Slab Guardian: Technical Research Report

Real Time IoT Energy Threshold Monitoring for K Electric Protected Consumers

Prepared by the Slab Guardian team, UBIT, University of Karachi
Internal research document, compiled over the course of our literature review and hardware evaluation phase

---

Preface

This document collects everything our team researched while deciding how to build Slab Guardian. It covers the socioeconomic reasoning behind the project, the physics of how electricity is actually measured, the hardware and software options we considered, the math behind every calculation the device performs, the regulatory landscape we have to operate inside, and the competitive gap we identified in the Pakistani market. Nothing here is guesswork. Every claim is tied back to either a manufacturer datasheet, a published paper we reviewed, K Electric's own tariff documentation, or a calculation we worked through ourselves.

The goal of putting all of this in one place is so that any of the four of us, or anyone reviewing our FYP, can trace exactly why we made each engineering decision instead of just seeing the final answer.

---

1. Why This Problem Actually Matters

1.1 The 200 unit cliff, explained properly

K Electric runs a tiered consumer classification system regulated by NEPRA. A residential, non Time of Use customer who keeps their monthly consumption at or under 200 units (kWh) for six consecutive months is classified as a Protected consumer. This status carries a real subsidy on the per unit rate. The moment a household crosses 200 units, even by a single unit, two separate penalties kick in at once, and this is the part most people genuinely do not understand until it hits their bill.

The first penalty is immediate. The billing structure does not just charge you extra for the units above 200. It reclassifies the entire bill under the higher, Unprotected slab rate. That is why a household

consuming 190 units might pay somewhere around 2500 PKR, while the same household consuming 210 units can suddenly see a bill north of 6000 PKR. The jump is not proportional to the two extra units consumed. It is proportional to losing the subsidy on all 210 units.

The second penalty is the one that actually does the long term damage. Once a household breaches the 200 unit line, Protected status is revoked for the following six billing cycles, regardless of how carefully they manage consumption afterward. Even if the household drops back down to 120 units the very next month, they are billed at Unprotected rates for half a year. Our research estimates this cumulative penalty at roughly 20,000 PKR over that period, based on the recurring monthly gap between Protected and Unprotected rates at typical consumption levels.

1.2 Why this is an information problem, not a consumption problem

This is the core insight our whole project is built on. The households affected by this are not necessarily using more electricity than they used to. They simply have no way of knowing, mid cycle, where they stand. A KE meter is read once every 30 days. There is no daily or weekly feedback. So a family runs their AC a bit more during a hot week, their fridge cycles a bit harder, maybe a geyser gets left on longer than usual, and none of that registers anywhere the household can see until the bill arrives, by which point the damage from a slab breach is already locked in for six months.

We are calling this consumption blindness in our documentation, and it is the direct justification for why a real time dashboard has value that a once a month bill simply cannot provide. If a household can see, on day 15, that they have already used 110 of their 200 unit budget, they can adjust behavior for the remaining half of the cycle. That is the entire mechanism through which this device creates financial value for a user, and it only works if the number they are looking at is accurate and available in real time.

The daily budget math that anchors this whole product is simple: 200 units divided across a 30 day cycle gives roughly 6.67 kWh per day. Everything the app displays, the gauge, the days remaining counter, the alerts, is built around helping a user pace themselves against that daily number instead of discovering the problem only at the end of the month.

---

2. The Physics of Measuring AC Power (and why this is harder than it sounds)

2.1 Voltage and current are not enough on their own

Karachi's residential supply is standard 220V, 50Hz, single phase AC. The naive way to measure power would be to just multiply voltage by current. That approach is wrong for a huge portion of household loads, and understanding why is central to why we picked the hardware we picked.

AC voltage and current are both sine waves oscillating at 50 times a second. For a purely resistive load, like a simple heater or an incandescent bulb, those two waves rise and fall perfectly in sync with each other. Multiplying them at every instant and averaging gives you the real power correctly.

But most household appliances are not purely resistive. Refrigerators, air conditioners, washing machine motors, and fans all contain motors or electronic power supplies that behave as inductive or capacitive loads. These loads shift the current waveform out of phase with the voltage waveform. The current might lag slightly behind the voltage, or in some cases lead it. When that phase shift exists, multiplying instantaneous voltage and current no longer gives you the actual power being consumed. It gives you something inflated.

2.2 The actual formulas involved

To measure this properly, a device needs to sample both waveforms fast enough to reconstruct their shape, then run the following calculations.

