Università della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

TELECOMMUNICATION ENGINEERING

Distributed Systems & Cloud/Edge Computing

"Final Project Report"

Professor

Prof. Claudio Savaglio

Students:

Mahmoud Mohamed (252048)

Mahmoud Ezzat Helal (235580)

Anno Accademico 2023 /2024

1 Introduction

The need for effective, real-time health monitoring systems is growing in the current healthcare environment. By utilizing the latest developments in Internet of Things (IoT) technologies, these systems provide the ability to monitor critical health indicators, guarantee prompt medical attention, and improve patient care in general. This project offers a comprehensive smart health monitoring system that continuously tracks three different health indicators, by combining contemporary IoT devices with cloud-based services.

Three different sensors were used which are: Pulse sensor, Temperature Sensor and Gas Sensor. These sensors were connected to an ESP32 microcontroller, which has the responsibility to gather critical health data and publish them, through MQTT, to AWS IoT Core. Through a lambda function, the data received in IoT Core will be treated with different other AWS services, such as: DynamoDB, SNS (Simple Notification Service), SES (Simple Email Service).

The user interface for this system is implemented by using Node-Red Dashboard. which was installed on an EC2 instance. Once the user joins the dashboard by using the provided credential, he can monitor continuously the different parameters measured by the system.

The objectives of this project can be summarized as follows:

1. Comprehensive Health Monitoring;
2. Reliable Data Storage;
3. Proactive Threshold Alerts;
4. Health Parameters Reports.

2 Analysis

The smart object is designed to be assigned to a specific person, to measure health parameters, such as heart rate and temperature, and external parameters that may affect the user health such as air quality. The system leverages a combination of several devices and cloud services to provide continuous, real-time monitoring of vital health metrics, giving the possibility for timely medical intervention and enhanced patient care.

The functional requirements of this project can be organized in several points:

- Health Monitoring: The system continuously monitors heart rate, temperature, and gas levels, providing real-time data visualization on the Node-RED dashboard.
- Data Transmission: MQTT Protocol that Ensures reliable and efficient communication between the ESP32, AWS IoT Core and Node-RED.

- Data Storage: Sensor data is securely stored in DynamoDB, ensuring reliable data logging and easy retrieval for historical analysis and reporting.
- Threshold Alerts: AWS SNS is triggered to send email or message alerts when sensor readings exceed predefined thresholds, ensuring prompt notification of potential health risks.
- Periodic and On-Demand Reporting: a CSV Reports is sent periodically (according to the periodicity defined at the implementation time), containing the stored data and their averages. Also, the user can request, through the dashboard a customized report related to a specific period by specifying the start and end dates.

The non-functional requirements of the system are:

- System must provide real-time monitoring with **low latency** by ensuring timely data updates.
- System must be always **available** to monitor continuously the health parameters.
- Dashboard must be **accessible** only to authorized users by using the provided credentials.
- Notifications must provide **clear** and **Accurate** information to the user.

Figure (1) illustrates the meta-model of the implemented project.

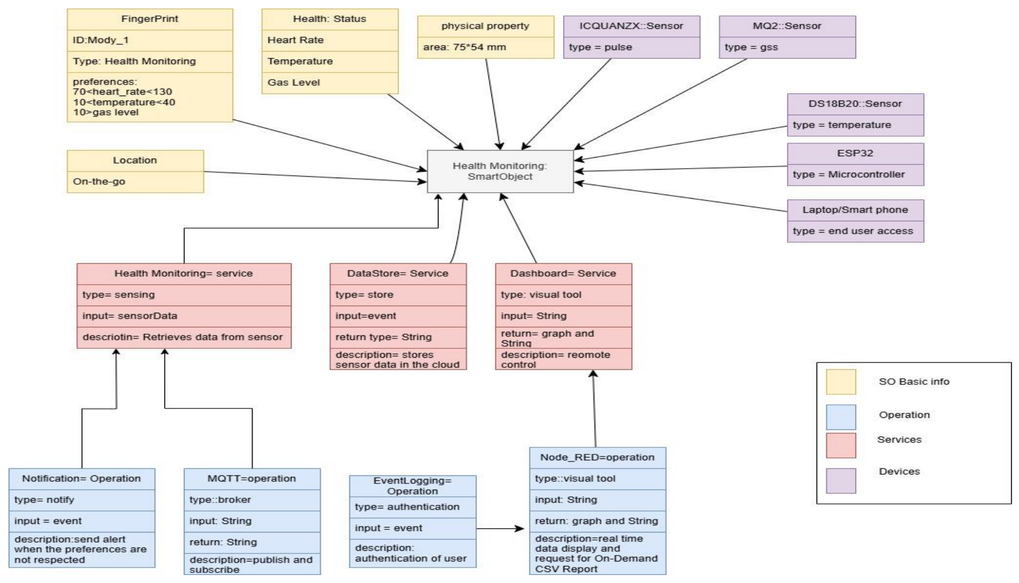


Figure 1: Analysis Model.

3 Design

The system is based on the use of broker-based communication with MQTT protocol for efficient, low-latency data transfer. The design of the system involves both: Hardware and Software components.

The hardware design choices are listed below:

- ESP32 Microcontroller: represents the central component;
- ICQUANZX Sensor: is optical heart rate sensor characterized by its suitability for embedding in wearable devices and health related electronics projects. The output of this sensor is analog;
- DS18B20 Sensor: is digital temperature sensor characterized by high accuracy, high resolution and lower response time;
- MQ-2 sensor: used to detect the presence of gas and smoke in the surrounding area. It has adjustable sensitivity through a potentiometer.

The software design choices:

- AWS IoT Core: used to connect devices to AWS IoT allowing them to securely communicate and interact with AWS IoT cloud Services;
- DynamoDB: used to store data. It is characterized by on-demand capacity mode which offers pay-as-you-go pricing;
- Lambda Function: used to run codes allowing the interaction between more services of AWS;
- EventBridge: used to generate events according to a predefined periodicity, that can trigger other services;
- SNS and SES: used to enable email and SMS notifications;
- EC2: provides on-demand, scalable computing capacity, with the goal of allowing the development, the deployment and the scaling of applications;
- Node-Red: used to create a user interface.

the several design choices are listed in the table (1).

Requirement	Design Solution
Real-time monitoring and control	Node-RED Dashboard on EC2 Instance
Lightweight communication protocol	Broker-based protocol (AWS IoT Core)
Notification System	System based on e-mail (AWS SNS, AWS SES)
Data Availability	Cloud-based Data Storage
Interoperability among different services	Permission allowed through AWS IAM
Cost efficiency and flexibility	Open Source IoT Protocols and Tools
Remote access and control	Dashboard Access via URL with Credentials
Privacy and security	Data Encryption with TLS, Local Authentication

Table 1: design choices.

4 Implementation

The implementation phase involved several steps, which are:

1. Hardware Implementation;
2. Software Implementation;
3. Testing and Deployment;

4.1 Hardware Implementation

The components used in the hardware implementation phase are illustrated in the table (2).

Model	Kind	Function	Board	GPIO Pins
ICQUANZX	Sensor	Measure heart rate	ESP32 Arduino	34
DS18B20	Sensor	Measure temperature	ESP32 Arduino	17
MQ-2	Sensor	Detect gas levels	ESP32 Arduino	35

Table 2: Hardware Implementation Choices.

The core element of the hardware implementation is the ESP32, such microcontroller was chosen for its powerful processing capabilities, built-in Wi-Fi, and its versatility in IoT applications. The ESP32 was programmed to read data from the sensors, and configured to send this data to AWS IoT Core and Node-Red. The figure (2) shows the circuit connections.

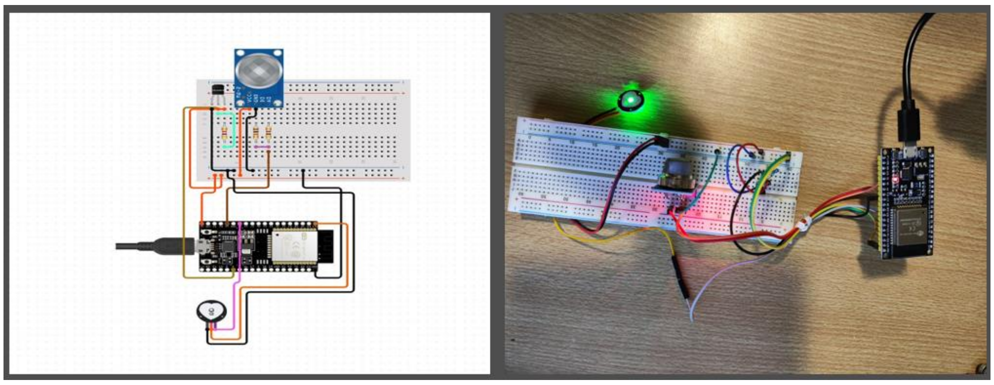


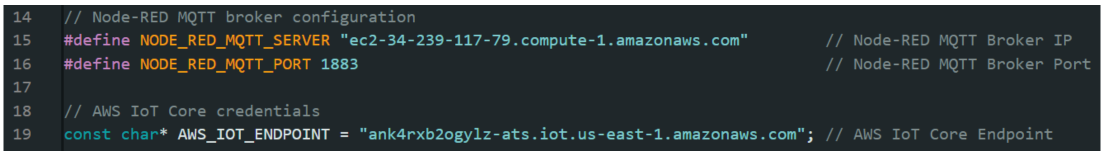
Figure 2: Circuit Implementation.

