1. **PURPOSE**

This project aims to build a temperature monitoring system with alarms and data logging.

1. **SCOPE**

The tasks that the system to implement performs are given as follows:

* reads temperature data from a sensor.
* records each temperature data to the external storage memory.
* triggers alarms if the measured temperature is higher or lower than the thresholds assigned by the user.
* sends the logged temperature data with its timestamp to the user prompt.

You will also find the details about the system task charasteristics below:

* reads temperature data once a second, periodically.
* records temperature data with its timestamp concerning the date and hour characteristics.
* enables the user to set the high and low temperature thresholds through a prompt.
* compares readings to the user specified thresholds and warns the user by lighing a red LED if the temperature is above or lighing a green LED, otherwise.
* sends the logged temperature values with their timestamps to the user prompt via UART protocol.
* informs the user about the inconvenient cases such as sensor disconnection and data corruption.

1. **SYSTEM DESIGN**

The system hardware is considered an integration of 6 units whose duties are given below:

**Temperature Measurement Unit:** This unit measures the ambient temperature via a sensor every 1 second periodically.

**Data Logging Unit:** This unit records each measured temperature data to an external storage memory.

**Alarm Triggering Unit:** This unit sets alarms based on configurable thresholds, compares the thresholds to the actual reading and informs the user by lighting a LED.

**Communication Unit:** This unit sends each temperature reading and the pile of data having been recorded so far.

**User Interface Unit:** This unit enables the user to enter the high and low threshold values via a PC through a console.

**Real-Time Clock Unit:** This unit enables the internal real-time clock peripheral of the MCU. This unit’s time and date settings are executed only once.

1. **THE IMPLEMENTATION OF THE SYSTEM**

All the units making up the system are made up of the COTS (Commercial off the Shelf) components. These components are preferred among the ones currently in stock or easily procurable. The system is based on the STM32F407G-DISC1 board manufactured by ST Microelectronics. The properties of this board are given below:

* STM32F407VGT6 microcontroller featuring 32-bit Arm Cortex-M4 with FPU

core, 1-Mbyte Flash memory and 192-Kbyte RAM

* USB OTG FS
* Flexible power-supply options: ST-LINK, USB VBUS, or external sources
* External application power supply: 3 V and 5 V
* On-board ST-LINK/V2-A debugger/programmer with USB re-enumeration

capability: mass storage, Virtual COM port, and debug port

The components resided on the units making up the system are outlined below.

**Temperature Measurement Unit:** This unit contains a digital sensor called DHT-11 which operates through one-wire protocol.

**Data Logging Unit:** This unit consists of an external EEPROM which records each temperature data with its timestamp. The EEPROM used in this unit is Microchip’s 24C256 which has a storage size of 256 Kilobytes. It communicates with the MCU through I2C protocol.

**Alarm Triggering Unit:** This unit consists of two different-colored LEDs. One is red, which indicates that the high threshold assigned by the user is lower than or equal to the current temperature reading. The other is green and is lit if the low threshold is higher than or equal to the current reading.

**Communication Unit:** This unit consists of an internal UART peripheral inside the MCU. This unit sets up a communication with a USB to UART controller connected to the PC.

**User Interface Unit:** This unit consists of a PC helping the user enter the threshold values via its keypad. It forms an interaction with the user by a console. All the data entered by the user and imported from the MCU is transmitted and received by CP2102 USB to the UART module.

**Real-Time Clock Unit:** This unit is an onboard peripheral of the MCU and is operated by a CR2032 Li battery.

The STM32F407 MCU used in the board operates at 192 MHz clock frequency. This frequency is obtained from an 8 MHz external crystal oscillator utilizing PLL (Phase Locked Loop) block of the MCU. The timestamp is obtained utilizing the internal RTC (Real Time Clock) unit of the MCU. This RTC unit is operated by a 32768 Hz external low crystal oscillator.

The system software consists of 10 units. These units can be classified into 4 layers which are sorted from innermost to outermost and are respectively given as follows:

**Hardware Abstraction Layer:** This layer is a bridge between the device drivers and the system hardware. The stm32f4xx\_hal\_msp.c and the stm32f4xx\_it.c fulfill this duty.

**Device Driver:** This unit forms a link between the application and the hardware abstraction layer. The units forming this layer eeprom.c rtc\_setting.c, uart.c, userInterface.c, tempSensor.c. This layer runs the devices, such as eeprom, rtc, and the uart modules.

**Operating System:** This unit is called by only application layer functions. This unit’s software is located in freertos.c file.

**Application:** This unit is on top of the system software. It calls device driver functions and operating system functions to task operations. The two file exists in this layer. One is main.c file, which is the backbone of application and common.c file which contains auxilliary functions.

The application software located in the main function executes 4 tasks, which are called user interface, temperature sensor measurement, data logging and alarm triggering, respectively. These tasks have equal priority. Therefore, they do not interrupt one another. Alarm triggering and user interface tasks execute within a shorter time than the other two tasks. Also, these tasks are more critical than the others. These are called every 100 msecs while the other two are called every 1000 msecs.

This project was performed using IAR Embedded Workbench IDE. The whole software was written in C programming language. The main framework and the hardware abstraction layer were constructed via the STM32CubeMX program.

The detailed software design of the main system and all other units are shown in Figure 1-5.

1. **THE TEST CASES VALIDATING THE SYSTEM**

**Normal operation:** The temperature sensor is read every 1 sec. If it is successful, the current date and time values are stored in the EEPROM with the temperature reading. Also, the current temperature value is compared to both high and low thresholds and the red or green LED is lit if it is not between these values. Finally, the current temperature with its timestamp is sent to the user prompt.

**User intervention:** The user disturbs the system with the commands typed via the keypad of the PC. These commands are Log Request, High Threshold, and Low Threshold commands.

**The Log Request** is entered typing ‘R’ and Line feed and carriage return characters. When this command is received by MCU, the sensor reading process is stopped. The overall temperature readings with timestamp values are read from EEPROM and all these data are sent to the user prompt in a readable format.

**The High Threshold** command is typed using 5 characters. The first character is ‘H’ followed by two-digit threshold value. If the temperature is one-digit value, it should be typed with preceding 0 char. Finally, the last two chars are Line Feed and Carriage Return ones. When this command is received, the high threshold value is updated and it is recorded to EEPROM. Alarm triggering task is executed and the red LED status is handled, accordingly.

**The Low Threshold** command is similarly typed using 5 chars. Unlike High Threshold command, its first char is’L’. The same procedures are executed except the green LED status is held accordingly.

In this stage, the random commands are entered in order to test the robustness. The “Invalid command!” error is sent to the prompt. To diversify these types of commands, the first chars are kept the same, but the succeding chars are altered. The response of the system is validated, accordingly.



**Figure 1.** The software flowchart of the system



**Figure 2.** User interface task software flowchart



**Figure 3.** Temperature measurement task software flowchart



**Figure 4.** Alarm triggering task software flowchart

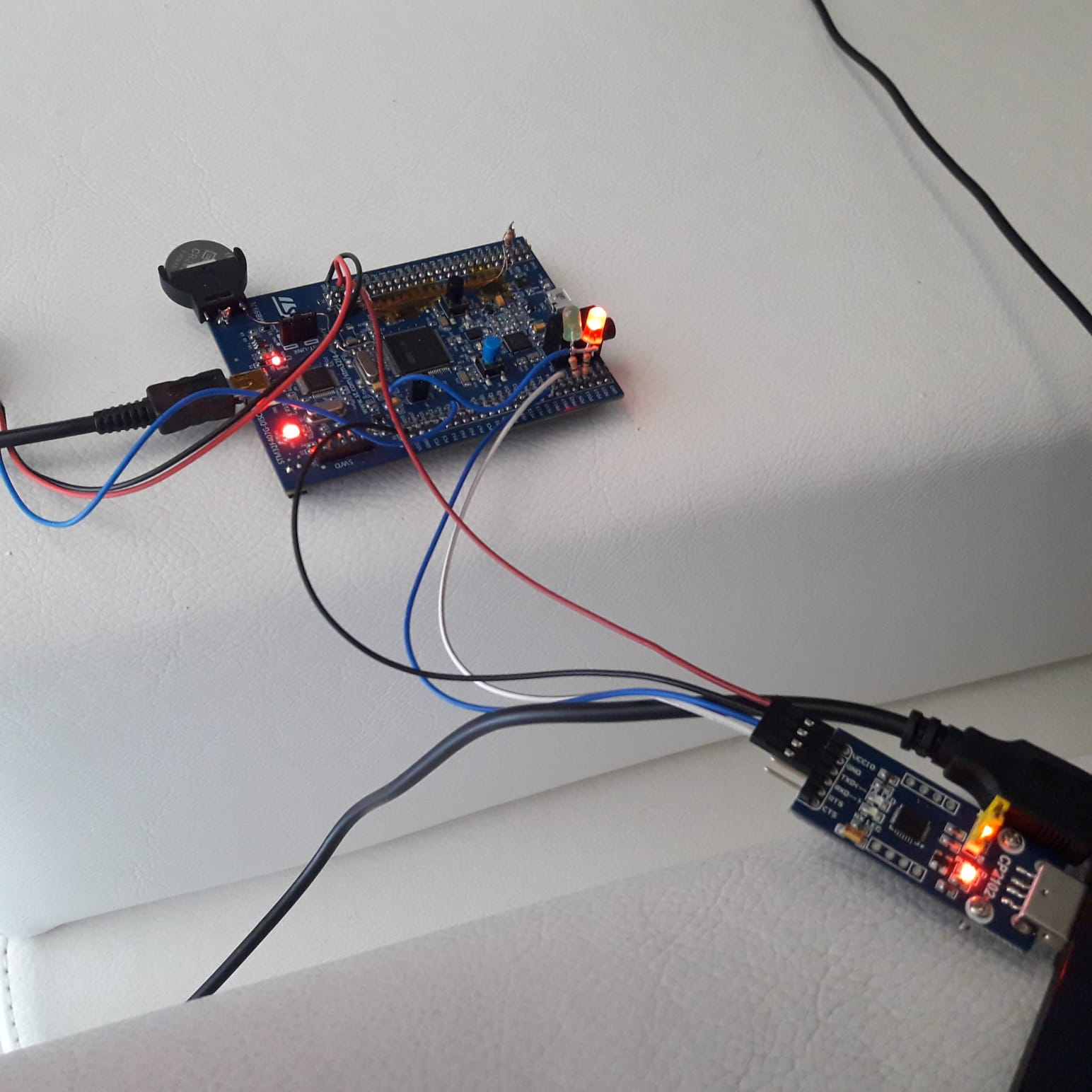


**Figure 5.** Data logging software flowchart

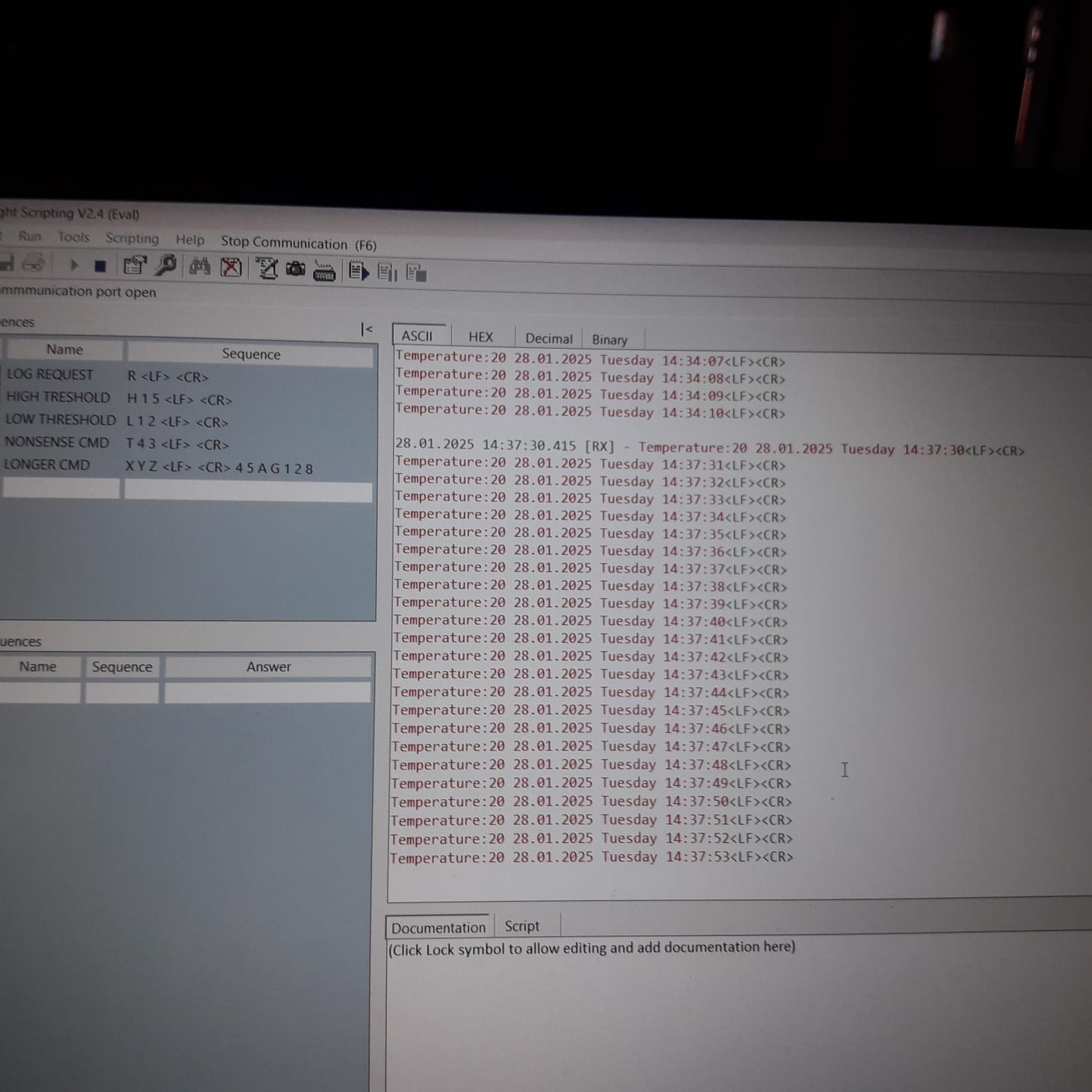
**Sensor disconnection** condition is also tested. In order to achieve this, data pin of the sensor is broken and the overall process is executed. As a result of this scenario, “Sensor Disconnection!” error is sent to the prompt.

**The corrupted data** such as the case that the temperature value is out of range (i.e. higher than 50) is simulated as if the temperature value were altered to a value higher than 50 after stopping the debugger and then resuming it. As a result of this scenario, “Invalid Sensor Data!” error is sent to the prompt.

All these test cases were covered utilizing ST-LINKV2 debugger and Docklight serial communication terminal. The PC was connected to the MCU via a CP2102 USB to UART converter module. Eventually, all these cases were satisfactorily completed and the system was assured at the developer side. The test setup and normal operation result which validated the system is shown in Figure 6, and 7 respectively.



**Figure 6.** Test setup of the system



**Figure 7.** Docklight output which validates the normal operation