Temperature Display Project

Authors:
Robert Zubek
Krystian Ruszczak

Project Goal

The main goal of this project is to fetch temperature data stored in a MySQL database and display it on an e-paper screen in the form of two temperature charts. These charts will represent the air temperature in a laboratory and some hardware temperature. They will provide a visual representation of temperature fluctuations, updating regularly to maintain accuracy. The data will be displayed over a five-hour time frame, offering an intuitive way to monitor and analyze temperature trends.

Technologies Used

Hardware:

1. Raspberry Pi 3
   - A versatile single-board computer that serves as the central processing unit for the project.
   - Responsible for fetching temperature data from database and processing it for display on the e-paper screen.
2. Waveshare 7.5-inch e-paper display
   - Used to visualize the temperature data in the form of clear, energy-efficient graphs.
   - E-paper technology ensures excellent readability in various lighting conditions and extremely low power consumption, making it ideal for continuous operation.
3. Waveshare e-paper HAT
   - A hardware attachment that facilitates seamless communication between the Raspberry Pi and the e-paper display.
   - It simplifies the connection process by providing compatible interfaces and voltage adjustments required for the display to function correctly.

Software:

1. Raspberry Pi OS
   - A lightweight Linux-based operating system optimized for Raspberry Pi hardware.
   - Provides a stable and reliable environment for running Python scripts.
   - The system is configured with SSH access for remote management and updates.
2. Python 3
   - The primary programming language used for implementing the project.
   - Python's rich ecosystem of libraries and simplicity make it ideal for handling hardware interaction and data visualization.

3. Python Libraries
    - Waveshare e-paper library: A set of pre-written functions and drivers tailored for controlling the e-paper display.
    - Pillow (PIL): Used for creating custom temperature charts and drawing elements.
    - Datetime and Time: Used for managing timestamps and implementing periodic updates to the charts.
    - Logging: Handles debugging and error reporting.
    - PyMySQL: Enables interaction with the MySQL database to fetch temperature data.

Development environment:

1. Visual Studio Code (VS Code): Used as the main IDE for writing and debugging code.
    - SSH Extension: The Remote - SSH extension in VS Code was used to connect to the Raspberry Pi remotely. It allowed editing and running code directly on the device, which was particularly useful for testing and debugging on the target hardware.

Implementation

Features Implemented:

1. System Installation
   - Successfully installed and configured the operating system on the Raspberry Pi device.
   - Ensured that all necessary system updates and drivers were installed.
2. Network Configuration
   - Configured the Raspberry Pi to operate in a local network environment.
   - Verified stable connectivity with the development machine via SSH.
3. Hardware Integration
   - Successfully interfaced with the Waveshare 7.5-inch e-ink display using the waveshare_epd library.
4. Automation
   - The program runs in a continuous loop, making it suitable for long-term use.
5. Temperature Chart Generation
   - Displayed temperature data on the Waveshare e-ink screen as two charts. The temperatures seen on the display are randomly generated by program.

Features To Be Implemented:

1. Data Retrieval from the Database
   - Integration with the actual database to fetch real temperature data was not completed, limiting the functionality to placeholder data.
2. Advanced Chart Features
   - Features like precise timestamp labeling, improved scaling for dynamic temperature ranges, and enhanced visuals (e.g., color-coded temperature alerts) remain unimplemented.
3. Full Error Handling

- Although the system handles basic operations, robust error handling for scenarios like database unavailability or unexpected input formats was not fully implemented.

Program Description

Overview

Prepared Python program displays temperature charts on a 7.5-inch e-paper screen. It periodically updates the display with new temperature data for two sources: the "Lab" and the "Device." The charts are refreshed every 15 seconds, displaying up to 5 most recent data points along with corresponding timestamps. The temperature data is currently generated randomly to simulate real readings, as the database integration is not yet implemented.

The program is run from a single file called temperature.py and uses the Waveshare e-paper library (waveshare_epd) to manage the display.

Functions:

- map_temp_to_y()

This function maps a given temperature value to its corresponding Y-coordinate on the chart. It ensures that the temperature values are properly scaled to fit within the graph's height. The function takes into account the chart's starting Y-coordinate, its height, and the defined temperature range (min_temp to max_temp), so that temperatures are visually positioned correctly on the graph.

