



SOFE 4610U: Internet of Things

Project Report

Group 12:

Khalil Zayed (100672228)

Fajer Zayed (100672347)

Yakhneshan Sivaperuman (100703400)

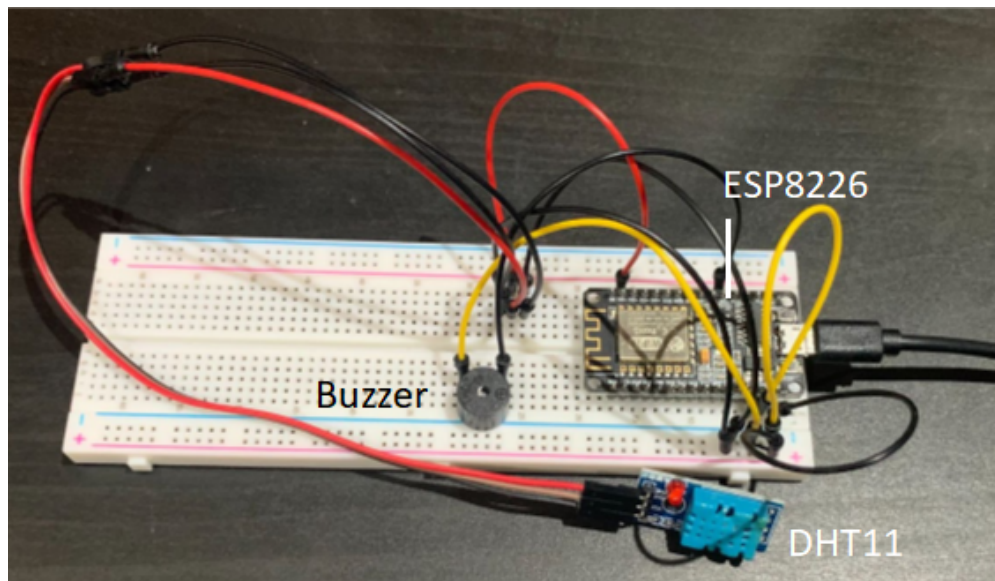
The fire alarm system design consists of three major components, the client, sensor and the IoT server. The design will apply a pub-sub architecture where the sensor continuously reads information from the real-world and publishes it to the central server where it is stored. The client subscribes to the central server to receive this information in real-time. All communication between the components is done over the internet.

Sensor

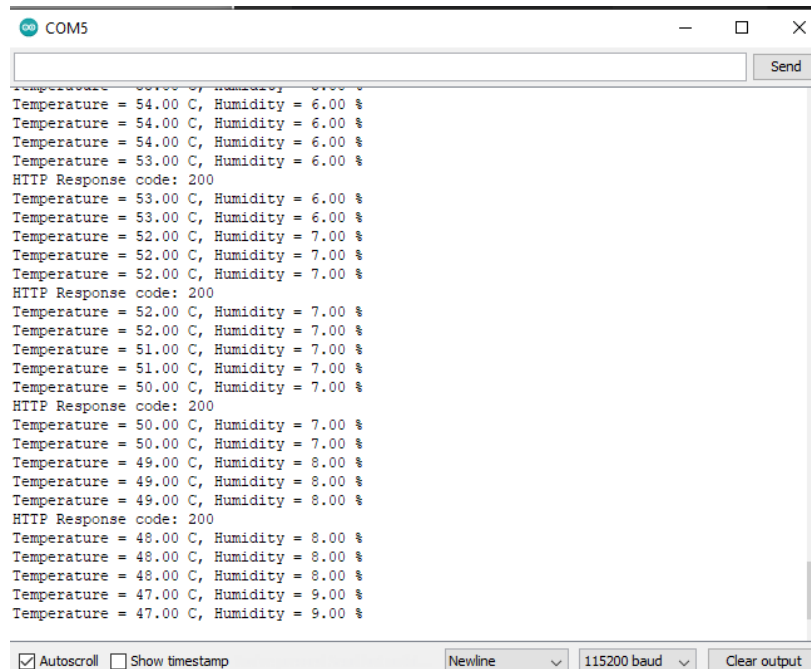
The sensor consists of several components working together to achieve its purpose:

- ESP8226 (microcontroller)
- DHT11 (temperature sensor)
- Buzzer (alarm sound)

Using the DHT11, temperature information is gathered periodically and processed by the ESP8226. The microcontroller will assess the temperature readings and determine whether to turn on the fire alarm or not. It is set to turn on the alarm if the temperature goes above 30 C. Moreover, every reading along with the fire alarm status are published to the central server using HTTP POST requests.



