**Center for Embedded and Cyber-Physical Systems**
**University of California, Irvine**

# ZotPlug: An IoT Smart-Plug System for Monitoring and Reducing Dormitory Energy Consumption

Prabhav Goyal, Christopher Chandler, Justin Chang, Kyle Chun, Jonghyun Choi, and Rainer Doemer

Technical Report CECS-26-02
May 26, 2026

Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
(949) 824-8919

{prabhavg,cachandl}@uci.edu
http://www.cecs.uci.edu

# ZotPlug: An IoT Smart-Plug System for Monitoring and Reducing Dormitory Energy Consumption

Prabhav Goyal, Christopher Chandler, Justin Chang, Kyle Chun, Jonghyun Choi, and Rainer Doemer

Technical Report CECS-26-02
May 26, 2026

Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
(949) 824-8919

{prabhavg,cachandl}@uci.edu
http://www.cecs.uci.edu

**Abstract**

*University dormitories experience significant energy waste because many electronic devices remain plugged in and consume power even when idle. Studies estimate that these* energy vampire *devices account for nearly* 30% *of unnecessary energy consumption in dormitory environments. To address this issue, we developed ZotPlug, a smart outlet system that monitors and controls energy usage at the individual outlet level.*

*The system measures real-time energy consumption using a dedicated power-metering integrated circuit connected to an ESP32 microcontroller and transmits telemetry to a cloud-based platform. Through a responsive dashboard application, users can visualize energy usage, remotely control connected devices, and configure scheduling features to promote more efficient electricity consumption. The platform also incorporates behavioral incentives that encourage students to reduce energy use through friendly competition and increased awareness of their consumption habits. Our experimental evaluation demonstrates that the system achieves measurement accuracy within* 5% *of expected values across a wide range of electrical loads. Overall, ZotPlug provides reliable outlet-level energy monitoring, while seamlessly integrating with a scalable cloud infrastructure and an intuitive cross-platform user interface that supports UC Irvine's long-term sustainability initiatives.*

*This undergraduate capstone project was developed in a two-quarter (six-month) period that spanned Fall 2025 and Winter 2026. Through this development cycle, we iteratively refined both the hardware and software subsystems from early proof-of-concept prototypes into a validated end-to-end implementation.*

# Contents

# List of Figures



# List of Tables

# ZotPlug: An IoT Smart-Plug System for Monitoring and Reducing Dormitory Energy Consumption

**Prabhav Goyal, Christopher Chandler,**
**Justin Chang, Kyle Chun, Jonghyun Choi, and Rainer Doemer**

Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
{prabhavg,cachandl}@uci.edu
http://www.cecs.uci.edu

## Abstract

*University dormitories experience significant energy waste because many electronic devices remain plugged in and consume power even when idle. Studies estimate that these* energy vampire *devices account for nearly* 30% *of unnecessary energy consumption in dormitory environments. To address this issue, we developed ZotPlug, a smart outlet system that monitors and controls energy usage at the individual outlet level.*

*The system measures real-time energy consumption using a dedicated power-metering integrated circuit connected to an ESP32 microcontroller and transmits telemetry to a cloud-based platform. Through a responsive dashboard application, users can visualize energy usage, remotely control connected devices, and configure scheduling features to promote more efficient electricity consumption. The platform also incorporates behavioral incentives that encourage students to reduce energy use through friendly competition and increased awareness of their consumption habits. Our experimental evaluation demonstrates that the system achieves measurement accuracy within* 5% *of expected values across a wide range of electrical loads. Overall, ZotPlug provides reliable outlet-level energy monitoring, while seamlessly integrating with a scalable cloud infrastructure and an intuitive cross-platform user interface that supports UC Irvine's long-term sustainability initiatives.*

*This undergraduate capstone project was developed in a two-quarter (six-month) period that spanned Fall 2025 and Winter 2026. Through this development cycle, we iteratively refined both the hardware and software subsystems from early proof-of-concept prototypes into a validated end-to-end implementation.*

## 1 Problem Statement

University campuses represent high-density environments for energy consumption, where hundreds of thousands of electronic devices are continuously connected across dormitories, laboratories, and public facilities. A significant portion of this energy is wasted through so-called *energy vampire* devices, which continue to consume power even when they are not actively in use. This idle energy consumption contributes substantially to resource waste and often goes unnoticed in shared living environments such as student housing.

Anderson *et al*. [1] reported that nearly 30% of the electrical energy consumption in dormitories occurs while the rooms are not occupied, highlighting inefficiencies in both user behavior and existing energy-management practices on residential campuses. Institutions such as UC Irvine therefore present a compelling opportunity to reduce unnecessary energy consumption for both environmental and economic reasons, especially considering that annual

electricity costs in large public universities can exceed $100,000 [2].

Furthermore, previous studies have shown that disconnecting unused electronic devices can reduce electricity usage by as much as 38% in certain scenarios [3]. These findings motivated the development of *ZotPlug* as a smart energy-management system capable of monitoring outlet-level power consumption and encouraging more sustainable energy usage behaviors within dormitory environments.

# 2 Project Overview

## 2.1 Development Timeline and Scope

The *ZotPlug* project was developed in two academic quarters that spanned Fall 2025 and Winter 2026. During the first phase of development, we focused on architectural exploration, proof-of-concept validation, and early interface prototyping for both the hardware and software subsystems. The second phase emphasized system refinement, hardware redesign, cloud integration, interface improvements, and final prototype validation. This iterative development approach enabled us to progressively improve the accuracy, scalability, and usability of the system while transitioning from an experimental prototype to a validated end-to-end implementation.

## 2.2 System Objectives

We developed *ZotPlug* as a smart-plug platform capable of real-time energy monitoring and power-metering to support sustainability incentives at UC Irvine. The system combines accurate electrical sensing hardware, an ESP32 microcontroller, scalable cloud infrastructure, and a responsive user interface into a unified IoT platform. Through this system, users can visualize device-level energy consumption, automate schedules, remotely control connected devices, and participate in incentive-based rewards programs designed to encourage more sustainable habits within dormitory environments.

## 2.3 Project Accomplishments

At the end of the initial development phase, we produced a functional hardware prototype capable of measuring device-level energy consumption, established the foundations of the cloud infrastructure and database architecture, and implemented a rudimentary user interface for interacting with *ZotPlug* devices.

During the second development phase, we substantially improved the system by redesigning the hardware circuitry to achieve higher measurement accuracy, refining the cloud infrastructure and backend communication logic, and expanding the user interface with improved responsiveness, visualization features, and device-management capabilities. Table 1 summarizes the main hardware and software milestones completed throughout the six-month development cycle.

# 3 Detailed Project Objectives

Throughout the six-month development cycle, the *ZotPlug* project evolved through several stages of technical refinement and feasibility analysis. At the beginning of the project, we established a broad set of hardware and software objectives intended to guide the development of a fully integrated smart energy-management system. As implementation progressed, practical engineering constraints, safety considerations, and project scope limitations required us to prioritize the most critical system features necessary for a reliable final prototype.

## 3.1 Original Design Objectives

The original hardware objectives focused on developing a compact and fully deployable smart-plug platform capable of accurate outlet-level power monitoring and remote device control:

1. Develop a smart-plug IoT device capable of real-time energy monitoring and accurate power reporting.
2. Design a compact printed circuit board (PCB) capable of safe interfacing with standard household wall outlets.

| | *Hardware* | *Software* |
|---|---|---|
| *September* | Begin initial circuit development and component selection. | Begin software-stack selection and basic frontend/backend development. Design UI/UX wireframes. |
| *October* | Order components and finalize clamp-current sensor circuit. | Continue frontend/backend development and standardize end-to-end communication. |
| *November* | Perform hardware accuracy testing of the clamp-current circuit. Begin metering IC research and circuit development. | Test frontend/backend functionality with dummy data and begin new UI/UX mockups. |
| *December* | Finish clamp-current sensor circuit and test functionality with live data. Finish metering IC circuit development. | Finish rudimentary user interface and basic server functionality with live hardware readings. |
| *January* | Begin PCB layout design with metering IC circuit schematic. | Begin user interface overhaul based on new UI/UX mockups. |
| *February* | Fabricate PCB layout and build 3D enclosure. Begin perfboard development. | Develop frontend graphs and backend data visualization logic. |
| *March* | Finish perfboard circuit with new 3D enclosure. Test hardware accuracy and live functionality. | Polish user interface and test all features with live hardware readings. |

Table 1: **Six-Month Development Timeline for *ZotPlug* across Fall 2025 and Winter 2026 Quarters.** This timeline summarizes the parallel hardware and software milestones completed throughout the two-quarter development cycle, illustrating the project's iterative progression from initial component selection and software architecture planning to final prototype integration, testing, and system validation.

3. Package hardware in a compact, safe, 3-dimensional printed enclosure suitable for dormitory deployment.

4. Integrate firmware-based energy-savings functionality, including scheduled operation and configurable energy limits.

The original software objectives emphasized scalable cloud communication, intuitive data visualization, and user engagement features:

1. Build a secure cloud-based backend infrastructure for seamless device-to-user communication.

2. Develop an intuitive dashboard interface for energy visualization, scheduling, and device management.

3. Integrate "gamification" features that encourage reduced energy consumption through rewards and peer competition.

4. Develop a dedicated mobile application capable of providing functionality equivalent to the web platform.

As development progressed, we refined these objectives to prioritize system stability, measurement accuracy, and reliable end-to-end integration. Several advanced features were ultimately de-prioritized in order to ensure successful completion of the core hardware and software infrastructure within the project timeline.

### 3.2 Refined Hardware Objectives

The final hardware objectives focused on developing a reliable and experimentally validated prototype capable of accurate real-time energy monitoring:

1. Design and validate a hardware architecture capable of accurate real-time voltage, current, and power monitoring (see Figure 3).

2. Develop a compact enclosure and prototype suitable for controlled low-voltage testing and demonstration (see Figure 9).

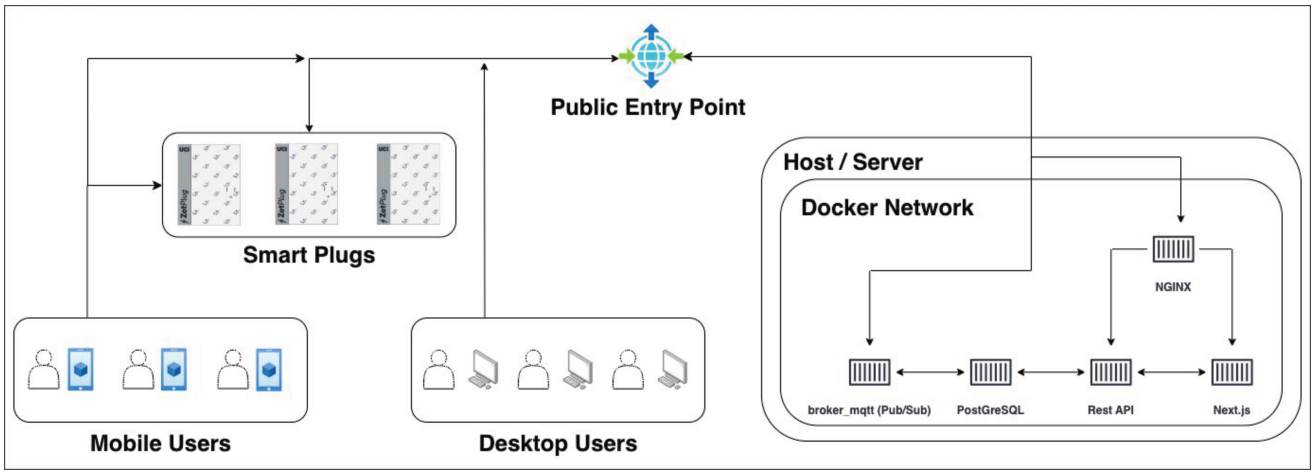



Figure 1: **System Level Software Architecture of the *ZotPlug* Platform.** This infrastructure diagram illustrates the interaction between the responsive user-interface, cloud-hosted backend services, and embedded hardware communication layer that together enable real-time energy monitoring, device control, and scalable data processing.

3. Transition the final experimental validation to a perfboard-based implementation in order to improve safety, reliability, and ease of hardware iteration during testing (see Figure 10).

4. Implement reliable remote device switching through an ESP32-controlled relay interface integrated with the cloud dashboard.

Although we successfully developed a PCB prototype, we were ultimately unable to finalize a production-ready PCB design suitable for safe deployment in high-voltage wall outlets within the available project timeline. In addition, advanced firmware-controlled energy optimization features were deferred in favor of ensuring stable monitoring, communication, and relay-control functionality.

### 3.3 Refined Software Objectives

The final software objectives emphasized scalable infrastructure, reliable communication, and responsive cross-platform usability:

1. Develop a secure and scalable backend infrastructure capable of supporting reliable device-to-user communication (see Figure 1).

