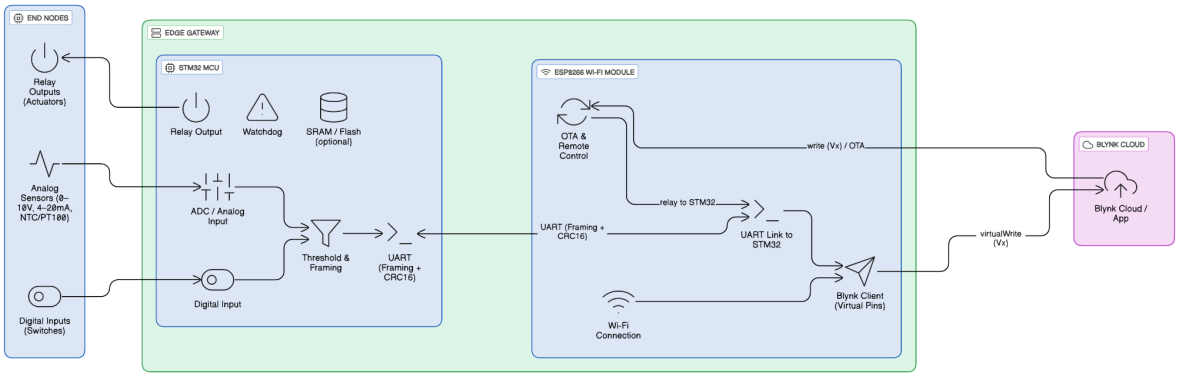
HomeX



1. Project Overview

This project focuses on developing a **Smart Home System** based on the **STM32 microcontroller** and the **ESP8266 Wi-Fi module**.
The goal is to design an IoT architecture that can **monitor environmental conditions** and **control home appliances** remotely through **Blynk Cloud** using a smartphone application.

Each **End Node** consists of various environmental sensors such as temperature, humidity, gas, and light sensors. The **Gateway** combines the STM32 MCU and the ESP8266 module.
STM32 handles **sensor acquisition, local processing, and actuator control**, while ESP8266 manages **Wi-Fi connectivity**, **Blynk communication**, and **user interaction** through the cloud.

The system enables **real-time monitoring and control** of smart home devices via Blynk App widgets (gauges, buttons, charts).
It also includes **watchdog protection** for reliability and can be extended with **Over-the-Air (OTA)** updates for firmware maintenance.

> The primary objective is to build a compact yet complete IoT system that demonstrates the full data flow chain:
> **Sensing → Local Processing → Communication → Cloud Visualization & Control.**

2. Basic Features

- **Sensor Data Acquisition**
  STM32 collects and processes data from various sensors:
  - Temperature & Humidity (e.g., DHT22, SHT30)
  - Light Intensity (LDR, BH1750)
  - Gas/Smoke Detection (MQ2, MQ135)
  - Motion Detection (PIR HC-SR501)

    Processed values are framed and sent to ESP8266 via UART.

- **UART Communication Protocol**
  STM32 and ESP8266 exchange structured frames with CRC16 for error checking.

- **Wi-Fi and Cloud Connectivity**
  ESP8266 connects to a Wi-Fi network and synchronizes data with **Blynk Cloud**.
  Sensor readings are published using **Blynk Virtual Pins (V0–Vx)**, displayed instantly in the app.

- **Remote Control Functionality**
  Users can remotely turn ON/OFF lights, fans, or pumps via Blynk App buttons.
  Commands received on Virtual Pins (e.g., V10, V11) are parsed by ESP8266 and relayed to STM32 through UART for relay activation.

- **Monitoring and Alerts**
  If measured values exceed defined thresholds (e.g., temperature > 50 °C, gas detected), STM32 generates an alarm message → ESP8266 → Blynk App notification.

- **System Reliability**
  Both STM32 and ESP8266 implement **Watchdog Timers** to ensure system recovery after faults or unexpected hangs.

- **Extensibility**
  The design allows adding more sensors, actuators, or upgrading to a local MQTT broker (e.g., EMQX, Mosquitto) instead of Blynk if desired.

3. Development Workflow

- **Version Control and Collaboration**
  All source code and documentation are maintained on **GitHub**.
- **Issue Tracking and Branching**
  Each new feature, bug fix, or task is tracked as a **GitHub Issue**.
  Branch naming follows the format:

```
<developer>/<issue_number>-<short_description>
e.g., vinh/3-add-uart-frame-parser
```

- **Pull Requests and Code Review**
  Each completed feature is merged through a Pull Request (PR).
  Team members review PRs before merging to the develop or main branch.

- **Daily / Weekly Sync Meetings**
  Short progress meetings (daily or bi-weekly) ensure alignment on current issues, integration status, and upcoming milestones.

4. Milestone 1: High-Level Design

- Design the **UART protocol** between STM32 and ESP8266 (data frame format, CRC verification, command set).
- Design the **data flow** between STM32 → ESP8266 → Blynk Cloud, including publish/subscribe mapping with Virtual Pins.
- Define **threshold logic** and event triggers for each sensor.
- Plan hardware interfaces (ADC, GPIO, relay control) and power requirements.
- Draft the **Blynk dashboard layout** (sensors, switches, charts, alarms).
- Verify Wi-Fi connectivity and Blynk token configuration process.

Deliverables:

- System block diagram
- UART frame structure document
- Blynk dashboard prototype
- Initial GitHub repository setup

5. Milestone 2: Core Functionality Implementation

- STM32:
    - Implement ADC and digital input reading for all sensors.
    - Apply threshold detection and framing logic.
    - Handle UART TX/RX tasks with CRC validation.
    - Implement relay control and safety timeout logic.
    - Add watchdog configuration.
- ESP8266:
    - Implement Wi-Fi connection routines (auto-reconnect).
    - Integrate Blynk library for Virtual Pin operations.

    - Develop UART parser to receive sensor frames and forward to Blynk.
    - Implement downlink command handling from Blynk to STM32 (e.g., relay control, threshold updates).
    - Add basic OTA update and reset handling.

Deliverables:

- Working STM32 firmware (sensor acquisition + UART TX)
- Working ESP8266 firmware (Wi-Fi + Blynk integration)
- Successful data transfer from STM32 → ESP8266 → Blynk dashboard
- Relay control from app confirmed

6. Milestone 3: System Integration and Testing

- Integrate STM32 and ESP8266 firmware with synchronized data exchange.
- Validate two-way communication (sensor data uplink, control command downlink).
- Ensure smooth Wi-Fi reconnection and Blynk synchronization after reset.
- Test sensor threshold triggers and real-time updates on dashboard.
- Conduct system stress test for continuous operation (watchdog verification).
- Validate end-to-end operation: **Sensor → STM32 → ESP8266 → Blynk Cloud → User → Relay.**
- Perform integration bug fixing and finalize documentation.

Deliverables:

- Complete Smart Home prototype
- Demonstration video and documentation
- GitHub repository with tagged release version
- Presentation slides summarizing system flow and test results