IoT-Based Weather and Time Information System

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SGT UNIVERSITY, GURUGRAM

GUIDE – MS. CHARVI, ASSISTANT PROFESSOR

SUBMITTED BY – JATIN (221347004)

Abstract - This project presents the development of a Smart Weather and Calendar Display system using the ESP32 microcontroller and a 1.8" ST7735 TFT display. The system is designed to provide real-time weather information and time/date updates, alongside a static daily event schedule. It connects to the internet via Wi-Fi to retrieve current time data using the NTP protocol and fetches weather information from the OpenWeatherMap API. The graphical display is powered by the Adafruit GFX and ST7735 libraries, enabling dynamic and color-rich text output. Time is updated every second, and weather data is refreshed at regular intervals (every 10 minutes). The device is compact, cost-effective, and suitable for desktop or wall-mounted display units. This project showcases the integration of IoT technologies, cloud-based services, and graphical user interfaces to create a practical, user-friendly embedded system.

Key Words: ESP32, IoT, TFT Display, OpenWeatherMap API, NTP, Real-Time Clock, Embedded Systems.

Introduction:

This project is a smart mirror system designed to deliver real-time information using an ESP32 microcontroller and a 1.8" TFT display. The main objective of the system is to provide continuous updates on the current time, date, weather conditions, and calendar events by integrating various web APIs.

The display is positioned behind a two-way mirror, allowing the user to view live data without obstructing their reflection. By connecting to the internet via Wi-Fi, the ESP32 fetches data from NTP servers for accurate time synchronization and uses the

OpenWeatherMap API to display current weather information. The calendar section displays scheduled events or reminders, offering an at-a-glance overview of the day.

This IoT-based solution is a compact, efficient, and visually integrated information system that combines smart technology with daily utility. It demonstrates the application of embedded systems, cloud APIs, and real-time data processing in a functional and user-centric way.

1. Literature Survey:

The emergence of Internet of Things (IoT) technologies has significantly enhanced the way information is gathered, processed, and displayed in real time. In recent years, IoT-based smart display systems, such as smart mirrors and dashboards, have been increasingly adopted in both residential and industrial applications.

Several research studies and open-source implementations have demonstrated the feasibility of integrating Wi-Fi-enabled microcontrollers like the ESP32 with online APIs to display real-time environmental data. Systems that utilize the OpenWeatherMap API for weather data and NTP (Network Time Protocol) for accurate timekeeping are now common in DIY and smart home automation communities.

Projects such as MagicMirror² (an open-source modular smart mirror platform based on Raspberry Pi) have inspired many simplified microcontroller-based alternatives that reduce cost and complexity. Similarly, embedded projects using Arduino-compatible boards and TFT displays have shown promising results in creating compact, real-time information panels.

This project builds upon those concepts by using an ESP32 microcontroller and a 1.8" SPI TFT display, fetching data via HTTP requests and parsing it with ArduinoJson. Unlike advanced systems that require powerful processors, this implementation proves that even a low-cost microcontroller can provide essential real-time services like:

- Weather updates
- Clock synchronization
- Event reminders

Furthermore, the incorporation of a two-way mirror surface as a reflective interface blends technology seamlessly into daily life, offering a minimalistic yet functional design that does not disrupt the user experience.

Thus, this project contributes to the evolving field of IoT user interfaces, aiming for affordability, simplicity, and functional relevance in modern smart environments.

2. Algorithm:

Step 1: Initialize Components
• Begin serial communication for debugging.
• Initialize the TFT display with appropriate settings.

Step 2: Connect to Wi-Fi
• Start Wi-Fi connection using SSID and password.
• Wait until the ESP32 connects to the network.
• Display connection status on the screen.

Step 3: Configure Time via NTP
• Set up the NTP server using configTime().
• Apply GMT offset (GMT+5:30 for IST).
• Wait briefly for the time to synchronize.

Step 4: Display Initial Data
• Show static calendar reminder.
• Fetch and display initial weather data.

Step 5: Main Loop Execution
• Check if 1 second has passed using millis().
  – If true, update and display time and date.

