Presentation Script

IoT-Based Smart Energy Monitoring System Using ESP32

Dustin Bloss · ETEC-495 Seminar Project · Spring 2026 · Prof. Lazaros Pavlidis

---

SLIDE 1 — Title

"Hi, my name is Dustin Bloss, and this is my ETEC-495 Seminar Project presentation for Spring 2026. My project is the IoT-Based Smart Energy Monitoring System Using ESP32, advised by Professor Pavlidis."

Estimated time: ~30 seconds

SLIDE 2 — Project Overview

"The idea behind this project is simple — most people have no idea which appliances in their home are wasting electricity until they get their bill at the end of the month. My system fixes that by giving you real-time visibility into your energy usage. It measures voltage, current, and power, displays the readings live on an LCD screen, and sends the data to the cloud so you can check it from any browser."

Estimated time: ~45 seconds

SLIDE 3 — Hardware Components

"Let me walk through each component."

ESP32 Dev Board

"This is the brain of the entire system. A small microcontroller with built-in Wi-Fi. It runs all the code — reads every sensor, calculates power, drives the LCD, and pushes data to the cloud."

ACS712 Current Sensor (30A)

"Measures how much electrical current is flowing through a wire using a magnetic field — called the Hall effect. No need to cut the wire. It outputs a small voltage that the ESP32 reads on GPIO35 to figure out how many amps are flowing."

Voltage Divider (R1=100kΩ, R2=10kΩ)

"Just two resistors. The ESP32 ADC can only safely handle up to 3.3 volts, so the divider scales the input voltage down proportionally before it hits the ADC pin. Without it, you would damage the board."

PZEM-004T v3.0

"A dedicated energy metering module — far more accurate than a basic sensor. It measures AC voltage, current, active power in watts, and cumulative energy in kilowatt-hours. It talks to the ESP32 over a serial connection called UART."

20x4 LCD Display (I2C)

"The local display. Shows voltage, current, power, and watt-hours in real time, updated every 500 milliseconds. Only needs two wires — SDA and SCL — using a protocol called I2C."

Hi-Link HLK-5M05 (Power Module)

"The power supply. Plugs straight into 120 volt mains and outputs a clean 5 volts DC to power the ESP32 and all other modules. No separate wall adapter needed."

CT Clamp (Current Transformer)

"Clips around the live wire without cutting it. Detects the magnetic field from current flow and feeds a proportional signal into the ACS712 sensor."

Breadboard + Jumper Wires

"Used for prototyping — no soldering needed. Components press into holes to make connections. Color-coded jumpers — red for power, black for ground — keep wiring organized and readable."

Estimated time: ~90 seconds

SLIDE 4 — Circuit Build

"Here's the actual prototype. You can see the ESP32 on the breadboard in the center, the PZEM-004T energy module at the top, the CT current clamp sensor, and the 20-by-4 LCD display. Everything is wired up with color-coded jumpers."

Estimated time: ~30 seconds

SLIDE 5 — System Architecture

"The data flow is straightforward. The appliance load is measured by the ACS712 and PZEM-004T, which feed into the ESP32. The ESP32 calculates power and energy, sends it to the LCD over I2C, and uploads to ThingSpeak over Wi-Fi every 15 seconds where it can be viewed in a browser."

Estimated time: ~40 seconds

SLIDE 6 — Software & Firmware

"The firmware runs entirely on the ESP32. Step 1 reads voltage from GPIO34 through the voltage divider. Step 2 reads current from GPIO35 via the ACS712. Step 3 reads the PZEM-004T over UART. Step 4 calculates power as V times I, and accumulates energy in watt-hours. Step 5 pushes all of that to the 20-by-4 LCD every 500 milliseconds, and Step 6 uploads to ThingSpeak every 15 seconds. Libraries used include WiFi.h, ThingSpeak.h, LiquidCrystal I2C, and PZEM004Tv30."

