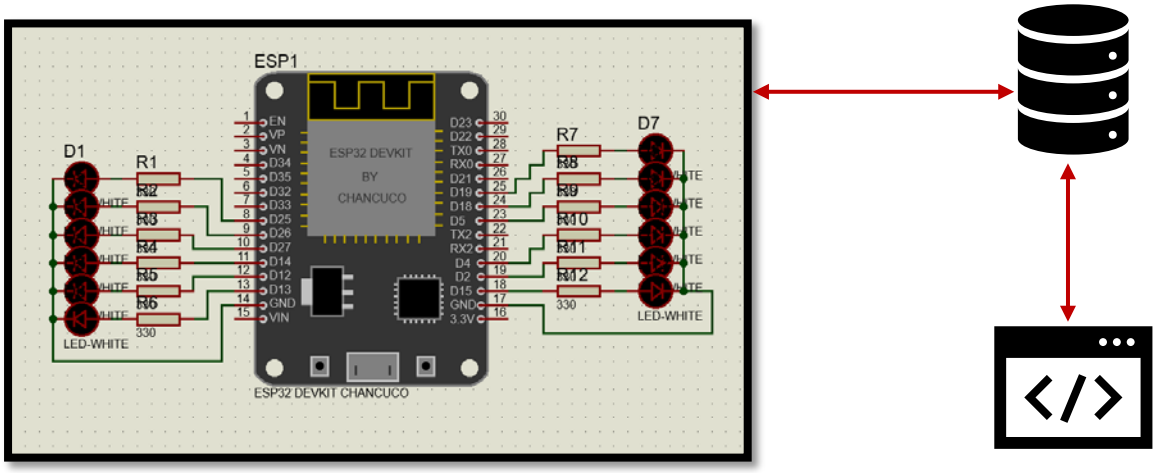
บทนำและโครงสร้างระบบ (Introduction & System Architecture)

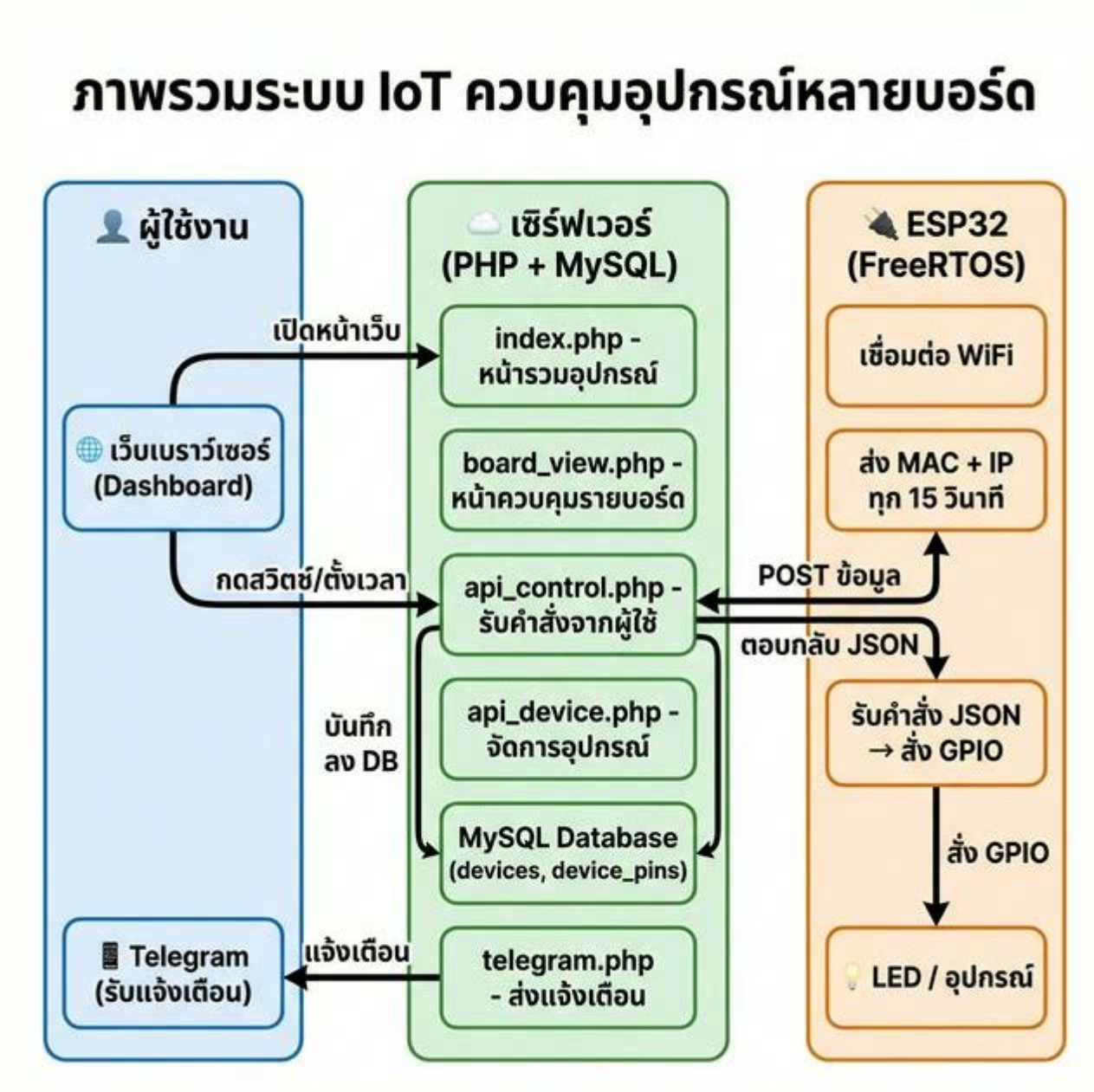
- **ชื่อโครงงาน:** ระบบควบคุมและสั่งการ IoT หลายบอร์ดแบบรวมศูนย์
- **ที่มาและความสำคัญ:** ปัจจุบันการใช้งาน IoT มักพบปัญหาความกระจัดกระจายของอุปกรณ์ โครงงานนี้จึงพัฒนา Platform กลางเพื่อจัดการบอร์ด ESP32 หลายตัวพร้อมกันผ่านหน้าเว็บเดียว (Centralized Dashboard) รองรับการขยายตัวของระบบในอนาคต
- **วัตถุประสงค์:**
    1. เพื่อสร้างระบบควบคุมส่วนกลางที่เข้าถึงได้ทุกที่ผ่านอินเทอร์เน็ต
    2. เพื่อพัฒนาระบบสั่งการอัตโนมัติ (Timer/Duration) ที่ยืดหยุ่น
    3. เพื่อตรวจสอบสถานะ Online/Offline และบันทึกประวัติการทำงานแบบ Real-time
    4. เพื่อเพิ่มความสะดวกในการใช้งานด้วยการปรับแต่ง UI/UX และระบบแจ้งเตือนผ่าน Telegram
- **เทคโนโลยีที่ใช้:**
    - **Hardware:** ESP32, LED Actuators, WiFi Connectivity
    - **Software:** C++ (Arduino/FreeRTOS), PHP (Backend), MySQL (Database), Bootstrap 5 (UI)
    - **Protocol:** RESTful API (JSON)

วิธีดำเนินงานและคู่มือการใช้งาน (Methodology & User Manual)

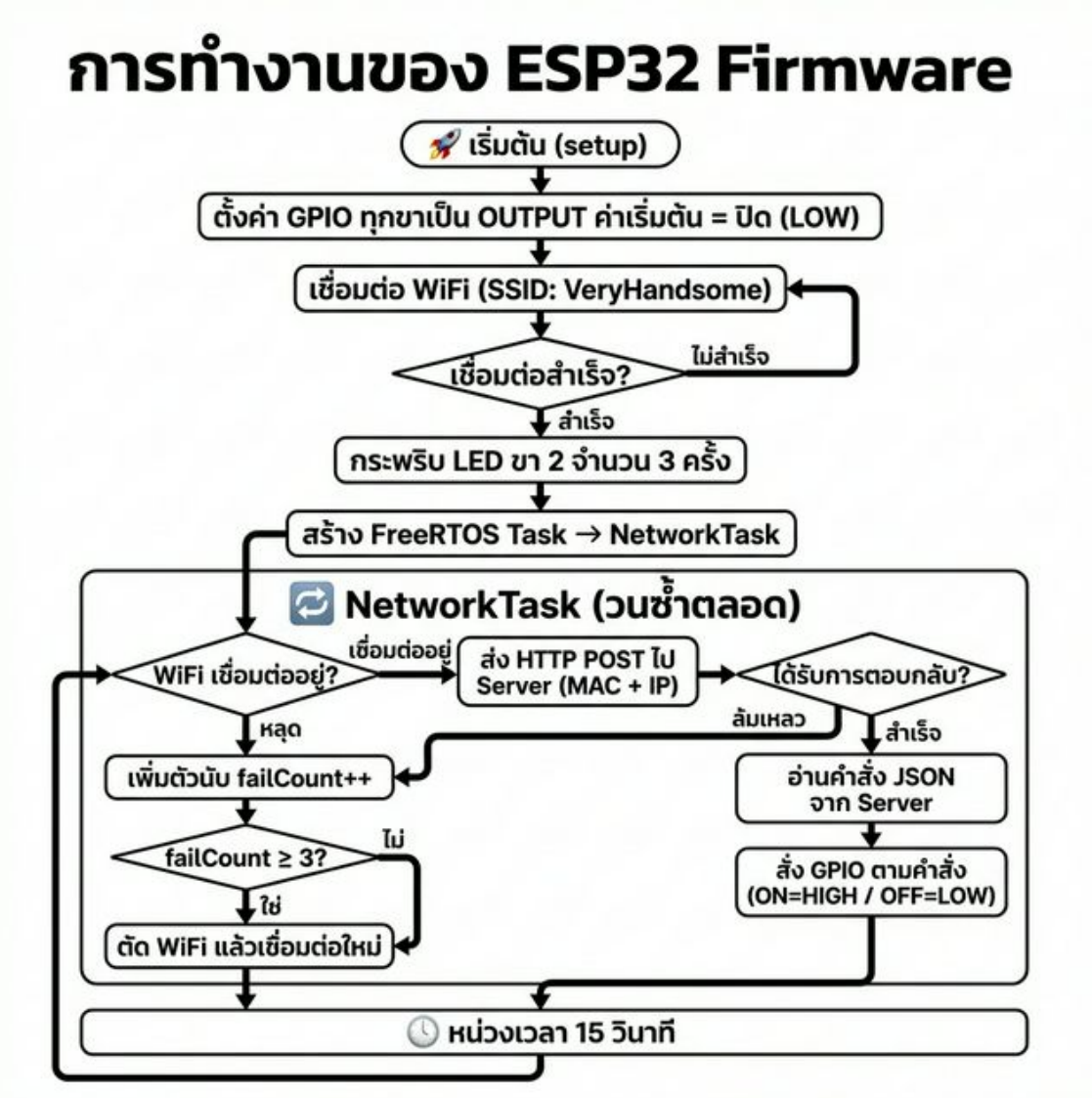


แนวคิดและการออกแบบเบื้องต้น

- **การทำงานของระบบ (System Logic):**
    - **ESP32:** ใช้ FreeRTOS แยกงาน Network เพื่อความเสถียร ส่ง MAC Address เพื่อระบุตัวตนอัตโนมัติ และรับคำสั่ง JSON ทุก 15 วินาที
    - **Server:** ประมวลผลตรรกะ (Logic) ทั้งหมด เช่น การเช็คช่วงเวลา (Timer) หรือตัวนับเวลาถอยหลัง (Duration) แล้วส่งสถานะกลับให้บอร์ด



ภาพรวมระบบ IoT ควบคุมอุปกรณ์หลายบอร์ด



การทำงานของ ESP32 Firmware

- **รายการอุปกรณ์ (Equipment List):**

    1. **ESP32 Microcontroller:** หน่วยประมวลผลหลักพร้อม WiFi ในตัว
    2. **LED/Relay Module:** อุปกรณ์แสดงผลหรือตัดต่อวงจรไฟฟ้า
    3. **Power Supply (5V):** แหล่งจ่ายไฟสำหรับบอร์ดและเซ็นเซอร์
    4. **Web Server & Database:** พื้นที่จัดเก็บข้อมูลและรันระบบ Backend

- **คู่มือการใช้งาน:**

    1. **Setup:** จ่ายไฟให้บอร์ดและเชื่อมต่อ WiFi
    2. **Dashboard:** เข้าผ่าน URL เพื่อดูรายการบอร์ด (เขียว=Online, เทา=Offline)
    3. **Control:** เลือกโหมด Manual (เปิด/ปิด), Timer (ตั้งช่วงเวลา), หรือ Duration (ตั้งเวลาดับเอง)
    4. **Customize:** ปรับแต่งธีมสีและขนาดตัวอักษรได้ที่เมนูการตั้งค่า (Settings)

1. แนวคิด IoT (Internet of Things)

