Software Requirements Specification

for

Arduino-Based Smart Temperature Monitoring System

Version 1.0 approved

Prepared by Ethan England, John Gahagan

Western Kentucky University

02/04/2025

Table of Contents

[separate page]

Revision History

Name	Date	Reason For Changes	Version
John Gahagan	2/5/2025	Initial Draft	1.0.0
Ethan England	2/5/2025	Initial Draft	1.0.0

Name	Contribution
John Gahagan	Started documentation,wrote all sections, and made appropriate diagrams. Also introduced a GitHub repository for the team to use for version control.
Ethan England	Created presentation, Edited sections and Formatted.

1. Introduction

This document specifies the software requirements for an Arduino-Based Smart Temperature Monitoring System. The system continuously measures temperature using a DHT11 sensor, processes data through an Arduino Uno R4 WiFi, and provides real-time feedback via LED indicators and a web dashboard. Users can configure their own temperature thresholds via a web interface using HTML, CSS, and JavaScript, with data storage and retrieval managed through Firebase.This system promotes energy efficiency by optimizing temperature control, reducing unnecessary energy consumption in temperature-sensitive environments, and helps prevent operational downtime by ensuring consistent environmental conditions.

This document is intended for developers, project managers, testers, and documentation writers involved in implementing and deploying the system.

1.1 Purpose

This document specifies the software requirements for an Arduino-Based Smart Temperature Monitoring System. The system measures real-time temperature using a DHT11 sensor and provides visual feedback via LEDs (Red, Green, Blue) based on user-defined temperature thresholds. A web-based dashboard allows users to modify these thresholds in real time using WebSockets.

1.2 Document Conventions

- Monospace font for code snippets.
- Bold text for key features and components.
- Sections are numbered according to IEEE SRS standards.

1.3 Project Scope

The Arduino-Based Smart Temperature Monitoring System aims to provide a real-time, configurable solution for temperature monitoring. The system offers:

- Real-time temperature readings displayed on a web dashboard.
- User-defined threshold settings for temperature classification.
- LED-based visual alerts for immediate feedback.
- WebSockets for instant communication between hardware and web interface.
- Firebase integration for storing temperature logs and user preferences.

This system could support industries requiring precise environmental control, such as smart homes, server rooms, greenhouses, and cold storage units.

1.4 References

DHT11 Temperature Sensor Datasheet

- Author: Aosong Electronics
- Version: 1.3
- URL: DHT11 Datasheet

Arduino Uno R4 WiFi Documentation

- Author: Arduino.cc
- Version: Latest
- URL: Arduino Uno R4 WiFi Docs

Firebase Realtime Database Documentation

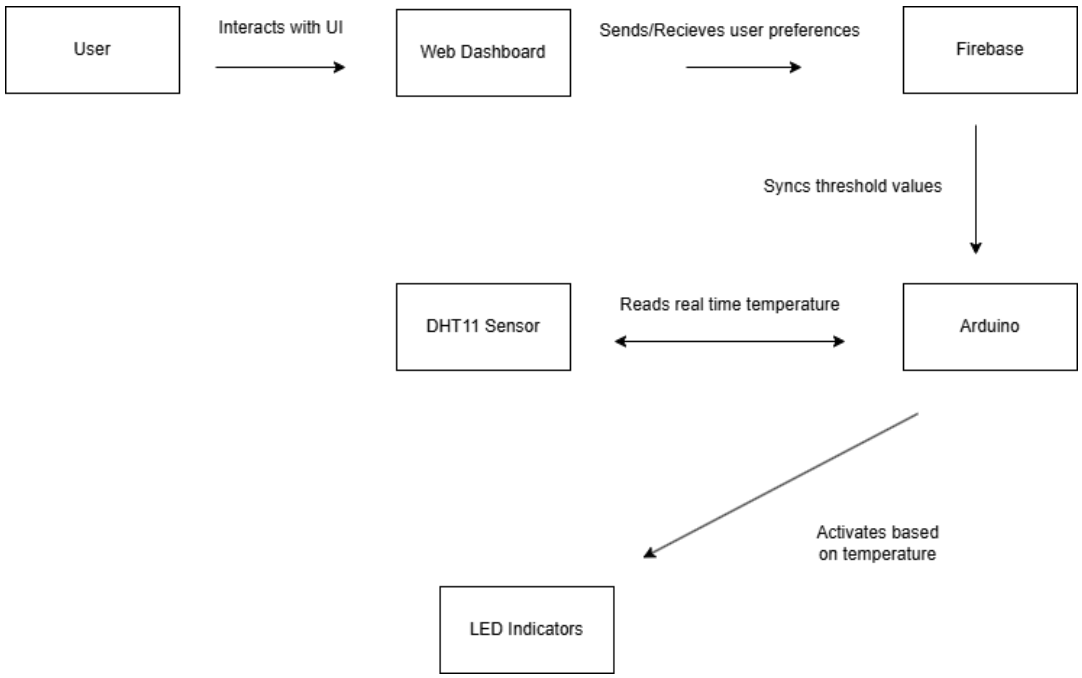
- Author: Google
- Version: Latest
- URL: Firebase Docs