RMS voltage and RMS current are calculated by squaring every sample, averaging those squares, then taking the square root:

V_rms = sqrt((1/n) times the sum of v_i squared for each sample)
I_rms = sqrt((1/n) times the sum of i_i squared for each sample)

Apparent power, measured in volt amps, is simply:

S = V_rms times I_rms

But apparent power is not what you get billed for. Real power, the number that actually shows up on your KE meter, factors in the phase relationship between voltage and current through the power factor:

P = S times PF

Power factor is a number between 0 and 1 that represents how in sync the current is with the voltage. A pure resistive load has a power factor close to 1. A motor heavy load might have a power factor of 0.7 or 0.8, meaning a meaningful chunk of the apparent power is not translating into real, billed consumption.

Finally, energy, which is what accumulates into your monthly unit count, is the integral of real power over time. In a digital system sampling at discrete intervals, this becomes a running sum:

E = the sum of (P average times the time interval) across every sample taken

This is the number that eventually needs to match K Electric's own meter to within a couple of units, because that number decides whether a household stays under 200 or not.

2.3 Why sampling rate genuinely changes your accuracy

One of the more concrete findings from the academic literature we reviewed (an IConGEET 2023 paper on IoT based energy monitoring) actually quantified this problem instead of just describing it. The researchers tested how sampling frequency affects RMS voltage accuracy on a pure sine wave and then again with realistic noise added.

With a clean sine wave, both a 200Hz and a 1000Hz sampling rate reproduced the true voltage with zero error, because a mathematically perfect sine wave is easy to reconstruct even from a handful of points. But once they introduced noise, which is what a real household circuit actually looks like, the story changed completely. At 200Hz sampling, the error jumped to almost 6 percent. At 1000Hz, it dropped to under 2 percent.

The same paper tested this on real loads. For a purely resistive heater, even a coarse 4 sample per cycle reading only produced about 3.84 percent error, because a heater's waveform is close to a clean sine wave anyway. But for an air conditioner, which is a genuinely nonlinear, motor driven load, 4 samples per cycle produced a staggering 16.92 percent error. That error only dropped to a more reasonable 1.81 percent once they sampled 178 points per cycle.

This single data point is why we did not seriously consider building our own raw ADC based sampling system for V1. Getting under 2 percent error on nonlinear household loads requires genuinely high sample rates and careful signal processing, and any mistake in that pipeline directly costs the user real money given how thin the margin for error is around the 200 unit cliff.

2.4 The Nyquist theorem, in plain terms

Underlying all of this is the Nyquist sampling theorem, which states that to accurately reconstruct a signal of a given frequency, you must sample it at more than twice that frequency. Since our mains frequency is 50Hz, the bare theoretical minimum is just above 100Hz. In practice, that theoretical minimum is nowhere near enough for accurate power measurement, because you are not just trying to detect that a wave exists, you are trying to accurately capture its shape, including higher frequency harmonics introduced by nonlinear loads like motors and switch mode power supplies. This is exactly why the paper above needed nearly 180 samples per cycle to get accuracy under 2 percent on an air conditioner.

---

3. Sensing Hardware, Option by Option

3.1 The build it yourself path: SCT-013 plus ZMPT101B plus raw ESP32 ADC

This is the path described in some of our early deep research notes, and it is worth documenting fully because understanding why we moved away from it is as important as understanding what we chose

instead.

Current sensing (SCT-013-000): This is a split core current transformer, meaning it clips around a wire without needing to cut or splice it. It works on Faraday's law of induction. The current flowing through the wire creates a magnetic field around it, and that changing magnetic field induces a proportional current in the transformer's internal winding. Because it never touches the copper conductor directly, this is the safest and only realistically permitted way to sense current in a residential distribution board without violating NEPRA's rules against tampering with wiring.

The SCT-013-000 variant does not have a built in burden resistor, meaning you have to size one yourself to convert its current output into a voltage the ESP32's ADC can read. The math for that, based on a similar academic source we reviewed, works out like this for a 100A rated clamp with a 2000 turn ratio:

Primary peak current equals RMS current times the square root of 2, so for a full 100A rating that is 100 times 1.414, which equals about 141.4A.

Secondary peak current equals that primary peak divided by the number of turns, so 141.4 divided by 2000 gives about 0.0707A.

