

Software Engineering for the Internet Of Things

Project Title: Elderly Patient Monitoring System

GitHub Link: <https://github.com/nawshin81/EHloT>

Group ID: 09

Submitted To: Prof. Davide Di Ruscio

Submitted By:

1. Kiran Singh Saud
2. Muhammad Sharjeel Maqsood
3. Nawshin Ulfat

Introduction.....	3
System Architecture.....	3
Technology.....	4
Hardware Simulation.....	5
Wireless network (MQTT).....	6
Why MQTT?.....	6
Node-RED.....	7
InfluxDB.....	8
Grafana.....	10
Data Source.....	10
Dashboard.....	12
Alert.....	13
Docker.....	14
Conclusion.....	14
Project set up.....	15

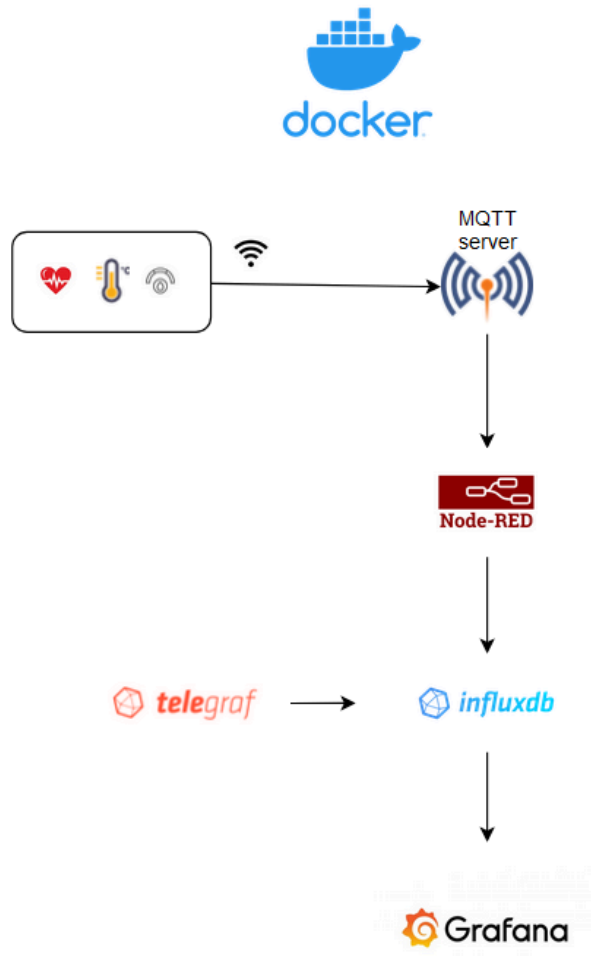
Introduction

The “Elderly Patient Monitoring System” is an IoT based solution for continuous health monitoring for elderly population. As with the growing age, the incidence of chronic diseases and the risk of emergencies like accidental falls increase. Therefore, it becomes crucial to have a system that monitors the crucial health parameters and promptly alerts caregivers and healthcare professionals in case of any abnormality or emergency.

The wearable device consists of three different sensors(temperature sensor, heart rate sensor, accelerometer sensor) that collects data from the patient body and transmit the data over the wireless network (using esp32 microcontroller) to the database. In case of an abnormal reading of data prompt alerts with be sent to the caregivers and healthcare personnels and they can also visualize the data on a dashboard.

System Architecture

The system was implemented by integrating multiple IoT based technologies like, Node-RED, InfluxDB, Grafana. These individual solutions solves parts of the system and combined provides the monitoring system.



Due to the lack of hardware components, the sensors data were simulated on an online based simulator. Multiple instances of the simulated device were created, [Patient 01](#), [Patient 02](#), [Patient 03](#) to check for the scalability. This device consists of ESP32 microcontroller board that is connected to the temperature sensor, heart rate sensor, and accelerometer sensor. This simulated device publishes the sensors data over the wifi network using MQTT protocol.

The data is accessed by the Node-RED, forwarded to InfluxDB to be stored and later sent to Grafana for visualization. All the services mentioned are dockerized except for the simulated device.

Technology

We have used the following technologies for the implementation of our system.

Hardware Simulation

The Elderly Patient Monitoring device was simulated using an online based embedded system simulator called [wokwi](#). The reason behind using this simulator is the ease of use. Wokwi provides the necessary sensors and components to simulate diverse embedded and IoT based solutions, as well as being online also helps with the teamwork.

The components and sensors used:

1. **ESP32 microcontroller:** Due to the compact size and low power consumption, it is ideal for wearable applications. Its built-in Wi-Fi capability also enables seamless integration with MQTT.
2. **DHT22 (Digital Temperature and Humidity Sensor):** This sensor provides both temperature and humidity from the environment. But, for our project we only used the temperature data.
3. **Heart rate sensor:** Wokwi doesn't provide any sensor for measuring heart rate. So, for the sake of simulation we used potentiometer which has the ability to mimic the variable resistance characteristic of a real heart rate sensor.
4. **MPU6050 (Accelerometer and gyroscope sensor):** It is a three-axis gyroscope-accelerometer sensor. We used this sensor to simulate acceleration data in all three axis that will help us to detect fall.

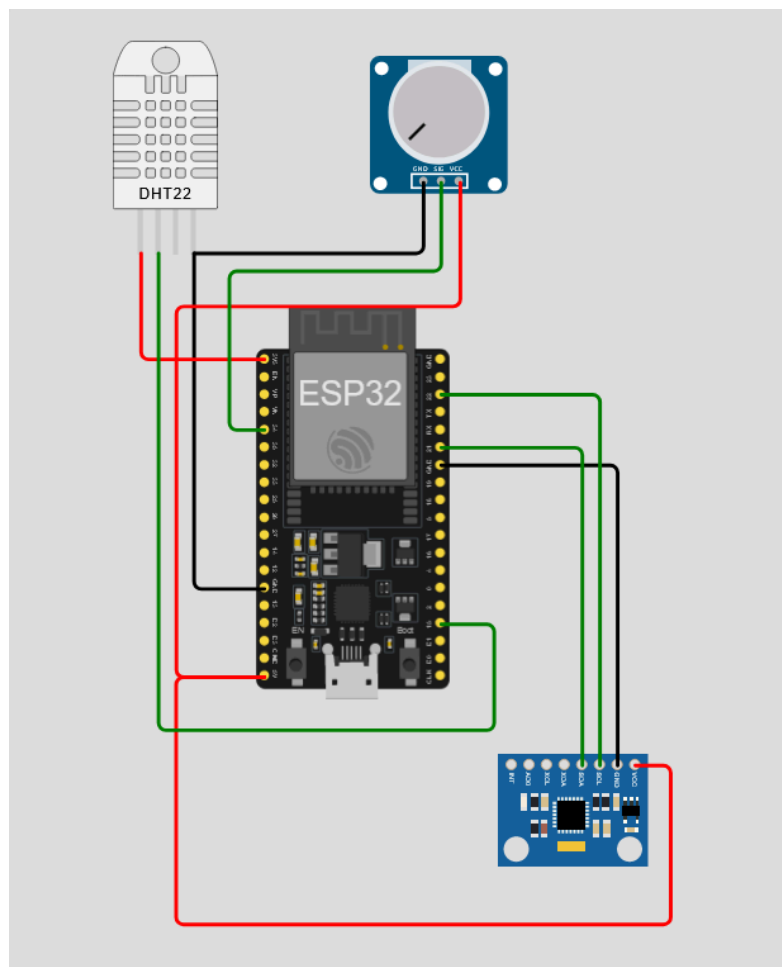


Figure 01: Wokwi simulation

Figure 01 shows the integration of the sensors on the microcontroller board.

Wireless network (MQTT)

Why MQTT?

As for the networking protocol we chose MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a lightweight and efficient data transfer protocol, which conserves both bandwidth and power. Thus, making it ideal for wearable IoT devices with limited battery life. It enables reliable and real-time communication, essential for transmitting vital health data from wearable devices to monitoring systems.

On the wokwi simulator, we used the [WiFi.h](#) and [PubSubClient.h](#) libraries to establish a Wi-Fi connection and handle MQTT communication for ESP32.

Here's how Wokwi allows MQTT connection with ESP32:

1. **Wi-Fi Connection:** The ESP32 was set to connect to a Wi-Fi network using the credentials provided ([ssid](#) and [password](#)). When simulating projects on Wokwi, it allows the simulated devices to connect to a virtual Wi-Fi network named "Wokwi-GUEST". This simulated network doesn't require a password and it mimics the behavior of a real Wi-Fi network.
2. **MQTT Server Connection:** The ESP32 then connects to an MQTT broker, in this case, [test.mosquitto.org](#). This is a public MQTT broker that allows devices to publish and subscribe to topics.

The code snippet for MQTT connection:

```
const char* ssid = "Wokwi-GUEST"; // wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org"; // mosquitto server url

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.println("Connecting to");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
}
```

Figure 02: Network connection code snippet

While running the project the sensors data were published to particular topics.

```
client.publish("/EHIoT/temp", temp.c_str()); // publish temp topic /EHIoT/temp
client.publish("/EHIoT/acceleration/x", xVal.c_str());
client.publish("/EHIoT/acceleration/y", yVal.c_str());
client.publish("/EHIoT/acceleration/z", zVal.c_str());
client.publish("/EHIoT/heartbeat", hbVal.c_str());
client.publish("/EHIoT/pid", pidString);
```

Figure 03: Data publish code snippet

Node-RED

We have used Node-RED to visualize the data flow as well as doing some data processing before storing them in InfluxDB.

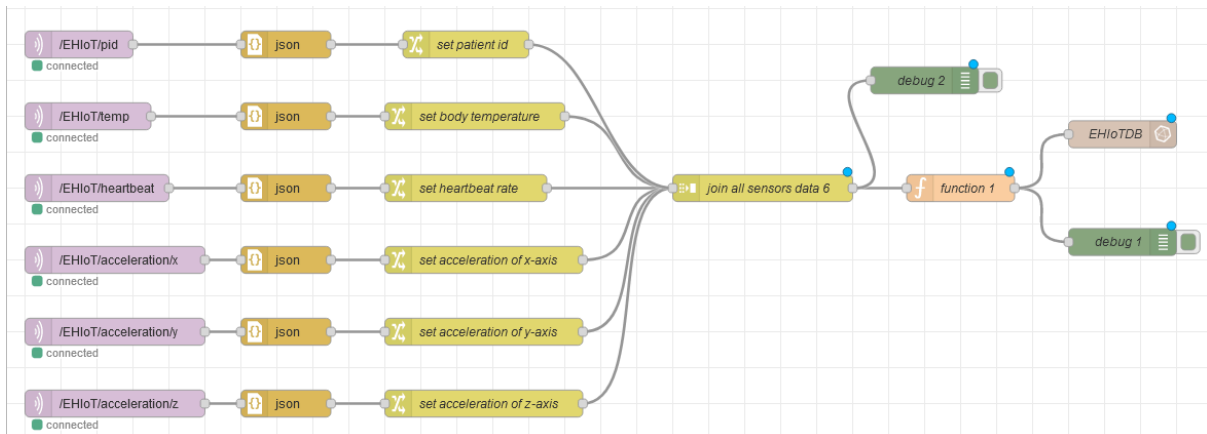


Figure 04: Node-RED complete flow

Figure 04 here represents the complete data flow of the Adult Patient Monitoring system.

We have configured the MQTT-in nodes to connect it to the MQTT broker (test.mosquitto.org) and set the respective topics. The port used here is 1883.

Edit mqtt in node

Delete
Cancel
Done

Properties

Server
test.mosquitto.org:1883

Action
Subscribe to single topic

Topic
/EHIoT/pid

QoS
2

Output
auto-detect (parsed JSON object, string or buffer)

Name
Name

Figure 05: MQTT connection

The data were initially published to six different topics and were later combined into one single object before storing in the database. Debug nodes were used to monitor the processed data.

InfluxDB

Our Elderly Patient Monitoring System uses InfluxDB as the database to store time-series data produced by the wearable device's sensors. Because InfluxDB is specifically built to handle time-stamped data efficiently, this high-performance, distributed, and scalable database is the perfect option for applications where real-time monitoring is crucial.

Edit influxdb out node

Delete Cancel Done

Properties

Name: EHIoTDB

Server: [v2.0] EHIoT

Organization: UNIVAQ

Bucket: Data_EHIOT

Measurement: Patients_Data

Time Precision: Seconds (s)

Figure 06: InfluxDB node configuration on Node-RED

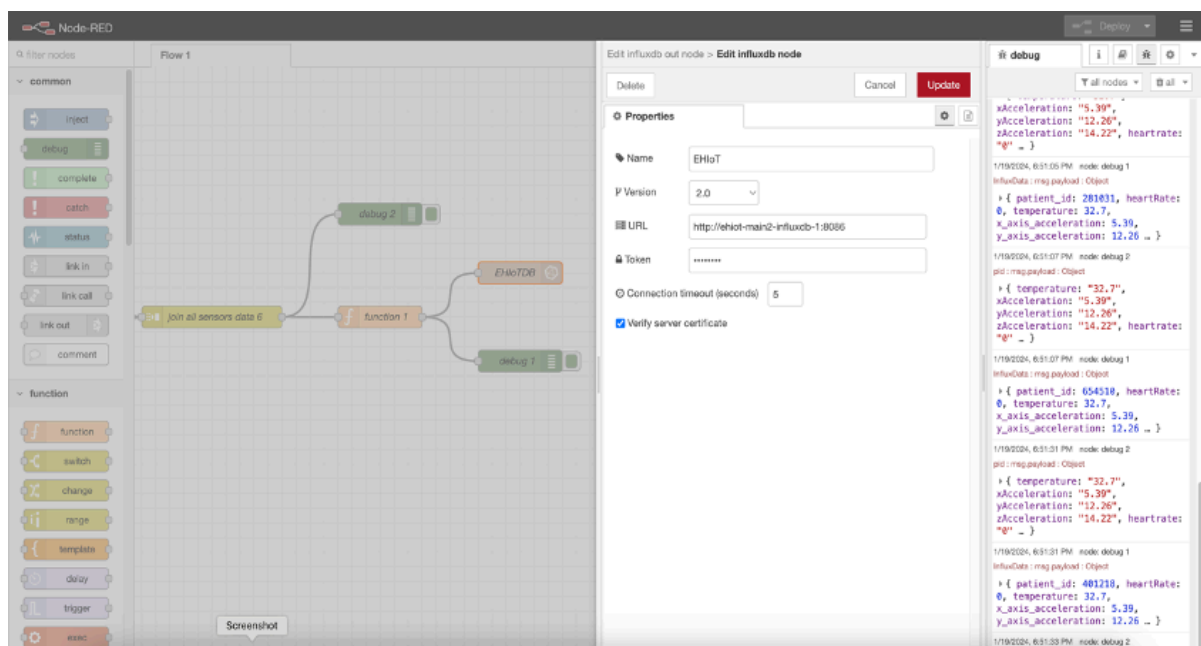


Figure 07: Influxdb connection

The data flow starts with the MQTT protocol-powered simulated Elderly Patient Monitoring device publishing sensor data across the wireless network. Vital health data is carried via wireless connection, which is received by Node-RED. Node-RED then serves as an intermediate, processing and transferring the data to InfluxDB. The port used here is 8086.

