

Smartura: Development of an Internet of Things-Based Electrical Control and Data Acquisition System

Chika Hayya Sabillah*

*Computer Engineering Technology, College of Vocational Studies, IPB University
chikahayya_asabillah@apps.ipb.ac.id,

Abstract

Indonesia's rising residential electricity consumption and the passive nature of conventional electrical terminals create risks of energy waste, overload, and fire hazards, since existing terminals offer no monitoring or remote-control capability. This research aimed to design and implement Smartura, an IoT-based smart electrical terminal integrating remote load control, real-time monitoring of electrical parameters, a usage timer, automatic overload protection, and energy/cost estimation in a single system. The system uses an ESP32 microcontroller, a relay actuator, ACS712 and ZMPT101B sensors for current and voltage sensing, and a website connected to Firebase for authentication and real-time data exchange. The method comprised a literature study, observation, hardware and software design, and performance testing on data collected from 5–11 June 2026. Results showed reliable on/off control through the dashboard, recording 5,785 relay-on and 408 relay-off data points out of 6,193 readings. Voltage and current monitoring achieved MAPE values of 1.69% (very good) and 12.38% (good), respectively. The timer and overload-protection algorithm worked as designed, disconnecting the relay when threshold criteria were met (demonstrated at a safe functional test threshold of 3.30 A / 630 W for lab safety, with an intended operational threshold set at 5.45 A / 1,200 W). Average measured values were 213.15 V, 0.164 A, 35.07 W, 3.62 kWh, and an estimated cost of Rp5,229.29. Smartura demonstrates a functional, low-cost IoT prototype for residential electrical monitoring and simple load control.

Keywords: internet of things, electrical monitoring, smart electrical terminal, overload protection, ESP32.

INTRODUCTION

Electrical energy consumption in Indonesia has grown significantly in line with economic development and improvements in public welfare. Data from the Ministry of Energy and Mineral Resources (ESDM) indicate that Indonesia's per-capita electricity consumption increased from 1,123 kWh in 2021 to 1,411 kWh in 2024, with a target of 1,464 kWh per capita in 2025. The residential sector is the largest contributor to national electricity consumption, accounting for a 43.14% share of total electricity sales, or 67.14 TWh, over the same period. National statistics further show that households are the largest contributor to national electricity consumption, accounting for as much as 44.79% of the total usage of 1,153,700 GWh (Permana et al., 2024). This situation gives rise to risks such as energy waste, overcurrent, short circuits, and fire hazards resulting from delayed or uninformed device control (Latifah & Nugeraha, 2024; Siahaan, 2024). Recent studies have also highlighted the importance of policy, energy efficiency, and the social and environmental impacts of surging electricity demand (Farghali et al., 2023; Hargreaves & Middlemiss, 2020; Jabir et al., 2018).

Although the growth in electricity consumption reflects a positive trend in economic development, the challenge of efficiency in the residential sector remains a serious issue. Household appliances such as air conditioners consume 958 GWh per year for every 1% of household penetration, while rice cookers and refrigerators account for 218 GWh and 209 GWh per 1% penetration, respectively. This high level of consumption is further exacerbated by the phenomenon of phantom load, or standby power, in which electronic devices continue to draw power even when not

actively in use (Siskandar et al., 2018; Surahman et al., 2017) . The impact of energy waste is not limited to the economic dimension for households but also has consequences for the burden on the national electricity system and for national energy-efficiency policy targets (Napis et al., 2023). Consequently, there is a need for electricity-consumption monitoring coupled with a control system to make it easier for users to consume electricity as needed.

The use of conventional electrical terminals is generally still passive, functioning merely as a medium for distributing electrical power without offering monitoring or control features. In daily activities, users still must switch electrical loads on and off manually by going directly to the location of the device. This condition becomes inefficient, particularly when the user is far from the device or when the load needs to be used only for a certain period that could otherwise be limited automatically. In addition, conventional electrical terminals generally cannot provide information on the amount of electricity being used, such as voltage, current, and estimated power, making it difficult for users to know the actual condition of the connected load. Figure 1 shows the State of the Art of Smartura as an IoT-based smart electrical terminal system used to monitor and control electricity usage.

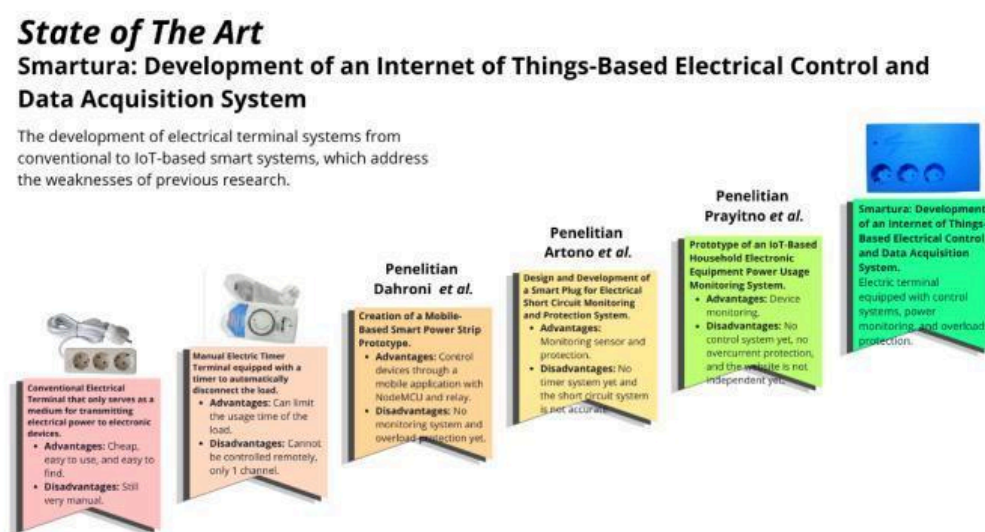


Figure 1. State of the Art of Smartura

The development of the Internet of Things (IoT) opens opportunities to enhance the function of electrical terminals, making them more interactive and informative. IoT is defined as a network of physical devices interconnected via the internet to exchange data and to monitor and control those devices automatically and in real time (Hermansyah et al., 2023; Siskandar et al., 2022). One important implementation of IoT in electrical systems is the development of smart sockets and smart electrical terminals that allow users to control, schedule usage, and remotely monitor power consumption through a mobile application or website (Baharudin, Wijaya, & Algusri, 2025; Permana et al., 2024; Rahmanto, Widodo, & Ayuningtyas, 2023; Siskandar et al., 2020). Research on a mobile-based smart power strip by Dahroni et al. developed a mobile-based smart power strip prototype using a NodeMCU and a relay to control electrical devices via smartphone. That study showed that an electrical terminal can be developed into a remotely controllable system. However, the system focused solely on on/off control and did not emphasize voltage, current, or power monitoring, data history, or overload protection (Dahroni et al., 2020).

Subsequently, Artono et al. developed a smart plug for monitoring and short-circuit protection systems. Their study showed that a smart plug can be used not only as a remote switch but also as a protective device. However, the protection developed was directed solely toward short-circuit conditions, whereas Smartura develops overload protection based on both estimated power and current (Artono et al., 2019). At a later stage, research on IoT-based electricity-usage monitoring began to develop. IoT-based household electric-power monitoring systems can help users learn about

the real-time electricity consumption of electronic devices. However, several studies placed greater emphasis on the monitoring aspect and did not fully integrate the functions of an electrical terminal, relay control, timer, overload protection, and cost estimation into a single system (Akbar et al., 2019; Prayitno et al., 2019).

Table 1. Comparison of Previous Studies and Smartura

Feature	Dahroni et al. (2020)	Artono et al. (2019)	Prayitno et al. (2019)	Smartura
Remote on/off control	✓	✓	✗	✓
Voltage monitoring	✗	✗	✓	✓
Current monitoring	✗	✓	✓	✓
Power monitoring	✗	✗	✓	✓
Usage timer	✗	✗	✗	✓
Overload protection	✗	✓	✗	✓
Energy estimation (kWh)	✗	✗	✗	✓
Cost estimation (Rp)	✗	✗	✗	✓
Historical data logging	✗	✗	✓	✓

As illustrated in Table 1, previous studies have generally focused on isolated or individual functions, such as remote control, multi-parameter monitoring, usage timers, or protection mechanisms. The primary scientific and technical contribution of this work lies in the seamless integration of five key functions remote control, real-time multi-parameter monitoring, usage timer, load-based overload protection, and cost estimation into a single, low-cost IoT prototype using an ESP32, Firebase, and a web interface without requiring a dedicated mobile application.

In this research, that concept is applied to a smart electrical terminal prototype named Smartura, which utilizes an ESP32 as the main controller, a relay as the on/off actuator, an ACS712 sensor for current reading, a ZMPT101B sensor for voltage reading, and a website integrated with Firebase for authentication and real-time data exchange. Smartura is designed not as a commercial energy-measurement instrument but as a three-channel smart electrical terminal prototype that emphasizes its core functions, namely remote on/off control, monitoring of basic electrical parameters, a usage timer, and simple overload protection. This approach was chosen so that the system would remain realistic to implement within the scope of a final project while still offering novelty and practical value. Through this design, the research is expected to demonstrate how the integration of hardware, an embedded system, a real-time database, and a web application can be applied within a simple yet functional IoT system.

METHODS

The research entitled “Smartura: Development of an Internet of Things-Based Electrical Control and Data Acquisition System” was carried out over the period from October 2025 to June 2026. The research was conducted at the Hardware Laboratory and the Bengkel Elektromekanik Laboratory, School of Vocational Studies, IPB University.

This research method is divided into three main parts: a literature study, design, and performance testing. The research method was structured sequentially so that the work would proceed in an orderly and systematic manner. The initial stage consisted of a literature study and identification of system requirements based on existing problems related to Internet of Things technology, particularly with respect to voltage- and current-monitoring systems and control systems for electrical terminals. The researcher gathered relevant literature to be cited as references, listed in the References section. This was followed by system analysis and the design of clear architecture, realized in the form of hardware and software composition. Functional performance testing was then carried out to ensure that the device could be monitored and controlled. A diagram of the research method is shown in Figure 2.

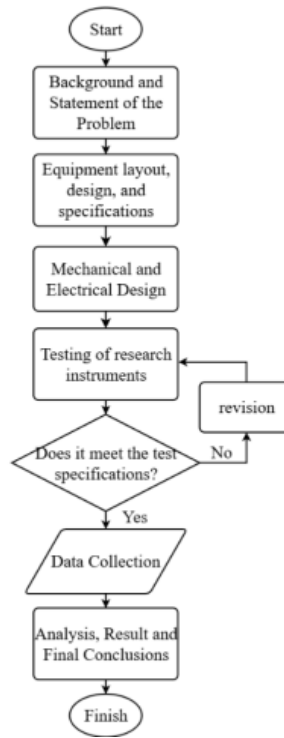


Figure 2. Research Method

The second stage was observation. Observation was carried out by examining the use of conventional electrical terminals, which still require manual control and do not yet provide electricity-usage information. The observation results showed that users need an electrical terminal that can be monitored, controlled remotely, and capable of providing real-time information on load conditions.

The third stage was preliminary experimentation and measurement. Experiments were carried out to test the readings of the ZMPT101B and ACS712 sensors, relay control, the ESP32's connection to Wi-Fi, data transmission to Firebase, and the display of data on the dashboard. Preliminary measurements were performed to determine whether the voltage, current, and estimated-power values could be read and would change in accordance with the load condition. Based on the information obtained from these two main methods, the literature study and observational requirements analysis for the equipment was produced, covering hardware, software, and the system, as shown in Table 2 and Table 3.

Table 2. Hardware Components

Component used	Alternative component	Function
ESP32-Wroom	ESP32 DevKit V1, NodeMCU ESP8266	Main microcontroller for reading sensors, controlling the relay, and sending data to Firebase.
Hi-Link HLK-10M05	HLK-PM01 5V, 5V DC adapter	Converts 220 V AC voltage into 5 V DC as the system's power supply.
1-Channel 5V Relay Module	2-channel 5V relay, solid-state relay	Electronic switch for connecting and disconnecting the electrical supply to the load.
ACS712	PZEM-004T, SCT-013	Current sensor for reading the load's electrical current.
ZMPT101B	PZEM-004T, AC voltage sensor	Voltage sensor for reading the AC voltage entering the device.
5V LED	3 mm/5 mm LED, buzzer	Wi-Fi connection status indicator.

Table 3. Software Components

Name	Description
Arduino IDE	Used to write, upload, and test the ESP32 program, including sensor reading, relay control, Wi-Fi Manager, timer, and Firebase communication.
VS Code	Used to develop the Next.js/React-based Smartura website, including the landing page, sign-up, sign-in, dashboard, and monitoring features.
Firebase	Used as the system backend, covering user authentication, the Realtime Database for monitoring, relay control, timer status, connection status, and data history.
Web browser	Used by the user as a search engine and to access the Smartura website, log in, view the dashboard, control the relay, monitor sensor data, and download measurement history.

Hardware components are used to carry out the device's physical functions, such as reading voltage, reading current, controlling the relay, and supplying power to the control circuit. Meanwhile, the software is used to manage the system's working logic, send data to Firebase, display information on the website, and receive commands from the user. Following the analysis of hardware and software requirements, the researcher proceeded to the next stage, namely the experimentation and measurement stage, testing the performance of the system across the hardware and software components. This experimentation, or performance testing, stage constitutes an evaluation process to measure effectiveness and efficiency in achieving the previously established objectives.

2.1 Hardware Design

Hardware design is the stage of designing the circuitry among the electronic components built to construct the system and mechanize the device's operation. This hardware design covers the block diagram, flowchart, and schematic of the circuit among components.

A block diagram is a representation of the communication among the electronic components used. This system begins with AC electrical current, which is fed in parallel to a step-down converter functioning as an AC-to-5V-DC converter, to a relay functioning as a switch, and to voltage and current sensors for measuring the voltage and current of the electricity entering the device. All the devices then communicate according to the program code loaded into the ESP32 microcontroller, which controls the functions of each component (El-Khozondar et al., 2024). The resulting data processed by the ESP32 is then sent to Firebase, after which the result is displayed on the Smartura website. An illustration of the block diagram is shown in Figure 3.

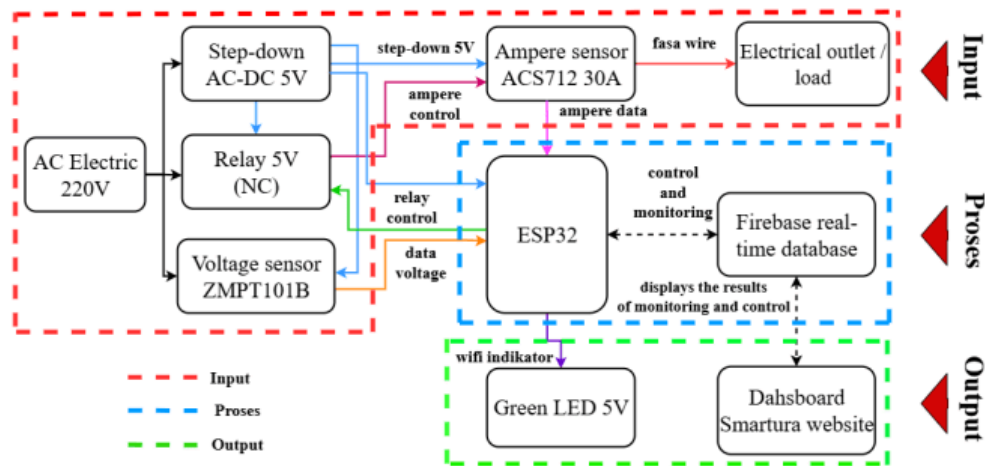


Figure 3. Hardware Block Diagram

This hardware circuit shows the wiring flow that is connected, beginning with the 220 V AC power source entering the 5V step-down module to supply the control circuit consisting of the ESP32, relay, indicator LED, and sensor circuit. The phase line of the AC electricity is first routed to the relay (Sundaramoorthi & Alexander, 2023), then to the ACS712 sensor, after which the neutral line is routed directly to the output terminal. Meanwhile, the ZMPT101B sensor is connected across the phase and neutral lines to read the AC electrical voltage (Liang et al., 2023). This arrangement allows voltage and current readings to be performed in a straightforward manner, while the relay continues to

function as the load-connecting and load-disconnecting switch. The arrangement of the circuit among Smartura's components is shown in Figure 4, and the pin initialization of the hardware components on the ESP32 is shown in Table 4.

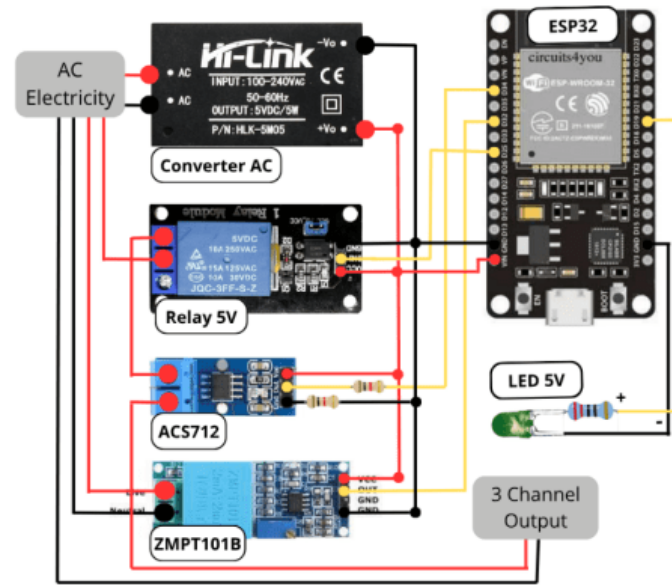


Figure 4. Hardware Component Circuit

Table 4. Hardware Pin Initialization on The ESP32

Component	Pin ESP32
Relay 5V	GPIO 25
ACS712	GPIO 34
ZMPT101B	GPIO 32
LED 5V	GPIO 19

The flowchart's operational flow is designed to be cyclical, since Smartura is a monitoring and control system that operates continuously if the device is connected to the electricity source and the Wi-Fi network. After data is sent to Firebase, the system returns to reading the latest command from the website, updating the relay status, reading sensor data, checking the timer, and detecting an overload condition. This cyclical operating pattern is required so that the dashboard can display changes in the device's condition in real time. Through this flow, Smartura can respond directly to user commands as well as to changes in load conditions. If the estimated power exceeds 1,200 W, or the current exceeds the specified threshold, the relay is automatically disconnected, and an overload notification is sent to the dashboard. The hardware flowchart for Smartura is shown in Figure 5.

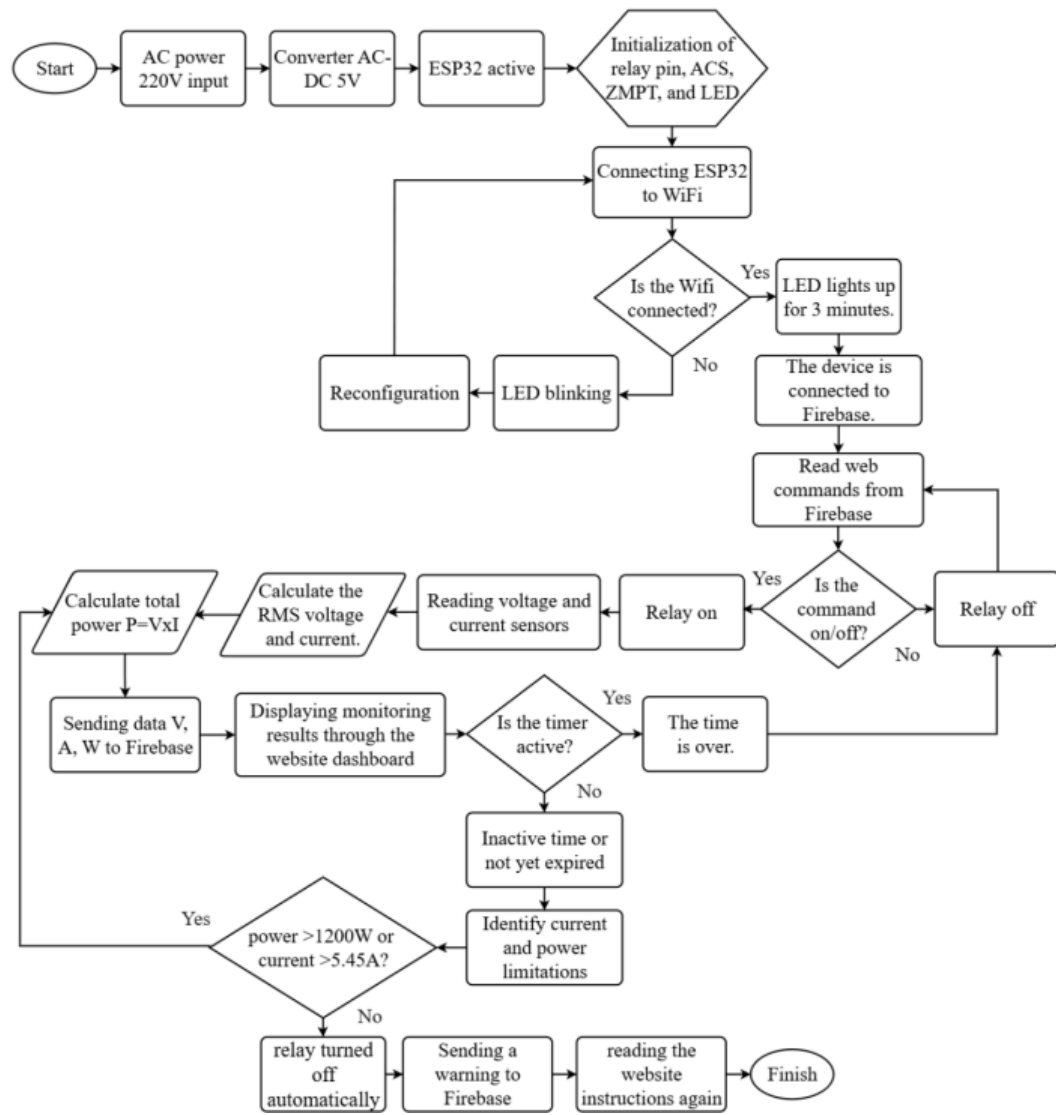


Figure 5. Hardware System Flowchart

2.2 Software Design

The website's initial display shows the landing page of the Smartura device as an introductory page, while the authentication page is used to restrict user access prior to entering the dashboard. This page contains an explanation and brief information about the Smartura device as an IoT-based smart electrical terminal with monitoring, load-control, timer-control, overload-protection features, and total energy and cost-estimation information. It also contains an authentication feature (sign in or sign up) for users to register their account by entering an email address and password.

Once successfully signed in, the user can monitor electricity conditions, control the relay, set the timer, view usage graphs, receive overload notifications, and obtain information on the amount of energy and cost incurred by the electronic devices used. The dashboard page serves as the main page for monitoring electricity conditions, controlling the relay, setting the timer, and viewing device status, receiving overload notifications, displaying historical graphs, and analyzing power usage and cost estimation.

This system architecture describes several of the main software functions on the Smartura dashboard page, namely the on/off control button, timer control, display of current- and voltage-sensor reading data, estimated power (watts), overload protection, and cost estimation.