To get good resolution on a typical Arduino style 5V ADC reference, you want the burden resistor to produce a voltage at peak current equal to half of that reference, so 2.5V. The ideal burden resistor value becomes 2.5 divided by 0.0707, which comes out to about 35.4 ohms, generally rounded down to a standard 33 ohm resistor to stay safely under the ADC ceiling.

Voltage sensing (ZMPT101B): This module steps down mains voltage to a safe level while providing galvanic isolation between the high voltage side and the low voltage side, which is a critical safety feature. It applies a DC bias, typically centered around 1.65V, so that the negative half of the AC waveform does not go below zero and get clipped by the ADC, which can only read positive voltages.

The ESP32 ADC problem: This is the part that made us hesitate the most on this path. The ESP32's internal ADC is a 12 bit SAR type converter, and it has documented, significant non linearity in two specific ranges, below about 0.15V and above about 2.45V. Since our voltage sensing setup is deliberately centering the waveform around the middle of that range to capture both halves of the AC cycle, the peaks and troughs of the wave are exactly where the ADC starts distorting the signal. That means the parts of the waveform carrying the most energy information are the parts being measured least accurately.

The two fixes for this are either building a software lookup table by manually characterizing your specific chip against a precision reference voltage and correcting readings in real time through interpolation, or bypassing the internal ADC entirely with an external 16 bit ADC like the ADS1115, which has 65,536 discrete levels compared to the ESP32's 4096, and comes with its own stable internal voltage reference immune to the ESP32's internal switching noise from its Wi-Fi radio.

Both fixes are viable for a team with more time and stronger embedded systems background, and this remains a completely reasonable path for a V2 that pursues appliance level detection, since that kind of

work requires access to the raw waveform shape anyway. But for V1, given our timeline, we decided this added too much research and debugging overhead for a problem that a purpose built chip already solves.

3.2 The path we chose: PZEM-004T V3.0

The PZEM-004T is a dedicated single phase AC energy metering module. Instead of handing you raw ADC samples that you have to process yourself, it does the entire RMS calculation, phase correction, power factor calculation, and cumulative energy integration internally, and simply hands you final numbers over a UART serial connection. It also includes its own precision voltage reference and current sensing pathway, so none of the ESP32 ADC non linearity issues described above even apply, since the ESP32 never touches the raw analog signal at all.

According to manufacturer specifications and the two actual proposal drafts our team wrote earlier this semester, this module achieves better than 98 percent measurement accuracy out of the box, which puts it well within our target margin for tracking a threshold as financially sensitive as the 200 unit line. The tradeoff is purely financial, it costs roughly 400 to 500 PKR more than sourcing a bare current sensor and doing the math yourself, and we consider that a completely worthwhile tradeoff given how much engineering time it saves and how much more reliable the resulting accuracy is.

It still uses the exact same SCT-013-000 clamp for current sensing on the input side, so the non invasive installation method and safety profile described above still applies. The module handles voltage sensing directly on its own AC line in and line out terminals, which is also where it derives its phase reference for the power factor calculation.

3.3 Why not ACS712

The ACS712 is a Hall effect based current sensor that comes up frequently in academic IoT energy monitoring projects, including some of the related work we reviewed. Its fundamental limitation for our use case is that it is an invasive sensor, meaning the wire has to be physically cut and the sensor inserted in series with the two ends. For a residential distribution board, this is both a bigger safety liability and closer to the kind of wiring modification that NEPRA's consumer service manual treats with more scrutiny than clip on, non invasive current sensing. Multiple academic projects using ACS712 with basic Arduino setups also reported accuracy errors in the 5 to 10 percent range, which given our earlier discussion about how a 2 unit error can cost a household 20,000 PKR, is simply not acceptable for this specific application.

3.4 The optical pulse counting alternative

We separately researched an entirely different sensing method that does not touch the household wiring at all. Nearly every KE digital meter has a small blinking LED that flashes a fixed number of times per unit of energy consumed, commonly 3200 pulses per kWh on single phase residential meters,

though this exact figure should be confirmed against the label on the specific meter in question since some meters use different pulse rates. By placing a light sensor, either a simple LDR or a faster responding phototransistor, directly over that LED and counting flashes, you get a number that will match K Electric's own billing exactly, because you are literally reading the same signal the meter uses internally to calculate the bill.