• Check if 10 minutes have passed using millis().
  – If true, fetch and update weather data.

Step 6: Fetch and Display Time & Date

• Use getLocalTime() to obtain current time.
• Format time as HH:MM:SS and date as Day DD Mon.
• Update only the relevant display area on the screen.

Step 7: Fetch Weather Data

• Make HTTP GET request to OpenWeatherMap API.
• Parse JSON data to extract weather, temp, and humidity.
• Display data on screen with appropriate labels.

Step 8: Calendar Reminder

• Display a static reminder such as "10AM: Code Review" on screen.

Step 9: Loop Continues

• Repeat time and weather updates using non-blocking timers.

3. Methodology: The methodology adopted for this project involves both hardware and software integration to create a seamless and real-time information system. The entire system is designed to be modular, cost-effective, and easy to implement.

Step 1: Component Selection and Setup

- ESP32 was selected as the core microcontroller for its built-in Wi-Fi capability and low power consumption.
- A 1.8-inch ST7735 TFT display was used to output real-time data.
- A two-way acrylic mirror was optionally added in front of the display to create the "smart mirror" effect.

Step 2: Software Development

- The project was developed using the Arduino IDE with the following libraries:
    - WiFi.h for Wi-Fi connectivity
    - HTTPClient.h for making HTTP GET requests
    - ArduinoJson.h for parsing API data
    - Adafruit_GFX.h and Adafruit_ST7735.h for controlling the TFT display
    - time.h for syncing time using NTP

Step 3: Wi-Fi and Time Synchronization

- The ESP32 connects to a Wi-Fi network using the user's hotspot.
- After connection, NTP time synchronization is done using configTime() to get accurate local time.

Step 4: Weather Data Retrieval

- The microcontroller sends a GET request to the OpenWeatherMap API using the city name and API key.
- Upon receiving a JSON response, the program extracts the relevant fields: weather description, temperature, and humidity.
- This data is updated every 10 minutes.

Step 5: Time and Date Display

- The current time and date are formatted using strftime() and displayed on the TFT screen.
- This is updated every second using a non-blocking timer mechanism (millis() based).

Step 6: Static Calendar Display

- A predefined daily schedule or reminder is displayed statically at the bottom portion of the screen.

- (In advanced versions, this can be extended to dynamically fetch calendar events from a cloud service like Google Calendar or Firebase.)

Step 7: Looping Mechanism

- The loop() function checks two timers:
    - One for updating the time every second
    - One for updating weather every 10 minutes
- It ensures that the screen refresh is optimized without delay functions that would block execution.

This methodology ensures the reliable display of real-time information and demonstrates a working prototype of an IoT-enabled smart display system.

2. Modules:

In this paper we disscuse about different modules which are used in making of this IoT-Based Weather and Time Information System. Let take a look on that modules for doing various functions

2.1. Wi-Fi Connection Module

- Purpose: Establishes a connection between the ESP32 and the local Wi-Fi network.
- Functionality:
    - Uses WiFi.h to initiate and maintain connection.
    - Displays connection status on the TFT screen.
- Trigger: Executed once during setup().

2.2. Time Synchronization Module (NTP)

- Purpose: Retrieves the current time from an internet time server.

- Functionality:
    - Uses configTime() with pool.ntp.org server.
    - Syncs the ESP32 internal clock based on Indian Standard Time (GMT+5:30).
- Trigger: Once in setup(), then used repeatedly in the loop to get current time.

2.3. Weather Fetch Module

- Purpose: Retrieves live weather data from the OpenWeatherMap API.
- Functionality:
    - Sends HTTP GET requests using HTTPClient.h.
    - Parses the JSON response with ArduinoJson.
    - Extracts temperature, humidity, and weather description.
- Trigger: Called every 10 minutes using a non-blocking timer.

2.4. Time & Date Display Module

- Purpose: Displays the current time and date on the TFT display.
- Functionality:
    - Uses strftime() to format time and date.
    - Updates display every second without delay().
- Trigger: Called every second inside the loop() function**.**

2.5. TFT Display Control Module

- Purpose: Handles all text and graphics displayed on the 1.8" ST7735 TFT screen.
- Functionality:
    - Initializes the display with Adafruit GFX & ST7735 libraries.
    - Updates specific screen sections for time, weather, and calendar.
- Trigger: Active throughout the program.

