

SPRINT 1

Date	16/11/2022
Team ID	PNT2022TMID14966
Project Name	Project – Signs with Smart Connectivity for Better Road Safety

Wokwi Simulation:

The screenshot displays the Wokwi simulation environment. On the left, the code editor shows the `sketch.ino` file with C++ code for an ESP32 connected to a DHT22 sensor and an LED. The code includes WiFi and MQTT libraries, sensor setup, and publish logic to an IBM Watson IoT Platform. On the right, the simulation window shows the physical circuit connections: the ESP32's digital pins 14 and 15 are connected to the DHT22's VCC and GND respectively; the DHT22's SDA and SCL pins are connected to ESP32 pins 21 and 22; and the LED's VCC pin is connected to ESP32 pin 13. The LED is labeled "ON" and has pins 1 through 8. The DHT22 sensor is labeled "DHT22". The simulation interface includes a "Simulation" tab, a toolbar with play/pause/stop buttons, and a status bar showing time (00:13.771) and battery (91%). The bottom of the screen shows the terminal output with the published JSON payload: {"temp":37.40,"humidity":86.00,"North":0,"South":0,"East":0,"West":0} followed by "Publish ok", "temp:37.40", "humidity:86.00", "Sending payload:", and another "Publish ok".

```
sketch.ino    diagram.json    libraries.txt    Library Manager ▾

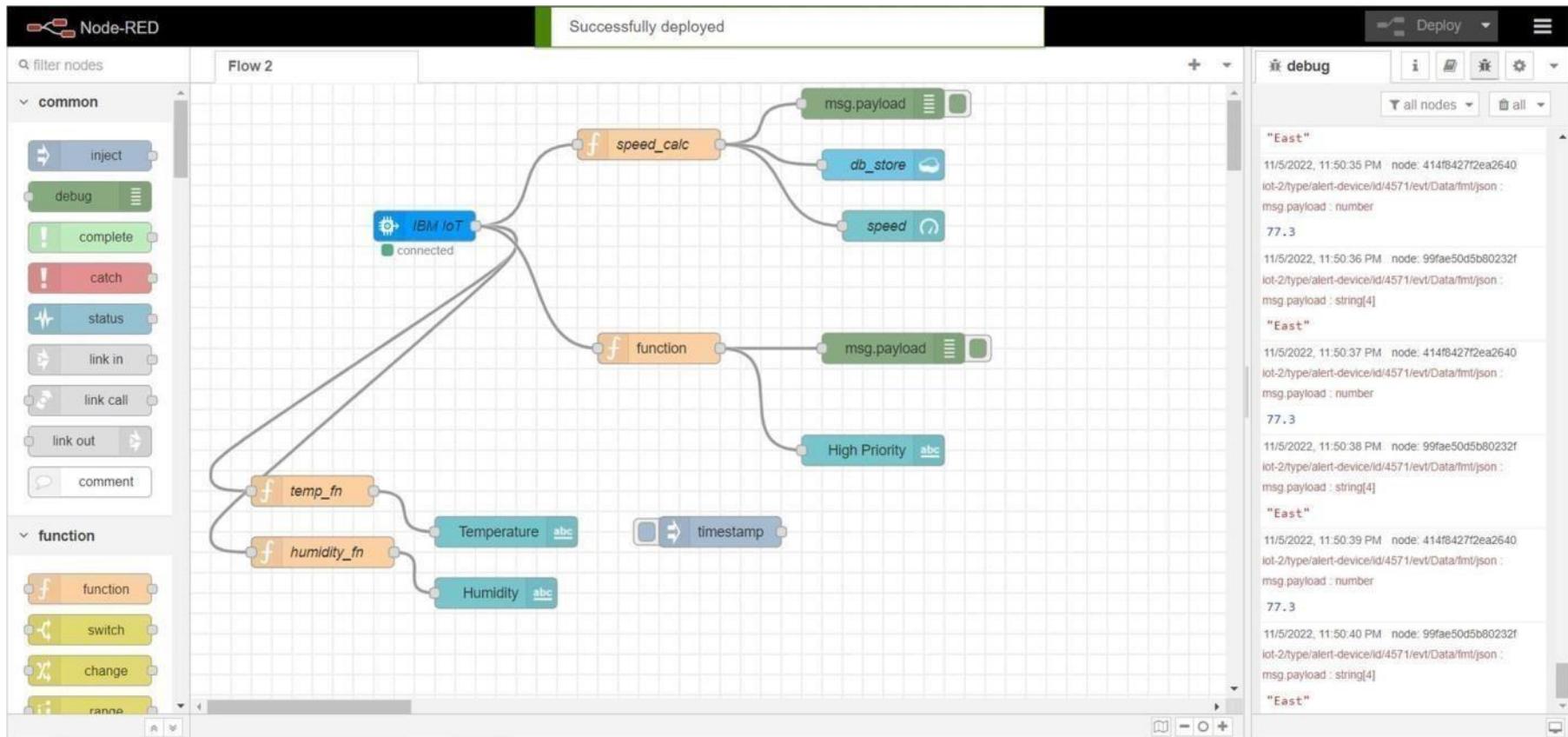
1 #include <WiFi.h>/library for wifi
2 #include <PubSubClient.h>/library for MQTT
3 #include "DHT.h"// Library for dht11
4 #define DHTPIN 5 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6
7 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connect
8
9 void callback(char* subscribeTopic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "psh4py"//IBM ORGANITION ID
14 #define DEVICE_TYPE "alert-device"//Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "4571"//Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "12345678" //Token
17 String data3;
18 float h, t;
19
20
21 //----- Customise the above values -----
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt>Data/fmt/json"; // topic name and type of event perform a
24 char subscribeTopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
28
29
30 //-
31 WiFiClient wifiClient; // creating the instance for wifiClient
32 PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client
```

IoT Device – IoT Platform

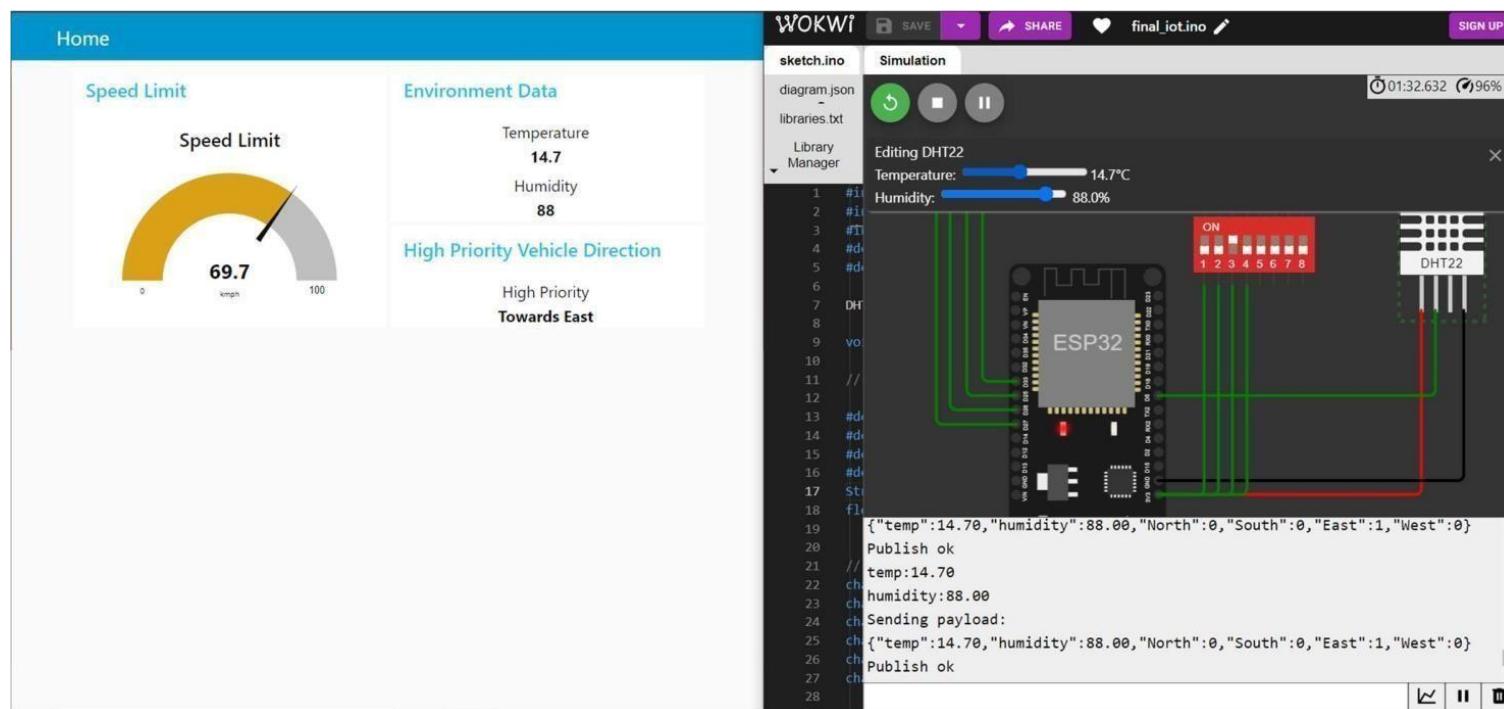
The screenshot shows a web-based interface for managing an IoT device. At the top, there's a header bar with a dropdown arrow, a square icon, the ID '4571', a green 'Connected' status indicator, the device name 'alert-device', and a 'Device' tab. Below the header is a navigation bar with tabs: 'Identity', 'Device Information', 'Recent Events' (which is underlined, indicating it's the active tab), 'State', and 'Logs'. A descriptive text below the tabs states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table follows, listing three recent events. The table has columns: 'Event', 'Value', 'Format', and 'Last Received'. Each row shows a 'Data' event with a JSON value containing 'temp': 23.4, 'humidity': 63, 'North': 1, 'South': 0, and a timestamp of 'a few seconds ago'.

