International Journal of Advanced Scientific Research
www.allscientificjournal.com
ISSN: 2456-0421
Received: 07-08-2025, Accepted: 07-09-2025, Published: 22-09-2025
Volume 10, Issue 3, 2025, Page No. 236-238

Design and implementation of an Arduino-based temperature monitoring system with graphical user interface support

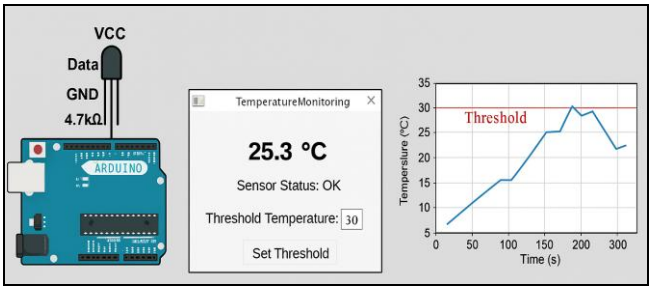
Ofem Ajah Ofem, Okoro Osahon, Ofem Obono Iwara, Bukie Paul Tawo, Essien Ewa Essien
Faculty of Computing, University of Calabar, Nigeria

Abstract
This paper presents the design and implementation of a low-cost, real-time temperature monitoring system using an Arduino Uno microcontroller and a DS18B20 digital temperature sensor. The system integrates a Python-based graphical user interface (GUI) developed with Tkinter to display live temperature readings and alert users when predefined thresholds are exceeded. Unlike conventional monitoring systems, this design emphasizes simplicity, affordability, and adaptability for embedded and IoT applications. Experimental validation confirms the system's accuracy and responsiveness, making it suitable for deployment in healthcare, industrial, and environmental settings.

Keywords: Arduino uno, Ds18b20, tkinter, real-time monitoring, temperature sensing, Iot

Introduction

Temperature monitoring is critical in domains ranging from medical diagnostics to industrial automation. Traditional systems often rely on expensive hardware or proprietary software, limiting accessibility and scalability. This paper introduces a microcontroller-based solution that leverages the Arduino Uno and DS18B20 sensor, coupled with a custom-built GUI, to deliver real-time feedback and threshold alerts. Unlike existing systems, this design offers a modular architecture that can be extended to support multiple sensors or remote access. By addressing the limitations of cost and complexity, the proposed system contributes to the growing field of open-source environmental monitoring.



Materials and Methods

1. System Architecture

The system comprises three core modules: the sensing unit (DS18B20), the processing unit (Arduino Uno), and the interface unit (Python GUI). Data flow begins with the sensor capturing ambient temperature, which is then transmitted via serial communication to the GUI for visualization and alert handling. A block diagram illustrating this architecture is recommended for clarity.

2. Circuit Design

Figure 1 shows the wiring configuration between the DS18B20 sensor and the Arduino Uno. The sensor's VCC, Data, and GND pins are connected to the Arduino's 5V, digital pin 2, and ground respectively. A 4.7kΩ pull-up resistor is placed between VCC and Data to ensure stable communication using the One-Wire protocol. This configuration enables reliable temperature acquisition with minimal hardware overhead.

3. Software Implementation

The software component is developed in Python using the Tkinter library. The Arduino transmits temperature data via serial communication, which is parsed and displayed in the GUI. The interface includes a threshold input field, a sensor status indicator, and a real-time temperature display. Figure 2 illustrates the GUI layout. The system also incorporates error handling to detect sensor disconnection and trigger alerts.

4. Testing Procedure

To validate the performance and reliability of the Arduino-based temperature monitoring system, a structured testing protocol was implemented in two distinct phases: unit testing and integration testing.

1. Unit testing focused on evaluating the accuracy of the DS18B20 digital temperature sensor. This was achieved by comparing its output against a calibrated laboratory-grade reference thermometer under controlled ambient conditions. Measurements were taken at regular intervals over a defined time window, and the resulting data were tabulated to assess consistency and deviation. The observed error margins were analyzed to determine the sensor's precision, with particular attention paid to its response stability and resolution across a range of temperatures.

2. Integration testing was conducted to assess the robustness of the communication pipeline between the Arduino microcontroller and the graphical user interface (GUI). The objective was to verify that temperature data acquired by the sensor were accurately transmitted, processed, and displayed in real time. This involved monitoring the responsiveness of the system under varying conditions, including rapid temperature fluctuations and simulated sensor disconnections. The GUI's threshold alert mechanism was also tested to confirm that it reliably triggered warnings when the

measured temperature exceeded user-defined limits. Successful integration was determined by the absence of data loss, latency, or misrepresentation during continuous operation.