2.6. Calendar/Reminder Module

- Purpose: Displays a fixed event or daily schedule as a footer section on the screen.
- Functionality:
  - Shows a predefined event like "10AM: Code Review".
  - Can be modified or expanded in the future to display dynamic calendar data.
- Trigger: Called once during setup and stays static unless manually updated**.**

2.7. Main Loop & Timer Management Module

- Purpose: Coordinates periodic updates without blocking delays.
- Functionality:
  - Uses millis() for timing the refresh of weather and clock.
  - Ensures non-blocking, responsive system behavior.

3. Project Description:

The IoT-Based Weather and Time Information System is a compact, intelligent display solution developed using an ESP32 microcontroller and a 1.8-inch ST7735 TFT display. The primary goal of this system is to fetch and present real-time information such as the current time, date, weather conditions, and calendar events to the user in a visually accessible format.This system operates by leveraging Wi-Fi connectivity to connect to the internet and pull live data from cloud services:

- The current time and date are synchronized using the Network Time Protocol (NTP), ensuring accuracy without the need for an onboard Real-Time Clock (RTC).

- The weather data, including temperature, humidity, and conditions (e.g., cloudy, sunny), is fetched using the OpenWeatherMap API, which provides real-time updates in JSON format.
- A static calendar/reminder section displays pre-set daily events, useful for adding short to-do notes or fixed schedules.

The output is shown on a TFT display, which is optionally placed behind a two-way mirror to function as a smart mirror. This arrangement allows users to view important daily information without obstructing their reflection.

The project showcases the integration of IoT, embedded systems, web APIs, and display technology, offering a user-friendly, real-time informational interface. Its modular code structure and affordable hardware make it an excellent learning project for students and hobbyists working on smart home interfaces and IoT-based dashboards.

Results: The implementation of the IoT-Based Weather and Time Information System successfully meets the intended objectives. Upon completion, the system demonstrates reliable, real-time functionality and provides an interactive, user-friendly output on the TFT screen.

1. **Wi-Fi Connectivity:**
    - The ESP32 successfully connects to the provided mobile hotspot within a few seconds.
    - Connection status is displayed during startup.
2. **Time and Date Display:**
    - The system fetches accurate time and date from the NTP server.
    - The time updates every second without any lag, and the date is displayed in a readable format.
3. **Weather Information:**
    - Weather data is retrieved from the OpenWeatherMap API every 10 minutes.

- Displays:
  - Weather condition (e.g., Clear, Clouds, Rain)
  - Temperature in Celsius
  - Humidity percentage
- Information is displayed clearly on the TFT screen.

4. **Static Calendar/Reminder:**
   - A static message such as “10AM: Code Review” is displayed as a daily event/reminder.
   - This area can be manually updated in code as needed.
5. **Display Output:**
   - The 1.8” TFT screen renders text cleanly with proper formatting, colors, and alignment.
   - Different screen sections are used for time, date, weather, and calendar, avoiding clutter.