Figure 6 is a code excerpt for the on/off control button, which can issue on/off commands to the device. It begins with a user command via the website: when the button is in the “on” state, the button icon turns green, and when in the “off” state, the button icon turns red. This control command is then sent first to the Firebase Realtime Database (RTDB); subsequently, the reading from the RTDB sends the command to the ESP32 as the control brain, which controls the relay function, is then processed, and the device switches off or on according to the user's command.

```

224 // Kontrol Saklar (Relay)
225 const toggleRelay = async () => {
226   try {
227     const isOverload = deviceData.alerts?.overload || deviceData.monitoring?.overload || false;
228     const currentState = deviceData.monitoring?.relay_state || deviceData.control?.relay_state || false;
229     const newState = !currentState;
230

```

Figure 6. Website Code for On/Off Control

Figure 7 is a code excerpt for the timer-control feature, which functions to provide automatic control in the form of a per-second countdown time limit. In this feature, the user enters a duration figure in minutes to control how long the device will remain on. Once the countdown time has elapsed, the relay path is automatically disconnected, and the device switches off or ceases to function; the user can also cancel the timer by pressing the available button.

```

257 // Memulai Timer ESP32
258 const startTimer = async () => {
259   try {
260     const minutes = parseInt(timerInput, 10);
261     if (!minutes || minutes <= 0) {
262       alert("Masukkan angka menit yang valid.");
263       return;
264     }
265     const durationMs = minutes * 60 * 1000;
266

```

Figure 7. Website Code for Timer Control

Figure 8 shows a code excerpt for the real-time monitoring of sensor data, consisting of electrical voltage data (volts), electrical current data (amperes), and estimated power-energy data (watts). On the dashboard page, the results of the sensor-data calculation and measurement are updated at a 2-second interval, both in numerical and graphical form. This interval was chosen as a compromise between the speed of data updates and the efficiency of the system's resource usage. A 2-second interval was deemed sufficient to display the electrical-load condition on a near-real-time basis without imposing excessive communication load on the ESP32, the Wi-Fi network, or Firebase. Furthermore, simple electrical parameters generally do not change significantly within an interval of less than 2 seconds, so this interval is still able to represent the system's condition adequately. For energy-usage analysis purposes, historical data is not stored every 2 seconds but rather every 1 minute. The separation of intervals between the real-time data and the historical data aims to reduce the volume of data stored in Firebase, making storage usage more efficient, while the dashboard remains able to display the device's condition information responsively.

```

823 /* Legend Manual */
824 <div style={{ display: 'flex', gap: '1.5rem', marginBottom: '1rem', flexWrap: 'wrap' }}>
825   <div style={{ display: 'flex', alignItems: 'center', gap: '0.4rem' }}>
826     <span style={{ width: 12, height: 3, background: '#34d399', borderRadius: 2, display: 'inline-block' }}></span>
827     <span style={{ fontSize: '0.8rem', color: 'var(--text-secondary)' }}>Tegangan (V)</span>
828   </div>
829   <div style={{ display: 'flex', alignItems: 'center', gap: '0.4rem' }}>
830     <span style={{ width: 12, height: 3, background: '#3b82f6', borderRadius: 2, display: 'inline-block' }}></span>
831     <span style={{ fontSize: '0.8rem', color: 'var(--text-secondary)' }}>Arus (A)</span>
832   </div>
833   <div style={{ display: 'flex', alignItems: 'center', gap: '0.4rem' }}>
834     <span style={{ width: 12, height: 3, background: '#ec4899', borderRadius: 2, display: 'inline-block' }}></span>
835     <span style={{ fontSize: '0.8rem', color: 'var(--text-secondary)' }}>Daya (W)</span>

```

Figure 8. Website Code for Sensor-Data Monitoring

Figure 9 is a code excerpt for the load- or overload-protection feature, which works such that if the current or the estimated power exceeds the maximum threshold, the relay disconnects the path, and the device automatically switches off and ceases to function. This feature serves to protect the device and other electronic equipment in the event of an overload while the device is in use. The maximum thresholds are 5.45 A, or 1,200 W. When the maximum threshold is detected to have been exceeded, the website interface displays an overload notification and a reset button, and the control button's status automatically switches to off mode. For critical events such as an overload, the system is designed to use real-time instantaneous logging so that the time of the event remains recorded with precision and fidelity.

```

535 /* Overload Alert Container Khusus */
536 isOverload && {
537   <div className="overload-alert animate-fade-in" style={{ background: 'linear-gradient(to right, rgba(239, 68, 68,
538     <div style={{ display: 'flex', alignItems: 'center', gap: '1rem' }}>
539       <div style={{ padding: '0.75rem', borderRadius: '50%', background: 'red', animation: 'spin 2s linear infi
540         <AlertTriangle color="white" />
541       </div>
542     </div>
543     <h3 style={{ margin: 0, color: 'red', fontSize: '1.25rem' }}>AMAS OVERLOAD LISTRIK! (>1200W)</h3>
544     <p style={{ margin: '0.25rem 0 0', color: 'red', fontSize: '0.9rem' }}>Sistem pintar kami mendeteksi a
545   </div>
546 </div>
547 <button
548   onClick={() => f

```

Figure 9. Website Code for Overload Condition

Figure 10 is a code excerpt from the cost-estimation feature. In this feature, the Smartura website contains a field for entering the cost of electricity consumption per kWh charged by PLN [the state electricity utility] based on the power tier installed at the household. The resulting cost figure is then displayed in the analysis column on the website. This calculation is obtained through measurement results from the voltage $\times$ current sensors—for example, a read voltage of 220 V and a total current across 3 channels of 1.5 A yield a total power reading of 330 W.

```

631 Pengaturan Konfigurasi Tarif */
632 v className="glass-panel animate-fade-in" style={{ padding: '2rem', animationDelay: '0.2s' }}>
633   <div style={{ display: 'flex', alignItems: 'center', gap: '0.75rem', marginBottom: '1.5rem' }}>
634     <Coins color="emerald-400" size={24} />
635     <h3 style={{ margin: 0, fontSize: '1.125rem' }}>Konfigurasi & Tarif</h3>
636   </div>
637
638   <div style={{ display: 'flex', flexDirection: 'column', gap: '1.25rem' }}>
639     <div style={{ display: 'flex', flexDirection: 'column', gap: '0.4rem' }}>
640       <label style={{ fontSize: '0.85rem', color: 'var(--text-secondary)' }}>Tarif listrik per kWh (Rp)</label>
641       <div style={{ display: 'flex', alignItems: 'center', gap: '0.5rem', background: 'var(--bg-tertiary)', padding: '0
642         <span style={{ paddingLeft: '0.5rem', color: 'var(--text-muted)', fontSize: '0.9rem' }}>Rp</span>
643         <input
644           type="number"
645           step="0.1"

```

Figure 10. Website Code for Cost Estimation

This cost-estimation feature is designed based on historical data collected at a 60-second interval. This interval was chosen with consideration for the storage capacity of the Firebase Realtime Database on the free-tier service, which provides 1 GB of storage and a monthly data-transfer limit of 10 GB. Recording historical data at a 60-second interval still adequately represents the precise power-consumption load and does not degrade the quality of user-habit analysis; furthermore, selecting a 60-second interval helps avoid burdening the database server, reduces Wi-Fi signal congestion on the ESP32, and still maintains accurate power calculations.

2.3 Casing Design

Casing design was carried out to determine the physical form, dimensions, component positions, and cable-routing paths so that the device could be used as a neat and safe smart electrical terminal. The resulting design resembles the shape of a typical conventional electrical terminal to facilitate ease of use, as shown in Figure 11. Based on the 3D design, the exterior of the casing consists of a hole for the green LED serving as the Wi-Fi indicator, 3 output channels for electrical devices, a power-cable port, and the Smartura logo. The design dimensions are $20 \times 12 \times 6$ cm, with empty internal space provided to separate the hardware components from the electrical-terminal output-channel path.

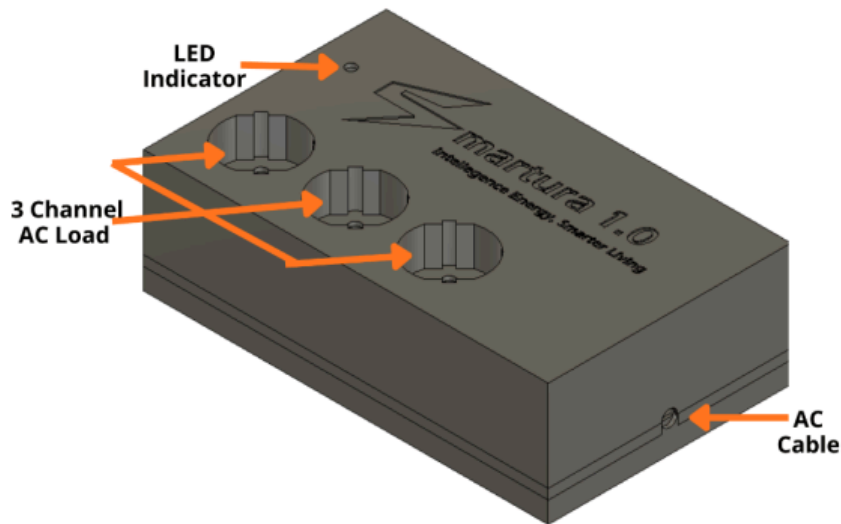


Figure 11. Casing Design

2.4 Performance Testing

This performance test was conducted to determine whether the Smartura system operates in accordance with its design. Testing was carried out by testing features and system functions such as relay control, sensor reading, cost estimation, timer, overload protection, and data history.

The Wi-Fi connection feature on the device serves to connect the ESP32 to the nearest Wi-Fi network so that the device can communicate with the database and the website via a Wi-Fi connection. Smartura leverages the advantage of the ESP32 microcontroller's built-in Wi-Fi Manager, which is a mechanism enabling the ESP32 to configure the Wi-Fi network dynamically, without needing to modify or re-upload the Arduino program code whenever the network to be used differs from the previous one. The Wi-Fi Manager works as follows: once the ESP32 is powered on, the device automatically creates a temporary local hotspot named SMARTURA_SETUP—this is referred to as Access Point mode. A web page then appears, directing the user to attempt to connect to an available SSID and enter its password.

A green LED indicator lamp is provided to show whether the Wi-Fi is already connected: the LED blinks continuously as long as the ESP32 has not connected to a nearby Wi-Fi network, but once the ESP32 has connected to Wi-Fi, the LED remains lit continuously for 3 minutes. If the connection is lost, such as during a power outage that disconnects the device from the internet network, the LED will blink until it can reconnect to the internet network. ESP32's Wi-Fi network connection code excerpt is shown in Figure 12.

```

313
314 // — 1. Pantau Koneksi Wi-Fi —————
315 bool wifiNow = (WiFi.status() == WL_CONNECTED);
316 if (wifiNow && !wifiOK) {
317     wifiOK = true; wifiTs = millis(); ledIjoOn = true;
318     digitalWrite(PIN_LED_IJO, HIGH); blinker.detach();
319     fbWrite(DEV "/status", "\"online\"", false);
320     Serial.println("Wi-Fi terhubung kembali.");
321 } else if (!wifiNow && wifiOK) {
322     wifiOK = false; blinker.attach(0.5, kedipIjo);
323     Serial.println("Wi-Fi terputus.");
324 }
325 if (ledIjoOn && millis() - wifiTs >= 180000) { digitalWrite(PIN_LED_IJO, LOW); ledIjoOn = false; }
326

```

Figure 12. ESP32 Wi-Fi Network Connection Code.