2. Design a responsive and intuitive user interface for energy visualization, device monitoring, and remote outlet control (see Figure 5).

3. Implement the core cloud infrastructure required for telemetry processing, database integration, and real-time dashboard updates.

4. Develop a mobile-compatible frontend architecture using shared React-based components to support both web and mobile platforms.

Although we initially planned to implement competitive gamification and reward-based energy saving features, we ultimately prioritized the development of stable core infrastructure and responsive monitoring capabilities. Similarly, while the initial project plans included a fully dedicated mobile application, we instead focused on delivering a mobile-responsive web interface that provided consistent functionality across desktop, tablet, and mobile devices.

## 4 System Implementation

This section describes the hardware and software implementation of the *ZotPlug* platform. We describe the evolution of the hardware architecture, the transition from the initial sensing design to the final

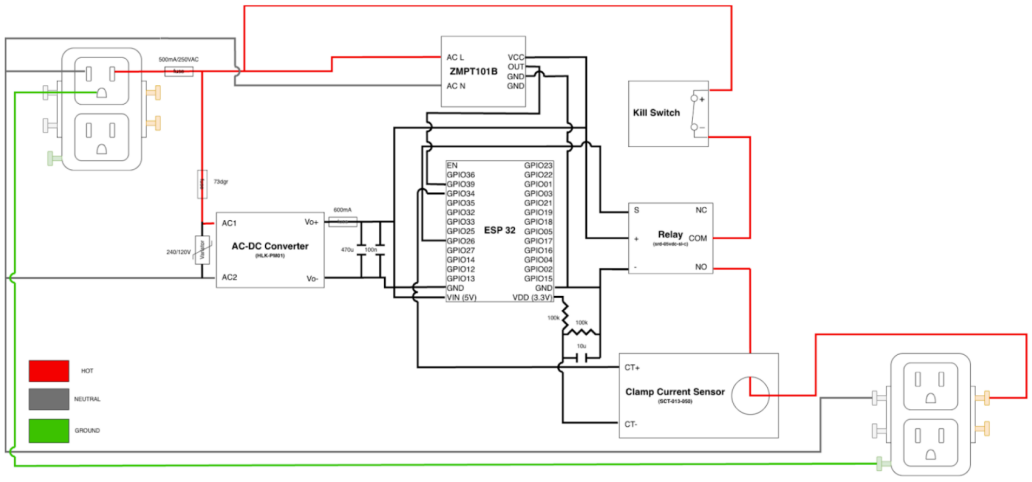



Figure 2: **Fall Quarter Current-Clamp-Based Circuit Architecture.** This initial hardware design explored non-invasive current sensing as an early proof-of-concept implementation; however, subsequent testing revealed insufficient measurement precision for reliable deployment.

metering-IC implementation, and the cloud infrastructure that supports real-time monitoring and device communication.

## 4.1 Hardware Implementation

The hardware subsystem underwent multiple design iterations throughout the project as we refined the sensing architecture, improved measurement accuracy, and integrated safer power delivery components. The early circuit schematic focused on validating basic sensing and control functionality, while the final design introduced a dedicated energy metering-IC to overcome limitations discovered during initial testing.

### 4.1.1 Current-Clamp Prototype Architecture

The initial circuit integrated an ESP32 microcontroller, a clamp-based current sensor, a relay switch, and a low-voltage power supply to validate core IoT functionality such as sensing, device control, and wireless communication (See Figure 2).

- The `ESP32 microcontroller` was selected due to its integrated WiFi capabilities, sufficient processing power for embedded applications, and multiple general-purpose input/output (GPIO) interfaces for sensor and relay control. Its built-in wireless networking eliminated the need for a separate communication module, simplifying the overall system architecture for an IoT-based smart plug device.

- A `current clamp sensor` was initially used to measure current non-invasively while maintaining galvanic isolation from the AC mains. This approach enabled safe early-stage experimentation, but inaccurate experimental results (see Section 8) motivated a transition to a different approach in the Winter quarter.

- The power for the low-voltage circuitry was provided using an `HLK-PM01 AC-DC converter`, which converts $120\,V$ AC mains power to a regulated $5\,V$ DC supply. This module was selected due to its compact form factor, electrical isolation, and widespread use in IoT power supply designs.

- To enable outlet-level power control, the system used an `SRD-05VDC-SL-C relay switch`. This relay safely switches standard household loads while remaining directly controllable through the ESP32 GPIO interface.

Although this architecture successfully demonstrated wireless monitoring and device switching, testing revealed that relying on the ESP32 ADC for

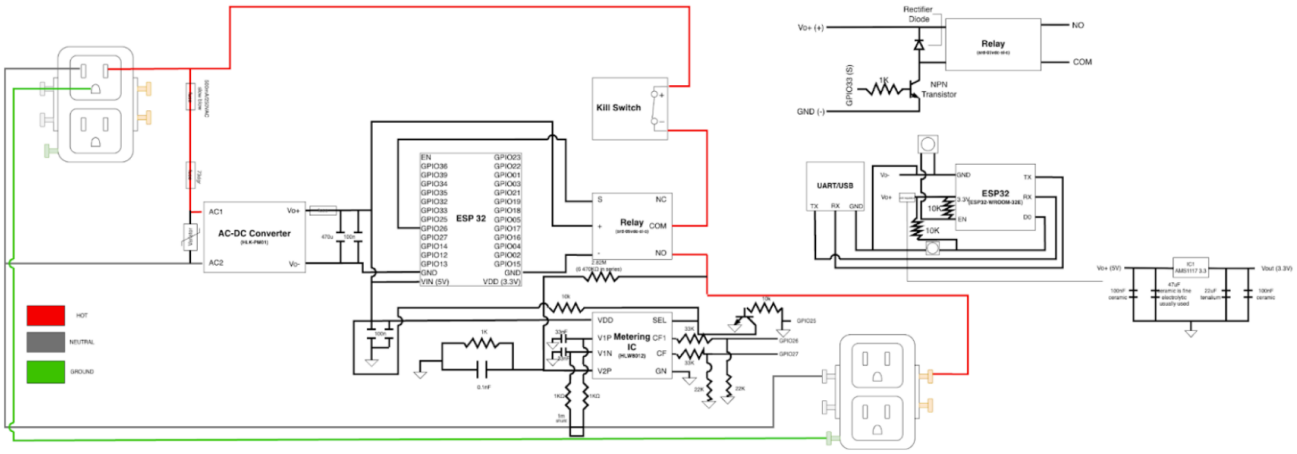



Figure 3: **Winter Quarter Metering-IC-Based Circuit Architecture.** This revised hardware design incorporates a dedicated metering integrated circuit (IC) to improve sensing accuracy and served as the foundation for the final validated perfboard and PCB implementations.

current measurement resulted in inaccurate and unstable readings.

