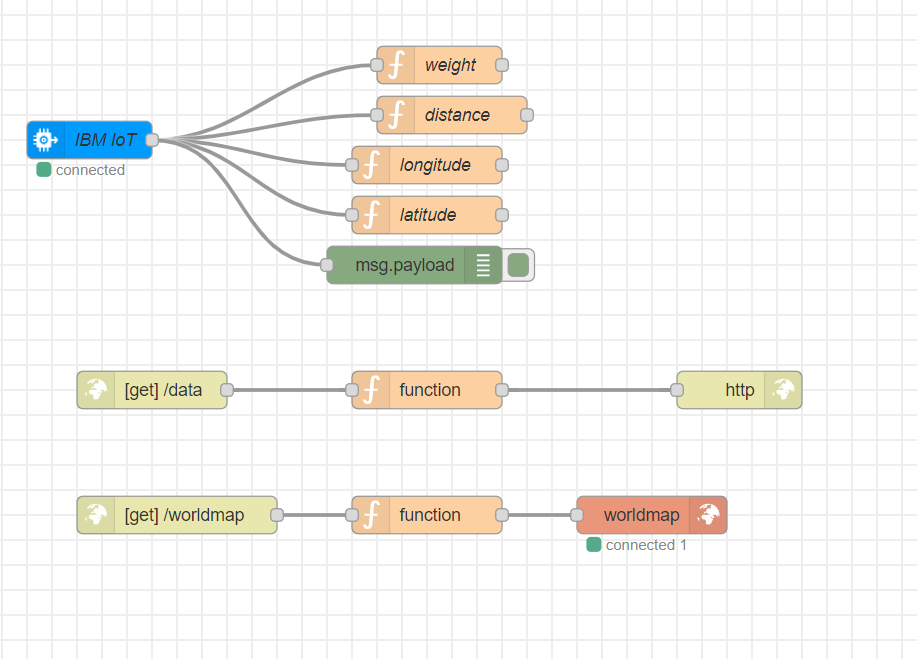
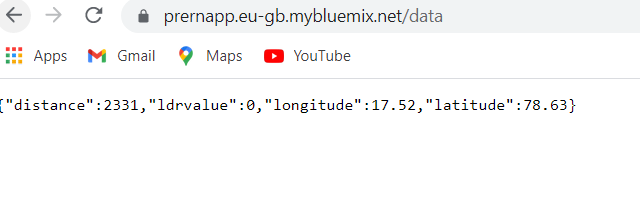
