Optimizing Environmental Surveillance: A Cloud-Centric Approach to Real-Time Temperature Monitoring

Balaji Prasad Padhi, Rasmita Rani Panda,
Bruti Narayana Ghadadi, Chandrakanta Patra, Abinash Bindhani, Suraya Prakash Malla,
Chiranjib Mohanty, Amit Mondal

Assistant Professor, EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Assistant Professor,EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student, EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student,EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student, EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student,EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student, EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India
Student,EEE, GIFT(Autonomous),Bhubaneswar, Odisha, India

ABSTRACT

In today's digital age, monitoring environmental factors such as temperature is indispensable across diverse industries, spanning from healthcare to food storage. This abstract introduces a sophisticated cloud-based temperature monitoring system meticulously crafted to deliver real-time data collection, analysis, and management capabilities. The system harnesses IoT (Internet of Things) sensors strategically deployed within the monitored environment to ensure a continuous stream of temperature data. These sensors are seamlessly integrated with a cloud platform, facilitating secure transmission and storage of the collected data.The architecture of the cloud-based system offers a myriad of benefits, including scalability, accessibility, and reliability. Through its scalable nature, the system can effortlessly accommodate fluctuations in data volume and user requirements. Moreover, stakeholders can access the temperature data from anywhere at any time, thanks to the system's inherent accessibility features. Additionally, the reliability of the cloud-based infrastructure ensures the seamless operation of the temperature monitoring system without compromising data integrity or system performance.Furthermore, the system incorporates advanced analytics algorithms designed to detect anomalies and forecast potential temperature fluctuations. This proactive approach to temperature management empowers stakeholders to preemptively address issues before they escalate, thereby minimizing risks and optimizing operational efficiency.The user interface of the system is designed to be intuitive and user-friendly, enabling stakeholders to access temperature data remotely via web or mobile applications. This accessibility facilitates informed decision-making and prompt responses to emerging situations, thereby enhancing overall operational effectiveness.Moreover, the system is designed to seamlessly integrate with existing enterprise systems, thereby augmenting operational efficiency and streamlining decision-making processes. By leveraging the synergies between the cloud-based temperature monitoring system and existing infrastructure, stakeholders can maximize the utility of their resources while capitalizing on the system's advanced functionalities.

Key Words: Digital age, Environmental monitoring, Temperature monitoring, Cloud-based system Real-time data collection, IoT sensors, Cloud platform integration, Advanced analytics algorithms, Anomaly detection, Remote access, Integration with existing systems, Synergies, Resource optimization, Functionality

1. INTRODUCTION

The aim of this thesis work is to optimize temperature monitoring and control, in terms of speed, simplicity, and accuracy, using a simple open sourced hardware and software; this will help in illustrating best practices to be kept under consideration while using such environments. Within this chapter, the theories behind the methods used are explained, and then the hardware and software components are introduced. This project can be used by developers in the field of software development, embedded systems that would like to migrate their work into the Arduino environment. The work can also be used by amateur programmers and first time users of the Arduino environment to start their own embedded systems projects. Due to the time limitations on this project, heat sinks are not taken into consideration as a temperature control mechanism. Also, Java and other programming languages that can be used to increase the attractiveness and the readability of the web interface, and the security of the webpage were not considered during the project.

1.1 Temperature Monitoring

The need for sophisticated and robust temperature monitoring systems is increasing, especially for businesses and organizations within the healthcare, food products, and electronics sectors. Such organizations utilize temperature monitoring technologies to monitor the temperatures of their products and processes; this is especially important to safeguard their products and meet regulatory standards within the region they are situated in. A sophisticated and robust temperature monitoring systems can be defined as systems that include and integrate the following components:

- Relatively accurate digital probes
- Thermal buffers
- Flexible and reprogrammable measurement device
- Data storage and representation
- Well-developed software.
- Alarming: Indication of abnormalities. Temperature monitoring can be also part of preventative reliability. This is important when a system is not performing high temperature processes, yet can be at the risk of overheating.