Sensor hardware components



```
COM5
Temperature = 54.00 C, Humidity = 6.00 %
Temperature = 54.00 C, Humidity = 6.00 %
Temperature = 54.00 C, Humidity = 6.00 %
Temperature = 53.00 C, Humidity = 6.00 %
HTTP Response code: 200
Temperature = 53.00 C, Humidity = 6.00 %
Temperature = 53.00 C, Humidity = 6.00 %
Temperature = 52.00 C, Humidity = 7.00 %
Temperature = 52.00 C, Humidity = 7.00 %
Temperature = 52.00 C, Humidity = 7.00 %
HTTP Response code: 200
Temperature = 52.00 C, Humidity = 7.00 %
Temperature = 52.00 C, Humidity = 7.00 %
Temperature = 51.00 C, Humidity = 7.00 %
Temperature = 51.00 C, Humidity = 7.00 %
Temperature = 50.00 C, Humidity = 7.00 %
HTTP Response code: 200
Temperature = 50.00 C, Humidity = 7.00 %
Temperature = 50.00 C, Humidity = 7.00 %
Temperature = 49.00 C, Humidity = 8.00 %
Temperature = 49.00 C, Humidity = 8.00 %
Temperature = 49.00 C, Humidity = 8.00 %
HTTP Response code: 200
Temperature = 48.00 C, Humidity = 8.00 %
Temperature = 48.00 C, Humidity = 8.00 %
Temperature = 48.00 C, Humidity = 8.00 %
Temperature = 47.00 C, Humidity = 9.00 %
Temperature = 47.00 C, Humidity = 9.00 %
```

☒ Autocroll ☐ Show timestamp Newline 115200 baud Clear output

Sensor console showing readings and successful HTTP POST request

Server

The central server consists of a RESTful API that the publishers and subscribers can utilize to exchange information. Temperature and fire alarm information is received from the sensors and stored in a local database. Clients that are subscribed to the server will be able to access this information through HTTP GET requests. The server is designed to simultaneously update temperature readings from the sensor and process client requests.

```

root@ubuntu: /var/www/iot-platform.ca/html
Got body: { temperature: '22.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '22.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '32.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '33.00', fireAlarm: 'false' }
Got body: { temperature: '22.00', humidity: '35.00', fireAlarm: 'false' }
Got body: { temperature: '22.00', humidity: '36.00', fireAlarm: 'false' }
Got body: { temperature: '22.00', humidity: '41.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '40.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '40.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '41.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '40.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '40.00', fireAlarm: 'false' }
Got body: { temperature: '23.00', humidity: '41.00', fireAlarm: 'false' }
Got body: { temperature: '24.00', humidity: '46.00', fireAlarm: 'false' }
Got body: { temperature: '28.00', humidity: '71.00', fireAlarm: 'false' }
Got body: { temperature: '34.00', humidity: '55.00', fireAlarm: 'false' }
Got body: { temperature: '42.00', humidity: '34.00', fireAlarm: 'false' }
Got body: { temperature: '48.00', humidity: '22.00', fireAlarm: 'false' }
Got body: { temperature: '52.00', humidity: '14.00', fireAlarm: 'true' }
Got body: { temperature: '54.00', humidity: '9.00', fireAlarm: 'true' }
Got body: { temperature: '55.00', humidity: '7.00', fireAlarm: 'true' }
Got body: { temperature: '55.00', humidity: '6.00', fireAlarm: 'true' }
Got body: { temperature: '53.00', humidity: '6.00', fireAlarm: 'true' }
Got body: { temperature: '52.00', humidity: '7.00', fireAlarm: 'true' }
Got body: { temperature: '50.00', humidity: '7.00', fireAlarm: 'false' }
Got body: { temperature: '49.00', humidity: '8.00', fireAlarm: 'false' }
Got body: { temperature: '47.00', humidity: '9.00', fireAlarm: 'false' }

