**Cyber Physical Systems (SEP 769)**

**Smart Temperature Control with LEDs**

Maged Eid, Shawn, Abrar, Akash

*1W Booth School of Engineering Practice and Technology, McMaster University,*

*Canada.*

Email address: [eidm6@mcmaster.ca](mailto:eidm6@mcmaster.ca), ….

***Keywords:*** *IoT, Cuber Physical Systems, Machine Learning, Temperature Full Control System, MQTT*

1. **Abstract**

This project use Raspberry PI, MQTT, HTML, CSS, and JavaScript to create a dynamic and responsive temperature control system. By utilizing MQTT as the communication protocol, we establish a connection between the physical devices and the web interface. The system has two key components: the Publisher and the Listener.

The Publisher class diligently monitors the current environmental temperature and, at regular intervals, transmits this data to the designated MQTT topic, "home/temperature/current." This information is then readily accessible to the HTML interface webpage, allowing users to visualize the real-time temperature and control it.

In parallel, the Listener class actively listens for updates on the "home/temperature/setpoint" topic. This topic receives the desired temperature input from the user via the HTML interface. By comparing the current temperature with the setpoint, the system intelligently determines the necessary actions. If the current temperature falls below the setpoint, the heater is activated, and the LED illuminates to signify this state. Conversely, if the current temperature exceeds the setpoint, the cooler is engaged, and the LED is illuminated. When the current temperature aligns with the setpoint, the LED indicates that no temperature adjustment is required.

# Introduction

## This project presents an approach to temperature control utilizing MQTT, HTML, and JavaScript. By seamlessly integrating these technologies, we have developed a system capable of monitoring and adjusting environmental temperature in real time, providing a user-friendly and efficient solution.

## The core of the system is based on two primary components: the Publisher and the Listener. The Publisher class is responsible for continuously monitoring the current temperature and transmitting this data to an MQTT broker. This data is then made accessible to the HTML interface, where users can visualize the current temperature.

## Simultaneously, the Listener class actively listens for updates on the designated MQTT topic, "home/temperature/setpoint." This topic receives the desired temperature input from the user via the HTML interface. By comparing the current temperature with the user-defined setpoint, the system intelligently determines the necessary actions [1]. If the current temperature falls below the setpoint, the heater is activated, and a visual cue is provided through the illumination of a red LED. Conversely, if the current temperature exceeds the setpoint, the cooler is engaged, and a blue LED is illuminated. When the current temperature aligns with the setpoint, a yellow LED indicates that no temperature adjustment is required.

## This project demonstrates the effectiveness of combining MQTT, HTML, and JavaScript for creating innovative and practical solutions. By leveraging the strengths of each technology, we have developed a system that offers precise temperature control, user-friendly interaction, and real-time monitoring capabilities. The system's core functionality involves monitoring the current temperature, comparing it to a user-defined setpoint, and adjusting the heating or cooling system accordingly. The system's responsiveness is enhanced using real-time data and LED indicators that provide visual feedback on the system's status.

## 3. Project Overview and Technology Background

# This project presents an advanced temperature control system that leverages MQTT, HTML, and JavaScript to create a seamless user experience for real-time environmental monitoring and temperature management. This system is designed with user convenience and efficiency at its core, offering a robust and practical solution for maintaining an optimal environment. Through innovative integration of these technologies, the system enables users to monitor and adjust the temperature according to personalized preferences.

# The system's core functionality is built around several critical features. One is real-time temperature monitoring, which tracks the current temperature and displays it clearly on a web-based interface. By presenting data in real-time, users are constantly informed about the current environmental conditions, ensuring they can adjust as needed. Another essential feature is the user-defined set point, which allows users to input their desired temperature range. This input is processed directly through a web interface, allowing users to tailor the system to meet their specific needs, whether for comfort, energy efficiency, or environmental control. Once users set a temperature range, the system automatically adjusts to maintain it, activating heating or cooling functions as necessary.

# A significant aspect of the system's functionality is automated temperature control. It responds to real-time data by dynamically activating heating or cooling elements based on the set points. This automation not only reduces manual adjustments but also optimizes the efficiency of the system, allowing for hands-free experience. Alongside automation, visual feedback is a key feature. By using LED indicators, the system provides immediate visual cues on its status:: a red LED for heating, a blue LED for cooling, and an LED to signify that the temperature aligns with the desired set point. This approach makes the system highly interactive and easy to understand, as users can glance at the indicators to assess its operational status.

# The MQTT (Message Queuing Telemetry Transport) [2] protocol is foundational to the project’s communication framework. Known for its lightweight nature and efficiency in low-bandwidth networks, MQTT is a publish-subscribe protocol ideal for IoT systems where minimal overhead and reliability are crucial. In this project, MQTT handles all communication between the temperature sensor, the control unit, and the web interface, ensuring smooth and reliable data flow. By serving as the communication backbone, MQTT allows for real-time temperature updates and immediate responses to user inputs, significantly enhancing the system’s reactivity and functionality.

# On the front end, HTML (Hypertext Markup Language) is used to build a clear, accessible, and interactive interface. HTML structures the webpage and displays content, enabling users to interact with the system directly. It includes sections to display the current temperature, fields to input the desired temperature set point, and visual indicators that show the system’s operational status. The simplicity and versatility of HTML make it an ideal choice for this project, providing a user-friendly layout that enhances interaction and ease of use.

# The JavaScript component adds a layer of dynamism and interactivity to the web interface. JavaScript allows the system to respond to real-time data and user input immediately. Key functionalities include fetching and displaying temperature data, processing user interactions, controlling the system’s behavior, and updating visual indicators dynamically. JavaScript retrieves current temperature readings from the MQTT broker [3] and updates them on the interface, ensuring users see the most recent data. It also handles user inputs, such as setting temperature preferences, and sends these commands to the control unit. Furthermore, JavaScript adjusts the system’s behavior according to the current temperature and set points, automating the transition between heating and cooling modes. The language’s adaptability in handling data and user commands makes it invaluable in creating an interactive, responsive system.

# In summary, this project exemplifies a successful application of MQTT, HTML, and JavaScript, with each component contributing unique strengths. MQTT facilitates reliable data exchange, HTML structures the user interface, and JavaScript enables interaction and dynamic updates. Together, they create an effective and user-friendly temperature control system that combines real-time monitoring with automated environmental adjustment. The result is a practical solution for modern IoT applications, offering enhanced user control, efficient automation, and a streamlined interface that ensures ease of use.

# 4. Project Methodology

The proposed temperature control system is designed as a client-server architecture. The server component, a microcontroller, is responsible for data processing and control. It reads temperature sensor data, processes user-defined set points, controls the heating/cooling system, and publishes temperature and system status to the MQTT broker. The client component, a web-based interface, is developed using HTML, CSS, and JavaScript. It fetches temperature and system status data from the MQTT broker, displays the data on the web interface, handles user input for setting temperature set points, and sends user commands to the microcontroller via MQTT.

To implement the system, the hardware components, including the microcontroller, temperature sensor, heating/cooling system, and network connectivity, are assembled and configured. The microcontroller is programmed to read temperature sensor data, process user commands, and control the heating/cooling system. The MQTT broker is set up to facilitate communication between the microcontroller and the web interface.

On the software side, the microcontroller code is developed to read temperature sensor data, process user-defined set points, control the heating/cooling system, and publish temperature and system status to the MQTT broker. The web interface is developed using HTML, CSS, and JavaScript. HTML and CSS define the structure and style of the page, while JavaScript handles dynamic behavior, including fetching and displaying data, processing user input, and sending commands to the microcontroller.

Thorough testing is a important part to ensure the system's functionality, reliability, and accuracy. The system is tested under various conditions to identify and address any issues or bugs. Based on the testing results, the system's performance and user experience are refined. Once the system is fully tested and optimized, it is deployed to a web server, and the microcontroller and MQTT broker are configured for production use. Secure and reliable communication is established between the components to ensure smooth operation and data integrity.

A phone with text on it

Description automatically generated

Figure (1) My MQTT App

## 5. Backend Development

The backend for this Cyber-Physical Systems project is designed to be resilient, modular, and secure, offering efficient and real-time control while safeguarding against security threats. The combined use of MQTT for communication, modular structure, and robust error-handling mechanisms make it well-suited for managing complex temperature control tasks, providing users with efficient and reliable experience.

# 5.1 Backend Overview and Architecture

The backend design for the Cyber-Physical Systems project is focused on establishing a robust connection and control system that interacts with the HTML frontend. This system is designed to control and manage the various settings (e.g., power, heating, cooling) in a temperature-regulated environment. The backend architecture uses a layered approach to ensure seamless communication, scalability, and fault tolerance.

To achieve this, the backend utilizes a message broker, specifically MQTT (Message Queuing Telemetry Transport) over WebSockets, to facilitate real-time communication between the front end and back end. MQTT is a lightweight, publish-subscribe protocol ideal for scenarios with minimal resources or limited network bandwidth. This is particularly suited for IoT-based projects, as it allows devices and web clients to communicate with low overhead. The Paho MQTT library, a JavaScript implementation for WebSockets, is integrated into the backend to manage interactions between devices, providing control commands to adjust the actual and desired temperatures.

The backend has three main parts working together. The Control Module is like the brains of the operation, taking commands like "on," "off," "heat," and "cool" from the front end. At the same time, the Temperature Monitoring Module is constantly checking the temperature and adjusting it to match what you've set. The Communication Handler Module is the system's messenger, using MQTT to talk to other devices and keep everything connected. This modular design makes it easy to update or change parts without affecting the whole system, so it can adapt to whatever you need.

# 5.2 Detailed Module Descriptions

The Control Module is a key component that receives and processes commands from the frontend buttons (On, Off, Heating, Cooling). When a user interacts with one of these buttons, a specific command is sent to the backend via MQTT. For example, when the "Heating" button is pressed, the backend increases the desired temperature until it reaches a set threshold. Conversely, when the "Cooling" button is pressed, the desired temperature decreases to maintain a comfortable environment. These actions are processed in real time, allowing users to have instant control over temperature settings.

To ensure the system remains within a safe range, the Control Module includes preset temperature limits (between 16°C and 30°C), which helps prevent the device from overheating or freezing. The module's logic enforces these limits and prevents undesired temperature adjustments. This is managed by continuously checking the current state and conditions, adjusting only within the preset parameters to safeguard both the system and the environment it controls.

The Temperature Monitoring Module is essential for tracking and reporting real-time temperature. This module utilizes sensors to measure the actual room temperature and then sends updates back to the front end through MQTT messages. The module compares the current temperature with the desired temperature set by the user. When a significant discrepancy exists, the module triggers the appropriate action to either increase or decrease the temperature accordingly.

One of the primary challenges addressed by this module is to keep the data flow efficient without overwhelming the MQTT broker or client with too many messages. This is managed through an adaptive monitoring approach, where updates are sent more frequently when the actual and desired temperatures vary widely. Additionally, the module implements a buffer mechanism, temporarily storing readings to avoid network congestion by batching and sending data at controlled intervals. This way, the backend maintains consistent performance even in high-demand scenarios.

# 5.3 Communication and Security

The Communication Handler Module handles the data exchange between the front end and back end using the MQTT protocol. It is configured to establish WebSocket connections over the Paho MQTT library, which provides a reliable framework for sending and receiving commands and temperature readings. The module manages MQTT topics, such as “temperature/actual” and “temperature/desired,” to differentiate between various types of data, ensuring that each component listens only to the relevant messages. This topic-based structure makes the system highly scalable, as new functionalities can be added by creating additional topics.

A crucial aspect of this module is error handling and reconnection logic. Given that MQTT is designed for lightweight, continuous communication, maintaining a stable connection over potentially unstable networks is essential. This module handles dropped connections by automatically reattempting establishing a link and resubscribing to all necessary topics once the connection is restored. This ensures that the backend is resilient against network interruptions, maintaining data integrity and operational reliability.

Security is paramount, especially when dealing with sensitive environmental controls. The backend leverages TLS (Transport Layer Security) to encrypt MQTT messages sent over WebSockets [4]. This prevents unauthorized access or data tampering by encrypting all data exchanged between the front end and back end. Additionally, authentication mechanisms are in place to ensure that only authorized users or devices can access and control the system.

Each component within the backend follows secure coding practices, such as input validation and data sanitation, to prevent vulnerabilities like command injection or buffer overflows. To manage user authorization, the backend assigns unique session tokens that allow temporary control over the system, expiring after a set period of inactivity. This helps protect against unauthorized access by limiting the duration of active control sessions.

# The visualization of object detection and connectivity was insightful because it provided a clear, dynamic representation of how different unit operations and equipment were identified and linked within the process diagram. By animating the progression of detected contours and connections, we could visually validate the accuracy of the detection and understand how the different components of the diagram interacted. This step helped highlight any potential errors in object detection, such as missed objects or incorrect associations between text and objects, allowing for immediate correction.

# Additionally, the visualization enabled us to see the logical flow between unit operations, offering a real-time perspective on how the chemical process might function. This insight allowed for better interpretation of the process structure and helped refine the connectivity detection algorithm, ensuring that only meaningful connections were drawn between the relevant objects. It also helped in evaluating the spatial arrangement of objects and the clarity of their interactions, which is critical when moving towards a full simulation of the process. These methods enabled us to create a comprehensive system that not only detects objects and labels them accurately but also interprets their connections, allowing for a full simulation of the chemical process.

A building with glass windows

Description automatically generated

Figure (2) Webpage for Controlling

# 6. Results and Discussion

This project demonstrates the successful implementation of a real-time temperature control system that leverages MQTT messaging to provide seamless communication between the client interface and the backend. By utilizing LED indicators, the system gives users a straightforward visual representation of current temperature conditions and system responses to set points, enhancing user experience and offering a highly responsive and adaptable control environment. Throughout testing and evaluation, the system consistently maintained effective bidirectional communication, efficiently adjusted environmental temperatures, and provided accurate status indications through LEDs.

The project yielded several key results. First, it effectively monitored and controlled temperatures within a predefined range (16°C to 30°C), ensuring both user comfort and equipment safety. When the user adjusted the desired temperature on the interface, the backend processed the commands and adjusted the environment accordingly, with responses showing minimal latency. The rapid response was facilitated by MQTT’s lightweight messaging protocol, which allowed for low-overhead communication. The modular backend design also proved advantageous, as it minimized processing delays, maintained stable connections, and provided smooth, uninterrupted updates on temperature and control statuses. The system’s robustness was apparent as it consistently recovered from network interruptions by reconnecting and resubscribing to MQTT topics, ensuring continuous operation without requiring user intervention.

One significant result observed was the reliability of the LED indicators in representing system status. LEDs responded immediately to changes, indicating whether the system was in a heating, cooling, or standby mode based on the desired temperature settings relative to the actual environmental readings. For example, when the actual temperature fell below the set point, the system entered a heating mode, and the corresponding LED illuminated, clearly notifying the user of the system's operational status. Similarly, in cooling mode, the LED accurately reflected the system’s state, making it easy for users to interpret real-time conditions visually. This clear visual feedback is beneficial in environments where immediate knowledge of system status is crucial, such as in labs or industrial settings, where environmental stability can impact results or equipment performance.

During testing, we evaluated the system’s performance under different network conditions, including latency and bandwidth constraints. MQTT’s efficient protocol ensured data packets were transmitted with minimal delay, even under limited network conditions, demonstrating its suitability for real-time applications. Furthermore, the MQTT protocol’s publish-subscribe model allowed data to be transmitted to multiple clients without impacting performance. This setup is beneficial for multi-user environments or scenarios where multiple devices require simultaneous access to environmental data, such as temperature values or control statuses. In practice, this would mean that both system operators and secondary devices, such as logging systems, could access data concurrently, enhancing the overall operational efficiency of the system.

However, some limitations were also identified. One challenge was optimizing the frequency of temperature updates to balance real-time accuracy with network efficiency. Continuous updates can create high network traffic, but updating too infrequently could compromise responsiveness. To address this, the system implemented an adaptive update mechanism that increased the update frequency when a significant deviation was detected between the actual and desired temperatures. This reduced unnecessary messaging and helped maintain efficient communication, but further optimizations could improve the system’s performance in high-demand environments.

Another area for discussion relates to the system’s scalability. While the MQTT protocol supports large-scale deployment, scaling the system across numerous devices or locations might introduce challenges, such as increased server load or complexity in managing numerous concurrent connections. Future enhancements could include load balancing mechanisms or decentralized MQTT brokers to distribute the load and reduce server strain. Additionally, security features such as encrypted MQTT channels were implemented, ensuring data privacy and protecting against unauthorized access. While the encryption strategy proved effective in safeguarding data, more comprehensive authentication and authorization mechanisms could be explored to enhance security further, especially if this system were deployed in a highly sensitive or regulated environment.

In summary, this project demonstrated that MQTT messaging, combined with a well-structured backend and LED indicators, can provide an efficient and user-friendly solution for temperature control and monitoring systems. The results indicate that this approach is viable for a range of applications where environmental control is necessary, such as industrial, commercial, or even home automation systems. The system’s strengths lie in its reliable communication, adaptability to different network conditions, and clear visual feedback through LEDs. Limitations, such as network optimization and scalability, present opportunities for future research and development. Overall, this project underscores the potential of MQTT-based control systems to enhance environmental management by providing responsive, real-time control mechanisms that are accessible to users across various sectors.

# 7. Work In Development

Implementing a Deep Learning Neural Network for Enhanced System Intelligence

To enhance the temperature control system, a deep learning neural network is under development to enable predictive temperature regulation, adaptive control, and dynamic adjustment of desired environmental conditions. The goal is to create a smart system that not only reacts to current conditions but anticipates future temperature needs based on patterns, user behaviors, and environmental data. By leveraging deep learning, the system will provide an intelligent layer that autonomously optimizes temperature settings, minimizing energy consumption while maintaining comfort and operational stability.

The neural network model is designed with a multi-layered approach, incorporating both convolutional and recurrent layers to effectively process and interpret time-series temperature data. The model consists of an input layer that receives historical temperature data, weather conditions, and user-defined set points. These inputs are then fed into convolutional layers that extract patterns, followed by a Long Short-Term Memory (LSTM) layer to capture temporal dependencies and forecast temperature trends. LSTMs are particularly suited to this application because they maintain memory of past inputs, allowing the system to predict temperature fluctuations over time. For the final layers, dense neural connections combine the insights gathered, producing output values that suggest optimal set points and heating or cooling intensities.

A distinctive advantage of this architecture is the integration of both convolutional and recurrent layers, which allows the model to handle real-time, high-dimensional temperature data effectively. Convolutional layers excel at identifying patterns in complex datasets, while LSTM layers allow the model to retain valuable insights from past data, learning from daily, weekly, and seasonal temperature trends. Together, this architecture forms the foundation for a system that anticipates temperature changes based on both historical patterns and immediate environmental factors.For the deep learning model to deliver accurate predictions, it requires substantial, high-quality data for training. Data collection for this project includes historical temperature data, occupancy patterns, external weather conditions, and user interaction records with the system. This data is fed into the model to identify patterns in heating and cooling requirements based on usage times, seasonality, and other variables. To enhance model generalizability, various scenarios are simulated, including extreme weather changes, atypical occupancy levels, and seasonal transitions.

# To achieve optimal model performance, the training process employs backpropagation with gradient descent, where the model iteratively adjusts its weights based on prediction errors. A Mean Squared Error (MSE) loss function evaluates the difference between predicted and actual temperature outcomes, providing feedback to the model to improve its predictive accuracy over successive epochs. Regularization techniques, such as dropout layers, are applied to mitigate overfitting and ensure that the model generalizes well to unseen data. Data augmentation is also explored by introducing minor noise variations, allowing the model to become robust in dynamic or noisy environments.

# To enable real-time adaptability, the system is being integrated with online learning capabilities. Online learning will allow the neural network to continuously refine its predictions based on new data, such as recent user adjustments, temperature fluctuations, or unexpected environmental changes. This ongoing learning loop is facilitated by a lightweight version of the neural network, which updates its weights incrementally without requiring full retraining. This adaptability is essential in dynamic environments where conditions may shift rapidly, as it ensures the system remains responsive and accurately tuned to current needs.

# Moreover, implementing a reinforcement learning component is being considered to allow the system to adapt based on feedback. For example, when the system adjusts the temperature in response to certain predictions, it records user satisfaction (measured through feedback or system overrides). Based on this feedback, the model learns which adjustments lead to favorable outcomes and which do not, allowing it to make better-informed decisions over time. This approach ensures the system remains aligned with user preferences and operational efficiency goals, creating a more intuitive and user-centric experience.

# In addition to enhancing comfort, a major focus of this development is improving energy efficiency. The neural network leverages predictive analytics to minimize energy consumption by preemptively adjusting the HVAC system in anticipation of temperature changes. For example, if a significant temperature drop is forecasted, the system may reduce heating output in advance, preventing unnecessary energy expenditure. Additionally, during off-peak hours or periods of minimal occupancy, the neural network can enter an “energy-saving” mode, maintaining minimal temperature levels while ensuring quick recovery times for comfort settings when users return.

A screenshot of a computer code

Description automatically generated

Figure (3) DL NN Smart Control

To achieve this, the neural network calculates an optimal temperature range that balances energy efficiency with user comfort. By monitoring fluctuations in environmental data, the system can dynamically adjust HVAC intensity, offering a “smart thermostat” capability. This adaptability is particularly beneficial in commercial and industrial settings where occupancy and temperature control needs fluctuate throughout the day. Moreover, the system can prioritize energy savings when external conditions allow, such as during mild weather, by operating at a reduced capacity or switching to eco-friendly modes when appropriate.

# The neural network is seamlessly integrated with the MQTT messaging protocol, which underpins the system’s communication infrastructure. MQTT’s publish-subscribe architecture allows the neural network’s predictions to be distributed to various devices within the IoT ecosystem, facilitating real-time, centralized control. For instance, if the neural network determines an impending temperature change, the prediction is published to the relevant topic, notifying the HVAC controller to adjust heating or cooling intensities accordingly. This integration ensures that all components within the IoT network, including temperature sensors, LED indicators, and user interfaces, remain synchronized, maintaining consistent and efficient temperature management.

# In practice, this setup not only improves performance but also supports scalability. As additional devices are added to the IoT network, the MQTT protocol allows for seamless integration, with the neural network’s predictions distributed effortlessly across the network. This architecture opens the door for future expansions, such as incorporating new sensors, supporting voice commands, or integrating with other building management systems. The MQTT framework also facilitates cross-device compatibility, enabling users to control the system remotely via mobile devices or smart home assistants. The deep learning neural network has the potential to transform this temperature control system into an intelligent and self-sustaining solution, adding value across various domains. Immediate benefits include enhanced user comfort, energy savings, and operational efficiency. Additionally, the system’s adaptability positions well for diverse applications, from residential smart homes to large-scale industrial facilities requiring precise environmental control.

# Looking forward, expanding the dataset and parameters to include more granular environmental variables—such as humidity, air quality, or even predictive maintenance indicators—could make the neural network’s predictions even more accurate and relevant. Further, implementing federated learning could allow the system to share insights across multiple installations while maintaining data privacy, enhancing its capabilities through collaborative learning across different deployment sites. Ultimately, these ongoing developments will enable a truly intelligent, responsive, and sustainable temperature control system, setting a new standard for autonomous environmental management.

# 8. References

 **Edge Analytics for Internet of Things (IoT) Applications**  
Mung Chiang, "Edge analytics for Internet of Things applications: Vision and challenges," *ACM Computing Surveys*, 2017. Available on ACM Digital Library: https://dl.acm.org/doi/10.1145/3093029

 **MQTT Protocol: Performance in IoT Applications**  
Rodríguez et al., "Performance Evaluation of the MQTT Protocol in IoT Applications," *Sensors*, MDPI, 2020. https://www.mdpi.com/1424-8220/20/6/1750

 **Comparative Analysis of MQTT Broker Performance**  
Al-Kashoash and Kemp, "MQTT Performance for IoT Applications," *MDPI Energies*, 2021. https://www.mdpi.com/1996-1073/14/19/5817

 **Security and Performance Analysis of MQTT with TLS in IoT Networks**  
Khaing and Naing, "Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks," *IEEE Conference Publication*, 2021. <https://ieeexplore.ieee.org/document/9717495>

# 9. Acknowledgements

I would like to express my gratitude to all those who have contributed to this project and who worked to make it work.

Thank you all.

Kind regards,

**Group (11)**