1.2 Digital Probes: Temperature Sensors

Industries can use the sensors for cold chain integrity, medical monitoring, equipment monitoring, and environmental monitoring. There are several kinds of temperature sensors that are available in the market; thermocouples, RTD's, thermistors, and semi-conductor based sensors. The different types of temperature sensors can be utilized for different applications within different sectors due to their different features in terms of responsiveness, accuracy, and temperature ranges. A semi-conductor based temperature sensor (SMT160-30), shown in Figure 1, is used

Figure 1:SMT16030 TO-92 casing temperature sensor

The temperature sensor has several features that were useful in the project, such as its energy efficiency, wide temperature range, low noise levels, low current, long term stability, direct interface with microcontrollers, and the small packaging.

1.3 Flexible and Reprogrammable Environments

In this project, different types of environments were researched to identify cheap and easy hardware and software to implement the project and finally get the results desired. The researched environments were:

- Pinguino PIC32
- STM32
- MPS430 LaunchPad
- Arduino Uno
- Processing™

Yet, only two of the above environments, Arduino and Processing™, were used in the project to implement the temperature monitoring device. Arduino is relatively cheap to buy, has a well-rounded community to help programmers, abundant in Europe, and features an open-source IDE with a wide range of libraries. While processing is completely free-to-use, and also features a community that assists programmers at all levels in their projects.More details following.

1.4 Arduino ide

The Arduino Integrated Development Environment (IDE) is a software application used to write, compile, and upload code to Arduino microcontroller boards. Here's a detailed overview of the Arduino IDE:

- Graphical User Interface (GUI):

The Arduino IDE features a user-friendly graphical interface designed to simplify the process of writing and uploading code.It consists of various panels and windows, including the main code editor, toolbar, message console, and status bar.

- Code Editor:

The central component of the Arduino IDE is the code editor where users write their Arduino sketches (programs).The editor provides syntax highlighting, auto-indentation, and code completion features to assist with writing code.Arduino sketches are written in C/C++ programming language, although the IDE abstracts many complexities to make it more accessible to beginners.

- Toolbar

The toolbar contains buttons for common actions such as compiling, uploading, verifying code, opening sketches, and saving sketches.It also includes options to select the Arduino board type, communication port, and serial monitor settings.

- Message Console

The message console displays feedback messages, compilation errors, warnings, and status updates during code compilation and uploading.It provides information about the progress of the compilation process and highlights any errors or warnings that need attention.

- Status Bar:

The status bar at the bottom of the IDE displays information about the current state of the Arduino IDE, such as the version number, line and column number in the code editor, and memory usage.

- Sketches:

In the Arduino IDE, code files are referred to as "sketches."

A sketch typically consists of two main functions: **setup()** and **loop()**. The **setup()** function is called once when the Arduino board starts, while the **loop()** function runs continuously in a loop after the **setup()** function completes.

Users can create, open, save, and manage sketches directly within the IDE.

- Library Manager:

The Arduino IDE includes a library manager that allows users to easily install and manage libraries (collections of pre-written code) for additional functionalities.Users can search for libraries, install new libraries, and update existing ones through the Library Manager interface.

- Serial Monitor:

The Arduino IDE provides a built-in serial monitor tool that allows users to communicate with the Arduino board via serial communication.It enables users to send data to the Arduino board and receive data from it, making it useful for debugging and monitoring applications.

- Board Manager:

The Board Manager feature allows users to install and manage board definitions for different Arduino-compatible boards.Users can select their target Arduino board from a list of supported boards and specify the appropriate processor, clock frequency, and other parameters.

- Preferences:

The Preferences menu in the Arduino IDE allows users to customize various settings such as font size, theme (dark or light), language, and additional board manager URLs.

1.5 LCD Interface

LCD panel consists of two patterned glass panels in which crystal is filled under vacuum. The thickness of glass varies according to end use. Most of the LCD modules have glass thickness in the range of 0.70 to 1.1mm

Normally these liquid crystal molecules are placed between glass plates to form a spiral staircase to twist the twist the light. Light entering the top plate twist 900 before entering the bottom plate. Hence the LCDs are also called as optical switches. These LCD cannot display any information directly. These act as an interface between electronics and electronics circuit to give a visual output.