4.2 Software Implementation

The software implementation could be divided into three main topics which are:

4.2.1 MQTT Communication

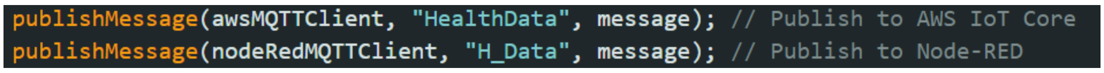
The MQTT is lightweight protocol able to implement a reliable communication. The ESP32 was programmed using the Arduino IDE to publish the data to AWS IoT Core and to Node-Red by using MQTT. The figure (3) illustrates the lines code related to such connection. While the figure (4) illustrates the topics on which the data is published.



```
// Node-RED MQTT broker configuration
#define NODE_RED_MQTT_SERVER "ec2-34-239-117-79.compute-1.amazonaws.com"        // Node-RED MQTT Broker IP
#define NODE_RED_MQTT_PORT 1883                                                   // Node-RED MQTT Broker Port

// AWS IoT Core credentials
const char* AWS_IOT_ENDPOINT = "ank4rxb2ogylz-ats.iot.us-east-1.amazonaws.com"; // AWS IoT Core Endpoint
```

Figure 3: MQTT Configuration.



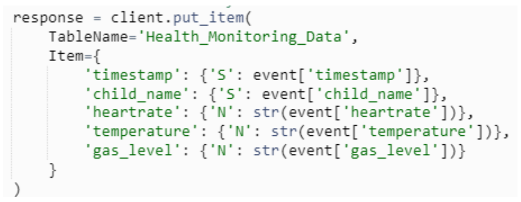
```
publishMessage(awsMQTTClient, "HealthData", message); // Publish to AWS IoT Core
publishMessage(nodeRedMQTTClient, "H_Data", message); // Publish to Node-RED
```

Figure 4: MQTT Topics.

4.2.2 AWS Services

Several AWS services were used in the implementation. The services are listed below:

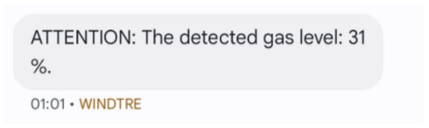
- IoT Core: a thing was created, and necessary certificates were generated and uploaded to the ESP32 for secure Communication;
- Lambda Function: two functions were implemented, where the first one is triggered by the IoT Core when new data arrives, and it has the responsibility to process the data in order to store it in the DynamoDB and to get the user notified when the data exceeds the predefined thresholds. While the second function has the task of creating: reports according to a specified periodicity, and On-Demand reports according to the start and end dates established by the user. The figure (5) illustrates the lines code used to put an item in the DynamoDB in the case of the first lambda function;



```
response = client.put_item(
    TableName='Health_Monitoring_Data',
    Item={
        'timestamp': {'S': event['timestamp']},
        'child_name': {'S': event['child_name']},
        'heartrate': {'N': str(event['heartrate'])},
        'temperature': {'N': str(event['temperature'])},
        'gas_level': {'N': str(event['gas_level'])}
    }
)
```

Figure 5: Item insertion in the DynamoDB.

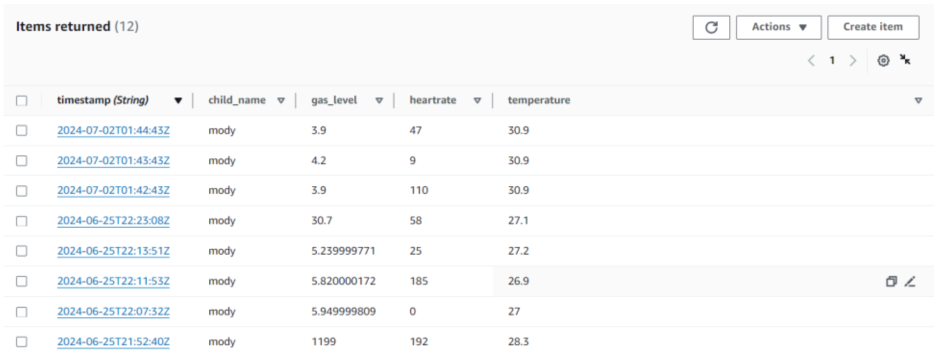
- SNS: when it is triggered by the lambda function, it sends an email/SMS message if needed to alert the user. The figure (6) shows the received SMS when a threshold is exceeded;


ATTENTION: The detected gas level: 31 %.

