Smart Socket System with 3 Independent Outlets

P0. Project Concept and Development Scope

The project aims to develop a **smart socket system featuring three independent outlets**, each individually controllable via a networked microcontroller (e.g., ESP32). The primary objective is to automate the power supply of electrical devices by providing remote control functionality through a mobile application or a web interface. This system serves as a practical use case for **decision-making control systems** aimed at monitoring and managing parameters within urban or residential environments, specifically designed to allow users to manage household appliances remotely.

P1. Resource Identification

Knowledge Base: Relay operating principles, switching circuits, mains voltage protection (AC), and microcontroller programming (Arduino/ESP-IDF).

Hardware Requirements:

- ESP32 Microcontroller
- 3-Channel Relay Modules (≥16A, 250VAC, with galvanic isolation)
- Isolated 230V to 5V DC Power Supply
- Three output sockets and connectors
- Insulating enclosure + fuse for protection + terminal blocks
- Mains power cable, connectors, and PCB or breadboard for prototyping

Software Requirements: IDE Arduino or PlatformIO, libraries: WiFi.h, PubSubClient (for MQTT), WebServer.h, source code for relay logic and the command interface.

P2. Development Model and Methodology

Physical Solution: A functional prototype will be developed by integrating the hardware components into a secure, insulated enclosure. The focus is on ensuring high-voltage safety and stable connectivity for the microcontroller.

Logical Solution:

- **Communication Architecture**: The project utilizes the **MQTT (Message Queuing Telemetry Transport)** protocol, which follows a Publisher/Subscriber model. This ensures efficient, low-latency data exchange between the hardware and the control interface.
- **Control Mechanism:** Commands are sent from the web interface to a central broker (HiveMQ), which then routes the instructions to the ESP32 microcontroller.

- **Security Protocol:** To prevent unauthorized access, the system implements a security layer based on the **SHA-256 (Secure Hash Algorithm 256-bit)**. This ensures that the access code is processed securely using a one-way cryptographic hash, protecting the integrity of the control system.

P3. SWOT Analysis

Strengths

- **Safety**: The use of high-quality components and an insulating enclosure provides protection against electrical hazards.
- **Independence**: The system allows for the individual control of three different electrical outlets, offering high flexibility for the user.
- **Security**: Access to the control interface is secured using the SHA-256 cryptographic algorithm, preventing unauthorized use.
- **Real-time Synchronization**: The use of the MQTT protocol ensures nearly instantaneous communication and real-time status updates between the physical device and the web interface.
- **Remote Accessibility**: The interface is hosted in the cloud (Vercel), allowing the user to manage the sockets from any location with an internet connection.

Weaknesses

- **Form Factor & Dimensions:** As a hand-built prototype, the device is larger than industrial products. Transitioning to a custom Printed Circuit Board (PCB) is required for professional miniaturization.
- **Lack of Power Monitoring (V1):** The current version manages ON/OFF states but does not measure real-time current consumption, a feature necessary for precise energy cost optimization.
- **Cloud Connectivity Dependence:** Remote control relies strictly on the availability of the MQTT broker. A local fallback mechanism would be needed to ensure 100% reliability during internet outages.
- **High-Voltage Safety Risks:** Managing 230V AC requires advanced protection knowledge; while the prototype is insulated, it remains a technical challenge compared to low-voltage IoT devices.

Opportunities

- **Advanced Energy Management:** Implementation of smart scheduling (e.g., charging devices during off-peak hours) to actively reduce electricity costs and optimize consumption.
- **Voice Control Integration:** Expansion to virtual assistants (Alexa, Google Assistant, HomeKit) to enhance user convenience and accessibility for the elderly or persons with disabilities.
- **Context-Aware Automation:** Integration of motion or light sensors to eliminate "phantom" standby power by automatically cutting supply when no activity is detected.
- **Scalability into a Smart Hub:** Modular evolution into a centralized decision-making system by linking multiple sensors (temperature, humidity) to trigger automated hardware responses.
- **Market Synergy:** Potential for integration with open-source smart home platforms (e.g., Home Assistant) for advanced customization and local data privacy.

Threats

- **Electrical Safety & Certification Standards:** Non-compliance with strict 230V AC regulations poses risks of fire or failure. Unlike certified commercial products (CE, RoHS), a manual prototype lacks the industrial stress testing required for 24/7 reliability.
- **Market Competition & Economies of Scale:** Established brands benefit from mass production and lower prices. The project must highlight the value of its 3 independent outlets and custom web integration to remain a competitive alternative.
- **MQTT Broker Dependency:** Since the system relies on an external broker (HiveMQ), any downtime of the cloud service or the broker would lead to a loss of remote control, highlighting a dependency on third-party infrastructure.
- **Cybersecurity Risks (Credentials & Data):** While it avoids port forwarding, the device still transmits data over the network. Without full TLS/SSL encryption for the MQTT connection, there is a risk of "man-in-the-middle" attacks where credentials or status updates could be intercepted.
- **Material Durability:** 3D-printed enclosures or manual assemblies do not offer the same thermal resistance (V0 fire rating) as industrial-grade plastics, which is a barrier for use in high-temperature or high-humidity environments.