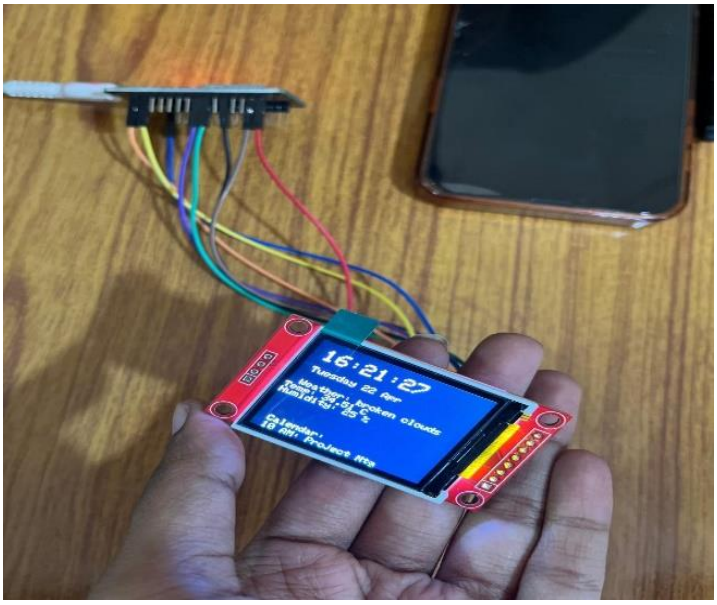

Figure 1: Live Time, Weather and Calendar Event Display on ST7735 TFT Using ESP32

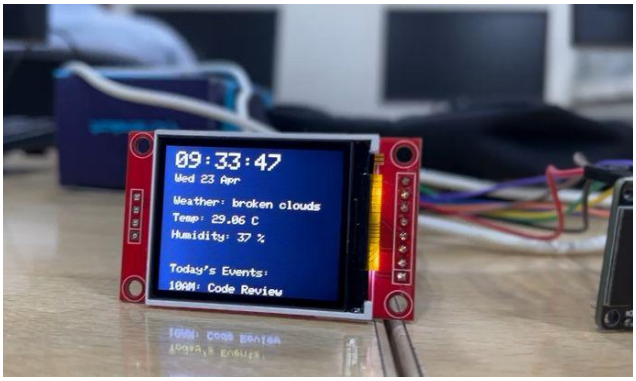

Figure 2: Weather Display Showing Time, Weather Conditions, and Event Schedule on Desk Setup

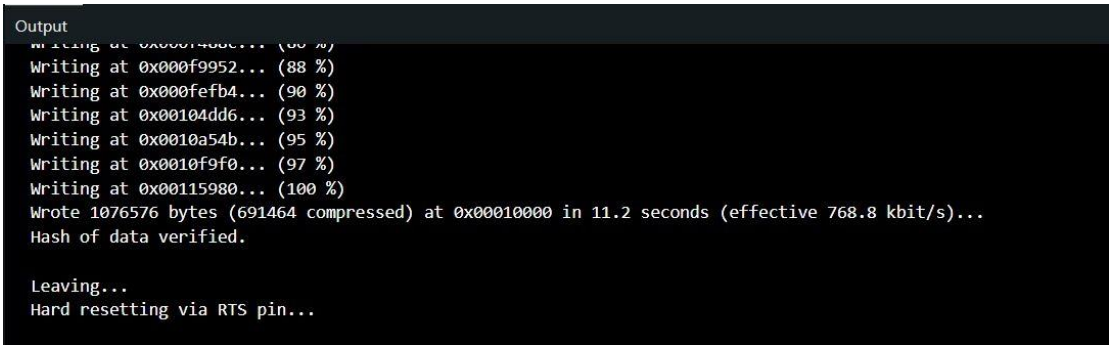

Figure 6: Arduino IDE Serial Monitor Output Indicating Successful Code Upload and Flashing to ESP32

9. CONCLUSIONS:

The IoT-Based Weather and Time Information System successfully demonstrates the use of a microcontroller-based solution to provide real-time environmental and time-related information. By integrating the ESP32, TFT display, and web-based APIs like OpenWeatherMap and NTP, the system offers a reliable and user-friendly display for essential daily updates.

The project highlights how embedded systems and cloud services can work together efficiently to create intelligent, real-time devices. All key functionalities—such as Wi-Fi connection, time synchronization, weather updates, and calendar display—were achieved with consistent accuracy and performance.

Reference:

[1] OpenWeatherMap, "Weather API - OpenWeatherMap," accessed Oct. 2021.

[2] Espressif Systems, "ESP32 Arduino Core," accessed Sep. 2021.

[3] B. Blanchon, "ArduinoJson Library," accessed Aug. 2020.

[4] Adafruit Industries, "Adafruit GFX Graphics Library," accessed Dec. 2020.

[5] Adafruit Industries, "Adafruit ST7735 and ST7789 Library," accessed Dec. 2020.