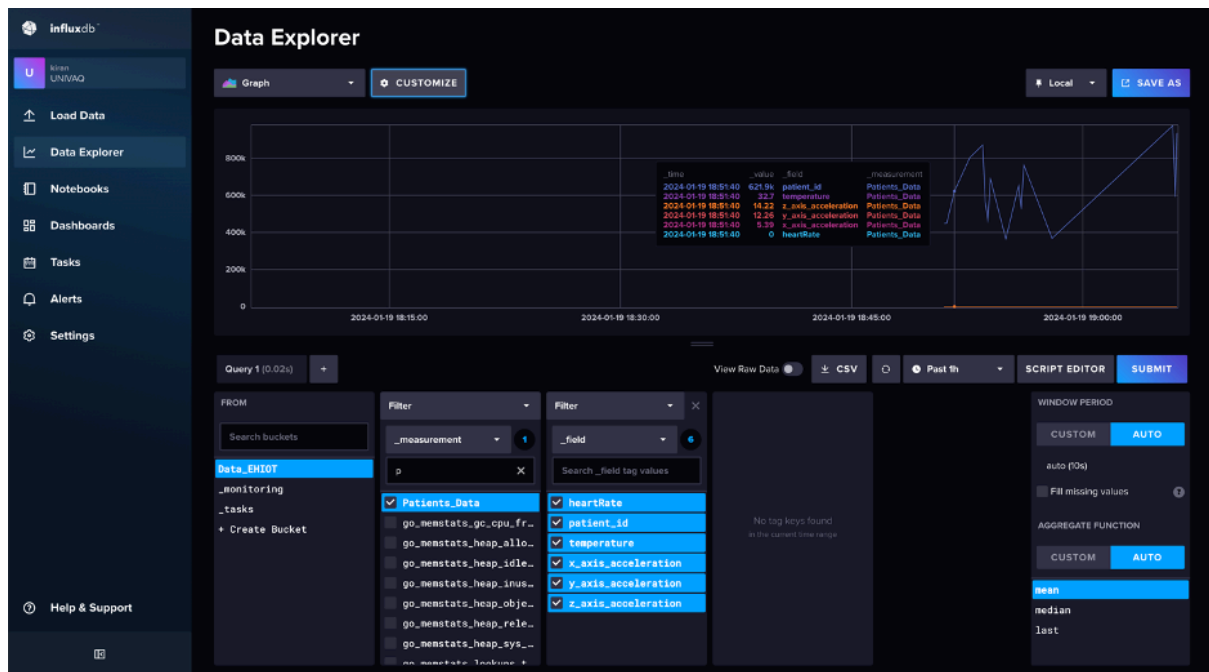


Figure 08: Influxdb dashboard

Figure 08 provides a view of our InfluxDB dashboard displaying real-time patient health data that is integrated via Node-RED as provided above.


This visual depiction gives caregivers and healthcare professionals a fast overview of the extensive dataset housed in InfluxDB and offers actionable insights.

Grafana

Data about the patient being monitored is visualized in the form of a dashboard through various graphs and charts. For this visualization, Grafana is being utilized because its highly customizable and intuitive interface and advanced querying capabilities. These features make it easy to identify patterns and trends as well as providing the options for alert and notifications.

Data Source

First and foremost step in building a dashboard is configuring a data source which will provide us data to perform analytics.

 influxdb

Type
InfluxDB

Alerting
Supported

Explore data

Build a dashboard

Type: InfluxDB

⚙ Settings

Name ⓘ influxdb

Default ☒

Query language

InfluxQL ▾

ⓘ


Please report any issues to:
<https://github.com/grafana/grafana/issues>

HTTP

URL ⓘ	http://ehiot-influxdb-1:8086	
Allowed cookies ⓘ	New tag (enter key to add)	Add
Timeout ⓘ	Timeout in seconds	

Figure 09: Setting Up Data Source - I

InfluxDB Details



Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.




Database	<input type="text" value="Data_EHIOT"/>	
User	<input type="text" value="admin"/>	
Password	<input type="password" value="configured"/>	<input type="button" value="Reset"/>
HTTP Method	<div> Choose</div>	
Min time interval	<div> 10s</div>	
Max series	<div> 1000</div>	

Figure 10: Setting Up Data Source - II

Upon configuring the data source, proceed to the dashboards section to create a new one. Within the newly established dashboard, generate visualizations using specific queries to facilitate analytics in the form of charts and graphs.

Dashboard



Figure 11: Grafana dashboard visualizing patient data

Figure 11 displays the different visualization for different metrics of patient data.

Alert

After Visualizations we added some alerts by setting up the threshold values.

The alarm thresholds are:

1. Heart rate: the normal range is between 70 to 140.
2. Temperature: From 35 to 38 considering both possibilities of hypothermia and fever.
3. Acceleration: We are not sure about the simulated sensor and the thresholds are not completely realistic. We assumed the value ranges for x-axis (-2 to 2), y-axis (-1 to 1), z-axis (-1 to 1).