Estimated time: ~60 seconds

SLIDE 7 — Development Journey

"The project started in weeks 1 and 2 with planning and confirming the ESP32 single-board approach with Professor Pavlidis. Week 3 I submitted the proposal and ordered parts. Weeks 5 through 7 I got the sensors wired up and got my first successful readings. Then in weeks 7 and 8 I hit the most difficult stretch — the ACS712 module caused an overcurrent event that fried the ESP32, the CT clamp was on the wrong wire, and the LCD screen was showing nothing because of a missing jumper. All three happened around the same time. Weeks 9 through 11 I rebuilt everything cleanly, got the LCD and ThingSpeak working, and confirmed the system was functioning end to end. The final weeks were testing, report writing, and this presentation."

Estimated time: ~60 seconds

SLIDE 8 — Issues & Solutions

"I ran into four real problems during this build. The first was that my ESP32 got fried by the ACS712 current sensor — the HW-670 module. Here is why that happened: the ACS712 runs on 5 volts, but the ESP32 ADC pins are only rated for 3.3 volts max. The output pin on the ACS712 can push voltage higher than what the ESP32 can safely handle, and if you connect it directly without stepping the voltage down first, you risk damaging or destroying the board. That is exactly what happened. I ordered a replacement ESP32, added a 1-amp inline fuse to the power rail, and made sure all sensor connections were verified against the datasheet before powering back on. The second issue was that my CT clamp was on the wrong conductor — it was on the neutral wire instead of the live wire, so the PZEM-004T was showing zero amps even with a load connected. I went back to the datasheet, moved the clamp to the live wire only, and it worked immediately. The third issue was the LCD screen — it powered on but showed absolutely nothing. Checked the I2C address, checked

all the wiring — everything looked correct. Turned out there was simply a jumper wire missing between the I2C backpack and the breadboard. Added one jumper and the screen came on. And the fourth was general wiring issues throughout the build — loose connections causing inconsistent readings. The fix was slowing down and verifying every connection against the wiring diagram before applying power at each stage."

Estimated time: ~90 seconds

SLIDE 9 — Testing & Results

"For final testing I used a 60-watt incandescent bulb as a known load. The expected voltage was 120 volts — I measured 118.3. Expected current was about half an amp — I measured 0.48 amps. Expected power was 60 watts — I measured 56.7 watts. All readings are within 5 percent of expected, the LCD refreshes every 500 milliseconds, and ThingSpeak updates every 15 seconds. Everything passed."

Estimated time: ~55 seconds

SLIDE 10 — Cost Analysis

"The total build cost came in at $86.41. The biggest single expense was the PZEM-004T at about $18, followed by the ESP32 boards — I had to buy two because of the one that got fried. The whole system costs a fraction of commercial smart energy monitors like the Sense, which runs $299 plus a monthly subscription."

Estimated time: ~40 seconds

SLIDE 11 — Time & Milestones

"Here you can see the Gantt chart alongside the latest prototype build. I originally planned 57 hours for the project and ended up at 88 hours — a 54 percent variance. The extra time was almost entirely from the unplanned troubleshooting caused by the fried ESP32 and the PZEM wiring issue. Everything else tracked close to the original plan."

Estimated time: ~40 seconds

SLIDE 12 — Social Impact

"For the social impact section I analyzed two markets. In the US, the target audience is the 66 million plus smart home households who want real-time energy data without a subscription fee. The DIY and open-source community on GitHub and Reddit is a natural fit.

For India, the pitch shifts to economics — high electricity costs, voltage fluctuation problems, and the government's national smart metering initiative all make this system directly relevant. The PZEM-004T also supports 230 volts out of the box, so no hardware changes needed."

Estimated time: ~60 seconds

SLIDE 13 — Conclusion

"To wrap up — I successfully built a real-time IoT energy monitoring system using an ESP32 as the sole controller. It measures voltage, current, power, and energy, displays live readings on a 20-by-4 LCD, and uploads to the ThingSpeak cloud over Wi-Fi — all for under $90 with no subscription. Going forward I'd like to add cloud-based historical logging, mobile push alerts for power spikes, and eventually a 3D-printed enclosure for a clean installation. Thank you."

Estimated time: ~45 seconds

Total estimated time: ~10 minutes