- Technology:

The liquid crystal display (LCD), as the name suggests is a technology based on the use of liquid crystal. It is a transparent material but after applying voltage it becomes opaque. This property is the fundamental operating principle of LCDs.

An LCD consists of two-glass panel with a cavity in between. The panels are sealed together. The inner surface of glass is coated with transparent material to form characters or symbols for display. The most common type of liquid crystal used is ‘nematis’. In this type of crystal the long rod type molecules are arranged in parallel. It changes the optical characteristics with change in direction by applying voltage to it.There are two common type of LCDs which use this material. They are :

- TN (Twisted nematic):

The twisted nematic field effect mode arranges the liquid crystal molecules by controlling their movement Witt electric voltage soaks twist them by 900 in the direction for their thickness. It controls the light passing through the polarized placed on the two plates of the LCD by controlling the movements of the liquid crystal molecules. Almost all the medium and small type segment LCD are these types. Hence this type is most common type used.

- STN (Super twisted nematic)

While the TN mode arranges the liquid crystal molecule by twisting them by 900 the STN effect mode arranges them by giving a still larger twist and provides a display by refringence effect of the liquid crystal. The LCD structure in STN mode is same as that in TN mode. But as it has a different arrangement of liquid crystal, and by bi-refringence effect of liquid crystal. The LCD structure in STN mode is same as that in TN mode but as it has a different arrangement of liquid crystal and bi-refringence effect there is a colour in display and also a background colour. In STN mode, a wide viewing angle is obtained. The STN mode also offers a high contrast display compare to the TN mode. This mode is widely used in large size full dot-matrix LCD modules. For colours it has multiple modes depending on the combination of the polarized and retardation film.

- Energy consumption

LCD normally requires very little energy to operate typically 5μA to 25μA at five volts (per square inch) for a display. In addition, auxiliary lighting will require supplementary energy. ALL LCD require a pure AC drive voltage. Inadvertent DC voltage, such as DC component in an AC signal, can significantly reduce the life of LCDs and must be limited to 50mv DC.

- Direct drive:

Direct drive, static or simplex drive, means that each segment of the LCD has an independent connection to the driver. Direct drive LCD has the highest contrast over the widest temperature ranges. They are widely used in outdoor application. Direct drives typically requires drive frequency between 30HZ and 60HZ frequency. Frequency below 30HZ will flicker the display. While frequency above 60HZ will excessive current draw in the circuit. This is very important for battery mode operation. If voltage frequency across the limit then LCD 'Off' segments can be come in adherently energized. This partial activation of segment is known as cross talk or ghosting.

LCD is available in a verity of model having one to four rows of 8 to 20 characters each. A display with two rows of 16 characters is used for this example project. Almost all aspects of the design can be used with other model of LCD, since the internal structure of the various LCD models are almost same, differencing only in the number of driver chips used. The display module is powered from a 5 V supply.

Connecting an LCD to a micro controller is very simple, requiring either bit or an 8-bit bus. A 4-bit interface saves I/O pins but requires that the command and data be split into 4-bit pieces, which are sent one after the other. Thus the saving in I/O lines comes at the price of more complicated software. To simplify understanding of the software the example uses a 8-bit interface. Three control lines are required in addition to the data line.

Figure 2: LCD Interface

The voltage at the V0 pin adjusts the contrast of the display. Normally this voltage is provided by an adjustable voltage divider.The control line E (Enable) enables or disables the display. When the display is enabling it monitors the value of the other two control lines and interprets the data lines accordingly. When the display is disabled it ignores the status of the other two control lines and places its data line drivers in a high impedance state (tri-state). The data bus can then be used for other purpose. The control line R/W (Read /Write) determines weather data is read from or written to the LCD. Finally, the RS (Register select) line distinguishes between commands and display characters.