This approach requires a comparator circuit, commonly built around an LM393 chip, to convert the slow analog light level swing into a clean digital pulse the ESP32 can count via an interrupt. It also requires physically shielding the sensor from ambient light so that a passing shadow or a room light turning on does not register as a false pulse. This method has real precedent in the hobbyist and home automation world, projects like Home Assistant's Glow, and commercial products like the Watz WiFi Counter, are built on exactly this principle.

We are treating this as a documented secondary validation option rather than a core V1 requirement. It adds genuine value as an independent cross check against drift, since it is mathematically guaranteed to agree with the KE meter, but it does not provide any of the real time voltage, current, or power factor data that NILM or detailed diagnostics would eventually need, and it adds another subsystem to build, wire, and explain without adding a new feature the user actually sees.

---

4. The Microcontroller Layer

4.1 Why ESP32 over Arduino or ESP8266

The ESP32 is a dual core microcontroller running at 240MHz with integrated Wi-Fi and Bluetooth built directly onto the chip. We compared this against two realistic alternatives.

A basic Arduino board, like an Uno or Nano, has no built in wireless connectivity at all, meaning we would need to add a separate Wi-Fi shield or module, increasing both cost and points of failure. It also runs a single core at a much lower clock speed, which becomes a real constraint once you are simultaneously trying to read a sensor on a tight timing loop while also managing a network connection.

The ESP8266, which is actually what the academic paper in our research set used, is a legitimate lower cost alternative with built in Wi-Fi, but it is single core and has meaningfully less RAM than the ESP32. For a project that intends to eventually explore on device machine learning inference for appliance detection in a future version, that headroom matters. The ESP32's second core also lets us deliberately separate time sensitive sensor sampling from Wi-Fi and MQTT publishing tasks, which matters a great deal in a city where the network connection itself might be unreliable due to load shedding related router restarts, without that unreliability ever being allowed to cause us to miss or corrupt a sensor reading.

4.2 LittleFS and why persistent storage is not optional

This is arguably the single most Karachi specific engineering decision in the entire project. Load shedding in our context is not an occasional inconvenience, it happens multiple times a day in many areas. Any design that keeps its running unit count purely in the microcontroller's RAM will lose that count on every single power interruption, silently resetting to zero and under reporting the household's true consumption for that cycle. Given that the entire product exists to help a household track their position relative to a 200 unit ceiling, a device that can forget its own count during the exact conditions it was built for is worse than having no device at all, because it creates false confidence exactly when the risk of a slab breach is highest.

LittleFS is a lightweight filesystem that lives directly on the ESP32's own internal flash memory, requiring no external SD card or additional hardware. The design decision we settled on is to write the cumulative unit count to flash only when it crosses a full new whole unit, rather than on every single sensor reading, since flash memory has a finite number of write cycles and writing on every reading, potentially every second, would wear it out far faster than necessary. On boot, the firmware reads the last saved value back out of flash before doing anything else, so a restart resumes exactly where it left off instead of starting from zero.

---

5. Communication and Backend Architecture

5.1 Why MQTT instead of plain HTTP requests

MQTT is a publish subscribe messaging protocol designed specifically for small, resource constrained devices operating over potentially unreliable networks, which describes our exact situation. Instead of the ESP32 opening a full HTTP connection every time it wants to send a reading, which is comparatively heavy for a small microcontroller and less resilient when the network itself is flaky, MQTT keeps a lightweight persistent connection open to a broker and simply publishes short messages to a named topic whenever new data is ready. Our backend subscribes to that same topic and receives every message as it arrives.

We are running Mosquitto as our broker, which is a widely used, open source MQTT implementation with a large amount of documentation and community support, making it a low risk choice for a team building this kind of pipeline for the first time.

5.2 Why InfluxDB instead of a standard relational database

Nearly every core feature in this product revolves around time windowed queries. How many units in the last 7 days. How many in the last 15. How does today compare to yesterday. A standard relational database like MySQL can technically answer these questions, but you would have to hand build the indexing and aggregation logic yourself, and it was never designed with time as a first class concept the way InfluxDB is.

InfluxDB is a time series database, meaning every data point is stored with a timestamp as its primary organizing structure, and its query language is built around expressing exactly the kind of range based aggregation our dashboard needs. This lets us get fast, clean 7, 10, and 15 day queries essentially for free instead of writing custom date range logic that would need to be optimized separately as our data grows.