WebSockets API Specification

- Author: Mozilla Developer Network (MDN)
- Version: Latest
- URL: WebSockets API

2. Overall Description

This system is a standalone IoT-based monitoring solution that can be integrated with smart home applications. It operates independently but can be extended with cloud services and additional sensors.



2.1 Product Perspective

This system is a standalone IoT-based monitoring solution that follows a modular approach, with clearly defined tasks for sensor reading, LED control, web communication, and data storage Making it extendable with smart home applications. It operates independently but can extend functionality by integrating with cloud services.

2.2 User Classes and Characteristics

General Users: Individuals who want to monitor and customize their environment.

Developers: Users interested in extending the system's capabilities.

2.3 Operating Environment

The system include a simple circuit with the following connections:

DHT11 Sensor:

VCC to 5V

GND to GND

Data to Digital Pin X

LEDs

Blue,Green, and Red LEDS connected to respective Digital Pins via 220Ω resistors. Cathodes which are the shorter side of the LED connection for all LEDs to GND.

Arduino Uno R4 WiFi:

Handles sensor data processing and WebSocket communication.

Hardware: Arduino Uno R4 WiFi, DHT11 sensor, LEDs (Red, Blue, Green), Breadboard, Jumper Wires.

Software: IDE, WebSocket Server, Web-based UI.

Power Source: USB or external power adapter.

2.4 Design and Implementation Constraints

The DHT11 sensor updates every 2 seconds, limiting real-time responsiveness.

WebSockets require a WiFi connection for real-time dashboard updates.

Resource Limitations: Arduino has limited memory (RAM/Flash) and processing power, requiring efficient task scheduling and memory management.

2.5 Assumptions and Dependencies

Assumptions:

Users have access to a WiFi network for dashboard communication.

The system runs continuously without manual intervention.

We assume that the internet and hosting services are working correctly, that the third party software and libraries are accurate and dependable, and that the hardware can handle the software.

Dependencies:

Hardware Constraints

- The system must run on an Arduino board with limited memory and processing power.
- The LED indicator is constrained to three states (Red, Yellow, Green) based on temperature readings.
- The temperature sensor must be compatible with Arduino and provide real-time readings.

Software Constraints:

- The system must handle task scheduling for temperature reading, LED control, and data communication.
- The firmware should be efficient to minimize processing delays and power consumption.

Network and Communication Constraints:

- Weak network connectivity could affect system performance.

3. System Features

Requirements:

Users have access to a WiFi network for dashboard communication.

The system runs continuously without manual intervention.

Features:

3.1 Real time Temperature Monitoring

3.1.1 Description

High Priority

The system continuously reads temperature data from the DHT11 sensor.

Data is displayed on the serial monitor and transmitted via WebSockets.

3.1.2 Stimulus/Response Sequences

User powers on the Arduino → System starts temperature monitoring.

Temperature changes → LED updates based on thresholds.

3.1.3 Functional Requirements

Temperature monitoring must be able to read data from the sensor and respond to it under 500ms.

3.2 LED-Based Temperature Indication

3.2.1 Description

High Priority
Three LEDS indicate temperature conditions. The Blue LED signifies it is too cold. Green LED for being just right. Red LED for being too hot. The system shall read temperature data from the sensor at a fixed interval.

3.2.2 Stimulus/Response Sequences

Temperature below cold threshold → Blue LED ON.

Temperature between cold and hot thresholds → Green LED ON.

Temperature above hot threshold → Red LED ON.

3.2.3 Functional Requirements

The system shall compare the current temperature with user-defined thresholds.

The system shall activate the corresponding LED based on threshold values.

3.3 Web Dashboard for Threshold Configuration

3.3.1 Description

High Priority: The system shall provide a web-based dashboard displaying:

- Current temperature reading.
- LED status.
- Last updated timestamp.

The web interface shall refresh temperature data every X seconds.

The system shall use secure authentication for accessing the web dashboard.

