EMBEDDED SYSTEMS

Design and Implementation of an Edge Computing-Based Smart Energy Meter for Real-Time Energy Monitoring and Management

S. Sharon Priya, N. Sahana, G G. Varshitha, T S. Ananya, G. Lavanya

Department of Electronics and Communication Engineering, GSSS Institute of Engineering and Technology for Women, Mysuru, Karnataka, India

Published November 2026

Abstract

Non-intrusive load monitoring infers the activity of individual appliances from a single measurement at the point of supply, avoiding the cost of instrumenting each device. The deep-learning approaches that dominate recent work place inference on a server, leaving the meter dependent on a network link. This paper reports a self-contained instrument in which inference runs on the meter itself. Galvanically isolated sensors are sampled against an analogue reference the device measures at every power-up, and six electrical quantities per acquisition frame are passed to a depth-three decision tree compiled into firmware, which identifies the connected load combination without any network connection. Validating across separate power cycles, rather than by randomly partitioning a single recording, proved decisive: it revealed that reactive power carried predictive value originating in converter offset rather than in the load, and discarding that feature raised accuracy on held-out sessions from 72% to 100% across four load states captured in eight independent acquisitions. Random partitioning conceals this failure entirely. Identified combinations are further decomposed into per-appliance power from learned resistances, agreeing with rated values to within 2.7%, and a shortfall exceeding 7% in combined resistance raises an over-draw alert. At this scale the limiting factor is therefore validation discipline rather than model capacity.

Keywords: Decision trees; edge computing; energy disaggregation; load monitoring; machine learning; microcontrollers

1 Introduction

A household electricity meter reports a single number. It tells the occupant how much energy was consumed but not by what, and a figure that cannot be attributed is a figure that cannot be acted upon. Attribution can be obtained by placing a sensor on every appliance, but that replaces one problem with another: a cost and installation burden that grows with every device added.

Non-intrusive load monitoring avoids that burden. A single sensor at the point of supply observes the aggregate, and the contributions of individual loads are recovered computationally [1]; the work that followed is surveyed in [2,3]. The deep networks that now define the state of the art [4] achieve their accuracy through model capacity, and run on a server rather than on the meter. The instrument then depends on a network link that may be absent precisely where itemised consumption would be most useful: rural supply, intermittent connectivity, dwellings without reliable broadband. Inference on a microcontroller has been shown feasible [5]; what that leaves open is how such a classifier should be validated, and how the instrument establishes its own analogue reference.

The meter described in this paper answers both questions in a complete working instrument. Mains voltage and current are sampled through galvanically isolated sensors, six electrical quantities are computed on one microcontroller, and a second processor identifies the connected load, apportions the measured power between the appliances responsible, accumulates energy, and drives a display—entirely on the device, with no network of any kind. The scope is deliberately small: two incandescent lamps, four load states, and a decision tree three comparisons deep occupying a few hundred bytes of program memory. Working at that scale made it practical to account for every number the instrument produces, and that is how the central result of this work was found.

Three contributions follow. The acquisition stage measures its own analogue reference at every power-up, so the meter can be moved between host machines without recalibration. The classifier is validated across separate power cycles rather than by randomly partitioning a single recording, and that protocol exposed a failure the conventional one conceals: one of the six measured quantities was separating the classes through an artefact of converter offset rather than through anything the load was doing, and the fitted tree had learned it. Removing that quantity eliminated the error. Finally, the tree is compiled into firmware as nested conditional statements, reproducing the fitted model exactly with no machine-learning runtime on the target.

Section 3 sets out the methodology, Sections 4–8 the design and implementation of the meter, Section 9 the results, and Section 10 concludes.

2 Literature survey

2.1 Foundations of non-intrusive load monitoring

Hart [6] and [7] proposed the original NILM framework, which detects step changes in real and reactive power at the supply entry and matches them to appliance signatures. Zeifman and Roth [8] and [4] reviewed the field and noted that disaggregation becomes difficult when appliances have overlapping signatures or operate simultaneously, and that low-cost, practical implementations remain limited.