```

Server side console receiving information from sensor

Client

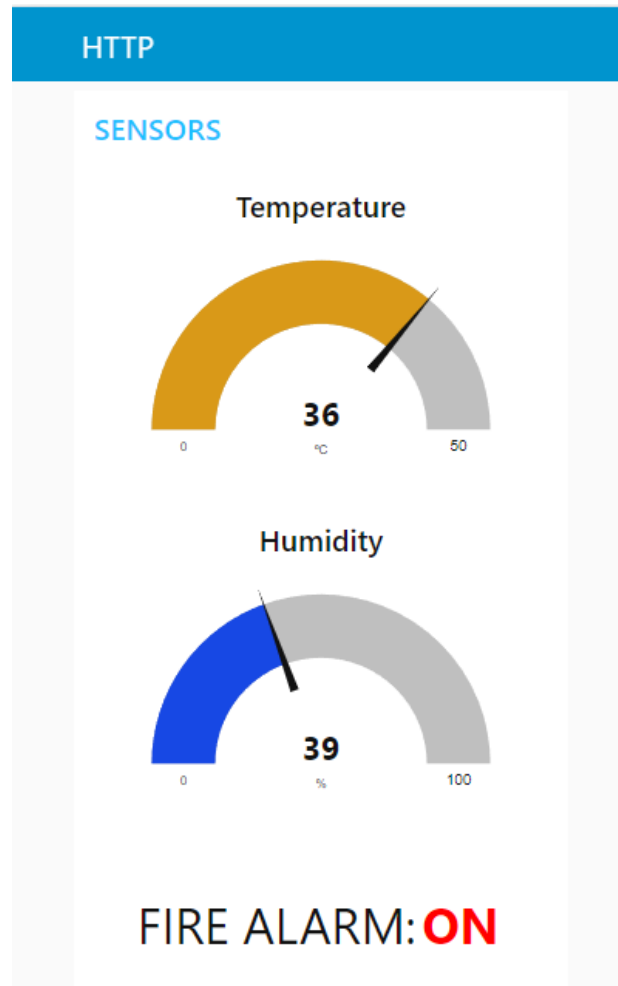
The client consists of a Node-RED interface that will periodically run GET requests to the backend server to update the temperature and fire alarm information. This information will arrive in the form of JSON objects that will need to be processed before displaying it on the screen for the user. The interface consists of Temperature Gauge, Humidity Gauge, and fire alarm status.

```

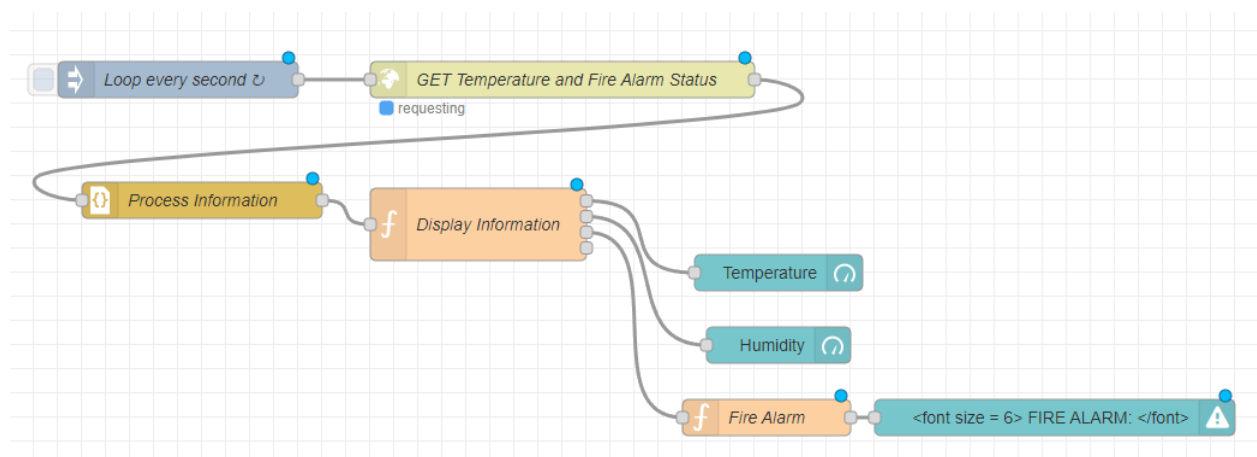
{ temperature: '52.00', humidity: '7.00', fireAlarm: 'true' }
Status Code: 200
Date in Response header: Mon, 29 Nov 2021 02:27:47 GMT
object
Response ended:
{ temperature: '50.00', humidity: '7.00', fireAlarm: 'false' }
Status Code: 200
Date in Response header: Mon, 29 Nov 2021 02:27:52 GMT
object
Response ended:
{ temperature: '49.00', humidity: '8.00', fireAlarm: 'false' }