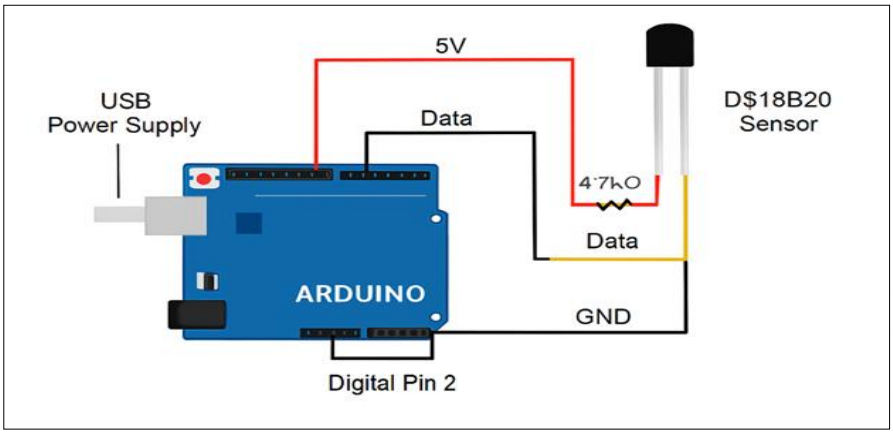


Fig 1: Circuit Diagram of Arduino DS18DB20Connection

Figure 1 illustrates the hardware configuration used to interface the DS18B20 digital temperature sensor with the Arduino Uno microcontroller. The sensor's three terminals, VCC, Data, and GND, are connected to the Arduino's 5V, digital pin 2, and ground respectively. A 4.7kΩ pull-up resistor is placed between the VCC and Data lines to ensure reliable communication. This setup enables accurate temperature readings via the One-Wire protocol, forming the foundation of the monitoring system.

3. Software Implementation

The Arduino was programmed to acquire temperature data and transmit it via the serial port. Python's Tkinter library was employed to create a GUI that displayed the current temperature, sensor status, and alert status. The GUI also allowed threshold adjustment, triggering both visual updates and warning messages when limits were exceeded.

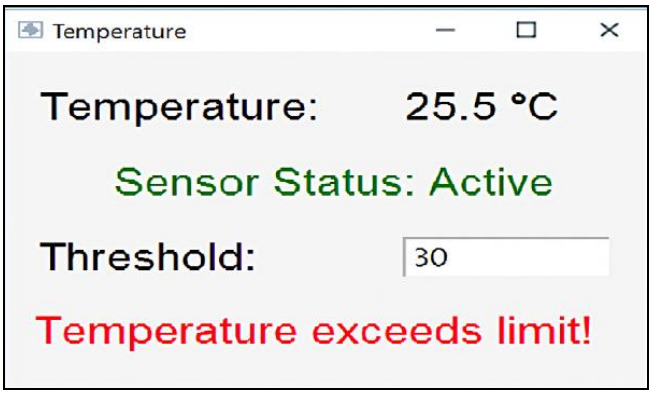


Fig 2: GUI Screenshot of Temprature Monitoring Interface

Figure 2 presents the graphical user interface (GUI) developed for real-time temperature monitoring. The interface displays the current temperature reading from the DS18B20 sensor, the operational status of the sensor, and a user-defined threshold value. When the measured temperature exceeds the threshold, a warning message is triggered. This GUI enhances user interaction and provides immediate feedback, making the system intuitive and responsive.

4. Testing Procedure

To validate the performance and reliability of the Arduino-based temperature monitoring system, a structured testing protocol was implemented in two distinct phases: unit testing and integration testing.

1. Unit testing

focused on evaluating the accuracy of the DS18B20 digital temperature sensor. This was achieved by comparing its output against a calibrated laboratory-grade reference thermometer under controlled ambient conditions. Measurements were taken at regular intervals over a defined time window, and the resulting data were tabulated to assess consistency and deviation. The observed error margins were analyzed to determine the sensor's precision, with particular attention paid to its response stability and resolution across a range of temperatures. This phase ensured that the sensor met the required specifications for environmental monitoring applications.

2. Integration testing

was conducted to assess the robustness of the communication pipeline between the Arduino microcontroller and the graphical user interface (GUI). The objective was to verify that temperature data acquired by the sensor were accurately transmitted, processed, and displayed in real time. This involved monitoring the responsiveness of the system under varying conditions, including rapid temperature fluctuations and simulated sensor disconnections. The GUI's threshold alert mechanism was also tested to confirm that it reliably triggered warnings when the measured temperature exceeded user-defined limits. Successful integration was determined by the absence of data loss, latency, or misrepresentation during continuous operation.

Together, these testing phases provided comprehensive validation of both the hardware and software components of the system, confirming its suitability for deployment in real-world temperature monitoring scenarios.

Results

The system successfully monitored temperature in real time. The DS18B20 readings were consistent, maintaining an error margin within ±0.5°C of the reference thermometer. Alerts were promptly triggered upon exceeding threshold values set in the GUI. The Tkinter interface displayed

current temperature, sensor status, and alert notifications effectively, as illustrated in Figure 2. Additionally, serial monitor output confirmed consistent data acquisition from the Arduino.

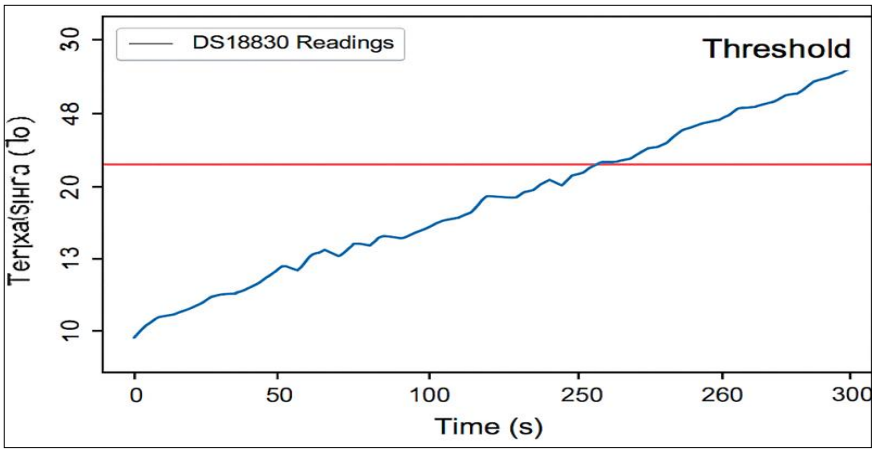


Fig 3: Plot of Recorded Temperatures Over Time

This figure shows a time-series plot of temperature data collected from the DS18B20 sensor over a 300-second interval. The blue line represents the sensor's readings, while the red horizontal line indicates the predefined temperature threshold. The graph highlights the system's ability to track gradual temperature changes and detect threshold breaches, validating its effectiveness for continuous environmental monitoring.

Table 1: Comparison of DS18B20 Readings with Reference Thermometer Values

Time (s)	DS18B20 Reading (°C)	Reference Thermometer (°C)	Error (°C)
0	25.3	25.5	-0.2
30	26.1	26.0	+0.1
60	27.0	27.2	-0.2
90	28.4	28.5	-0.1
120	29.0	29.1	-0.1
150	29.8	30.0	-0.2
180	30.5	30.6	-0.1
210	31.2	31.3	-0.1
240	31.9	32.0	-0.1
270	32.5	32.6	-0.1
300	33.0	33.1	-0.1

Table 1 compares temperature readings obtained from the DS18B20 sensor with those from a calibrated reference thermometer at regular intervals. The error column quantifies the difference between the two measurements. The results demonstrate that the DS18B20 maintains a high degree of accuracy, with deviations consistently within ±0.2°C. This confirms the sensor's suitability for reliable temperature sensing in embedded applications.

Discussion

The results confirm the system's capability to deliver accurate and timely temperature readings. The DS18B20 sensor exhibited minimal deviation from the reference thermometer, validating its reliability for real-world applications. The GUI responded effectively to threshold breaches, and the integration between hardware and software was seamless One limitation observed was the sensor's sensitivity to ambient electrical noise, which could be mitigated through shielding or the use of twisted-pair cables. Additionally, while the current system supports a single sensor, future iterations could incorporate multi-sensor support and remote data logging via cloud integration. These enhancements would broaden the system's applicability in distributed IoT environments.

Conclusion

This paper successfully implemented and validated a microcontroller-based temperature monitoring system featuring a real-time GUI. The integration of the DS18B20 sensor with the Arduino Uno and Python interface proved effective for accurate and responsive temperature tracking. The system's modularity and affordability make it a viable solution for various applications, including healthcare, industrial automation, and environmental monitoring. Future work will focus on expanding sensor support, improving data security, and enabling remote access capabilities.

References

1. Abu Radia MA, El Nimr MK, Atlam AS. IoT-based wireless data acquisition and control system for photovoltaic module performance analysis. e-Prime - Advances in Electrical Engineering, Electronics and Energy,2023:6:100348. https://doi.org/10.1016/j.prime.2023.100348
2. Brown K. Advancements in temperature monitoring systems: A comprehensive review. Journal of Industrial Monitoring,2018:12(4):234–245.
3. Chacko B, Peter JV. Temperature monitoring in the intensive care unit. Indian Journal of Respiratory Care,2018:7(1):28. https://doi.org/10.4103/ijrc.ijrc_13_17
4. Elyounsi A, Kalashnikov A. Evaluating suitability of a DS18B20 temperature sensor for use in an accurate air temperature distribution measurement network. Engineering Proceedings,2021:10(1):56. https://doi.org/10.3390/ecsa-8-11277
5. Garcia R, Lee S. Human error in data logging: Implications and solutions. Journal of Data Management,2020:15(3):120–134.
6. Geng X, Zhang Q, Wei Q, Zhang T. A mobile greenhouse environment monitoring system based on the Internet of Things. IEEE Access,2019:7:1–1. https://doi.org/10.1109/ACCESS.2019.2941521