A logo with a blue circle and red stripes

Description automatically generated

**T.C.**

**MARMARA UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**CSE4074 PROGRAMMING ASSIGNMENT:  
REMOTE SENSING APPLICATION**

Utku BAYGUVEN

# Project Summary

The project involves the creation of a networked system comprising temperature and humidity sensors, a gateway, and a server. The system simulates the generation, transmission, and monitoring of sensor data through applications. There is no need for a physical hardware in this project other than our computers. Key features are; randomized sensor data generation, communication via TCP and UDP, multi-threaded server processes, and a web interface for data visualization.

# Our Solution Approach

Our solution approach centers on the following key aspects:

**Sensor Simulation:**

* Temperature and humidity sensors are simulated as applications generating random values.
* Temperature sensor communicates via TCP, while the humidity sensor utilizes UDP.
* Data is sent periodically, with additional conditions for humidity sensor communication.
* Every second, the temperature sensor generates a value randomly between 20 and 30 and sends it to the gateway along with the timestamp.
* The humidity sensor generates random values between 40 and 90 every second but sends information only if the humidity value exceeds 80. Additionally, every 3 seconds, the humidity sensor sends an 'ALIVE' message.

**Gateway Functionality:**

* The gateway reads data from sensors and forwards it to the server over TCP.
* Sensor activity monitoring is implemented, triggering alerts for potential failures.
* Threading is employed to handle concurrent connections, ensuring efficient communication.

**Server Implementation:**

* The server receives messages from the gateway based on a predefined protocol.
* A handshake occurs between the gateway and server at the outset.
* Data related to connected devices is stored, and a web interface is provided for data visualization.
* HTTP processes on port 8080 enable access to temperature and humidity data.

**Implementation Choices:**

* Socket programming is utilized for communication.
* Multi-threaded server processes enhance scalability and responsiveness.
* No external libraries are employed, aligning with project requirements.

# Encountered Problems and Solutions

**Problem:** There was an error with threads that an extra humidity thread was created during initiation.

**Solution:** There was bug with the code itself that we fixed later during development.

**Problem:** Humidity sensor wasn’t working without temperature sensor.

**Solution:** Humidity sensor has only one thread that is running and for the temperature sensor we constantly open and close a thread in order to solve this problem.

# Usage Explanation

The system's usage is straightforward:

* Execute server.py to start the server, which listens for incoming connections.
* Execute gateway.py to handle connections from sensors and forward data to the server.
* Run temperature\_sensor.py and humidity\_sensor.py to simulate sensor data generation.
* Access the web interface at 'http://localhost:8080/temperature' and 'http://localhost:8080/humidity' for data visualization and . 'http://localhost:8080/gethumidity' for last humidity data.

# 5. Classes

5.1 Humidity Sensor Class

The humidity\_sensor.py script is designed to emulate the functionality of an environmental sensor that monitors humidity levels. It operates by periodically generating random humidity values within a defined range (40 to 90 percent). This simulation is achieved through the use of Python's random library, which provides the capability to produce these fluctuating humidity readings.metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

5.2 Temperature Sensor Class

The script temperature\_sensor.py functions as a simulated temperature sensor, continuously generating random temperature values within a specified range (20 to 30 degrees). It establishes a TCP connection to a server located at 'localhost' on port 5001. Once connected, it repeatedly sends temperature data in a structured message format, including a timestamp, at one-second intervals. This script effectively mimics the behavior of a real-world temperature sensor, sending data to a server for processing, analysis, or storage in a real-time and continuous manner.

metin, ekran görüntüsü, yazılım, multimedya içeren bir resim

Açıklama otomatik olarak oluşturuldu

5.3 Server Class

1. handle\_gateway\_connection(conn, addr): This function manages the connection with the gateway. It receives data (temperature or humidity readings, or 'ALIVE' signals) from the gateway, processes it, and stores it in a data structure, ensuring thread safety with a lock.
2. construct\_http\_response(path): This function constructs an HTTP response based on the requested path. It retrieves either temperature or humidity data (or the last real humidity value) from the stored data and formats it into an HTML response.
3. handle\_client\_connection(client\_socket, addr): This function handles incoming client connections to the HTTP server. It reads the client's request, determines the requested path, generates the appropriate HTTP response using construct\_http\_response, and sends this response back to the client.
4. start\_http\_server(): This function initializes and starts the HTTP server. It listens for incoming connections, handles gateway connections in a separate thread, and spawns new threads for each client connection to process client requests.
5. accept\_gateway\_connections(): This function runs in its own thread, listening for and accepting connections from the gateway. For each connection, it starts a new thread using handle\_gateway\_connection to manage the communication with the gateway..

5.4 Gateway Class

1. **handle\_temperature\_connection(conn, addr)**: This function manages the connection with a temperature sensor over TCP. It continuously receives temperature data, converts it into JSON format, and forwards it to another server (on TCP port 5003) using a TCP client socket. The function includes a handshake mechanism for connection validation and error handling to manage sensor disconnections and attempts to reconnect.
2. **handle\_humidity\_connection(sock)**: This function handles humidity data received through a non-blocking UDP socket. It processes incoming data, distinguishes between humidity readings and 'ALIVE' signals, and forwards this data in JSON format to another server (on TCP port 5003) using a TCP client socket. The function includes logic to detect sensor disconnections based on the absence of 'ALIVE' messages and attempts reconnection.
3. **Main Loop**: In the main part of the script, TCP and UDP sockets are set up for temperature and humidity sensors, respectively. It creates and manages separate threads for handling temperature and humidity data using the above functions. The script ensures continuous operation and manages the lifecycle of these threads, restarting them as necessary in case of disconnections or errors. This loop is the central coordinator for the gateway's operations, enabling simultaneous and efficient handling of data from both types of sensors.

**Outputs**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Gateways humidity and temp sensor offs and other received datas

metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Gateway accept message and received data from server.py

**Logs**

**metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Gateway.py logs

metin, ekran görüntüsü, menü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Server.py logs

**metin, ekran görüntüsü, yazı tipi, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturulduWeb Interfaces**

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu