Real-Time River Water Quality Monitoring And Control System

TEAM ID : PNT2022TMID50683

TEAM MEMBERS:

S.PRIYA THARSHINI(TL)

F.ANTONY LOORTHU SARA

P.BENISHTTA

M.MULLAIKODI

INTRODUCTION:

•River water quality can be monitored by the web application.

• We can be able to know if there are any dust particles present in the water.

•The PH level of the water can be monitored.

•Water temperature can be monitored.

•Alerting the authorities if the water quality is not good so that they can go and announce the localities not to drink that water.

SOFTWARE REQUIRED:

- IBM CLOUD SERVICES
    1. IBM Watson IoT Platform
    2. Node-RED Service
    3. Cloudant DB
- WOKWI
- MIT APP INVENTOR(MOBILE APP)

Objective:

•Sending random pH values and temperature values will be sent to the IBM IoT platform.

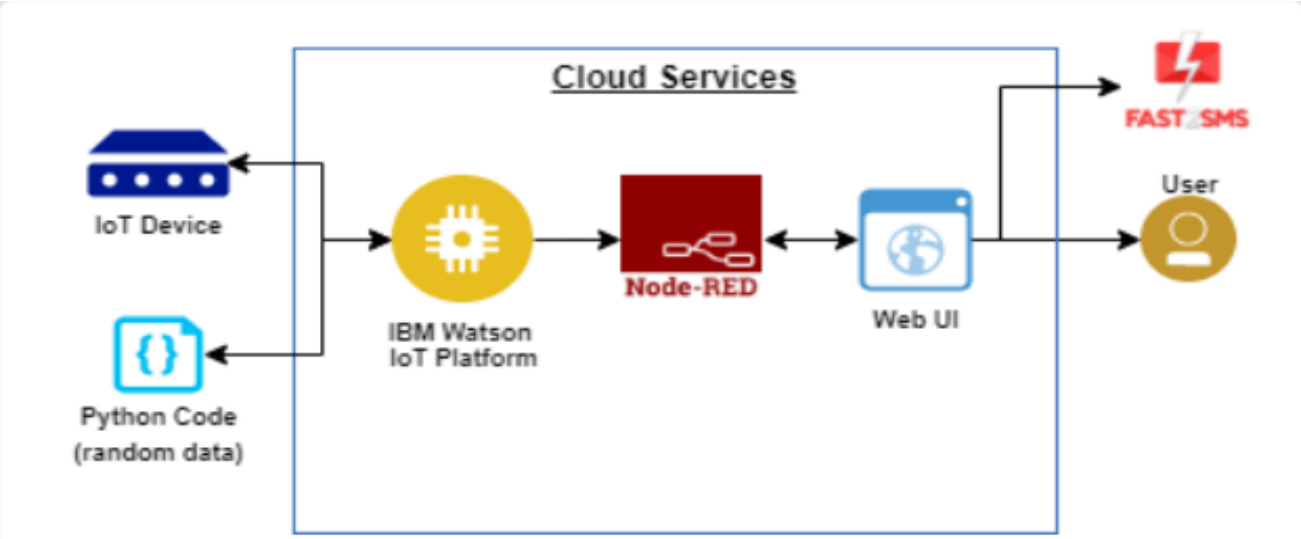
•Sensors values can be viewed in the Web Application.

•Notifies the admin the random values cross the threshold value.

•To accomplish this, we have to complete all the activities:

- •Create and configure IBM Cloud Services.
- •Create IBM Watson IoT Platform.
- •Create a device & configure the IBM IoT Platform.
- •Create Node-RED service.
- •Create a database in Cloudant DB to store location data.
- •Develop a web Application using Node-RED Service.

Technical Architecture:



Advantages:

•Monitoring is necessary to ensure that our waters can continue to support the many different ways we use these resources and to track whether protection and restoration measures are working.