### 4.1.2 Metering-IC System Architecture

To address the measurement instability observed in the initial prototype, the final hardware architecture incorporated a dedicated power-metering IC. Rather than relying on the ESP32 for direct analog sampling, the revised design offloaded voltage, current, and power calculations to specialized hardware, allowing the microcontroller to focus primarily on digital signal processing, relay control, and wireless communication.

More specifically, our chosen metering-IC was the `HLW8012`, which performs real-time voltage, current, and power calculations and outputs frequency-based signals proportional to the measured electrical parameters. This design change substantially improved the reliability and scalability of the *ZotPlug* hardware and enabled accurate real-time energy monitoring suitable for dormitory deployments.

Figure 3 illustrates the final circuit architecture based on the metering-IC. The upper-left portion of the schematic contains the AC input stage, where the hot, neural, and ground conductors are represented in red, gray, and green, respectively. Along the top of the schematic, the hot line passes through the primary protection and switching path, including the fuse, kill switch, and relay, before reaching the AC output stage on the bottom of the circuit.

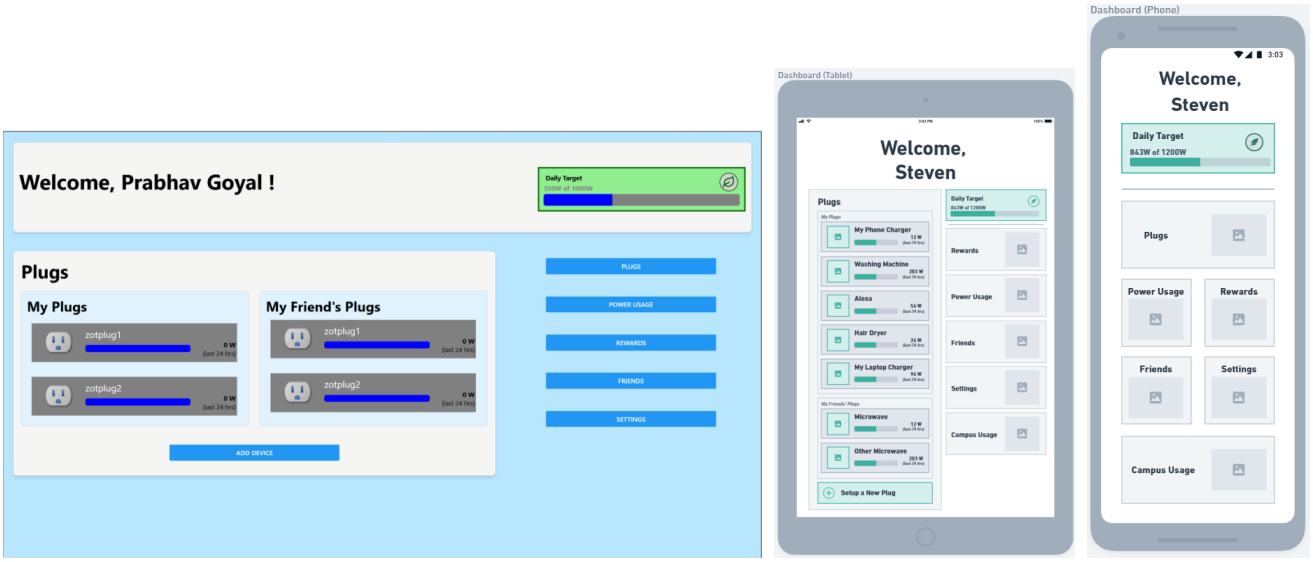
The hot line is also routed to the HLK-PM01 AC–DC converter to generate, which generates regulated low-voltage power for the control electronics, and to the HLW8012 metering-IC through a voltage-divider network that scales the mains voltage to a safe proportional measurement signal. Similarly, at the bottom, the neutral line passes through a shunt resistor that produces a lower proportional current measurement for the metering-IC before continuing to the AC output stage. The neutral line also connects to the AC–DC converter to complete the power-supply circuit.

The center portion of the schematic contains the metering-IC and its supporting sensing circuitry, including the voltage-divider and current-sensing components required for real-time power calculations. The right side of the schematic integrates the ESP32, relay-control circuitry, and status-indicator LEDs responsible for wireless communication, device control, and system debugging.

## 4.2 Software Implementation

The *ZotPlug* software platform consists of a cloud-based backend infrastructure and a cross-platform frontend interface that enables real-time device monitoring and control.

The backend services are implemented using




(a) Desktop (b) Tablet (c) Mobile

Figure 4: **Fall Quarter User Interface Prototype Across Desktop, Tablet, and Mobile Form Factors.** This initial dashboard mock-up represents the first-stage software interface developed during Fall Quarter and illustrates the foundational design concepts used to evaluate responsive layouts, cross-platform usability, and early user interaction workflows.


(a) Desktop (b) Tablet (c) Mobile

Figure 5: **Winter Quarter Dashboard Interface Across Desktop, Tablet, and Mobile Form Factors.** This refined interface demonstrates substantial improvements made during Winter Quarter, including enhanced navigation, real-time energy visualization, and a more polished design intended to support intuitive monitoring of device-level energy consumption.

Node.js v20 with the Express.js v5.1.0 web framework and run on a Google Cloud Platform (GCP) virtual machine within Docker v27.x containers. NGINX v1.27.x (Engine X reverse proxy) manages incoming traffic and routes requests to internal services, including the REST API, the MQTT broker, and the database. Device telemetry is transmitted using the MQTT protocol (Message Queuing Telemetry Transport) and is stored in a PostgreSQL v18 relational database.

The frontend dashboard is implemented using Next.js v16 with a React-based full-stack web framework built on React v19.2.4. The shared component architecture enables deployment across both web and mobile platforms using React Native v0.81.5, enabling a consistent user experience across web and mobile devices.

The end-to-end workflow operates as follows: *ZotPlug* devices publish telemetry data to the MQTT broker, which is processed by backend services and stored in the database. The frontend dashboard retrieves this information through REST API requests, permitting users to monitor energy usage, manage connected devices, and configure energy-savings schedules.

To improve coordination between the hardware and software teams, we developed several internal documentation and workflow tools. Most notably, we used an API specification tool, Swagger Documentation, to document backend endpoints and streamline communication between frontend and backend developers. In addition, we maintained project setup instructions and troubleshooting documentation through the GitHub Wiki, allowing the team to quickly reference solutions and streamline the development process.