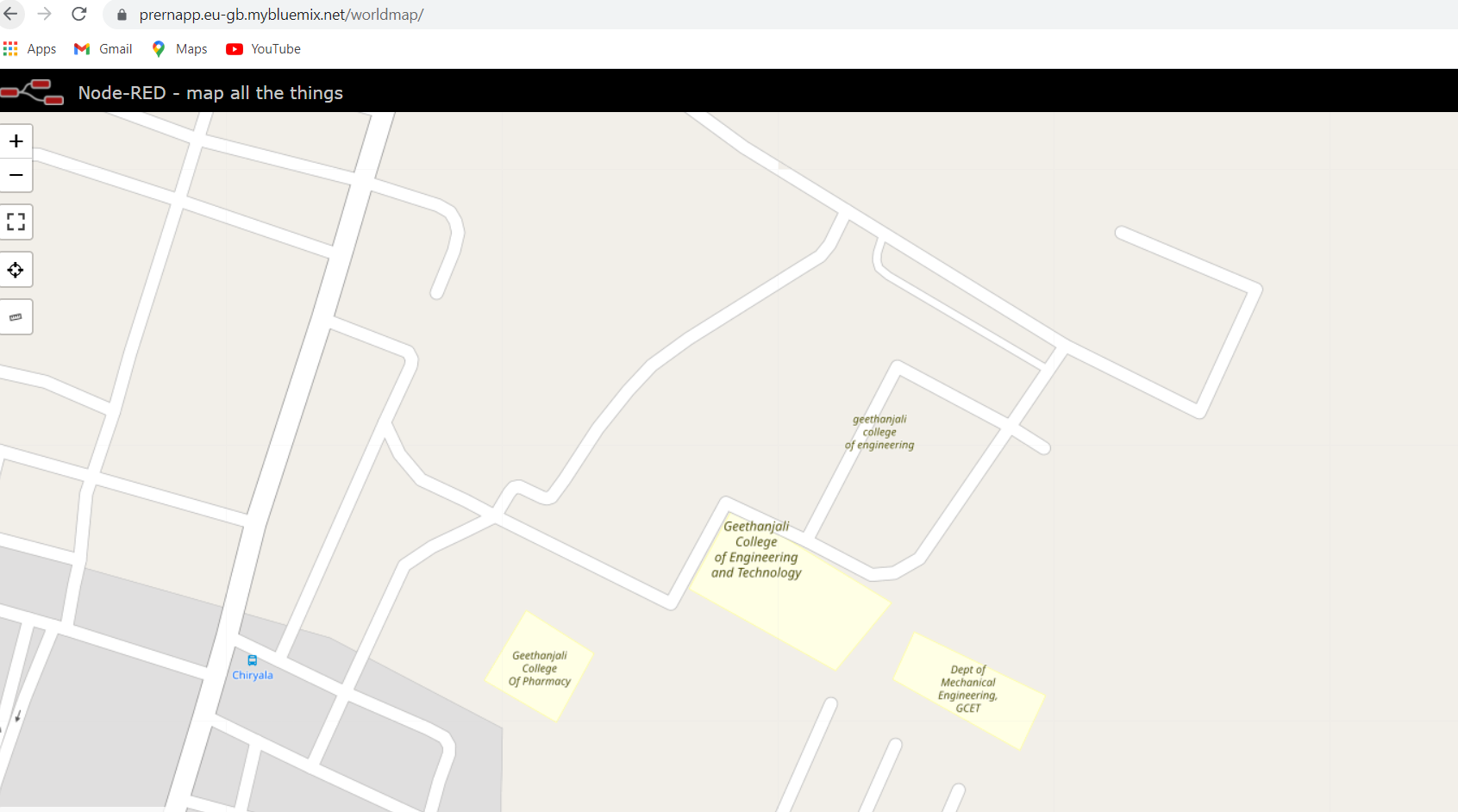
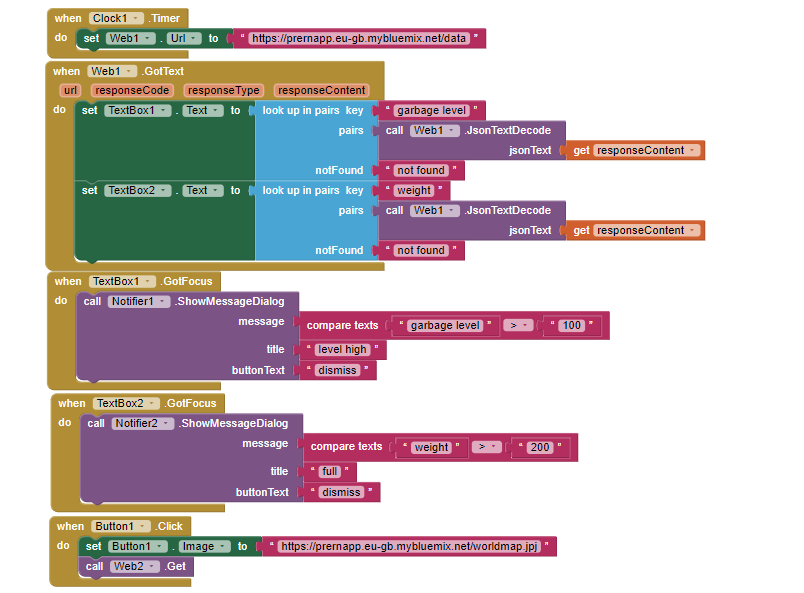