The relay-control feature is the main control system on the Smartura device used to issue on or off commands. This control button is placed on the web page already integrated with the device, and its mechanism uses the relay function to disconnect and connect the COM path to the relay so that it can be controlled virtually via the web. This relay-based on/off control feature will only function if the device is connected via Wi-Fi; when the connection is lost, the control system will not function, and the relay will forcibly disconnect the COM path to issue an off command to the device. Besides on/off control, the relay's main function is as a switch for connecting the electrical current to the Smartura device, for example, through the timer feature and the overload feature. The code excerpt for controlling the relay condition via the ESP32 is shown in Figure 13.

```

349 // --- 4. Kontrol Relay ---
350 if (isOverload) {
351     digitalWrite(PIN_RELAY, LOW); digitalWrite(PIN_LED_OREN, HIGH);
352 } else {
353     digitalWrite(PIN_LED_OREN, LOW);
354     bool target = relayState;
355
356     // Cek timer habis
357     if (timerActive && timerFinishMs > 0 && ntpOK() && epochMs() >= timerFinishMs) {
358         Serial.println("Timer habis! Relay OFF.");
359         timerActive = false; timerFinishMs = 0; timerDurMs = 0;
360         relayState = false; target = false;
361         if (wifiOK) {
362             fbWrite(DEV "/control", "{\\\"relay_state\\\":false,\\\"timer_duration_ms\\\":0}");
363             updateTimerNode();
364         }
365     }
366     if (timerActive && !ntpOK()) target = false; // Fail-safe: NTP belum sinkron
367     if (!wifiNow) target = false; // Fail-safe: Wi-Fi mati

```

Figure 13. ESP32 On/Off Control System Code

The processing of current- and voltage-sensor data is obtained from the current and voltage measured through the ACS712 and ZMPT101B sensors, which then send their data to ESP32. The sensor data stored on the ESP32 is then processed, and the processed results are sent to the Firebase database and displayed on the Smartura device's website monitoring dashboard. The code excerpt for reading and transmitting sensor data via the ESP32 is shown in Figure 14.

```

135 // PEMBACAAN SENSOR
136 // =====
137 float bacaArus() {
138     static int buf[400];
139     long sum = 0; int n = 0;
140     unsigned long t = millis();
141     while (millis() - t < 20 && n < 400) {
142         buf[n] = analogRead(PIN_ACS); sum += buf[n++]; delayMicroseconds(40);
143     }
144     if (!n) return 0.0;
145     float avg = (float)sum / n;
146     double sq = 0;
147     for (int i = 0; i < n; i++) { float d = buf[i] - avg; sq += d * d; }
148     float rms = sqrt(sq / n) * 3.3 / 4095.0 / ACS_SENS;
149     return (rms > ACS_NOISE) ? sqrtf(rms * rms - ACS_NOISE * ACS_NOISE) : 0.0;
150 }
151
152 float bacaTegangan() {
153     int mx = 0, mn = 4095; unsigned long t = millis();
154     while (millis() - t < 20) {
155         int v = analogRead(PIN_ZMPT);
156         if (v > mx) mx = v; if (v < mn) mn = v;
157     }

```

Figure 14. ESP32 Monitoring Display Code

The timer-control feature serves to set a time limit for how long the device will operate according to the user's wishes. In this timer control, the system only provides a countdown based on the specified limit; for example, if the user sets the timer active for a countdown of 4 hours, then 4 hours later the device will cease functioning by disconnecting the COM path on the relay. The code excerpt for the timer function on the ESP32 is shown in Figure 15.

```

236 // Timer baru dari web (timer_duration_ms > 0 = perintah baru)
237 if (durMs > 0 && !cancelTmr && ntpOK()) {
238     timerDurMs = durMs;
239     timerStartMs = epochMs();
240     timerFinishMs = timerStartMs + (uint64_t)durMs;
241     timerActive = true;
242     relayState = true;
243     Serial.printf("Timer %ld ms aktif (selesai: %ld s)\n", durMs, (long)(timerFinishMs/1000));
244     // Hapus perintah agar tidak re-trigger saat poll berikutnya
245     fbWrite(DEV "/control", "{\\\"timer_duration_ms\\\":0,\\\"timer_cancel\\\":false}");
246     updateTimerNode();
247     return; // Relay sudah ON dari timer, skip baca relay_state di bawah
248 }

```

Figure 15. ESP32 Timer-Control Code

The overload-protection feature serves to protect the device as well as the electronic equipment connected to the Smartura terminal. This system is built based on current-sensor thresholds and estimated power-usage limits. It works such that if the load usage exceeds the current-sensor threshold or the estimated-power threshold, the device will forcibly disconnect the relay's COM path

and the device will remain in an off state. The maximum thresholds set for this device are 5.45 A and 1,200 W; this is because the device uses an SRD 5 V 10 A relay module, with the calculation using the formula from Equation 1 and applying a tolerance margin of 50%.

This threshold is used to safeguard the wiring path, components, terminal, and load characteristics. The electronic components used are of small wattage scale (phone chargers, laptop chargers, water-pump chargers, fans, etc.) with specifications $\leq 1,000$ W. The code excerpt for overload detection and system reset is shown in Figure 16.

```

334
335 // — 3. Deteksi Overload —————
336 if (!isOverload && (daya > BATAS_DAYA || arusHalus > BATAS_ARUS)) {
337   Serial.printf("OVERLOAD! %.1f W / %.2f A → Relay OFF\n", daya, arusHalus);
338   isOverload = true; relayState = false;
339   timerActive = false; timerFinishMs = 0;
340   prefs.putBool("overload", true);
341   digitalWrite(PIN_RELAY, LOW); digitalWrite(PIN_LED_OREN, HIGH);
342   if (wifiOK) {
343     fbWrite(DEV "/control", "{\"relay_state\":false}");
344     fbWrite(DEV "/alerts", "{\"overload\":true}");
345     updateTimerNode();
346   }
347 }
348

```

Figure 16. ESP32 Overload-Detection Code

2.5 Data Analysis Techniques

The Smartura test-result data was analyzed using a quantitative descriptive analysis method. The analysis was performed to determine the characteristics of the sensor readings, the performance of the monitoring system, electrical-energy estimation, electricity-cost estimation, and the success of Smartura's main functions. The data analyzed includes the values from the voltage sensor, the current sensor, the estimated-power calculation, relay status, internet-network connection status, and the electricity-usage history stored in the Firebase Realtime Database (RTDB). The analysis results are then presented in the form of tables and graphs to illustrate changes in values under each condition during testing.

Sensor-reading analysis was conducted by observing the minimum, maximum, and average values of the voltage, current, and estimated-power parameters. The estimated-power value was calculated based on the voltage and current readings using Equation 1. In addition, sensor-reading validation was carried out by comparing the values read by Smartura against reference values from a measuring instrument. The difference between the Smartura value and the reference value was calculated using the percentage-error equation shown in Equation 2.

$$P = V \times I \quad (1)$$

Explanation:

P = power estimate (watts)

V = voltage (volts)

I = current (amperes)

$$\text{Error (\%)} = \left| \frac{\text{Smartura Value} - \text{Reference Value}}{\text{Reference Value}} \right| \times 100\% \quad (2)$$

In addition to using the error value for each test data point, this study also used the Mean Absolute Percentage Error (MAPE) method to determine the overall average percentage error of the sensor readings. The MAPE value was obtained by comparing Smartura's sensor-reading results against the actual value from the reference measuring instrument. The smaller the MAPE value, the better the sensor's accuracy level. In this study, MAPE was used to evaluate the validation results of the voltage and current sensors so that the overall sensor-reading error rate could be observed, rather than from a single test data point alone. The MAPE formula is shown in Equation 3.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{\text{Reference} - \text{Smartura}}{\text{Reference}} \right| \times 100\% \quad (3)$$

Analysis of energy and electricity-cost estimation was carried out based on historical power data stored in the system. Because historical data is stored periodically at a specific recording interval,

electrical energy is calculated from the power value and the recording time interval. Energy and electricity-cost estimation are calculated using Equations 4 and 5.

$$\text{Energy (kWh)} = \text{Power (W)} \times \text{interval_second} / 3.600.000 \quad (4)$$

$$\text{CostEstimate (Rp)} = \text{Power (kWh)} \times \text{Electricity Rate in kWh} \quad (5)$$

The analysis results are used to evaluate Smartura's ability to perform real-time electricity-energy monitoring, store historical data, calculate estimated energy usage, and provide electricity-usage cost information to users.

RESULTS AND DISCUSSION

3.1 Smartura Prototype

The Smartura prototype is the result of implementing a smart electrical terminal designed as a power-distribution medium with a load-control system, timer control, monitoring, and simple overload protection. The physical form of Smartura is shown in Figure 17. The prototype was built to resemble a typical conventional electrical terminal, with its use focused on simple households, so that its main components are housed within a single casing for a neater and more efficient design.

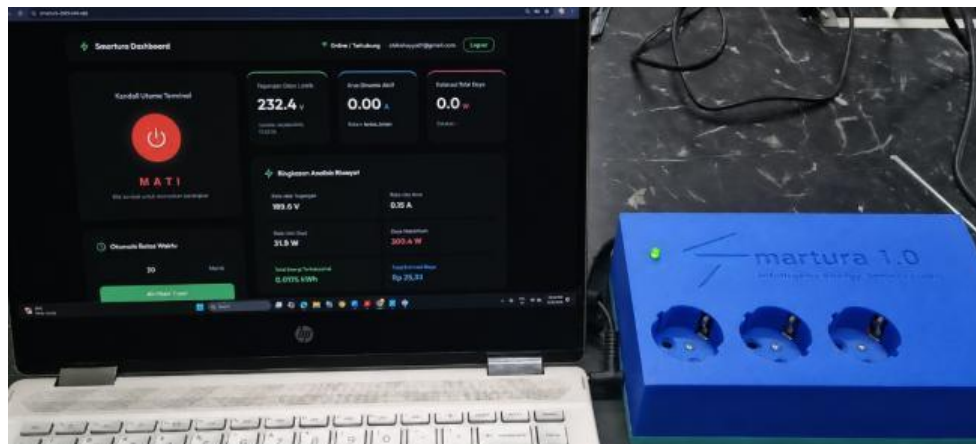


Figure 17. Smartura Prototype

Based on the implementation results, Smartura's main components are interconnected in accordance with the design. The ESP32 functions as the main controller for reading sensors, controlling the relay, running the timer, and sending data to Firebase. The ZMPT101B sensor is used to read voltage, the ACS712 sensor is used to read current, while the relay is used as an electronic switch to disconnect and connect the electrical flow to the load. The AC-DC power supply is used to provide power to the control circuit. Table 5 shows the general specifications of the Smartura electrical terminal.

Table 5. Smartura Spesification

Spesification	Detail
Device type	IoT-based electrical terminal
Dimensions	• Length: 20 cm; • Width: 12 cm; • Height: 6 cm
Weight	± 900 g
Casing material	PLA+ filament
Microcontroller	ESP32-Wroom
Actuator	5 V relay
Sensors	ZMPT101B and ACS712
Input power	AC 100–240 V, 50 Hz
Communication medium	Wi-Fi 2.4 GHz

Component selection also considered the requirements for use in simple households. In a household installation with 1,300 W of power, the operating current is in the range of 5–6 A when calculated using an apparent-current approximation at approximately 220 V. Therefore, the use of a relay with a maximum contact capacity of 10 A was deemed sufficient for the prototype's needs, since the component's capacity exceeds this operating current. Although the relay used has a maximum specification of 10 A, or approximately 2,200 W at 220 V, this study set a protection threshold of 1,200 W, or approximately 5.45 A, as the system's operational limit. This threshold was set to provide a safety margin, so that none of Smartura's components—such as the relay, current sensor, PCB traces, connectors, power supply, and internal wiring—operate near their maximum capacity. In addition to improving system reliability, this operational limit also helps reduce the risk of temperature rise resulting from high loading and provides additional protection for connected electronic equipment. This limitation also reflects a degree of caution, given that the device remains at the prototype stage.

Compared with commercial smart electrical terminals generally already available on the market, Smartura has not yet been positioned as a market-ready final product. Smartura is not intended to compete directly with commercial smart plugs but rather to serve as an IoT platform prototype that can be further developed for research, educational, or user-specific implementation purposes. The primary value of this research lies in the process of system design and implementation, namely how the components used can operate in accordance with their designed functions. With its open architecture, Smartura also offers more flexible development opportunities, since the system can be modified as needed, for example, by adding control and monitoring for each individual channel, improving sensor accuracy, and conducting longer-duration durability testing.

3.2 Wi-Fi Connection Implementation on the ESP32

Smartura uses a Wi-Fi connection as the main communication medium between the ESP32 hardware and the Firebase Realtime Database. Wi-Fi was selected based on the availability of a built-in 2.4 GHz Wi-Fi module on the ESP32 microcontroller, eliminating the need for an additional external communication module. Furthermore, a Wi-Fi network is the most commonly available infrastructure in households, so users do not need to provide additional equipment to operate Smartura.