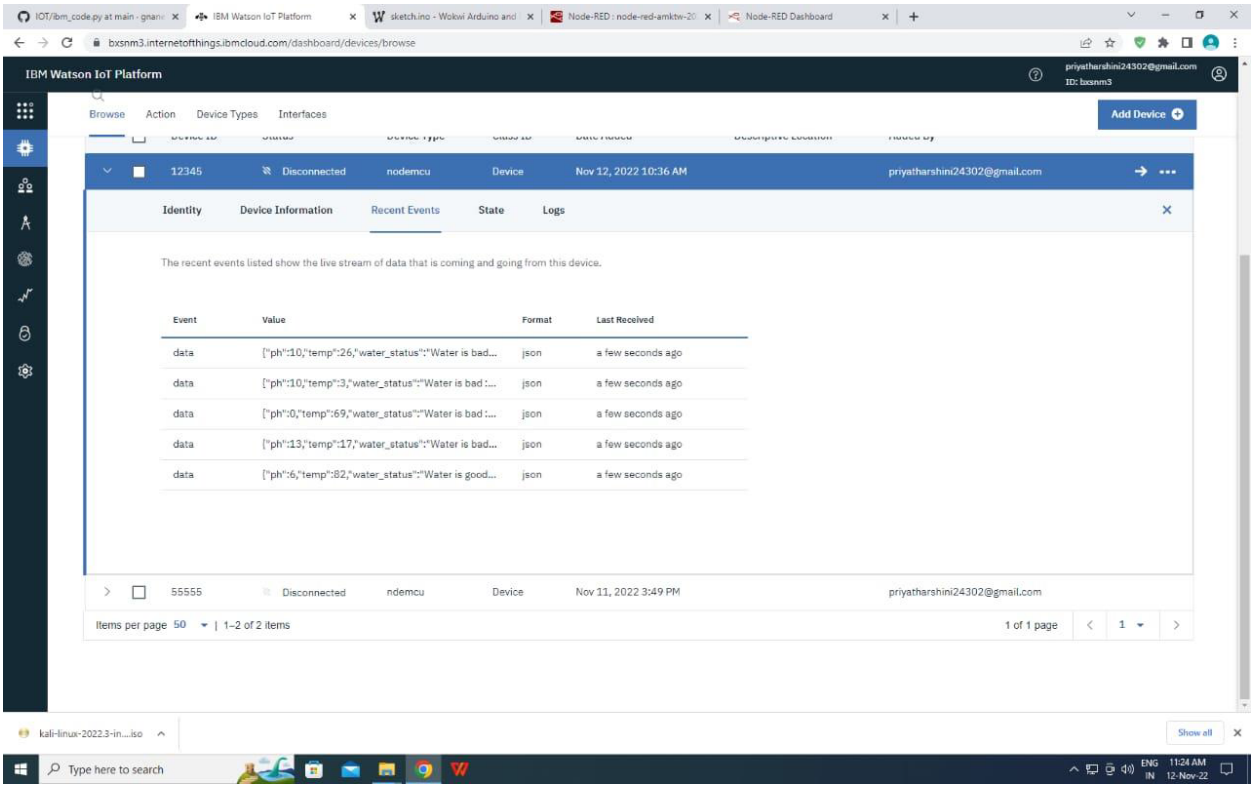
•This sleek and easily portable water quality meter can give you accurate measurements on pH, and the temperature of your water.

WATSON IOT PLATFORM:

•Launch the iot platform in Cloud service.

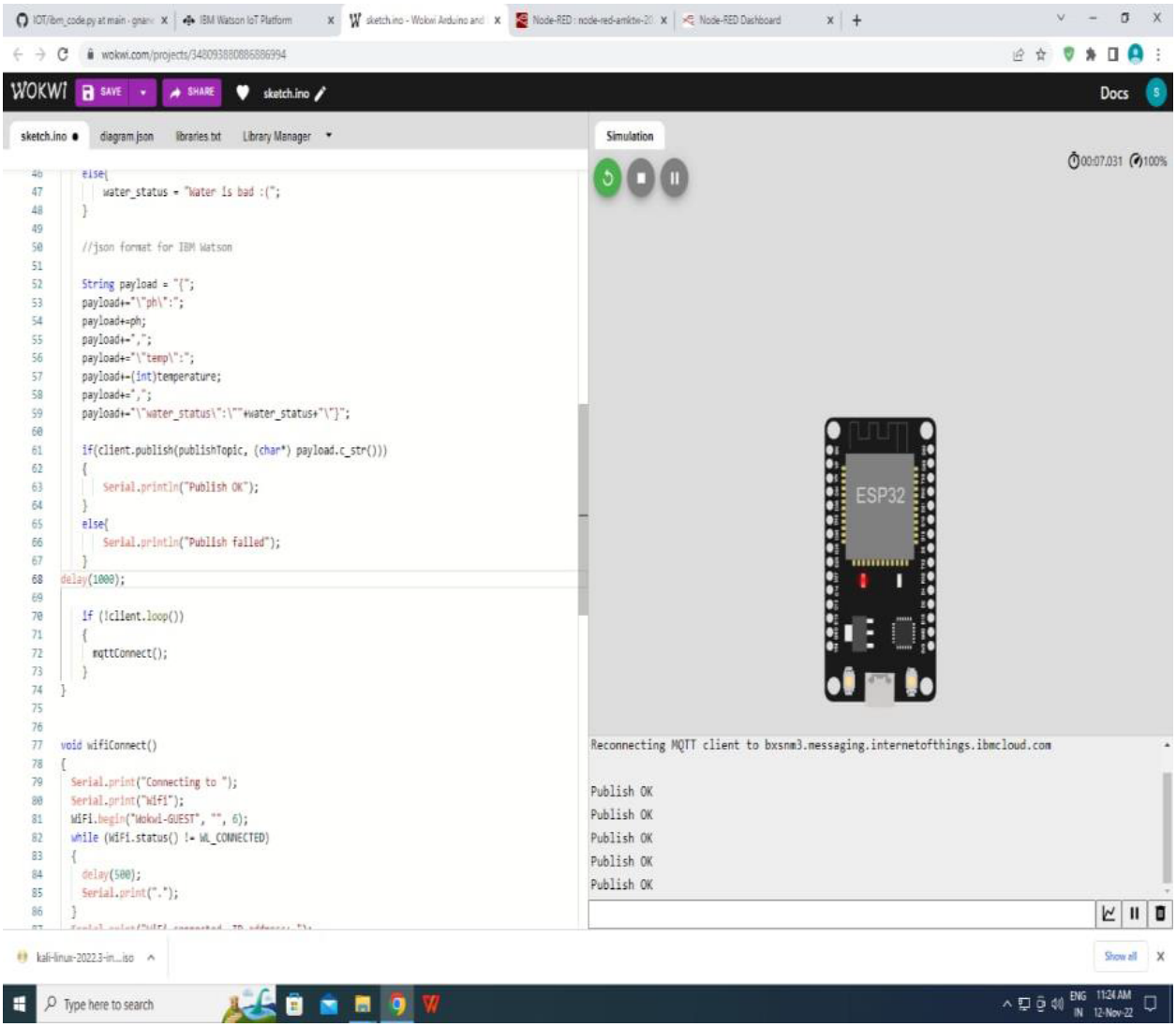
•Add devices in the iot platform.

•Copy the device credentials like device id, token, key .. to connect with wokwi wifi and node red .



WOKWI SOFTWARE:

- In the wokwi software, take ESP32.
- Write a code to connect that ESP32 with the watson IOT platform devices.
- Give random values of ph and temperature range.



CODE :
Used in wokwi to connect devices:

```
#include<time.h>
#include<WiFi.h>
#include<PubSubClient.h>

#define ORG "bxsnm3"
#define DEVICE_TYPE "nodemcu"
#define DEVICE_ID "12345"
#define TOKEN "CvF9tldLEy0-)U&0B0"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);

float temperature  = 0;
int ph = 0;
String water_status="";


void setup() {
  Serial.begin(99900);
   wifiConnect();
   mqttConnect();
}

void loop() {

  srand(time(0));

    //initial variables and random generated data
```

```
    temperature = random(0,100);
    ph = random(0,14);

    //set a flame status


    if(ph <= 8 && ph >= 6){
        water_status = "Water is good!!";

    }
    else{
        water_status = "Water is bad :(";
    }

    //json format for IBM Watson

    String payload = "{";
    payload+="\"ph\":";
    payload+=ph;
    payload+=",";
    payload+="\"temp\":";
    payload+=(int)temperature;
    payload+=",";
    payload+="\"water_status\":\""+water_status+"\"}";

    if(client.publish(publishTopic, (char*) payload.c_str()))
    {
        Serial.println("Publish OK");
    }
    else{
        Serial.println("Publish failed");
    }
delay(100);

    if (!client.loop())
    {
      mqttConnect();
    }
```

```
}

void wifiConnect()
{
  Serial.print("Connecting to ");
  Serial.print("Wifi");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());

}

void mqttConnect()
{
  if (!client.connected())
  {
    Serial.print("Reconnecting MQTT client to ");
    Serial.println(server);
    while (!client.connect(clientId, authMethod, token))
    {
      Serial.print(".");
      delay(500);
    }

    Serial.println();
  }
}
```

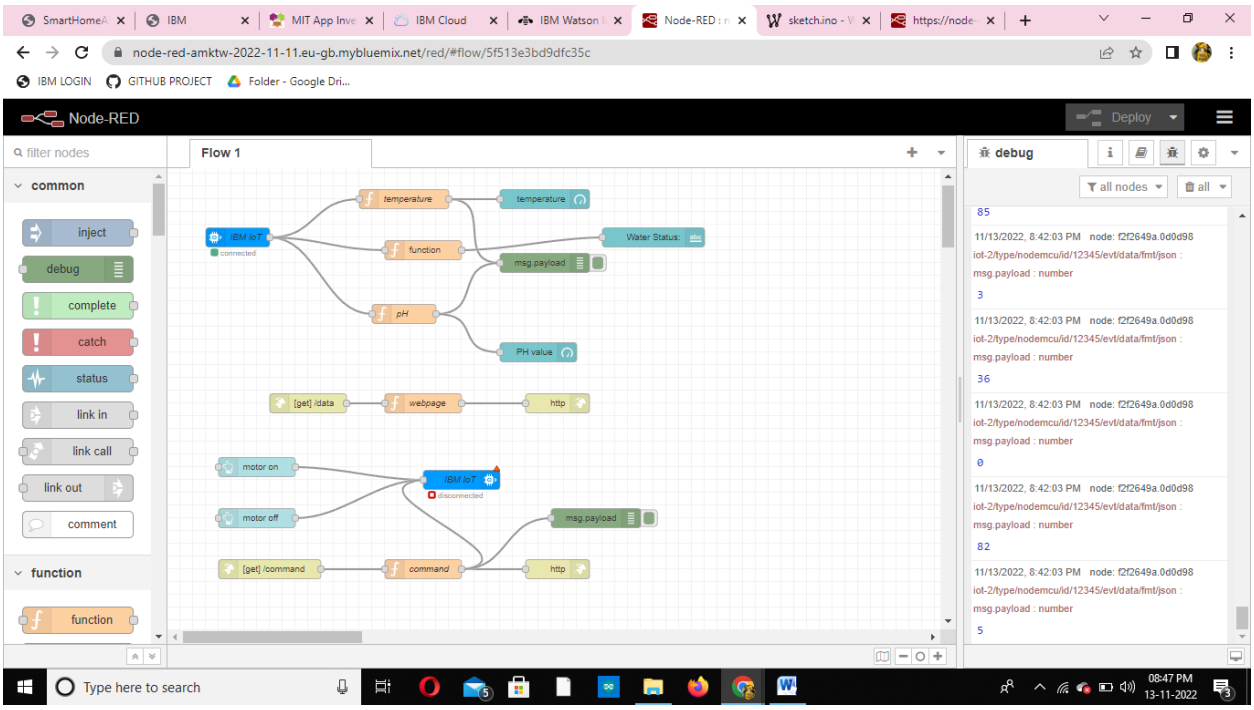
NODE RED:

• Configure the Node-RED flow to receive data from the IBM IoT platform.

