Date	10 November 2022
Team ID	PNT2022TMID20427
Project Title	Industry-Specific Intelligent Fire Management System

Display the temperature values:

Submitted by

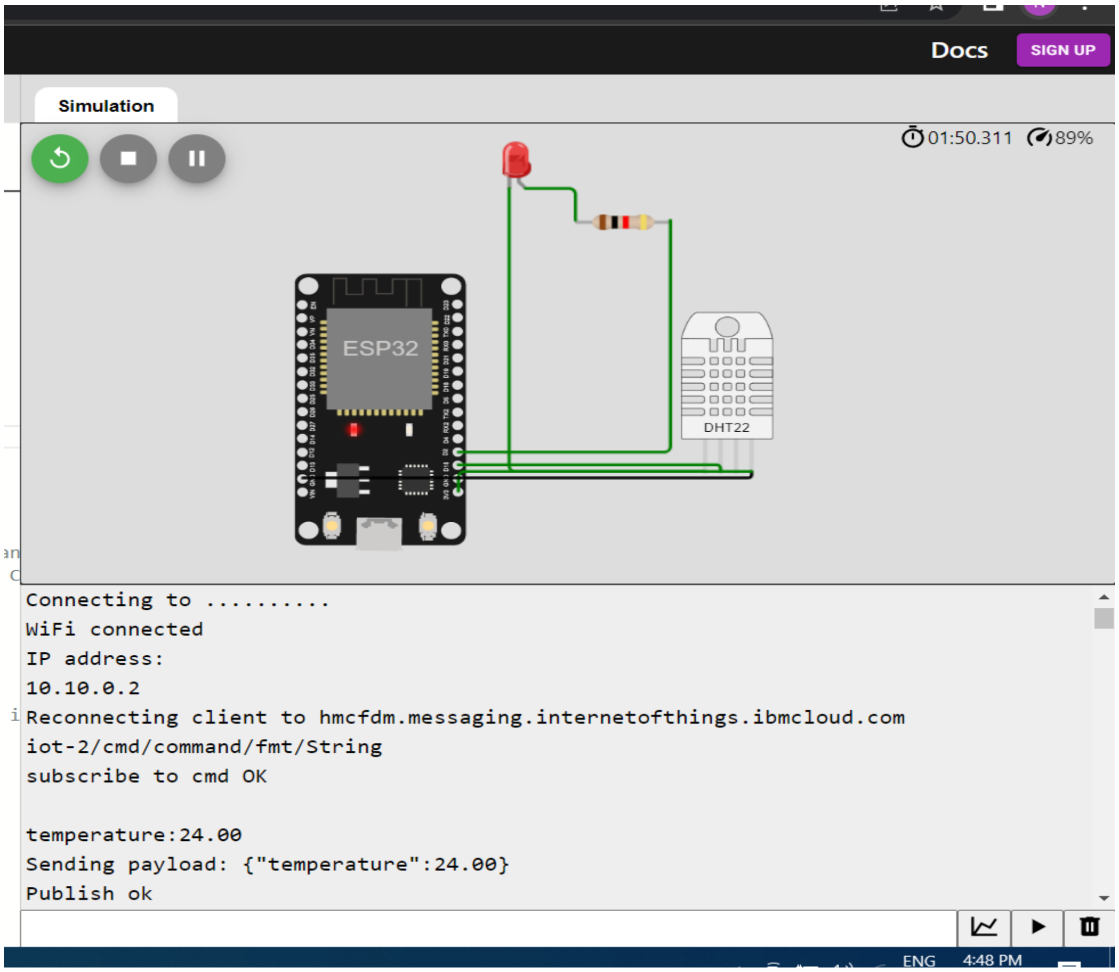
Ganesh Arravind B – 49621911063

Lokesh Durai V – 49621911027

Navenraj B M – 49621911026

