Enhanced Real-Time Temperature Monitoring System Using a DS18B20 Sensor with Bluetooth Communication and a Multi-Level LED Alert Mechanism

Alexsis A. Bongcayao, Jellian Anne P. Regalario, Jose C. Felipe Jr., Mark Nery Codilla, Sandy A. Padrique, Sheenalu A. Beronilla

Computer Engineering Department, Eulogio "Amang" Rodriguez Institute of Science and Technology, Nagtahan Street, Sampaloc, Manila, 1008, Philippines

DOI: https://doi.org/10.47772/IJRISS.2026.100600761

Received: 13 June 2026; Accepted: 18 June 2026; Published: 06 July 2026

ABSTRACT

This study presents the development of an enhanced real-time wireless temperature monitoring system. The system used an Arduino Uno microcontroller, a waterproof DS18B20 digital temperature sensor, and an HC-05 Bluetooth module. It is developed to improve the reliability, usability, and monitoring capabilities of an existing temperature monitoring system. The major enhancements of this system include wireless Bluetooth communication, a multi-level LED and buzzer alert mechanism, hysteresis-based alarm logic, automated email notifications, Excel-based data logging, and a modern Python-based graphical user interface (GUI). The system continuously monitors temperature conditions in real time and classifies them into normal, warning, and critical levels. It also provides corresponding visual, audible, and email alerts. Moreover, experimental testing across normal, warning, and critical temperature conditions demonstrated stable and accurate performance relative to a reference thermometer. Statistical analysis revealed a mean deviation of 0.021% and a standard deviation of 0.047%, indicating high measurement accuracy and consistency throughout the testing conditions. The results indicate that the developed system provides a reliable and efficient solution for real-time temperature monitoring and offers significant improvements in functionality and user accessibility over the original design.

Keywords — DS18B20 Sensor, Temperature Monitoring, Bluetooth Communication, Email Notification, Python GUI

INTRODUCTION

Temperature monitoring plays a significant role in many fields such as in the industrial processes, environmental observation, food storage, and aquaculture [7]. With that, maintaining an appropriate temperature range is an essential thing to ensure safety and efficiency as well as product quality. Moreover, continuous monitoring is required in many applications to prevent damage, reduce risks, and support decision-making. With this, real-time temperature monitoring has become more accessible, cost-effective, and efficient with the advancement of microcontroller-based systems [1], [2].

Some previous studies have explored the development of microcontroller-based temperature monitoring systems for real-time applications. One study on the topic is "Design and Implementation of Microcontroller-Based Temperature Monitoring System," which presented a system capable of measuring and displaying temperature using a microcontroller, a sensor, and the graphical user interface they created. The system also demonstrated the ability to monitor temperature conditions. It also provides basic alerts through visual indicators and a buzzer [7]. However, we have noticed that their system has certain limitations in terms of durability and user feedback, despite its effectiveness.

One of our identified limitations on the original system is the use of the LM35 temperature sensor, in which it is sensitive to moisture and may not be suitable for environments that involve liquids or humid conditions [4]. Additionally, their system relies solely on a basic indicator mechanism that provides limited visual feedback, so users must rely heavily on numerical readings from the display. Generally, the limitations we identified in their

system reduce its practicality for real-world applications where quick interpretation of important data is essential, especially in terms of durability.

To address these limitations, this study proposes an enhanced version of their temperature-monitoring system. The proposed enhancement includes the use of a DS18B20 digital waterproof temperature sensor, in which it offers improved durability and can operate effectively in many environmental conditions [6]. Furthermore, we also came up with a multi-level LED alert mechanism for the system to provide much clearer and more immediate visual feedback by indicating different temperature states such as normal, warning, as well as critical levels [8], [3]. On the other hand, the software component of their system relied on Microsoft Visual Basic 6, which today is considered outdated in modern application development. Due to its limited compatibility with the current operating system and reduced development support, the system may encounter challenges or difficulties in terms of maintainability and future scalability [5]. In response, our study will integrate a Python-based monitoring interface that provides modern real-time visualization and improved software flexibility.

In addition, wireless communication capability was incorporated into the enhanced system via an HC-05 Bluetooth module. Unlike the original system, which relied on a direct USB connection between the microcontroller and the monitoring computer, the developed system can transmit temperature data wirelessly to the monitoring software. This enhancement further improves its portability and flexibility during operation while also reducing cable dependency [9], [10]. Additionally, an automated email alert notification feature was implemented to provide remote warning messages whenever the critical temperature conditions are detected, thereby improving, monitoring responsiveness and user awareness [6].

Finally, through these improvements, the proposed system aims to provide a more reliable, user-friendly, and especially practical solution for real-time temperature monitoring. Our enhanced system not only maintains the core functionality of the original design but also improves its capability in real-world scenarios by increasing durability, compatibility, and enhancing user interaction.

METHODOLOGY

A simple flowchart is presented in Fig. 1 to illustrate the operational process of the enhanced real-time wireless temperature monitoring system. The system's overall process includes temperature sensing, real-time monitoring, alert generation, wireless data transmission, graphical visualization, automated email notification, and historical data logging. The enhanced system uses a waterproof DS18B20 digital temperature sensor to continuously measure temperature. Also, the acquired temperature readings are processed by the Arduino Uno microcontroller and displayed locally on a 16×2 I2C LCD module. Then, the processed data will be transmitted wirelessly through an HC-05 Bluetooth communication module to a Python-based graphical user interface (GUI) for real-time monitoring, visualization, and automated recording.

The Arduino Uno microcontroller component serves as the central processing unit of this developed prototype. It is responsible for processing sensor readings, controlling the LED and buzzer alert mechanisms, updating the LCD display, and managing Bluetooth communication with the monitoring software. The system is powered by the portable power bank integrated into the enclosure, allowing standalone operation without a continuous wired connection to a computer or laptop. Additionally, the developed Python-based GUI receives temperature data wirelessly and provides real-time monitoring, graphical visualization, automated Excel-based data logging, and email notifications during critical temperature conditions. Figure 1 presents the overall operational flow of the enhanced temperature monitoring system.