The system will allow users to configure their custom threshold through the web interface.

The system will then adjust to their settings. The system means LEDs, web dashboard, etc.

Medium Priority: If the network connection is lost:

- The system shall display a “Connection Lost” message on the web interface.
- The system shall retry the connection every Y seconds.
- If the issue persists beyond five minutes, it shall store data locally and attempt to sync once the connection is restored.

3.3.2 Stimulus/Response Sequences

User updates threshold on the dashboard → WebSocket sends new values to Arduino.

Arduino receives threshold update → Adjusts LED behavior accordingly.

3.3.3 Functional Requirements

The system shall allow users to set custom temperature thresholds.

The system shall update thresholds in real time.

4. Data Requirements

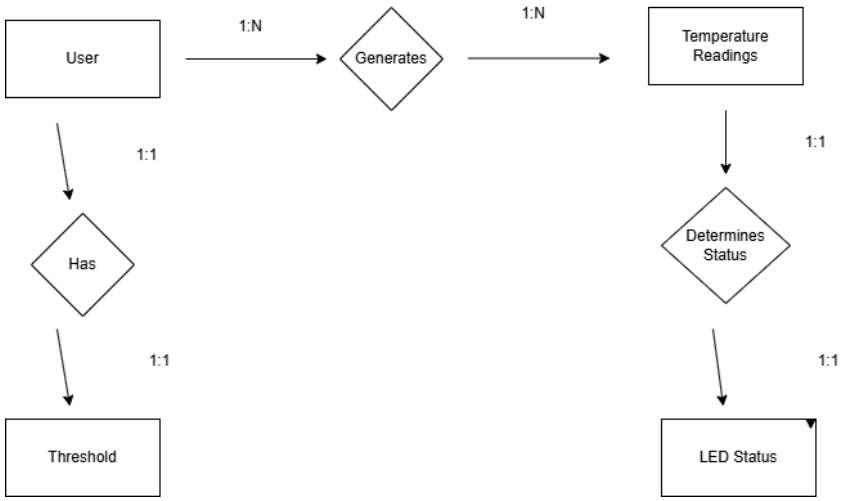
This section outlines the data that the system will collect, process, and output to provide real-time temperature monitoring functionality.

4.1 Logical Data Model

The system will collect temperature readings every 2 seconds.

User-defined temperature thresholds will be stored in Firebase.

Logged temperature values will be stored with timestamps for historical analysis.



4.2 Data Dictionary

The data dictionary defines the structure, format, and constraints of data elements used in the Arduino-Based Smart Temperature Monitoring System. Below are the key data entities and their attributes:

4.2.1 Data Entities and Attributes

Entity	Attribute	Data Type	Length	Format	Description
User	UserID (PK)	String	36	UUID	Unique identifier for each user
	Email	String	255	user@example.com	User email for authentication
	Password	String	64	Hashed	Securely store hashed

					password
Temperature Reading	ReadingID (PK)		36	UUID	Unique identifier for each reading
	UserID (FK)	String	36	UUID	Links reading to a specific user
	Timestamp	DateTime		YYYY-MM-DD HH:MM:SS	Data and time of the temperature reading
	Temperature	Float		XX.XX	Temperature in Celsius or Fahrenheit
Threshold Settings	ThresholdID (PK)	String	36	UUID	Unique identifier for threshold settings
	UserID (FK)	String	36	UUID	Links threshold to a specific user
	ColdLimit	Float		XX.XX	User defined lower temperature limit
	HotLimit	Float		XX.XX	User defined upper temperature limit
LED Status	StatusID (PK)	String	36	UUID	Unique identifier for each status change
	ReadingID (FK)	String	36	Temperature Readings	Temperature Readings

4.2.2 Data Constraints and Validation

Temperature Readings:

- Must be within the range of 0°C to 50°C (DHT11 sensor limits).
- Must be recorded every 2 seconds.

Threshold Settings:

- ColdLimit cannot be greater than HotLimit.
- Default thresholds will be predefined (e.g., Cold: 15°C, Hot: 30°C) if not set by the user.

User Authentication Data:

- Emails must follow standard email formatting rules.
- Passwords must be stored in hashed format to ensure security.

4.3 Reports

4.3.1 Temperature Log Report

- Purpose: Provides a historical record of temperature readings for analysis.
- Content:
    - Timestamp (Date and Time of Reading)
    - Temperature Reading (in Celsius or Fahrenheit)
    - Humidity Reading (if applicable)
    - LED Status at the time of the reading (Too Cold, Just Right, Too Hot)
- Sort Order: Ordered chronologically by timestamp.
- Output Formats: Available in CSV, PDF, and JSON for easy export.
- Retention: Logs data from the past 30 days, with options for user-defined retention periods.