01:01 • WINDTRE

Figure 6: SMS Notification.

- EventBridge: used to establish the periodicity of the reports. And it is the element that triggers the second lambda function.
- SES: used to send the generated CSV reports to the users via email.
- DynamoDB: one table was created to store the data processed by lambda. The table includes the attributes shown in the figure (7);


Items returned (12)
Actions
Create item
1

timestamp (String)	child_name	gas_level	heartrate	temperature
2024-07-02T01:44:43Z	mody	3.9	47	30.9
2024-07-02T01:43:43Z	mody	4.2	9	30.9
2024-07-02T01:42:43Z	mody	3.9	110	30.9
2024-06-25T22:23:08Z	mody	30.7	58	27.1
2024-06-25T22:13:51Z	mody	5.239999771	25	27.2
2024-06-25T22:11:53Z	mody	5.820000172	185	26.9
2024-06-25T22:07:32Z	mody	5.949999809	0	27
2024-06-25T21:52:40Z	mody	1199	192	28.3

Figure 7: DynamoDB Table

- EC2: the Node-Red was installed on EC2 instance to achieve higher flexibility, higher availability and to allow remote access.

4.2.3 Node-Red

Node-Red flow is created to receive the data from the ESP32 and to display them on the dashboard providing a user-friendly interface to enable the real-time monitoring and to give the possibility to the user to request an On-Demand Reports by selecting the dates. The figure (8) illustrates the dashboard.

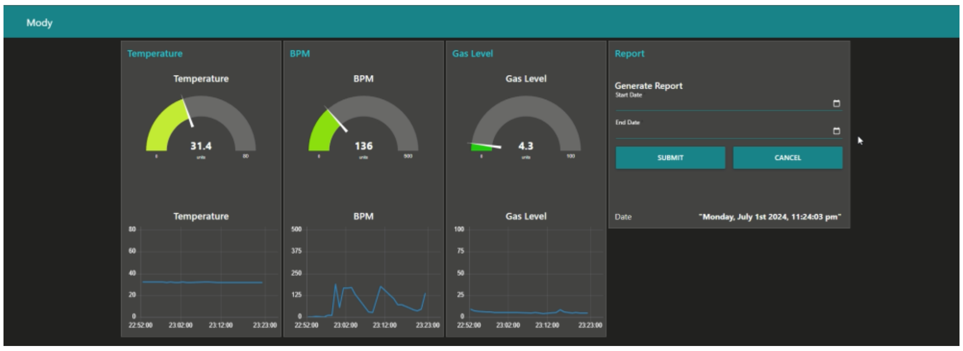
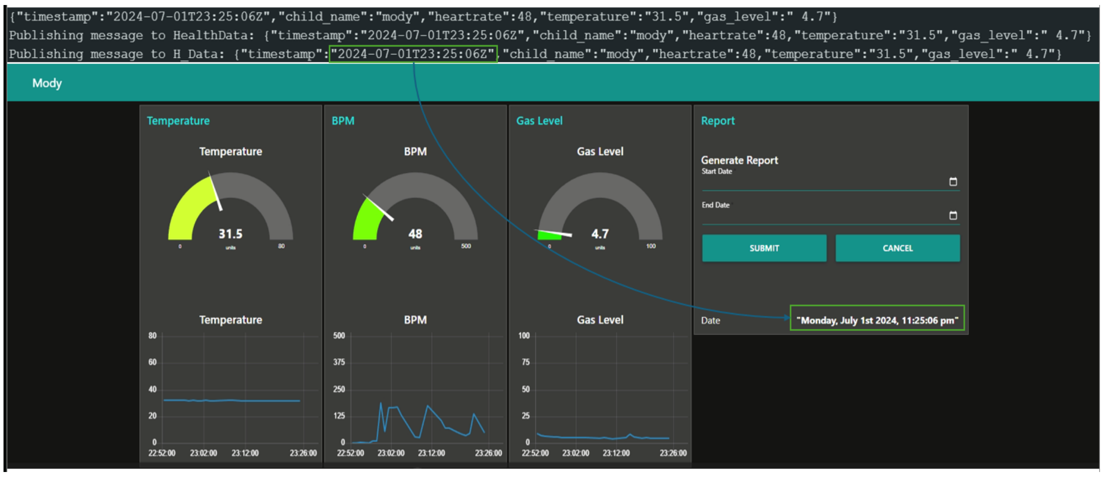


Figure 8: User Interface.