# 5 Engineering Standards and Communication Protocols

This section summarizes the engineering standards, communication protocols, and infrastructure conventions adopted throughout the development of the *ZotPlug* platform. These standards guided the design of the hardware subsystem, secured device-to-cloud communication, and supported scalable deployment of the overall system architecture.

## 5.1 Hardware Standards and Safety Considerations

The hardware subsystem incorporates standard electrical isolation, PCB layout, and low-voltage safety practices to improve operational reliability and reduce electrical hazards during testing and prototyping. More specifically, the hardware design incorporated guidance from the following industry standards:

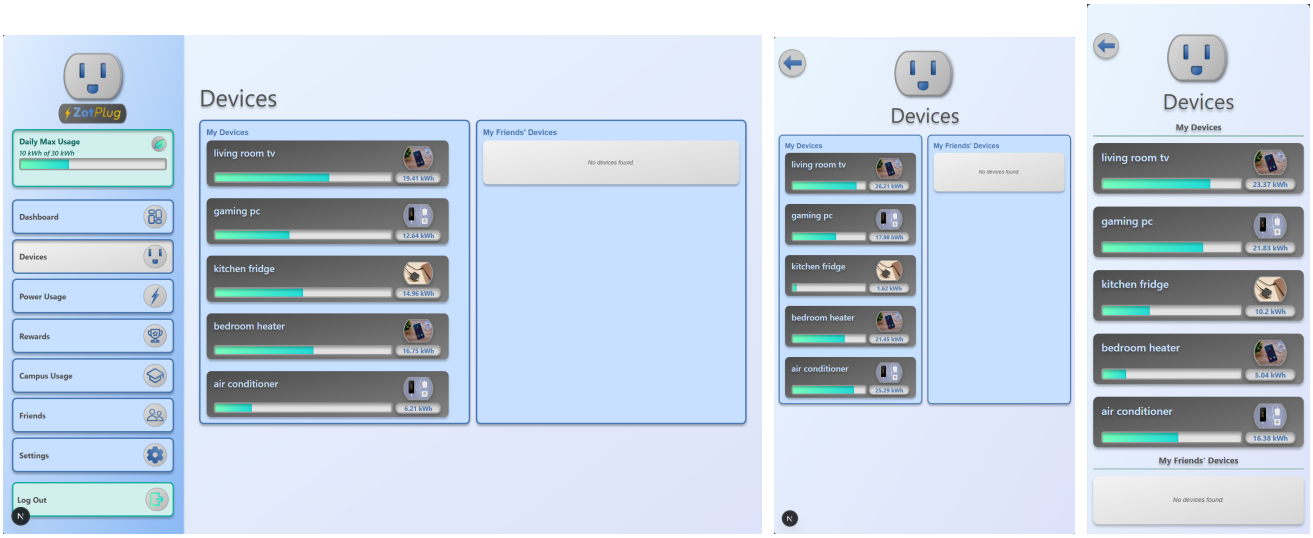
- IEC 60664: Insulation coordination for equipment within low-voltage systems.
- IEC 62368: Safety requirements for audio/video, information, and communication technology equipment.
- IPC-2221B: Generic PCB design standards governing trace spacing, electrical clearance, and board-level layout practices.
- NEMA 1-15/5-15: Standardized North American outlet and plug specifications relevant to residential electrical systems.

## 5.2 Software Protocols and Infrastructure Standards

The software incorporates established networking, authentication, and IoT standards to support secure device communication, scalable backend deployment, and reliable frontend interaction. The software platform integrates the following communication and infrastructure standards:

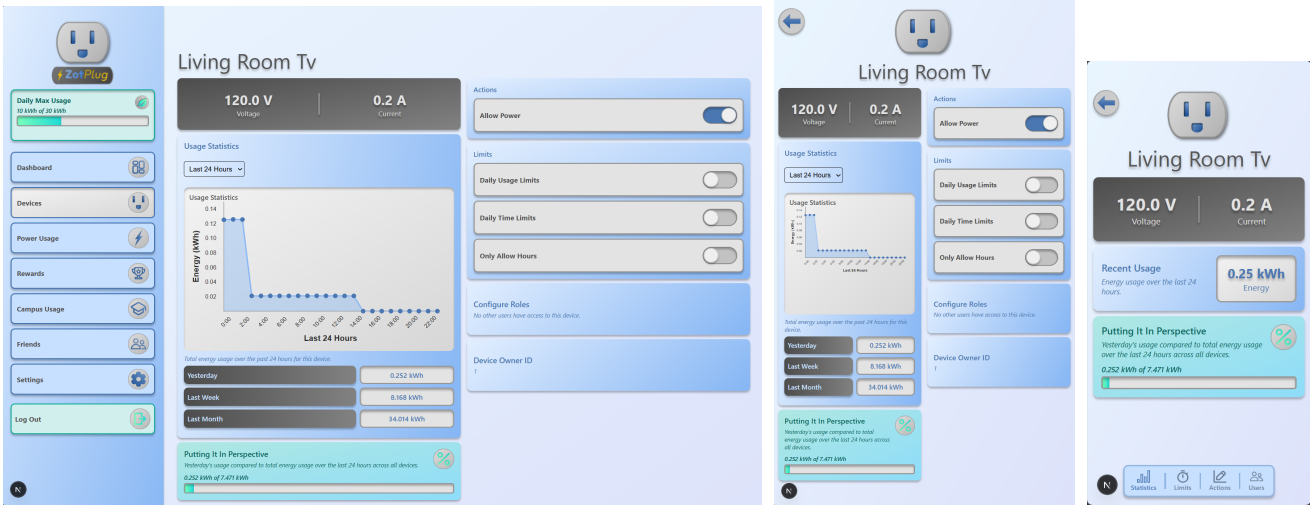
### 5.2.1 Network Security and Transport Protocols

The *ZotPlug* platform relies on modern transport and encryption standards to provide secure communication between embedded devices, frontend clients, and cloud-hosted backend services. These protocols ensure confidentiality, integrity, and reliable network connectivity throughout the system architecture.




(a) Desktop (b) Tablet (c) Mobile

Figure 6: **Winter Quarter Device Management Interface Across Multiple Device Platforms.** This interface enables users to monitor registered *ZotPlug* devices, compare energy consumption patterns, and access system-wide usage insights through a unified and responsive management dashboard.




