ProIT-D_WS2324 - Internet of Things (IoT)

Project Documentation of Kalyan Shencottah

Hochschule fuer Technik und Wirtschaft (HTW), Berlin, January 2024

# Motivation

This Project is part of deliverable of Winter Semester (WS2324) Coursework Internet of Things (IoT) guided by Professor Alexandar Huhn for the Programme, Professional IT Business and Digitalization at Hochschule for Technik und Wirtschaft (HTW), Berlin.

Motivation of this Project is to gain understanding of the IoT concepts as discussed in the class. Concepts include Protocols such as MQTT, Eclipse Paho, and COAP, use of softwares such as Node-Red to create Flows that gets deployed to Raspberry PI, Use of Thonny to deploy code to Microcontroller, Simulation using WokWi and interaction with Telegram Bot.

This documentation explains the literatures I surveyed to gain Conceptual understanding User Interface used, Data Processing aspects, Connectivity and Nodes, and then the execution of tasks to gain practical experience. Screenshots are provided to support my understanding and experience gained.

# User Interface

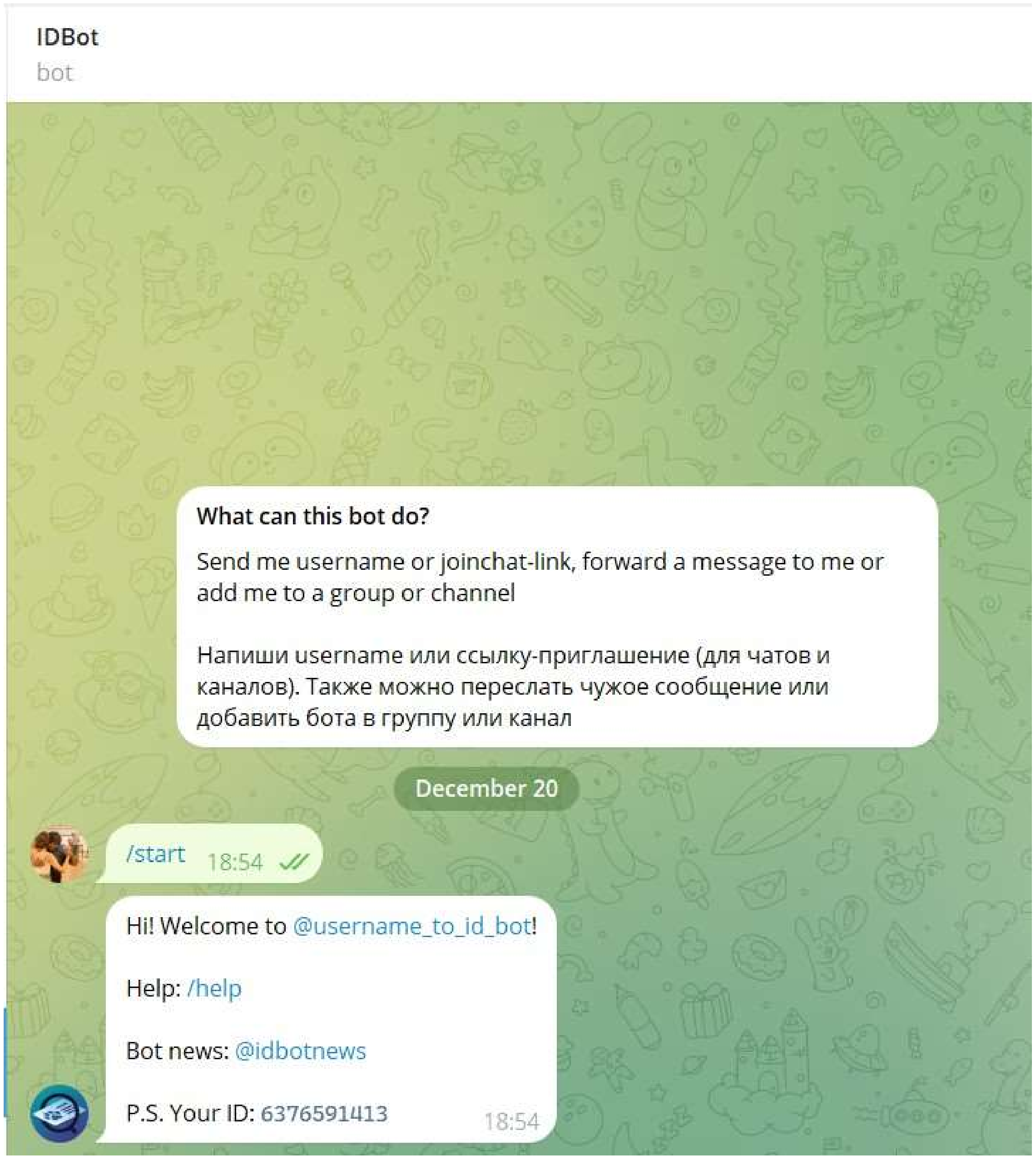
IoT devices open the door to all sorts of computing potential, but they can also produce a flood of telemetry data that users need to properly collect and monitor to ensure those devices are working properly. For this purpose, we need a proper User Interface that helps collect and monitor data. I did literature survey of different Interfaces such as Grafana, Matrix, Telegram and Node Red.

Grafana is being used in many industries (for instance in an aquaponic farm in South Africa, in an electroplating factory in Ohio or Monitoring a pet python in UK). Grafana Dashboards are also being used in monitoring Satellites and collecting data(source: Libre Space Foundation).

Matrix.org is a new open standard for distributed, real-time communication. IoT platform can be built on Matrix.

User Interfaces used in this project are Node Red and Telegram. Node Red was used to create Flows that gets deployed to Raspberry PI (ESP8266 and ESP32). Telegram interface is established through TelegramBot. A Bot ID and Telegram Token identifies communication to the Bot I had created.

