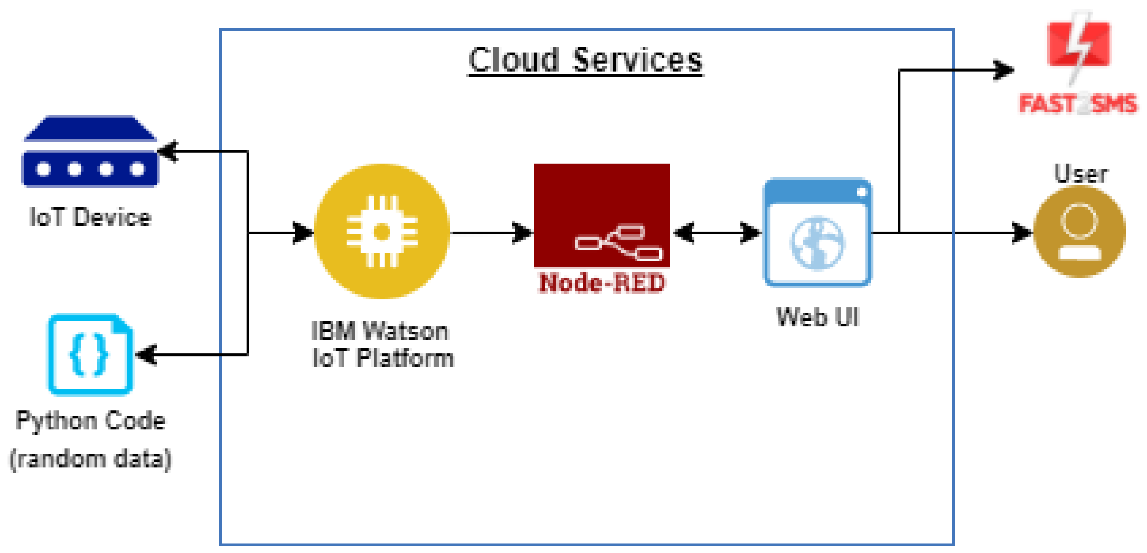
Project Development Phase
Delivery of Sprint - 4

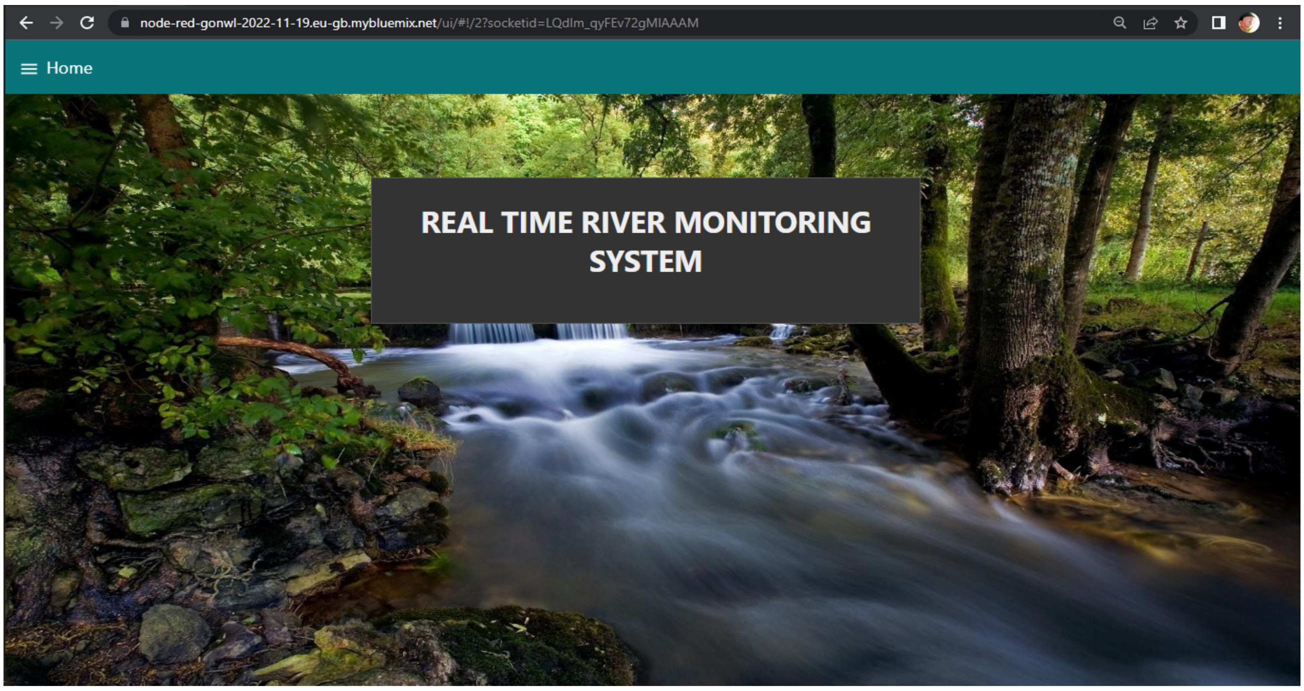
Date	13 November 2022
Team ID	PNT2022TMID20460
Project Name	IOT Based Real-Time River Water Quality Monitoring and Control System

