# Introduction

[insert introduction]

## Problem Definition

The demographic landscape is undergoing a transformative shift globally, with an increasing aging population requiring specialised attention and care. 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. [insert reference] 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. [insert reference]

Research shows that traditional monitoring systems often fall short in providing a seamless, unobtrusive solution tailored to the unique needs of the elderly. [insert reference] 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. [insert reference]

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 break down the project requirements.

* IoT Device
  + Temperature Sensor
  + Backlight Display
  + IoT Software
* Utilise some aspect of machine two machine (M2M) connectivity.
* Use internet connectivity.
* 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).

## IoT Device

### Hardware

* **Arduino Uno WIFI R2 Board:** This board will provide seamless integration of IoT sensors, providing the necessary computational power for data processing and transmission.
* **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-LCD RGB Backlight Display v5.0:** This will provide a clear visual representation of temperature variations. The visual feedback enhances user comprehension and interaction.
* **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.
* **Code Design**: The code design will include configurations to establish internet connectivity using Domain Name System (DNS), ensuring accurate time updates through Network Time Protocol (NTP), communication with cloud services via MQTT with Mosquitto, sending data updates to ThingSpeak, and obtaining network settings seamlessly through Dynamic Host Configuration Protocol (DHCP). These configurations are vital for the project's functionality.

## 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 (see Figure 2).

## 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 visualization, analytics, and storage. 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 3).

## 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 Arduino program is designed to monitor and visualize temperature data, providing real-time updates through various communication channels. The following breaks down the Arduino program operation and system design (see Figure 4).

## IoT Device - Temperature Thresholds and LCD Display

* **Functionality:** 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.
* **Implementation:** The Arduino program dynamically adjusts the LCD display colour based on predefined temperature thresholds.

## NTP Service

* **Functionality:** Network Time Protocol (NTP) is utilised to acquire the current time, which is then displayed alongside the temperature when monitoring the device via MQTT.
* **Implementation:** The Arduino program synchronises its internal clock with an NTP server, ensuring accurate and real-time timekeeping

## MQTT Broker

* **Functionality:** An MQTT broker, facilitated by Mosquitto, is utilised for machine-to-machine (M2M) connectivity. The Arduino sends temperature and time updates to the broker, allowing other devices (e.g., a PC) to subscribe and receive real-time information.
* **Implementation:**
  + MQTT Broker: The Arduino program establishes MQTT communication, periodically sending updates to the broker.
  + MQTT Subscriber: 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.

## HTTP Service

### Data Visualisation

* **Functionality:** ThingSpeak serves as a cloud platform for data organisation, visualisation, and analytics. The Arduino sends temperature updates to ThingSpeak, where data is organised into channels and fields.
* **Implementation:** The Arduino program utilises ThingSpeak APIs to seamlessly send temperature data to designated fields within specific channels. Each channel represents a unique IoT device or sensor.

### Channel and Field Organization

* **Functionality:** 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).
* **Implementation:** The Arduino program structures data to align with ThingSpeak's channel and field architecture, ensuring efficient organisation and retrieval of temperature-related information.

### HTTP Communication

* **Functionality:** 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.
* **Implementation:** The Arduino program incorporates HTTP protocols to send and receive temperature data, leveraging ThingSpeak's APIs for robust interaction.

In summary, the Arduino program intelligently manages temperature data, adjusting LCD display colours, synchronising time through NTP, establishing M2M connectivity via MQTT, and visualising data on ThingSpeak through HTTP communication.

# Circuit Design

# Project Testing phase

# Data Analytics

# Legal & Ethical Evaluation

## Legal Considerations

Data Privacy:

* Requirement: Ensure compliance with data privacy regulations to protect the personal information collected through the IoT device.
* Implementation: Implement encryption protocols and anonymize data to adhere to regulations such as the General Data Protection Regulation (GDPR) (European Commission, 2016).

Intellectual Property:

* Requirement: Verify that the project does not infringe on existing patents or intellectual property rights.
* Implementation: Conduct thorough patent searches and ensure that all code and designs adhere to open-source or appropriately licensed frameworks.

## Ethical Considerations

Informed Consent:

* Requirement: Ensure users are adequately informed about the purpose and functionality of the IoT device.
* Implementation: Provide clear documentation and consent forms for users, emphasizing transparency in data collection and usage.

Accessibility:

* Requirement: Design the IoT device to be accessible to all users, including those with disabilities.
* Implementation: Incorporate features such as voice commands or alternative interfaces to enhance accessibility (World Health Organization, 2011)

## Commercial & Economic Context

Cost-Effectiveness:

* Requirement: Evaluate the economic feasibility and cost-effectiveness of the IoT solution.
* Implementation: Conduct a cost-benefit analysis considering both initial implementation costs and long-term maintenance expenses.

Market Viability:

* Requirement: Assess the potential market demand and viability of the developed IoT solution.
* Implementation: Conduct market research to identify user needs and preferences, ensuring alignment with the commercial market.

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

Environmental Impact:

* Requirement: Evaluate the environmental impact of the IoT device, considering its lifecycle.
* Implementation: Choose materials and manufacturing processes with lower environmental impact and design for energy efficiency.

Inclusivity:

* Requirement: Ensure the IoT device caters to a diverse user base, considering cultural and demographic variations.
* Implementation: Design interfaces and user experiences that are culturally sensitive and inclusive (World Health Organization, 2013).

# conclusion

# source code

# reference

# appendix

A diagram of a cloud

Description automatically generated

Figure 1. A project implementation diagram.

A white clouds with black text

Description automatically generated

Figure 2. A project implementation diagram demonstrating M2M.

A white cloud with black text

Description automatically generated

Figure 3. A project implementation diagram demonstrating data visualisation.

A diagram of different types of computer components

Description automatically generated with medium confidence

Figure 4. A system design diagram.