The following image displays the IDBot and its content. Having a Telegram BotID is important. Anyone will be able to send message to another using the BotID.



IDBot

bot

What can this bot do?

Send me username or joinchat-link, forward a message to me or add me to a group or channel

Напиши username или ссылку-приглашение (для чатов и каналов). Также можно переслать чужое сообщение или добавить бота в группу или канал

December 20

/start 18:54

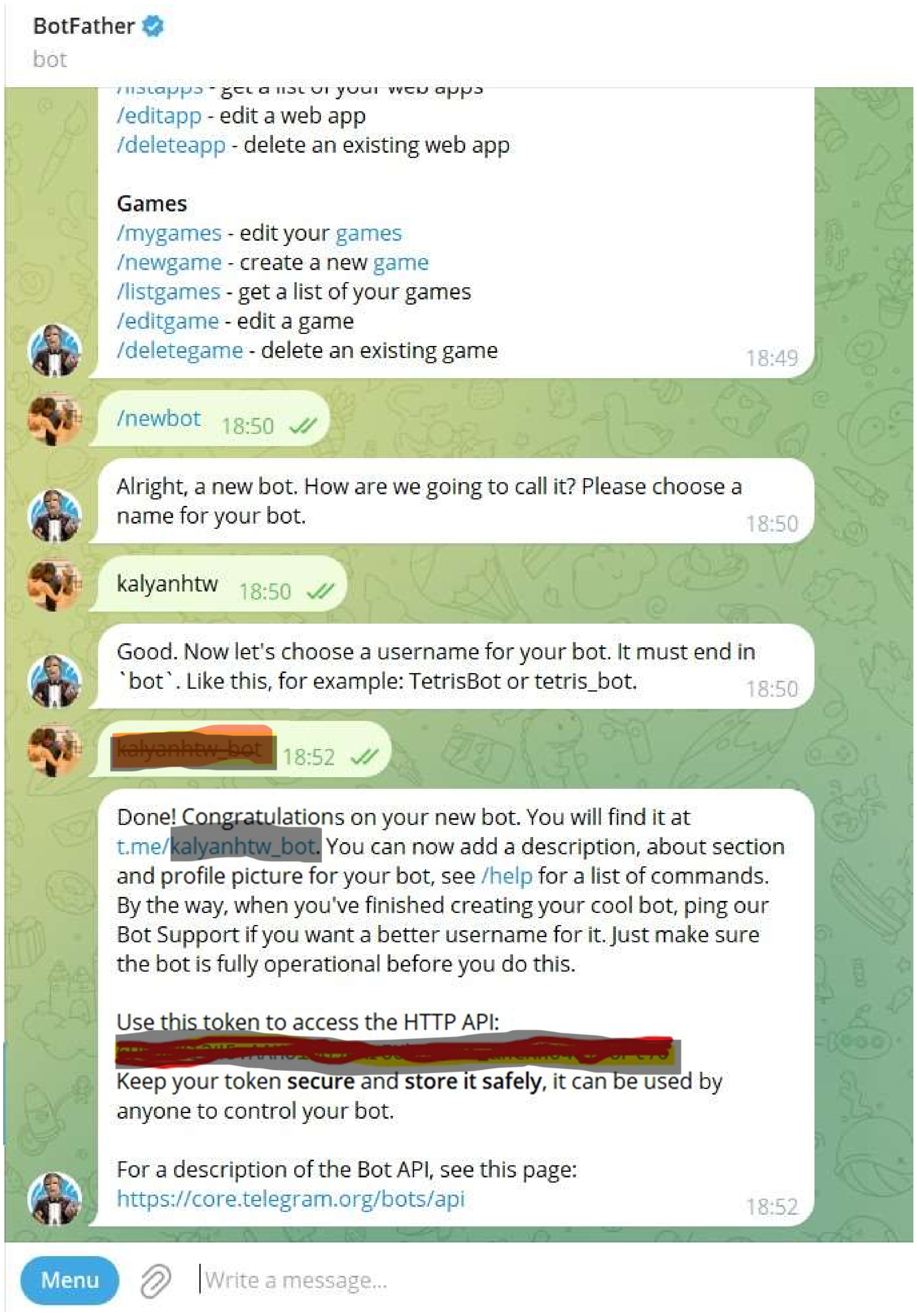
Hi! Welcome to @username_to_id_bot!

Help: /help

Bot news: @idbotnews

P.S. Your ID: 6376591413 18:54

Telegram's BotFather helps create a Bot. A Bot need to have suffix as "bot" or "_bot". Token ID is unique for a user and it helps communicate through HTTP API. The following screen displays the look and feel of Telegram's BotFather. One can see the Bot user with and without "bot" suffix is displayed. Generated Token ID is masked for security reasons.



BotFather

bot

/listapps - get a list of your web apps
/editapp - edit a web app
/deleteapp - delete an existing web app

**Games**
/mygames - edit your games
/newgame - create a new game
/listgames - get a list of your games
/editgame - edit a game
/deletegame - delete an existing game 18:49

/newbot 18:50

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 18:50

kalyanhtw 18:50

Good. Now let's choose a username for your bot. It must end in \`bot\`. Like this, for example: TetrisBot or tetris_bot. 18:50

[redacted] 18:52

Done! Congratulations on your new bot. You will find it at t.me/kalyanhtw_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
[redacted]
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:
https://core.telegram.org/bots/api 18:52

## Data Processing

Data Processing is facilitated using different Protocols such as UDP, MQTT, UART, Eclipse Paho and COAP. Data Processing involves communication on different OSI Layers using the above Protocols. The functionality also involves reading from and writing to Databases such as Influx DB.

