PROJECT ABSTRACT

E-Paper Weather Display Station using ESP32

A paper-like weather station — ESP32 fetching live forecast from a public API onto a zero-power-hold e-paper display with local DHT22 sensing — delivered as a built-to-order student project with firmware, API setup guide, report, PPT and viva kit.

01. Title

E-Paper Weather Display Station using ESP32

02. Introduction / Background

Electrophoretic (e-paper) displays hold an image with zero power: charged pigment particles sit in their last position until the next refresh, so the screen shows the weather even unplugged. They reflect ambient light like paper, making them perfectly readable in direct sunlight where LCDs wash out — at the cost of slow, full-refresh updates. The ESP32's HTTPS-capable WiFi stack lets a microcontroller consume modern REST APIs directly: a GET request to a weather service returns JSON that the firmware parses into temperature, humidity and conditions. The project pattern — wake, fetch, render, sleep — is the canonical battery IoT loop.

03. Problem Statement

LCD weather stations glow constantly, draw continuous power and become unreadable in the sunlight they are meant to report on. Phone apps show the weather but disappear into the phone. What a desk, wall or farm shed needs is a glanceable, paper-like display that updates itself a few times an hour and sips power — a project that teaches HTTPS, JSON parsing and e-paper framebuffer rendering on real hardware. The deliverable must document the API setup honestly, since the free API key is the one part the builder must obtain themselves.

04. Objectives

- Fetch current weather and forecast from a public weather API over HTTPS.
- Parse the JSON response with ArduinoJson into display-ready fields.
- Render a clean layout (temperature, humidity, conditions, forecast) to an e-paper framebuffer over SPI.
- Add local temperature–humidity sensing with a DHT22 for an indoor/outdoor comparison.
- Cycle on a deep-sleep timer (default 30 minutes) between updates.
- Make the location configurable without recompiling.

- Hold the last image with zero power between refreshes.

05. Proposed Methodology

The e-paper module is driven first with static test images to validate the SPI framebuffer pipeline and the full-refresh behaviour. The firmware then layers an HTTPS client: request, response buffering and ArduinoJson parsing of temperature, humidity, condition code and forecast fields, with error handling for network and API failures. The DHT22 provides the local reading rendered alongside the API data. Deep-sleep wakeup on a timer closes the loop; location and API key live in configuration constants documented in the setup guide. Testing covers failed-fetch behaviour (old image retained, retry next cycle), refresh timing and overnight cycle stability.

06. System Architecture / Working

The RTC timer wakes the ESP32, which joins WiFi, performs the HTTPS API request, and parses the JSON into weather fields while reading the DHT22 for local conditions. The layout engine composites temperature, humidity, condition icon, forecast and local reading into the framebuffer, then pushes the full refresh to the e-paper panel. The controller re-enters deep sleep until the next cycle. Between refreshes the display holds the last image with no power draw — the visible state is always the last successful update, with the timestamp showing its age.

07. Hardware / Software Requirements

- ESP32 DevKit (HTTPS client, deep sleep).
- SPI e-paper display module (2.9-inch class).
- DHT22 temperature–humidity sensor (±0.5 °C).
- Free-tier key for a public weather API (buyer-obtained; guide included).
- USB power or battery for the sleep-cycling design.
- Software: Arduino IDE (C/C++ firmware), ArduinoJson library.

08. Expected Outcomes

A working station that wakes on its timer, updates the e-paper with current weather plus local sensor readings, and sleeps — the display remaining perfectly readable in sunlight with the image held power-free between updates. The API setup guide takes the builder from key signup to first successful fetch, and failed fetches degrade gracefully by retaining the last image.

09. Applications

- Home and office glanceable weather displays.
- Classroom IoT teaching platform.
- Farm-shed weather boards (with enclosure).

- Desk gadgets and tech gifts.
- Low-power IoT design demonstrations.

10. Future Scope

Multi-day forecast graphs, partial-refresh fast updates and tri-colour e-paper panels would enrich the display. Solar charging with a larger battery, indoor air-quality sensors (CO2, PM2.5) and a wall-mount enclosure complete the path to a permanent installation.