4.3.2 User Threshold Settings Report

- Purpose: Summarizes user-defined threshold settings over time.
- Content:
    - User ID
    - Temperature Thresholds (Cold Limit, Hot Limit)
    - Last Modified Date
- Sort Order: Ordered by last modified date.
- Output Formats: Available in JSON.

4.4 Data Acquisition, Integrity, Retention, and Disposal

4.4.1 Data Acquisition

- The DHT11 sensor collects temperature readings every 2 seconds.
- The Arduino Uno R4 WiFi processes and transmits data via WiFi/WebSockets to the web dashboard and Firebase.
- Users can input custom temperature thresholds through the web dashboard, which are stored in Firebase for persistent access.

4.4.2 Data Integrity and Protection

- *Encryption: All data transmissions between the Arduino, Firebase, and the web dashboard use TLS encryption to prevent unauthorized interception.*
- *Authentication: User settings and access permissions are managed through Firebase Authentication.*
- *Data Validation:*
    - *Temperature readings outside the sensor's valid range (0°C to 50°C) will be discarded.*
    - *Threshold updates will be validated to prevent conflicts (e.g., cold threshold cannot exceed hot threshold).*

4.4.3 Data Retention and Disposal Policies

- Temperature logs will be retained for 30 days to allow short-term trend analysis.
- User-defined thresholds persist indefinitely until modified or deleted by the user.
- Archived Data: After 30 days, temperature logs will be archived and stored for up to 6 months before deletion.
- Cached Data: The web dashboard may cache recent temperature readings for faster performance but will not store long-term records locally.
- Interim Backups:
- Firebase implements automatic daily backups to prevent data loss.
- In case of a system failure, the Arduino device will retain the last known thresholds in EEPROM memory to resume operation upon reboot.

5. External Interface Requirements

The Arduino-Based Smart Temperature Monitoring System interacts with both users and external hardware/software to ensure seamless communication, real-time updates, and efficient monitoring. Below are the necessary interface specifications.

5.1 User Interfaces

Web Dashboard: Provides users with a graphical representation of real-time temperature data. Allows users to set and modify temperature thresholds. Built using HTML, CSS, JavaScript, and WebSockets for real-time data updates.

Physical LED Indicators: Red LED: Indicates temperature is above the hot threshold. Green LED: Indicates temperature is within the acceptable range. Blue LED: Indicates temperature is below the cold threshold.

5.2 Software Interfaces

The Arduino-Based Smart Temperature Monitoring System interfaces with various software components to ensure smooth operation, real-time data exchange, and user interaction. The following describes these interfaces, their purposes, data formats, and communication mechanisms.

5.2.1 External Software Components

Firebase Realtime Database (Google, Latest Version)

Purpose: Stores user-defined temperature thresholds and temperature logs.

Data Format: JSON structure.

Communication Protocol: REST API & WebSockets.

Mappings: Sensor readings are pushed to Firebase; user-defined thresholds are pulled by Arduino.

Web Dashboard (Custom, Developed using HTML, CSS, JavaScript, WebSockets)

Purpose: Provides a graphical interface for users to monitor temperature and modify thresholds.

Data Format: JSON payloads.

Communication Protocol: WebSockets for real-time updates; HTTP requests for settings retrieval.

Mappings:

Sensor data → WebSockets → Dashboard display.

User-defined thresholds → HTTP API → Firebase → Arduino.

Arduino IDE (Latest Version, C++ Programming Language)

Purpose: Develops, compiles, and uploads firmware to the Arduino Uno R4 WiFi.

Dependencies: Uses standard Arduino libraries (DHT11, WebSockets, WiFiClient, Firebase).

5.2.2 Operating System Requirements

Arduino Board: Runs on bare metal firmware in the Arduino C++ environment
Web Dashboard: Compatible with any OS that supports a modern browser.
Database and Hosting: Firebase runs on Google Cloud infrastructure.

5.2.3 Data Flow and Communication

Component	Input	Output	Format	Protocol
DHT11 Sensor	Temperature and Humidity	Digital Reading	Integer	GPIO Signal
Arduino Uno R4 WiFi	Sensor Data, User Thresholds	Web Dashboard Update	JSON	WebSockets
Web Dashboard	User Threshold Settings	Firebase Update	JSON	HTTP API
Firebase	Sensor Logs, User Settings	Arduino and Dashboard	JSON	WebSockets and REST API

5.3 Hardware Interfaces

The Arduino-Based Smart Temperature Monitoring System interfaces with multiple hardware components to ensure accurate temperature monitoring and data transmission. Below is a detailed description of each hardware interface and its interaction with the software components.