(a) Desktop (b) Tablet (c) Mobile

Figure 7: **Winter Quarter Device Level Monitoring Interface Across Multiple Devices for Detailed Energy Analysis.** This view provides users with detailed statistics for individual *ZotPlug* devices, including energy consumption trends, historical usage metrics, and interactive controls for device-level monitoring and analysis.

- `TLS 1.2 / TLS 1.3`: Encryption protocols used to secure communication between clients, backend services, and cloud infrastructure.
- `HTTPS`: Secure web communication protocol integrating HTTP with TLS-based encryption.
- `TCP/IP`: Core IPv4 networking protocol suite supporting communication between embedded devices, backend services, and frontend clients.

#### 5.2.2 Web Architecture and Authentication

The frontend and backend infrastructure follows widely adopted web-development and authentication standards to support scalable communication, secure user access, and reliable session management across the platform.

- `RESTful API`: HTTP-based application programming interface architecture supporting communication between frontend clients and backend services.
- `JSON Web Tokens (JWT)`: Stateless authentication mechanism used to securely validate user identity across client applications.
- `Secure Session Cookies`: Session management mechanism used to maintain authenticated user access within the web dashboard.

#### 5.2.3 IoT Communication Standards

The embedded hardware relies on lightweight communication protocols optimized for low-power IoT deployments and real-time telemetry exchange. These standards enable efficient communication between the ESP32 microcontroller, the MQTT broker, and the cloud infrastructure while minimizing network overhead.

- `IEEE 802.11n`: Wireless networking standard providing WiFi connectivity for ESP32-based communication.
- `MQTT (ISO/IEC 20922)`: Lightweight publish/subscribe messaging protocol supporting real-time telemetry transmission and remote device control.
- `MQTT Access Controls List (ACLs)`: Access control mechanism restricting broker communication to authenticated and authorized devices.

#### 5.2.4 Cloud Infrastructure Standards

The backend infrastructure follows modern cloud-computing and reverse-proxy deployment practices to support scalability, service isolation, and reliable external access to platform resources.

- `NGINX Reverse Proxy`: Reverse proxy infrastructure responsible for routing external traffic to internal backend services.
- `Google Cloud Platform (GCP)`: Cloud computing platform hosting the backend infrastructure, database services, and containerized applications.

# 6 Security and Safety Considerations

This section describes the hardware and software security mechanisms incorporated into the *ZotPlug* platform. We discuss the electrical protection features integrated into the hardware subsystem, the network and authentication safeguards implemented within the cloud infrastructure, and several additional protections that would be necessary for production-scale deployment.

## 6.1 Electrical Safety and Hardware Protection

The *ZotPlug* circuit schematic (see Figure 3) was designed to safely interface with the $120V$ AC mains power while protecting both the user and the device from electrical, thermal, and over-current hazards. Although the final prototype was validated under controlled low-voltage conditions, the schematic incorporates multiple protection mechanisms intended to support safe operation under household AC power.

On the high-voltage side of the system, a $6A$ slow-burn fuse is placed along the AC input to protect the circuitry from sustained over-current conditions while

tolerating short inrush current spikes commonly produced during appliance startup. A metal-oxide varistor (MOV) is connected across the mains input to suppress transient voltage spikes and protect downstream components from electrical surges. The system also incorporates an isolated AC–DC converter to step down the $120V$ AC input to a regulated $5V$ DC supply while maintaining galvanic isolation between the mains interface and the low-voltage control circuitry.

Additional protections were implemented on the low-voltage side of the design. A $600mA$ fast-blow fuse protects the $5V$ rail supplying the ESP32, relay switch, and metering IC from abnormal current draw conditions. A 73°C thermal fuse was added near the AC–DC conversion stage to mitigate overheating conditions and reduce the risk of thermal failure. We also sized the PCB traces carrying load current using standard current-carrying trace calculations ($2mm$ width with $1oz$ copper) to maintain safe operating temperatures under a $5A$ load. In addition, decoupling and filtering capacitors were placed near sensitive components to stabilize voltage levels and reduce electrical noise within the sensing circuitry. Finally, all electronics were enclosed within a 3D-printed insulated casing to minimize accidental user contact with conductive elements and reduce electrical shock hazards.

For safety reasons, we did not test the prototype directly under $120V$ AC mains power. Instead, we validated the system using controlled $12V$ AC power supplies while designing the circuit to theoretically support household AC operation using the protection mechanisms and safety margins described above.

### 6.2 Software Security and Access Control

The *ZotPlug* software infrastructure incorporates multiple security mechanisms intended to protect backend services, secure device communication, and restrict unauthorized system access.

The backend services are hosted on a hardened *Google Cloud Platform (GCP)* virtual machine configured with restricted firewall rules to minimize unnecessary external exposure. An `NGINX` reverse proxy isolates internal services from direct public access and routes requests to backend APIs, database services, and MQTT communication endpoints. The system also implements rate limiting to mitigate denial-of-service attacks and reduce abusive network traffic.

All communication between embedded devices, frontend clients, and backend services is encrypted using TLS-based transport security. Authenticated user sessions utilize expiration-based session tokens to reduce the likelihood of session hijacking and unauthorized persistent access. On the IoT side of the platform, MQTT access-control lists (ACLs) restrict publish and subscribe permissions to authenticated devices only, preventing unauthorized hardware from interacting with the broker infrastructure.

Additional protections were incorporated at the application layer. The platform enforces strong-password requirements to reduce susceptibility to password-cracking attacks. Moreover, the system minimizes the amount of personally identifiable information collected from users by limiting stored data to that required for core platform operation.

### 6.3 Future Security Enhancements

A full-scale production deployment of *ZotPlug* would require additional hardware, firmware, and infrastructure protections beyond those implemented in the prototype system. Future security improvements would include the following:

1. Unique device identity provisioning for authenticated IoT device enrollment.
2. Secure over-the-air (OTA) firmware update infrastructure.
3. Hardware-level protections such as secure boot and encrypted flash storage.
4. Dedicated secure credential storage for WiFi and MQTT broker credentials.
5. Additional electrical protections, including ground-fault protection and certified isolation barriers for the $120V$ AC interface.
6. Formal compliance testing against consumer electrical safety standards such as UL and IEC certification requirements for residential deployment.