Proposed Block Diagram:

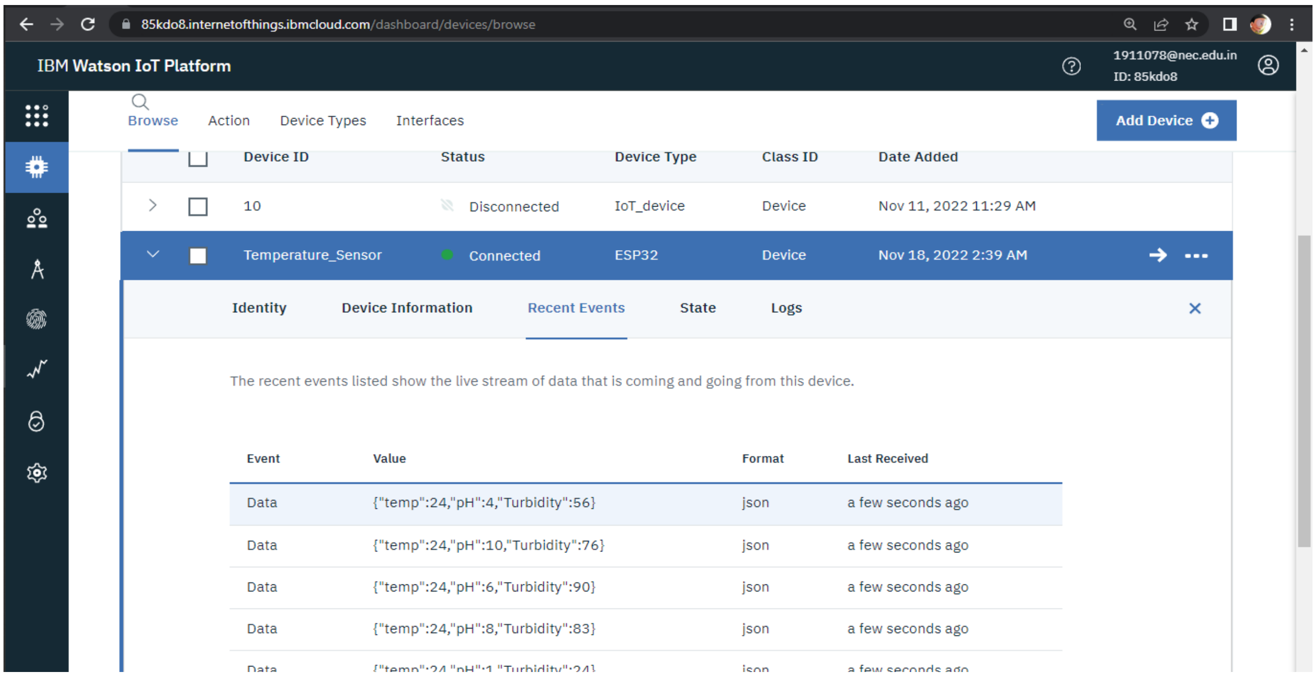


As per proposed diagram, the sensor data sent to **IBM Watson Platform**.The data from the IBM Watson Platform has been read by Node-RED and displayed it in a Web UI. Our Web UI consists of 2 parts: homepage and sensor data readings.