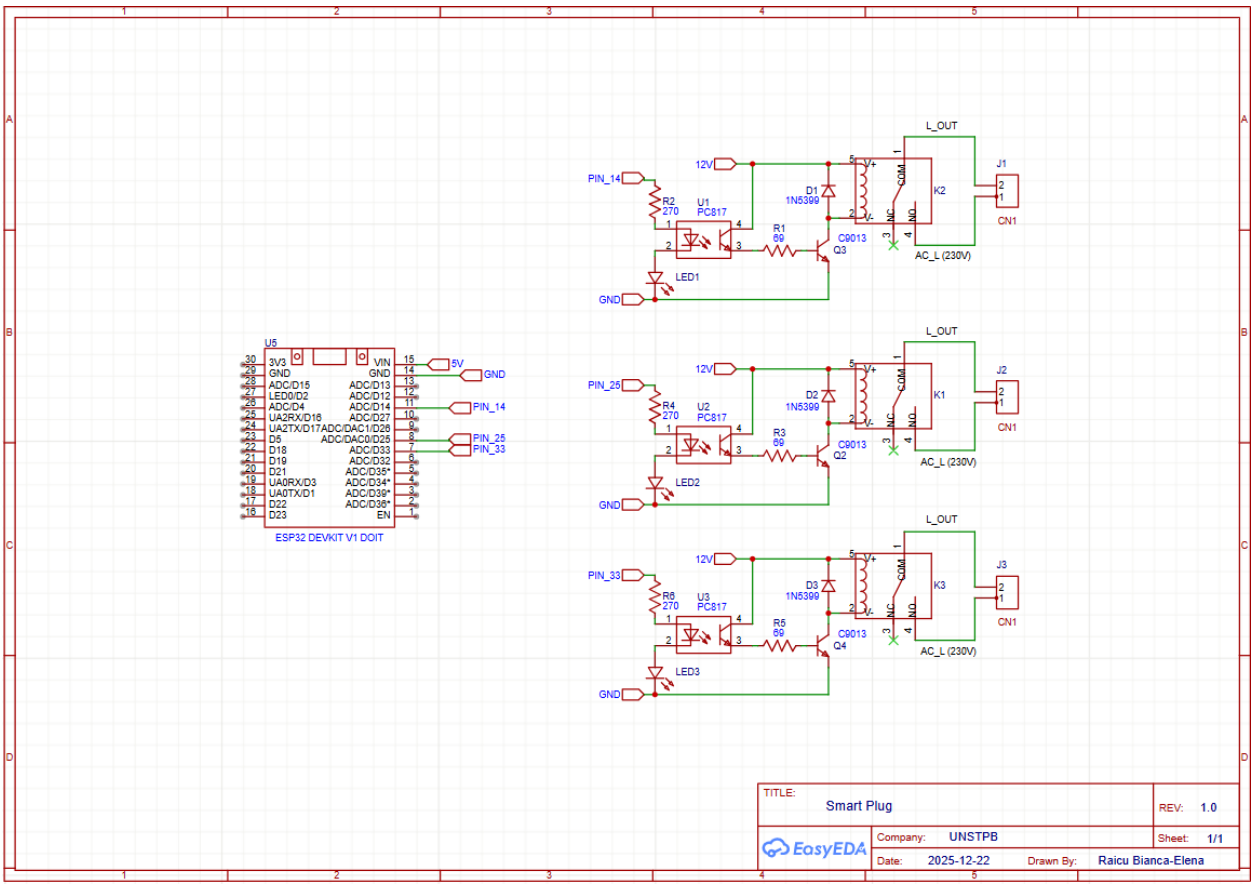
P4. Solution Design

The circuit design was developed with a primary focus on **operational safety** and ensuring **galvanic isolation** between the control logic (ESP32 microcontroller) and the high-power stage (230V AC load). The system features three independent channels, each controlled via dedicated digital pins on the ESP32 (GPIO 14, 25, and 33).

To protect the microcontroller from voltage spikes or electromagnetic interference (EMI) originating from the mains, an isolation stage was implemented using **PC817 optocouplers**. These components facilitate signal transmission via light, eliminating any direct electrical contact between the 3.3V logic and the power circuitry. To ensure optimal operation, **270Ω resistors** were calculated and integrated in series with the internal photodiode and the status LED to limit the forward current.

The switching stage utilizes **NPN transistors (C9013)** acting as electronic switches to amplify the current required to drive the 12V relay coils. Each relay is protected by a **flyback diode (1N5399)** connected in parallel with the coil. These diodes dissipate the counter-electromotive force (back EMF) generated when the coil is de-energized, effectively preventing transistor breakdown due to inductive spikes.

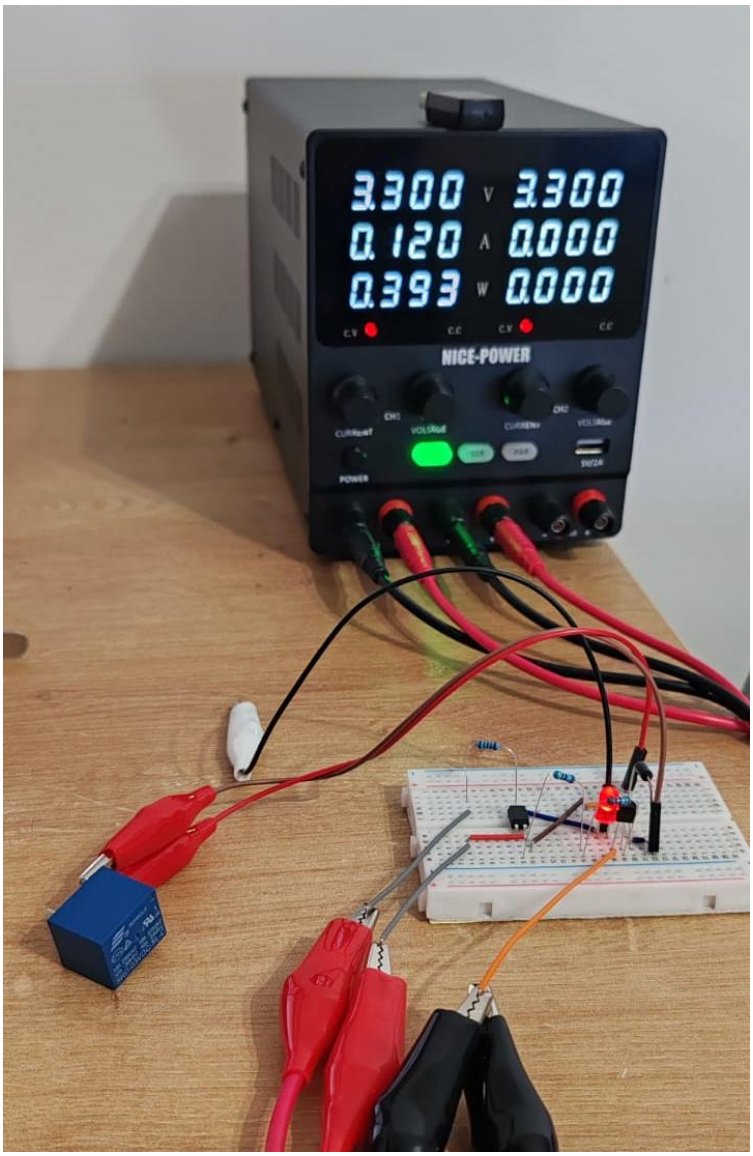
The AC integration routes the **Phase (AC LINE)** through the **COM** and **Normally Open (NO)** terminals, ensuring the load is powered only upon a "HIGH" signal from the ESP32. The **J1-J3 terminals** provide a secure, modular interface for electrical appliances.



P5. Implementation

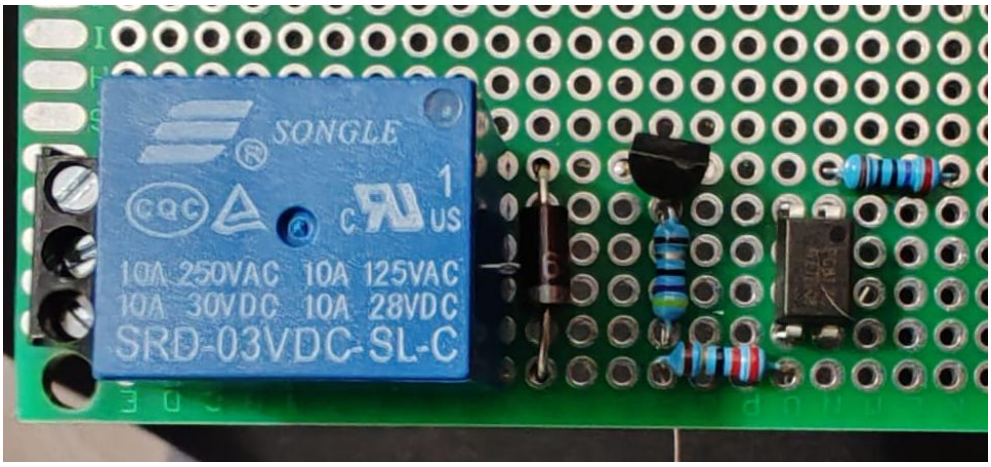
1. Prototyping and Breadboard Validation

This initial phase served as the practical foundation of the project, focusing on verifying component integrity and validating the logical schematic previously simulated. A breadboard was used to temporarily interconnect the ESP32 microcontroller with the isolation stage (PC817 optocouplers and control transistors). To power the circuit and simulate control signals, a laboratory power supply was utilized, set to 3.3V to match the ESP32 logic levels. This stage was critical for monitoring the real-time behavior of status LEDs and measuring the relay coil's current consumption, ensuring that all electrical parameters remained within the **safety margins** of the selected semiconductor components.



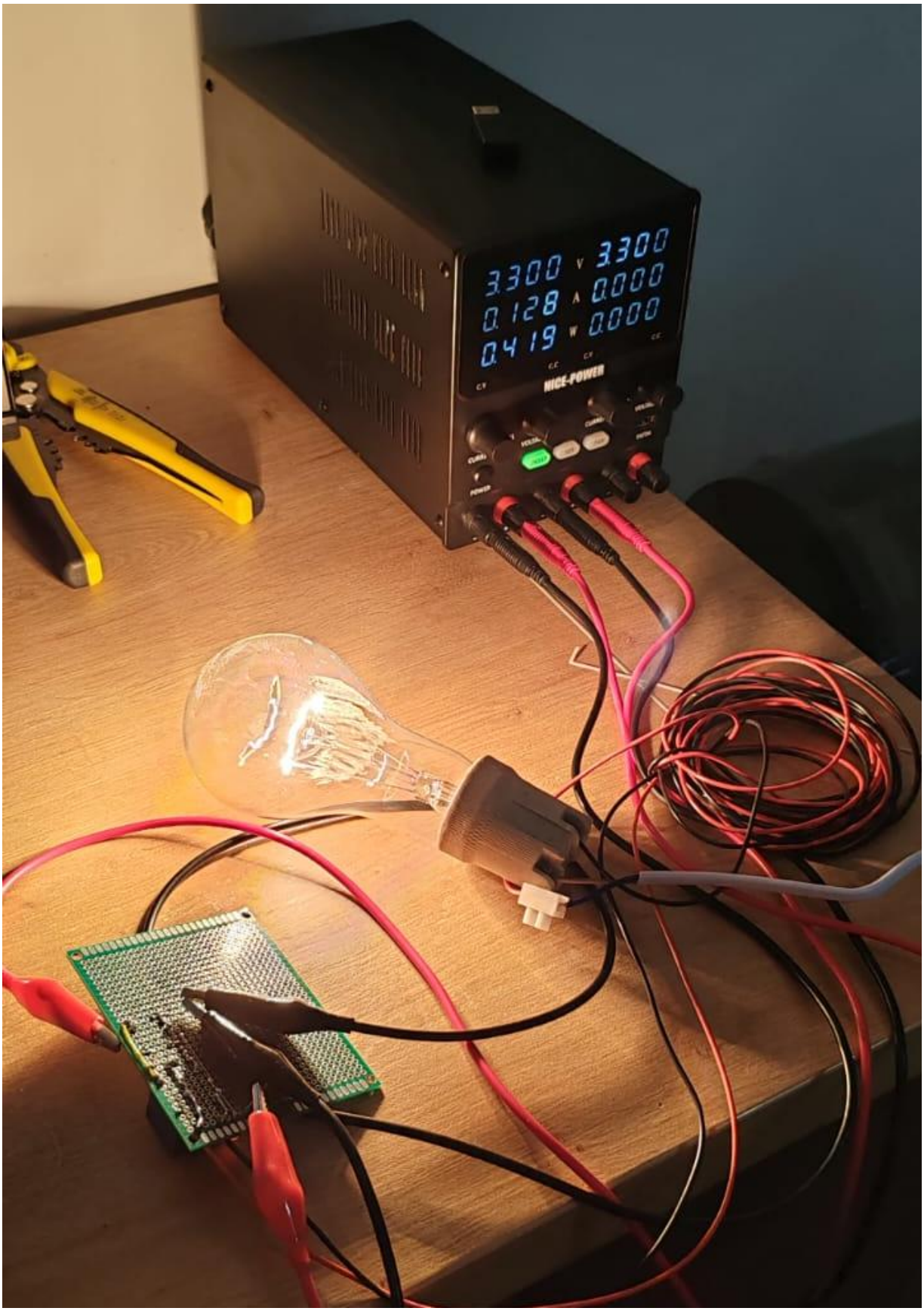
2. Hardware Implementation on Perfboard

Transitioning from the breadboard to a perfboard marked the development of a compact, reliable device, emphasizing resource efficiency and system durability. This stage required rigorous layout planning to maintain safe physical clearance between the low-voltage control logic and the high-voltage (230V AC) power traces, ensuring operational safety—a core principle of sustainable engineering. I performed the final soldering of the Songle relays and integrated 1N5399 flyback diodes to prevent component degradation caused by inductive voltage spikes, thereby extending the system's lifecycle. Using a perfboard ensured a robust assembly, eliminating intermittent connections and reducing material waste associated with failed prototypes. I adopted a modular assembly strategy, validating a single control channel before replication to ensure the **reliability and scalability** of the final system, aligning the hardware's performance with the requirements of an integrated energy management node.



3. Load Testing and System Performance Analysis

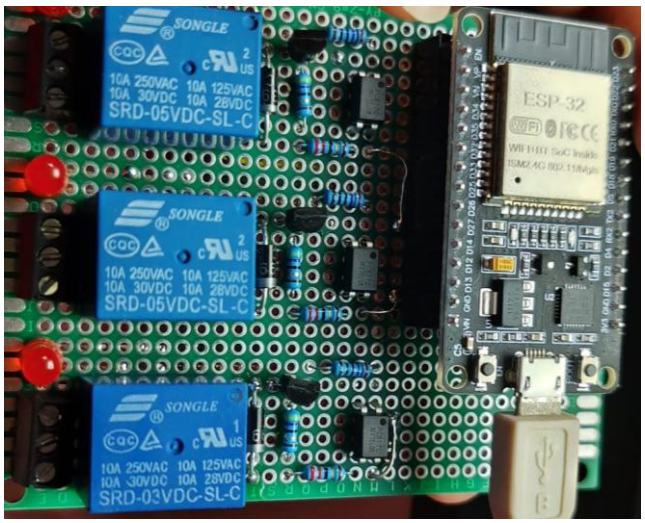
The final phase involved testing the system under real-world conditions by switching a resistive load (a high-power incandescent bulb). During this stage, I monitored the power consumption via the laboratory power supply, which reported a stable current of **0.128A** and a power dissipation of approximately **0.42W** per active relay. This data confirmed the high efficiency of the control circuit and ensured that the switching transistors maintain optimal thermal stability during prolonged operation. The success of this test demonstrated that the optical galvanic isolation functions flawlessly, allowing for the secure management of mains voltage (230V) while safeguarding both the microcontroller and the user. These results validate the system’s reliability as an energy-efficient solution for smart environment control.

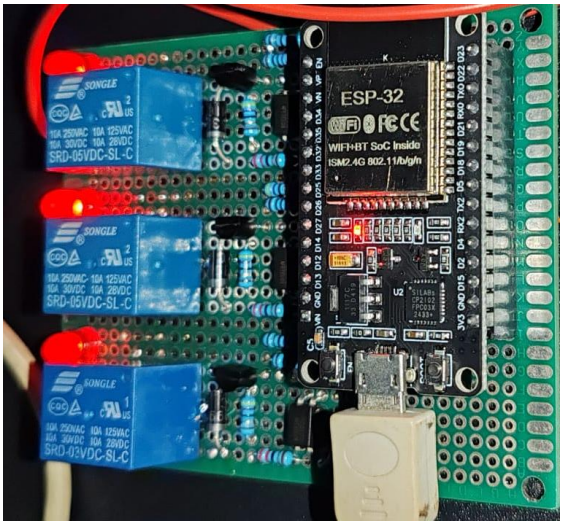

3.300 V 3.300
0.128 A 0.000
0.419 W 0.000
NICE-POWER

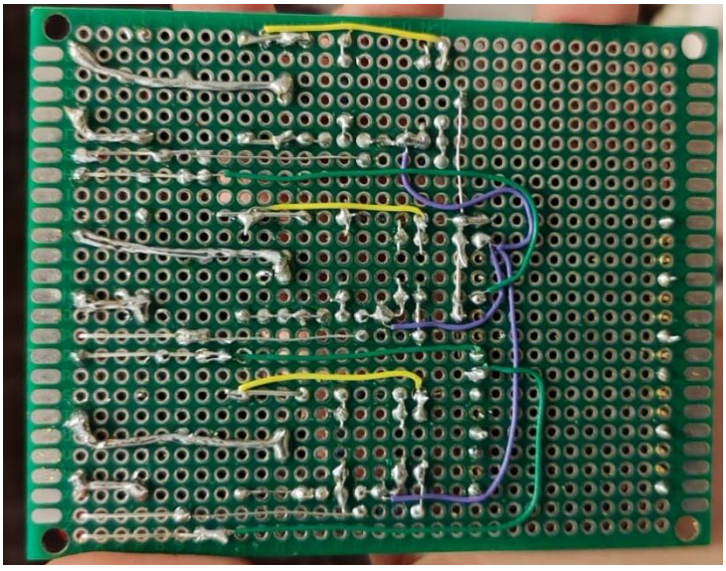
4. Assembly of the Central Control Unit

This stage focused on integrating all electronic components onto the final prototyping board, forming the "brain" of the control system. The ESP32 microcontroller was mounted on dedicated sockets, and the wiring for the three control channels was finalized. The circuit features a clear modular organization: the three optocouplers for isolation, the power transistors, and the three Songle relays arranged to facilitate easy access to the output terminals.

While the electronic core is now fully functional and validated, it represents the primary functional unit of the system. The next critical step involves integrating this unit into a protective enclosure and establishing the electrical connections to the external outlets. This final phase will ensure a professional aesthetic and provide the necessary insulation for the safe handling of the 230V AC mains voltage. (Included is a photograph of the underside, demonstrating the quality and integrity of the solder joints).







5. Microcontroller Programming via Arduino IDE

The system's intelligence was developed in **C++**, utilizing an **MQTT-based communication model** instead of a traditional HTTP server. This approach enables instantaneous remote control without complex network configurations. I implemented the **WiFiManager library** to eliminate hardcoded credentials; upon startup or network loss, the ESP32 generates a password-protected captive portal for dynamic Wi-Fi configuration. The microcontroller manages digital pins (GPIO 14, 25, 33) as outputs, utilizing an **asynchronous callback function** to process commands received from the Vercel-hosted interface via secure topics. This architecture ensures minimal latency and high stability, transforming the device into a genuine, portable IoT node.

Below is the implementation of the **Remote Toggle** logic, which processes incoming data packets and provides real-time status updates to the dashboard:

```
void callback(char* topic, byte* payload, unsigned int length) {
  String message = "";
  for (int i = 0; i < length; i++) message += (char)payload[i];

  Serial.println("Comanda primită: " + message);

  if (message == "1_TOGGLE") {
    state1 = !state1;
    digitalWrite(pin1, state1 ? HIGH : LOW);
    client.publish(topic_status, state1 ? "1:ON" : "1:OFF");
  }
  if (message == "2_TOGGLE") {
    state2 = !state2;
    digitalWrite(pin2, state2 ? HIGH : LOW);
    client.publish(topic_status, state2 ? "2:ON" : "2:OFF");
  }
  if (message == "3_TOGGLE") {
    state3 = !state3;
    digitalWrite(pin3, state3 ? HIGH : LOW);
    client.publish(topic_status, state3 ? "3:ON" : "3:OFF");
  }
}
```

The complete implementation, including network configurations and auxiliary libraries, is available in the official project repository: github.com/bia12340/smart_plug.

6. Web Interface Development (HTML/CSS/JS)

To provide an intuitive and secure user experience, I developed a responsive web application using **HTML5, CSS3, and JavaScript**, hosted on the **Vercel** platform. The interface features a minimalist design where interactive elements provide real-time visual feedback, reflecting the relays' status through color-coded updates.

Technically, the application bypasses local IP dependencies by utilizing WebSockets (WSS) to communicate with the MQTT broker. Security is integrated through a **SHA-256 hashing layer**; access credentials are never stored in plain text but as encrypted digital fingerprints verified asynchronously during login. Additionally, I implemented localStorage to maintain active sessions on authorized devices, striking a balance between robust security and user accessibility.

A. Security Logic (Hashing)

The system uses the Web Crypto API to hash user input, comparing it against a pre-defined hex string to grant access without exposing the actual password in the source code.

```
async function checkPass() {
    const input = document.getElementById('pass-input').value;

    // Algoritm SHA-256 pentru a cripta parola introdusă
    const msgBuffer = new TextEncoder().encode(input);
    const hashBuffer = await crypto.subtle.digest('SHA-256', msgBuffer);
    const hashArray = Array.from(new Uint8Array(hashBuffer));
    const hashHex = hashArray.map(b => b.toString(16).padStart(2, '0')).join('');
    const parolaCriptata = "3f21e50243a4f1b966e1acf9b60fd0dad7def7c4c5593385e1fca6f3099a4261";

    if(hashHex === parolaCriptata) {
        localStorage.setItem("isLoggedIn", "true");
        document.getElementById('login-screen').style.display = "none";
    } else {
        alert("Cod incorect!");
    }
}
```

B. MQTT Protocol & Connectivity

The application establishes a secure connection via wss://broker.hivemq.com:8884/mqtt, subscribing to status topics to ensure the UI stays synchronized with the hardware state globally.

```
const client = mqtt.connect('wss://broker.hivemq.com:8884/mqtt');
const topicCmd = 'bia_smart_12340/priza/comenzi';
const topicStatus = 'bia_smart_12340/priza/status';

client.on('connect', () => {
    document.getElementById('status').innerText = "Online";
    document.getElementById('status').style.color = "#22c55e";
    client.subscribe(topicStatus);
});
```

C. UI Synchronization (Asynchronous Logic)

The interface utilizes an event-driven model where the client.on('message') listener parses incoming payloads and triggers updateUI() functions, ensuring minimal latency between the physical device and the digital dashboard.

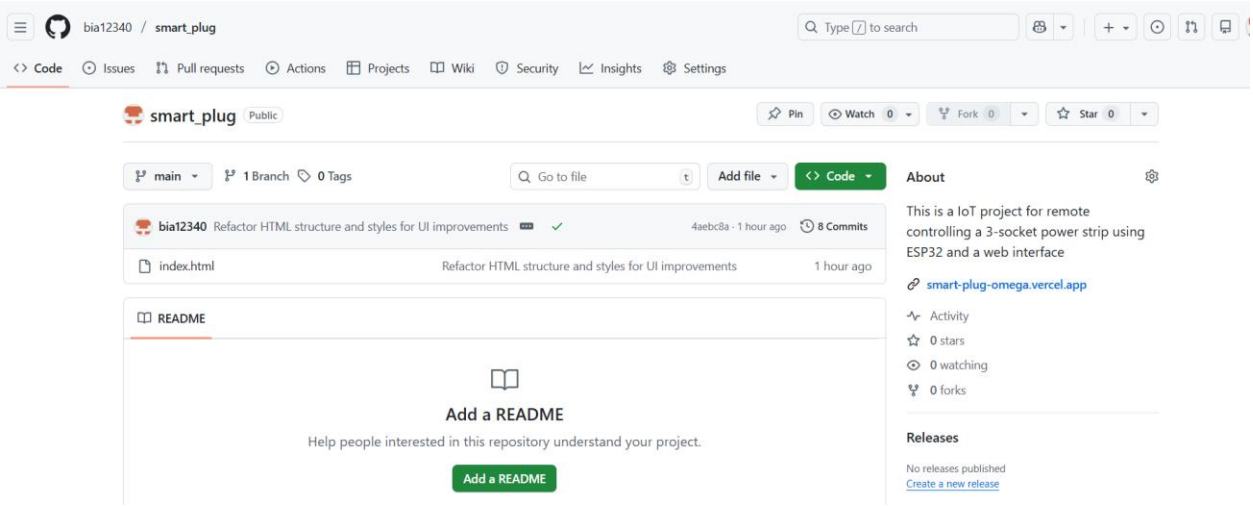
```
client.on('message', (topic, payload) => {
  const msg = payload.toString();
  const [id, state] = msg.split(':');
  updateUI(id, state);
});

function toggle(id) {
  client.publish(topicCmd, id + "_TOGGLE");
}
```

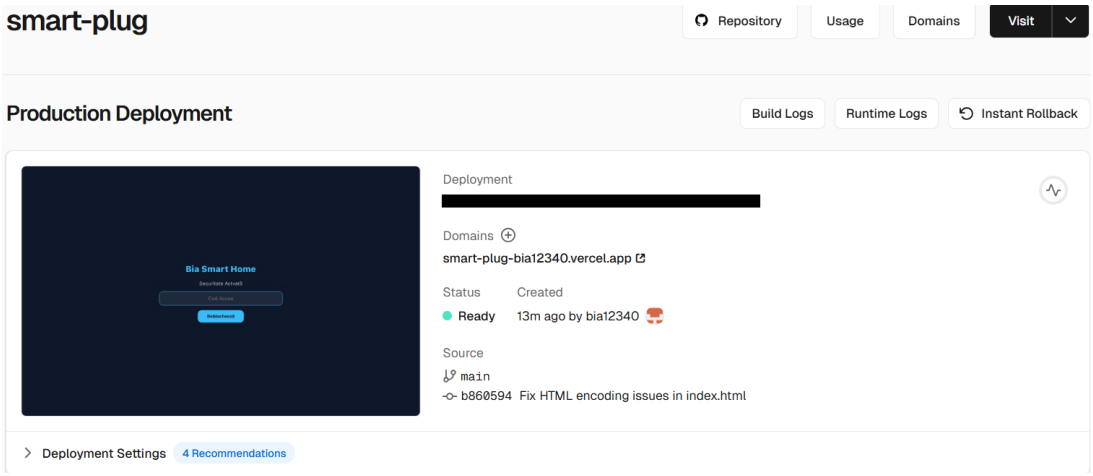
7. Version Control via GitHub and Vercel Deployment

For code management and version traceability, I utilized **GitHub**, maintaining the project's source files in a dedicated repository. This approach enabled a streamlined workflow through integration with the **Vercel** platform, which performs automatic deployment with every code update.

By generating a secure public link (HTTPS), I eliminated local network constraints. Due to the architecture based on **WebSockets** and the MQTT broker, the interface hosted on Vercel communicates instantaneously with the ESP32, regardless of the user's geographic location. Consequently, the system benefits from a stable cloud infrastructure, offering global control over the smart outlets via any mobile device without compromising data security or response speed.

Project Repository: https://github.com/bia12340/smart_plug

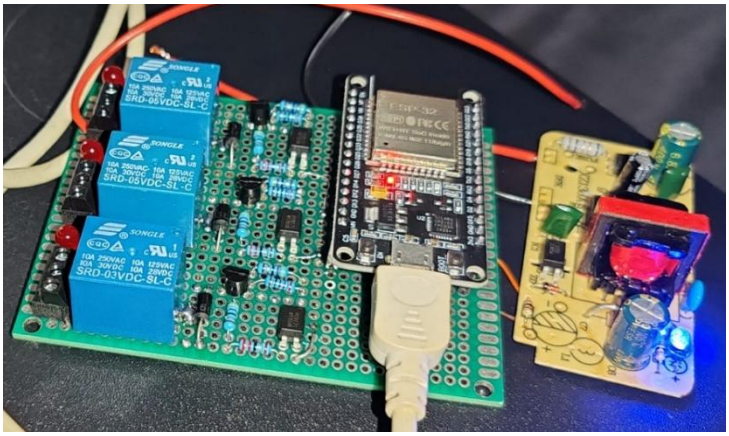


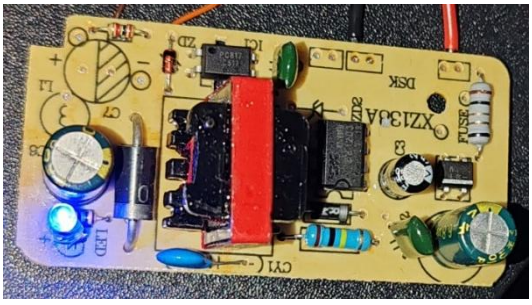


8. Power Supply System (AC-DC Converter)

Following the successful validation of the firmware, I integrated a compact XZ138A AC-DC Step-Down Converter to transform the 230V AC mains voltage into the stable 5V DC required by the ESP32 and relay modules. This component is vital as it eliminates the dependency on external USB power, evolving the prototype into a fully autonomous, standalone device.

The power module includes a specialized filtering stage—featuring the electrolytic capacitors visible in the assembly—and an isolation transformer. These elements ensure a stable current flow and safeguard the logic components against mains voltage fluctuations or electrical noise. By integrating this internal power supply, I finalized the transition from an externally powered model to a fully integrated IoT system, ready for final enclosure mounting and deployment in a real-world energy management context.

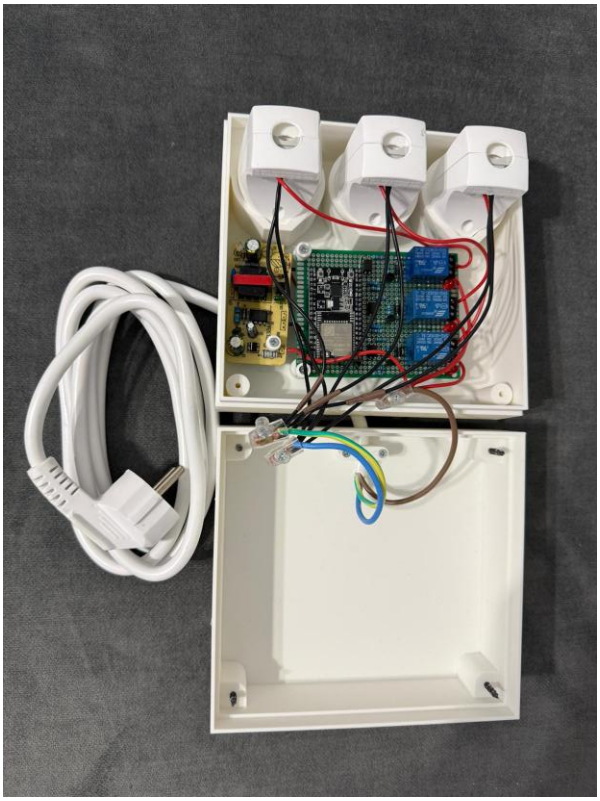


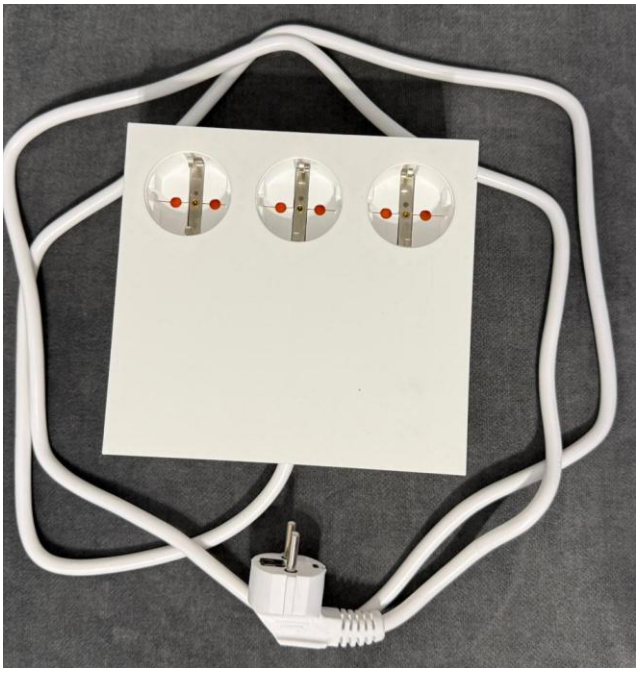


9. Hardware Integration and Final Assembly

The final phase of the project involved consolidating the electronic circuitry into a compact, secure, and functional structure. The logic components (ESP32 and the XZ138A power supply), along with the relay bank, were integrated into a custom-designed 3D-printed enclosure. This housing was specifically engineered to ensure electrical insulation and adequate ventilation, preventing thermal buildup during operation.

