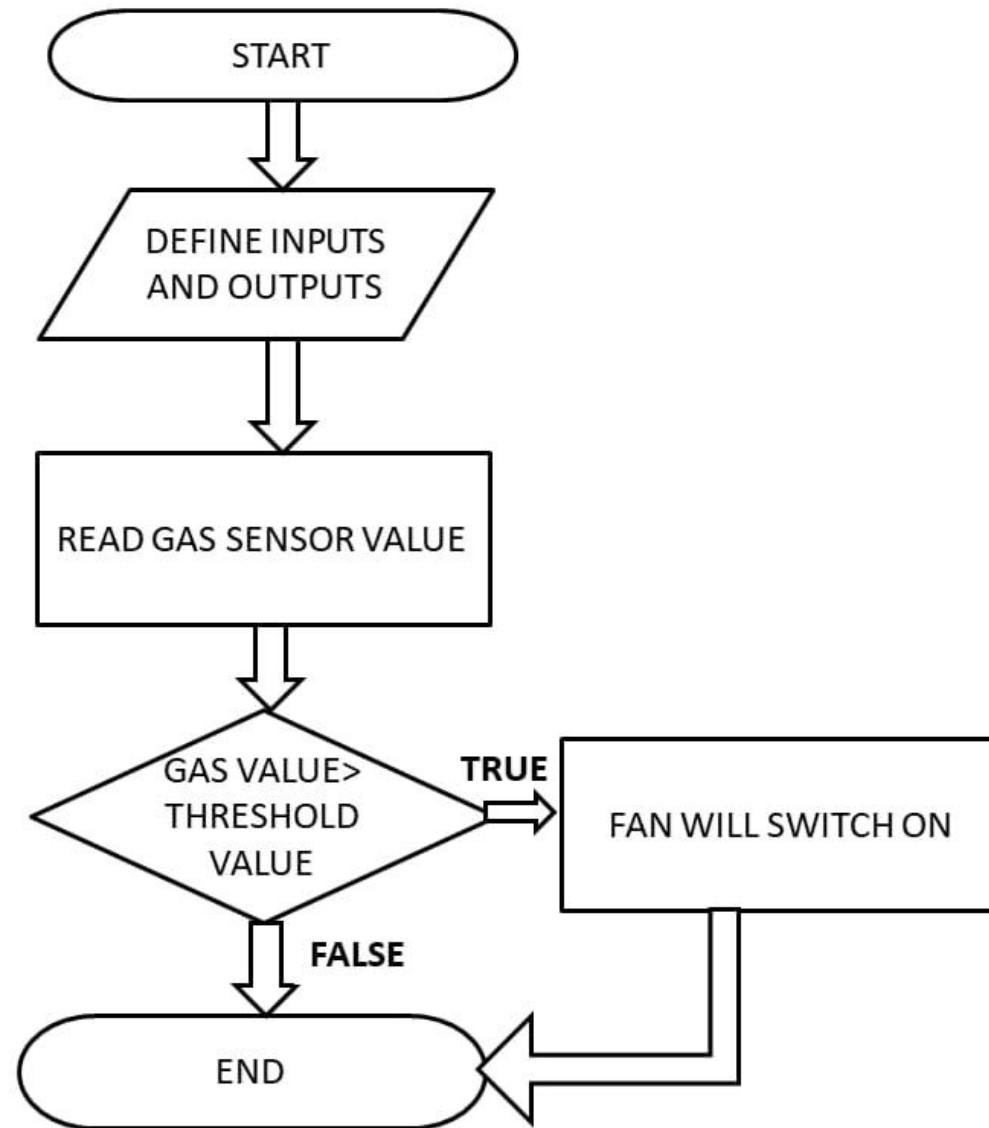


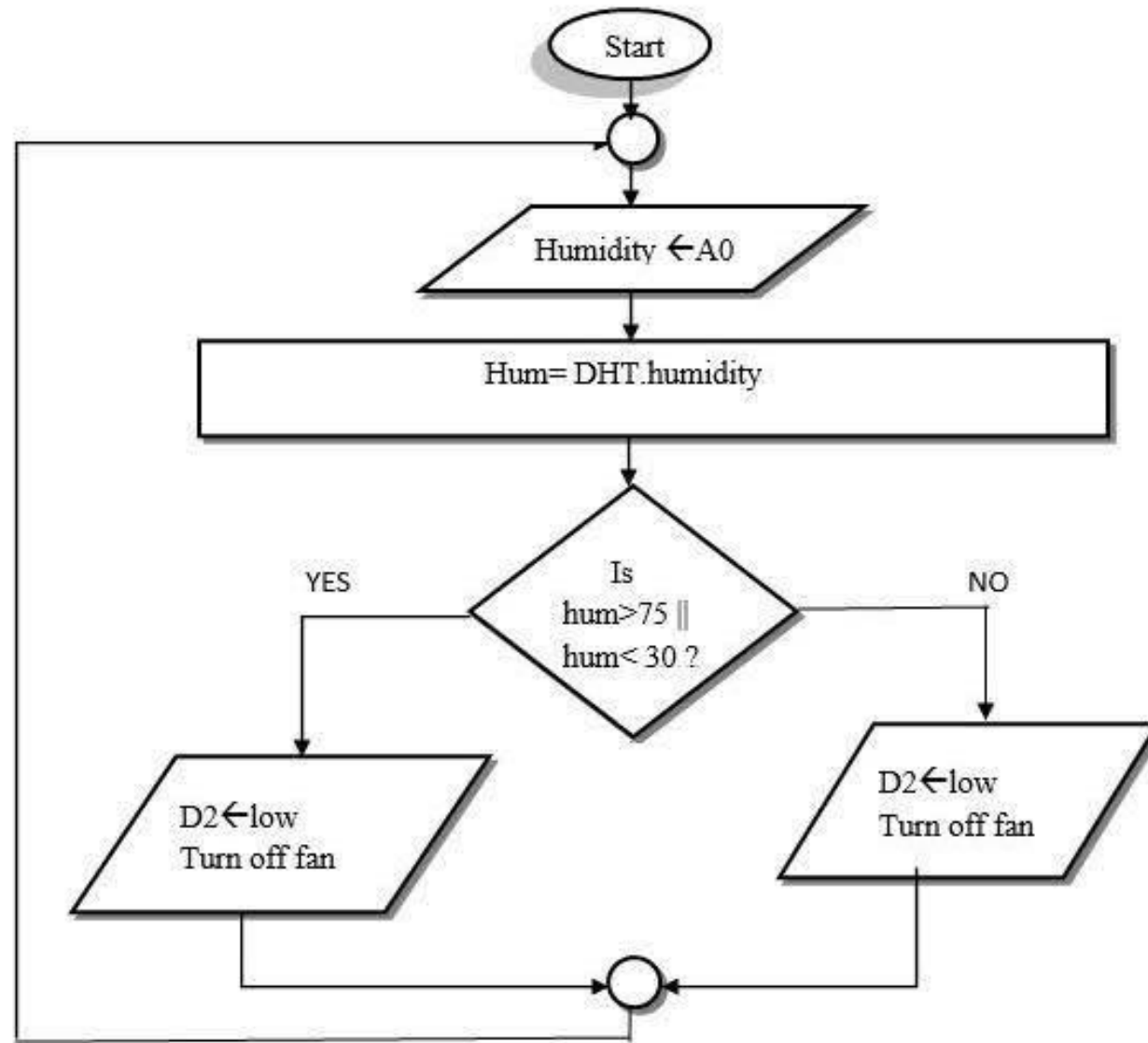
All objects, including the **human body**, at **temperatures above absolute zero** (0 Kelvin / -273.15 °C) emit heat energy in the form of infrared radiation. The hotter an object is, the more radiation it emits. This radiation is not visible to the human eye because it is emitted at infrared wavelengths. The **PIR SENSOR** is specifically designed to detect such levels of **infrared radiation**. A PIR sensor consists of two main parts:

1. A pyroelectric sensor, which you can see in the image below as a round metal with a rectangular crystal in the center.
2. A special lens called a **fresnel lens** which Focuses the infrared signals on the **pyroelectric sensor**.