5.3 Why Node.js for the backend

Our backend server exists to subscribe to the MQTT topic, write incoming readings into InfluxDB, run our threshold and alert logic, and expose a REST API that the Flutter app can call. Node.js was chosen mainly because its ecosystem has mature, well documented libraries for both MQTT client behavior and JSON handling, and JSON is the natural format for both our MQTT payloads and our API responses, meaning there is very little translation friction between layers. This was as much a practical choice based on what tooling fits together cleanly as it was a deep technical requirement, since a Python or PHP backend could technically accomplish the same job.

5.4 Firebase for authentication and push notifications

We chose not to build our own user authentication system or our own push notification infrastructure from scratch, because both of those are solved problems with significant security and reliability considerations that add no unique value to what makes Slab Guardian different from any other app. Firebase Authentication handles account creation and login securely without us having to manage password hashing or session security ourselves, and Firebase Cloud Messaging lets our backend trigger a push notification to a user's phone the moment their consumption crosses the 50, 75, or 90 percent thresholds we track, even if the app itself is closed at the time. This is what actually makes the product proactive instead of just being another dashboard someone has to remember to open.

---

6. The Mobile Application Layer

6.1 Why Flutter

Flutter lets a single codebase compile to both Android and iOS, which matters enormously given our team only has one dedicated mobile developer and a five month timeline. Building and maintaining two fully separate native codebases was never realistic for a team our size, and Flutter's widget based architecture, combined with strong community tutorials in Urdu and Hindi that our mobile developer is already working through, made it the practical choice.

6.2 What the interface actually needs to communicate

The single most important design principle for the app is that a user should be able to understand their situation at a glance, without reading numbers or interpreting a graph. This is why the color coded gauge, green for safely on pace, yellow for approaching the budget, red for likely to exceed 200 units by the end of the cycle, is the centerpiece of the home screen rather than a raw kWh figure. Detailed engineering data like voltage, current, and power factor readings are reserved for an advanced or secondary view, since that level of detail does not help the average household make a daily decision about whether to run the AC a little less.

---

7. Machine Learning and Appliance Level Detection (Researched for V2, not built in V1)

7.1 What NILM actually is

Non intrusive load monitoring is the practice of taking a single aggregate power signal at the main entry point of a house and algorithmically figuring out which individual appliances are contributing to that total, without needing a separate sensor on every device. Every appliance has a distinct load signature, sometimes called a power fingerprint, based on how its current draw behaves when it turns on, runs, and turns off.

Our research identified three broad categories of appliance behavior that NILM systems have to handle differently.

Simple on and off devices, like a water pump or a toaster, show up as clean, discrete step changes in real power and are the easiest category to detect, sometimes reliably identifiable through basic step change detection algorithms alone, without any machine learning at all.

Multi state devices, like washing machines that cycle through wash, rinse, and spin phases each drawing different amounts of power, are harder to track with simple thresholding and generally require a statistical model like a Factorial Hidden Markov Model to properly follow the state transitions.

Continuously variable devices, like inverter air conditioners or dimmable lighting, are the hardest category by far, since their power draw shifts smoothly rather than in discrete steps, and reliably identifying them typically requires deep learning approaches.

7.2 Why we are not attempting this in V1

Reliable NILM requires either a large, high quality labeled dataset of appliance signatures, which does not really exist for common Pakistani household appliances like local compressor equipped refrigerators from brands such as Dawlance or PEL, since almost all published NILM datasets are built around Western appliances with entirely different electrical signatures, or it requires access to raw high

frequency waveform data that our PZEM-004T does not expose, since it only gives us summary values. Attempting appliance level detection in V1 would mean either abandoning our accurate, low risk sensing approach or trying to build an entirely separate raw sampling pipeline in parallel, both of which would put our core accuracy goal at risk. This is explicitly reserved for a V2, where a sequence to point learning approach, which trains a model to predict a single target appliance's consumption from a window of aggregate power data, appears to be the most memory efficient method for eventually running on ESP32 class hardware through TensorFlow Lite for Microcontrollers.

---

8. Regulatory and Safety Research

8.1 What NEPRA and K Electric actually permit

Our research into the K Electric Consumer Service Manual and NEPRA's regulatory framework indicates that monitoring consumption on the consumer's own side of the meter using non invasive sensors, such as our clip on current transformer, is generally treated as a permitted energy management practice, distinct from meter tampering, which specifically refers to interfering with utility owned equipment like the meter itself. This distinction is exactly why our entire hardware design commits to a non invasive installation method as a hard requirement rather than a preference. Any solution that required cutting or splicing the household's main service wire would move into much riskier regulatory and safety territory, and would likely face real consumer rejection regardless of its technical merits, since asking someone to modify their own home wiring is a significant trust barrier.