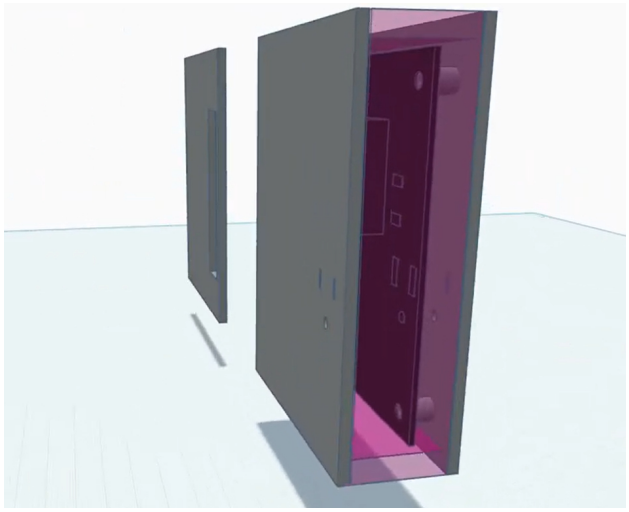


Figure 8: **Preliminary Enclosure Design for the *ZotPlug* Hardware Prototype.** This early schematic illustrates the initial mechanical enclosure concept developed during Winter Quarter to house the hardware subsystem and guide subsequent iterations of physical packaging and structural integration.


Figure 9: **Early Printed Circuit Board Prototype with Integrated 3D-Printed Enclosure.** This intermediate Winter Quarter prototype represents the first surface-mount implementation of the hardware design and served as an important evaluation stage prior to transitioning to the final perfboard-based implementation.

# 7 Prototype Development and Refinement

This section summarizes the progression of the *ZotPlug* platform from its initial proof-of-concept implementation to the final validated prototype. We discuss the evolution of both the hardware and software subsystems, the limitations identified during early testing, and the refinements introduced throughout the second development phase.

## 7.1 Initial Prototype Development

### Hardware Prototype

At the end of the first development phase, we completed an initial hardware prototype based on the circuit schematic shown in Figure 2. The system successfully measured current from the connected loads, estimated energy consumption, and transmitted telemetry data to the backend infrastructure for storage and visualization.

Although the prototype demonstrated the feasibility of real-time monitoring and wireless device communication, the design exhibited several important limitations. The clamp-based sensing architecture produced a physically bulky hardware layout, while the ESP32 analog-to-digital converter (ADC) generated noisy and inconsistent measurements that reduced overall sensing accuracy. These issues, discussed further in Section 8, ultimately motivated the transition to a dedicated metering-IC architecture during the second development phase.

### Software Prototype

During the first development cycle, we implemented the foundational backend infrastructure, database architecture, and a basic user interface that allowed users to register and interact with their *ZotPlug* devices (see Figure 1). This implementation established the platform as a functional minimum viable product capable of supporting end-to-end communication between the embedded hardware, backend services, and frontend dashboard.

Despite achieving core functionality, the software prototype still lacked several important features and refinements. The interface was constructed primarily from early UI/UX wireframes intended to validate

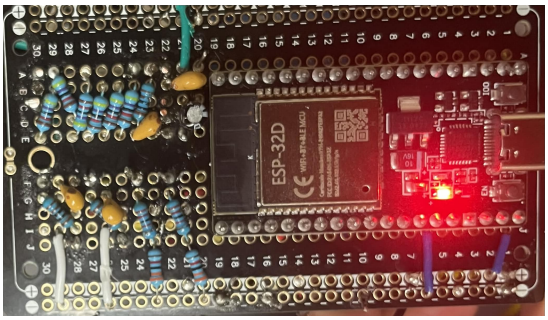



Figure 10: **Final Perfboard-Based Hardware Prototype Used for System Validation.** This implementation provided the most reliable electrical performance during Winter Quarter testing and served as the final validated hardware platform for functional evaluation and measurement accuracy analysis.

user workflows rather than provide a finalized visual design, resulting in a relatively unpolished appearance (see Figure 4). In addition, the backend analytics infrastructure required for graphical energy visualization remained under development, preventing the dashboard from displaying detailed historical usage metrics during the initial phase of the project.

### 7.2 Final Prototype Refinement

#### Hardware Refinement

Following the evaluation of the initial prototype, we introduced several major revisions to improve measurement accuracy, system reliability, and hardware integration. Experimental testing revealed that the clamp-current sensor and the ESP32 ADC produced unstable and inaccurate readings, making reliable power monitoring difficult under varying load conditions. To address these limitations, we redesigned the sensing architecture around a dedicated energy metering-IC capable of performing accurate voltage, current, and power calculations in hardware (see Figure 3).

We also redesigned the physical layout to reduce the overall system footprint and improve component integration through both perfboard and custom PCB implementations. Although we successfully designed and fabricated a PCB prototype, manufacturing and safety limitations prevented direct mains-level validation. As a result, we performed final system testing using the perfboard implementation under controlled low-voltage conditions (see Figures 9 and 10).

#### Software Refinement

The second development phase focused primarily on improving the frontend user experience and expanding the platform's data-analytics capabilities. We extended the backend infrastructure to compute energy statistics and support graphical data visualization, while simultaneously redesigning the user interface using updated UI/UX mockups and responsive layouts.

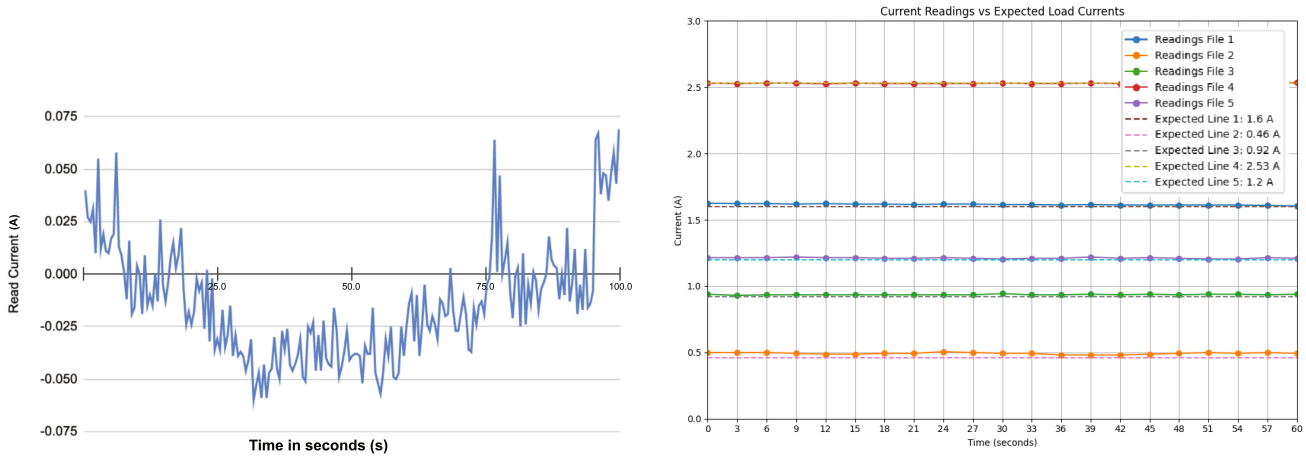
These revisions produced a significantly more polished and feature-complete dashboard with improved navigation, device management, and real-time monitoring capabilities (see Figures 4, 5, 6, and 7). The final software infrastructure supporting these features, including the cloud backend and device communication pipeline, is illustrated in Figure 1.

