COMP-SCI 5577
Internet of Things
Mini-Project1 Report

Professor: Dr Qiuye He
Term members:
Hui Jin
Jiarui Zhu
Harshitha Bashyam
Feb 06, 2025
Github Link: https://github.com/nanxuanhui/Iot_miniproject.git

Content

Introduction

This project was implemented using an ESP32 board and a DHT11 sensor. The system reads temperature and humidity values from the sensor and provides real-time monitoring via a Serial Monitor. A text-based menu allows the user to start/stop monitoring and to set custom thresholds for the sensor values, while the LED provides visual feedback based on these thresholds.

Implementation

1. Hardware Setup

- **ESP32 Board**

The ESP32 serves as the central controller. It is responsible for reading sensor data, processing user commands via the Serial Monitor, and controlling the LED indicator.

- **DHT11 Sensor**

The sensor's + is connected to the ESP32's 3.3V supply, - is connected to the common ground, and the DATA pin is connected to GPIO4. It measures temperature (in °C) and humidity (in %) with a limited resolution (temperature in 1°C steps).

- **LED Indicator**

The LED is connected to GPIO2. The LED provides visual feedback based on the sensor data and threshold settings.

2. Software Architecture

- **Sensor Data Acquisition**: Using non-blocking timing via the millis() function, the DHT11 sensor is read every 2 seconds.
- **Serial Command Processing**: A text-based menu is implemented. The handleCommand() function parses user input commands (start, stop, settemp, sethum, show, menu) and updates the system settings accordingly.
- **LED Control**: Based on the sensor readings and threshold settings, the LED is controlled in one of three modes:
  - OFF: When conditions are within acceptable ranges.
  - ON Continuously: When readings exceed the high thresholds.
  - BLINKING: When readings fall below the low thresholds. Blinking is achieved using a non-blocking timer
- **Error Handling**: The software checks for valid sensor readings by verifying that the returned values are non-negative. Any error (indicated by negative values) is reported via the Serial Monitor.

3. System Flow

- ### Startup:

a. The system initializes Serial communication and sets the LED control pin as an output.
b. The startup menu is printed to the Serial Monitor.
c. Monitoring is off by default.

- ### User Interaction:

a. The user inputs commands via the Serial Monitor.
b. The command processor updates the monitoring state and threshold values.
c. When the user enters start, monitoring begins; when stop is entered, monitoring is paused and the menu is re-displayed.

- **Data Acquisition and Display:**

a. When monitoring is active, sensor readings are taken at regular intervals.
b. The acquired temperature and humidity are displayed in real time.

- **LED Control:**

a. The program compares sensor values with custom thresholds.
b. Based on the comparison, the LED is set to one of the three modes (OFF, ON, BLINKING).
c. For blinking, the code toggles the LED state using a timer to ensure a fast response without delaying sensor updates.

4. Communication and Interactivity

- **Serial Monitor Interface**: The user communicates with the ESP32 through the Arduino Serial Monitor. Commands are input as text, and the ESP32 responds with status messages and sensor readings.
- **Dynamic Configuration**: Users can change temperature and humidity thresholds on the fly. This allows for real-time tuning of the LED indication based on environmental conditions.
- **Feedback**: Each command input is processed immediately, and feedback ("Monitoring started." or the current threshold settings) is provided, ensuring a responsive user interface.

5. Non-blocking Timing Implementation

- **millis() Function**: Instead of using blocking functions like delay(), the project uses millis() to keep track of elapsed time.
- **Sensor Read Interval**: Sensor data is read every 2 seconds based on the elapsed time check.
- **LED Blink Control**:The LED blinking is controlled using its own timer (250 ms), allowing for simultaneous sensor updates and LED control without one interfering with the other.

LED Indication Logic

1. The LED in this project is used to provide immediate visual feedback regarding the environmental conditions. The indication logic is based on comparing the sensor readings (temperature and humidity) with user-configurable thresholds. The LED operates in three distinct modes:
   a. **OFF (LED is not lit):**
   - **Condition**: When both the temperature and humidity readings are within the acceptable range—that is, above the low thresholds and below the high thresholds.