The Wi-Fi network-configuration process on Smartura uses the Wi-Fi Manager library. The main advantage of the Wi-Fi Manager compared with the conventional (hard-coded) configuration method is that users do not need to modify or re-upload the program code in the Arduino IDE each time they want to change the Wi-Fi network. The entire configuration process can be carried out directly through a browser on a smartphone or laptop without requiring programming knowledge. Consequently, Smartura can be relocated and connected to a different Wi-Fi network with flexibility.

Connecting Smartura to a Wi-Fi network is accomplished through several steps. First, when Smartura is powered on for the first time, or when it does not yet have stored Wi-Fi credentials, the ESP32 automatically creates an Access Point mode with the network name “SMARTURA_SETUP,” which can be seen in the list of Wi-Fi networks on the user's device, as shown in Figure 18.

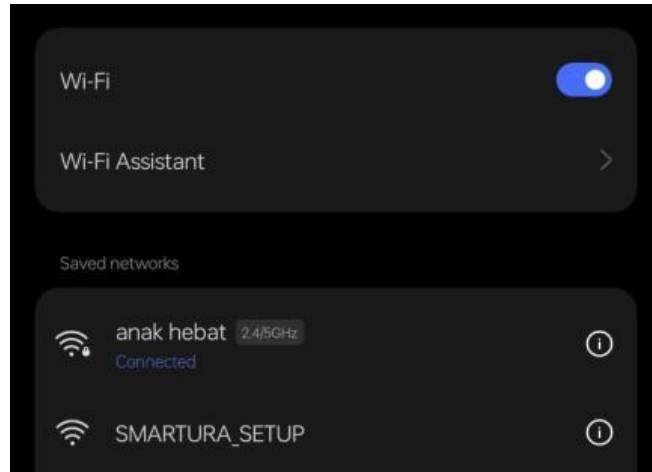


Figure 18. SMARTURA_SETUP Hotspot Network Displayed in The Wi-Fi list

Once the user connects to the SMARTURA_SETUP network, they are automatically directed to open the Wi-Fi Manager portal page at address 192.168.4.1. On this portal page there is a “Configure WiFi” button to begin the network configuration, as shown in Figure 19.



Figure 19. Wi-Fi Manager Portal Page

When the “Configure WiFi” button is pressed, the system displays the list of Wi-Fi networks available in the vicinity of the device. The user selects the desired network, then enters the SSID and password in the fields provided and presses the “Save” button to store the credentials, as shown in Figure 20. Once the connect button is selected, the Wi-Fi Manager attempts to connect the ESP32 to that network. If the authentication process succeeds, the SSID and password information is stored in the ESP32's internal memory, after which the device automatically restarts and connects to the newly configured Wi-Fi network.

The screenshot shows a web browser interface for configuring a Smartura device. At the top, a black header bar contains the text "Sign in to SMARTURA_SETUP" and the IP address "192.168.4.1". Below the header, the username "anak hebat" is displayed next to a lock icon and a signal strength indicator. The main form area includes two input fields: "SSID" with the value "anak hebat" and "Password" with masked characters "*****". Below the password field is a checkbox labeled "Show Password". At the bottom of the form are two large blue buttons labeled "Save" and "Refresh".

Figure 20. Wi-Fi SSID and Password Configuration

After the credentials are saved, ESP32 attempts to connect to the entered Wi-Fi network. This process is indicated by the message “Saving Credentials — Trying to connect ESP to network” on the Wi-Fi Manager page, as shown in Figure 21. If the connection succeeds, the SMARTURA_SETUP access point automatically closes and the ESP32 switches to normal operational mode.



Figure 21. Credential-Saving Process and Connection Attempt

Once the Wi-Fi connection is successfully established, the device's green LED lights up for 3 minutes as a visual indicator that Smartura has successfully connected to the internet network. The ESP32 then performs a time synchronization via NTP (Network Time Protocol) and sends an “online” status to the Firebase Realtime Database, so that the website dashboard can display the device's connection status in real time. The entered Wi-Fi address is stored in the ESP32's internal memory, so that on subsequent power-ups Smartura will connect automatically without requiring reconfiguration.

If the Wi-Fi connection is lost during operation, the device's green LED will blink as an indicator that the connection is currently unavailable. When the Wi-Fi network becomes available again, the ESP32 automatically reconnects and sends an “online” status to Firebase. However, if at some point the Wi-Fi network changes or is no longer available, the ESP32 will re-enable Access Point mode so that the user can perform a reconfiguration through the same web page. Through this mechanism, the Wi-Fi Manager provides a high degree of flexibility, since the device can be used at various locations with different Wi-Fi networks without requiring any changes to the program.

3.3 Website and Firebase Implementation

The Smartura website is used as the user's main interface for operating the system. This website includes a dashboard page that functions to display device status, the on/off control button, timer control, sensor monitoring data, and system notifications. Implementation of this website is a critical part of the system, since all control commands on Smartura are issued through the dashboard and then sent to the Firebase Realtime Database.

A website was chosen as the user interface because it is more flexible and accessible across devices. Smartura users are not limited to Android users but can also use an iPhone, laptop, tablet, or other device, provided they have a browser and an internet connection. With this approach, users do not need to install a dedicated application, making the system easier for the public to use. In addition, website feature updates can be carried out on the system side without needing to build a different application version for each operating system. The system-usage flow is designed in three main steps:

- Real-time monitoring: After successfully logging in, users can view the real-time condition of the electrical terminal on the dashboard, covering the connection status (whether on/off), voltage-value information, information on the current flowing, information on the current estimated power consumption, and information on total energy and cost.
- Load and timer control: Users can press the main control button on the dashboard to instantly disconnect or connect the electrical current via the web. If a user wishes to limit the duration of load usage (for example, charging a laptop for 80 minutes), the user simply enters the duration figure in the timer field and presses the “Activate Timer” button. The system will then display a countdown and automatically switch off the terminal once the time has elapsed.
- Overload-warning handling: If the connected load exceeds the threshold—that is, a current value >5.45 A and a power value $>1,200$ W—the system automatically switches off the terminal and displays a warning notification on the dashboard. After disconnecting the electronic load from the Smartura terminal, the user can press the “System Confirmed Safe (Reset)” button on the dashboard to return the device to its normal operational condition.

Based on the implementation results, the website and Firebase have proven capable of serving as the connector between the user and the Smartura device. When a user issues a command through the website, the command data changes within Firebase. This data change is then read by the ESP32 to execute the corresponding function, such as activating the relay, switching off the relay, running the timer, or disconnecting the load in the event of an overload. Conversely, data from the ESP32 is also sent to Firebase and displayed back on the dashboard. The Smartura dashboard display is shown in Figures 22 and 23.

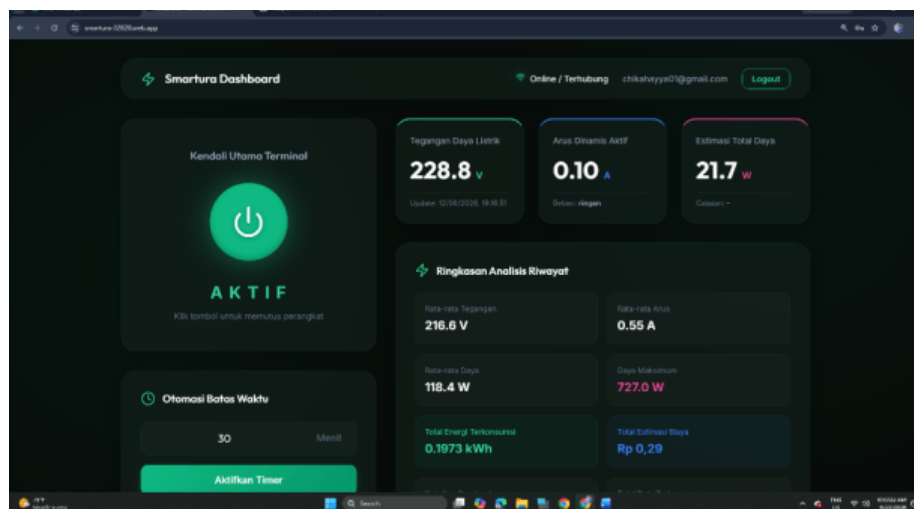


Figure 22. Smartura Dashboard — Numerical Monitoring View

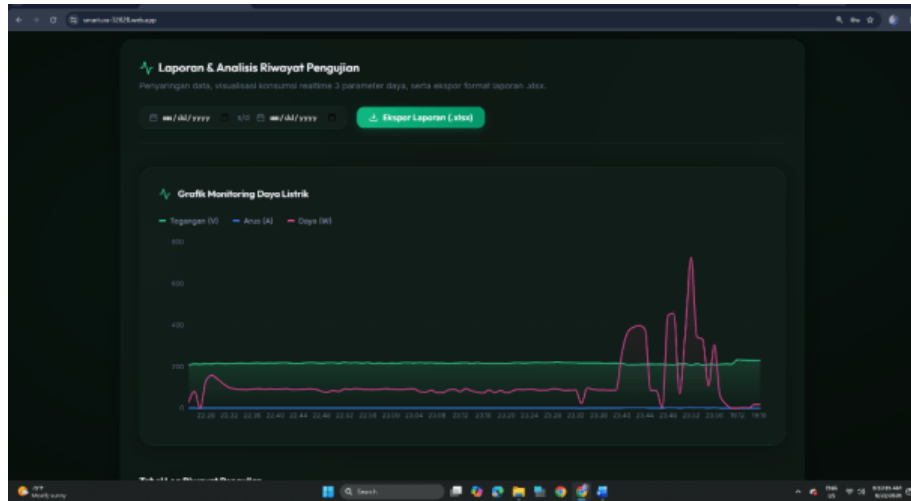


Figure 23. Smartura Dashboard — Graphical Monitoring View

3.4 Voltage, Current, and Estimated-Power Monitoring Test Results

Monitoring tests were conducted to determine Smartura's ability to read and display the electrical parameters of voltage, current, and estimated power. Data collection was performed from 5 June 2026 to 11 June 2026, yielding a total of 6,193 data points. The voltage parameter was read using the ZMPT101B sensor, the current parameter was read using the ACS712 sensor, while the estimated power was obtained from the calculation of voltage and current processed by the ESP32. The resulting reading data was then sent to Firebase and displayed on the Smartura website dashboard.

This test also demonstrates that Smartura was used continuously throughout the data-collection period, with the device connected to AC electricity. The electronic devices used during testing included a table fan, LED lamp, laptop charger, phone charger, water-pump charger, iron, and rice cooker. This one-week test served as an initial functional test to observe the system's ability to perform continuous monitoring and data logging. However, this test cannot yet be used as a basis for asserting the device's long-term durability or service life. The data-collection results are shown in Table 6.

Table 6. Data-Collection Results

Parameter		Result
Data period	5 June 2026 – 11 June 2026	
Number of data points	6,193	
Total energy (kWh)	3.62	Based on the results, average value
Corrected estimated cost (Rp)	5,229.29	
Average voltage (V)	213.15	
Average current (A)	0.16	
Average power (W)	35.07	
Overload events detected (data points)	0.00	the voltage over the
Note	Cost calculated using a tariff of Rp1,444.70/kWh	

observation period was 213.15 V. The minimum voltage value was recorded at 200.63 V on 7 June 2026, while the maximum voltage value was recorded at 235.10 V on 8 June 2026. In general, the daily average voltage values ranged from 210.90 V to 215.97 V. This shows that the Smartura system was able to record changes in the voltage value throughout the testing process.

The current readings show an average value of 0.164 A. The daily current value for most of the testing period ranged, on average, from 0.149 A to 0.179 A. The highest current value was recorded on 11 June 2026 at 3.496 A, with a daily average of 0.585 A. The increase in current on that date confirms greater load usage compared with other test days. This condition also affected the estimated-power value, since power is calculated based on the voltage and current readings.