```

Client side console showing GET response body



Node-RED client interface

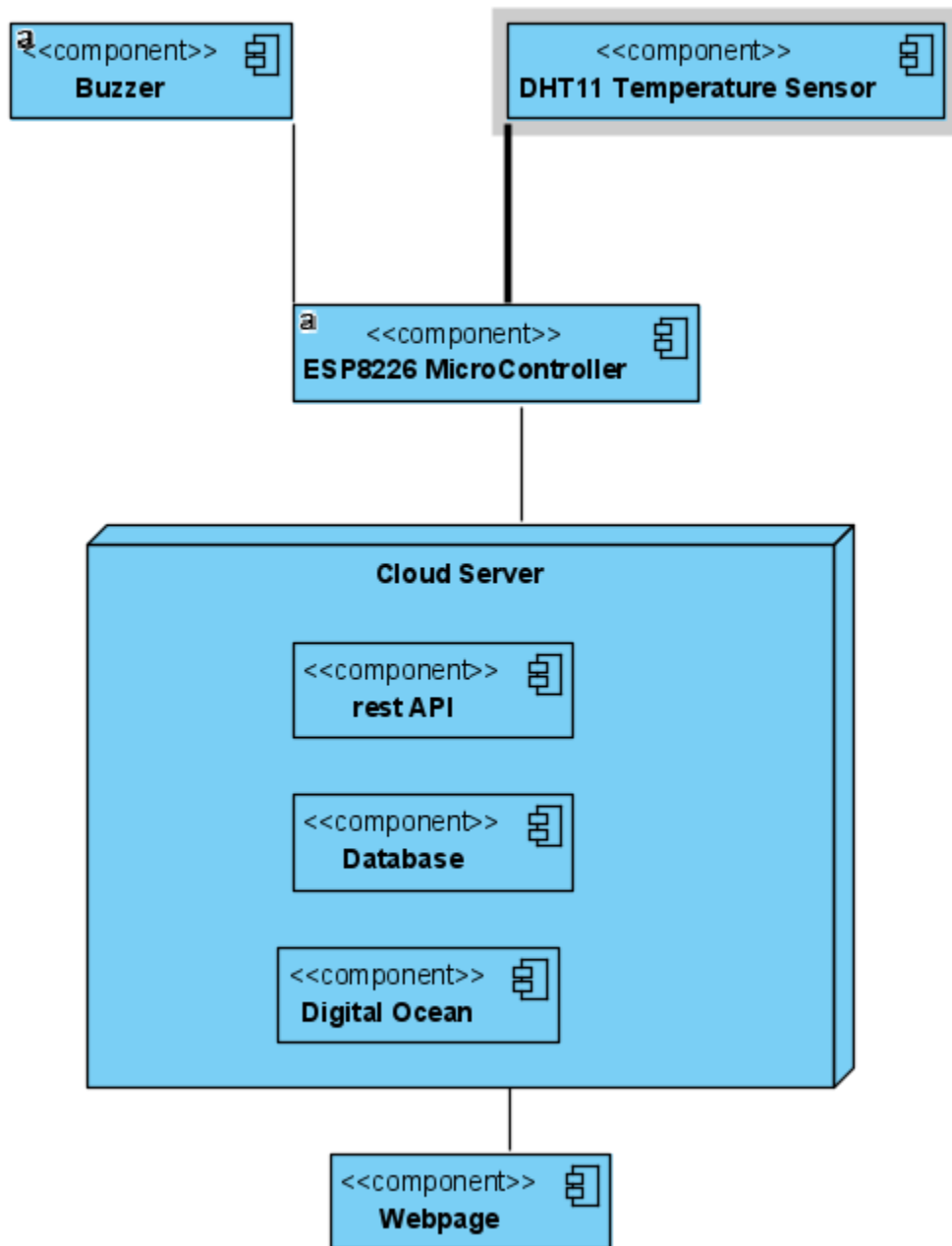


Node-RED interface design and flow of information

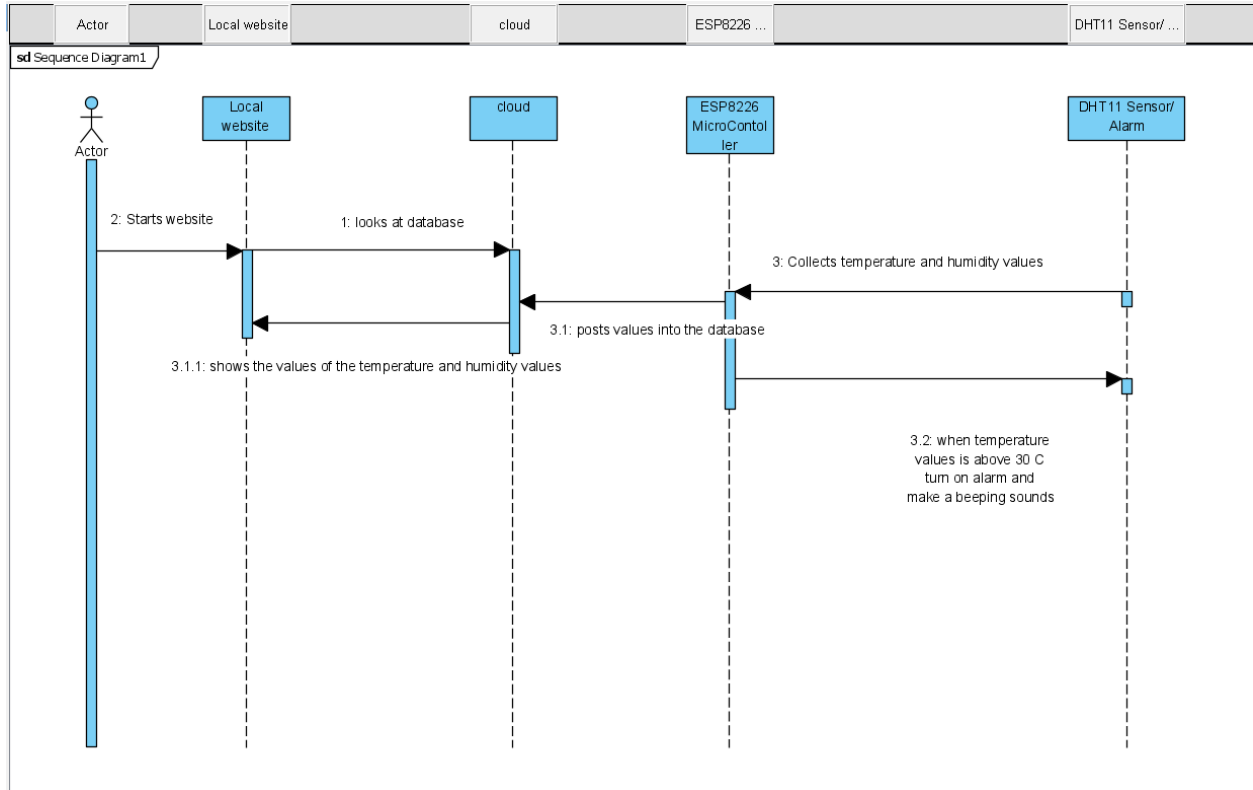
UI Test Cases

Test number	Test case type	Description	Test step	Expected result	Status
TC-1	Functionality	the website should display and change the temperature values as the temperature changes	Bring a lighter close to the temperature sensor	The temperature should get warmer and have a increase in the temperature	pass
TC-2	Functionality	The buzzer should beep when the temperature increases past 30 C	Bring a lighter close to the temperature sensor making the temperature increase past 30 C	The buzzer should start beeping	pass
TC-3	database	temperature and humidity values should save to a database	The server simultaneously updates the database	Temperature and humidity values can be shown in the database	pass

Architectural diagram



Sequence Diagram



Demo:

[Fire Alarm](#)