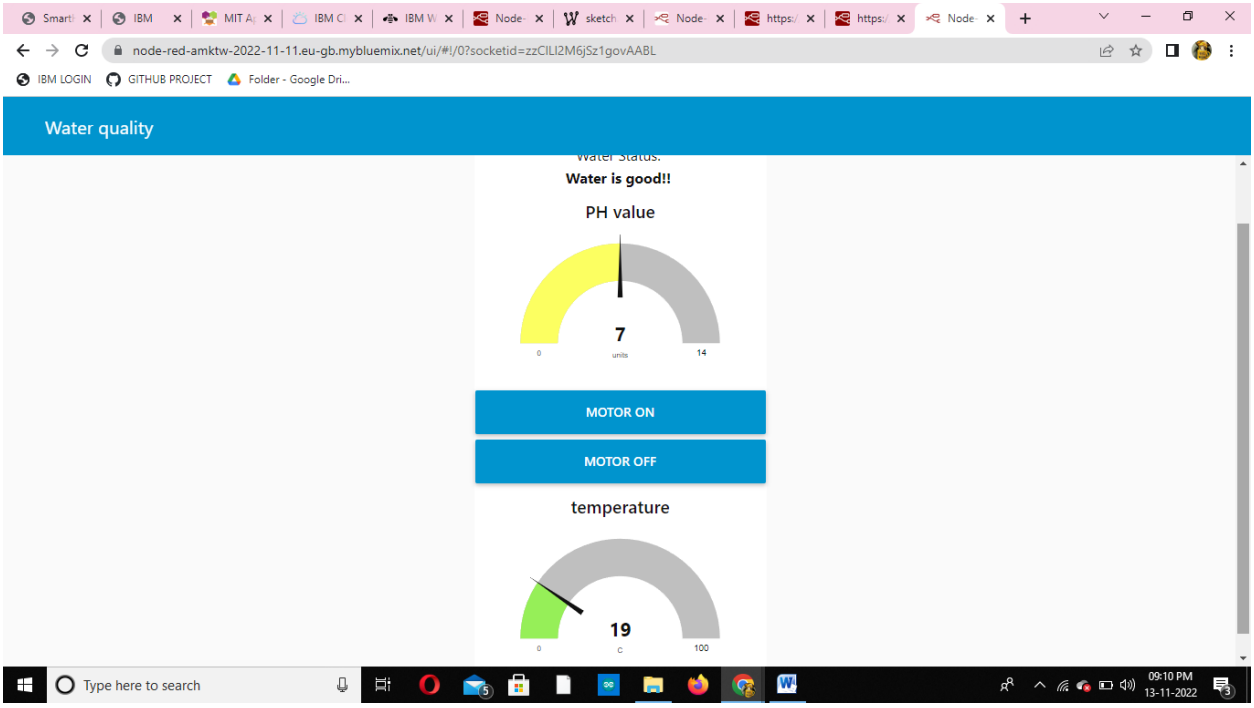
• And also use Cloudant DB nodes to store the received sensor data in the cloudant DB.

• Visualize the data in graphical format.

• Create an HTTP API for communicating with Mobile applications.