Event	Value	Format	Last Received
Data	{"temp":23.4,"humidity":63,"North":1,"South":0,...	json	a few seconds ago
Data	{"temp":23.4,"humidity":63,"North":1,"South":0,...	json	a few seconds ago
Data	{"temp":23.4,"humidity":63,"North":1,"South":0,...	json	a few seconds ago

Node Red



Node Red Web UI



Home

Speed Limit

Speed Limit

70.5 kmph

Environment Data

Temperature 15.5

Humidity 91.5

High Priority Vehicle Direction

High Priority Towards South

WOKWI

sketch.ino

diagram.json

libraries.txt

Library Manager

Editing DHT22

Temperature: 15.5°C

Humidity: 91.5%

02:23:068 91%

ON

DHT22

```
1 #include <DHT.h>
2 #define DHTPIN 2
3 #define DHTTYPE DHT22
4
5 void setup() {
6     // initialize serial communication:
7     Serial.begin(9600);
8
9     // set the data pin as an input:
10    pinMode(DHTPIN, INPUT);
11
12    // start library:
13    DHT.begin(DHTPIN, DHTTYPE);
14
15    // print basic info:
16    Serial.print("temp:");
17    Serial.print(DHT.readTemperature());
18    Serial.print(" humidity:");
19    Serial.print(DHT.readHumidity());
20
21    // publish to MQTT topic:
22    ch.publish("temp", temp);
23    ch.publish("humidity", humidity);
24
25    // send a payload:
26    ch.publish("payload", "{\"temp\":15.50,\"humidity\":91.50,\"North\":0,\"South\":1,\"East\":0,\"West\":0}");
27
28 }
```

Cloudant Database

The screenshot shows the Cloudant NoSQL Database interface. On the left, a sidebar menu includes options like All Documents, Query, Permissions, Changes, Design Documents, and a Log Out button. The main area displays a table of documents in the 'data_iot' database. The table has two columns: '_id' and 'Payload'. The '_id' column lists document IDs, and the 'Payload' column lists numerical values. A total of 14 documents are listed.

_id	Payload
060cc88d44faf11288e9cdfd7d8de45a	35
060cc88d44faf11288e9cdfd7d904e58	60
060cc88d44faf11288e9cdfd7d90c3f9	45.5
060cc88d44faf11288e9cdfd7d92a313	60
2314e7571ab5925365e082f191bb2c9c	52.5
26939bb99e5c84bed4f6a20342a22ab2	35
26939bb99e5c84bed4f6a20342a7ccd5	44
3ffa1240575d0cd0d7f848833802e389	55
48a3afbcf5f840466e09ed279d3c3451	53
48a3afbcf5f840466e09ed279d3c5b7c	53
48a3afbcf5f840466e09ed279d3c9545	53
52730057f2d5fde2d21dfaaaabc10dc8	55