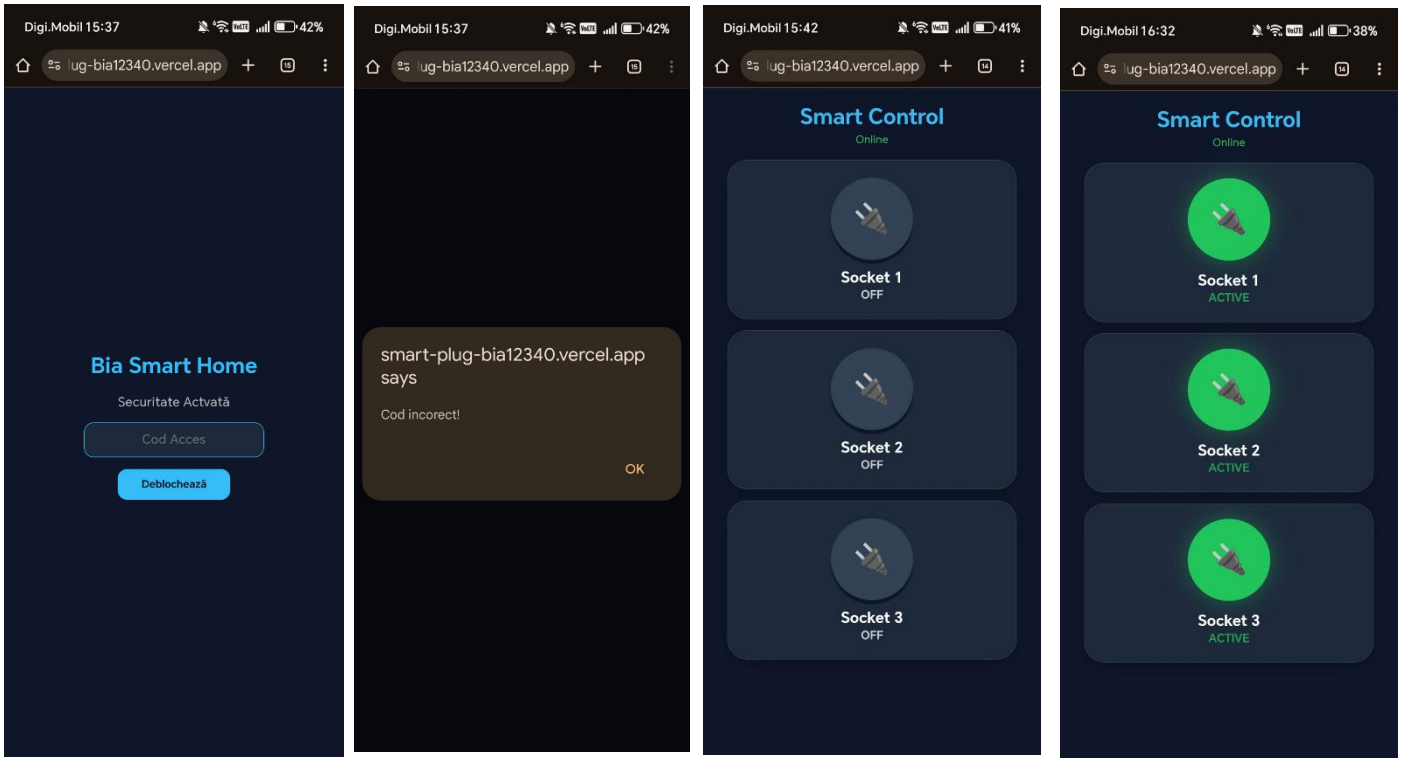
High-power connections were established using copper conductors appropriately sized for residential loads, concluding with the assembly of the main power plug and the individual output sockets. This integration transforms the prototype from a laboratory test board into a deployable IoT device, bridging the gap between software precision and a robust, aesthetically professional hardware construction.





10. User Interface and Operational Workflow

To demonstrate the system's functionality and security features, the following images illustrate the complete user journey. This includes the secure authentication screen with SHA-256 hash verification, the error handling mechanisms for unauthorized access, and the real-time synchronization between the digital dashboard and the hardware's power states (Active/Inactive). The final visual confirmation showcases the physical ESP32 unit successfully driving high-power loads upon receiving remote commands.



11. Configuration and User Instructions

The system was engineered for intuitive use, with the setup process divided into three streamlined steps:

- **Network Connectivity:** Upon initial power-up (or when moved to a new location), the device attempts to connect to the previously saved network for 3 seconds. If unavailable, the smart plug generates its own Wi-Fi Access Point named **"Configurare_Priza_Smart"**. The user connects to this network (password: *lumos123*) and inputs the local Wi-Fi credentials via the captive portal that automatically appears.
- **Authentication:** Once the device is online, the user accesses the web interface hosted on Vercel. To enable the control features, the secure access code (*Lumos3*) must be entered.
- **Control and Monitoring:** After logging in, each outlet's state can be toggled by pressing the corresponding buttons. The interface provides real-time feedback by updating the color-coded status indicators, confirming the successful bidirectional communication between the cloud and hardware via the MQTT protocol.

P6. References

https://randomnerdtutorials.com/projects-esp8266/

https://www.w3schools.com/js/

https://www.hivemq.com/mqtt/

https://auth0.com/blog/hashing-passwords-one-way-road-to-security/

https://emn178.github.io/online-tools/sha256.html

https://vercel.com/docs/git

https://www.youtube.com/watch?v=LvweY-6NV6A

https://www.youtube.com/watch?v=VnfX9YJbaU8

https://www.youtube.com/watch?v=DMm20Z7SF4M