Check the controls in IBM IoT platform in recent events by connecting the devices:

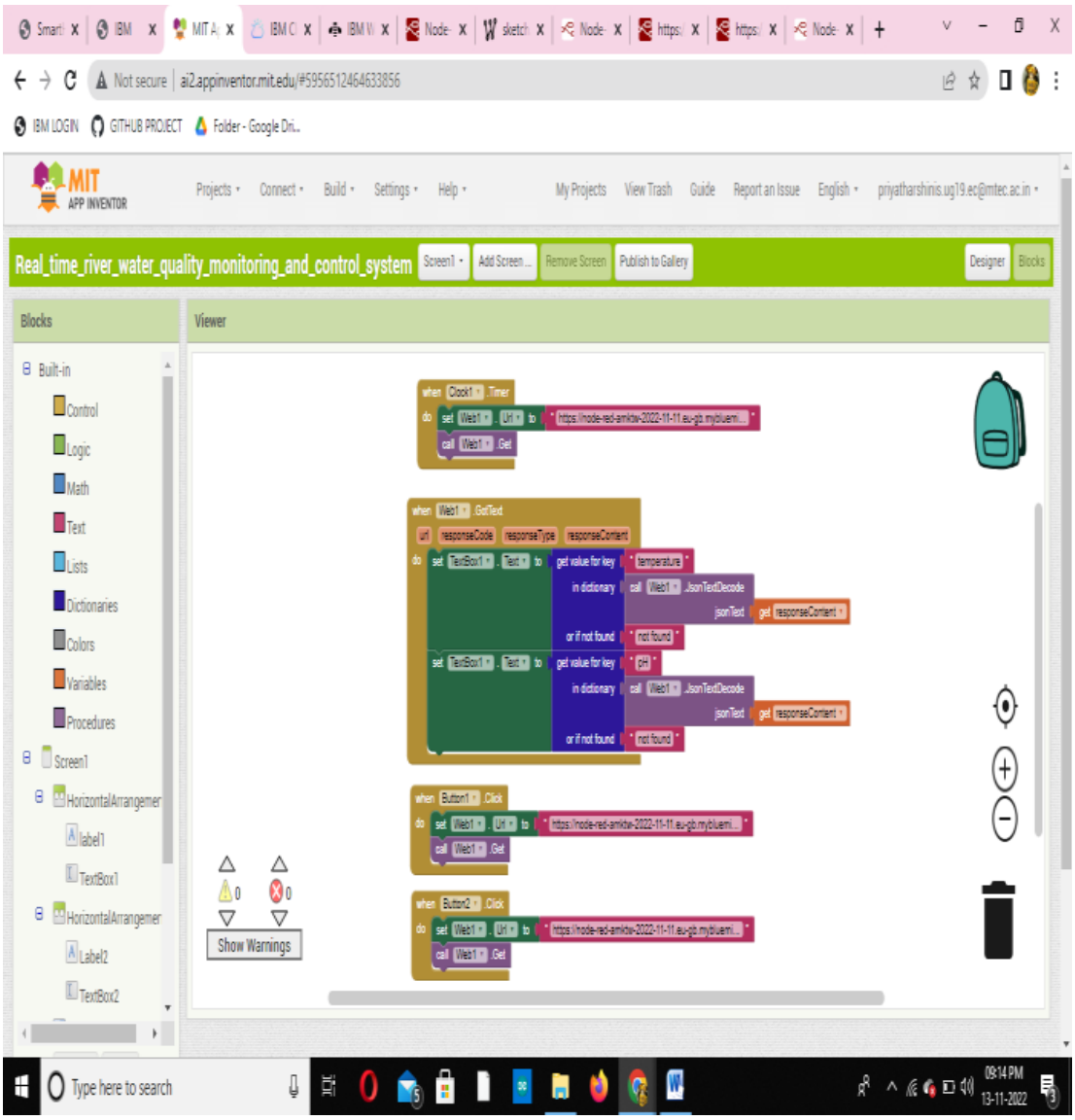


BUILDING MOBILE APP:

• Designing UI to display the Water temperature, and pH values sensor values.

•Configure the application to receive the data from the cloud.

•Configure the mobile app(MIT APP) to send commands to users using buttons.



MESSAGE TO USER:

The temperature and ph value of water is displayed in the mobile app of user .