Abiswetha S - 49621911002

Wokwi link: https://wokwi.com/projects/348683544624628306

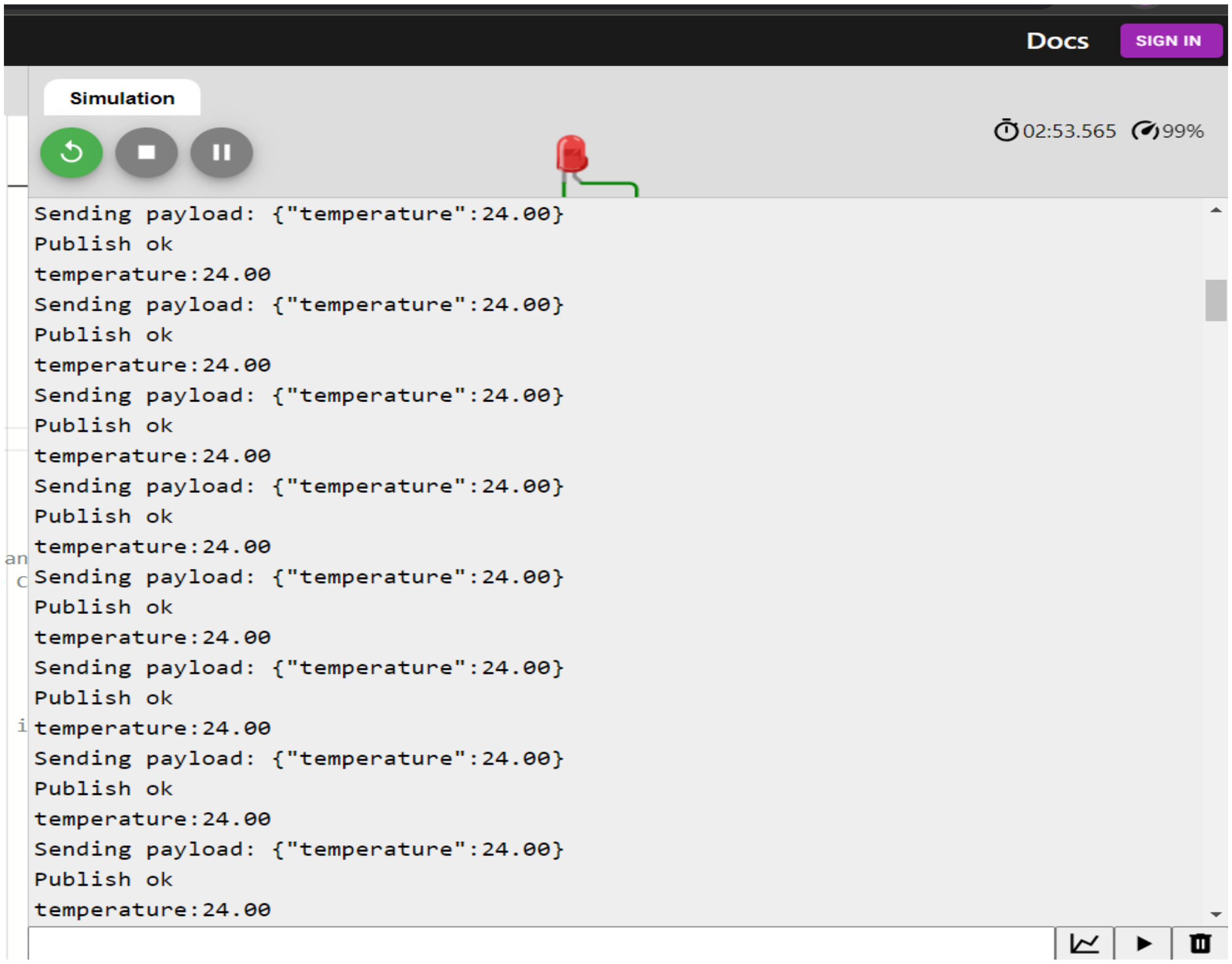


```
Connecting to ..........
WiFi connected
IP address:
10.10.0.2
Reconnecting client to hmcfdm.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
```

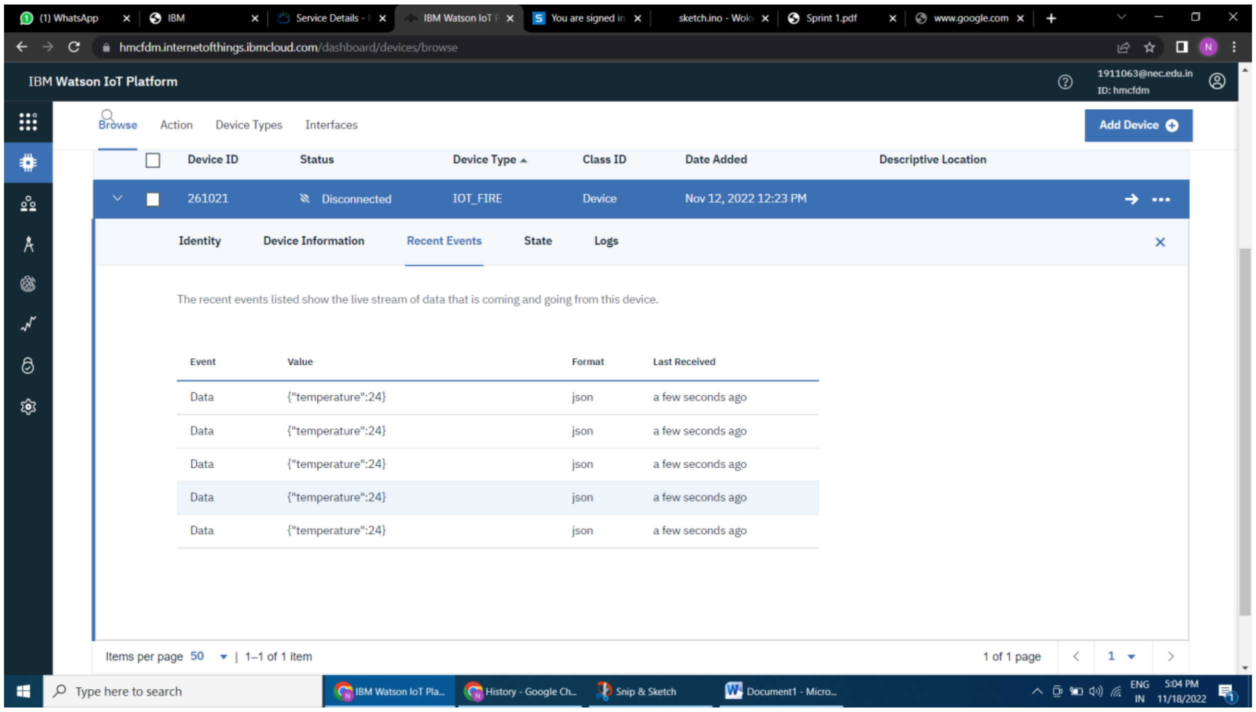
CODING:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 2
DHT dht (DHTPIN, DHTTYPE);
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-------credentials of IBM Accounts------
#define ORG "hmcfdm"//IBM ORGANITION ID
#define DEVICE_TYPE "IOT_FIRE"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "261021"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1911063abcdefgh" //Token
String data3;
float t;
//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type ofevent
perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENTcommand
type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling thepredefined
client id by passing parameter like server id,portand wificredential
void setup()// configureing the ESP32
{
 Serial.begin(115200);
 dht.begin();
 pinMode(LED,OUTPUT);
 delay(10);
 Serial.println();
 wificonnect();
 mqttconnect();
}
void loop()// Recursive Function
{
 t = dht.readTemperature();
 Serial.print("temperature:");
 Serial.println(t);
 PublishData(t);
 delay(1000);
 if (!client.loop()) {
 mqttconnect();
 }
}
/*....................................retrieving to
Cloud...............................*/
void PublishData(float temp) {
 mqttconnect();//function call for connecting to ibm
 /*
 creating the String in in form JSon to update the data to ibm cloud
 */
 String payload = "{\"temperature\":";
 payload += temp;
 payload += "}";
 Serial.print("Sending payload: ");
 Serial.println(payload);
 if (client.publish(publishTopic, (char*) payload.c_str())) {
 Serial.println("Publish ok");// if it sucessfully upload data on the cloud then
it will print publish ok in Serial monitor or else it will print publish failed
 } else {
 Serial.println("Publish failed");
 }
}
void mqttconnect() {
 if (!client.connected()) {
 Serial.print("Reconnecting client to ");
 Serial.println(server);
 while (!!!client.connect(clientId, authMethod, token)) {
 Serial.print(".");
 delay(500);
 }

 initManagedDevice();
 Serial.println();
 }
}
void wificonnect() //function defination for wificonnect
{
 Serial.println();
 Serial.print("Connecting to ");
 WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials toestablish the
connection
```