HOME PAGE:


node-red-gonwl-2022-11-19.eu-gb.mybluemix.net/ui/#!/2?socketid=LQdlm_qyFEv72gMIAAAM
Home
REAL TIME RIVER MONITORING SYSTEM

DATA PUBLISHED TO WATSON IOT PLATFORM

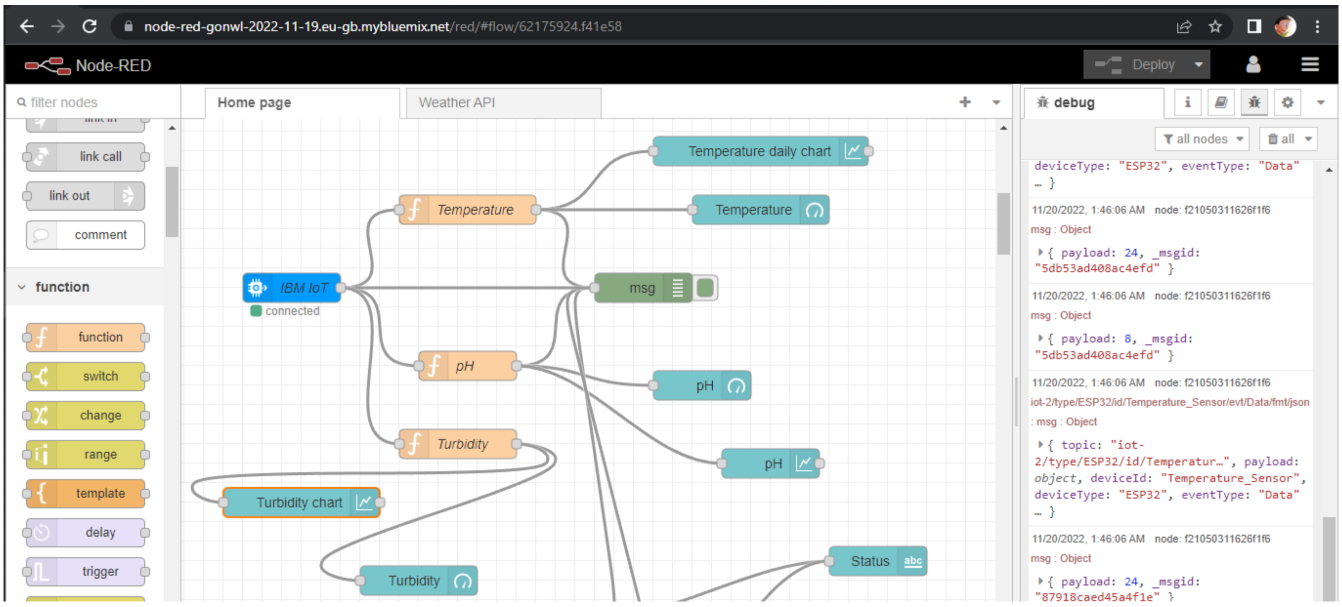


Device ID	Status	Device Type	Class ID	Date Added
10	Disconnected	IoT_device	Device	Nov 11, 2022 11:29 AM
Temperature_Sensor	Connected	ESP32	Device	Nov 18, 2022 2:39 AM