In computer networking, the User Datagram Protocol (UDP) is one of the core communication protocols of the Internet protocol suite used to send messages (transported as datagrams in packets) to other hosts on an Internet Protocol (IP) network. Within an IP network, UDP does not require prior communication to set up communication channels or data paths.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms and is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

MQTT (MQ Telemetry Transport) is a lightweight, publish-subscribe, machine to machine network protocol for message queue/message queuing service. It is used in the field of Internet of Things (IoT) as it is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP, but also possibly over QUIC(a Transport Layer Protocol). It is an open OASIS standard and an ISO recommendation (ISO/IEC 20922).

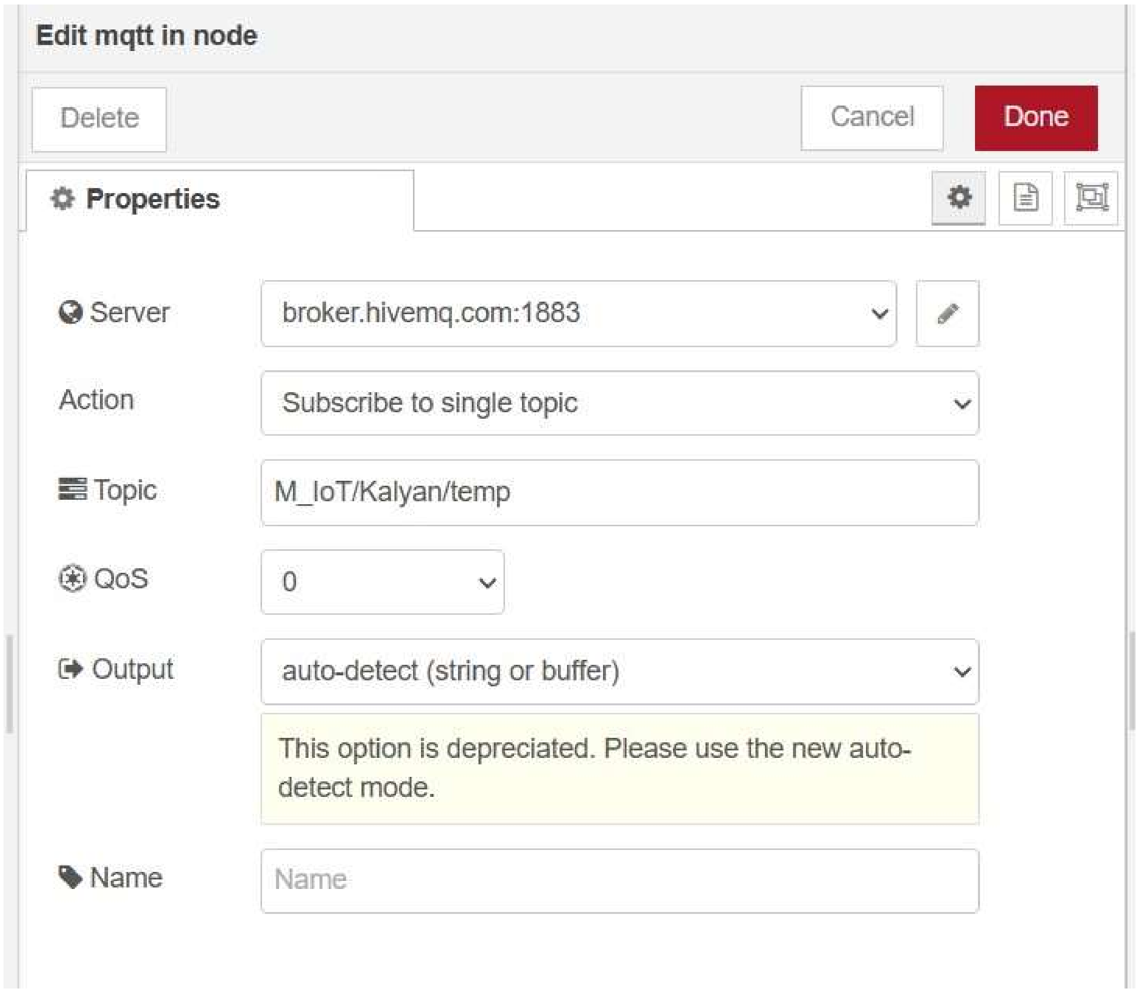
A UART (Universal Asynchronous Receiver/Transmitter) is a hardware device designed for asynchronous communication. This is not a communication protocol like SPI or I2C but a physical circuit. In this case, you can configure the data format and transmission speed. It is one of the most used device-to-device communication protocols. UART is mainly used by embedded systems, microcontrollers, and computers as a device-to-device hardware communication protocol. In contrast to the other communication protocols, UART uses only two wires for transmission and reception.

For outbound communications, it transforms parallel data into serial data. For inbound communications, it transforms serial data into parallel data. The parity bit is added to outgoing transmissions, and the parity bit is checked for inbound transmissions. Manages to interrupt requests and device management, involving synchronizing the operating speed of the computer and the device.

Eclipse Paho is a MQTT (Message Queuing Telemetry Transport) implementation. Paho is available on various platforms and programming languages like Java, C, C#, JavaScript and Python.

Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. CoAP is designed to enable simple, constrained devices to join the IoT even through constrained networks with low bandwidth and low availability.

Node Red also establishes connectivity among IoT Devices and facilitates Data Processsing. It facilitates connection in the IoT infrastructure, allowing for easy integration of hardware devices, APIs, and online services. One can connect different devices and services by connecting nodes in one's desired flow. Node-RED is built on Node.js, an open-source, cross-platform server environment. One can visualize the flow of data and functions on data before deploying Node-Red to any IoT Devices(e.g. Raspberry PI)



|  |  |
|---|---|
| Server | broker.hivemq.com:1883 |
| Action | Subscribe to single topic |
| Topic | M_IoT/Kalyan/temp |
| QoS | 0 |
| Output | auto-detect (string or buffer) |
|  | This option is depreciated. Please use the new auto-detect mode. |
| Name | Name |

## Connectivity

Connectivity plays a significant role in IoT. Some Connectivity aspects I surveyed include LoRa, LoRaWAN, Bluetooth, and WiFi. Protocols like MQTT and COAP that were discussed in above sections form an integral part in establishing Connectivity.