The average estimated power over the test period was 35.07 W. From 5 to 10 June 2026, the daily average power ranged from 31.50 W to 38.00 W. The highest maximum power value occurred on 11 June 2026 at 727.03 W, with a daily average power of 125.21 W. Based on these results, Smartura is already able to display changes in the voltage, current, and estimated-power values in

accordance with the load condition used. The minimum, maximum, and average voltage, current, and estimated-power values during the test period are shown in Table 7.

Table 7. Data-Monitoring Results

Date	Min (V)	Max (V)	Avg (V)	Min (A)	Max (A)	Avg (A)	Min (W)	Max (W)	Avg (W)
2026-06-05	200.95	225.24	210.90	0.00	0.23	0.15	0.00	46.34	31.50
2026-06-06	204.45	219.54	212.47	0.00	0.36	0.18	0.00	77.89	38.00
2026-06-07	200.63	232.41	211.99	0.00	0.40	0.15	0.00	84.45	32.03
2026-06-08	203.04	235.10	213.83	0.00	0.60	0.17	0.00	109.54	35.56
2026-06-09	205.13	223.77	214.52	0.00	0.18	0.16	0.00	38.99	33.74
2026-06-10	203.45	223.26	213.06	0.00	0.18	0.15	0.00	38.84	32.61
2026-06-11	207.95	222.63	215.97	0.00	3.50	0.58	0.00	727.03	125.21

Based on the test results, the voltage value read by Smartura changed on each observation date. This change shows that the system can record the variation in the incoming electrical voltage. The current value also changed in accordance with the load condition used. When the load was small or inactive, the current value tended to be low, whereas when the load was active, the current value increased. In Table 5, the daily minimum average current value is 0.00; this occurred because Smartura was not connected to a load or electronic device, so the displayed value was 0.00. The estimated power was obtained from the calculation between voltage and current, so the power value likewise changed according to the magnitude of the load connected to the electrical terminal.

3.5 Validation of Voltage and Current Sensor Data Readings

The sensor-data-reading validation test compares Smartura's sensor measurements with a multimeter and calculates the error using the formula in Equation 2 and then analyzes the resulting accuracy percentage. The comparison results between Smartura's sensor readings and the multimeter are shown in Table 8.

Table 8. Sensor Testing and Validation Results

Load Type	Smartura (V)	Multimeter (V)	Error (V)	Smartura (A)	Multimeter (A)	Error (A)
Table fan	218	222	1.80%	0.10	0.14	28.57%
Iron	214	219	2.28%	1.63	1.57	3.82%
Soldering iron	230	232	0.86%	0.20	0.19	5.26%
Washing machine	217	221	1.81%	0.70	0.75	6.67%
Television	213	218	2.29%	0.26	0.24	8.33%
Rice cooker	217	220	1.36%	1.40	1.37	2.19%
Phone charger	216	221	2.26%	0.03	0.05	40.00%
Laptop charger	218	220	0.91%	0.25	0.24	4.17%

Once the error values for each test had been obtained, the MAPE value was calculated to determine the overall average percentage error of the sensor readings. The MAPE calculation was performed by summing all the error values and then dividing by the number of test data points, as shown in Equation 3. The MAPE value is used to assess Smartura's sensor accuracy relative to the reference measuring instruments, namely a multimeter for the voltage sensor and a clamp meter for the current sensor.

Sensor validation was conducted under controlled indoor conditions at a constant room temperature, with the Smartura device connected to a stable 220 V AC PLN supply. Each load type was measured three consecutive times, and the average reading was used as the representative Smartura value. The reference measuring instrument used was an ANENG ST182 digital clamp meter and multimeter, featuring a specified accuracy of $\pm(1.5\% + 3)$ for AC voltage measurements and $\pm(2.5\% + 3)$ for AC current measurements. The MAPE classification used in this study follows the evaluation criteria proposed by Lewis (1982) as cited in Klimberg et al., (2010), which is widely adopted to evaluate instrument and system accuracy: MAPE < 10% indicates high accuracy (very good), 10% – 20% indicates good accuracy, 20% – 50% represents reasonable accuracy, and > 50% represents inaccurate performance.

The relatively high current-sensing error observed for low-current loads 28.57% for the table fan (0.14 A) and 40.00% for the phone charger (0.05 A) is attributable to the inherent sensitivity limitation of the ACS712 sensor at currents below 0.1 A. The ACS712-05B module has a nominal

sensitivity of 185 mV/A. When combined with the ESP32's 12-bit ADC resolution over a 3.3 V measurement range (approximately 0.8 mV per ADC step, equivalent to ~4.3 mA per step), even a single ADC step fluctuation introduces a substantial percentage error when the measured current is very small. This limitation aligns with the findings reported by Pardede & Brahmana (2023) and Wibowo et al., (2024). Consequently, the current sensor in Smartura is more suitable for monitoring electrical loads with currents ≥ 0.2 A, while caution is advised when interpreting readings for low-power standby devices.

Based on Table 7, the voltage error values obtained were 1.80%, 2.28%, 0.86%, 1.81%, 2.29%, 1.36%, 2.26%, and 0.91%. This yields a voltage-sensor MAPE value of 1.69%, indicating that the voltage-sensor readings have a small error rate, so that the voltage-measurement results can be considered highly accurate.

For the current sensor, the error values obtained were 28.57%, 3.82%, 5.26%, 6.67%, 8.33%, 2.19%, 40.00%, and 4.17%. This yields a current-sensor MAPE value of 12.38%, indicating that the current-sensor readings fall within the good category. However, the error values for several small loads, such as the table fan and phone charger, remain quite large. This occurred because the current values read fall within a small range, so that even a slight change can produce a large percentage error. The table results show that the voltage-sensor readings exhibit a good level of accuracy, evidenced by the relatively small voltage error for each load, with a MAPE value of 1.69%. This value indicates that Smartura's voltage sensor can read voltage with reasonable stability and closely approximates the multimeter's measurement results.

In the current-sensor test, the resulting MAPE value was 12.38%. This result indicates that the current sensor's performance remains good, but its error rate is higher than that of the voltage sensor. A large error is observed for loads with small currents, such as the table fan and phone charger. This is because the ACS712 sensor is less sensitive for measuring small currents below 0.1 A. In addition, the Arduino IDE program was also set such that current readings below 0.10 A are treated as noise, which can affect the sensor's reading results. The ESP32's ADC resolution can also affect the reading results, particularly when the measured current is very small.

3.6 Energy- and Electricity-Cost-Consumption Estimation Results

The Smartura system records historical data periodically every 60 seconds. This interval was selected considering data-storage efficiency, bandwidth usage, and the energy-consumption monitoring needs at the household scale. Based on the data structure used, each historical data point has a size of approximately 350 bytes, covering information on voltage, current, estimated power, relay status, overload status, Wi-Fi status, timer status, load status, time stamp, and other supporting information. With a Firebase Realtime Database storage capacity of 1 GB under the Spark Plan service, the system is estimated to be able to store approximately 2.8 million historical data points, equivalent to more than five years of continuous logging using a 60-second interval. This interval selection considered data-storage efficiency, bandwidth usage, and the characteristics of electrical-load changes in household use, while still producing historical data representative enough for the energy-estimation process.

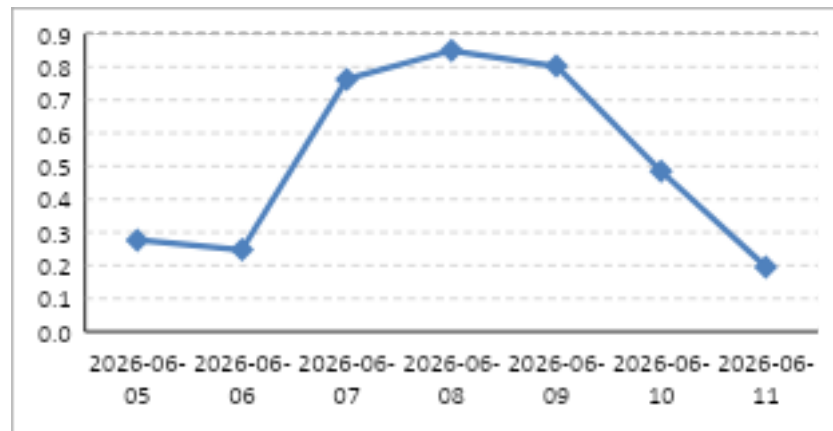
Energy- and electricity-cost-consumption estimation is carried out based on the historical power data stored in the Smartura system, which is derived from the voltage, current, and estimated-power readings sent to Firebase at a 60-second interval during the test period. The energy calculation is performed by converting the power value and the data-recording interval into kWh units, while the cost estimation is obtained by multiplying the total energy by the electricity tariff per kWh.

The electrical-energy calculation is performed using Equation 4, namely by multiplying the measured power value by the total measurement time (number of data points $\times$ data-collection interval), then dividing by 3,600,000 to obtain the energy in kWh units. The electricity-consumption cost estimate is then calculated using Equation 5, namely by multiplying the resulting electrical energy by the base electricity tariff of Rp1,444.70/kWh. For example, on 5 June 2026, an average power of 31.5 W was obtained from 527 measurement data points at a 60-second interval, yielding an energy of 0.27738 kWh, rounded to 0.3 kWh. This energy was then multiplied by the electricity tariff, resulting in an estimated cost of Rp399.7. The same calculation process was applied to all test data, yielding the estimated energy and cost values shown in Table 9.

Table 9. Energy- and Electricity-Consumption-Cost Estimation Results

Date	Data Count	Avg Volt	Avg Arus	Avg Power	Total Energy (kWh)	Estimated Cost (Rp)	Max Power
2026-06-05	527	210.9	0.1	31.5	0.3	399.7	46.3
2026-06-06	391	212.5	0.2	38.0	0.2	357.7	77.9
2026-06-07	1429	212.0	0.2	32.0	0.8	1102.0	84.5
2026-06-08	1433	213.8	0.2	35.6	0.8	1226.8	129.5
2026-06-09	1427	214.5	0.2	33.7	0.8	1159.3	39.0
2026-06-10	892	213.1	0.2	32.6	0.5	700.4	38.8
2026-06-11	94	216.0	0.6	125.2	0.2	283.4	727.0

Based on the processed data, the total estimated energy accumulated over the period from 5 June 2026 to 11 June 2026 was 3.62 kWh. With an electricity tariff of Rp1,444.70/kWh, the resulting estimated electricity-consumption cost was Rp5,229.29. The average daily estimated energy over the test period was 0.52 kWh, while the average daily estimated cost was Rp747.04. These values show that Smartura can provide simple electricity-usage and cost information based on the monitored data stored. The daily change in estimated energy over the test period is shown in Graph 1.



Graph 1. Energy-Consumption Graph (kWh)

Based on the daily data graph based on Graph 1, the highest estimated energy (kWh) occurred on 8 June 2026 at 0.84 kWh, with an estimated cost of Rp1,226.80. This value is higher than on other days because a relatively large number of data points were recorded and load usage occurred over a longer duration. The lowest estimated energy occurred on 11 June 2026 at 0.20 kWh, with an estimated cost of Rp283.40. Although the highest maximum power of 727.03 W occurred on that date, the total energy remained low because the data-recording duration was only about 1.55 hours. However, the displayed energy and cost values remain estimates, since Smartura's power calculation does not yet account for the power factor. Accordingly, these estimation results can provide monitoring information on electricity usage at the Smartura electrical terminal and are not intended as a replacement for the kWh-meter measurements present in the household.

3.7 Relay Control, Timer, and Overload Protection Testing

Relay-control testing was conducted to determine the success of the Smartura website in issuing on and off control commands to the electrical terminal. Testing was carried out through the dashboard page by pressing the control button available on the website. The command from the website is sent to the Firebase Realtime Database, which is then read by ESP32 to set the relay condition. Based on the test results, the relay can respond to the on and off commands issued via the website. When the on button is activated, the relay enters an active state so that the electrical load can turn on. Conversely, when the off button is pressed, the relay disconnects the electrical flow so that the load enters an off state. These results indicate that the remote-control system on Smartura already functions properly through the website. Based on the test data, the relay status was recorded as "on" for 5,785 data points and "off" for 408 data points out of a total of 6,193 data points. This data shows that changes in the relay status can be stored by the system and monitored through the website. Figure 24 displays the website's condition when the device is switched off. When the device is in the off

state, the control-button icon is red, whereas when the device is in the on state, the control-button icon on the website is green. When the device is off, it can clearly be seen that the voltage sensor remains active and continues to receive data; this is because the voltage-sensor component path is not directly connected to the relay, so that the voltage monitoring display continues to show the sensor's readings.

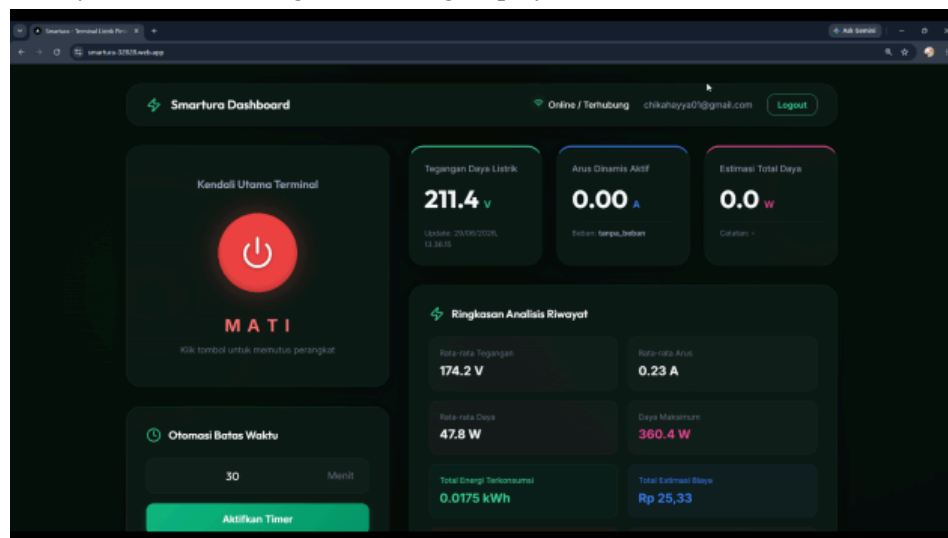


Figure 24. Relay Off Condition

Figure 25 shows the condition of the website when the device is actively in use, clearly indicated by the green color of the control button. This on/off control system has been tested and successfully executed, the response time with the device was also successful, and the remote-control method likewise worked successfully, since the device was connected to a home/nearby Wi-Fi network. The dashboard display when the relay is in active condition is shown in Figure 25.

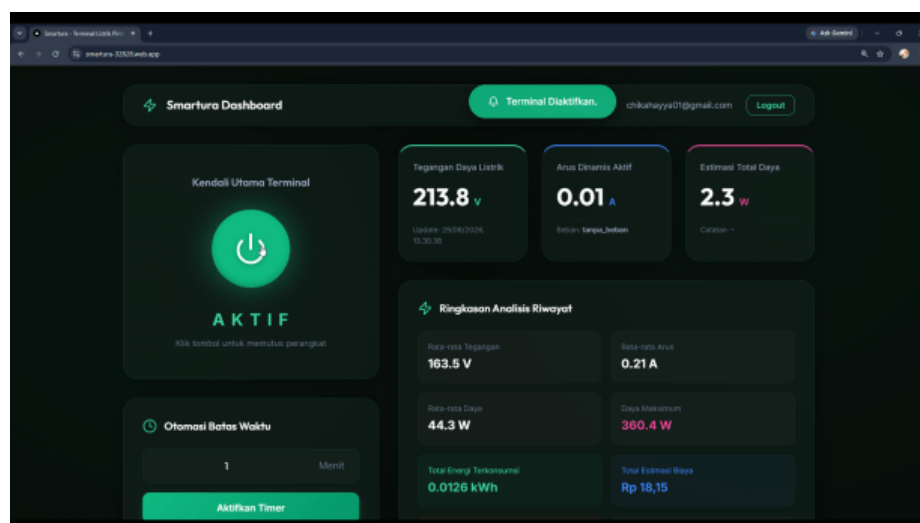


Figure 25. Relay On Condition

The timer test shown in Figure 26 was carried out to determine Smartura's ability to execute load control based on a time limit set by the user. In the functional test through the website, the timer was successfully executed according to the specified duration of 1 minute, or 60 seconds. Once the specified time had elapsed, the relay changed to the off condition, so that the electrical flow to the load was automatically disconnected. This feature is highly useful when a user wishes to limit the usage time of an electronic device for greater control.

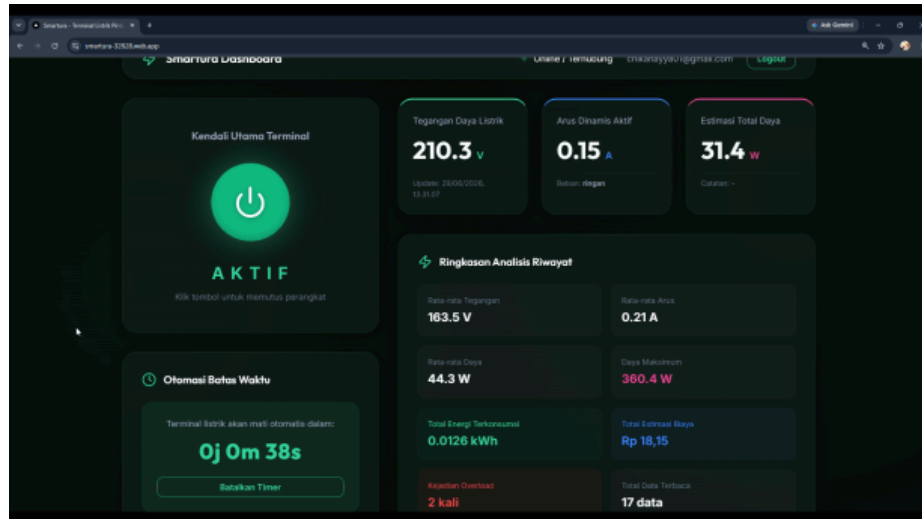


Figure 26. Timer Active Condition

The overload-protection test was conducted to determine the system's response when the load exceeds the specified safety threshold. On Smartura, the protection threshold is set at 1,200 W, or approximately 5.45 A. This threshold was selected considering a standard 1,300 W household power tier with an operating current of approximately 5.9 A at 220 V voltage. The SRD-05VDC-SL-C relay module has a maximum rated contact capacity of 10 A 2,200 W at 220 V. A safety margin of 50% was applied to set the operational limit at 1,200 W to protect PCB traces, terminal wiring, and internal components from thermal stress and continuous high-current operations.

During the continuous 6-day historical monitoring period (5–10 June 2026), 0 overload events were recorded because the maximum load used reached 727.03 W, remaining below the 1,200 W limit. To evaluate the functional response of the overload protection algorithm without subjecting the laboratory electrical infrastructure or prototype wiring to excessive high-current hazards, a functional demonstration test was conducted on 11 June 2026 by temporarily configuring a lower software threshold of 3.30 A / 630 W in the ESP32 firmware. When connected loads (rice cooker, iron, and table fan) exceeded 630 W, the ESP32 successfully detected the overload condition within less than 1 second, automatically tripped the relay to disconnect power, sent an alert to Firebase, and displayed a warning notification with a manual reset button on the web dashboard (Figure 27). This experiment provides a valid proof-of-concept demonstration for the software tripping algorithm and real-time alert system, although full-scale testing at the 1,200 W threshold remains to be verified under industrial laboratory testing conditions. Smartura have been successfully implemented as the main functions of the website-based load-control system. The dashboard display when the overload-protection feature is active is shown in Figure 27.

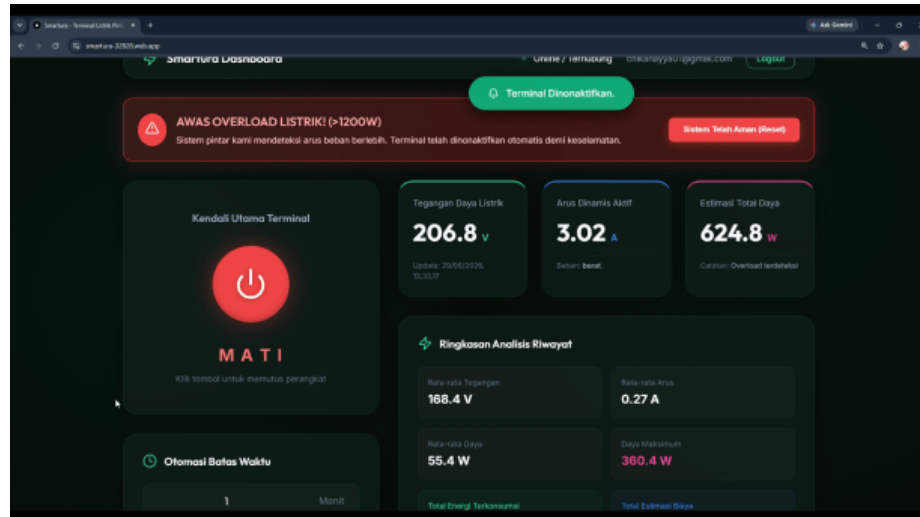


Figure 27. Overload Condition

Figure 27 is the result of an overload test that was not conducted using the actual threshold of 5.45 A or 1,200 W, since a larger load was needed for the test. The test was instead carried out by modifying the Arduino IDE code prior to compiling it onto the ESP32, using a threshold of 3.30 A and 630 W. After an overload occurs, the user is expected to press the reset button to return the system to its normal condition following the overload event. This system is designed to prevent the relay from automatically reactivating, thereby improving the safety and operational stability of the Smartura system.

Based on the results of relay-control, timer, and overload-protection tests, Smartura has demonstrated initial suitability as a smart electrical terminal prototype for simple household use. Testing conducted over one week showed that the system was able to remain connected to AC electricity, read sensor data, transmit data to Firebase, and execute relay control via the dashboard. However, this test still constitutes an initial functional test, so it cannot yet be used to definitively state the device's service life.

An estimate of Smartura's operational lifespan can be derived from the specifications and datasheets of its main components, particularly the relay actuator and the power supply module. The SRD-05VDC-SL-C relay specifies a rated electrical endurance of $\geq 100,000$ operations under full electrical load and a mechanical endurance of $\geq 10,000,000$ operations. The Hi-Link HLK-10M05 AC-DC power supply module has a Mean Time Between Failures (MTBF) exceeding 100,000 hours (sim 11.4 years of continuous operation) under standard operating conditions. Assuming a light-to-moderate residential usage profile with an average of 20 to 50 relay switching operations per day, the relay's electrical endurance theoretically supports several years of reliable switching. Accounting for thermal dissipation within the casing and component aging, a conservative theoretical lifespan estimate of 3–5 years is calculated.

The present study did not perform long-term durability or endurance testing. Therefore, the operational lifespan of the Smartura prototype cannot be determined from the experimental results. The one-week functional testing period demonstrates the operational functionality of the system but is insufficient to estimate long-term service life. In addition, the actual lifetime of the prototype may depend on relay switching frequency, load characteristics, operating temperature, power-supply conditions, wiring quality, and other environmental factors. Future studies should conduct endurance testing under representative load and switching conditions to quantitatively evaluate the reliability and service life of the system.