- LCD Controller device characteristics

The HD44780 contains 80 bytes of internal RAM called Data display RAM (DDR) that is used for presentation of characters in the LCD. The size of the DDR is independent of the LCD configuration (number of rows and character). For an LCD having two rows of characters, the leftmost character in the first row is assigned to address 0 of the DDR. Each following character position in the first display row is assigned to the next following address in turn until the 40^{th} character location reached, which is assigned to DDR address 27 hex. The character locations in the second row of the display are assigned to DDR address 40 hex through 67 hex. If for example a character is to be written to the third position from the left in the second row of the display, it must be written to DDR, so that for example with a display, it must be written to DDR location 42hex.If the LCD module displays fewer than 40 characters per row, then these are mapped into a 'window' within the DDR, so that for example with a display of 16 characters per line only 16 of the 40 available DDR locations per line can be displayed at one time. The HD44780 supports commands to move this window to the left or to the right to allow various regions of the DDR to be displayed.

In additions to the DDR the HD44780 has a character genator ROM (CGROM) and a character-generator RAM (CGRAM).The CG ROM contains the dot-matrix patterns for the standard(fixed)character set,whilethe CGRAM allows the user to program additional character. Either eight4 X 7-point or four 5 X 10-point characters may be stored on the **CGRAM.**

1.6 LCD Display

- The principles of lcd technology

In this section, we will explain everything ranging from the properties of liquid crystal molecules to the basic principle of display technology by using TN type liquid crystals as an example.

- The parallel arrangement of liquid crystal molecules along grooves

When coming into contact with grooved surface in a fixed direction, liquid crystal molecules line up parallelly along the grooves.

- Natural state

When liquid crystals are sandwiched between upper and lower plates, they line-up with grooves pointing in directions 'a' and 'b,' respectively

The molecules along the upper plate point in direction 'a' and those along the lower plate in direction 'b,' thus forcing the liquid crystals into a twisted structural arrangement./ (figure shows a 90-degree twist) (TN type liquid crystal)

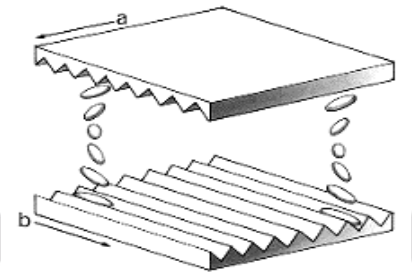

Figure 3 : The light also "twists" as it passes through the twisted liquid crystals

- Light travels through the spacing of the molecular arrangement

Light passes through liquid crystals, following the direction in which the molecules are arranged. When the molecule arrangement is twisted 90 degrees as shown in the figure, the light also twists 90 degrees as it passes through the liquid crystals.

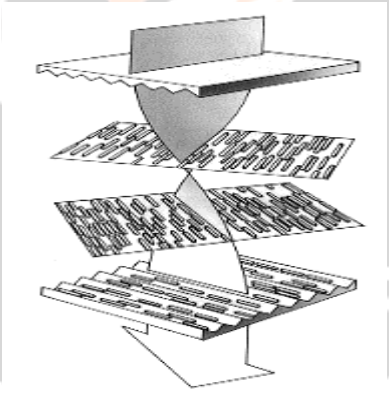

Figure 4: Light bends 90 degrees as it follows the twist of the molecule

- Molecules rearrange themselves when voltage is applied When voltage is applied to the liquid crystal structure, the twisted light passes straight through.

The molecules in liquid crystals are easily rearranged by applying voltage or another external force. When voltage is applied, molecules rearrange themselves vertically (along with the electric field) and light passes straight through along the arrangement of molecules.

- Blocking light with two polarizing filters

When voltage is applied to a combination of two polarizing filters and twisted liquid crystal, it becomes a LCD display.

- Nodemcu

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).[8]. The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson[10] and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects).

2. HISTORY

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9.Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform,and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to the NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In the summer of 2015 the original creators abandoned the firmware project and a group of independent contributors took over. By the summer of 2016 the NodeMCU included more than 40 different modules.

- ESP8266 Arduino Core

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".[18] This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

3. THEORY

As human being there are things that we cannot control and to precisely predict, especially when it comes to natural phenomenon related to weather. As a result, a laboratory room temperature must be kept 20 to 25 Celsius or else the room might warm and may cause a student to be unable to concentrate on their study. Faculty has to choose either to

place a person to monitor the temperature, or to save on human capital by developing a system that can monitor the temperature from other places at any given time. In order to solve the problem, Temperature Room Monitoring System using Internet of Things that can be accessed anywhere and anytime through the Internet is built.

This project use NodeMCU ESP8266 an Arduino based microcontroller where collection of instructions or program are stored for process thus enable this system to properly function and will automatically retrieve the temperature data. The device have a build in WIFI module that enable status communication and allow the system view temperature data via ThingSpeak webserver. This device will integrate with DHT11 sensor temperature and humidity since it is the heart of the system. Temperature sensor device will capture temperature data and send to ThingSpeak server and Blynk cloud. By this method, temperature and humidity data can be monitored over internet, and logged data and graph can be displayed over time on the ThingSpeak dashboard and Blynk application. The main purpose of this project is to make it easy for user to monitor the current temperature and analyse the temperature reading over time graph. This project will be connected with a smartphone device through a mobile application that will be created using Blynk.

3.1 Process & components of cloud based temperature monitoring system

A cloud-based temperature monitoring system typically involves sensors to collect data, a gateway to transmit data to the cloud, cloud infrastructure for data storage and processing, and a user interface for visualization and control.

- Sensors: These are devices that measure temperature and can be placed in various locations such as warehouses, server rooms, or transportation vehicles.
- Gateway: Acts as a bridge between the sensors and the cloud, collecting data from sensors and sending it securely to the cloud platform.
- Cloud Infrastructure: This includes servers, databases, and other resources hosted in the cloud that store the temperature data and perform analysis.
- Data Storage: The temperature data collected from sensors is stored in databases, typically using technologies like SQL or NoSQL databases.
- Data Processing: Analyzing the collected data for trends, anomalies, and insights. This can involve algorithms for predictive maintenance, quality control, or compliance monitoring.
- User Interface**:** A web-based or mobile application that allows users to visualize temperature data in real-time, set alerts for temperature thresholds, and generate reports.
- Security Measures: Encryption, authentication, and access control mechanisms to ensure the confidentiality, integrity, and availability of the data.
- Integration: The system may need to integrate with existing enterprise systems such as inventory management or supply chain systems for a seamless workflow.
- Scalability & Reliability: Cloud-based systems are designed to be scalable to handle large volumes of data and reliable to ensure continuous monitoring without downtime.
- Maintenance & Support: Regular maintenance, updates, and support services to ensure the smooth operation of the system over time.

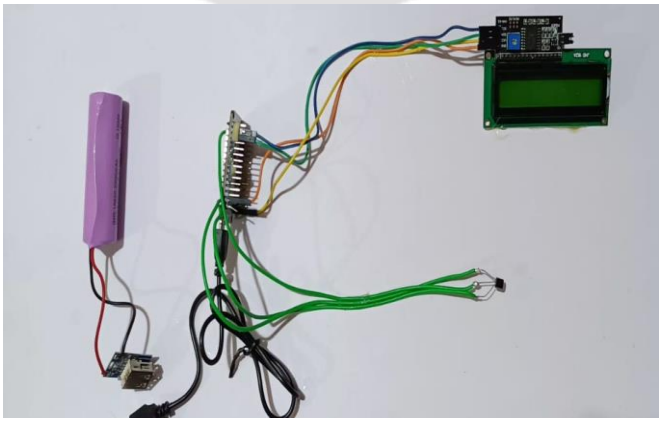

Figure. 3:Temperature Monitoring System

3.2 Blynk IoT

Blynk is an IoT (Internet of Things) platform that allows users to easily create mobile applications to control and monitor various IoT devices. Here are some key details about Blynk:

- Platform: Blynk is a software platform that runs on servers and mobile apps (iOS and Android). It provides a visual drag-and-drop interface for building IoT applications.
- Connectivity: Blynk supports a wide range of hardware including Arduino, Raspberry Pi, ESP8266, and others. It can connect to these devices over the internet using Wi-Fi, Ethernet, or cellular connections.
- Mobile Apps: Blynk provides mobile apps for iOS and Android that allow users to create custom interfaces to control and monitor their IoT devices. These apps can display data, control outputs, and receive notifications from the connected devices.

- Widgets: Blynk offers a variety of widgets that can be added to the mobile app interface, such as buttons, sliders, gauges, displays, and more. These widgets can be easily configured to interact with the connected IoT devices.
- Cloud Services: Blynk provides cloud-based services that handle the communication between the mobile app and the connected devices. This allows users to access and control their devices remotely, even when the devices are behind firewalls or on different networks.
- Scripting: Blynk supports scripting using the Blynk Library, which allows users to write custom code to extend the functionality of their IoT applications.
- Pricing: Blynk offers both free and paid plans, with the free plan providing limited functionality and the paid plans offering more advanced features and higher usage limits.

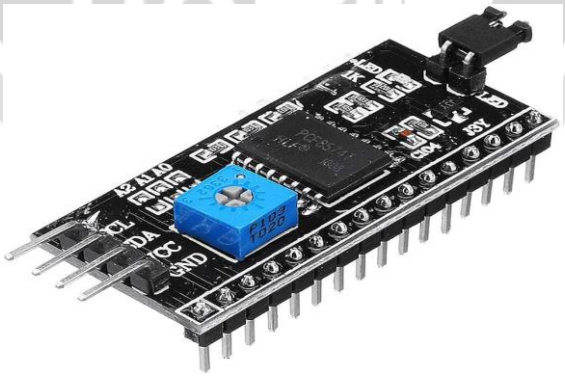

Figure.4: I2C-Inter Integrated Circuit

I2C (Inter-Integrated Circuit) is a communication protocol used for connecting and transmitting data between various electronic components and microcontrollers. Here are the key details about I2C:

- Bus Structure: I2C uses a bus structure, which means multiple devices can be connected to the same set of wires. The bus consists of two main lines:

SCL (Serial Clock Line): This line carries the clock signal.

SDA (Serial Data Line): This line carries the data.

- Master-Slave Architecture: I2C uses a master-slave architecture, where one device (the master) controls the communication and the other devices (the slaves) respond to the master's requests.
- Addressing: Each I2C slave device has a unique 7-bit or 10-bit address, allowing the master to communicate with specific devices on the bus.
- Data Transfer: Data is transferred in a serial fashion, one bit at a time, with the master generating the clock signal. The data is transferred in packets, with a start condition, address, data, and stop condition.
- Synchronization: I2C uses a synchronization mechanism, where the master and slave devices synchronize their clock signals to ensure reliable data transfer.
- Speed: I2C supports three different communication speeds:

Standard mode: Up to 100 kbit/s

Fast mode: Up to 400 kbit/s

High-speed mode: Up to 3.4 Mbit/s

- Pull-up Resistors: I2C requires pull-up resistors on both the SCL and SDA lines to maintain the logic levels when the bus is idle.
- Advantages: I2C is a widely used protocol due to its simplicity, low cost, and the ability to connect multiple devices on the same bus. It is commonly used in embedded systems, sensors, and various electronic devices.
- Applications: I2C is used in a variety of applications, including microcontrollers, real-time clocks, ADCs, DACs, memory chips, and more.

I2C is a robust and flexible communication protocol that allows for easy integration of various electronic components and devices within a system.

3.3 Thingspeak :

Figure 5: Temerature Vs Time Graph

ThingSpeak is a popular IoT (Internet of Things) platform that allows users to collect, analyze, and visualize data from various devices and sensors. Here are some key details about using ThingSpeak for IoT projects:

- Data Collection: ThingSpeak provides a simple way to collect data from IoT devices and sensors. Devices can send data to ThingSpeak channels using various communication protocols, including HTTP, MQTT, and UDP.
- Channels: ThingSpeak organizes data into "channels," where each channel can have multiple fields to store different types of data. Users can create and configure their own channels to suit their IoT project's needs.
- Visualization: ThingSpeak offers built-in data visualization tools, such as charts, gauges, and indicators, allowing users to easily analyze and interpret the collected data.
- Analysis: ThingSpeak supports various analysis tools, including MATLAB integration, which enables users to perform advanced data processing and analysis directly on the platform.
- Triggers and Alerts: ThingSpeak allows users to set up triggers and alerts based on the data received from their IoT devices. These triggers can be used to automate actions or send notifications when certain conditions are met.
- IoT Integration: ThingSpeak can be integrated with a wide range of IoT devices and platforms, including Arduino, Raspberry Pi, ESP8266, and others, making it a versatile choice for IoT projects.
- API Access: ThingSpeak provides a RESTful API, which allows users to programmatically interact with their channels and data, enabling custom integrations and applications.
- Scalability: ThingSpeak can handle large amounts of data and support a growing number of connected devices, making it suitable for projects of varying scale.
- Pricing: ThingSpeak offers both free and paid plans, with the free plan providing limited functionality and the paid plans offering more advanced features and higher usage limits.
- Data Logging: ThingSpeak allows users to log data from their IoT devices over time, creating a historical record that can be analyzed and visualized. This is useful for monitoring trends, identifying patterns, and troubleshooting issues.
- Geolocation: ThingSpeak supports geolocation data, which can be useful for IoT applications that involve tracking the location of devices or assets.
- Plugins and Extensions: ThingSpeak offers a range of plugins and extensions that can be integrated with the platform, such as IFTTT (If This Then That) for automating actions, and Zapier for connecting ThingSpeak to other web services.
- Mobile Apps: ThingSpeak provides mobile apps for both iOS and Android, allowing users to monitor and control their IoT devices on the go.
- Security: ThingSpeak includes various security features, such as access control, data encryption, and the ability to secure your channels and API keys.

3.4 Blynk app & Email Notification

Blynk is an IoT platform that can be used in conjunction with email notifications to create a robust IoT system. Here's how you can use Blynk and email notifications together:

- Connecting Devices to Blynk:

Connect your IoT devices (e.g., Arduino, Raspberry Pi, ESP8266) to the Blynk platform using the Blynk library.Configure your devices to send relevant data to Blynk.Setting up Email Notifications in Blynk:In the Blynk app or dashboard, create a "Email" widget and configure it to send notifications.Specify the email address(es) that should receive the notifications.Customize the notification message content, including the data from your IoT devices.

- Triggering Email Notifications:

In the Blynk app or dashboard, create "Triggers" that will send email notifications based on certain conditions or events.For example, you can set up a trigger to send an email notification when a sensor value exceeds a certain threshold.Blynk supports various types of triggers, such as value changes, timer-based triggers, and more.

- Integrating with Third-Party Email Services:

Blynk natively supports email notifications using its own email service.However, you can also integrate Blynk with third-party email services, such as Gmail, Outlook, or custom SMTP servers, to leverage their features and capabilities.This allows you to customize the email settings, authentication, and delivery options to suit your specific needs.

- Monitoring and Troubleshooting:

Blynk provides logs and monitoring capabilities to help you track the status of your email notifications.You can view the delivery status, any errors or failures, and other relevant information to ensure that your email notifications are functioning as expected.By combining Blynk's IoT platform with email notifications, you can create a powerful system that:Collects data from your IoT devices triggers email alerts based on specific conditions notifies stakeholders or users about important events or anomalies enhances the overall monitoring and responsiveness of your IoT project .This integration allows you to leverage the strengths of both Blynk and email notifications to build a more comprehensive and user-friendly IoT solution

4. METHODS

In the following sections, the two different methods that are used in the project are highlighted and explained thoroughly. The methods are divided into two functions; the first function includes the project with live temperature representation using Processing, while the second function includes the project with data logging using a terminal and an SD card.Also, the reasons behind choosing the methods are emphasized within every section. Note that for both of the ftions, the temperature will be represented live on a webpage for ease of access of the user from anywhere as explained in 2.3.Figure 2.1 shows the design of the Arduino board used as a temperature monitoring device.

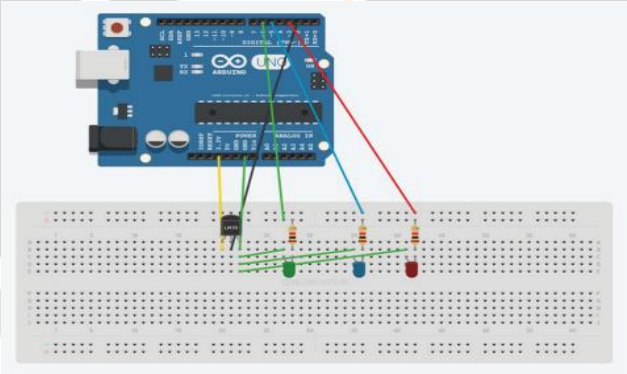

Figure 6:Arduino Circuit Board Design

5. FUTURE WORK

Due to time limitations, some functionalities of a temperature monitoring devices were excluded. If time and resources were not an issue, java and other programming languages would be used to increase attractiveness of the webpage and optimize the security of the html page. Also, FPGA can also be used to add an extra functionality to the thesis and identify best practices in using VHDL or LabVIEW programming. The parallelism in the FPGA design can be helpful in building a stronger temperature monitoring device in terms of speed and logging. Finally, an RTC circuit can be attached to the Arduino board to provide live time stamps for the temperature values. This can increase the readability and reduce the dependability on Processing and computer terminals for time.

6. CONCLUSION

In conclusion, the thesis work is important due to the introduction of new and more robust temperature monitoring techniques and the comparison between them. It also helped in providing new best practices for first time and experienced users within the embedded systems environment. This also helped in building a robust and sophisticated temperature monitoring device.

It is important to identify the different factors that builds a robust temperature monitoring device, such as the different temperature probes, programming environments, terminal programs, and visual alarming hardware and software. Identifying those factors at the beginning helped map a proper plan for the thesis work. It is also important to understand the exact functionality of the temperature sensor before writing down the code to facilitate the thesis work.Within this thesis work, two functions were built to identify the differences between two approaches of temperature monitoring. Arduino, Processing, and CoolTerm were used to build the two functions. An SD card was used to log the temperature values externally. Also, best practices for all the environments were provided.

REFERENCES

[1] W. KRA, "The Course of Temperature in Disease," Am J Medical Science, vol. 57, pp. 425-447, 1869.

[2] D. Parekh, "Designing Heart Rate, Blood Pressure and Body Temperature Sensors for Mobile On- Call System," p. 47, 2010.

[3] L. M. Sund , C. Forsberg and L. Wahren , "Normal Oral, Rectal, Tympanic and Axillary Body Temperature in Adult Men and Women," Scand J Caring Science , vol. 16, pp. 122-128, 2002.

[4] R. Want, "An Introduction to RFID technology," RFID Technology, vol. 1, pp. 1-9, January - March 2006.

[5] I. Yamada, S. Shiotsu , A. Itasaki and S. Inano, "Secure Active RFID Tag System," p. 5, 2005.

[6] D. Pardo , "Design Criteria for Full assive Long Range UHF RFID Sensor fo Human Body Temperature Monitoring," p. 8, 2007.

[7] Nadine , Pesonan; Kaarle, Jaakkola; Jerome , Lamy; Kaj, Nummila; Jouko, Marjonen, "Smart RFID Tags," in Development and Implementation of RFID technology, Finland, I-Tech, 2009, p. 554.

[8] R. Clauberg, "RFID and Sensor Networks," in RFID Workshop, Switzerland, Sept 27, 2004.

[9] Kenichi Agawa, Massimo Alioto, Wenting Zhou, Tsung Te Liu, Louis Alorcon, Kimiya Hajkazemshirazi, Mervin John, Jesse Richmong, Wen Li and Jan Rabaey, "Design and Verification of an Ultra Low Power Active RFID Tag with Multiple Power Domains," in SASIMI Proceeding, California, USA, 2010.

[10] Hequn Chu, Guangmin Wu and Jianming Chen, "Study of Simulation of Semi-Active RFID Tags using Piezoelectric Power Supply for Mobile Process Temperature Sensing," in 2011 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, Kunming, China, March 20-23, 2011.

[11] Amelia S.carr, Man Zhang, Inge Klopping and Hokey Min, "RFID Technology : Implication for Healthcare Organization," American Journal Business, vol. 25, no. 2, pp. 25-40, 2010.