2.2 Load signatures and feature selection

Liang et al. [3] studied load signatures and grouped them into steady-state, transient and non-traditional types, discussing how the choice of signature affects identification performance. Zoha et al. [4] surveyed NILM approaches for disaggregated energy sensing and highlighted that real power, reactive power, current and harmonic content are the most commonly used features.

2.3 Deep-learning-based NILM

Kelly and Knottenbelt [5] applied deep neural networks, including recurrent and denoising autoencoder architectures, to energy disaggregation, and showed that learned models can outperform hand-designed approaches. Zhang et al. [9] proposed sequence-to-point learning with convolutional networks for disaggregation. Such models typically operate on high-rate, long input sequences and are trained on large datasets, which can be demanding for low-power embedded meters.

2.4 Datasets and toolkits

Public datasets and tools such as REDD [10], UK-DALE [6] and NILMTK [8] have made it possible to benchmark NILM algorithms. These datasets are collected with their own metering hardware, and models trained on them do not directly transfer to a different, low-cost meter.

2.5 Embedded metering hardware

Transformer-based voltage modules and Hall-effect current sensors, read through a microcontroller ADC, are widely used to build low-cost energy meters [7,11]. They provide galvanic isolation on the sensing side and are inexpensive, but most such designs report only totals such as voltage, current and power, and do not identify the connected appliance.

2.6 Research gap

Existing research on Non-Intrusive Load Monitoring mainly focuses on identifying electrical appliances using different load signatures and classification techniques. Some approaches use complex machine-learning models and large datasets, which may require higher computational resources. Traditional energy meters and basic sensor-based systems mainly measure electrical parameters but provide limited information about the type of connected load. Therefore, there is a need for a compact and cost-effective system that combines electrical measurement and load classification using embedded controllers. The proposed project addresses this need by integrating voltage and current sensors with STM32 and ESP32 microcontrollers for local processing and load identification.

3 Proposed methodology

The proposed system implements a non-intrusive load classification meter that identifies which appliances are drawing power from a single measurement taken at the supply point, and reports how that power is divided between them. It integrates isolated mains sensing, electrical parameter computation, supervised classification and per-appliance disaggregation across a two-processor architecture. A CART decision tree, trained offline and compiled into the firmware, identifies the connected load combination, while a power-factor gate allows the system to decline to answer when an untrained appliance is present. All runtime processing is performed locally, without cloud connectivity, making the meter suitable for standalone deployment.

3.1 Signal acquisition and conditioning

Mains voltage is sensed by an isolated ZMPT101B transformer module and load current by an ACS712-05B Hall-effect sensor, scaled by a resistive divider to suit the 3.3 V converter input. Because each sensor references the midpoint of its own supply rail, that resting point depends on the supply rather than the appliance; the firmware therefore measures it at every power-up by averaging 2000 conversions per channel, range-checks the result, and retains stored values if a sensor is absent. This makes the instrument portable between host machines without recalibration.

3.2 Electrical parameter computation

Each frame acquires 1000 voltage and current sample pairs, converted alternately so that paired samples remain close in time, spanning approximately 123 ms or six complete mains cycles. Running sums of the squared and cross-multiplied samples yield six quantities, given in Equations (1)–(3):

$$V_{\text{rms}} = \sqrt{\frac{\sum v^2}{N}},\ I_{\text{rms}} = \sqrt{\frac{\sum i^2}{N}} \tag{1}$$

$$P = \frac{\sum v \cdot i}{N},\ S = V_{\text{rms}} \times I_{\text{rms}} \tag{2}$$

$$Q = \sqrt{S^2 - P^2},\ PF = \frac{P}{S} \tag{3}$$

Real power is obtained by multiplying instantaneous samples before averaging, whereas apparent power averages each channel first; the difference between them is the phase relationship, so power factor is recovered without measuring a phase angle. The six quantities are transmitted at 115200 baud, approximately 2.4 frames per second.