```
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void initManagedDevice() {
 if (client.subscribe(subscribetopic)) {
 Serial.println((subscribetopic));
 Serial.println("subscribe to cmd OK");
 } else {
 Serial.println("subscribe to cmd FAILED");
 }
}
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
 Serial.print("callback invoked for topic: ");
 Serial.println(subscribetopic);
 for (int i = 0; i < payloadLength; i++) {
 //Serial.print((char)payload[i]);
 data3 += (char)payload[i];
 }
 Serial.println("data: "+ data3);
 if(data3=="lighton")
 {
Serial.println(data3);
digitalWrite(LED,HIGH);
 }
 else
 {
Serial.println(data3);
digitalWrite(LED,LOW);
 }
data3="";
}
```



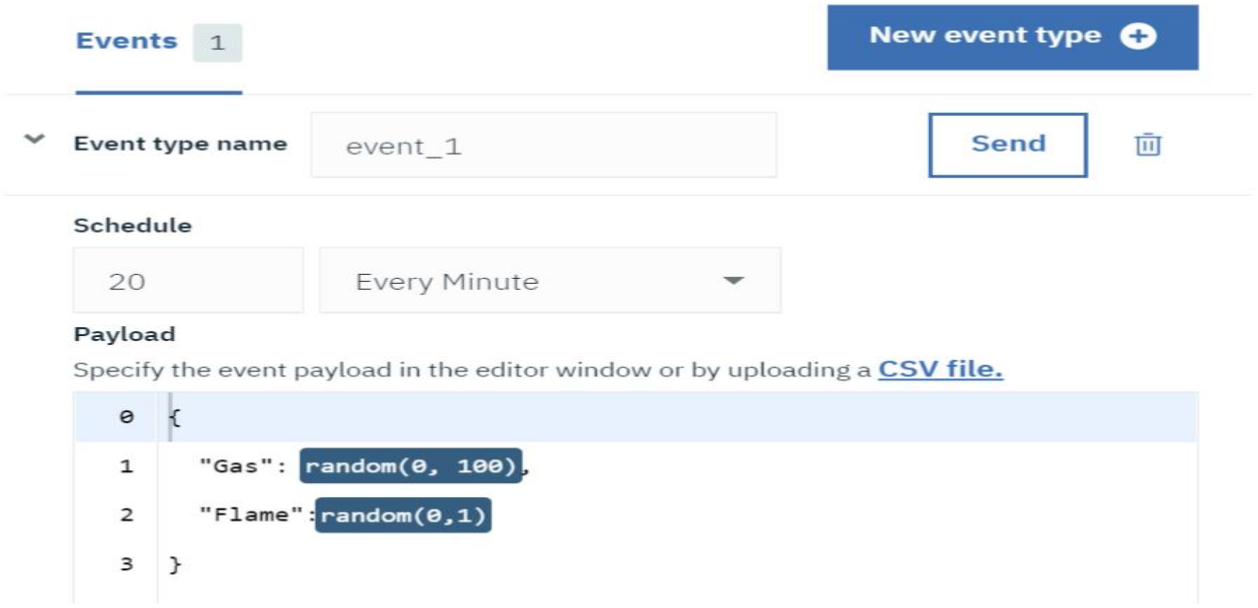
```
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
Sending payload: {"temperature":24.00}
Publish ok
temperature:24.00
```



Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
261021	Disconnected	IOT_FIRE	Device	Nov 12, 2022 12:23 PM	

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago



Event type name: event_1

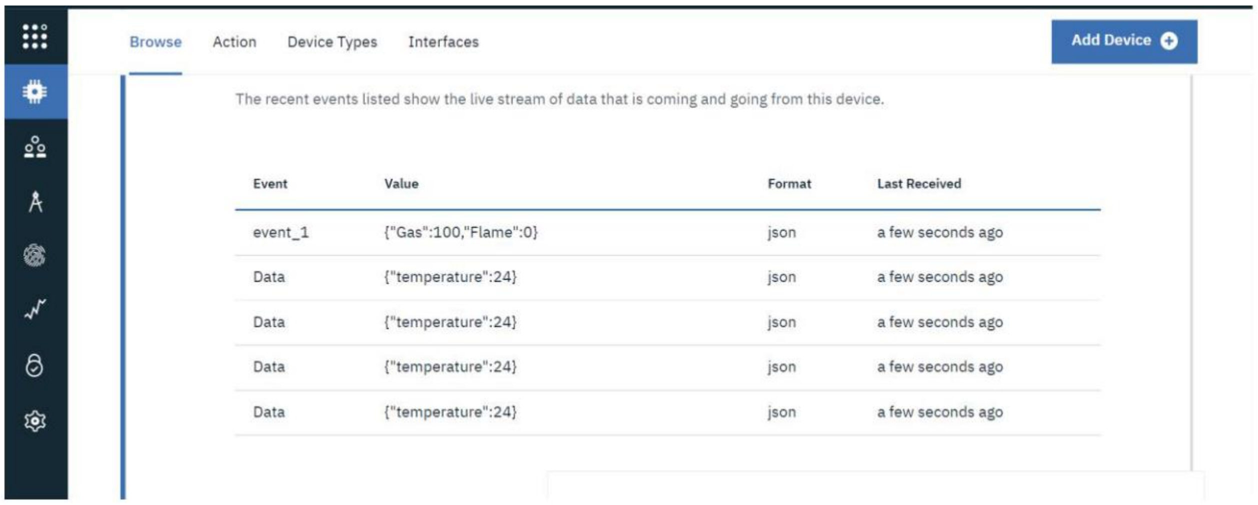
Schedule: 20 Every Minute

Payload

Specify the event payload in the editor window or by uploading a CSV file.

```
{
  "Gas": random(0, 100),
  "Flame":random(0,1)
}
```

Displaying gas sensor & flame sensor values:



The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event_1	{"Gas":100,"Flame":0}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago