|  |  |
| --- | --- |
| ElderSense  A Smart IoT Solution for Elderly Environmental Monitoring | Abstract  "ElderSense" is an IoT project focusing on real-time environmental monitoring for elderly care. Utilising Arduino-based technology, the system ensures seamless integration of temperature sensors, cloud-based analytics, and SMS notifications. The project aims to enhance the quality of life for the elderly and support caregivers through a responsive and inclusive solution.  JESSICA MURPHY  COM6005M - THE INTERNET OF THINGS - 2023-24 |

Table of Contents

[Introduction 2](#_Toc155962579)

[Problem Definition 2](#_Toc155962580)

[Project Requirements 3](#_Toc155962581)

[Project Implementation 3](#_Toc155962582)

[IoT Device 3](#_Toc155962583)

[Hardware 3](#_Toc155962584)

[Software 3](#_Toc155962585)

[Machine to Machine (M2M) 4](#_Toc155962586)

[Internet Connectivity 4](#_Toc155962587)

[Data Visualisation 4](#_Toc155962588)

[Additional 5](#_Toc155962589)

[System Design 5](#_Toc155962590)

[IoT Device Components 5](#_Toc155962591)

[Network Infrastructure 5](#_Toc155962592)

[Cloud Services 5](#_Toc155962593)

[PCs 5](#_Toc155962594)

[Project Testing phase 6](#_Toc155962595)

[Hardware Integration Testing 6](#_Toc155962596)

[Network Connectivity Test 7](#_Toc155962597)

[Machine-to-Machine (M2M) Connectivity Test 7](#_Toc155962598)

[Data Storage and Analytics Verification 7](#_Toc155962599)

[SMS Notification System Verification: 8](#_Toc155962600)

[User Interface Testing 8](#_Toc155962601)

[Legal & Ethical Evaluation 9](#_Toc155962602)

[Legal Considerations 9](#_Toc155962603)

[Ethical Considerations 9](#_Toc155962604)

[Commercial & Economic Context 9](#_Toc155962605)

[Sustainability & Equality, Diversity, and Inclusion (EDI) Issues 9](#_Toc155962606)

[Conclusion 10](#_Toc155962607)

[Source code 10](#_Toc155962608)

[Reference 11](#_Toc155962609)

[Appendix 12](#_Toc155962610)

[Project Architectural Diagrams 12](#_Toc155962611)

[Circuit Design 15](#_Toc155962612)

[Data Analytics 15](#_Toc155962613)

[Testing Phase 17](#_Toc155962614)

# Introduction

This project delves into the realm of IoT, focusing on the development of a smart environmental monitoring system. With a core objective of addressing the critical need for real-time temperature monitoring in living spaces, this project integrates cutting-edge hardware, software, and connectivity technologies to create a responsive and user-centric solution.

## Problem Definition

The demographic landscape is undergoing a transformative shift globally, with an increasing aging population requiring specialised attention and care (Stavropoulos *et al.*, 2020). One prominent challenge faced by the elderly and their caregivers is the need for real-time monitoring of environmental conditions, particularly temperature variations, within living spaces (*Cold Weather Safety for Older Adults*, 2024). In many instances, maintaining a comfortable and safe environment becomes a critical factor in ensuring the health and well-being of elderly individuals, especially when their ability to regulate their surroundings may be compromised (*Winter deaths report: how does cold weather affect mortality in the UK?*, 2022).

Research shows that traditional monitoring systems often fall short in providing a seamless, unobtrusive solution tailored to the unique needs of the elderly (Selvaraj and Sundaravaradhan, 2019). Therefore, this project aims to develop an innovative IoT-based solution utilising temperature sensors and data analytics to create a responsive environment. By doing so, the project aims to enhance the quality of life for elderly individuals while additionally supporting caregivers in their responsibilities (Iancu and Iancu, 2020).

This project will delve into the intricacies of hardware and software integration, exploring the capabilities of Arduino-based technology, and importance of cloud-based services for efficient data storage and analytics. It is anticipated that this project will not only address a specific problem but also contribute to the broader discourse on the responsible and inclusive use of technology in the realm of elderly care.

# Project Requirements

The following breaks down the project requirements.

1. IoT Device
   1. Temperature Sensor
   2. Backlight Display
   3. IoT Software
2. Utilise some aspect of machine two machine (M2M) connectivity.
3. Use internet connectivity.
4. Use data storage and data analytics using any IoT platform.

# Project Implementation

The project implementations require a range of hardware, software, and cloud services to achieve the objectives outlined in the introduction and project requirements (see Figure 1). Below explains how the Arduino program intelligently manages temperature data, adjusting LCD display colours, synchronising time through network time protocol (NTP), establishing machine to machine (M2M) connectivity via messaging protocol (MQTT), and visualising data on ThingSpeak through hypertext transfer protocol (HTTP) communication.

## IoT Device

### Hardware

* **Arduino Uno Wi-Fi R2 Board:** This board will provide seamless integration of IoT sensors, providing the necessary computational power for data processing and transmission (*ARDUINO UNO WiFi REV2*).
* **Grove - Base Shield:** This boardeasily connects various Grove modules (sensors, actuators, displays, etc.) to an Arduino without the need for complex wiring. The shield features standardised connectors and a plug-and-play design. This board streamlines the process of prototyping the IoT artefact for this project(*Grove - Starter Kit v3 | Seeed Studio Wiki*, 2023a)
* **Grove Seeed Studio Temperature Sensor v1.3:** This is a precise temperature sensor compatible with the Arduino board that will ensure accurate and real-time environmental monitoring within living spaces (*Grove - Starter Kit v3 | Seeed Studio Wiki*, 2023b).
* **Grove-LCD RGB Backlight Display v5.0:** This will provide a clear visual representation of temperature variations. The visual feedback enhances user comprehension and interaction (*Grove - Starter Kit v3 | Seeed Studio Wiki*, 2023c).
* **Power Supply:** In development mode, the Arduino is powered through USB connected to a PC. During runtime, a reliable 9V battery ensures continuous operation and sustained data capture, optimising energy efficiency for prolonged usage.

### Software

* **VS Code IDE:** Visual Studio Code (VS Code) Integrated Development Environment (IDE) has been chosen for programming the Arduino board. This platform streamlines firmware development and code editing, providing a user-friendly interface for efficient project development.
* **Programming Languages**: Arduino Sketch (C++) is the programming language used for the project. This language facilitates seamless communication between the Arduino board and the connected sensors, enabling effective data processing and transmission (*Arduino Reference - Arduino Reference*).
* **Code Design**: Temperature thresholds are defined to determine the colour of the Arduino LCD display. Green signifies an optimal temperature, blue indicates cold temperatures, and red warns of high temperatures. The Arduino program dynamically adjusts the LCD display colour based on predefined temperature thresholds.

The code design will include configurations to establish internet connectivity using Domain Name System (DNS), ensuring accurate time updates through Network Time Protocol (NTP) while also obtaining network settings seamlessly through Dynamic Host Configuration Protocol (DHCP). The Arduino program will enable M2M communication with cloud services via MQTT with Mosquitto and SMS with Twilio. Additionally, data updates will be sent to ThingSpeak for data visualisation and data analytics. These configurations are vital for the project's functionality.

## Machine to Machine (M2M)

* **MQTT Mosquitto Platform:** MQTT via Mosquitto has been chosen to establish machine-to-machine (M2M) connectivity, with the Arduino serving as the MQTT publisher. The MQTT broker at test.mosquitto.org facilitates communication, ensuring reliable data transmission between the Arduino and other connected devices e.g. PC (see Figure 2).

The PC establishes a connection to the MQTT broker by subscribing via the command prompt. By opening the Mosquitto folder “C:\Program Files\mosquitto”, the following command, “mosquitto\_sub -h test.mosquitto.org -t warmth-checker”, establishes a connection to enable periodic temperature updates alongside the time at which the temperature sample was taken.

* **SMS Twilio Platform:** The Twilio platform has been chosen to establish a M2M connection, enhancing communication through SMS messages sent to a designated mobile phone. To achieve this, an SMS provider receives real-time updates from the MQTT broker through a Python monitoring module that subscribes to the MQTT broker (see Figure 3). Notably, the project utilises the Patho python library to enhance the efficiency and reliability of SMS communication within the system. This integration ensures timely notifications, enhancing the project's overall communication capabilities.

## Internet Connectivity

* **Router**: A reliable internet connection is necessary for efficient data transmission between the Arduino device, ThingSpeak, and the MQTT broker. A stable connection is paramount for the project's real-time responsiveness.
* **Network Time Protocol (NTP):** NTP is utilised to maintain real-time accuracy on the Arduino. This ensures synchronised and precise timestamps for temperature updates, contributing to the overall effectiveness of the project.
* **Domain Name Server (DNS):** DNS serves as a name resolution service, converting human-readable domain names into IP addresses. In the context of the project, DNS enables the Arduino to connect to cloud services and other network entities using readable network addresses.
* **Dynamic Host Configuration Protocol (DHCP):** DHCP is employed to dynamically assign IP addresses to the Arduino device. This automated process simplifies network configuration, allowing the Arduino to operate seamlessly on a Wi-Fi network without manual IP address assignments.

## Data Visualisation

* **ThingSpeak Platform:** ThingSpeak is selected as the project's cloud platform for database management, data visualisation, analytics, and storage (*IoT Analytics - ThingSpeak Internet of Things*). Its user-friendly interface simplifies these processes. A dedicated channel in ThingSpeak is set up to receive and analyse data sent from the Arduino. The platform acts as the central hub for data storage and analytics, providing real-time insights through its intuitive user interface (see Figure 4).

ThingSpeak organises data into channels, acting as containers for specific IoT devices. Each channel comprises fields representing different data types. Three fields were used which were, temperature time (the time at which the temperature sample was taken against the time it was received on ThingSpeak), temperature (the temperature at a given time), and temperature state (0 for ok, 1 for hot, and 2 for cold). The Arduino program utilises ThingSpeak APIs and structures data to align with ThingSpeak's channel and field architecture. This ensures seamless and efficient organisation and retrieval of temperature-related information.

## Additional

* **Responsive Environment:** The project incorporates a responsive environment that dynamically adjusts to temperature variations. This adaptability activates the LED display to communicate changes, enhancing user interaction and the project's overall environmental responsiveness.

# System Design

The architecture of the IoT-based temperature monitoring system revolves around the seamless operation of the Arduino program, orchestrating real-time temperature monitoring and visuali***s***ation through a variety of communication channels. The system design is detailed below, outlining key components and their interactions (see Figure 5).

## IoT Device Components

1. **MQTT Publisher**: The Arduino is configured as an MQTT publisher, responsible for sending temperature and time updates to the MQTT broker.
2. **HTTP Client**: The Arduino serves as an HTTP client, leveraging the HTTP service supported by ThingSpeak for the seamless transfer of temperature data to designated channels.
3. **NTP Client**: Utilising the Network Time Protocol (NTP), the Arduino synchronises its internal clock with an NTP server, ensuring accurate and real-time timekeeping, displayed alongside temperature updates when monitored through MQTT.

## Network Infrastructure

1. **Router**: Ensures a stable internet connection via Wi-Fi is maintained for efficient data transmission.
2. **DNS**: Provides domain name resolution services, converting human-readable domain names into IP addresses for connection to cloud services and other network entities.
3. **DHCP**: Dynamically assigns IP addresses to the Arduino, simplifying network configuration and enabling seamless operation on a Wi-Fi network without manual IP assignments.

## Cloud Services

1. **MQTT Broker**: Facilitated by Mosquitto, the MQTT broker establishes machine-to-machine (M2M) connectivity. The Arduino, configured as an MQTT publisher, sends temperature and time updates to the broker, allowing other devices such as PCs to subscribe and receive real-time information.
2. **HTTP Service**: HTTP, supported by ThingSpeak, is used for communication between the Arduino and the cloud platform. It facilitates the seamless transfer of temperature data to ThingSpeak channels. The Arduino program incorporates HTTP protocols to send and receive temperature data, leveraging ThingSpeak's APIs for robust interaction.
3. **NTP Service**: Network Time Protocol (NTP) is utilised to acquire the current time, which is then displayed alongside the temperature when monitoring the device via MQTT. The Arduino program synchronises its internal clock with an NTP server, ensuring accurate and real-time timekeeping alongside temperature updates.
4. **SMS Provider**: The system integrates with an SMS provider, acquiring data from the MQTT broker via a Python monitoring module. This data is used to send SMS notifications to designated contacts, providing crucial alerts through mobile phone providers.

## PCs

1. **Data Visualisation**: PCs act as recipients of temperature data, transforming raw information into a user-friendly visual format, providing a clear representation of temperature variations.
2. **MQTT Subscriber**: The PC establishes a connection to the MQTT broker by subscribing. Through this subscription, the PC receives temperature updates in real-time, ensuring an up-to-date representation of environmental conditions.
3. **Carer Monitor**: The Carer Monitor, a Python module, actively retrieves data from the MQTT broker and forwards it to the SMS provider for notifying designated contacts. This ensures timely communication and enhances the overall responsiveness of the system.

# Project Testing phase

The testing phase is a crucial step to ensure the reliability and functionality of the IoT artifact. To comprehensively assess different aspects of the system, follow these steps:

1. For testing purposes, the temperature thresholds have been set to the following:
   * Hot >= 25℃
   * Cold <= 24℃
   * 25℃ > OK > 24℃
2. Ensure that VS Code is open, and the Arduino is connected to a PC via a USB connection.
3. Open the serial monitor and configure the settings correctly (see Figure 12).
4. Initiate the 'start monitoring' option to observe and display all processes executed by Arduino, with periodic updates sent to the terminal.

Upon successful connection, proceed with the following tests.

## Hardware Integration Testing

1. **Compatibility Verification:**

* Confirm the seamless integration of Arduino components, including the temperature sensor, LCD display, and connectivity modules by ensuring successful firmware programming onto the Arduino (see Figure 13).

1. **Hardware Connection Confirmation:**

* Verify the successful connection of hardware components by monitoring the serial monitor for periodic real-time updates of Arduino temperature samples (see Figure 14).

1. **LCD Display**: 
   1. Power Connection Test:
   * Power the Arduino using either the USB connection or a battery.
   * Confirm that the LCD lights up, displaying a light blue colour, indicating a successful power connection (see Figure 15).
   1. Temperature Threshold Variation:
   * Observe the LCD display as the temperature varies.
   * Confirm that the screen remains green for an acceptable temperature, turns red for too hot, and blue for too cold based on the programmed thresholds (see Figure 15).
   1. Heating Test:
   * Use a hairdryer to heat up the temperature sensor.
   * Verify that the LCD display changes to red, indicating the temperature is too hot (see Figure 15).
   1. Cooling Test:
   * Place ice cubes in a bowl and position them next to the sensor.
   * Confirm that the LCD display changes to dark blue, indicating the temperature is too cold (see Figure 15).
2. **Temperature Sensing Accuracy:**
3. Controlled Environment Setup:
   * Create a controlled environment with a stable temperature.
   * Use a thermometer to measure the temperature of the temperature sensor and set the reference temperature (see Figure 16 and Table 1).
4. Comparison Test:
   * Place the temperature sensor in the controlled environment.
   * Compare the sensor readings with the known reference temperature.
   * Verify that the sensor readings are within ± 1.5℃ of the reference temperature (see Figure 16 & Table 1).
5. Display Verification:
   * Check the temperature displayed on both the terminal of the serial monitor and the LCD display.
   * Confirm that the displayed temperature matches the sensor readings in the controlled environment (see Figure 16 and Table 1).

## Network Connectivity Test

1. **Wi-Fi Connection Check:**
   * Verify the device's capability to connect to the internet.
   * Integrate error handling in the Arduino code to display on the LCD and serial monitor if no Wi-Fi connection is available.
2. **Serial Monitor Verification:**

* Open the serial monitor to check and confirm a successful Wi-Fi connection.
* Ensure that the terminal in the serial monitor displays the status of the Wi-Fi connection accurately.

1. **Router Status Test:**

* Turn off the router to simulate no internet connection.
* Confirm that the serial monitor indicates the device is not connected to the internet (see Figure 17).

1. **Router Activation Test:**

* Turn on the router to establish an internet connection.
* Verify that the serial monitor displays the successful establishment of an internet connection through the desired router (see Figure 17).

1. **Error Display on LCD:**

* Confirm that the LCD display shows relevant text if no Wi-Fi connection can be found (see Figure 15).

## Machine-to-Machine (M2M) Connectivity Test

1. **MQTT Subscriber Test:**
   * Open a command prompt and enter the command in the Mosquitto program: **mosquitto\_sub -h test.mosquitto.org -t warmth-checker**.
   * Verify periodic updates display the current temperature detected by the Arduino, along with corresponding timestamps (see Figure 18).
2. **VS Code Serial Monitor Check:**

Inspect the serial monitor in VS Code to confirm:

* + - Successful connection to the MQTT broker and NTP server (see Figure 19).
    - Updates received by the Arduino from the NTP server (see Figure 14).
    - Confirmation when the Arduino publishes to the MQTT broker.

## Data Storage and Analytics Verification

1. **ThingSpeak Functionality Check:**

* Confirm proper ThingSpeak functionality by monitoring the Arduino via the serial monitor.
* Upon successful data transfer to ThingSpeak, a confirmation line will be printed to the terminal (see Figure 14).

1. **Online Visualisation Validation:**

* Visit the ThingSpeak website at 'https://thingspeak.com' and navigate to the designated channel for the IoT artifact.
* Verify the correct display of graphs presenting the three fields: temperature time, temperature, and temperature state (see Figures 7, 8, 9, 10, 11).

## SMS Notification System Verification:

1. **Carer Monitor Execution:**

* Open the terminal in VS Code and navigate to the correct folder of the .py script.
* Execute the Python script simulating the carer monitor using the command 'python warmth\_monitor.py' (see Figure 20).

1. **Monitoring Data Updates:**
   * Verify periodic updates in the terminal, indicating data sent from the Arduino to the MQTT broker (see Figure 20).
2. **Critical Temperature Threshold Alerts:**

* Observe the terminal for a change in the temperature state to 'too hot' or 'too cold'.
* Confirmation lines in the terminal should indicate the initiation of an SMS message (see Figure 20).

1. **SMS Notification Confirmation:**
   * Receive a text message on the designated mobile phone following a change in temperature state (see Figure 21).

## User Interface Testing

1. **Wi-Fi Connection Display:**
   * Verify the LCD display response to a successful Wi-Fi connection by ensuring it displays the current temperature and state.
   * Confirm that turning off the router triggers the LCD to display 'Not connected' (see Figure 15).
2. **Dynamic Temperature Display:**
   * Observe the LCD display for real-time updates in temperature and state as the environmental conditions change.
   * Confirm the display transitions to 'ok', 'hot', or 'cold' based on predefined temperature thresholds (see Figure 15).

# Legal & Ethical Evaluation

## Legal Considerations

* **Data Privacy**: Ensuring compliance with data privacy regulations will be necessary as the project involves collecting and transmitting data. This means adherence to laws like GDPR, or regional equivalents are important. The project implementation will need to include robust measures to protect user information. (encryption protocols and anonymise data)
* **Intellectual Property**: Consideration must be given to intellectual property rights; it is crucial to verify that the project does not infringe on existing patents or intellectual property rights. To ensure this all code and designs will adhere to open-source or appropriately licensed frameworks.

## Ethical Considerations

* **Informed Consent**: Users, especially the elderly and caregivers, should provide informed consent regarding data collection and usage. Clear documentation and consent forms for users will be necessary to provide transparent communication about the purpose of the system and the data it gathers fosters ethical practices.
* **Accessibility**: Ensuring the technology is accessible to individuals with diverse abilities promotes inclusivity. The project should consider factors like user interfaces suitable for those with impaired vision or hearing. To do this the project will utilise bright colour indicators of temperature as well as SMS notifications to users when there are significant temperature changes. Future alterations of the IoT device could include voice commands for users as well as a user-friendly interface for users to adjust of the temperature thresholds for the temperature sensor on the device as well as the Wi-Fi connection.

## Commercial & Economic Context

* **Cost-Effectiveness**: Ensuring that the technology remains affordable is paramount as it will help contribute to the device’s widespread use. Affordability will enhance accessibility particularly for those with limited financial resources. This device has been developed using affordable components from the Grove starter kit as well as a low-cost Arduino board. (Conduct a cost-benefit analysis considering both initial implementation costs and long-term maintenance expenses)

## Sustainability & Equality, Diversity, and Inclusion (EDI) Issues

* **Environmental Impact**: Evaluating the environmental impact of the project is crucial. Minimising energy consumption and utilising eco-friendly components contribute to the overall sustainability of the system. (Evaluate the environmental impact of the IoT device, considering its lifecycle.
* **Cultural Sensitivity**: Consideration of cultural differences is essential to ensure the project is respectful and inclusive. Adapting the system to diverse cultural norms and practices demonstrates a commitment to diversity.
* **Equitable Access**: Striving for equitable access ensures that the benefits of the project are not restricted to a specific demographic. Addressing socio-economic factors and providing solutions for various user groups contribute to a more inclusive system.
* **User Representation**: Ensuring diverse representation in the development process prevents biases and promotes a system that caters to a broader spectrum of users. Different perspectives contribute to the creation of a more inclusive and effective artifact.

# Conclusion

In summary, this project addresses the need for real-time monitoring. Using simple and smart technology, the system adapts to temperature changes, making life better for elderly individuals while also aiding caregivers. By combining easy-to-use hardware and software, this project not only solved a specific problem but also contributed to discussions on responsible technology use for elderly care.

Attention has been made to legal and ethical concerns, making sure the system respects privacy and intellectual property. It's a reliable, user-friendly solution that considers accessibility and fairness for everyone. The project successfully brings together different technologies, like SMS notifications and data visualisation and overall is a practical and impactful solution that improves the quality of life of an aging population, showing the positive side of innovative technology.

# Source code

https://github.com/JessicaMu/IoT\_temperature\_checker

# Reference

*Arduino Reference - Arduino Reference*. Available at: https://www.arduino.cc/reference/en/ (Accessed: 12 January 2024).

*ARDUINO UNO WiFi REV2* *Arduino Official Store*. Available at: https://store.arduino.cc/products/arduino-uno-wifi-rev2 (Accessed: 12 January 2024).

*Cold Weather Safety for Older Adults* (2024) *National Institute on Aging*. Available at: https://www.nia.nih.gov/health/safety/cold-weather-safety-older-adults (Accessed: 12 January 2024).

*Grove - Starter Kit v3 | Seeed Studio Wiki* (2023a). Available at: https://wiki.seeedstudio.com/Grove\_Starter\_Kit\_v3/ #grove---base-shield (Accessed: 12 January 2024).

*Grove - Starter Kit v3 | Seeed Studio Wiki* (2023b). Available at: https://wiki.seeedstudio.com/Grove\_Starter\_Kit\_v3/ #grove--temperature-sensor (Accessed: 12 January 2024).

*Grove - Starter Kit v3 | Seeed Studio Wiki* (2023c). Available at: https://wiki.seeedstudio.com/Grove\_Starter\_Kit\_v3/ #grove---lcd-rgb-backlight (Accessed: 12 January 2024).

Iancu, I. and Iancu, B. (2020) ‘Designing mobile technology for elderly. A theoretical overview’, *Technological Forecasting and Social Change*, 155, p. 119977. Available at: https://doi.org/10.1016/j.techfore.2020.119977.

*IoT Analytics - ThingSpeak Internet of Things*. Available at: https://thingspeak.com/ (Accessed: 12 January 2024).

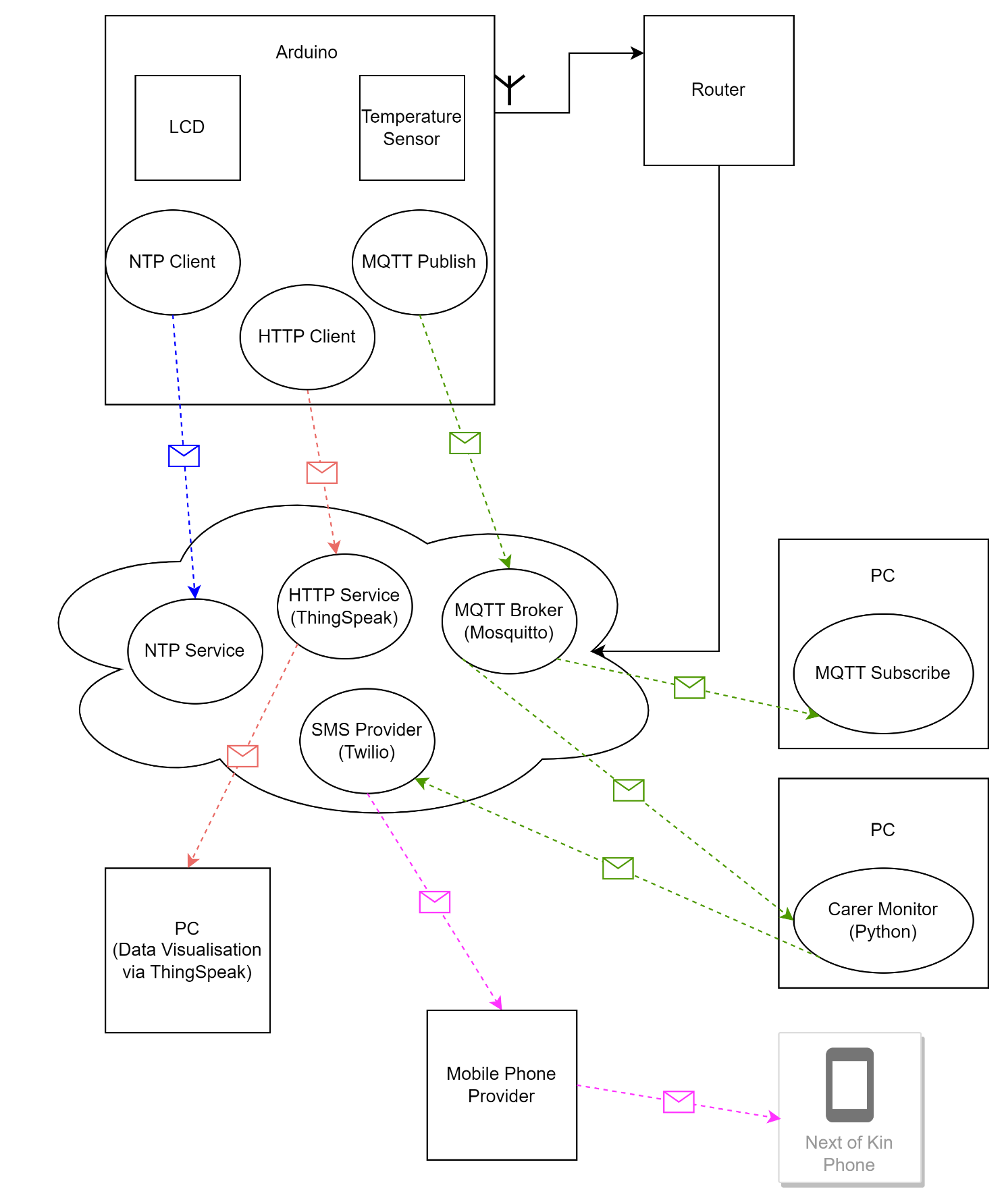
Selvaraj, S. and Sundaravaradhan, S. (2019) ‘Challenges and opportunities in IoT healthcare systems: a systematic review’, *SN Applied Sciences*, 2(1), p. 139. Available at: https://doi.org/10.1007/s42452-019-1925-y.

Stavropoulos, T.G. *et al.* (2020) ‘IoT Wearable Sensors and Devices in Elderly Care: A Literature Review’, *Sensors*, 20(10), p. 2826. Available at: https://doi.org/10.3390/s20102826.

*Winter deaths report: how does cold weather affect mortality in the UK?* (2022) *PPP Taking Care*. Available at: https://taking.care/blogs/resources-advice/winter-deaths-report-does-cold-weather-affect-mortality-in-uk (Accessed: 12 January 2024).

# Appendix

## Project Architectural Diagrams



1. A project implementation diagram.

A white cloud with black text

Description automatically generated

1. A project implementation diagram demonstrating M2M via an MQTT broker and subscriber.

A white clouds with black text

Description automatically generated

1. A project implementation diagram demonstrating M2M via MQTT broker and subscriber including SMS notification .

A white cloud with black text

Description automatically generated

1. A project implementation diagram demonstrating data visualisation.

A diagram of a computer

Description automatically generated

1. A system design diagram.

## Circuit Design

A computer screen shot of a computer

Description automatically generated

1. The circuit diagram displaying the components used for the IoT artefact.

## Data Analytics

A screen shot of a graph

Description automatically generated

1. This displays the time at which a temperature sample was taken compared to the time at which the temperature sample was received by ThingSpeak.

A screenshot of a computer

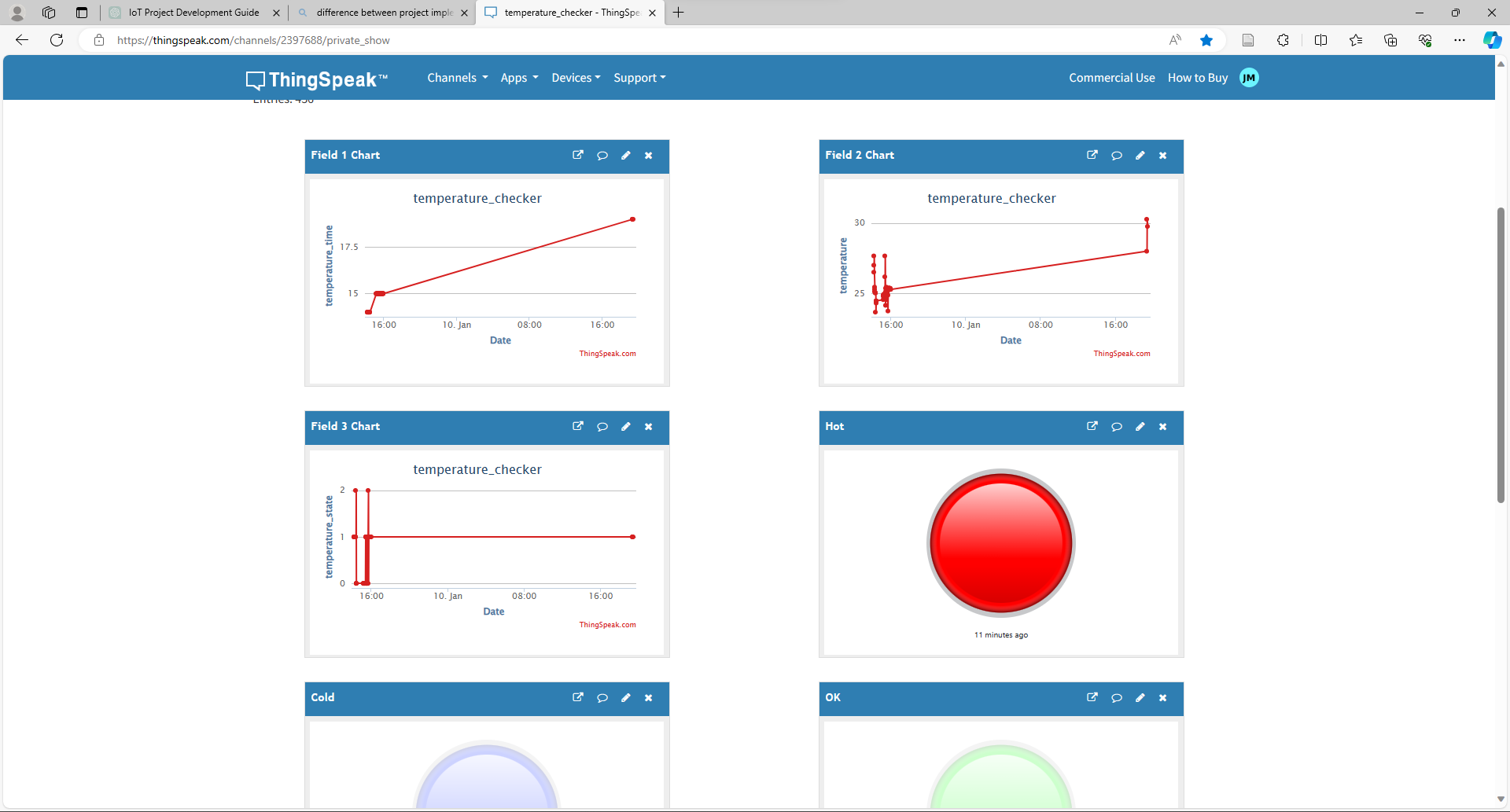
Description automatically generated

1. This displays the temperature sample alongside the time at which the temperature sample was taken.

A screenshot of a computer

Description automatically generated

1. This displays the temperature sample alongside the time at which the temperature sample was taken.



1. Figure 9. A quick visualisation of the state. Red for hot, blue for cold, green for ok.



1. A quick visualisation of the temperature sample as well as the temperature samples alongside their state. Red for hot, blue for cold, green for ok.

## Testing Phase

A screenshot of a computer

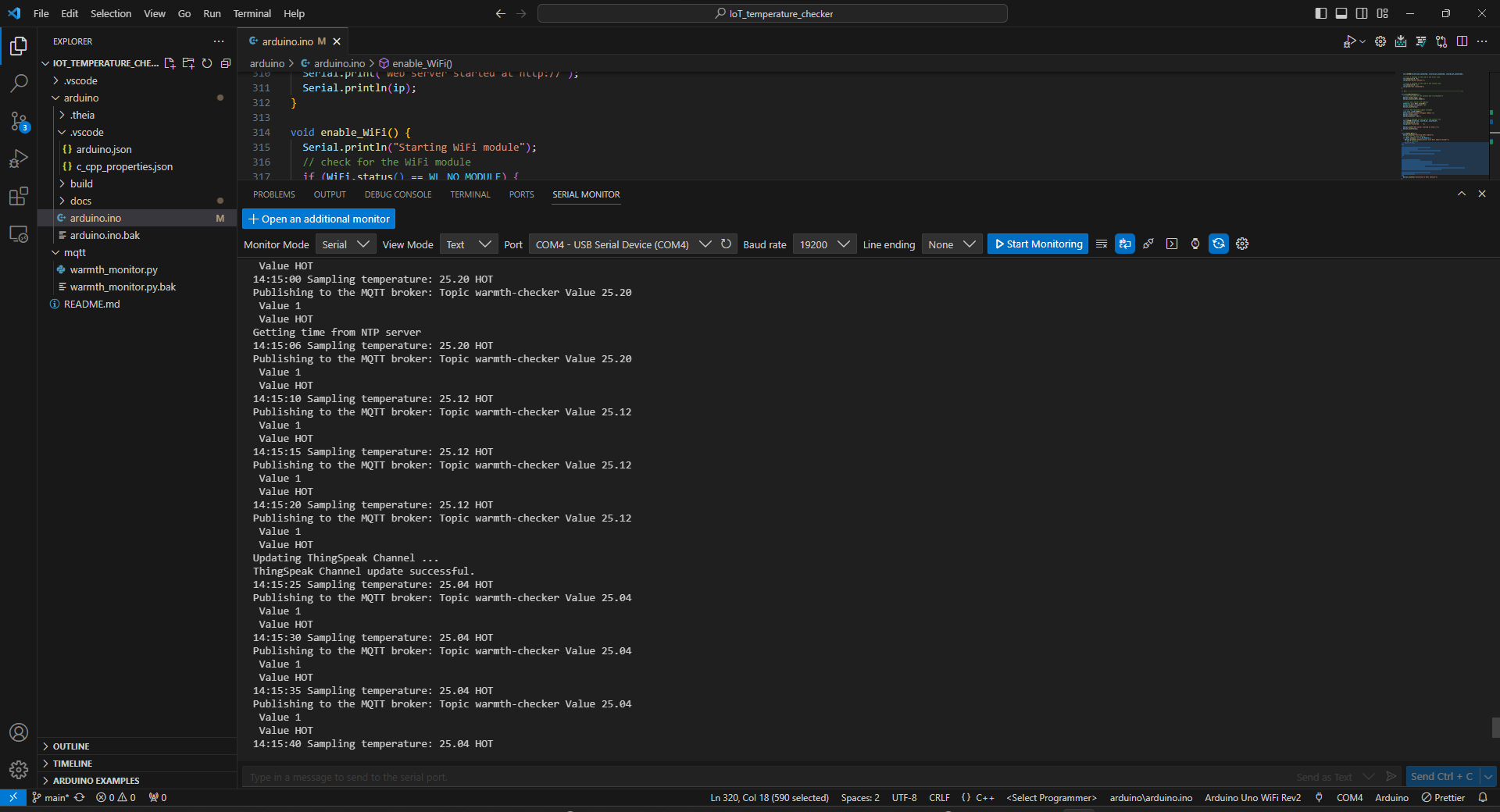
Description automatically generated

1. VS Code serial monitor settings.

A screenshot of a computer

Description automatically generated

1. VS Code output terminal displaying successful firmware upload to the Arduino.



1. Periodic updates of the Arduino temperature sample and successfully updating ThingSpeak as well as getting NTP server updates.

*A blue rectangular electronic device with a blue screen

Description automatically generatedA digital thermometer and a screwdriver

Description automatically generatedA digital thermometer and a blue screen

Description automatically generatedA digital thermometer and a digital thermometer

Description automatically generated*

1. LCD displaying light blue for no internet connection, green for ok temperature state, red for hot and dark blue for cold.

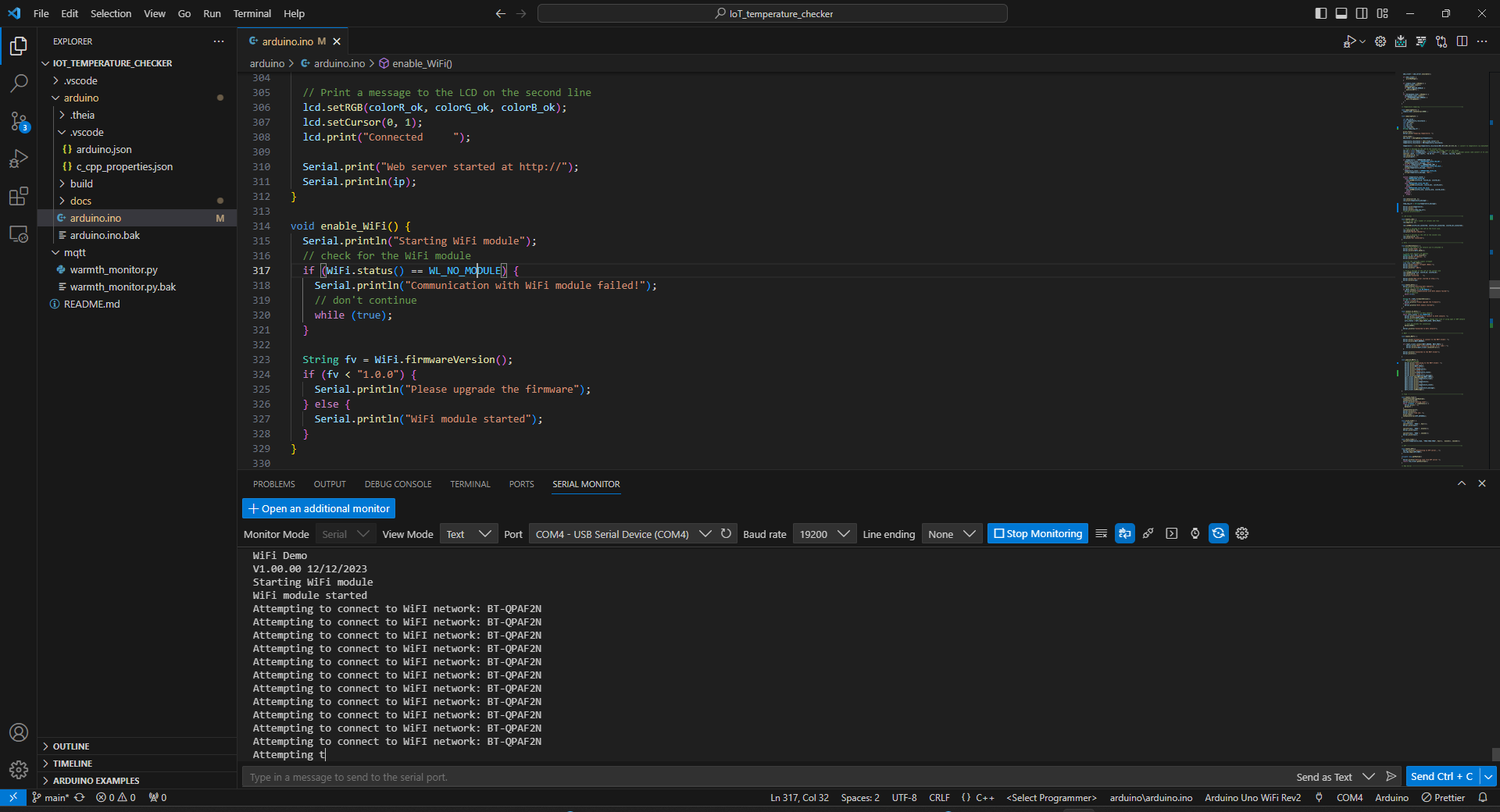
*A screenshot of a computer

Description automatically generated*

1. The environment temperature detected by a thermometer. The temperature sensor sample can be seen via the VS Code serial monitor and LCD Display (see Table 1).

|  |  |  |  |
| --- | --- | --- | --- |
| **State** | **Thermometer (℃)** | **VS Code Serial Monitor (℃)** | **LCD Display (℃)** |
| **OK** | 25.3 | 24.08 | 24.8 |
| **Cold** | 25.0 | 23.84 | 23.84 |
| **Hot** | 26.3 | 25.06 | 25.6 |
|  |

1. Table showing temperature sensor readings taken from the Arduino compared to a thermometer.



1. VS Code serial monitor displaying unsuccessful connection to a Wi-Fi network

A screenshot of a computer

Description automatically generated

1. VS Code serial monitor displaying successful connection to a Wi-Fi network, NTP server and MQTT broker.

A screenshot of a computer

Description automatically generated

1. Command Prompt displaying successful subscription to MQTT broker showing real time temperature samples.

A screenshot of a computer program

Description automatically generated

1. VS Code terminal displaying successful python subscription to MQTT broker showing real time temperature samples. Additionally displaying successful messaging of ‘next of kin’.

A screenshot of a chat

Description automatically generated

1. Desired mobile phone being notified of temperature state change via SMS notification.