PROJECT ABSTRACT

Smart Power Strip with Per-Outlet Energy Monitoring, Scheduling and Android App (ESP32)

A built-to-order hybrid project: a 4-outlet smart extension board where each outlet is individually switched by a relay channel and individually measured by its own PZEM-004T energy sensor, with on-device scheduling, overload cut-off, MQTT telemetry and a companion Android app. Delivered with working prototype, ESP32 firmware, Android app source and APK, wiring documentation, calibration procedure, report, PPT and viva kit.

01. Title

Smart Power Strip with Per-Outlet Energy Monitoring, Scheduling and Android App (ESP32)

02. Introduction / Background

Electricity bills arrive as a single monthly number, so households and small labs cannot tell which device wastes power. Whole-house energy meters show only the aggregate, and single-outlet smart plugs are usually locked to a vendor cloud and cover one socket at a time. The useful engineering gap is a multi-outlet board where every socket is both switchable and honestly measurable. This project fills it: a 4-outlet smart power strip built around an ESP32, where each Indian 3-pin outlet passes through its own relay channel and its own PZEM-004T energy sensor. Voltage, current, active power, cumulative energy, frequency and power factor are read per outlet over Modbus RTU, daily schedules run on the ESP32 itself with an NTP-synced clock, and a Kotlin Android app over local Wi-Fi provides live readings, toggles, schedule editing and tariff-based cost estimates.

03. Problem Statement

Plug loads are invisible: a whole-house meter cannot attribute consumption to individual devices, and single-outlet smart plugs do not scale to a board and depend on a vendor cloud. Students of IoT and electrical systems rarely get to build the complete sensing-to-actuation chain on mains hardware — sensor networking, relay control, real-time scheduling and a mobile control surface in one demonstrable build. This project addresses that by delivering a 4-outlet extension board with genuine per-outlet measurement (one PZEM-004T per outlet on a unique Modbus address), per-outlet relay switching with overload cut-off, on-device schedules, local MQTT telemetry and an Android companion app — all verifiable with the included buyer-run calibration procedure.

04. Objectives

- Measure per-outlet voltage, current, active power, cumulative energy, frequency and power factor using four PZEM-004T modules, each on a unique Modbus slave address.
- Switch each of the 4 outlets independently through a 4-channel 10 A relay module from firmware, schedules or the Android app.

- Implement on-device daily scheduling on the ESP32 with NTP time sync, stored in flash so timers execute with no phone or internet present.
- Provide per-outlet overload cut-off at a configurable limit (default 6.5 A design target) with buzzer indication and a fused mains inlet.
- Publish readings and relay states over local Wi-Fi to an MQTT broker and expose an HTTP API for direct control.
- Build a Kotlin Android app with live per-outlet cards, on/off toggles, a schedule editor and monthly cost estimates at the configured tariff.
- Supply a buyer-run calibration procedure so measurement accuracy is verified on the student's own build.
- Deliver the complete student kit: prototype, firmware, app source and APK, wiring docs, report, PPT and viva Q&A.;

05. Proposed Methodology

The system is developed in four stages. (1) Hardware: a white enclosure carries four Indian 3-pin sockets; each outlet's phase line passes through one channel of the relay module and one PZEM-004T sensor, with the PZEMs set to unique Modbus addresses and a fused mains inlet. (2) Firmware (Arduino C++ on ESP32): polls the four sensors over UART/Modbus RTU about once per second, evaluates the schedule table and overload limits, drives the relays, publishes to MQTT, and serves the HTTP API. (3) App: a Kotlin Android app connects over local Wi-Fi for live readings, toggles, schedule editing and cost display. (4) Verification: the student runs the included calibration procedure against a known reference load and records each channel's readings for the report.

06. System Architecture / Working

Each outlet is an independent sensing-and-switching channel: mains phase runs through the relay contact and the PZEM-004T's measurement path, while neutrals share a common busbar. The ESP32 polls all four PZEMs over Modbus RTU, collecting V, I, P, kWh, frequency and power factor per outlet roughly once per second. The firmware layer compares readings against the on-device schedule table (NTP-synced clock) and the per-outlet overload limit; on overload it opens that outlet's relay, sounds the buzzer and publishes a trip event. Readings and states flow over Wi-Fi to the local MQTT broker and the HTTP API. The Android app subscribes to the same feed for its live cards and sends toggle/schedule commands back; schedules themselves execute in firmware, so the app is a control surface, not a dependency.

07. Hardware / Software Requirements

- ESP32 development board (Wi-Fi)
- 4 × PZEM-004T energy sensor modules with current transformers (one per outlet)
- 4-channel relay module (10 A 250 VAC per datasheet)
- 4 × Indian 3-pin 230 V sockets, enclosure, fused mains inlet, screw terminals
- Buzzer and per-outlet status LEDs
- Arduino IDE with ESP32 core (C++ firmware), Modbus and MQTT libraries
- Local MQTT broker (Mosquitto class) on the Wi-Fi network

- Android Studio, Kotlin (companion app); NTP access for clock sync

08. Expected Outcomes

- A working 4-outlet prototype with genuine per-outlet energy measurement and switching.
- On-device schedules that execute without the phone or internet.
- Per-outlet overload cut-off with local indication and app flagging.
- Live per-outlet readings, toggles, schedules and tariff-based cost estimates in the Android app.
- A documented calibration procedure with buyer-verified channel readings for the report.
- Complete report, PPT and viva Q&A; covering Modbus sensing, relay control, scheduling and MQTT telemetry.

09. Applications

- Teaching mains interfacing, Modbus sensor networking and relay control in IoT/electrical labs.
- Household and hostel energy awareness: attributing units and cost to individual devices.
- Demonstrating on-device scheduling vs cloud-dependent automation.
- Base platform for student extensions: cloud logging, voice control, or per-outlet prepaid metering.

10. Future Scope

- Cloud dashboard with remote-outside-LAN access (configurable extension).
- Per-outlet energy budgets with automatic cut-off at a monthly kWh cap.
- Voice-assistant integration via the MQTT/HTTP API.
- 16 A outlet variant for heavier loads with appropriately rated sockets and wiring.
- Load-disaggregation study: comparing measured per-outlet data against NILM estimates.