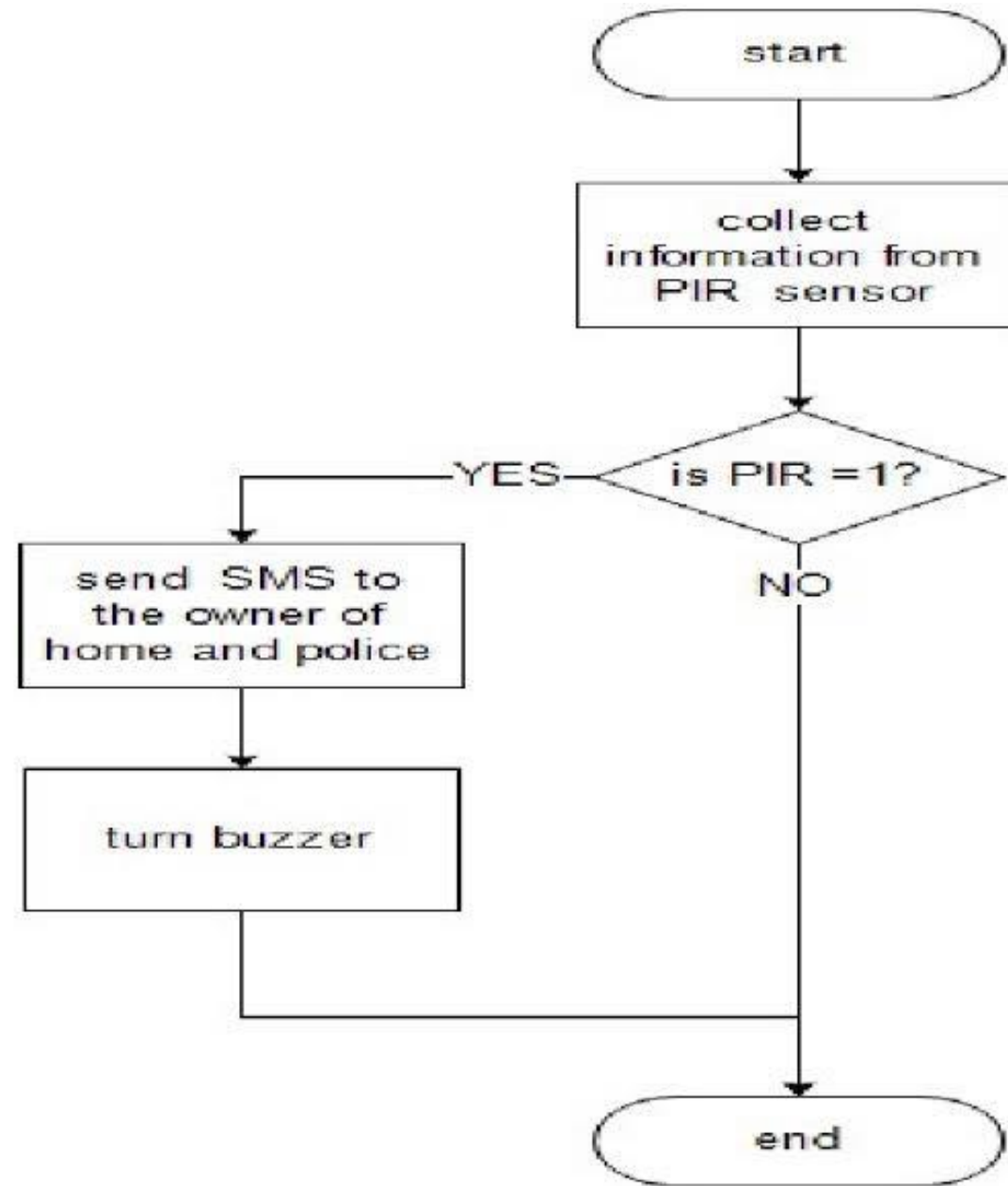












## Arduino IoT Cloud Variables description

The following variables are automatically generated and updated when changes are made to the Thing

```
String message;  
int gas_data;  
int pir;  
CloudRelativeHumidity humidity;  
CloudTemperature temperature;
```

```
#include "thingProperties.h"  
#include "DHT.h"  
#define DHTpin 2          // D4 on the nodemcu ESP8266  
#define DHTTYPE DHT11  
DHT dht(DHTpin, DHTTYPE);  
int gas_sensor=A0;  
int pir_sensor=D7;
```

```
void setup() {  
  pinMode(D0,OUTPUT);  
  // Initialize serial and wait for port to open:  
  Serial.begin(9600);  
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found  
  delay(1500);  
  pinMode(gas_sensor,INPUT);  
  pinMode(pir,INPUT);  
  pinMode(BUZZER,OUTPUT);  
  pinMode(led,OUTPUT);  
  // Defined in thingProperties.h  
  initProperties();  
  
  // Connect to Arduino IoT Cloud  
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
  setDebugMessageLevel(2);  
  ArduinoCloud.printDebugInfo();  
}
```

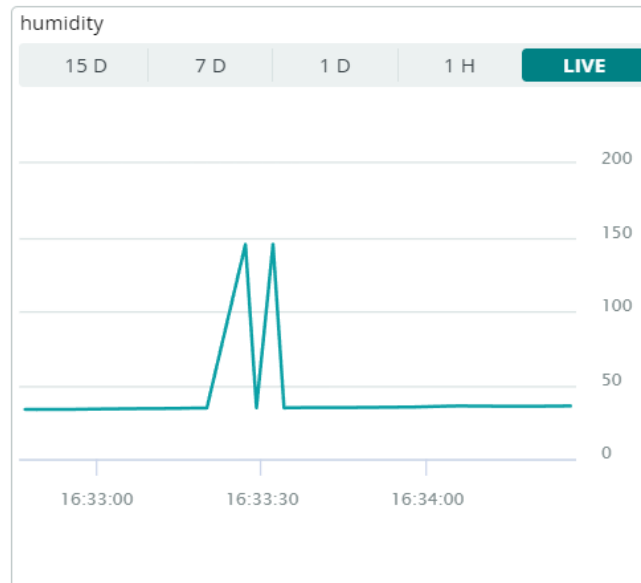
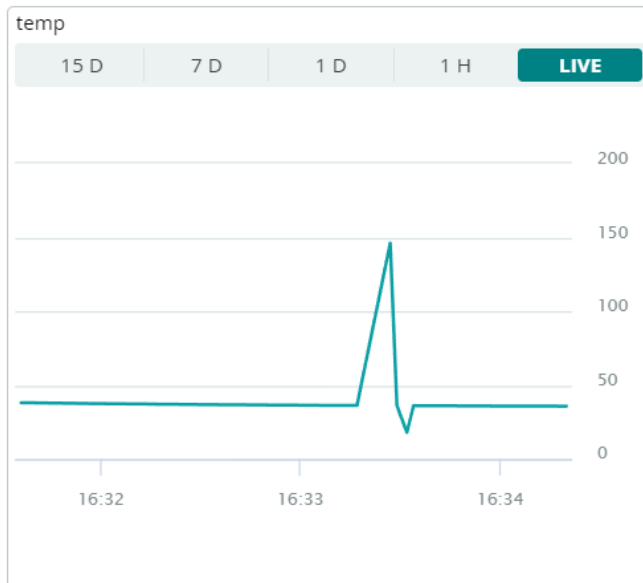
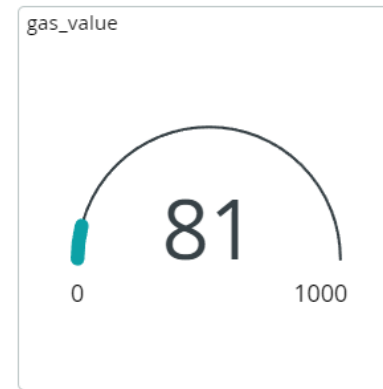
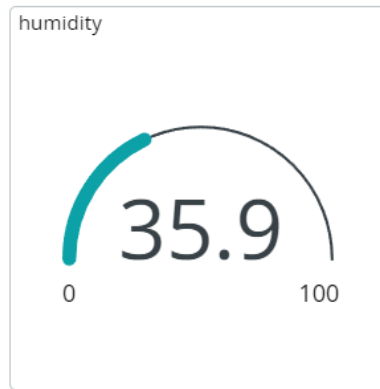
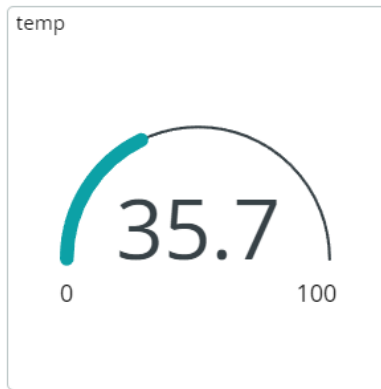
```
void loop() {  
    ArduinoCloud.update();  
    float hm=dht.readHumidity();  
    Serial.print("Humidity ");  
    Serial.println(hm);  
    float temp=dht.readTemperature();  
    Serial.print("Temperature ");  
    Serial.println(temp);  
    humidity=hm;  
    temperature=temp;  
    message="Temperature=" + String(temperature)+ "Humidity"+ String(humidity)+ "gas-value="+ String(gas_data);  
    gas_data=analogRead(gas_sensor);  
    pir=analogRead(pir_sensor);  
    Serial.print("gas_value");  
    Serial.println(gas_data);  
}
```

```
void onHumidityChange() {  
    if(humidity>70)  
    {  
        digitalWrite(D0,HIGH);  
    }  
    else  
    {  
        digitalWrite(D0,LOW);  
    }  
}
```

```
void onGasDataChange() {  
    // Add your code here to act upon GasData change  