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

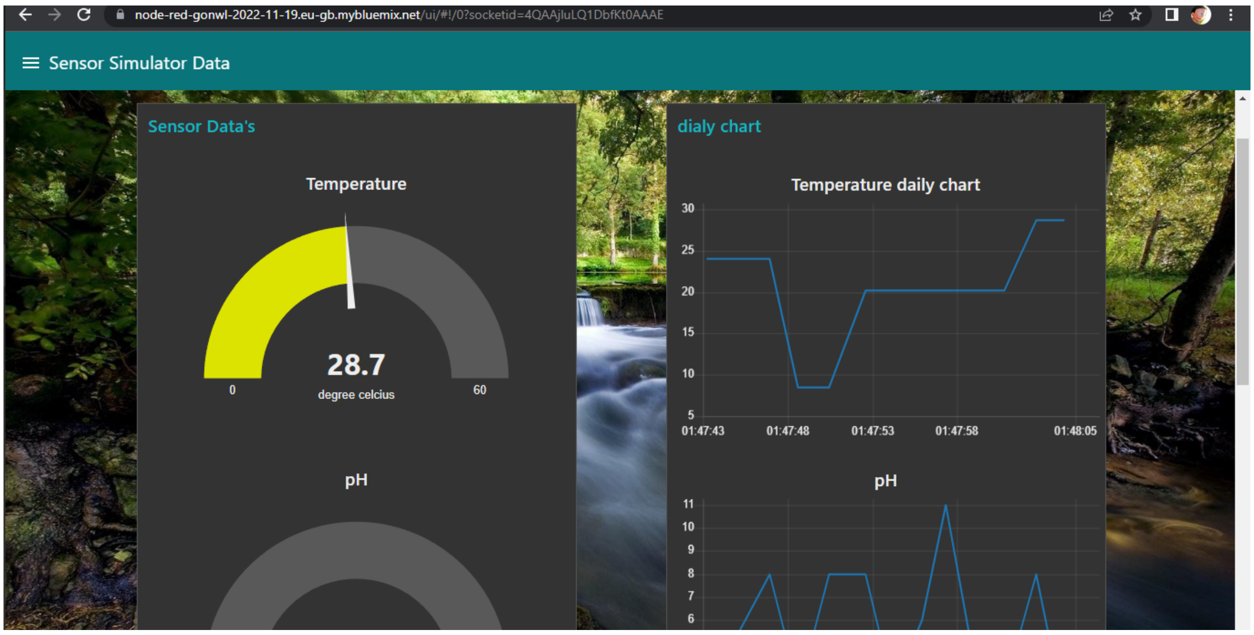
Event	Value	Format	Last Received
Data	{"temp":24,"pH":4,"Turbidity":56}	json	a few seconds ago
Data	{"temp":24,"pH":10,"Turbidity":76}	json	a few seconds ago
Data	{"temp":24,"pH":6,"Turbidity":90}	json	a few seconds ago
Data	{"temp":24,"pH":8,"Turbidity":83}	json	a few seconds ago
Data	{"temp":24,"pH":1,"Turbidity":24}	json	a few seconds ago