- draw_temperature_chart()

The function generates and updates two temperature charts on the e-paper display. The function manages the layout, including dimensions, gridlines, and labels for temperature values and timestamps. Using PIL (Python Imaging Library), it creates and manipulates graphical elements before rendering them on the display.

It uses predefined parameters like chart width (270px), height (250px), and the gap between charts (50px) to arrange the visual elements. Font settings, with different sizes for chart titles (24px) and axis labels (18px), control the appearance of the text on the display.

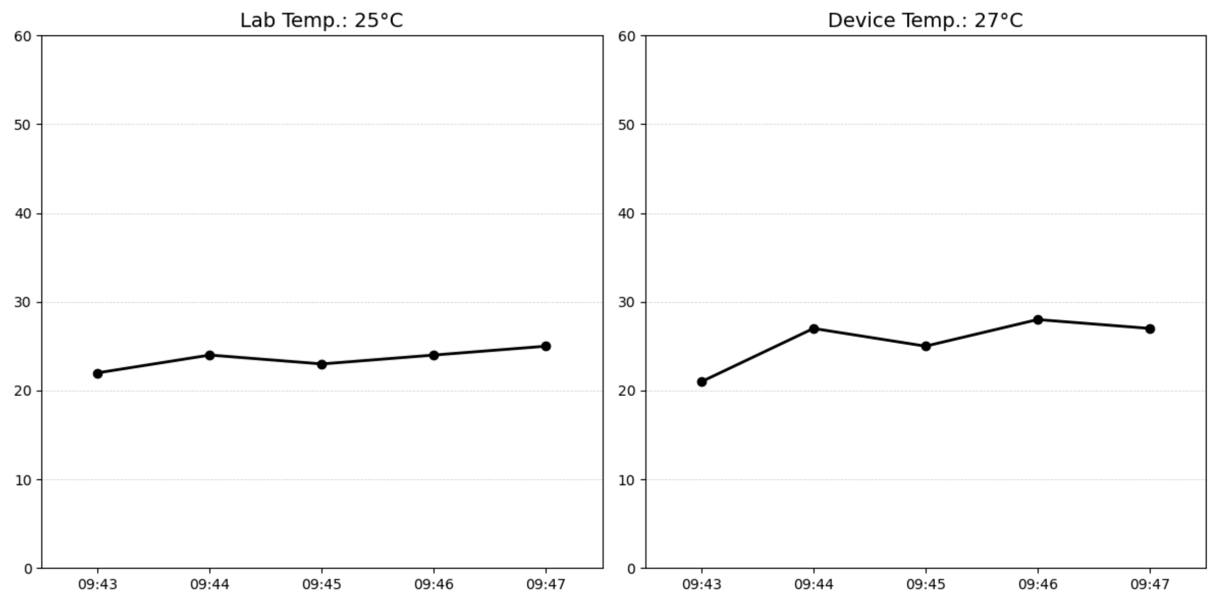
These settings, along with the temperature range (0-60°C) and step intervals (10°C), ensure the data is clearly presented.

If any unexpected error occurs while drawing the temperature graphs, the program logs the error message using the logging module.

- Main Script Logic

The program begins by initializing the e-paper display and clearing any previous content. In a continuous loop, it generates random temperature values for both the lab and the device. To keep the display readable, the program only stores the last five temperature readings with their corresponding timestamps. These values are used to update the charts by calling the draw_temperature_chart() function. After each update, the program pauses for 15 seconds using the time.sleep() before generating new data and refreshing the graphs.

The program also includes error-handling mechanism. If an IOError occurs, such as an issue with the e-paper display, the error message is logged for debugging. Additionally, it handles user interruptions. When the user presses **ctrl + c**, triggering a KeyboardInterrupt, the program logs the termination and calls the module_exit function from the e-paper library, and exits safely without leaving the display in an undefined state.



Simulated look of temperature charts showed on display.

Issues and Challenges Encountered

During the project, two primary issues were encountered. The first was related to the e-paper screen, which began malfunctioning unexpectedly. Despite attempts to troubleshoot, it was unclear whether the issue stemmed from a hardware failure or a software configuration, resulting in the display failing to properly update or even show the intended content. This posed a significant challenge, as the e-paper display is central to the project's functionality.

The second issue involved difficulties with retrieving temperature data from the MySQL database. Attempts were made to connect directly to the MySQL database or retrieve data via a dedicated page hosted on another Raspberry Pi. However, due to issues with the e-paper, these efforts could not be fully completed or resolved.