    if(gas_data>100)  
    { digitalWrite(D0,HIGH);  
      digitalWrite(BUZZER,HIGH);  
      message="gas detected" + String(gas_data);  
    }  
    else  
    { digitalWrite(D0,LOW);  
      digitalWrite(BUZZER,LOW);  
      message="gas not detected" + String(gas_data);  
    }  
}
```

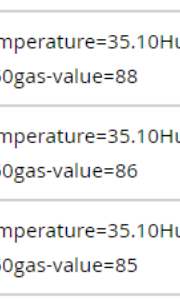
```
void onPirChange() {  
    {  
        digitalWrite(pir,HIGH);  
    }  
    else  
    {  
        digitalWrite(pir,LOW);  
    }  
}
```

kitchen



Timeline showing the sequence of events:

- 16:34: Call
- 16:36: Call
- 16:38: Call



Temperature=35.10 Humidity=3.50 gas-value=86 16:38

Temperature=35.10 Humidity=3.65 gas-value=88 16:38

Temperature=35.10 Humidity=3.65 gas-value=86 16:38

Temperature=35.10 Humidity=3.65 gas-value=85 16:38

Temperature=35.10 Humidity=3.64 gas-value=88 16:38

Temperature=35.10 Humidity=3.64 gas-value=89 16:38

Temperature=35.10 Humidity=3.64 gas-value=88 16:38

Type a message

[illegible]