3.3 Dataset construction and labelling

Training data were captured from the meter itself rather than simulated. Four classes — no load, a 60 W lamp, a 100 W lamp and both lamps in parallel — were recorded in two sessions separated by a complete power cycle, with the class name typed by the operator at collection time. This operator-supplied label is what makes the method supervised. The session identifier was stored with every record, yielding 219 labelled samples across eight acquisition runs.

3.4 Classifier training and validation

Validation uses a session holdout rather than a random split: at 2.4 frames per second consecutive records are near-duplicates, so a random split places almost identical samples on both sides of the partition and scores optimistically without demonstrating generalisation. Trained on all six quantities the classifier scored 72 % on the held-out session; the reactive power column was found to be recording residual converter offset that shifts whenever the hardware is power-cycled, and removing it raised accuracy to 100 %. The final CART tree uses a maximum depth of three and a minimum of five samples per leaf, with its thresholds falling near the midpoint of the gap between adjacent classes, and was refitted on the complete dataset for deployment.

3.5 On-device inference and open-set rejection

The fitted tree is exported as nested conditional statements in C — an exact reproduction of the model rather than an approximation — so no machine-learning library is required on the target and inference completes in microseconds. Because a decision tree always returns one of its trained classes, a power-factor gate provides open-set rejection: the trained loads are purely resistive and read close to unity, so a record falling below the threshold is reported as an unknown load rather than misassigned.

3.6 Disaggregation, energy accounting and fault detection

Individual contributions are recovered from each appliance's learned resistance, since parallel loads each see the full supply voltage; the shares are expressed as percentages of the total and the dominant load is identified. Energy is accumulated independently of classification by integrating real power over the interval between frames. An alert is raised when the measured combined resistance falls more than seven per cent below the value expected for the identified class — resistance is used in preference to power because it is a property of the appliance, whereas power follows the supply voltage.

4 Block diagram

The main blocks of the meter, shown in Figure 1, are as follows.