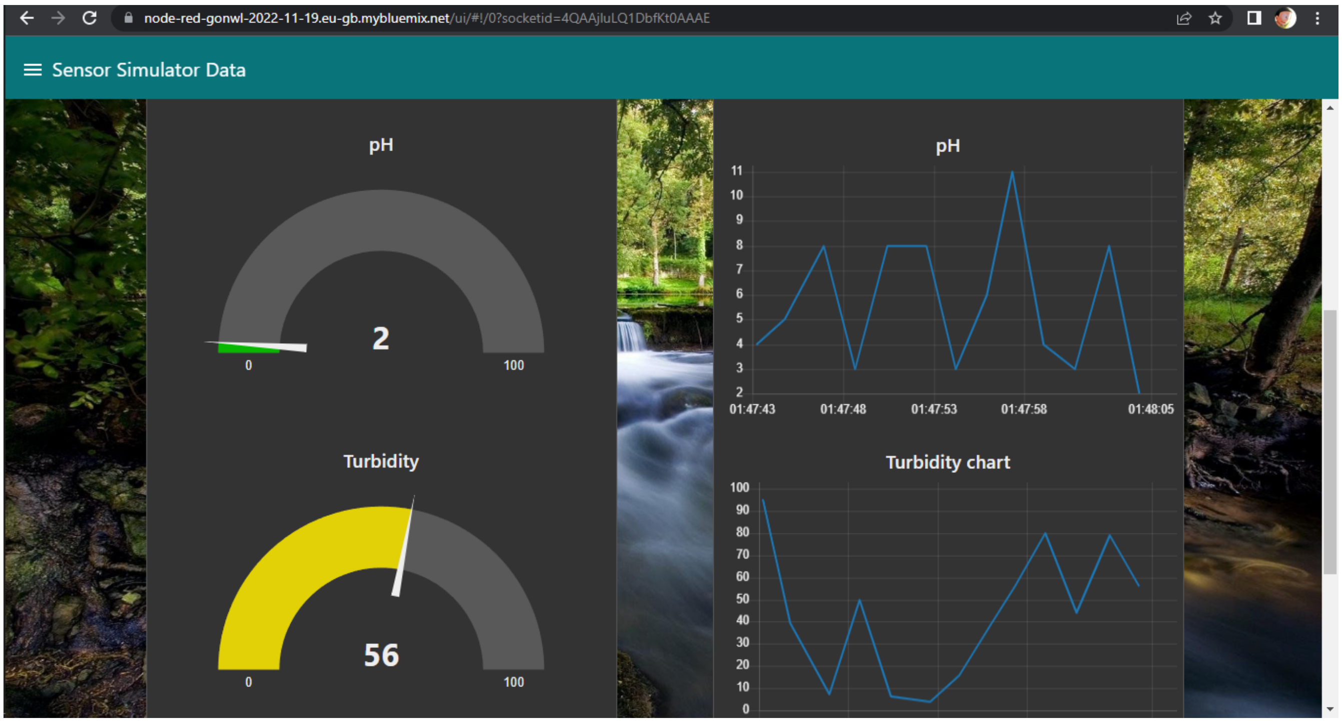
DATA PUBLISHED TO NODE RED FROM IBM WATSON IOT



```
deviceType: "ESP32", eventType: "Data"
… }
11/20/2022, 1:46:06 AM   node: f21050311626f1f6
msg : Object
▸{ payload: 24, _msgid:
"5db53ad408ac4efd" }
11/20/2022, 1:46:06 AM   node: f21050311626f1f6
msg : Object
▸{ payload: 8, _msgid:
"5db53ad408ac4efd" }
11/20/2022, 1:46:06 AM   node: f21050311626f1f6
iot-2/type/ESP32/id/Temperature_Sensor/evt/Data/fmt/json
: msg : Object
▸{ topic: "iot-
2/type/ESP32/id/Temperatur…", payload:
object, deviceId: "Temperature_Sensor",
deviceType: "ESP32", eventType: "Data"
… }
11/20/2022, 1:46:06 AM   node: f21050311626f1f6
msg : Object
▸{ payload: 24, _msgid:
"87918caed45a4f1e" }
```

DATA VIEWED IN WEB UI USING NODE RED:





CODE FOR RANDOM VALUE GENERATOR:

```
#include <stdio.h>
#include <stdlib.h>
#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);// creating the instance by passing pin and typr of
dht connected

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "85kdo8"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "Temperature_Sensor"//Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "1911078abcdefgh"      //Token
String data3;
```

```
float t;
int pH;
int turb;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
command 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 the
predefined 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
{

  pH = random(0,14);
  turb = random(0,100);
  t = dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("pH:");
  Serial.println(pH);
  Serial.print("Turbidity:");
  Serial.println(turb);
  PublishData(t, pH, turb);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



/*.....................................retrieving to
Cloud...............................*/

void PublishData(float temp, int pH, int turb) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"temp\":";
  payload += temp;
  payload += "," "\"pH\":";
  payload += pH;
  payload += "," "\"Turbidity\":";
  payload += turb;
  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 to establish
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="";
}
```