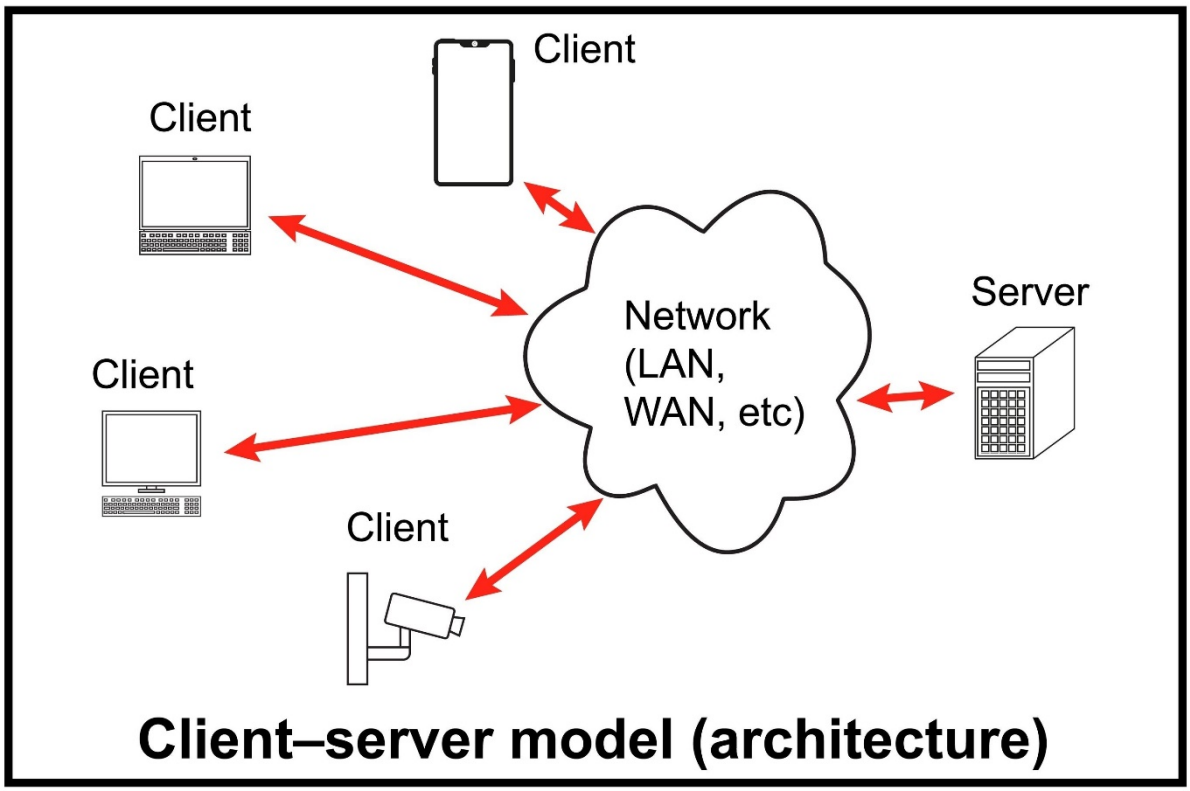
IoT คือเครือข่ายของอุปกรณ์ทางกายภาพที่มีการฝังเซนเซอร์ ซอฟต์แวร์ และเทคโนโลยีการเชื่อมต่อ เพื่อให้สามารถ **แลกเปลี่ยนข้อมูล** กับอุปกรณ์อื่นหรือระบบบนอินเทอร์เน็ตได้โดยไม่ต้องผ่านมนุษย์

- **กระบวนการทำงาน:** Sensor (รับค่า) ⟶ Gateway (ส่งข้อมูล) ⟶ Cloud (ประมวลผล) ⟶ User UI (แสดงผล/สั่งการ)

2. สถาปัตยกรรม Client-Server

เป็นโมเดลการสื่อสารที่แบ่งหน้าที่ชัดเจนระหว่างผู้ขอและผู้ให้บริการ:

- **Client:** อุปกรณ์ปลายทาง (เช่น มือถือ, ESP32) ที่ส่งคำขอ (Request) ไปยังเครื่องแม่ข่าย
- **Server:** เครื่องคอมพิวเตอร์ที่รอรับคำขอ ประมวลผล และส่งผลลัพธ์ (Response) กลับไป



3. RESTful API และ JSON

- **RESTful API:** เปรียบเสมือน "พนักงานเสิร์ฟ" ที่รับออเดอร์จาก Client ไปบอก Server โดยใช้มาตรฐาน HTTP (GET, POST, PUT, DELETE)
- **JSON (JavaScript Object Notation):** คือรูปแบบภาษาที่ใช้ส่งข้อมูล มีลักษณะเป็นคู่ Key: Value ซึ่งอ่านง่ายทั้งคนและคอมพิวเตอร์ เช่น {"temperature": 25, "status": "on"}

4. FreeRTOS และ Multitasking

โดยปกติ Microcontroller จะทำงานทีละคำสั่ง (Single Task) แต่ **FreeRTOS** คือระบบปฏิบัติการขนาดเล็กที่ทำให้ ESP32 สามารถทำ **Multitasking** หรือ "การทำงานหลายอย่างพร้อมกัน" ได้โดยการสลับงานไปมาอย่างรวดเร็ว (Scheduling) หรือแบ่งงานให้แต่ละ Core ของ CPU

5. งานวิจัย Smart Home

งานวิจัยในด้านนี้มุ่งเน้นไปที่การเพิ่มความสะดวกสบายและความปลอดภัย โดยอาศัยระบบอัตโนมัติ (Automation) เช่น:

- การใช้ Machine Learning ทำนายพฤติกรรมการใช้ไฟเพื่อประหยัดพลังงาน

- ระบบตรวจจับการล้มของผู้สูงอายุด้วย AI และเซนเซอร์

6. งานวิจัยระบบควบคุมแบบรวมศูนย์ (Centralized Control System)

เป็นการวิจัยเกี่ยวกับการจัดการอุปกรณ์จำนวนมากจากจุดเดียว (Central Hub) เพื่อลดความซับซ้อนในการเชื่อมต่อ ข้อดีคือจัดการง่าย แต่ข้อควรระวังคือถ้า Hub ล่ม ทั้งระบบจะใช้งานไม่ได้ งานวิจัยยุคใหม่จึงมักเสนอระบบ **Distributed** หรือ **Edge Computing** เข้ามาเสริม

7. ESP32

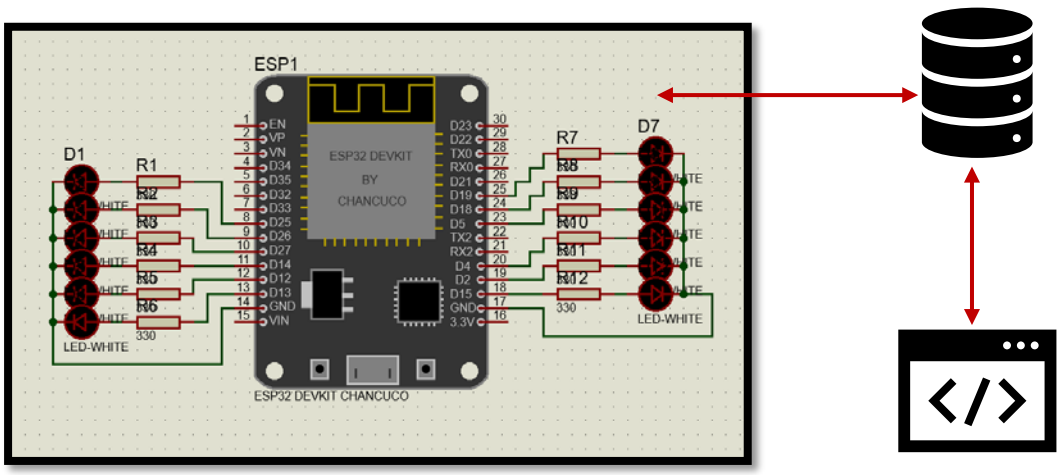
Microcontroller ยอดนิยมสำหรับงาน IoT เพราะมี **Wi-Fi และ Bluetooth ในตัว** ราคาถูก และประสิทธิภาพสูง (Dual-core) รองรับการเขียนโปรแกรมด้วย Arduino IDE, MicroPython หรือ ESP-IDF

8. RELAY (รีเลย์)

คือสวิตช์อิเล็กทรอนิกส์ที่ใช้กระแสไฟต่ำจาก ESP32 ไป **ควบคุมการเปิด-ปิดเครื่องใช้ไฟฟ้าที่ใช้ไฟสูง** (เช่น หลอดไฟ 220V) โดยใช้หลักการของแม่เหล็กไฟฟ้าในการดึงหน้าสัมผัสสวิตช์ให้เชื่อมต่อกัน

Project Overview & Objectives

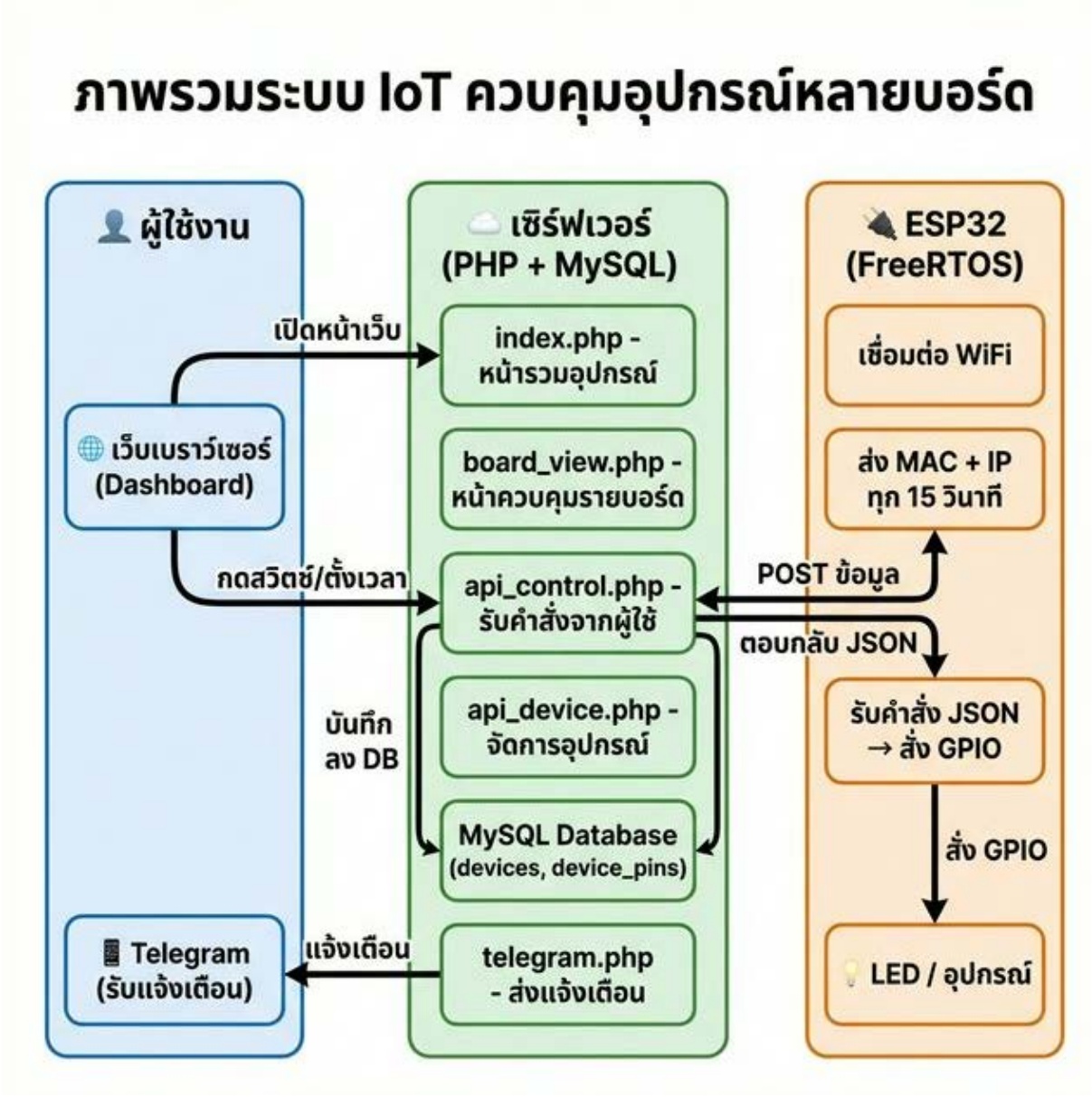
- **Project Title:** Universal Multi-Board IoT Control & Automation System
- **Abstract:** This project proposes a centralized web-based platform designed to manage and monitor multiple ESP32 boards simultaneously. It eliminates the need for hardcoding IDs by using MAC Address identification, providing a scalable and user-friendly solution for smart automation.
- **Objectives:**
    1. **Centralized Management:** Control multiple IoT devices from a single web dashboard.
    2. **Flexible Automation:** Implement multi-mode operations including Manual, Timer-based, and Duration-based triggers.
    3. **Real-Time Monitoring:** Track Online/Offline status and operation logs instantly.
    4. **Enhanced UX:** Allow users to customize the UI theme and receive notifications via Telegram Bot.
- **System Components:**



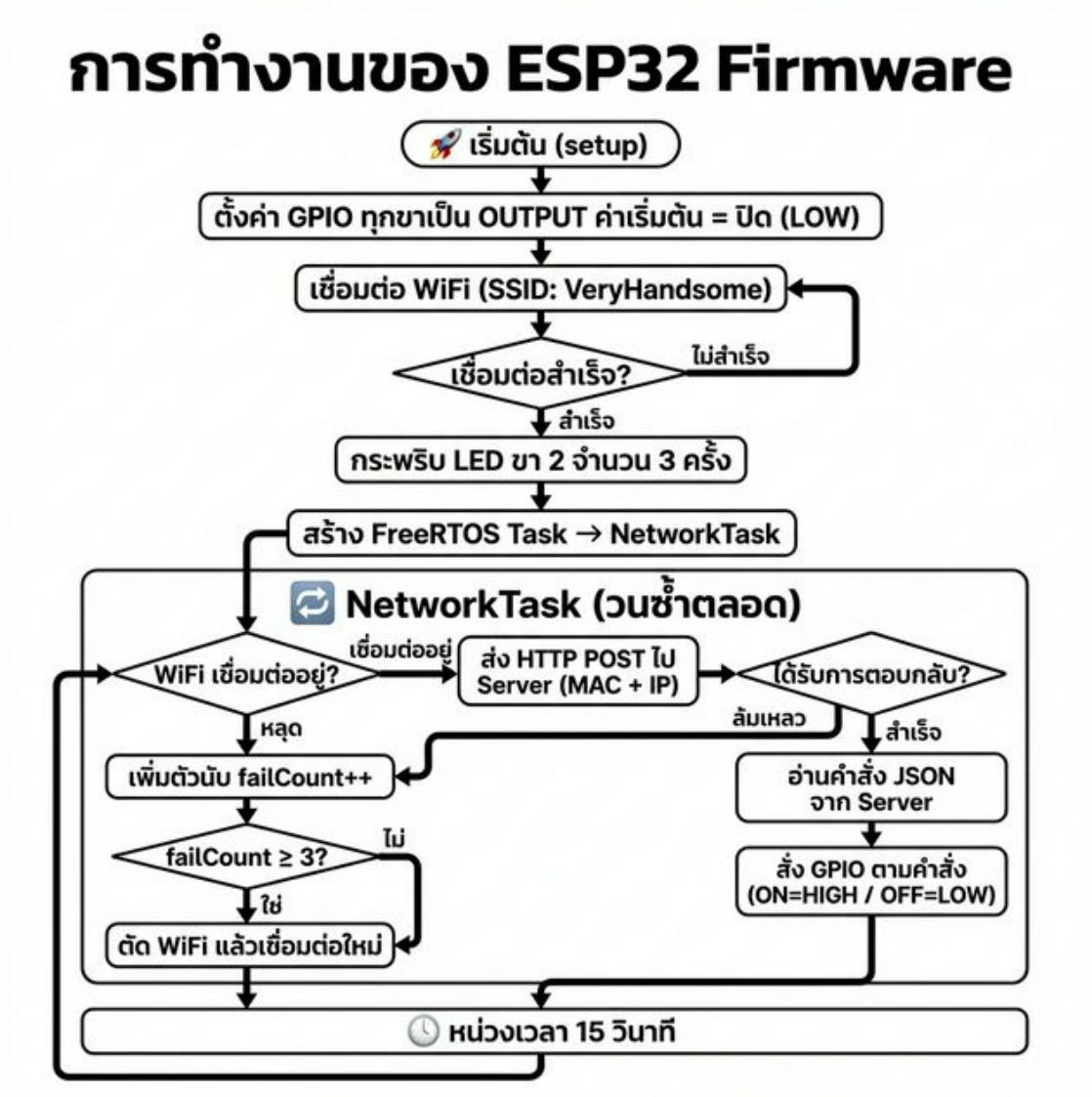
Conceptual and preliminary design.

- **Microcontroller:** ESP32 (Multitasking via FreeRTOS).

- **Backend:** PHP & MySQL for logic processing and data storage.
- **Frontend:** HTML5, CSS3 (Bootstrap), and JavaScript (jQuery).



Overview of IoT systems controlling multiple devices/boards.



How the ESP32 Firmware Works

Page 2: Methodology & Operations

- **Technical Methodology:**
    - The system utilizes a **Polling Mechanism** where the ESP32 requests instructions from the server every 15 seconds via RESTful API.
    - The server acts as the "Brain," calculating time-based logic (Timer/Duration) and updating the SQL database.

    - **Security:** Device authentication is handled automatically through unique MAC Addresses.

- **System Flowchart:**
- **Hardware & Software List:**
    1. **ESP32 Board:** Main controller with built-in WiFi.
    2. **Actuators (LEDs/Relays):** To simulate or control electrical appliances.
    3. **PHP/MariaDB Server:** To host the API and Management Dashboard.
    4. **Telegram API:** For instant alert messaging.
- **User Instructions:**
    1. **Connection:** Power the ESP32; it will auto-connect to the predefined WiFi.
    2. **Monitoring:** Access the dashboard to view all active boards.
    3. **Action:** Use the "Manage Board" page to toggle switches or set automated schedules.
    4. **Personalization:** Click the "Gear" icon to adjust background colors, card styles, and font sizes.

1. IoT Concept (Internet of Things)

The **Internet of Things (Things)** refers to the network of physical objects embedded with sensors, software, and connectivity (Wi-Fi, LoRa, Zigbee) that allows them to collect and exchange data over the internet.