- **Explanation**: This state indicates that the environment is within the desired parameters. No abnormal condition is detected, so the LED remains off.

b. ON Continuously (Steady LED):

- **Condition**: When the temperature exceeds the high threshold (default is 30°C) or when the humidity exceeds the high threshold (default is 70%).
- **Explanation**: A continuously lit LED signifies that the environment is too hot or too humid. This constant illumination serves as a clear alert to the user that conditions may be potentially hazardous or require attention.

c. BLINKING (Flashing LED):

- **Condition**: When the temperature falls below the low threshold (default is 15°C) or when the humidity falls below the low threshold (default is 30%).
- **Explanation**: A blinking LED indicates that the environmental conditions are below the desired level (e.g., too cold or too dry). The flashing action provides an immediate and noticeable signal that the current conditions might need corrective measures (such as increasing temperature or humidity).

2. Details

a. Timing Control:

The blinking effect is achieved using non-blocking timing with the millis() function. The LED toggles its state (on/off) at a set interval (every 250 ms), which produces a flashing effect while still allowing the system to continue reading sensor data.

b. Threshold Comparison:

- **High Thresholds**: Determine if the LED should be on continuously.
- **Low Thresholds**: Determine if the LED should blink.

c. Priority Logic:

The logic is designed with a priority order. If either sensor reading exceeds its high threshold, the LED is set to continuously ON. If this condition is not met, then if either reading falls below its low threshold, the LED is set to blink. If neither condition applies, the LED remains OFF.

d. User Configurability:

The threshold values for both temperature and humidity can be modified dynamically via the Serial Monitor. This allows users to fine-tune the LED indication behaviour based on the specific environmental conditions or their preferences.

e. Benefits:

- **Immediate Visual Feedback**: Users can quickly assess whether the environment is within acceptable ranges or if corrective actions are needed.
- **Simplicity**: Using one LED to represent three different states minimizes hardware complexity while still conveying meaningful information.

Challenges Faced

1. Ensuring that the code was written in the correct order—specifically, processing and displaying temperature before humidity. If the code is not structured in this specific order, the displayed values may become reversed (i.e., the humidity value may appear in place of temperature and vice versa). This required careful attention during

2. The DHT11 sensor provides integer temperature readings with a 1°C resolution. This limited precision sometimes made it hard to observe minor fluctuations in stable environments.
3. Implementing non-blocking timing with millis() to manage both sensor readings and LED blinking without interfering with Serial communication required careful design.
4. Different versions of the DHT11 library may provide varying methods for accessing sensor data. Ensuring the correct functions were used and that the temperature and humidity readings were not swapped was one challenge.

Conclusion

This project successfully demonstrates the integration of an ESP32 board with a DHT11 sensor to provide real-time environmental monitoring and visual feedback. Through careful design and implementation, the system reads temperature and humidity data, compares the values against user-defined thresholds, and then uses distinct LED states to indicate whether the environmental conditions are within, above, or below the desired ranges.

The development of a user-friendly serial menu that allows dynamic control of the monitoring process. Users can start or stop the sensor readings, adjust threshold values, and immediately receive feedback on the current environmental conditions.

The system's non-blocking timing mechanism—implemented using the millis() function—ensures that sensor data acquisition and LED control operate concurrently without interfering with each other or the serial communication.

While the project encountered challenges such as ensuring the correct order of processing sensor data (to prevent temperature and humidity values from being swapped), library compatibility issues, and limitations inherent to the DHT11 sensor's resolution, these challenges provided valuable learning opportunities. They not only prompted the implementation of robust error handling and careful debugging but also highlighted areas for future improvement.

Future enhancements could include integrating higher-precision sensors for more accurate environmental monitoring, adding wireless connectivity (such as Wi-Fi or Bluetooth) to allow remote access to sensor data, and refining the serial interface or even developing a graphical user interface (GUI) for improved usability. These improvements would extend the project's functionality and applicability in more demanding scenarios.

Reference

[1] https://lastminuteengineers.com/esp32-pinout-reference/