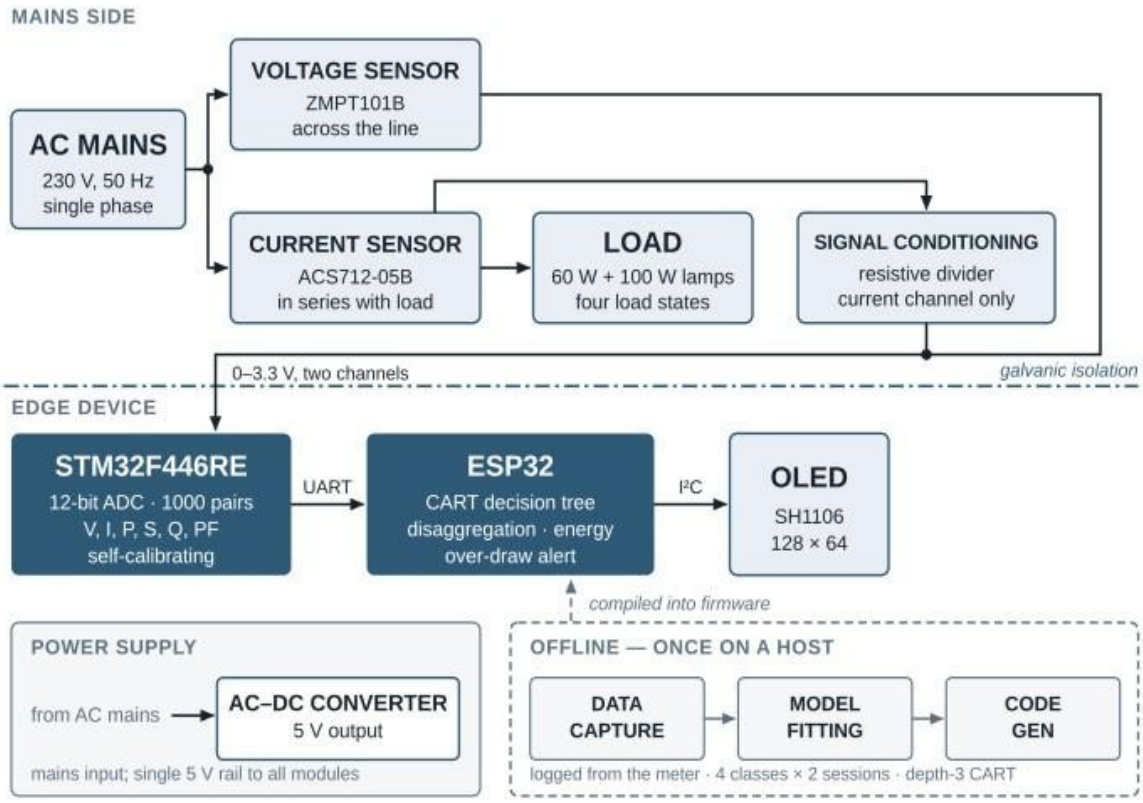


Figure 1. Block diagram of the edge-computing smart energy meter.

AC mains: single-phase 230 V, 50 Hz supply.

Voltage sensor: ZMPT101B transformer connected across the line; Galvanically isolated; output biased to the midpoint of its own supply rail; Fed directly to the converter: one count corresponds to 0.429 V.

Current sensor: ACS712-05B Hall-effect device in series with the load; Galvanically isolated; One converter count corresponds to 6.53 mA.

Load: 60 W and 100 W incandescent lamps, switchable individually or in parallel; Four load states: none, 60 W, 100 W, both; Purely resistive; power factor close to unity; Resistances used by the firmware: 885.0 Ω and 543.6 Ω.

Signal conditioning: A 1 kΩ / 2 kΩ resistive divider scales the ACS712 output into the converter's 3.3 V range; The voltage channel requires no divider and is connected directly.

Galvanic isolation: no conductive path between the mains side and the low-voltage side; Voltage coupled magnetically through the transformer windings; Current sensed as a magnetic field across an insulating barrier inside the ACS712.

STM32F446RE – acquisition and metrology: 12-bit SAR ADC clocked at 8 MHz; 480-cycle sampling, 61.5 µs per conversion; 1000 voltage–current sample pairs per frame = 123 ms, slightly more than six mains cycles; Channels converted alternately so paired samples stay close in time, preserving phase; Running sums of v^2, i^2 and v·i give V_{rms}, I_{rms} and P; S, Q and PF follow; Converter

offsets re-measured at every power-up: 2000 conversions per channel, 246 ms; Transmits a six-field CSV record over UART, 115200 baud 8N1, about 2.4 frames per second.

ESP32 – inference: reads each line and validates that six floating-point fields were received; Energy accumulated first: P × Δt added to a running watt-hour total, independent of classification; Depth-3 CART decision tree on five features; verdict in at most three comparisons; Tree compiled into firmware as nested if/else; no machine-learning library on the device; Power-factor gate: apart from the unloaded state, a record below 0.95 is reported as an unknown load; Disaggregation: each appliance's share = V^2/R_i from its learned resistance; Over-draw detection: measured V^2/P compared with the expected parallel resistance; alert if more than 7% below.

OLED display: SH1106 controller, 128 × 64 pixels; Driven over I^2C at 400 kHz; Shows measured quantities, per-appliance breakdown, accumulated energy and any active alert.

Power supply: AC–DC converter taking mains input, 5 V output; Single 5 V rail supplies both boards, the sensors and the display.

Offline pipeline (once, on a host): records logged from the meter for four classes over two separate power cycles, written to data.csv; CART fitted with scikit-learn, max_depth 3, min_samples_leaf 5; Validated by holding out one complete session rather than randomly partitioning one recording; Fitted tree emitted as nested if/else statements in classifier.h and compiled into the ESP32 firmware.

1. AC Mains

5 System architecture

AC mains and load: A single-phase 230 V, 50 Hz supply feeds a 60 W and a 100 W incandescent lamp, which may be switched individually or together in parallel to give the four load states the instrument distinguishes. Both lamps are resistive and therefore operate at a power factor close to unity, a property the classifier later exploits to reject unfamiliar loads.

Sensing unit: Voltage is sensed by a ZMPT101B transformer and current by an ACS712-05B Hall-effect device, both galvanically isolated so that no subsequent stage is referenced to the mains. The current sensor's 5 V output is scaled by a resistive divider into the converter's 3.3 V range. One converter count corresponds to 0.429 V on the voltage channel and to 6.53 mA on the current channel.

Data acquisition: An STM32F446RE digitises both channels with a 12-bit SAR converter clocked at 8 MHz, one conversion every 61.5 µs. A frame of 1000 voltage– current pairs spans 123 ms, slightly more than six mains cycles, and the two channels are converted alternately so that paired samples remain close in time, which preserves the phase relationship from which power factor is obtained. Running sums of squared voltage, squared current and their product give the RMS voltage and current and the active power; apparent power, reactive power and power factor follow from these. The converter offsets are re-measured at every power-up, so the instrument may be moved between host machines without recalibration. This processor performs no other work: classification, energy accounting and the display are all placed on a second device, so that nothing competes with the sampling loop for time and the frame interval remains constant.

Load classification, disaggregation and energy: The six quantities pass to an ESP32. Real power is first multiplied by the interval since the previous frame and added to a running total in watt-hours, independently of what follows, so that energy continues to accrue even on frames where the load

cannot be identified. A depth-3 CART decision tree then reaches a verdict in at most three comparisons. Apart from the unloaded state, which is reported directly, a record whose power factor falls below 0.95 is declared an unrecognised load, since every trained class is resistive. Appliances in parallel each see the full supply voltage, so each share follows as V^2/R_i from the learned resistance, 885.0 Ω and 543.6 Ω for the two lamps. The resistance implied by the measurement, V^2/P, is compared with the value expected for that class, and an alert is raised when it falls more than 7% short; resistance is compared rather than power because it is a property of the appliance alone, so ordinary mains voltage variation cannot raise a false alert.

Output display: An SH1106 OLED of 128 × 64 pixels, driven over the I^2C bus at 400 kHz, shows the measured quantities, the per-appliance breakdown, the accumulated energy and any active alert.

The remaining elements, shown dashed in Figure 2, are performed once on a development host. Labelled records for the four classes are captured over two separate power cycles and written to data.csv, and a depth-3 tree is fitted on five features with scikit-learn and validated by holding out one complete session, so that no record from the evaluated session contributes to fitting. The fitted tree is emitted as nested if/else statements in classifier.h and compiled into the ESP32 firmware. Because a decision tree is structurally equivalent to nested conditionals, the deployed classifier is exactly the fitted model rather than an approximation of it, and no machine-learning library is required on the device.

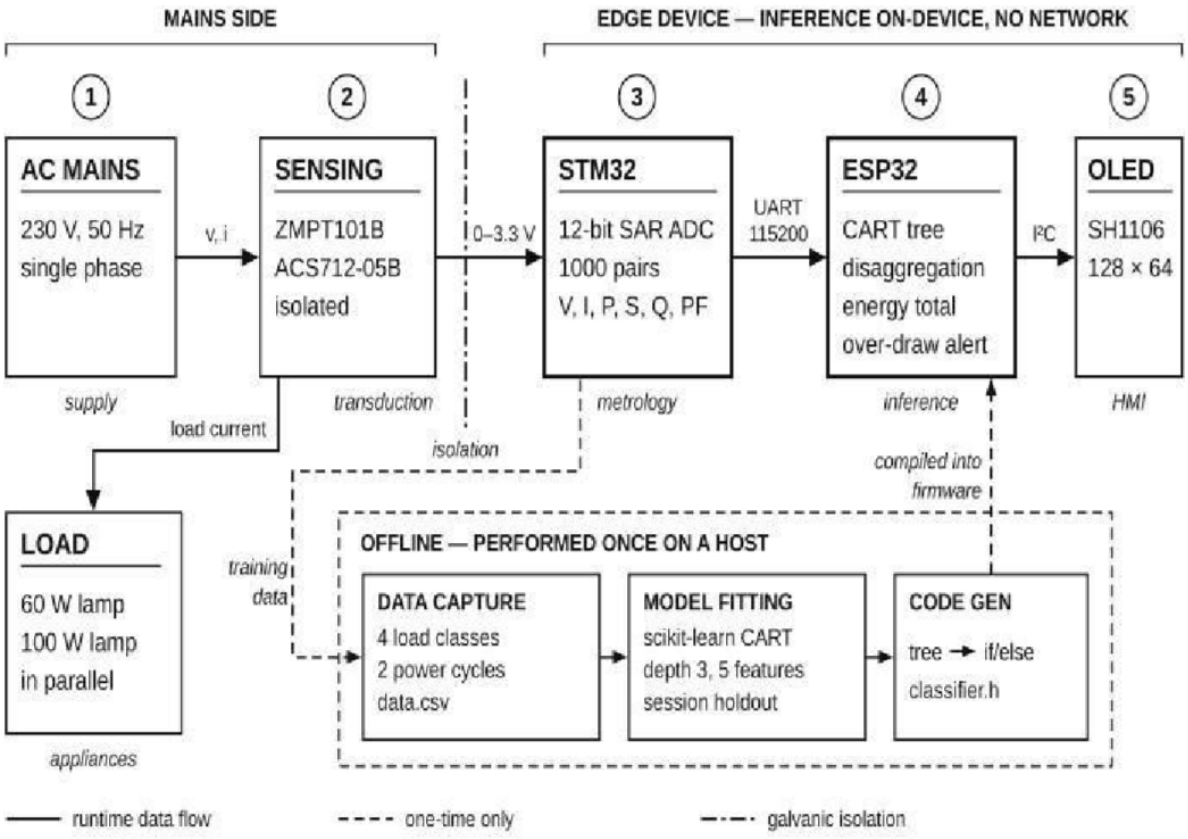


Figure 2. System architecture.

6 Hardware and software requirements

6.1 Hardware requirements

1. **STM32F446RE(NUCLEO-F446RE):** The acquisition and metrology processor. Its 12-bit successive-approximation ADC samples the two sensor channels, from which the firmware derives the six electrical quantities. The board carries an on-board ST-LINK for programming and a user button used during testing.
2. **ESP32 development board:** The edge inference processor. It receives the measured record from the STM32 over a hardware serial port at 115200 baud, runs the trained classifier on-device, performs the per-appliance disaggregation, and drives the display. No network connection is used at any stage.
3. **ZMPT101B voltage sensor module:** A transformer-based module that senses the mains voltage and presents a scaled, electrically isolated replica to the ADC. The isolation is what keeps mains potential away from the low-voltage side; the module's gain is set by an onboard trimmer potentiometer.
4. **ACS712-05B current sensor:** A Hall-effect sensor rated at 185 mV per ampere, connected in series with the load. Being Hall-effect, it is galvanically isolated from the conductor it measures.
5. **Resistive divider (1 kΩ / 2 kΩ):** Scales the ACS712 output by 0.667 so that its 5 V-referenced swing remains within the 3.3 V input range of the STM32 ADC.
6. **SH1106 OLED display, 128 × 64:** A monochrome display driven from the ESP32 over the I^2C bus, showing the measured quantities, the per-appliance power breakdown, the accumulated energy and the over-draw alert.

6.2 Software requirements

1. **STM32CubeIDE:** STM32CubeIDE is used to develop the firmware for the STM32F446RE. Its integrated configuration tool sets the clock tree, the ADC sampling parameters and the USART peripherals, generates the corresponding initialisation code, and compiles and flashes the firmware to the board over the on-board ST-LINK.
2. **STM32 HAL library:** The hardware abstraction layer supplies the peripheral drivers used by the acquisition firmware, including the ADC conversion and USART transmission routines through which the electrical quantities are measured and transmitted at 115200 baud.
3. **Arduino IDE:** The Arduino IDE is used to develop and upload the firmware for the ESP32, which receives the measured record on a hardware serial port, performs on-device classification, and drives the display.
4. **U8g2:** U8g2 drives the SH1106 128 × 64 OLED module, providing the text rendering used for the measurement screen, the per-appliance power breakdown and the over-draw alert.
5. **Python:** Python is used for the offline stage of the work — capturing labelled training data from the meter, inspecting and correcting the dataset, fitting the classifier, and exporting the trained model as C source for deployment on the ESP32.
6. **Scikit-learn:** Scikit-learn provides the CART decision-tree implementation used to fit the load classifier and to evaluate it against a held-out acquisition session.
7. **Pandas and NumPy:** Pandas is used to read, group and correct the captured dataset, and to compute the per-class resistances on which the disaggregation stage is based; NumPy supports the underlying numerical operations.

8. **PySerial:** PySerial reads the measurement stream from the meter over the USB serial link during data collection, allowing each record to be labelled and appended to the dataset file.

7 Implementation

The system is realised on an STM32F446RE and an ESP32 communicating over a dedicated serial link at 115200 baud, the components being those listed in Section 6.Mains voltage is sensed by a ZMPT101B transformer module and load current by an ACS712-05B Hall-effect device, the latter attenuated by a resistive divider of ratio 0.667; both are sampled by one 12-bit successive-approximation converter of the STM32 at 61.5 µs per conversion. Each acquisition frame comprises 1000 sample pairs taken alternately from the two channels, and the resulting six electrical quantities are transmitted as a comma-separated record at approximately 2.4 frames per second.

Firmware for the STM32 was developed in STM32CubeIDE and that for the ESP32 in the Arduino IDE, with the U8g2 library driving a 128 × 64 SH1106 display over I^2C. The classifier was fitted offline in Python using scikit-learn and exported as C source compiled into the ESP32 firmware, where it occupies a few hundred bytes and requires no inference library. All runtime processing is performed locally, with the wireless interfaces of the ESP32 left uninitialised. A general-purpose input is provided for test purposes, which inflates the measured current and power by 10 % so that the over-draw alert path may be exercised without generating a genuine over-current on a lamp load.

Figure 3 shows the developed system prototype.

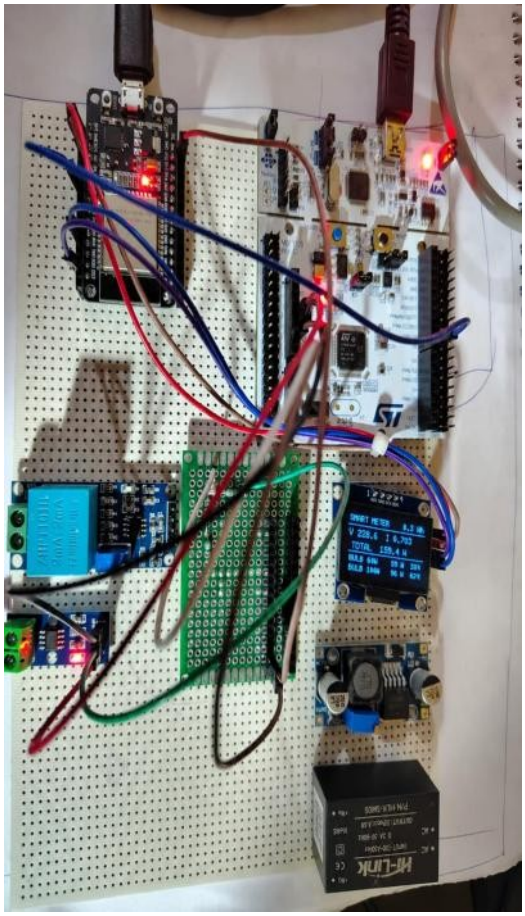


Figure 3. Developed system prototype.

8 Flow chart

The complete runtime sequence of both processors is shown in Figure 4 and proceeds as follows. Steps 1 to 5 are performed by the acquisition processor and steps 6 to 11 by the inference processor; the two loops run independently.