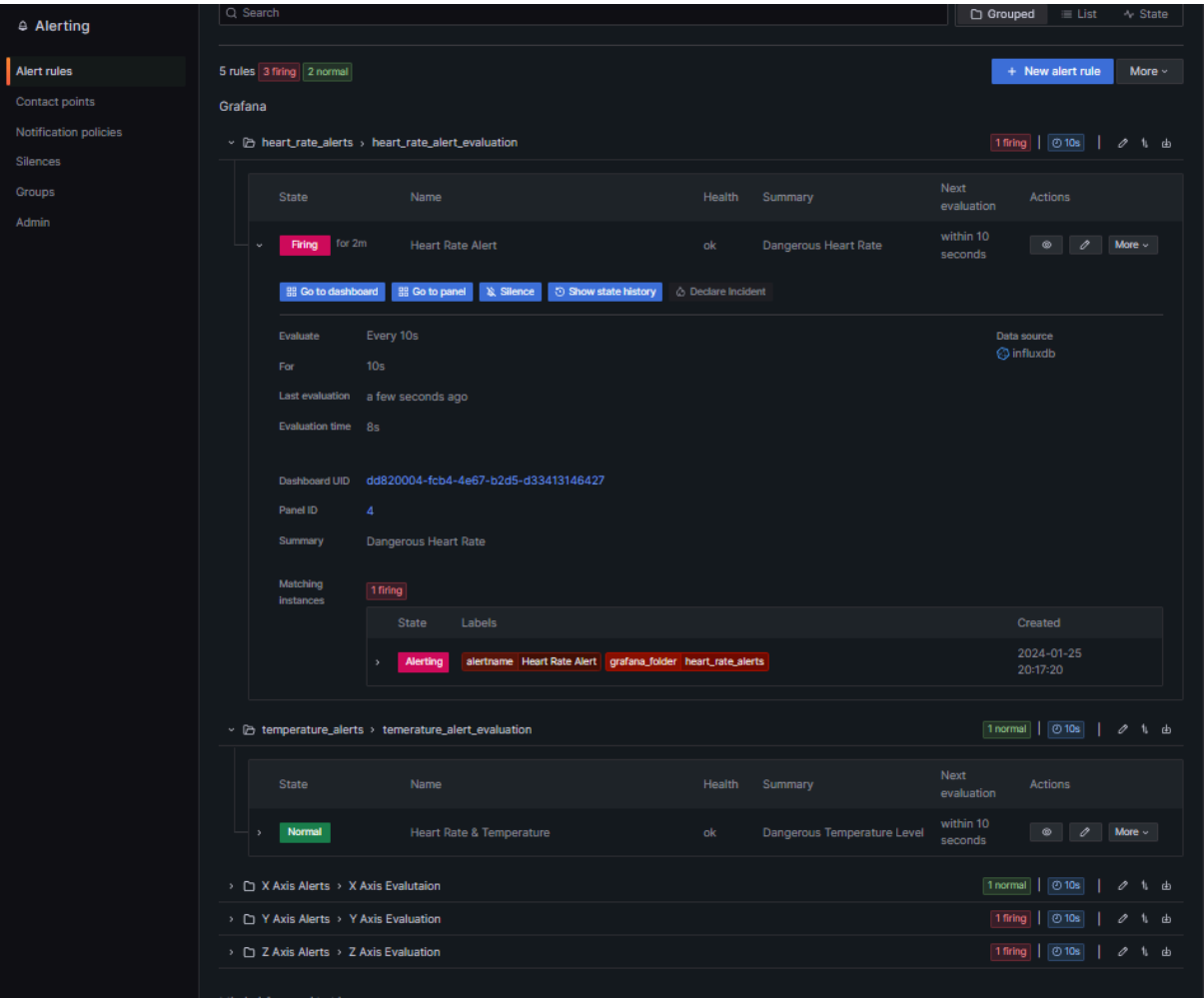


Figure 12: Grafana Alerts

Docker

The complete project has been dockerized.



Figure 13: Docker containers

Conclusion

This Elderly Patient Monitoring System promises to enhance the care and safety of elderly patients through advanced technology. By utilizing MQTT, Node-RED, Docker, InfluxDB, Grafana, this system ensures comprehensive monitoring with real-time data processing and visualization, contributing significantly to the well-being of elderly patients.

Project set up

The steps to set up the project:

1. Clone the project from the GitHub link: <https://github.com/nawshin81/EHIoT>
2. Run Docker Desktop
3. Go to the terminal and run the commands “docker-compose build” and “docker-compose up”
4. Go to the [Wokwi simulation link](#) to run the data simulation. Please keep in mind, switching over to another tab/ screen pauses the simulation. So, keep the simulation tab open on a divided screen/ monitor.
5. Visit <http://localhost:1880> to see the Node-RED flow.
6. Visit <http://localhost:8086> to access InfluxDB.
7. In the InfluxDB login interface input the username and password for authentication.
 - Username: admin
 - Password: password
8. Visit <http://localhost:3000> to access Grafana.
9. In the Grafana login interface input the username and password for authentication.
 - Username: admin
 - Password: password
10. Navigate to the dashboard and open the “Patient Monitoring Dashboard”.