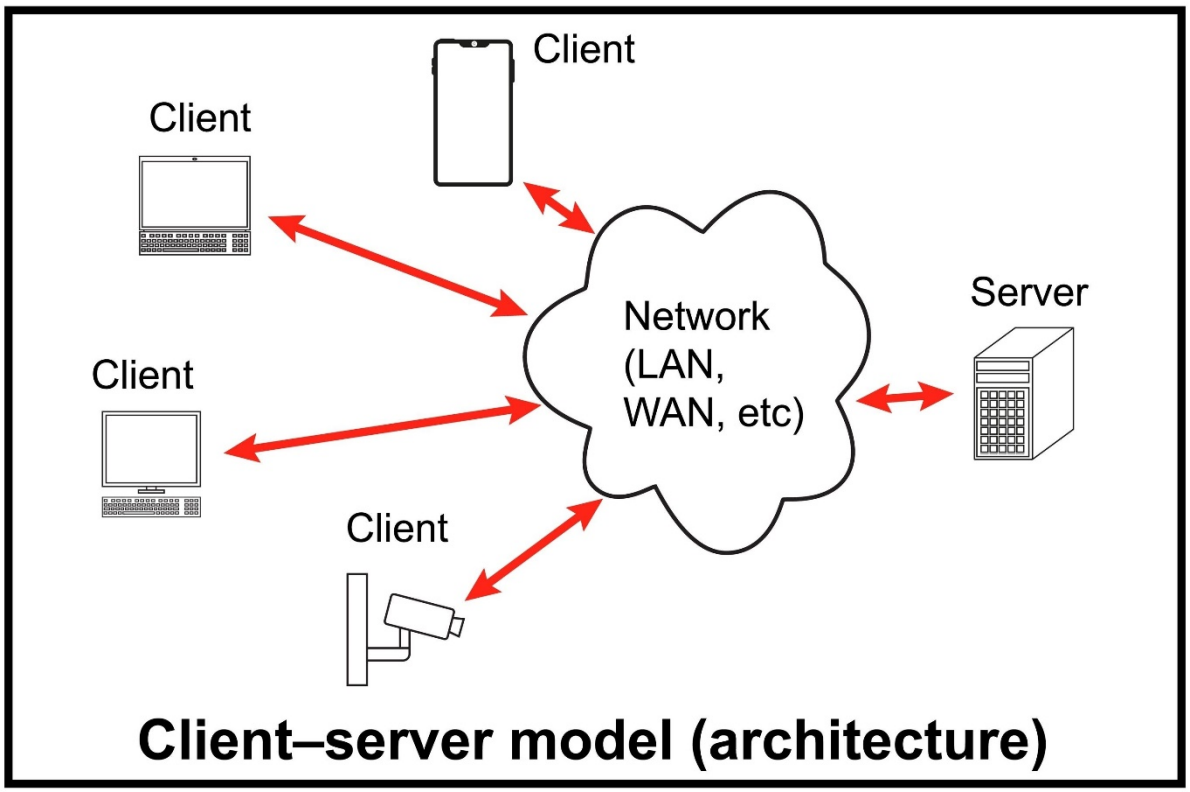
- **Core Flow:** Perception Layer (Sensors) ⟶ Network Layer (Gateway/Internet) ⟶ Application Layer (Cloud/Dashboard).

{ "action": "image_generation", "action_input": "An infographic diagram of the IoT (Internet of Things) ecosystem showing smart devices, a cloud server, and a mobile dashboard connected by digital lines, clean flat design." }

2. Client-Server Architecture

A distributed application structure that partitions tasks between the providers of a resource or service (**Servers**) and service requesters (**Clients**).

- **Client:** Sends a request (e.g., an ESP32 asking for weather data).
- **Server:** Processes the request and sends back a response (e.g., a Database or Web Server).



3. RESTful API and JSON

- **RESTful API:** A set of rules that allow two machines to communicate via HTTP. It uses standard methods like **GET** (read), **POST** (create), **PUT** (update), and **DELETE**.
- **JSON (JavaScript Object Notation):** The "language" used to package the data. It is lightweight and text-based, making it easy for both humans and machines to parse.
    - *Example:* {"device_id": "ESP32_01", "relay_status": true}

4. FreeRTOS and Multitasking

FreeRTOS (Real-Time Operating System) is an open-source kernel designed for microcontrollers.

- **Multitasking:** Instead of running code in a single linear loop, FreeRTOS allows the CPU to manage multiple **Tasks** simultaneously by switching between them based on priority. This is crucial for ESP32 applications that need to handle Wi-Fi, sensors, and motor control at the same time.

5. Smart Home Research

Current research focuses on **Home Automation (HA)** systems that improve energy efficiency, security, and accessibility. Key trends include:

- **Context-Awareness:** Systems that adjust lighting or temperature based on human presence.
- **Energy Management:** Using AI to optimize electricity consumption during peak hours.

6. Centralized Control System Research

This research focuses on a topology where a single **Master Controller** (Hub) manages all peripheral nodes.

- **Pros:** Simplified data aggregation and unified security policies.
- **Cons:** "Single Point of Failure"—if the central hub goes down, the entire network fails. Recent research explores **Hybrid Centralized** models where local processing (Edge Computing) happens before sending data to the hub.

7. ESP32

The **ESP32** is a powerful, low-cost System-on-a-Chip (SoC) with integrated **Wi-Fi and Dual-mode Bluetooth**.

- **Key Specs:** Dual-core processor, rich GPIOs (General Purpose Input/Output), and low power consumption, making it the industry standard for DIY and industrial IoT prototypes.

8. RELAY

A **Relay** is an electromagnetic switch that allows a low-power signal (like the 3.3V from an ESP32) to control a high-power circuit (like a 220V AC lamp). It provides **Galvanic Isolation**, protecting the delicate microcontroller from high-voltage surges.