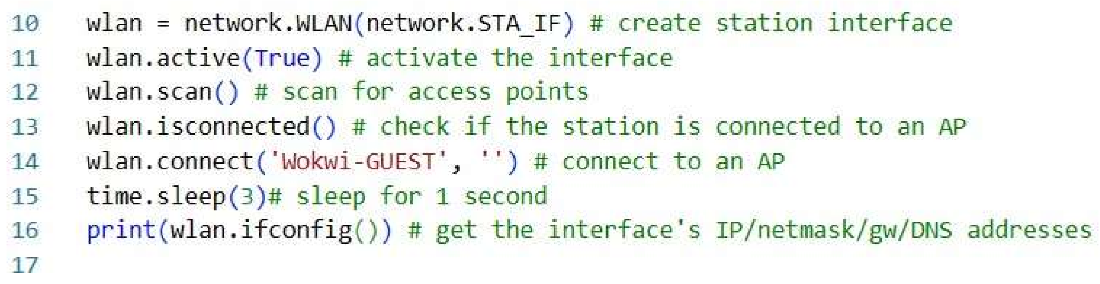
LoRa (from "long range") is a physical proprietary radio communication technique. It is based on spread spectrum modulation techniques derived from chirp spread spectrum (CSS) technology. LoRaWAN (wide area network) defines the communication protocol and system architecture. LoRaWAN is an official standard of the International Telecommunication Union (ITU), ITU-T Y.4480.

Together, LoRa and LoRaWAN define a low-power, wide-area (LPWA) networking protocol designed to wirelessly connect battery operated devices to the internet in regional, national or global networks, and targets key Internet of things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services. The low power, low bit rate, and IoT use distinguish this type of network from a wireless WAN that is designed to connect users or businesses, and carry more data, using more power.

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

There are also providers like Adafruit IO that provides cloud service. It is a service meant for storing and retrieving data and uses MQTT Protocol. It helps connectivity of one's project over the Internet, say, to control Motors, read Sensor Data and more.



```
wlan = network.WLAN(network.STA_IF) # create station interface
wlan.active(True) # activate the interface
wlan.scan() # scan for access points
wlan.isconnected() # check if the station is connected to an AP
wlan.connect('Wokwi-GUEST', '') # connect to an AP
time.sleep(3)# sleep for 1 second
print(wlan.ifconfig()) # get the interface's IP/netmask/gw/DNS addresses

```

## Nodes

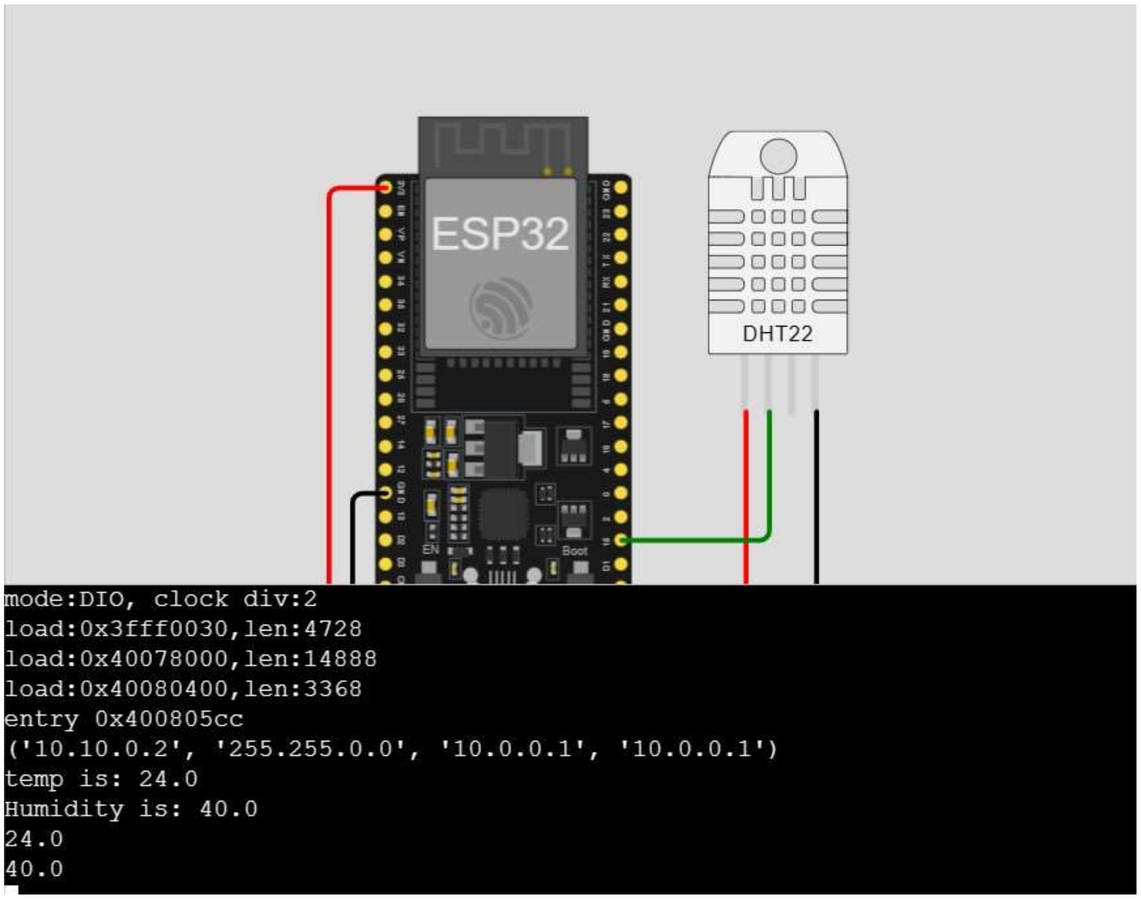
IoT Nodes serve as devices that establish connectivity to the Internet via a gateway, effectively enabling the integration of the physical world into the vast realm of the Internet. Within an IoT ecosystem, these nodes function as crucial components for bridging the gap between the physical and virtual world.

Examples of Nodes in IoT include Microcontroller (with Wireless connection), Sensors and Actuators. Nodes help connect with Microcontroller devices like ESP8266 and ESP32, Arduino. Micro Python, Thonny and Arduino IDE are the programming interfaces that helps deploy code to the devices.

Nodes like Sensors and Actuators uses Interfaces like 1 Wire Bus(half-duplex serial Bus), PWM (uses Pulse Width Modulation concept to count pulses or control the voltage if connected to Electric devices), SPI and I2C (SPI is a full-duplex communication and I2C is a half-duplex communication. SPI supports single-master and I2C supports multi-master/multi-slave), and UART (a hardware device designed for asynchronous communication).

Other Nodes or devices that could be connected for different measurements include Magnetometer, Accelerometer, Gyroscope, Ultrasonic Distance Measurement, Positioning. There are sensors that does the measurements.

Following picture depicts the ESP32 Node Simulator used to collect Temperature and Humidity data. A wifi-connection is established first and data collected using MicroPython code.



```
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
('10.10.0.2', '255.255.0.0', '10.0.0.1', '10.0.0.1')
temp is: 24.0
Humidity is: 40.0
24.0
40.0
```

## Node Red Flow Design

Node Red was used to design flows that gets deployed to the Raspberry PI. Dashboard was created to present the Data collected for Temperature and Humidity. Groups were designed to display the data I had collected and also to display data collected by others.

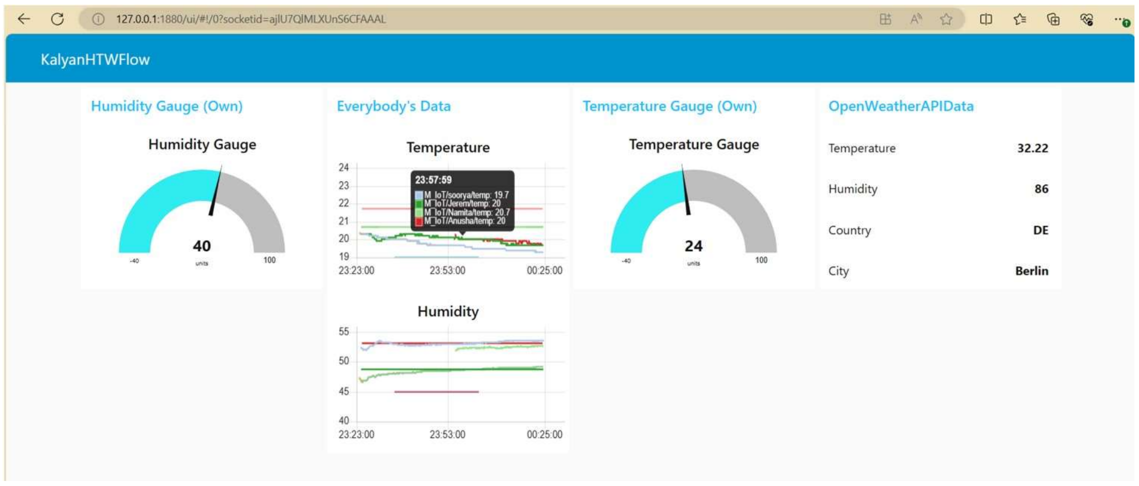
Explored Trigger functionality to trigger Temperature and Humidity Data.

Installed PostgreSQL libraries in Node-Red to facilitate communication with PostgreSQL Relational Database

## Tests and Results

### Testing the connection to ESP32 with Wokwi simulator

The task involve establishing connection with ESP32 using WiFi connectivity. Temperature and Humidity Data are collected and displayed. Data displayed include the Data from my own account and data collected by other team members. The following screen displays the output of Temperature and Humidity data collected by me and data collected by others.

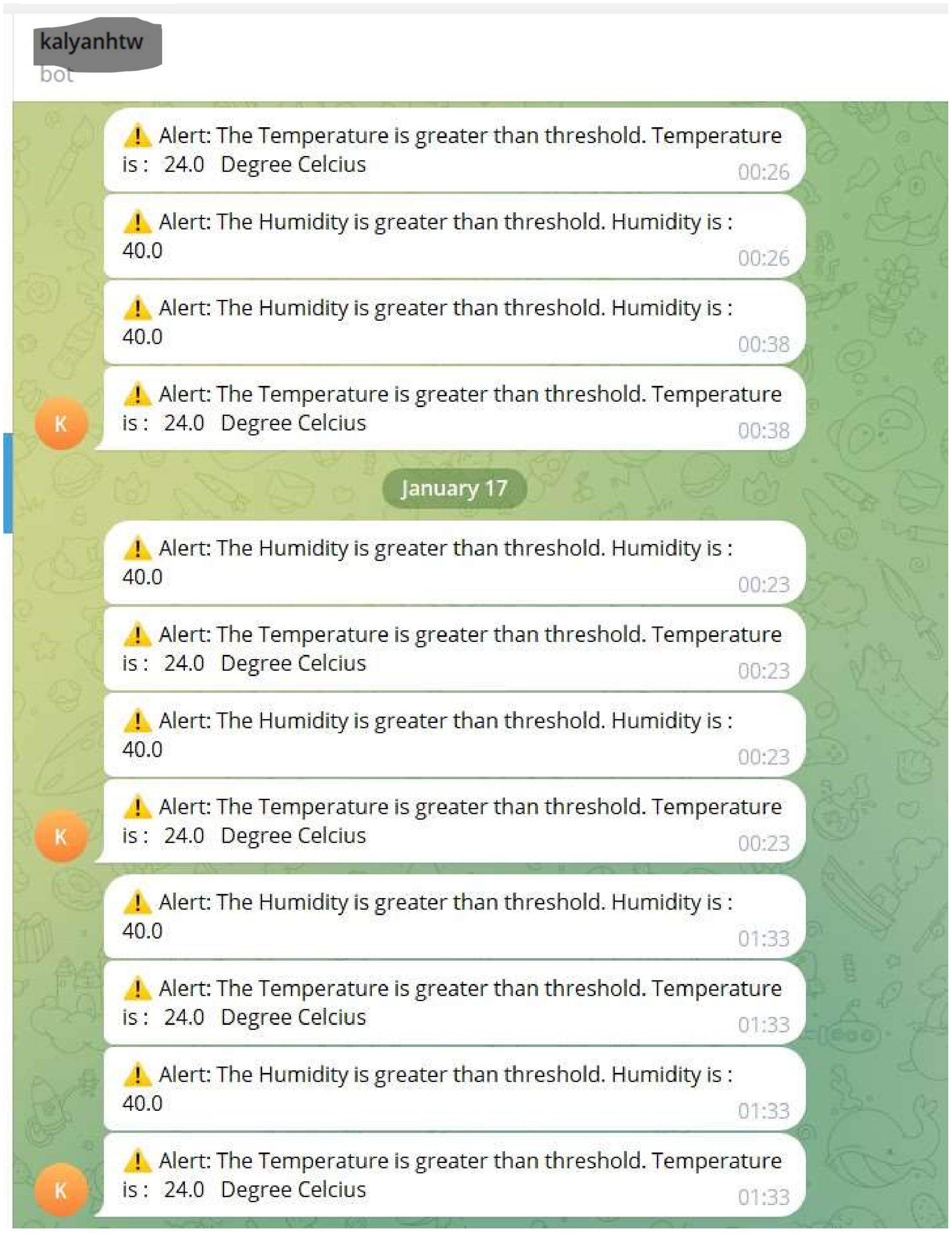


| | |
|---|---|
| Temperature | 32.22 |
| Humidity | 86 |
| Country | DE |
| City | Berlin |

### Testing the connection with Telegram Interface

The task involves sending the Temperature and Humidity data to Telegram Interface. Alert is generated if the Temperature collected goes beyond a certain Threshold.

The following screen displays the Alert sent to Telegram Bot if the Temperature and Humidity data goes beyond certain threshold.



[redacted]
bot

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  00:26

⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  00:26

⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  00:38

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  00:38

January 17

⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  00:23

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  00:23

⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  00:23

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  00:23

⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  01:33

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  01:33

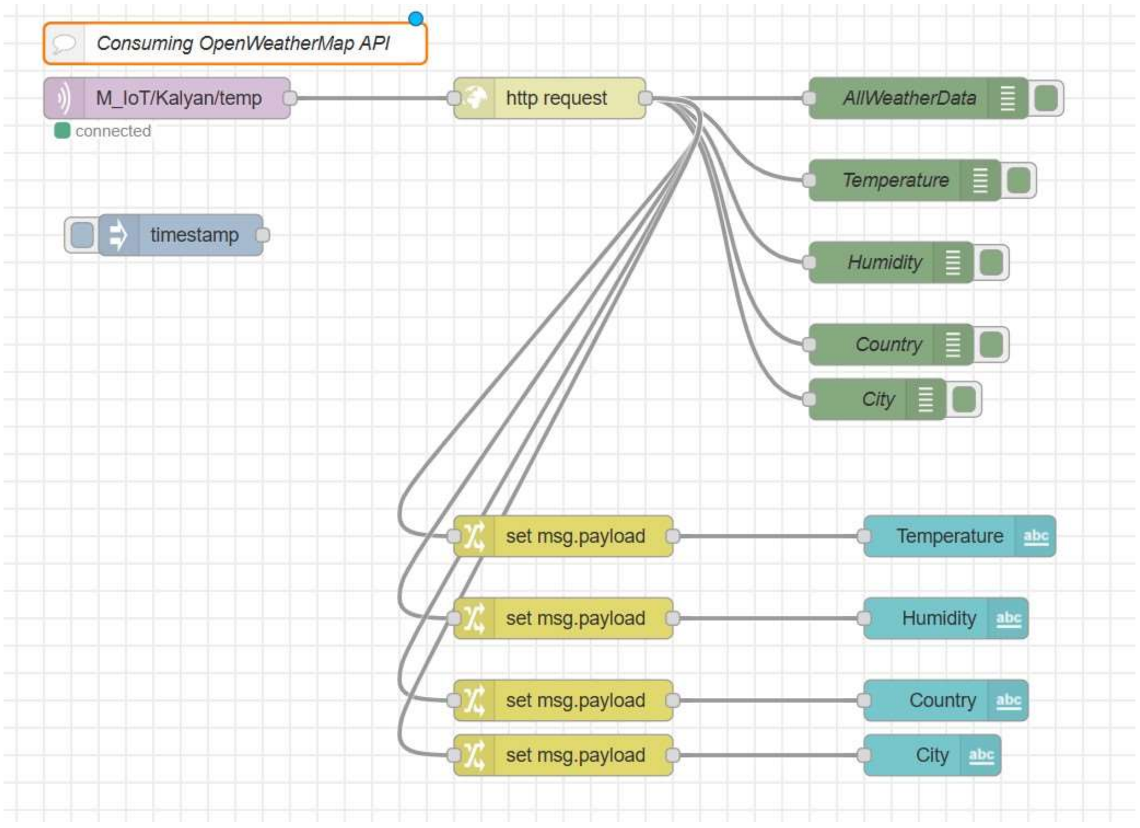
⚠ Alert: The Humidity is greater than threshold. Humidity is : 40.0  01:33

⚠ Alert: The Temperature is greater than threshold. Temperature is :  24.0   Degree Celcius  01:33

## Consuming OpenWeatherMap API and displaying weather data

The task involve consuming Openweathermap API to collect data pertaining to weather in a City. The API I had used collect Temperature and Humidity Data pertaining to Berlin in Deutschland. Imperial units are used.

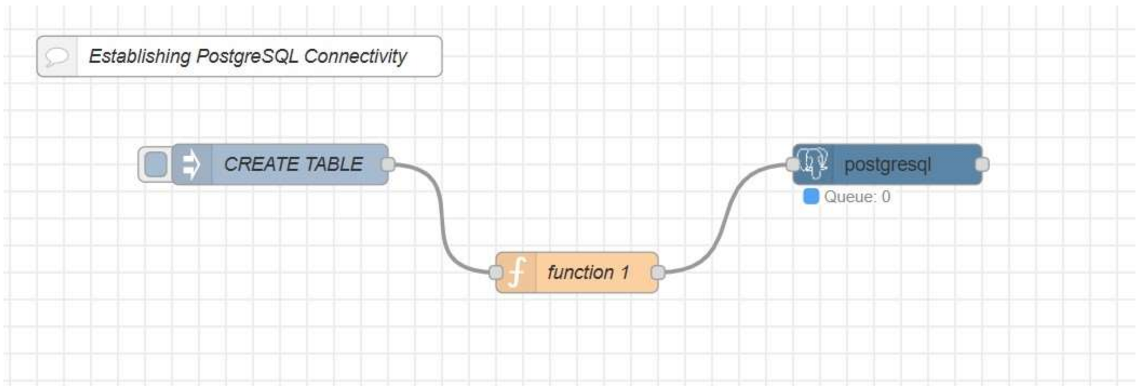
The following screen displays the design of consuming OpenWeatherMap API.

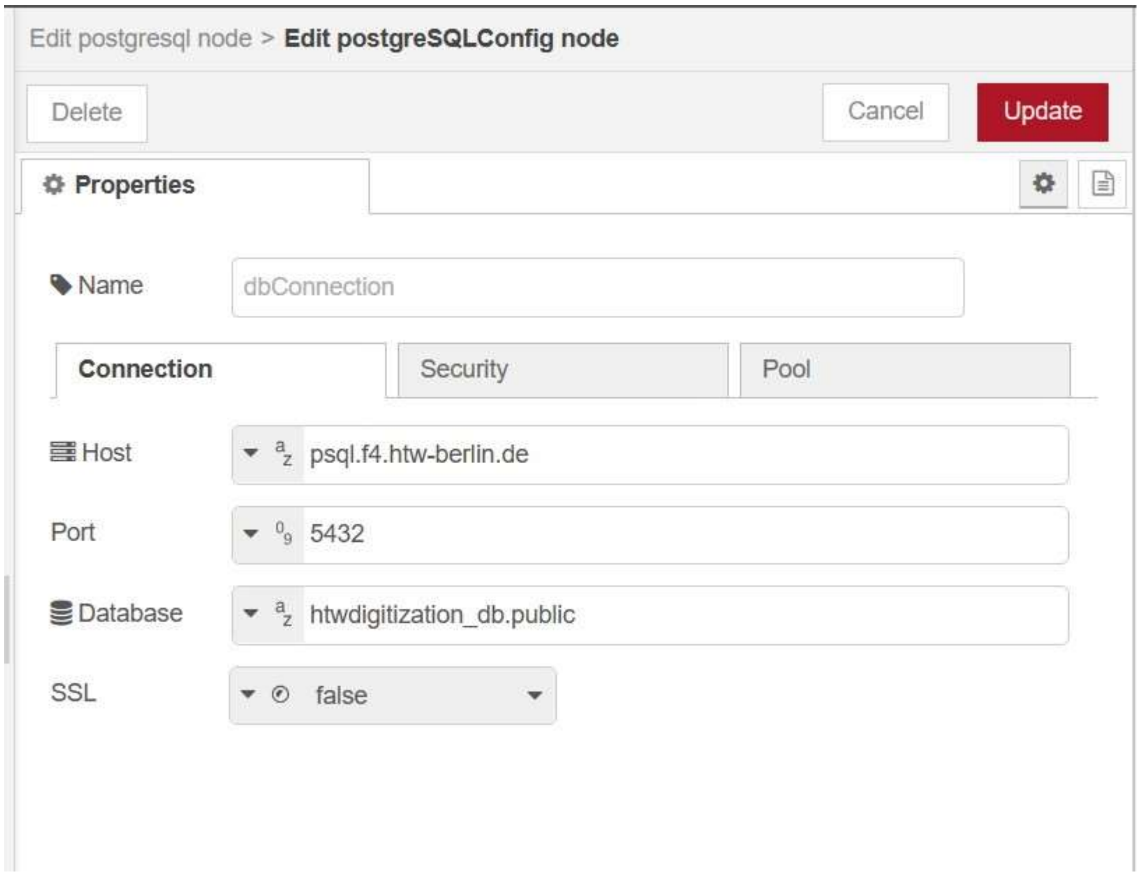


## Establishing Connectivity to PostgreSQL Database

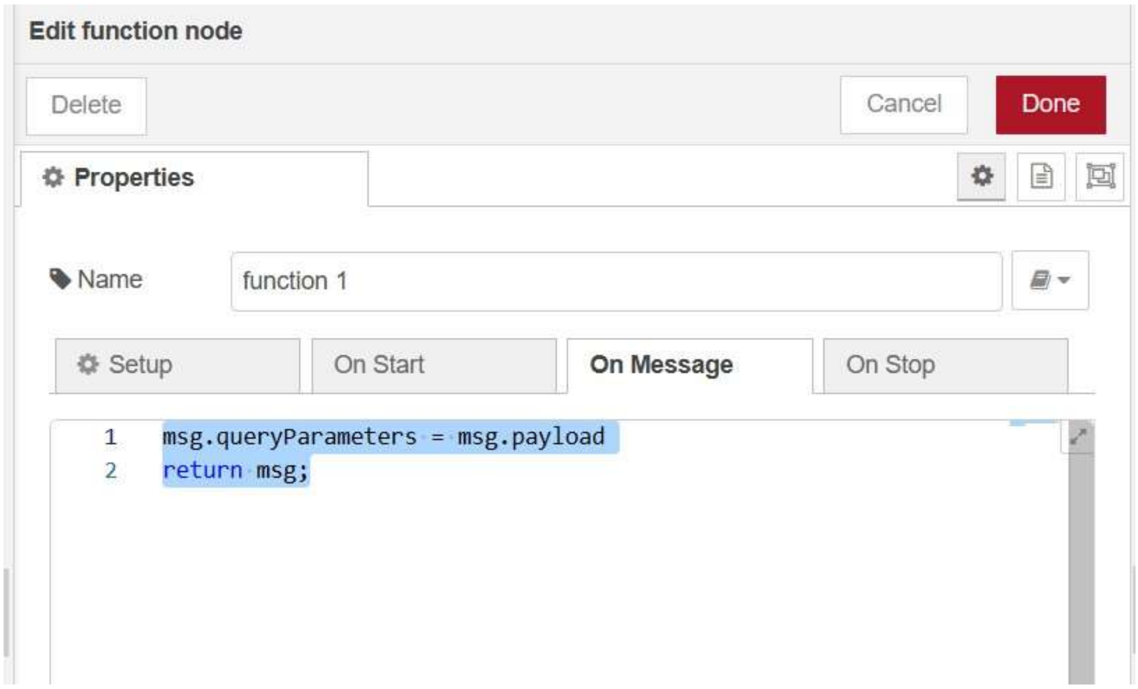
Attempt made to establish connectivity to PostgreSQL Database I had created named htwdigitization_db. Schema name is "public". The task was to CREATE a TABLE to store WEATHER data with fields Temperature and Humidity. The Database is hosted in HTW Host psql.f4.htw-berlin.de with Port 5432.

The following screen displays the design of connecting to PostgreSQL Database





Name: dbConnection
Host: psql.f4.htw-berlin.de
Port: 5432
Database: htwdigitization_db.public
SSL: false



Name: function 1

```
msg.queryParameters = msg.payload
return msg;
```

Table created at the Database(htwdigitization_db, schema:public)

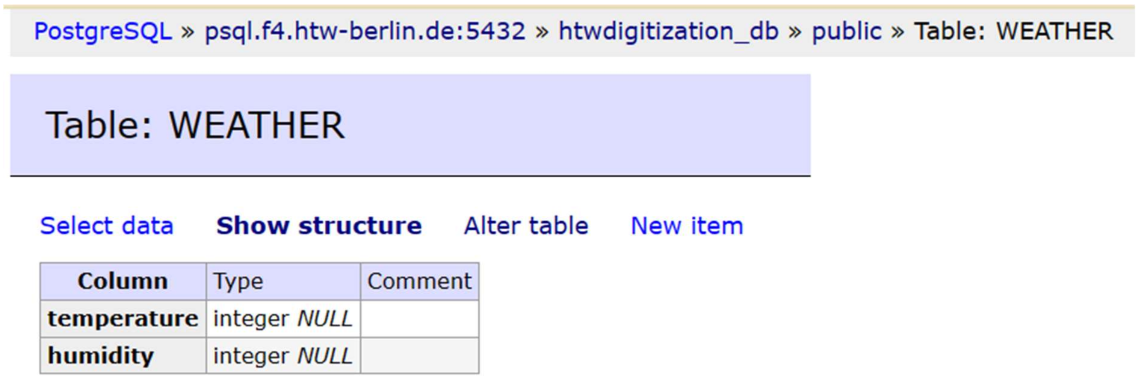

PostgreSQL » psql.f4.htw-berlin.de:5432 » htwdigitization_db » public » Table: WEATHER

Table: WEATHER

| Column | Type | Comment |
|---|---|---|
| **temperature** | integer *NULL* | |
| **humidity** | integer *NULL* | |

DML statements(SELECT, INSERT, UPDATE, DELETE) could be written based on the data obtained from msg.payload through MQTT broker.

## Lessons Learned and Conclusion

The tasks helped me gain deeper understanding of the Interoperability among the User Interfaces, Communication Protocols, Connectivity aspects and Hardware Devices.

Following are the Lessons learned and challenges:

-> Challenges include troubleshooting connectivity issues as there are interactions between Hardware, Drivers, Softwares, and Protocols.

-> Communication with external interfaces like Telegram, Raspberry PI, Databases (say, PostgreSQL) requires more deeper understanding of the interface issues and troubleshooting of the same.

-> For Database, Hosting solution need to be finalized, whether it could be hosted in local machine or in a shared server environment, or Cloud. Certificates need to be established (SSL, TLS Handshake)

-> Consuming API, in this case OpenWeatherMap.org, incurs cost. Requires API Key and understanding of how to consume the same (Metrics, Language, Country Codes etc.)

Overall, Internet of Things is a field that requires deeper understanding of User Interfaces, Data Processing, Connectivity, Nodes and interoperability of those concepts.