4.3 Testing

Each component was tested individually and then integrated into the complete system. Some sensor readings are converted to provide to the user a simple digital numeric value as in the case of pulse rate sensor. The dashboard’s real-time monitoring capabilities were checked as shown in the figure (9).



```
{"timestamp":"2024-07-01T23:25:06Z","child_name":"mody","heartrate":48,"temperature":"31.5","gas_level":" 4.7"}
Publishing message to HealthData: {"timestamp":"2024-07-01T23:25:06Z","child_name":"mody","heartrate":48,"temperature":"31.5","gas_level":" 4.7"}
Publishing message to H_Data: {"timestamp":"2024-07-01T23:25:06Z","child_name":"mody","heartrate":48,"temperature":"31.5","gas_level":" 4.7"}
```

Figure 9: Real-Time Monitoring.

The completed implementation of the project is illustrated in figure (10).

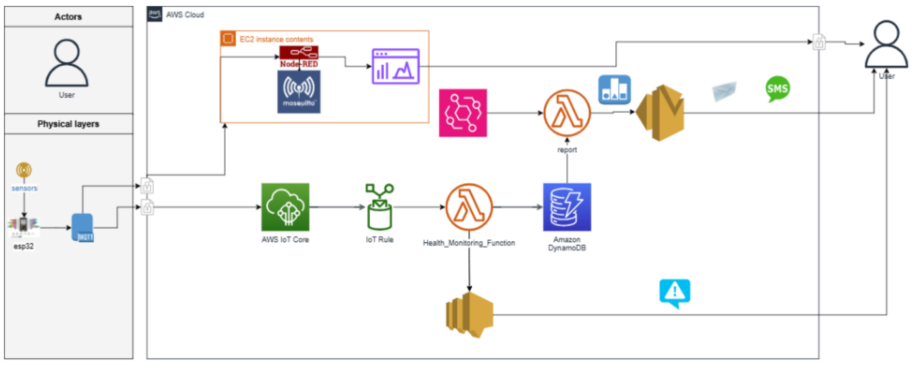
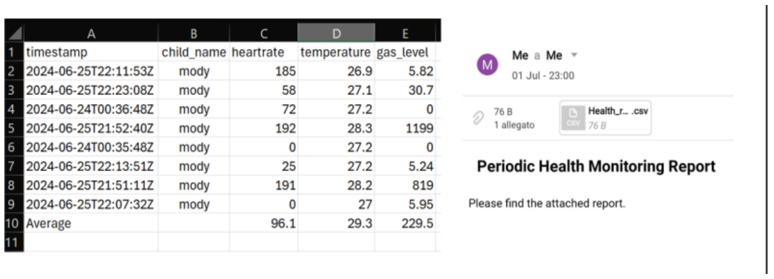


Figure 10: Complete Overview of the project.

The figure (11) illustrates the periodic reports received via email.



	A	B	C	D	E
1	timestamp	child_name	heartrate	temperature	gas_level
2	2024-06-25T22:11:53Z	mody	185	26.9	5.82
3	2024-06-25T22:23:08Z	mody	58	27.1	30.7
4	2024-06-24T00:36:48Z	mody	72	27.2	0
5	2024-06-25T21:52:40Z	mody	192	28.3	1199
6	2024-06-24T00:35:48Z	mody	0	27.2	0
7	2024-06-25T22:13:51Z	mody	25	27.2	5.24
8	2024-06-25T21:51:11Z	mody	191	28.2	819
9	2024-06-25T22:07:32Z	mody	0	27	5.95
10	Average		96.1	29.3	229.5
11					

Me a Me
01 Jul - 23:00
76 B
1 allegato
Health_r... .csv
76 B

Periodic Health Monitoring Report

Please find the attached report.

Figure 11: Periodic Report.

5 Conclusion

This project illustrates an example of how the integration of IoT technology into healthcare could improve remote health monitoring capabilities. By using several AWS services, the system ensures reliable data storage, real-time monitoring, and timely alerts. Future enhancements could include adding more sensor types, such as GPS module to add some tracking capabilities, and incorporating ML algorithms for predictive health analytics.