## 8 System Evaluation and Experimental Analysis

This section evaluates the performance of the *ZotPlug* prototypes and analyzes the design decisions that guided the transition from the initial sensing architecture to the final validated implementation. We first define the evaluation criteria used to assess measurement accuracy, then compare the performance of the Fall and Winter prototypes under experimental testing conditions.

### 8.1 Evaluation Methodology

Since *ZotPlug* is designed to provide users with meaningful insight into their energy usage patterns, the system does not require laboratory-grade precision. However, the measurements must remain sufficiently accurate for users to track long-term consumption trends and compare usage behavior across devices and users. Accordingly, we defined the primary success criterion as maintaining energy measurements within




(a) Fall Results

(b) Winter Results

Figure 11: **Comparative Measurement Accuracy Between Fall and Winter Hardware Implementations.** The Fall Quarter current-clamp-based design exhibited significant measurement instability and noise, whereas the Winter Quarter metering-IC implementation achieved substantially improved precision and consistency across multiple $12V$ AC load conditions.

approximately $\pm 5\%$ of reference measurements obtained using a calibrated multimeter.

## 8.2 Initial Prototype Evaluation

Experimental testing of the Fall prototype revealed significant inaccuracies in the clamp-current sensing architecture and the ESP32 ADC measurement pipeline (see Figure 11(a)). After calibration using the EmonLib library, we evaluated the stability of the readings by measuring a device drawing minimal current and recording the results over time. Even under these controlled conditions, the system exhibited an average noise offset of approximately $0.03A$, which produced substantial measurement error in low-power operating conditions, as demonstrated below.

Assume that an $18W$ LED lamp is connected to a $120V$ outlet. In this case, the expected current draw is approximately

$$I = \frac{P}{V} = \frac{18}{120} = 0.15A.$$

This means that an error of $0.03A$ would represent a current measurement error of 20%. The resulting instantaneous power error would be

$$P_{\text{error}} = 120 \times 0.03 = 3.6W.$$

If the lamp remained active continuously over a 24-hour period, the resulting energy error would be

$$E_{\text{error}} = 3.6W \times 24 = 86.4Wh = 0.0864kWh.$$

Since the expected energy consumption of an $18W$ lamp in this case would be

$$18W \times 24 = 432Wh = 0.432kWh,$$

the relative error would become

$$\frac{0.0864}{0.432} \times 100 \approx 20\%.$$

This result exceeded the target 5% accuracy threshold by a substantial margin. Furthermore, this experiment represented a best-case operating scenario with minimal system interference. Additional testing revealed that enabling WiFi and Bluetooth communication on the ESP32 further increased ADC instability and introduced larger fluctuations in the measured current values. These findings demonstrated that the clamp-current sensing architecture was not suitable for reliable outlet-level energy monitoring.

## 8.3 Final Prototype Validation

To address the limitations identified during Fall-quarter testing, the Winter prototype incorporated a

dedicated energy metering IC. We conducted validation testing using the perfboard implementation of the revised circuit (see Figure 3) powered by a safer $12V$ AC source. To evaluate measurement precision, we tested the system across electrical loads ranging from $0.46A$ to $2.53A$. In addition, we compared these readings with a calibrated multimeter and calculated the percent error for each measurement (see Figure 11(b)).

As shown in Table 2, all measurements satisfied the 5% accuracy criterion, while several operating conditions produced substantially lower error values. These results confirmed that integrating a dedicated metering IC significantly improved both the accuracy and stability of the *ZotPlug* hardware compared to the original sensing architecture.

| Load (A) | Measured (A) | Percent Error (%) |
|---|---|---|
| 0.46 | 0.47978 | 4.30 |
| 0.92 | 0.93699 | 1.85 |
| 1.60 | 1.608 | 0.50 |
| 2.53 | 2.53 | 0.04 |

Table 2: **Winter Prototype Accuracy Evaluation Under Variable Electrical Load Conditions.** This table summarizes the measured current values obtained during hardware validation testing and compares them against expected load values, demonstrating the high measurement accuracy achieved by the final metering-IC-based implementation.

# 9 Conclusion and Future Work

This paper presented the design and implementation of *ZotPlug*, a smart-plug platform that enables students to monitor and manage energy usage at the individual outlet level. The final prototype integrates embedded hardware, scalable cloud infrastructure, and a responsive web dashboard capable of providing real-time energy visualization and remote device control.

Experimental evaluation demonstrated that the system can provide reliable outlet-level energy monitoring for the purpose of increasing student awareness of energy consumption in dormitories. The results also showed that integrating a dedicated metering IC significantly improved measurement accuracy and system stability compared to the initial sensing architecture.

Future work could further extend *ZotPlug* into a fully deployable smart energy-management platform through production-ready PCB refinement, secure over-the-air firmware updates, enhanced electrical safety certification, and machine-learning-based energy analytics.

# 10 Acknowledgments

The authors thank Oscar Chinchilla for his valuable consultation and technical insight during the hardware development phase in the Winter quarter.

# References

[1] Kyle Anderson, Kwonsik Song, SangHyun Lee, Hyunsoo Lee, and Moonseo Park. Energy consumption in households while unoccupied: Evidence from dormitories. *Energy and Buildings*, pages 335–341, 2015.

[2] Noor Islam Jasim, Saraswathy Shamini Gunasekaran, Nouar AlDahoul, Ali Najah Ahmed, Ahmed El-Shafie, Mohsen Sherif, and Moamin A. Mahmoud. Toward sustainable campus energy management: A comprehensive review of energy management, predictive algorithms, and recommendations. *Energy Nexus*, 18:100435, 2025.

[3] Geoffrey Kavulya and Burcin Becerik-Gerber. Understanding the influence of occupant behavior on energy consumption patterns in commercial buildings. In *Proceedings of Computing in Civil Engineering*, pages 569–576, June 2012.