CONCLUSION

This research has successfully designed and implemented Smartura as an Internet of Things-based smart electrical terminal integrated with an ESP32, Firebase, and a website, such that all of the research objectives were achieved. The on/off control system was successfully operated remotely through the dashboard without requiring direct physical interaction with the device, as evidenced by the recording of 5,785 relay-on status data points and 408 relay-off status data points out of a total of 6,193 data points collected from 5–11 June 2026. Real-time monitoring of voltage, current, and estimated power was also successfully executed using the ZMPT101B and ACS712 sensors, with an accuracy level validated through a voltage-sensor MAPE value of 1.69% (very good) and a current-sensor MAPE value of 12.38% (good) and is therefore deemed suitable for basic electrical-parameter monitoring needs at the prototype scale.

The timer feature for automatically limiting load-usage duration was successfully executed in accordance with the duration specified by the user, with overload tripping successfully demonstrated using a safe functional test threshold of 3.30 A / 630 W. The system also successfully calculated and displayed total energy and estimated electricity cost based on the historical data stored in Firebase, with average results of 213.15 V voltage, 0.164 A current, 35.07 W power, 3.62 kWh total energy, and an estimated cost of Rp5,229.29 over the data-collection period. With these achievements, Smartura has successfully combined the functions of remote load control, basic electrical-parameter monitoring, usage-time limitation, automatic overload protection, and energy- and electricity-cost estimation within a single integrated system.

However, long-term durability and service life were not determined in this study because endurance testing was not performed. Future work should focus on repeated sensor validation, improved low-current measurement, direct testing at the nominal overload threshold under controlled safety conditions, and long-term endurance testing.

REFERENCES

- Afandi R, Satria B, Lubis SA. (2025). Instal: Jurnal Komputer Analysis of IoT Design for Monitoring Electrical Energy Use in Homes. Instal: Jurnal Komputer. 17(02):72–85. doi:<https://doi.org/10.54209/jurnalinstall.v17i02.361>.
- Akbar, M. F., Wilantara, P., Ikhsan, M., Ikhtiarta, H., Siskandar, R., Novianty, I., & Irzaman. (2019). The assembling of electrical socket for electricity usage monitor and electronic device control with ESP8266 microcontroller basis. In AIP Conference Proceedings (Vol. 2169). American Institute of Physics Inc. doi:10.1063/1.5132652
- Allegro MicroSystems. (2017). ACS712 fully integrated, hall-effect-based linear current sensor IC datasheet (Doc. No. ACS712-DS). Worcester, MA: Allegro MicroSystems, LLC. Diperoleh dari <https://www.allegromicro.com>
- Artono B, Winarno B, Hidayatullah NA. (2019). Rancang Bangun Smart Plug Untuk Sistem Monitoring Dan Proteksi Hubungsingkat Listrik. Jurnal ELTIKOM. 3(2):77–84. doi:10.31961/eltikom.v3i2.123.
- Baharudin A, Wijaya TK, Algusri M. (2025). Implementasi Sistem Pembatas dan Kontrol Beban Listrik Jarak Jauh Berbasis IoT. Sigma Teknika. 8(1):70–75.
- Christopoulos M, Spantideas S, Giannopoulos A, Trakadas P. (2024). Deep Reinforcement Learning for Smart Home Temperature Comfort in IoT-Edge Computing Systems. In: Proceedings of the 1st International Workshop on MetaOS for the Cloud-Edge-IoT Continuum. New York, NY, USA: Association for Computing Machinery. pp 1–7.
- Dahroni A, Pratama MF, Putra E. (2020). Pembuatan Prototype Smart Power Strip Berbasis Mobile. KILAT. 9(1):18–27. doi:10.33322/kilat.v9i1.752.
- Dhananjaya NAAG, Made I, Suarjaya AD, Putu I, Almantara S. (2025). Smart Plug Socket with ESP-NOW and MQTT for Efficient Power Monitoring and Control via Android. Volume 6.

- El-Khozondar HJ, Mtair SY, Qoffa KO, Qasem OI, Munyarawi AH, Nassar YF, Bayoumi EHE, Halim AAEBA El. (2024). A smart energy monitoring system using ESP32 microcontroller. *e-Prime - Advances in Electrical Engineering, Electronics and Energy*. 9. doi:10.1016/j.prime.2024.100666.
- Elorbany K, Bayılmış C, Balta S. (2021). A Smart Socket Equipped With IoT Technologies for Energy Management of Electrical Appliances. *Sakarya University Journal of Computer and Information Sciences*. 4(3). doi:10.35377/saucis.04.03.863272.
- Farghali M, Osman AI, Chen Z, Abdelhaleem A, Ihara I, Mohamed IMA, Yap PS, Rooney DW. (2023). Social, environmental, and economic consequences of integrating renewable energies in the electricity sector: a review. *Environ Chem Lett*. 21(3):1381–1418. doi:10.1007/s10311-023-01587-1.
- Fikhri AA, Ula M, Sayuti M, Taufiq, Nurdin. (2025). Perbandingan Kinerja Protokol MQTT dan HTTP Dalam Komunikasi Data Internet of Things Artikel Penelitian. *Jurnal Infomedia: Teknik Informatika, Multimedia, dan Jaringan*. 10(1):2549–1180.
- Hargreaves T, Middlemiss L. (2020). The importance of social relations in shaping energy demand. *Nat Energy*. 5(3):195–201. doi:10.1038/s41560-020-0553-5.
- Hasan MK, Ahmed MM, Pandey B, Gohel H, Islam S, Khalid IF. (2021). Internet of Things-Based Smart Electricity Monitoring and Control System Using Usage Data. *Wirel Commun Mob Comput*. 2021. doi:10.1155/2021/6544649.
- Hermansyah H, Rahmawati NA, Achmad A. (2023). Sistem Kendali Jarak Jauh Beban Listrik Rumah Tangga Berbasis Internet of Things (IoT). *Jambura Physics Journal*. 5(1):29–36. doi:10.34312/jpj.v5i1.19026.
- Jabir HJ, Teh J, Ishak D, Abunima H. (2018). Impacts of demand-side management on electrical power systems: A review. *Energies (Basel)*. 11(5). doi:10.3390/en11051050.
- Klimberg, R., Sillup, G., Boyle, K., & Tavva, V. (2010). Forecasting performance measures - What are their practical meaning? *Advances in Business and Management Forecasting*, 7, 137–147. doi:10.1108/S1477-4070(2010)0000007012
- Latifah A, Nugeraha RA. (2024). Design and Build a Smart Electrical Socket Using IoT Based Arduino Uno Microcontroller. *Journal of Electrical, Electronic, Information, and Communication Technology*. 6(2):72. doi:10.20961/jeeict.6.2.92607.
- Liang THL, Phuong T Le, Hung HW. (2023). Design and Evaluation of Wireless Power Monitoring IoT System for AC Appliances. *Energies (Basel)*. 16(1). doi:10.3390/en16010163.
- Napis, Farhan M, Rahmatulloh, Hakim AR, Apriyanto MT. (2023). Meningkatkan Kesadaran Masyarakat dalam Budaya Hemat Energi Melalui Penyuluhan Efisiensi Penggunaan Listrik Rumah Tangga. *Jurnal Pendidikan dan Pengabdian Masyarakat*. 6(2):107–115. doi:10.29303/jppm.v6i2.4980.
- Oh J. (2020). IoT-based smart plug for residential energy conservation: An empirical study based on 15 months' monitoring. *Energies (Basel)*. 13(15). doi:10.3390/en13154035.
- Pardede JO, Brahmana K. (2023). Improving Reading Accuracy of ACS712 Current Sensor with ATmega328 10-Bit ADC: Enhancing Resolution to 5mA/BIT via AD620 Differential Amplifier and Kalman Filters. *Journal of Technomaterial Physics*. 5(2):104–110. doi:10.32734/jotpv5i2.11355.
- Park Y, Han J. (2025). Smart Home Advancements for Health Care and Beyond: Systematic Review of Two Decades of User-Centric Innovation. *J Med Internet Res*. 27:e62793. doi:10.2196/62793.
- Permana BAC, Sadali M, Sudianto A, Harianto, Wijaya LK. (2024). Sistem Kendali Perangkat Listrik dan Monitoring Daya Listrik Berbasis Internet Of Things. *Infotek: Jurnal Informatika dan Teknologi*. 7(2):544–551. doi:10.29408/jit.v7i2.26534.

- Prayitno B, Palupiningsih P, Agtriadi HB, Tinggi S, Pln T. (2019). Prototipe Sistem Monitoring Penggunaan Daya Listrik Peralatan Elektronik Rumah Tangga Berbasis Internet of Things. 12(1).
- Qingxian Zeming Langrui Electronic. (2020). ZMPT101B precision voltage transformer datasheet. Hebei, China: Zeming Electronics Co., Ltd. Diperoleh dari <https://www.zeming-e.com>
- Rahmanto R, Widodo S, Ayuningtyas B. (2023). Pemanfaatan IoT untuk Manajemen Listrik Terintegrasi yang Dapat Diatur Berdasarkan Waktu Secara Online. Jurnal Pengabdian Masyarakat (ANDHARA). <http://jurnal.undira.ac.id/index.php/jpmk/>.
- Saharo S, Zaki Hamidi EA, Nurmalasari RR. (2022). Power Monitoring System of Home-scale Internet of Things (IoT). In: Proceedings of the 2022 16th International Conference on Telecommunication Systems Services and Applications, TSSA 2022. Institute of Electrical and Electronics Engineers Inc.
- Sanchez-Sutil F, Cano-Ortega A. (2023). Smart plug for monitoring and controlling electrical devices with a wireless communication system integrated in a LoRaWAN. Expert Syst Appl. 213:118976. doi:<https://doi.org/10.1016/j.eswa.2022.118976>.
- Shenzhen Hi-Link Electronic Co., Ltd. (2020). HLK-10M05 AC-DC Power Module Technical Datasheet. Shenzhen Hi-Link Electronic Co., Ltd.
- Siahaan MN. (2024). Rancang Bangun Smart Socket dalam Mengatasi Over Current Menggunakan Relay dan Mikrokontroler.
- Siskandar, R., Fadhil, M. A., Kusumah, B. R., Irmansyah, I., & Irzaman, I. (2020). INTERNET OF THINGS: AUTOMATIC PLANT WATERING SYSTEM USING ANDROID. Jurnal Teknik Pertanian Lampung (Journal of Agricultural Engineering), 9(4), 297. doi:10.23960/jtep-l.v9i4.297-310
- Siskandar, R., Pramudianto, R. D., Hasan, N. A., & Novianty, I. (2018). Penerapan Komunikasi Berbasis Cahaya Tampak pada Prototipe Kendaraan Remote Control Guna Meningkatkan Keamanan dan Otomatisasi Komunikasi Antar Kendaraan.
- Siskandar, R., Santosa, S. H., Wiyoto, W., Kusumah, B. R., & Hidayat, A. P. (2022). Control and Automation: Insmoaf (Integrated Smart Modern Agriculture and Fisheries) on The Greenhouse Model. Jurnal Ilmu Pertanian Indonesia, 27(1). doi:10.18343/jipi.27.1.141
- Songle Relay. (2018). SRD series miniature PCB power relay datasheet. Ningbo, China: Ningbo Songle Relay Co., Ltd. Diperoleh dari <https://www.songle.com>
- Sundaramoorthi P, Alexander AS. (2023). Cloud-Based MTT Protocol for Power Monitoring and Socket Control System. Int J Adv Sig Img Sci. 9(1).
- Surahman U, Maknun J, Krisnanto E. (2017). Survey on household energy consumption of public apartments in Bandung City, Indonesia. In: 8th International Conference on Architecture Research and Design (AR+DC). Bandung: IPTEK Journal of Proceedings Series.
- Wibowo A, Gede I, Jayendra B, Utari DY, Rasyiid M, Aprilianto RA. (2024). Design of ACS712-based Current Sensing Monitoring Evaluation for PWM Solar Charge Controller. Volume 22. <https://journal.unnes.ac.id/journals/sainteknol>.
- Wibowo F, Suheri S, Yugianus P. (2024). An Economical Internet of Things (IoT) Driven Approach for Non-Intrusive Smart Energy Monitoring. G-Tech: Jurnal Teknologi Terapan. 8(2):1069–1078. doi:10.33379/gtech.v8i2.4159.