CHANGED CODE, INPUTS AND OUTPUTS

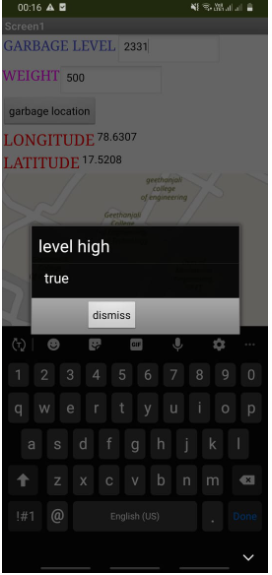


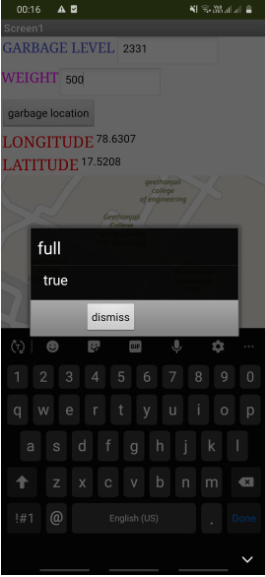












CODE

#include <WiFi.h>

#include <PubSubClient.h>

int distance;

int a;

String command;

String data="";

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

// CHANGE TO YOUR WIFI CREDENTIALS

const char\* ssid = "dlink-299C";//your wifi ssid

const char\* password = "njpa2410";//your password

// CHANGE TO YOUR DEVICE CREDENTIALS AS PER IN IBM BLUMIX

#define ORG "vrwhb5"

#define DEVICE\_TYPE "ESP32"

#define DEVICE\_ID "13579"

#define TOKEN "Pre12345@" // Authentication Token OF THE DEVICE

// PIN DECLARATIONS

int echopin=4;

int trigpin=16;

const int ldrin = 34;

int ldrvalue;

float longitude=17.5208;

float latitude =78.6307;

//-------- Customise the above values --------

const char publishTopic[] = "iot-2/evt/Data/fmt/json";

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/cmd/home/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883, callback, wifiClient);int publishInterval = 5000; // 30 seconds

long lastPublishMillis;

void publishData();

void setup() {

pinMode(echopin,INPUT);

pinMode(trigpin,OUTPUT);

Serial.begin(115200);

Serial.println();

wifiConnect();

mqttConnect();

}

void loop() {

digitalWrite(trigpin,HIGH);

delay(1000);

digitalWrite(trigpin,LOW);

int duration=pulseIn(echopin,HIGH);

distance=(duration\*0.034)/2;

delay(1000);

int ldrvalue= analogRead(ldrin);

delay(1000);

if (millis() - lastPublishMillis > publishInterval)

{

publishData();

lastPublishMillis = millis();

}

if (!client.loop()) {

mqttConnect();

}

}

void wifiConnect() {

Serial.print("Connecting to "); Serial.print(ssid);

WiFi.begin(ssid, password);

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);

}

initManagedDevice();

Serial.println();

}

}

void initManagedDevice() {

if (client.subscribe(topic)) {

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void callback(char\* topic, byte\* payload, unsigned int payloadLength) {

Serial.print("callback invoked for topic: ");

Serial.println(topic);

for (int i = 0; i < payloadLength; i++) {

command+= (char)payload[i];

}

Serial.print("data: "+ command);

command= "";

}

void publishData()

{

String payload = "{\"d\":{\"distance\":";

payload += distance;

payload += ",""\"ldrvalue\":";

payload += ldrvalue;

payload += ",""\"longitude\":";

payload += longitude;

payload += ",""\"latitude\":";

payload += latitude;

payload += "}}";

Serial.print("\n");

Serial.print("Sending payload: "); Serial.println(payload);

if (client.publish(publishTopic, (char\*) payload.c\_str())) {

Serial.println("Publish OK");

} else {

Serial.println("Publish FAILED");

}

}