APRIL 9, 2025

LIVERPOOL HOPE UNIVERSITY 1844

INTERNET OF THINGS COURSEWORK 1

IOT DEVICE CASE STUDY: SMART WEATHER MONITORING SYSTEM USING ESP-32

SAFEER AHMED
MSC ROBOTICS ENGINEERING

Contents

1. Introduction

Weather monitoring plays a crucial role in various sectors, from agriculture to transportation, helping us understand and predict environmental conditions. With advancements in technology, weather monitoring systems have evolved from traditional manual methods to more efficient and automated IoT-based systems. This report presents an in-depth analysis of an ESP32-based smart weather monitoring system that collects, processes, and transmits environmental data to users in real time.

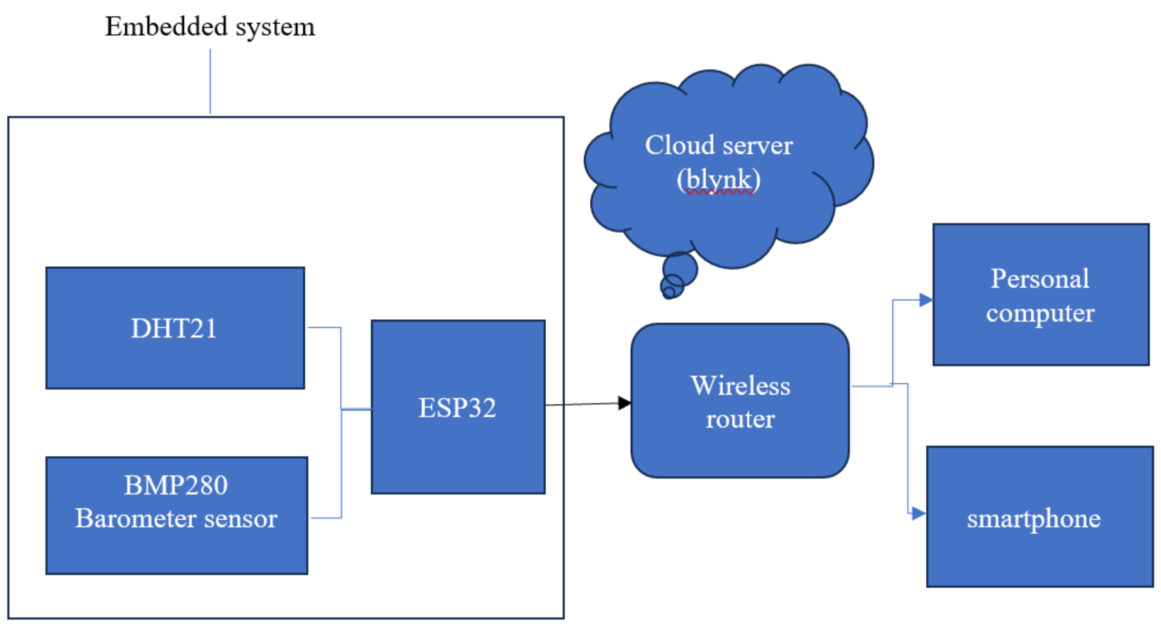
The core of the system is the ESP32 microcontroller, which interacts with various sensors, including the DHT21 for temperature and humidity, and the BMP280 for barometric pressure. These sensors continuously gather data on environmental conditions, which the ESP32 processes and sends to the cloud for remote monitoring. Using platforms like Blynk Cloud, users can access and visualize this data through smartphones or computers, enabling them to monitor weather conditions from anywhere in the world.

This report explores the design, working, and components of the ESP32-based weather monitoring system, detailing the role of each sensor and how they interact with the microcontroller. Additionally, it discusses the embedded system's power management features, ensuring efficient operation even in remote areas where power supply may be limited. The communication protocols, data analysis, and potential applications of this system in sectors such as agriculture, smart cities, and transportation are also examined.

By integrating hardware and software components, this smart weather monitoring system not only provides real-time data but also contributes to enhanced decision-making in various industries, making it a valuable tool for modern-day environmental management.

2. Overview of Key Components:

2.1 block diagram:



Block diagram of esp32 based smart weather monitoring system (Weather Monitoring with ESP32).

2.2 Working of ESP32-Based Weather Monitoring System:

The weather monitoring system is designed using an embedded system consisting of sensors and an ESP32 microcontroller. It collects environmental data and transmits it to users via the internet. Here's a step-by-step explanation of its working (Development of an ESP-32 Microcontroller Based Weather Reporting Device, no date).

2.2.1 Sensors

Collect Environmental Data

- **DHT21 Sensor**: This sensor measures temperature and humidity in the surrounding environment.
- **BMP280 Sensor**: This is a barometric pressure sensor that also provides temperature and altitude data. It is useful for atmospheric pressure monitoring.

2.2.2 ESP32 Microcontroller (Core Unit)

- The ESP32 acts as the central processing unit.
- It reads the sensor values from both DHT21 and BMP280 using appropriate libraries and communication protocols (like I2C or digital input).
- The ESP32 processes this data and prepares it to be sent to the cloud.

2.2.3 Wireless Communication

- The ESP32 is Wi-Fi enabled and connects to a wireless router using Wi-Fi credentials configured in the firmware.
- Through this router, the ESP32 sends data to an online cloud server**.**

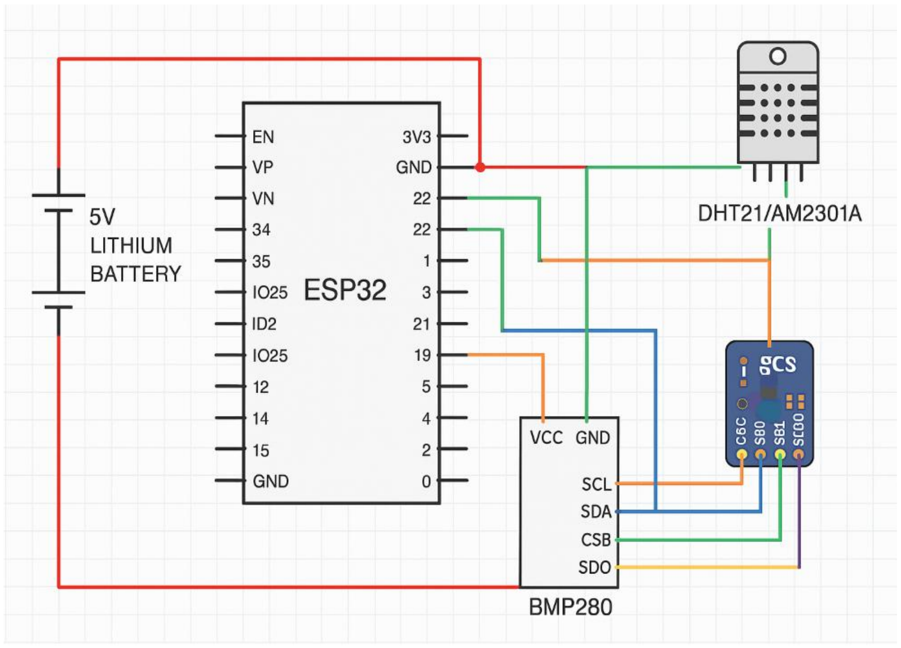
2.2.4 Cloud Server (Blynk)

- The system uses Blynk Cloud, a popular IoT platform, to store and visualize real-time weather data.
- Data is sent to Blynk using APIs and appears in the form of charts, gauges, or displays on the Blynk dashboard.

2.2.5 Remote Monitoring

- Users can access the weather data via:
    - Smartphones (using the Blynk mobile app)
    - Personal Computers (through web-based dashboards or apps)
- This enables real-time weather monitoring from anywhere in the world with internet access.

2.3 Circuit diagrams an ESP-32 based smart weather monitoring system:



circuit diagram of ESP32 based smart weather monitoring device (Weather Monitoring with ESP32).

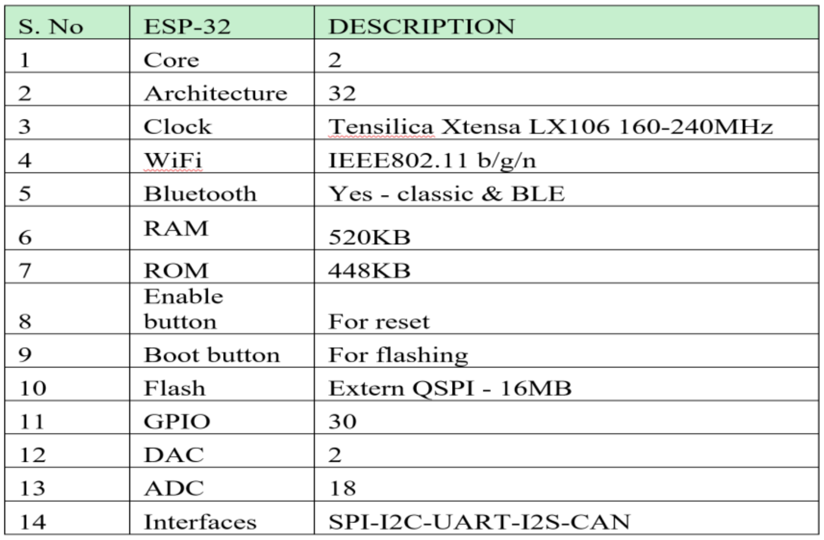
2.4 Embedded System:

An embedded system combines hardware and software to perform specific tasks. In a smart weather monitoring system, sensors connected to the ESP32 provide edge processing and cloud data transmission. The ESP32 is preferred for its low-power consumption and long-range Wi-Fi, making it ideal for such devices (Sdiopr, 2021).

2.4.1 ESP32 Microcontroller:

The ESP-32 is very powerful and affordable option with built-in long-range Wi-Fi and Bluetooth, ideal for such IoT applications. In smart monitoring device, it collects data from sensors (temperature, humidity etc.), process it using its onboard CPU, and transmits it wirelessly to the cloud/server or an LCD display connected to the indoor unit locally. Its ability to collect, process and continuously tracking weather in real time without consuming much power make it ideal choice (Episensor, n.d.).

Specifications:



S. No	ESP-32	DESCRIPTION
1	Core	2
2	Architecture	32
3	Clock	Tensilica Xtensa LX106 160-240MHz
4	WiFi	IEEE802.11 b/g/n
5	Bluetooth	Yes - classic & BLE
6	RAM	520KB
7	ROM	448KB
8	Enable button	For reset
9	Boot button	For flashing
10	Flash	Extern QSPI - 16MB
11	GPIO	30
12	DAC	2
13	ADC	18
14	Interfaces	SPI-I2C-UART-I2S-CAN

ESP32 Microcontroller Specifications (Electronics Hub, 2022).

2.4.2 SENSORS:

I. **Temperature and Humidity Sensor:** We use a low power and reliable temperature sensors such as DHT21/AM2301A, while having options like Dht11 and Dht22. DHT21 is also known a AM2301A is a digital temperature and humidity sensor which offers reliable, affordable and accurate environmental sensing.

DHT21/AM2301A (hobbycomponents, n.d.).

Specifications:

Feature	Specification
Temperature Range	-40°C to +80°C
Temperature Accuracy	±0.5°C
Humidity Range	0% to 100% RH
Humidity Accuracy	±3% RH (at 25°C)
Operating Voltage	3.3V to 5.5V
Signal Output	Digital (single-bus interface)
Sampling Rate	1 reading every 2 seconds

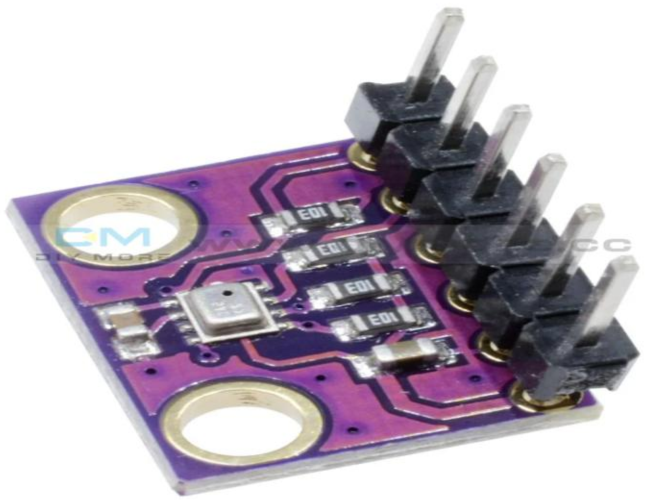
Working Principle:

- **Humidity Sensing**: Uses a capacitive sensor element to measure relative humidity in the air.
- **Temperature Sensing:** Uses a thermistor to detect temperature changes.
- An internal ADC (Analog to Digital Converter) converts analog signals to digital output.
- Communicates with microcontrollers (e.g., Arduino, ESP32) via a single-wire protocol, making it easy to integrate.

Why we use DHT21/AM2301A in weather monitoring systems:

- High Accuracy: Perfect for precise monitoring in smart agriculture, greenhouses, and home automation.
- Low Power: Ideal for battery-powered devices.
- Easy Integration: Works with platforms like Arduino and ESP32 without needing ADC.
- Compact and Affordable: Small, cost-effective, and suitable for portable systems.
- Real-Time Monitoring: Provides live updates and triggers alerts for weather changes.

II. **Barometer (BMP280 3.3V):** The BMP280 is a digital barometric pressure sensor develop by Bosch Sensor Tec. It is upgraded version of BMP180. It is used to collect pressure related information for the weather monitoring device accurately.



BMP280 Barometer sensor (bosch, n.d.)

Specifications:

Feature	BMP280 (3.3V Version)
Supply Voltage	3.3V (ideal for ESP32)
Communication Interface	I²C and SPI (selectable)
Pressure Range	300 hPa to 1100 hPa
Pressure Accuracy	±1 hPa
Temperature Range	-40°C to +85°C
Temperature Accuracy	±1.0°C
Resolution	0.01 hPa (pressure), 0.01°C (temperature)
Current Consumption	~2.7 µA (in normal mode)
Operating Mode	Ultra-low power to high resolution modes

Working Principle:

- **Pressure Sensor**: Measures absolute atmospheric pressure using a MEMS sensor**.**
- **Temperature Sensor**: Measures ambient temperature for environmental and compensation use.
- The pressure readings can be used to calculate altitude and weather trends**.**
- The sensor contains a calibration EEPROM**,** and data is processed via onboard ADC.

Why BMP280 is ideal for ESP32-based weather monitoring:

- **Compatibility:** operates on only 3.3v, which means it directly compatible with ESP32 without level shifter.
- **Support SPI and 12C communication protocol:** That means it can easily connect directly with ESP 32.
- Low power consumption and high data accuracy

2.5 Software analysis:

The C++ code for ESP32 is already uploaded on site (Blynk, 2025), the code start reads temperature, humidity, rain, pressure, and light using sensors. It connects to Wi-Fi and Blynk IoT, displays values on an LCD, and sends data to Blynk cloud. The loop continuously updates sensor readings and controls a virtual LED based on LDR input while maintaining a network connection. We have to create the Blynk mobile web dashboard, we can open blynk site on laptop or can download app from play store in mobile, after creating account add gauge for different parameters like temperature, humidity etc. When device will turn on it will show respective data on indoor screen and also it will send data to blynk cloud.

2.6 Applications of a Smart Weather Monitoring System:

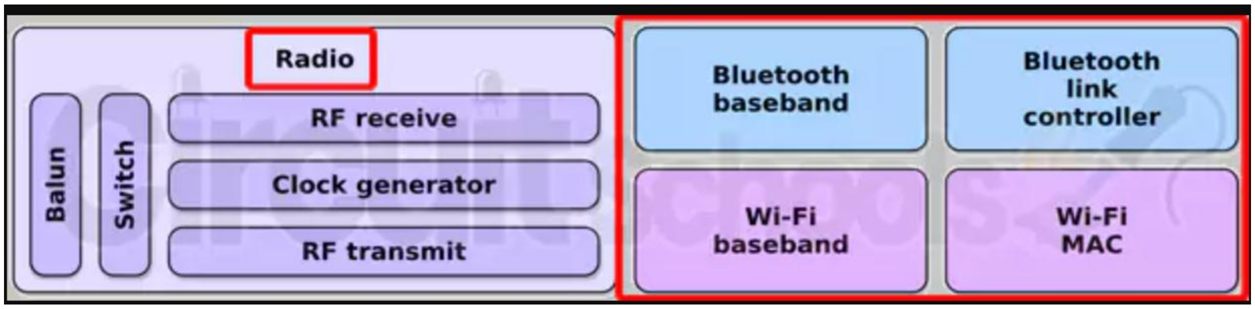
1. **Agriculture**: Farmers use weather monitoring systems to track temperature, humidity, and rainfall, ensuring proper irrigation, preventing plant diseases, and improving crop yield. IoT-based systems help save water and reduce farming costs (Rika Sensor, 2024).
2. **Smart cities**: These systems monitor pollution, air quality, and predict extreme weather, aiding disaster management, urban planning, and public health (SOLARMAN, 2025).
3. **Transportation**: Weather data helps shipping companies, airlines, and road transport navigate safely, avoiding storms and delays (Uniconverge Tech, n.d.).
4. **Global policies**: Countries use environmental data to create effective environmental policies (Uniconverge Tech, n.d.).

2.7 Power Management:

The transmission of data to devices such as indoor display, server and alarms are achieved via the power system. A rechargeable lithium-ion battery with a duration of 72 hours is advantageous in the event of a power outage, as it guarantees uninterrupted data reception. Solar panel can be used to recharge the batteries during the day and also make the system totally off grid. During night time hours, battery power is utilized for the purpose of energizing circuits and sensors. To reduce power consumption, we can use different strategies, also the ESP32 is designed for low power consumption, with features like deep sleep mode, which reduces power usage to less than 5 µA (Circuit Schools, n.d.), Sensors and communication modules are activated only when needed, minimizing energy draw. Can adjusts processor clock rates based on workload (Episensor, n.d.).

2.8 Communication Protocol:

The ESP32's built-in Wi-Fi module enables high-speed data transfer to the Blynk server, being compatible with 802.11 b / g / n in the 2.4GHz band, reaching speeds of up to 150 Mbits/s. It also includes Bluetooth communication compatible with Bluetooth v4.2 and Bluetooth Low Energy (BLE) (Circuit Schools (n.d.).



ESP 32 internals showing its radio block (Circuit Schools (n.d.)

2.9 Data Analysis:

1. Data Collection: Sensors gather raw data, which is transmitted to the ESP32 for preprocessing (Appinventiv, 2024).

2. Data Processing: The ESP32 can perform edge computing to preprocess data locally, reducing latency (Appinventiv, 2024). Processed data is sent to the Blynk server, where advanced analytics can be performed

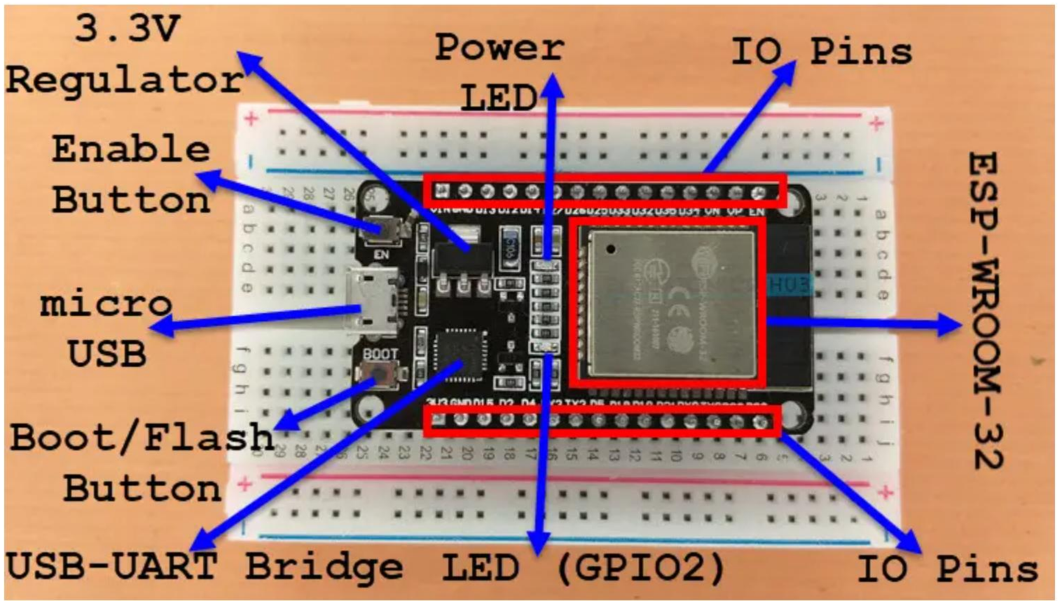
3. Utilization: The processed data is used for real-time decision-making (e.g., triggering alerts for extreme weather) or predictive analytics (e.g., forecasting rainfall patterns). Machine learning algorithms enhance the accuracy of predictions.

3. In-depth analysis of embedded systems in smart weather monitoring device:

3.1 Microcontroller: The smart weather monitoring device using ESP32 combines both hardware and software in an embedded system to collect, process, and transmit environmental data. The ESP32 enables real-time data collection, energy-efficient performance, and smooth cloud integration. It continuously monitors weather factors like temperature, humidity, and pressure, sending the data to a cloud platform for analysis and remote access. The ESP32 is a cost-effective, low-energy microcontroller with built-in Wi-Fi and Bluetooth, offering various processing options. Designed by Espressif Systems, it is an upgrade from the earlier ESP8266 and is widely used in development kits with multiple connectivity options.

3.1.1 Layout:

We will see what a typical ESP32 Development Board consists of by taking a look at the layout of one of the popular low-cost ESP Boards available in the market called the ESP32 DevKit Board.



Layout of ESP 32 DevKit board

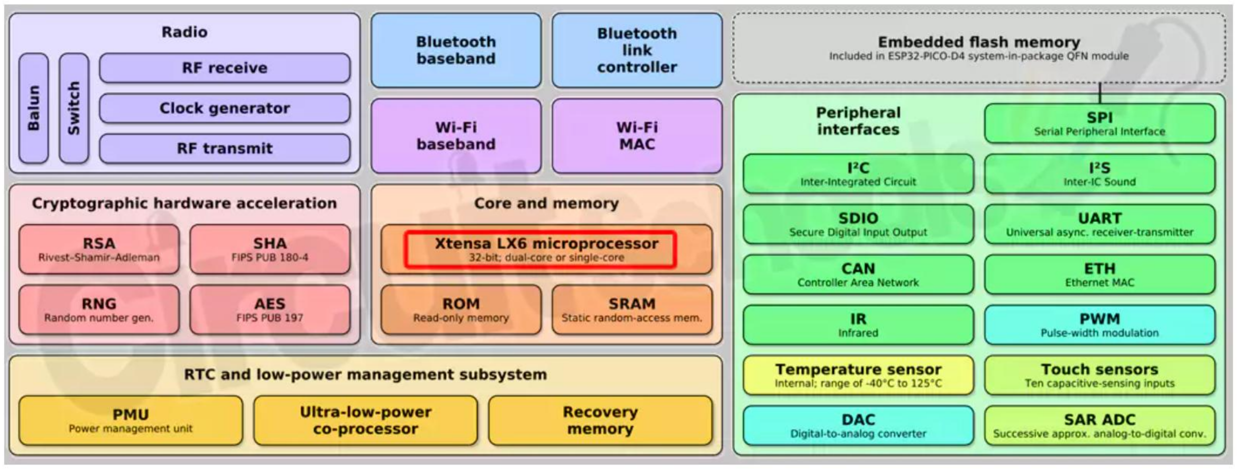
As you can see from the figure above, the ESP32 Board consists of the following:

- ESP-WROOM-32 Module
- Two rows of IO Pins (with 15 pins on each side)
- CP2012 USB – UART Bridge IC
- micro–USB Connector (for power and programming)
- AMS1117 3.3V Regulator IC
- Enable Button (for Reset)
- Boot Button (for flashing)
- Power LED (Red)
- User LED (Blue – connected to GPIO2)
- Some passive components

An interesting point about the USB-to-UART IC is that its DTR and RTS pins are used to automatically set the ESP32 in to programming mode (whenever required) and also rest the board after programming.

3.1.2 ESP32 Architectural Block diagram:

Below is the Architectural block diagram of ESP32 which shows all the functional blocks of ESP32 SOC (Episensor, n.d.).


Radio
Balun
Switch
RF receive
Clock generator
RF transmit
Bluetooth baseband
Bluetooth link controller
Embedded flash memory
Included in ESP32-PICO-D4 system-in-package QFN module
Wi-Fi baseband
Wi-Fi MAC
Peripheral interfaces
SPI
Serial Peripheral Interface
I²C
Inter-Integrated Circuit
I²S
Inter-IC Sound
Cryptographic hardware acceleration
RSA
Rivest-Shamir-Adleman
SHA
FIPS PUB 180-4
RNG
Random number gen.
AES
FIPS PUB 197
Core and memory
Xtensa LX6 microprocessor
32-bit; dual-core or single-core
ROM
Read-only memory
SRAM
Static random-access mem.
SDIO
Secure Digital Input Output
UART
Universal async. receiver-transmitter
CAN
Controller Area Network
ETH
Ethernet MAC
IR
Infrared
PWM
Pulse-width modulation
RTC and low-power management subsystem
PMU
Power management unit
Ultra-low-power co-processor
Recovery memory
Temperature sensor
Internal; range of -40°C to 125°C
Touch sensors
Ten capacitive-sensing inputs
DAC
Digital-to-analog converter
SAR ADC
Successive approx. analog-to-digital conv.

Architectural block diagram of ESP32 (Episensor, n.d.).

3.1.2.1 Core and Memory:

1. **Xtensa LX6 Microprocessor:** The ESP 32 is powered by a dual or single-core Xtensa LX6 processor, operating at a clock of up to 240 MHZ. This makes it ideal for IoT applications. It can execute instructions 15 times faster than normal Arduino uno. As you can observe from the above core block image, it has an ultra-low-power co-processor that is used to perform analog-digital conversions and other operations while the device is operating in deep sleep low-power mode. In this way, a very low consumption by the SoC is achieved (Episensor, n.d.).

Memory:

- **ROM memory (448 KiB):** this memory is write-only, that is, you cannot reprogram it. This is where the codes that handle the Bluetooth stack, the Wi-Fi physical layer control, some general-purpose routines, and the bootloader to start the code from external memory are stored (Episensor, n.d.).
- **RTC SRAM (16 KiB):** this memory is used by the co-processor when the device operates in deep sleep mode.

- **Efuse (1 Kilobit)**: 256 bits of this memory are used by the system itself and the remaining 768 bits are reserved for other applications.
- **Flash embedded (Embedded flash):** This memory is where our application code is stored. The amount of memory varies depending on the chip used:
    - 0 MB (chips ESP32-D0WDQ6, ESP32-D0WD, ESP32-S0WD)
    - 2 MB (chip ESP32-D2WD)
    - 4 MB (Chip ESP32-PICO-D4)

For ESP32s that do not have embedded memory or simply when memory is insufficient for your application, it is possible to add more memory externally:

- Up to 16 MB of external flash memory can be added. This way you can develop more complex applications.
- It also supports up to 8 MB of external SRAM memory.

3.2 Pins: Brief description of the ESP32 30-pin Dev Board pin usage (Episensor, n.d.):

- **Power Pins**: VIN, 3V3, and GND supply and regulate power.
- **GPIO Pins (0–39)**: General-purpose digital I/O, many support ADC, DAC, PWM, and touch.
- **ADC Pins**: GPIO32–GPIO39, GPIO0, GPIO2, etc., read analog signals.
- **DAC Pins**: GPIO25, GPIO26 output analog signals.
- **Touch Pins**: GPIO0, GPIO2, GPIO4, etc., used for capacitive touch sensing.
- **UART**: GPIO1, GPIO3, GPIO16, GPIO17 used for serial communication.
- **SPI**: GPIO12–GPIO14, GPIO18–GPIO23 for SPI devices.
- **I2C**: GPIO21 (SDA), GPIO22 (SCL) for sensor communication.
- **RTC Pins**: Used for low-power real-time clock functions.

3.3 Different ways to program

The ESP32 is user-friendly because it supports multiple programming environments, including:

- Arduino IDE
- PlatformIO IDE (VS Code)

- LUA
- MicroPython
- Espressif IDF
- JavaScript

For our upcoming projects, we'll use the Arduino IDE, but feel free to explore other options too.

3.4 Software Architecture

3.4.1 Firmware Implementation

The firmware for ESP32-based weather monitoring systems is usually developed using the Arduino IDE due to its simplicity and library support. It follows a modular structure with sections for sensor reading, data processing, network communication, and power management. Sensors like the BMP280 (I2C) and DHT (digital input) are initialized during setup, while the main loop handles periodic data collection and cloud updates (Satria et al., 2024). Error handling manages issues like sensor failures or Wi-Fi drops. To save power, especially in solar-powered setups, deep sleep mode is often used during idle times to extend battery life (Satria et al., 2024).

3.4.2 Communication Protocols and Data Management

ESP32-based weather monitoring systems use multiple communication protocols to ensure smooth data flow. Locally, I2C and SPI connect sensors and displays to the ESP32 (mpythonboard, 2024). For wireless communication between devices, ESP-NOW offers a fast, low-power option ideal for sensor networks (UCI IEEE, 2025). Data is usually sent to cloud platforms using HTTP or MQTT, formatted in JSON for easy integration. The system collects, processes, and transmits data, often applying calibration, averaging, and unit conversion. Some setups use edge computing to analyze data locally before sending it, which reduces bandwidth use and enables quicker responses to changing weather conditions (Intel DevMesh, n.d.).

3.4.3 Blynk IoT Platform Integration

The Blynk IoT platform allows easy remote monitoring and control of ESP32-based weather stations (Blynk, n.d.). After installing the Blynk library and setting up authentication tokens, the ESP32 sends data to the Blynk server using virtual pins for each parameter like temperature and humidity. Users can create custom dashboards with widgets like graphs and gauges. Blynk also supports alerts for threshold breaches, stores historical data for analysis, and provides API access for integration with other platforms (Blynk, n.d.).

4. Challenges and future improvements of esp32-based weather monitoring device:

4.1 Challenges:

While the ESP32-based weather monitoring system is a powerful and flexible solution, it does come with a few practical challenges. One of the main issues is power consumption, especially when the device is placed in remote areas without easy access to electricity. Even though the ESP32 has low-power modes, continuous, sensing and data transmission can quickly drain its power source. Another challenge is sensor accuracy—over time, sensors used to measure temperature, humidity, or pressure may need calibration to ensure reliable readings. Connectivity is also a concern, particularly in rural or isolated locations where Wi-Fi signals are weak or unavailable. Additionally, the hardware durability of the system can be affected by harsh weather conditions like rain, dust, or extreme temperatures. Lastly, there are security concerns, as transmitting data over open networks can expose the system to potential breaches if not properly protected.

4.2 Future Improvements:

To make the system more efficient and reliable, several upgrades can be considered. Using solar panels combined with a good power management strategy can significantly extend battery life, making the device more sustainable for long-term outdoor use. Incorporating edge computing would allow data to be processed directly on the device, reducing the need for constant internet access and lowering data usage. In areas with limited Wi-Fi, using LoRa or NB-IoT for communication could offer better coverage and reliability. To protect the hardware, placing the device in a weatherproof (IP65-rated) enclosure would help it withstand tough environmental conditions. Finally, implementing AI or machine learning algorithms for predictive analysis and strong data encryption for secure transmission can greatly enhance the device’s functionality and safety.