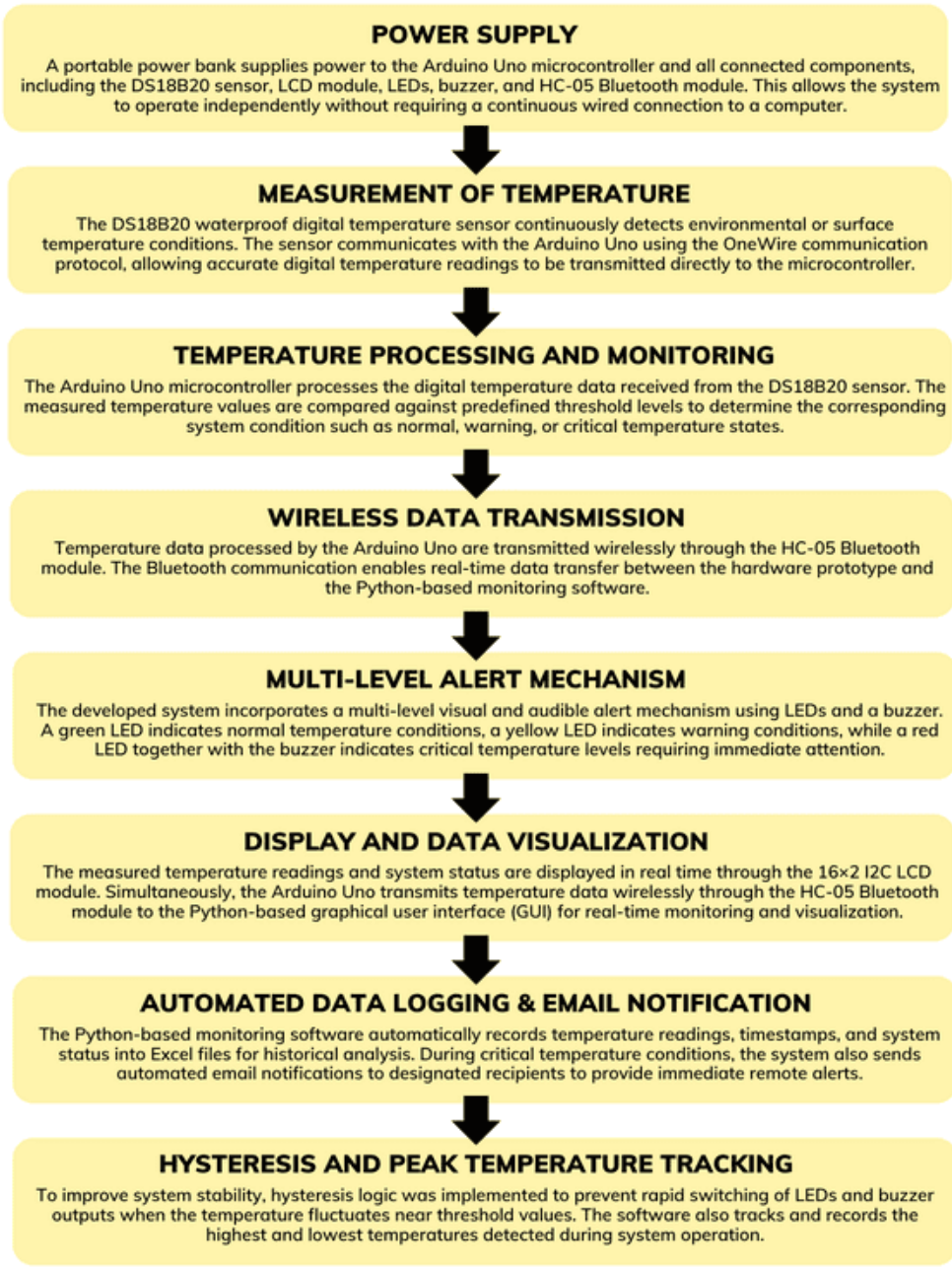

Figure 1. Overall operational flow of the enhanced temperature monitoring system

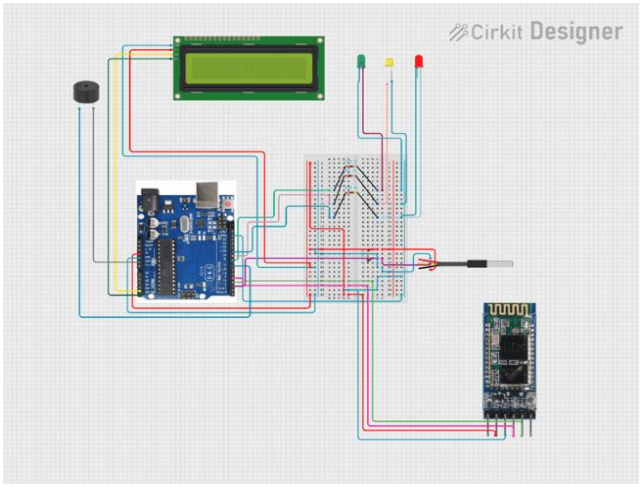

Figure 2. Pictorial circuit diagram of the enhanced temperature monitoring system

Figure 2 presents the pictorial circuit diagram of the developed enhanced real-time temperature monitoring system. The hardware configuration of this system consists of the following components: an Arduino Uno microcontroller, a DS18B20 waterproof digital temperature sensor, a 16×2 I2C LCD module, three LED indicators (green, yellow, and red), a buzzer alarm, and an HC-05 Bluetooth communication module. The Arduino Uno in this system serves as the central controller, processing temperature readings from the DS18B20 sensor and activating the corresponding visual and audible alert mechanisms based on predefined temperature thresholds.

The DS18B20 sensor is connected to a digital input pin of the Arduino Uno using the OneWire communication protocol, together with a pull-up resistor to ensure stable data transmission. Also, the 16×2 I2C LCD module is connected via the SDA and SCL communication lines to display real-time temperature readings and system status information. Furthermore, the green, yellow, and red LEDs are connected to dedicated digital output pins and function as multi-level visual alert indicators, while the buzzer provides an audible warning during critical temperature conditions.

In addition, an HC-05 Bluetooth module is integrated into the enhanced system to enable wireless communication between the Arduino Uno and the Python-based graphical user interface (GUI). The wireless communication capability of the developed system eliminates the need for continuous wired data transmission and enables real-time remote monitoring of temperature data through the software application. The overall hardware configuration is shown in Figure 2. It demonstrates the integration of sensing, processing, alert generation, display, and wireless communication components within the enhanced temperature monitoring system.

Hardware Assembly / Prototype Development

This enhanced prototype consists of the following components: an Arduino Uno microcontroller, a DS18B20 waterproof digital temperature sensor, an HC-05 Bluetooth communication module, and a 16×2 I2C LCD module. It also includes LEDs, a buzzer alarm, jumper wires, a breadboard, and a portable power bank integrated within the enclosure. In this system, the Arduino Uno serves as the main controller, processing temperature readings, activating alert mechanisms, and transmitting monitoring data to the Python-based graphical user interface (GUI) developed.

The DS18B20 digital temperature sensor was utilized as the primary sensing component of the system. This is to replace the LM35 sensor used in the original study because, unlike the LM35, the DS18B20 provides digital temperature measurements and is waterproof. That is why it is better suited for operation in environments exposed to moisture and varying conditions. Also, the sensor communicates with the Arduino Uno via the OneWire protocol, using a pull-up resistor configuration to ensure reliable data transmission.

To improve the system’s portability and usability, an HC-05 Bluetooth module was integrated to enable wireless communication between the hardware prototype and the Python-based monitoring application. The hardware components were interconnected using jumper wires on a breadboard platform. While the LEDs and buzzer were connected to dedicated output pins to provide multi-level visual and audible alerts. Additionally, a portable power bank served as the primary power source for the developed prototype, enabling the system to operate independently without requiring continuous power from a computer or laptop.

Programming Process

The process of programming this system involved integrating microcontroller programming and software interface development. The system was designed primarily to perform real-time temperature monitoring, automated alert activation, graphical visualization, and automated data logging via communication between the system's hardware and software components.

Additionally, the developed system's programming structure was divided into two main components: Arduino programming and Python-based graphical user interface (GUI) development.

Arduino Programming

The system prototype was programmed using the Arduino IDE (Integrated Development Environment). The developed firmware acquired temperature readings from the DS18B20 digital temperature sensor to update the LCD display, evaluate temperature thresholds, and control alert mechanisms. The operational sequence of the microcontroller firmware is illustrated in Fig. 3(a). The program continuously monitors the detected temperature and classifies it as normal, warning, or critical based on predefined threshold values. The corresponding LED indicators and the buzzer alarm are also automatically activated based on the detected temperature status. Furthermore, the processed temperature data is transmitted wirelessly through the HC-05 Bluetooth communication module to the developed Python-based monitoring application, which also enables real-time monitoring without even requiring a direct USB communication link between the hardware and software components.

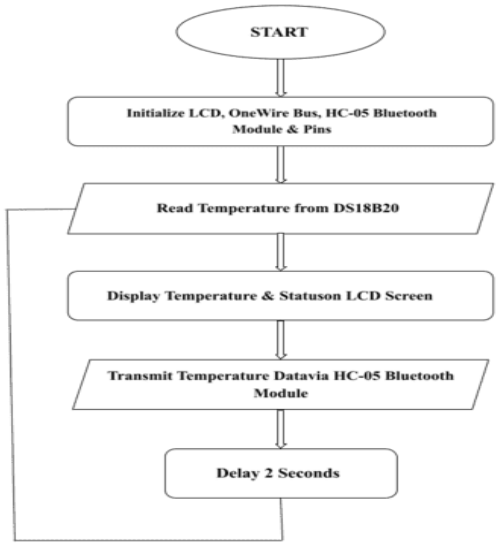

Figure 3 (A). Arduino data acquisition & display loop

Figure 3(b) illustrates the decision-making process used by the developed temperature monitoring system for temperature classification and alert generation. The system compares the measured temperature to predefined threshold values and automatically activates the corresponding alert indicators. Temperatures ranging from 20 °C to 30 °C are classified as normal conditions in this system and activate the green LED indicator. While temperatures between 31 °C and 40 °C are considered warning conditions, they trigger the yellow LED indicator. Meanwhile, temperatures above 41 °C are already considered critical, triggering the red LED and buzzer alarm.