5.3.1 Supported Hardware Devices

- Arduino Uno R4 WiFi: The microcontroller that processes temperature readings, controls LEDs, and communicates with the web dashboard.
- DHT11 Temperature and Humidity Sensor: Measures ambient temperature and transmits data to the Arduino.
- LED Indicators (Red, Green, Blue): Provides real-time visual feedback based on temperature thresholds.
- Power Supply (5V via USB or External Adapter): Powers the Arduino and connected components.

5.3.2 Data and Control Interactions

- DHT11 Sensor → Arduino Uno R4 WiFi
  - Input Data: Digital temperature and humidity readings
  - Format: 8-bit digital signal via single-wire communication
  - Valid Values: Temperature range 0°C to 50°C, Humidity range 20% to 90% RH
  - Timing Considerations: Sensor updates every 2 seconds
- Arduino Uno R4 WiFi → Web Dashboard (via WiFi & WebSockets)
  - Input Data: User-defined temperature thresholds from the dashboard
  - Output Data: Real-time temperature updates

    - Format: JSON payloads via WebSockets
    - Timing Considerations: Updates sent every 2 seconds
- Arduino Uno R4 WiFi → LED Indicators
    - Input Data: Temperature threshold evaluations
    - Output Data: LED status changes
    - Format: Digital HIGH/LOW signals to GPIO pins
    - Valid Values:
        - Red LED ON if temperature > user-defined hot threshold
        - Green LED ON if temperature is within user-defined range
        - Blue LED ON if temperature < user-defined cold threshold
    - Timing Considerations: LED status updates occur immediately upon temperature change

5.3.3 Communication Protocols

- DHT11 Sensor: Uses single-wire digital communication protocol.
- Arduino → Web Dashboard: Uses WiFi (IEEE 802.11) and WebSockets for real-time data exchange.
- Arduino → LEDs: Uses GPIO digital output signals.

5.3.4 Power Requirements

- The system is powered via USB 5V (standard) or an external 9V-12V power adapter.
- LEDs operate on 3.3V - 5V with a 220Ω current-limiting resistor.

5.4 Communications Interfaces

WiFi Connectivity: Required for WebSocket communication.

6. Quality Attributes

6.1 Usability

The web dashboard will feature a clean and intuitive interface designed for users with minimal technical expertise.

LED indicators provide straightforward, immediate feedback without requiring users to interact with the software.

6.2 Performance

Temperature should update within 2 seconds of change. WebSocket response time should be under 500ms. User threshold changes will reflect within 1 second.

6.3 Security

User Authentication & Access Control Users must log in using Firebase Authentication (email/password or OAuth). Only authenticated users can modify temperature thresholds.

Data Security & Privacy All data transmissions will be encrypted using TLS/HTTPS. Firebase database access will be restricted with role-based permissions (users can only modify their own data).

System Protection & Recovery Unauthorized API requests will be rejected with authentication checks. The system will log failed login attempts and block multiple failed authentication requests.

6.4 Safety

Electrical Safety: Arduino operates at low voltage,but if it is connected to higher voltage components (e.g., relays, motors, external power supplies), there is a risk of short circuits, overheating, or electric shock. Careful usage and caution of overheating components are necessary.

To prevent static energy buildup, anti static bands when operating circuits are encouraged.

7. Internationalization and Localization Requirements

The Arduino-Based Smart Temperature Monitoring System will be adapted specifically for Italy by considering various internationalization and localization factors.

7.1 Language Support

- The web dashboard and user interface shall support Italian (Italiano) as a secondary language option.
- Translations will follow standard Italian spelling conventions to ensure clarity for Italian users.

7.2 Date, Time, and Measurement Formatting

- The system shall use the DD/MM/YYYY date format, which is the standard in Italy.
- Temperature readings shall default to Celsius (°C), as Italy follows the metric system.

7.3 Units of Measurement

- The system shall use the metric system for Italian users.
- Humidity levels shall be expressed in percentage (%), as is standard in Italy.

8. Other Requirements

The system should be compatible with future IoT expansions.

Data should be accessible remotely via a secure login system.

The project must adhere to IEEE SRS standards.

Appendix A: Glossary

Cathode: a negative charge electron entering in an electronic device.

LED: Light Emitting Diode

Appendix B: Analysis Models