8.2 Physical safety requirements we are designing around

Any device placed inside a residential distribution board needs to be housed in a non flammable, non conductive enclosure. The current transformer clamp must never damage the insulation on the wire it clips around. The device's own power supply should be on a dedicated, isolatable circuit so it can be safely serviced without disturbing the rest of the household's electrical system. Karachi's grid also produces frequent voltage transients specifically during load shedding transitions, which is why our design includes metal oxide varistors to shunt voltage spikes, optoisolators to keep any mains side disturbance from reaching the microcontroller's logic circuitry, and decoupling capacitors placed close to sensitive pins to filter out high frequency electrical noise.

---

9. Competitive Landscape

Our review of existing solutions available in Pakistan found a consistent gap. Commercial smart plugs like TP Link Kasa monitor individual appliances rather than whole house consumption, require a stable internet connection to function at all, and are priced in a range that puts them out of reach for the exact

income bracket most affected by the 200 unit cliff. OpenEnergyMonitor and EmonPi, an open source whole house monitoring approach using a similar current clamp method to ours, requires a Raspberry Pi, which is comparatively expensive and not always easy to source locally, and was designed for technically proficient hobbyists in Western markets with no built in awareness of KE style slab billing at all. Existing Pakistan specific research into smart metering infrastructure, meanwhile, focuses almost entirely on the utility side of the equation, helping KE manage the grid, rather than giving the household itself any usable, real time information.

We did not find any existing product that combines whole house non invasive sensing, genuine resilience against frequent load shedding through local persistent storage, an alert system built specifically around the 200 unit KE threshold, and a total cost low enough to be realistic for a middle or lower income household, all inside a single mobile first product. That combination is the actual gap Slab Guardian is built to fill.

---

10. Calibration Methodology (planned)

Our calibration approach has two layers. The first is component level calibration, verifying that our specific PZEM-004T unit, paired with our specific SCT-013 clamp, reports accurate values when tested against a known load, such as an appliance with a clearly labeled wattage like an electric kettle or heater, before it ever goes into a real household. The second is field level calibration, running the completed device continuously alongside the actual K Electric meter in a real home over a period of roughly two weeks, logging both readings daily, and calculating cumulative deviation over that window. Our target, based on how thin the margin for error is around the 200 unit threshold, is to stay within 2 units of the official KE reading across that full field test period. Any consistent offset we observe gets investigated for its root cause, whether that is a wiring issue, a clamp placement issue, or a genuine calibration constant that needs adjusting, rather than simply being corrected with an arbitrary fudge factor without understanding why the discrepancy exists in the first place.

---

11. Summary of Key Decisions and Why

Decision area	What we chose	Core reason
Current sensing	SCT-013-000 split core clamp	Non invasive, NEPRA compliant, safe to install without cutting wires
Power measurement	PZEM-004T V3.0	Handles RMS, phase correction, and power factor internally, avoids ESP32 ADC non linearity entirely, over 98 percent accurate out of the box

---

Decision area	What we chose	Core reason
Microcontroller	ESP32	Dual core lets us separate sensing from networking, built in Wi-Fi, enough headroom for a future V2
Local persistence	LittleFS on internal flash	Survives Karachi's frequent load shedding without losing the running unit count
Messaging protocol	MQTT via Mosquitto	Lightweight, built for constrained devices on unreliable networks
Time series storage	InfluxDB	Purpose built for the 7, 10, and 15 day windowed queries our whole dashboard depends on
Backend	Node.js and Express	Clean fit with JSON heavy MQTT and API data, mature MQTT libraries
Auth and notifications	Firebase	Avoids reinventing secure authentication and push infrastructure from scratch
Mobile app	Flutter	One codebase for both Android and iOS, realistic for a one person mobile team
Appliance level detection	Deferred to V2	Needs a raw waveform data source and a local appliance dataset we do not yet have

---

Closing Note

Everything above represents the actual reasoning trail behind our hardware and software choices, not just a list of parts. Where we found published data, like the sampling frequency error figures from the IConGEET 2023 paper, we used it directly to justify a decision instead of relying on assumption. Where we found a gap, like the total absence of KE specific NILM datasets, we documented it honestly as a limitation reserved for future work rather than something we glossed over. This is the version of the research we are standing behind for our proposal defense.