5. Conclusion:

This case study has highlighted the practical and efficient use of the ESP32 microcontroller in building a smart weather monitoring system. By integrating sensors such as the DHT21 and BMP280, the system is capable of accurately measuring vital weather parameters like temperature, humidity, atmospheric pressure, and altitude. The ESP32's built-in Wi-Fi makes it easy to transmit this data to cloud platforms like Blynk, allowing users to monitor real-time weather updates directly from their smartphones or computers—anytime, anywhere.

What makes this system particularly valuable is its wide range of real-world applications. From supporting smart agriculture and improving city infrastructure to aiding disaster management and transportation planning, the system delivers localized, real-time insights that can help make smarter, data-driven decisions. For instance, farmers can use the system to plan irrigation more efficiently, while transport authorities can use it to adjust routes based on weather conditions.

Moreover, by incorporating renewable energy sources like solar panels and rechargeable batteries, the system becomes not only energy-efficient but also sustainable in off-grid or power-sensitive areas. This improves its reliability and long-term usability, even during outages or in remote locations.

Overall, the ESP32-based weather monitoring system offers a powerful, affordable, and scalable solution for environmental data tracking. Its modular nature allows for future enhancements—whether that means adding new types of sensors, improving data analytics through machine learning, or expanding communication protocols—making it a smart investment for a wide variety of industries and research fields.

6. References:

1.Kumar, A.J.S., 2017. *Air Quality Monitoring System Based on IoT using Raspberry Pi*. In: International Conference on Computing, Communication and Automation, Haryana.

2. Weather Monitoring with ESP32 (with static tokens). No date.

3. Sdiopr, 2021. *Development of an ESP-32 Microcontroller Based Weather Reporting Device*. Available at: https://sdiopr.s3.ap-south-1.amazonaws.com/doc/Revised-ms_JERR_88570_v3.pdf [Accessed 9 April 2025].

4. Electronics Hub, 2022. *Electronics Hub*. [online] Available at: https://www.electronicshub.org [Accessed 9 Apr. 2025].

5. Blynk, 2025. *Blueprint Template for Weather Monitoring Device*. Available at: https://blynk.cloud/dashboard/49755/blueprints/Library/TMPL4j57WtmD7 [Accessed 9 April 2025].

6. Rika Sensor (2024) *How Weather Sensors Enhance Smart City Initiatives?* Available at: https://www.rikasensor.com/a-news-how-weather-sensors-enhance-smart-city-initiatives.html

7. SOLARMAN (2025) *Applications of Weather Sensors*. Available at: https://www.solarmanpv.com/applications-of-weather-sensors/

8. Uniconverge Tech (n.d.) *IoT Based Smart Agriculture Monitoring System*. Available at: https://www.uniconvergetech.in/blog/iot-based-smart-agriculture-monitoring-system

9. Circuit Schools (n.d.) *What is ESP32, how it works and what you can do with ESP32*. Available at: https://www.circuitschools.com/what-is-esp32-how-it-works-and-what-you-can-do-with-esp32/

10. Episensor (n.d.) *Optimising power consumption in IoT devices*. Available at: https://episensor.com/knowledge-base/optimising-power-consumption-in-iot-devices/

11. Beni Satria et al. (2024) An Implementation IoT Weather Station Based On ESP32. Jurnal Scientia. Available at: https://infor.seaninstitute.org/index.php/pendidikan/article/download/2629/2357/

12. mpythonboard (2024) *ESP32 Communication: A Complete Overview*. Available at: https://www.mpythonboard.com/blogs/blogs/esp32-communication-a-complete-overview

13. UCI IEEE (2025) *Project 6: Weather Station*. Available at: https://ieee.ics.uci.edu/ops/project_6.html

14. Intel DevMesh (n.d.) *IoT based Weather Data System using ESP32*. Available at: https://devmesh.intel.com/projects/iot-based-weather-data-system-using-esp32

15. Blynk (n.d.) *Weather Monitoring on ESP32 (WiFi provisioning & OTA)*. Available at: https://blynk.io/blueprints/weather-monitoring-with-esp32-with-wifi-provisioning-ota

16. Beni Satria et al. (2024) *An Implementation IoT Weather Station Based On ESP32*. Jurnal Scientia. Available at: https://infor.seaninstitute.org/index.php/pendidikan/article/download/2629/2357/

17. Appinventiv (2024) 'IoT Data Analytics: Types, Use Cases, and Implementation', Appinventiv. Available at:
Appinventi