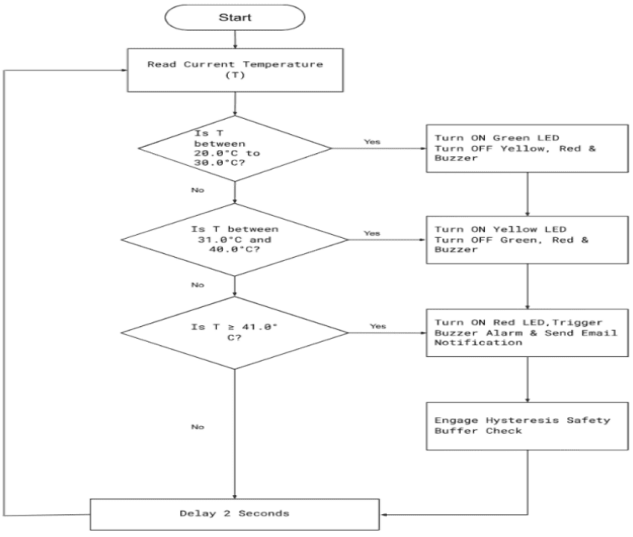

Figure 3 (B). Multi-level alert & hysteresis decision tree

Furthermore, to prevent rapid switching of the LEDs, a buzzer alarm and a system status were implemented when the measured temperature fluctuates near the threshold values, using a so-called hysteresis-based control logic. This enhancement improved the stability and reliability of the monitoring system during continuous operation.

While the Arduino firmware manages temperature acquisition and alert activation, the developed Python-based monitoring application provides an additional layer of notification. So, whenever a critical temperature condition is detected, the software automatically generates and sends an email alert to a designated recipient. This feature enables remote notifications and improves the overall responsiveness of the system's monitoring process.

Python-Based GUI Development

The software used in the original study is Microsoft Visual Basic 6, which is already considered outdated in modern application development and may also pose compatibility issues with the current operating system. So, a Python-based graphical user interface (GUI) was developed to replace the VB6 software described in the original paper. This enhanced software component primarily provided real-time temperature monitoring, graphical visualization, system status indication, and automated data logging.

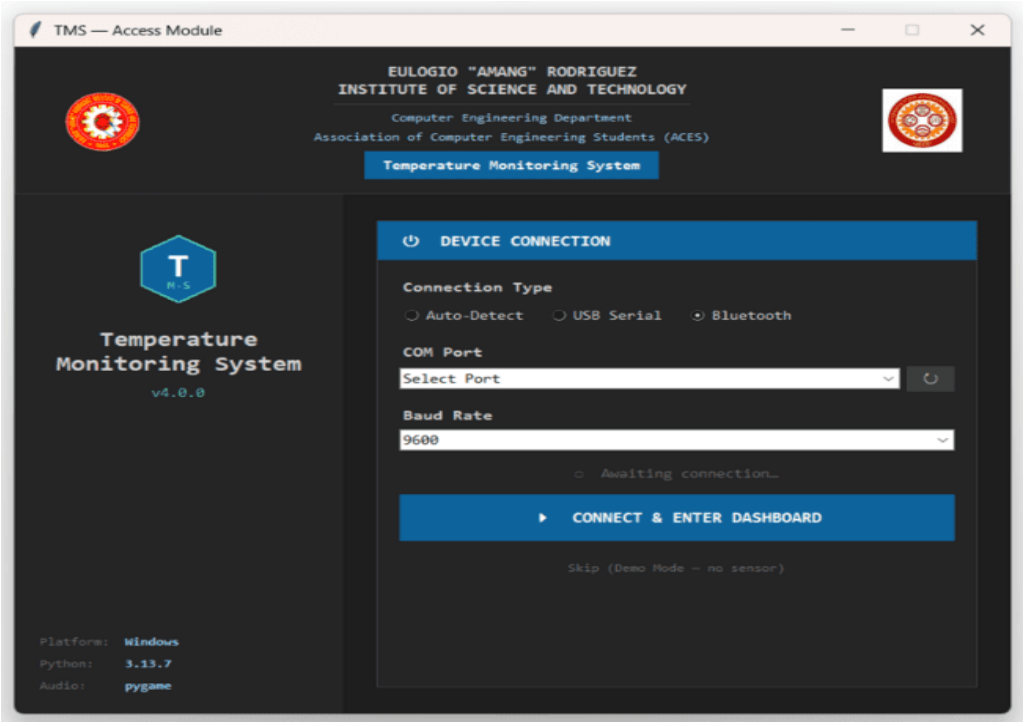

Figure 4. Python-Based GUI – Main Access Module Tab

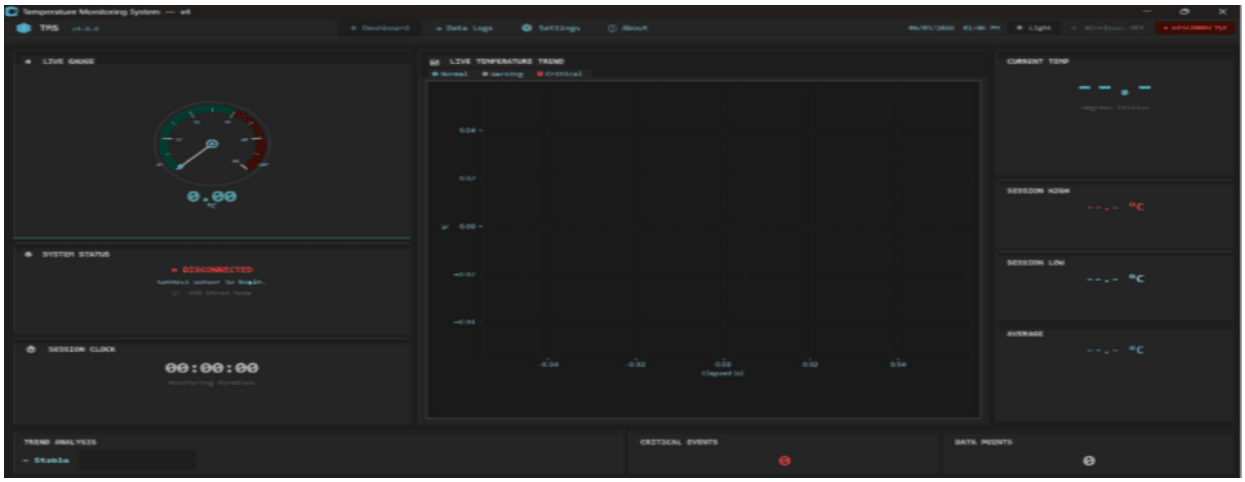

Figure 4.1. Python-Based GUI – Monitoring Dashboard Tab

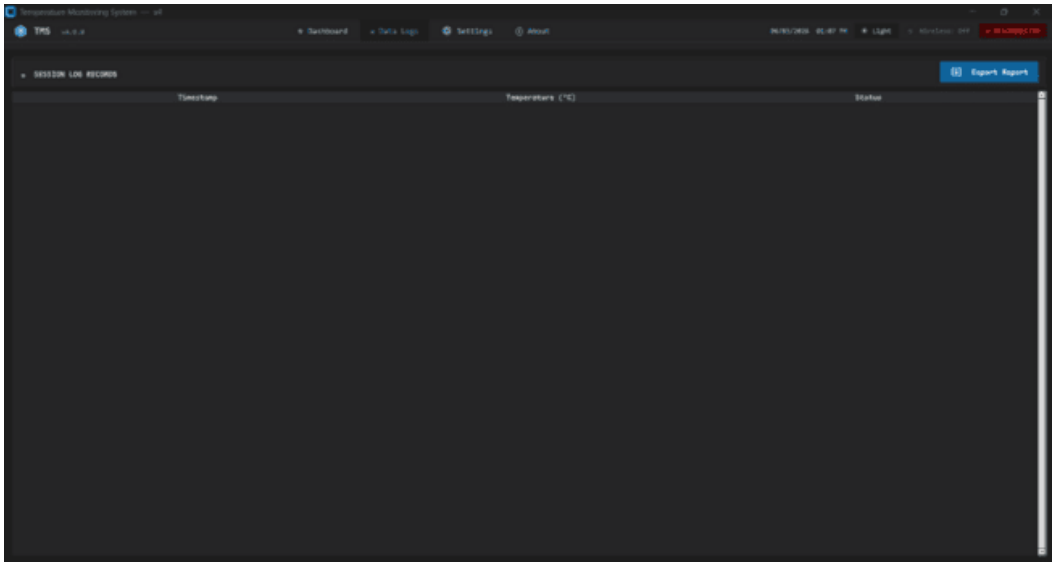

Figure 4.2. Python-Based GUI – Data Logs Tab

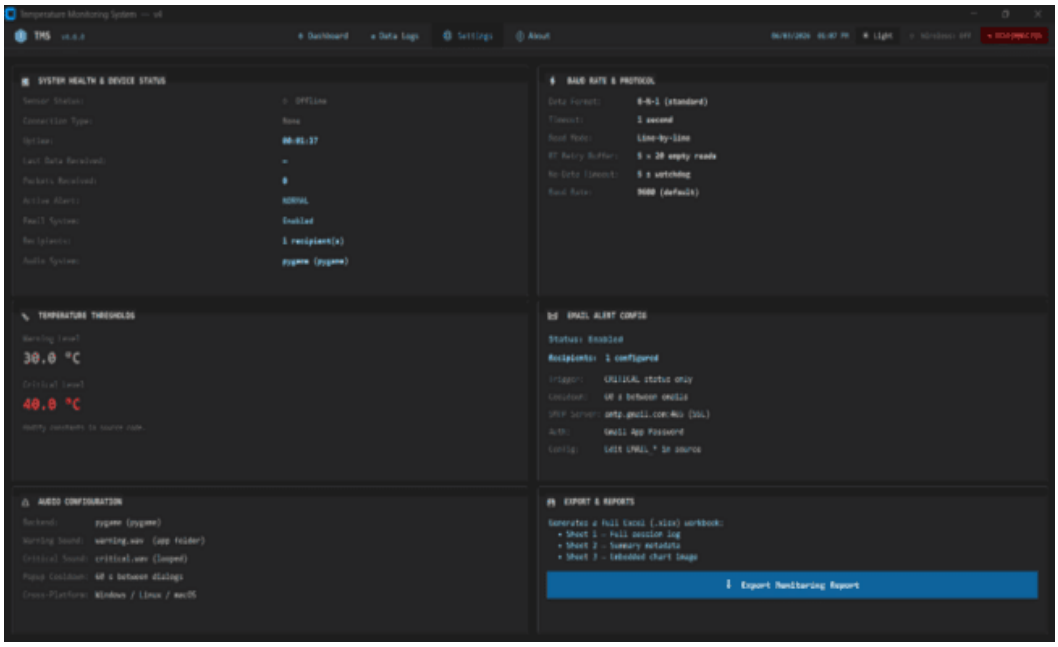

Figure 4.3. Python-Based GUI – Settings and Configuration Tab

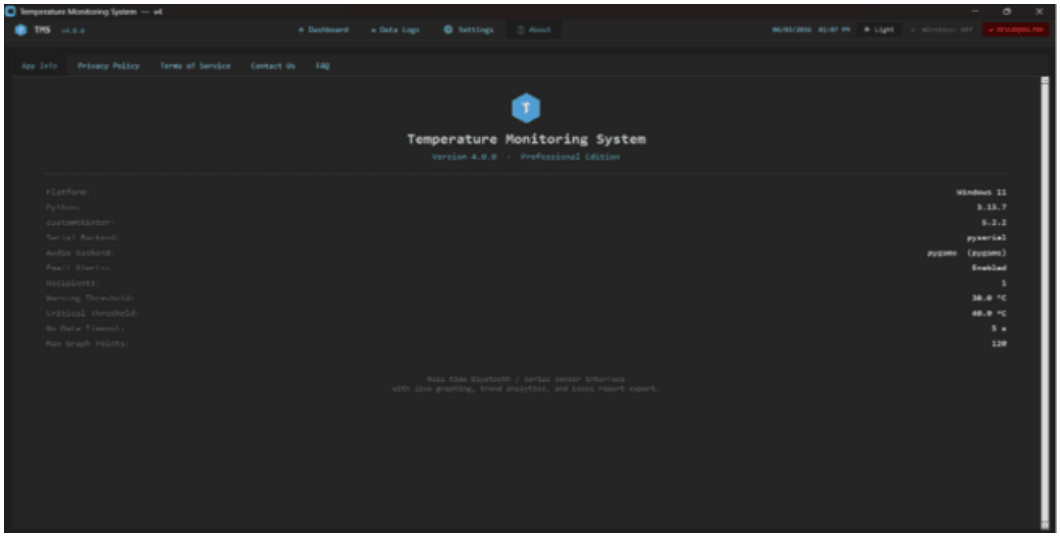

Figure 4.4. Python-Based GUI – About Tab

The developed GUI was created and implemented using several Python libraries, including Tkinter and CustomTkinter for the graphical user interface; Matplotlib for real-time graphical visualization; PySerial for serial and Bluetooth communication; and OpenPyXL for automated Excel-based data logging. These libraries enabled the development of a modern monitoring platform capable of displaying live temperature readings from the sensor, graphical trend analysis, historical temperature records, and system status information. Furthermore, the software also includes a dedicated Access Module that allows users to select available communication ports and establish wireless connectivity using the HC-05 Bluetooth module before accessing the main monitoring interface.

The overall operational logic for the Python-based monitoring application is illustrated in Fig. 5. Upon startup, the application initializes the graphical user interface, communication services, and data logging components. After a valid communication port has been selected and connected, the software continuously receives temperature data from the hardware prototype via the HC-05 Bluetooth module. Also, the received temperature readings are processed in real time, displayed on the monitoring dashboard, recorded in the Excel logging system, and is also evaluated according to their predefined temperature thresholds. Whenever a critical temperature is detected, the application automatically generates an email notification and also updates the corresponding alert indicators within the monitoring interface.

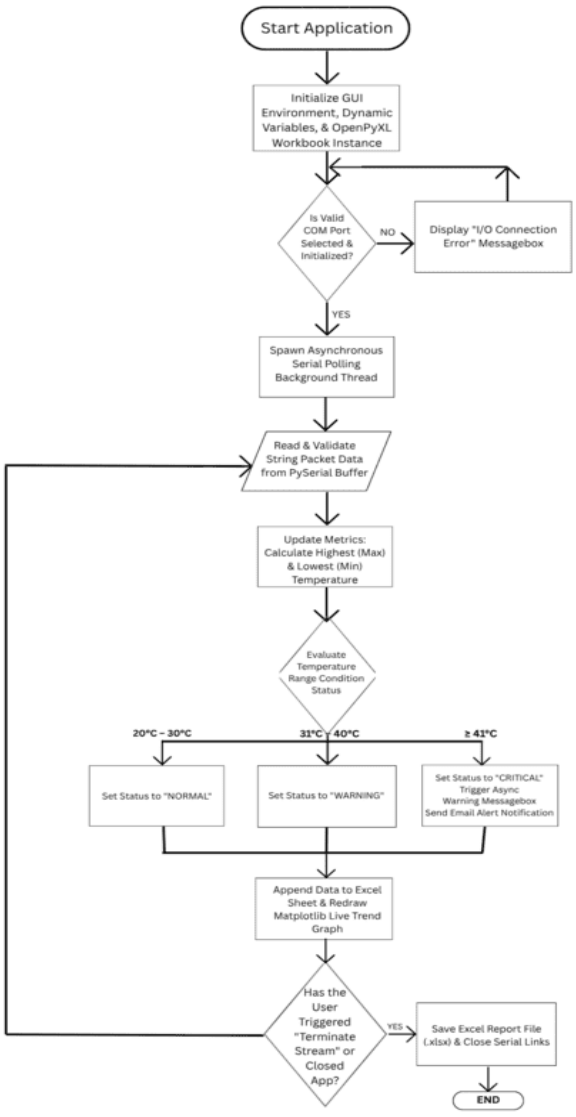

Figure 5. Flowchart of the python programming

Lastly, this Python-based monitoring system was converted into a standalone executable to improve portability and compatibility across Windows systems.

RESULTS AND DISCUSSION

This section presents and discusses the results obtained from the developed enhanced real-time temperature monitoring system under different temperature conditions. The performance of both the hardware prototype and the Python-based graphical user interface (GUI) was evaluated for temperature monitoring accuracy, real-time visualization, automated alert response, and data logging.

The developed system was tested under normal, warning, and critical temperature conditions to evaluate the effectiveness of the DS18B20 temperature sensor, the multi-level LED alert mechanism, the buzzer alarm system, and the automated monitoring interface. Furthermore, the functionality of the developed Python-based GUI application was evaluated based on its ability to display real-time temperature readings, provide graphical visualization, log historical data, and indicate system status.

Python-Based Temperature Monitoring Graphical User Interface

The developed Python-based graphical user interface (GUI) provides real-time monitoring and visualization of temperature readings obtained from the DS18B20 sensor. The software consists of a Main Access Module and a monitoring dashboard interface. The Access Module allows users to communicate with the hardware prototype via the HC-05 Bluetooth module, while the monitoring dashboard provides live temperature monitoring and system status visualization. The GUI displays the current temperature value, temperature condition status, graphical trend analysis, highest and lowest recorded temperatures, and automated historical data logs together with their corresponding date and time records.

The developed interface enables wireless communication between the Arduino Uno microcontroller and the monitoring software via the HC-05 Bluetooth module. In addition, the system automatically records and stores temperature data in the developed logging system for future monitoring and analysis. The software can also automatically generate email notifications when the critical temperature threshold is reached, providing an additional layer of remote alerting and improving monitoring responsiveness.

Figures 6 and 6.1 present the operational interface of the developed Python-based graphical user interface (GUI). The software provides real-time temperature monitoring, graphical visualization, automated data logging, wireless Bluetooth connectivity, system status indication, and email-based critical-temperature notification, enhancing the overall monitoring capabilities of the developed system.

Figure 6. Results Through Python-Based GUI (Dashboard)

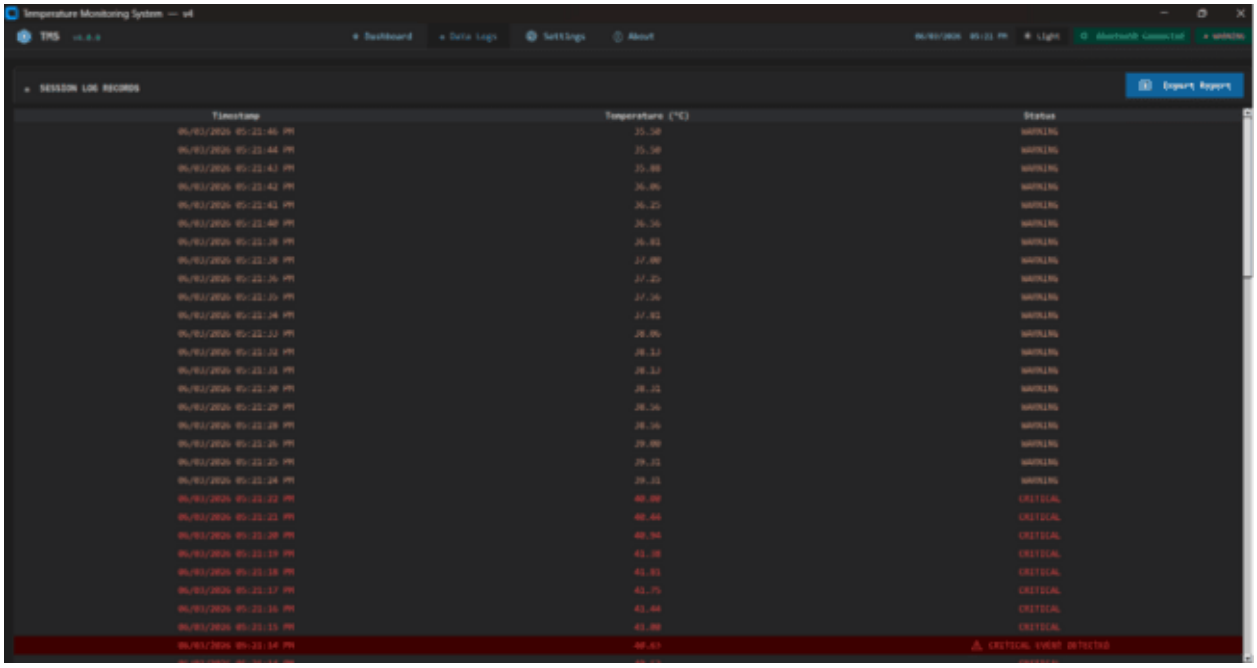

Figure 6.1. Results Through Python-Based GUI (Data Logs)

The developed GUI successfully displayed real-time temperature readings and system status updates during testing. As the DS18B20 sensor detects temperature changes, the graphical visualization updates continuously. Also, the feature automated data logging successfully recorded the temperature readings, including their corresponding time and date values. This functionality enhanced the monitoring, recording, and historical analysis capabilities of the developed system.

Table 1. Functions of the enhanced GUI

No.	Function
1	Set the communication port
2	Connect or disconnect the monitoring system
3	Display the current temperature reading
4	Display system status condition
5	Display the real-time temperature graph
6	Display highest recorded temperature
7	Display lowest recorded temperature
8	Record and save temperature logs with date and time
9	Clear historical data logs
10	Display project title and monitoring dashboard
11	Wireless Bluetooth Communication
12	Email notification during critical temperature condition

Temperature Monitoring System without GUI

The developed temperature monitoring system was also evaluated independently, without the Python-based graphical user interface (GUI). During standalone operation, the hardware prototype successfully performed its primary functions, including real-time temperature monitoring, automated alert activation, and direct temperature display through the integrated 16×2 I2C LCD module. The system was powered by an integrated portable power bank, demonstrating its capability to operate independently without requiring continuous connection to a computer or laptop.

The developed system consists of an Arduino Uno microcontroller, a DS18B20 waterproof digital temperature sensor, an HC-05 Bluetooth communication module, LEDs, a buzzer alarm, an LCD module, and a portable power bank. Each component successfully performed its designated function during system testing. The Arduino Uno served as the primary controller, processing temperature readings, activating the corresponding visual and audible alert mechanisms, and transmitting temperature data wirelessly to the developed monitoring software.

Figure 7 shows the completed enhanced real-time temperature monitoring system prototype, incorporating a DS18B20 sensor, a multi-level LED alert mechanism, an HC-05 Bluetooth communication module, an integrated power bank, and an LCD-based local monitoring interface.

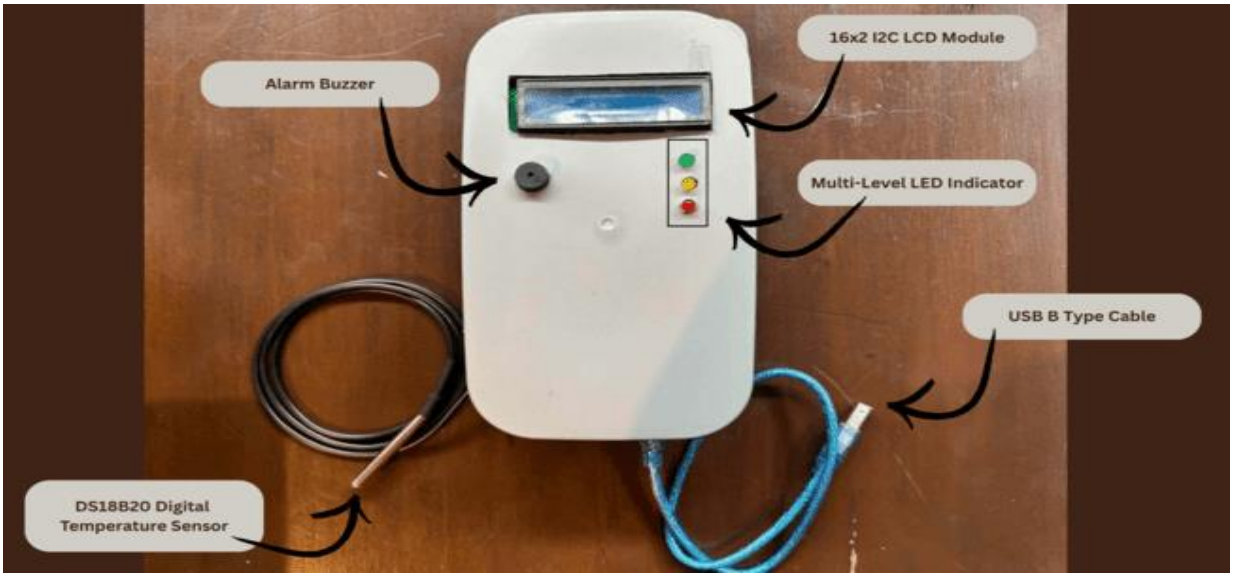

Figure 7. Developed Enhanced Temperature Monitoring System Prototype

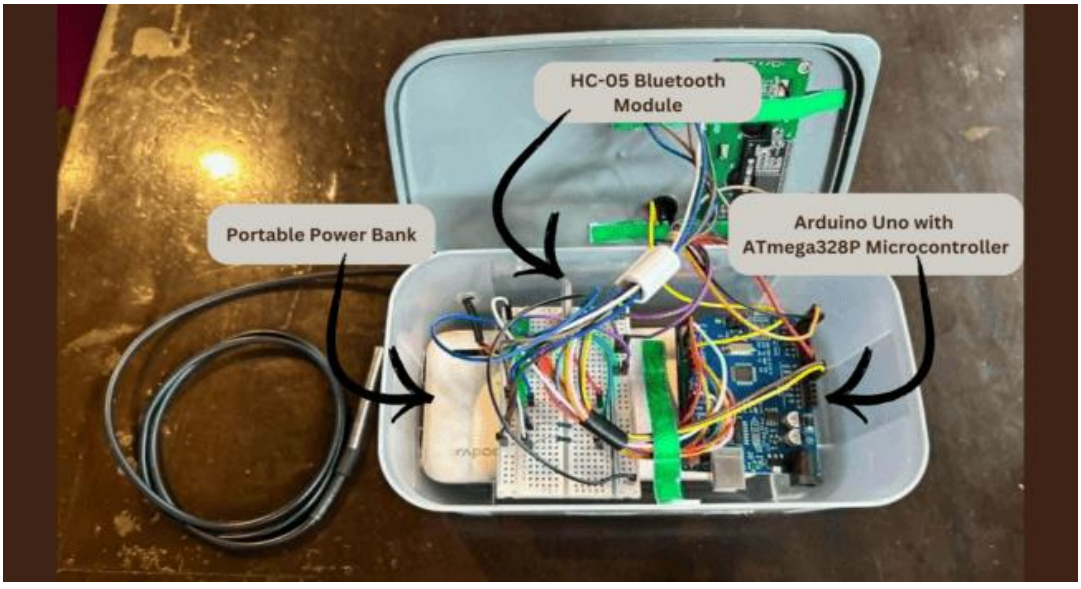

Figure 7.1. Developed Enhanced Temperature Monitoring System Prototype

The system was tested under different temperature conditions, including normal, warning, and critical, to evaluate its responsiveness and accuracy.

The DS18B20 sensor in our hardware prototype successfully detected the temperature variations in real time. It also activated the corresponding LED indicators and buzzer alarm according to their programmed temperature thresholds. Furthermore, the HC-05 Bluetooth module successfully transmitted temperature data to the developed monitoring application during testing, enabling wireless real-time monitoring. The integrated power bank also provided stable power to the system throughout the testing period, allowing continuous standalone operation without external power sources.

Figures 8, 9, and 10 show the actual system responses under normal, warning, and critical temperature conditions, respectively.

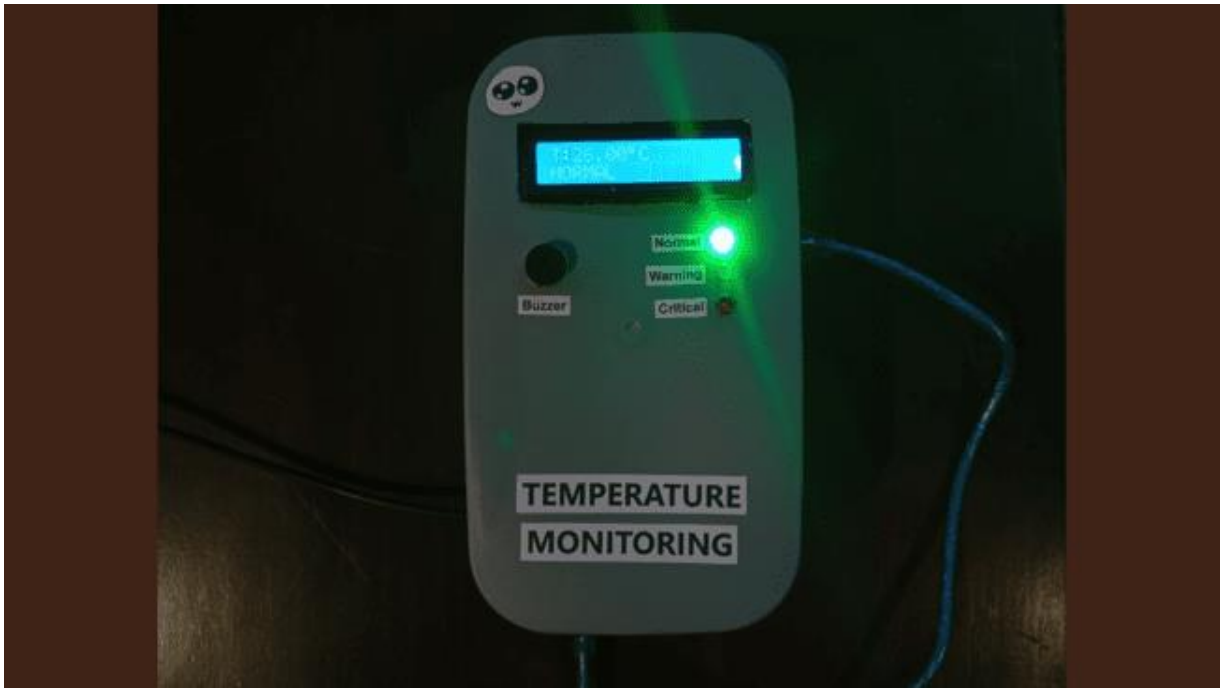

Figure 8. System Result During Normal Temperature Condition

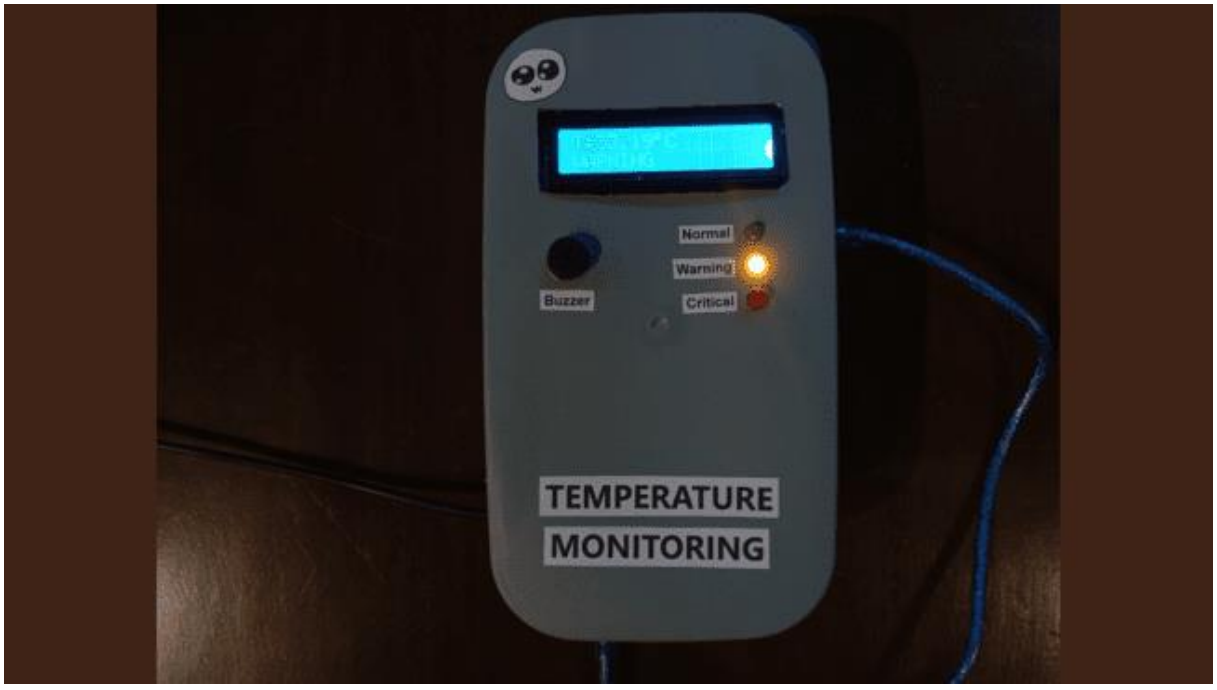

Figure 9. System Result During Warning Temperature Condition

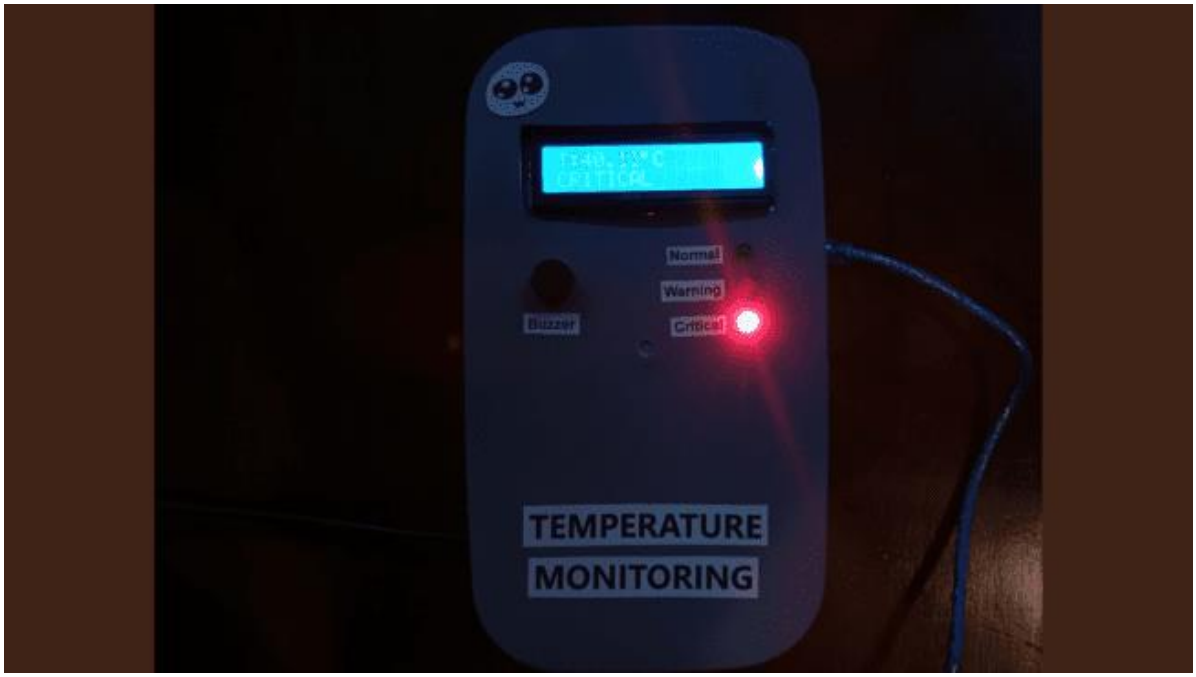

Figure 10. System Result During Critical Temperature Condition

The performance of the developed temperature monitoring system was evaluated based on its responsiveness and temperature measurement accuracy under different environmental conditions. The DS18B20 digital temperature sensor demonstrated stable, accurate temperature readings throughout all tests. The system's accuracy was determined by comparing readings from the developed monitoring system with those from a reference thermometer.

The developed system was tested at three temperature conditions: normal, warning, and critical. The temperature measurements were recorded at short time intervals to evaluate the system's real-time monitoring capability. Unlike the original study, which used hourly intervals, the developed system used shorter intervals due to its continuous, real-time operation.

The normal operating temperature range for the system was set to 20 °C to 30 °C. Meanwhile, temperatures between 31 °C and 40 °C were classified as warning conditions, while those above 41 °C were considered critical. During testing, the system automatically activated the corresponding LED indicators and buzzer alarm based on the detected temperature condition.

The percentage deviation between the temperature readings obtained from the developed system and the reference thermometer was calculated using Equation 1. The thermometer reading was considered as the reference or actual temperature value during the testing procedure.

Equation 1. Percentage Deviation Formula

$$Deviation = \left(\frac{\text{System Temperature} - \text{Thermometer Temperature}}{\text{Thermometer Temperature}} \right) \times 100$$

The developed system was tested under three predefined temperature conditions. For normal-condition testing, the DS18B20 sensor was exposed to ambient room temperatures ranging from 20 °C to 30 °C. Warning-condition testing was performed by gradually exposing the sensor to a moderate heat source to achieve temperatures between 31 °C and 40 °C. Critical-condition testing was conducted by placing the sensor near a higher-temperature source, specifically the surface of an air fryer, resulting in temperatures above 41 °C. For each condition, five consecutive readings were recorded and compared with a reference thermometer to evaluate measurement accuracy and system responsiveness.

Table 2. Results during normal temperature condition

Time	Thermometer Temperature (°C)	System Temperature (°C)	Status	LED	Buzzer	Deviation (%)
18:20:14	29.31	29.31	NORMAL	ON	OFF	0
18:20:18	25.81	25.81	NORMAL	ON	OFF	0
18:20:22	24.06	24.06	NORMAL	ON	OFF	0
18:20:26	22.94	22.94	NORMAL	ON	OFF	0
18:20:30	30.37	30.37	NORMAL	ON	OFF	0

Table 2 shows the readings recorded by the developed temperature monitoring system under normal temperature conditions. The measured temperatures ranged within the predefined normal operating range of 20 °C to 30 °C, and under these conditions, the system successfully activated the green LED indicator while the buzzer alarm remained inactive.

The readings from the DS18B20 sensor were closely similar to those from the reference thermometer, resulting in zero deviation. This just indicates that the developed monitoring system provided stable, accurate temperature measurements under normal environmental conditions.

Table 3. Results during warning temperature condition

Time	Thermometer Temperature (°C)	System Temperature (°C)	Status	LED	Buzzer	Deviation (%)
18:20:49	31.37	31.37	WARNING	ON	OFF	0
18:20:53	31.81	31.81	WARNING	ON	OFF	0
18:20:57	32.06	32.06	WARNING	ON	OFF	0
18:21:02	34.35	34.38	WARNING	ON	OFF	0.09%
18:21:06	40.06	40.06	WARNING	ON	OFF	0

Moving on, Table 3 presents the readings recorded under warning temperature conditions. The measured temperatures ranged from 31 °C to 40 °C, which corresponded to the predefined warning threshold of the developed system; therefore, during this condition, the yellow LED indicator was activated while the buzzer alarm remained inactive.

The warning temperature condition was achieved by gradually exposing the DS18B20 sensor to a much more elevated temperature source. The readings from the developed system remained closely aligned with the thermometer readings, resulting in minimal deviation. This demonstrates the system's ability to properly detect rising temperatures and provide early warning before they reach critical levels.

Table 4. Results during critical temperature condition

Time	Thermometer Temperature (°C)	System Temperature (°C)	Status	LED	Buzzer	Deviation (%)
18:21:19	41.31	41.36	CRITICAL	ON	ON	0.12%
18:21:23	41.65	41.69	CRITICAL	ON	ON	0.10%

18:21:27	41.13	41.13	CRITICAL	ON	ON	0
18:21:31	41.44	41.44	CRITICAL	ON	ON	0
18:21:35	42.49	42.49	CRITICAL	ON	ON	0

Table 4 shows the readings obtained under critical temperature conditions. The measured temperatures reached 41 °C or higher, triggering the critical alert condition of the developed monitoring system. During this state, the red LED indicator and buzzer alarm were automatically activated to provide both visual and audible warnings.

The critical temperature condition was achieved by exposing the sensor to a higher temperature environment, such as the hot surface of an air fryer. As a result, the system successfully detected the temperature increase in real time and accurately activated the programmed safety responses. However, slight deviations observed during critical temperature testing may be attributed to rapid temperature changes and sensor response delay under elevated temperature conditions. Still, the readings from the DS18B20 sensor remained closely comparable to the thermometer readings, resulting in very minimal deviation throughout the testing process. Furthermore, upon reaching the critical temperature threshold, the developed software successfully generated an automated email notification, in addition to the visual and audible alerts, providing an additional layer of warning for remote monitoring.

The experimental results in Tables 2-4 indicate that the developed enhanced real-time temperature monitoring system accurately detected temperature variations across different environmental conditions, namely normal, warning, and critical temperature states. Furthermore, the DS18B20 digital temperature sensor demonstrated stable and responsive performance throughout all conducted tests.

The developed system also successfully implemented the intended multi-level alert mechanism using green, yellow, and red LED indicators, along with the buzzer alarm system. In addition, the minimal to zero deviation observed between the developed system and the reference thermometer indicates that the system can provide reliable, accurate temperature monitoring.

Compared to the original study, the developed system provides improved monitoring capability through the integration of a modern Python-based graphical user interface (GUI), automated data logging, real-time graphical visualization, enhanced multi-level alert mechanisms, wireless Bluetooth communication, portable standalone operation through an integrated power bank, and automated email notifications during critical temperature conditions.

Statistical Performance Analysis

To further evaluate the accuracy and consistency of the developed temperature monitoring system, the mean deviation and standard deviation of the recorded measurements were calculated using Equations 2 and 3.

Equation 2. Mean Deviation

$$\bar{x} = \frac{\sum x}{n}$$

where:

- $\bar{x}$ = mean of deviation values
- x = individual deviation value
- n = number of samples

Equation 3. Standard Deviation

$$SD = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

where:

- SD = standard deviation
- x = individual deviation value
- $\bar{x}$ = mean of deviation values
- n = number of samples

The accuracy and stability of the developed temperature monitoring system were further evaluated using the deviation values obtained during testing under normal, warning, and critical temperature conditions. The percentage deviation between the developed system and the reference thermometer was used as the basis for evaluating measurement performance.

Table 5 presents the statistical summary of the experimental results.

Table 5. Statistical summary of the experimental results

Metric	Value
Number of Samples	15
Minimum Deviation	0.00%
Maximum Deviation	0.12%
Mean Deviation	0.021%
Standard Deviation	0.047%

As shown in Table 5, the calculated mean deviation and standard deviation values were very low, indicating that the developed temperature monitoring system produced accurate and consistent measurements across normal, warning, and critical temperature conditions.

The results indicate that the developed temperature monitoring system achieved a mean error rate (mean deviation) of only 0.021% relative to the reference thermometer. Furthermore, the computed standard deviation of 0.047% demonstrates that the recorded measurements remained highly consistent throughout all testing conditions. Using the obtained deviation values, the developed system achieved an estimated accuracy of approximately 99.98% relative to the reference thermometer. The low deviation values confirm that the DS18B20 sensor provided accurate and stable temperature measurements, while the developed monitoring system maintained reliable performance across normal, warning, and critical temperature ranges.

The evaluation conducted in this study focused on short-term real-time performance testing under controlled temperature conditions. Long-term stability testing over extended operational periods was not included within the scope of the present study and is recommended for future investigation.

CONCLUSION

This study successfully developed an enhanced real-time temperature monitoring system using an Arduino Uno microcontroller and a DS18B20 waterproof digital temperature sensor. The developed system continuously monitored temperature conditions in real time and provided visual and audible alerts via a multi-level LED and buzzer mechanism. Temperature readings were displayed via both the integrated LCD module and the developed Python-based graphical user interface (GUI), while wireless data transmission was enabled via an HC-05 Bluetooth communication module.

Several enhancements were successfully implemented compared to the original study. These include replacing the LM35 sensor with a DS18B20 waterproof digital sensor, integrating a multi-level alert mechanism, implementing hysteresis-based alarm control, enabling automated Excel-based data logging, providing real-time graphical visualization, supporting Bluetooth communication, enabling portable power bank operation, and sending automated email notifications during critical temperature conditions. Furthermore, the GUI application was converted to a standalone executable to improve portability and compatibility across Windows systems.

Based on the experimental results obtained under normal, warning, and critical temperature conditions, the developed system demonstrated accurate and responsive performance with minimal to zero deviation from the reference thermometer readings. Statistical analysis further revealed a mean deviation of 0.021% and a standard deviation of 0.047%, indicating high measurement accuracy and consistency throughout the testing conditions. The system successfully activated the appropriate LED indicators, buzzer alarms, Bluetooth communication functions, and email notification features according to the programmed temperature thresholds. Overall, the developed system achieved the study's objectives and demonstrated significant improvements in reliability, usability, portability, and monitoring capabilities.

Although the system exhibited stable performance during short-term testing under controlled conditions, long-term stability evaluation was beyond the scope of the present study. Future studies may further enhance the system through cloud-based data storage, IoT integration, mobile application support, remote internet-based monitoring, advanced data analytics, and extended long-term performance testing under diverse environmental conditions.

REFERENCES

1. Caranguian, M., & Panganiban, E., “Tilapia Fishpond Monitoring System with Fishkill Prevention,” International Journal of Emerging Trends in Engineering Research, vol. 8, no. 7, pp. 3478–3482, 2020, doi: 10.30534/ijeter/2020/96872020. [Online]. Available: https://doi.org/10.30534/ijeter/2020/96872020
2. Casinillo, R. M. L., So, A. L. A., Mandaya, M. V., Dabalos, S. A. J., Enriquez, M. C. S., & Cane, J. F. A., “Development of Arduino-Based High Heat Detector Temperature Control Prototype for Household Appliances,” IAES International Journal of Robotics and Automation (IJRA), vol. 13, no. 2, pp. 140–159, 2024, doi: 10.11591/ijra.v13i2.pp140-159. [Online]. Available: https://doi.org/10.11591/ijra.v13i2.pp140-159
3. Gabeja, A., Conde, L. J. A., Olfindo, M. K. T., Fabro, M. N., & Beloria, N. G. R., “Room Safety Monitoring Using a Temperature Sensor Alarm System,” International Journal of Research and Scientific Innovation (IJRSI), vol. 13, no. 1, pp. 599–610, Jan. 2026, doi: 10.51244/IJRSI.2026.13010051. [Online]. Available: https://doi.org/10.51244/IJRSI.2026.13010051
4. Kelechi, A. H., Alsharif, M. H., Anya, A. C., Bonet, M. U., Uyi, S. A., et al., “Design and Implementation of a Low-Cost Portable Water Quality Monitoring System,” Computers, Materials & Continua, vol. 69, no. 2, pp. 2405–2424, 2021, doi: 10.32604/cmc.2021.018686. [Online]. Available: https://doi.org/10.32604/cmc.2021.018686
5. Microsoft, “Visual Basic 6.0 Support Policy,” Microsoft Learn, 2026. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-basic-6/visual-basic-6-support-policy. [Accessed: May 25, 2026].
6. Syahid, R. I. M., & Hendryani, A., “Development of a Temperature Monitoring System for MRI Chiller and Helium Compressor Using ESP32 with Web-Based Interface,” Electromedic: Journal of Medical Electronic, vol. 2, no. 2, pp. 109–114, Jan. 2026.

7. Tayab, U. B., & Yuen, C. W., "Design and Implementation of Microcontroller Based Temperature Monitoring System," Journal of Advanced Research Design, vol. 30, no. 1, pp. 1–11, 2017.
8. Uddin, M. N., & Nyeem, H., "Engineering a Multi-Sensor Surveillance System with Secure Alerting for Next-Generation Threat Detection and Response," Results in Engineering, vol. 22, Art. no. 101984, 2024. [Online]. Available: https://doi.org/10.1016/j.rineng.2024.101984
9. Wahyuningsih, A., Maslebu, G., & Setiawan, A., "Wireless Bluetooth-Based SIPESUT (Body Temperature Monitoring System) Prototype," Journal of Science and Science Education, vol. 5, no. 1, pp. 44–51, 2021, doi: 10.24246/josse.v5i1p44-51. [Online]. Available: https://doi.org/10.24246/josse.v5i1p44-51
10. Alina, A. T., "Temperature and Humidity Monitoring Through Bluetooth HC-06," Journal of Electrical Engineering and Electronic Technology, vol. 12, no. 4, 2023.