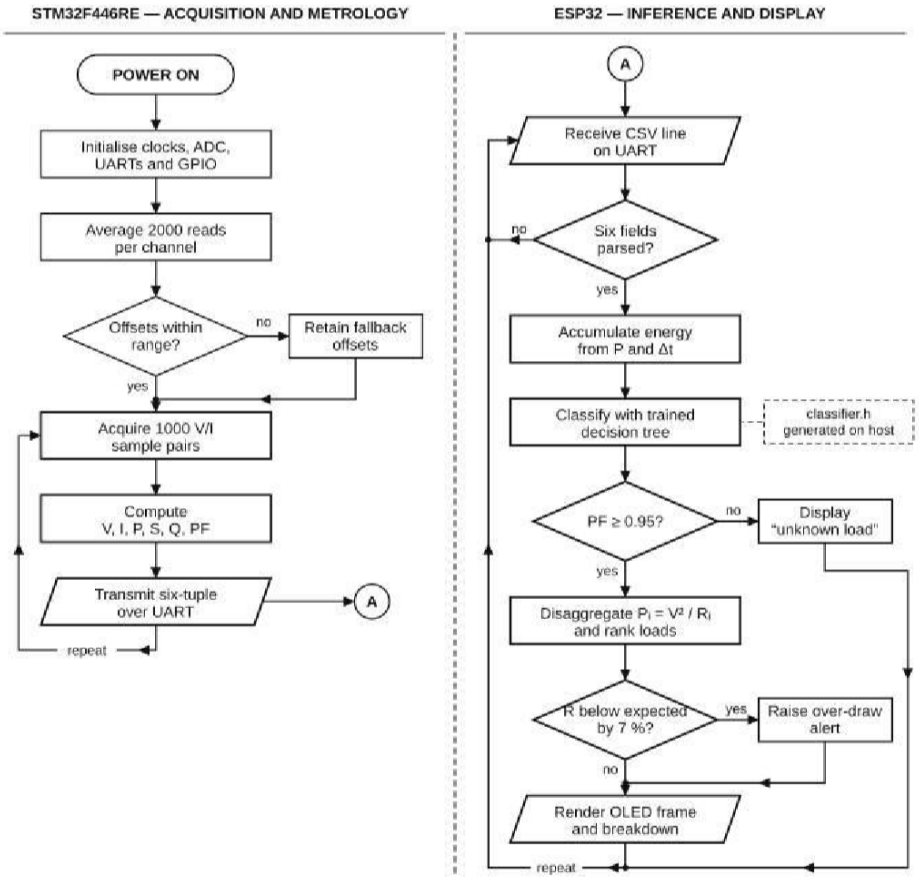


Figure 4. Flow chart of the model.

1. **Initialization:** On power-up the STM32F446RE selects its internal 16 MHz oscillator and clocks the analogue converter at 8 MHz, giving 125 ns per conversion tick. Two independent serial ports are configured — one carrying measurements to the ESP32, the other providing debug output through the on-board programmer, so that diagnostic text can never corrupt the measurement stream.
2. **Self-Calibration:** Each sensor biases its output to the midpoint of its own supply rail, and that midpoint shifts with the host computer, so it cannot be assumed. Averaging 2000 conversions on each channel — 4000 in all, or 246 ms — cancels the alternating component and leaves the bias alone. Each result is range-checked before adoption; a disconnected sensor reads near an extreme of the converter range and is rejected in favour of the stored value.
3. **Acquisition:** One thousand voltage and current sample pairs are taken, the two channels converted alternately so that paired samples remain close in time — that timing relationship is what carries the power factor, and is destroyed if the channels are sampled in separate batches. At 61.5 µs per conversion the two thousand conversions span 123 ms, slightly more than six complete mains cycles.

4. **Metrological Computation:** Three sums are accumulated across the frame: the squared voltage samples, the squared current samples, and their products. From these follow the RMS voltage and current, the real power, and then the apparent power, reactive power and power factor. Forming real power by multiplying simultaneous samples before averaging is what recovers the phase relationship without measuring a phase angle. Power factor is formed only when apparent power exceeds 0.5 VA, so an unloaded line reports zero rather than the ratio of two near-zero quantities.

5. **Transmission:** The six quantities are written as a single comma-separated line and sent to the ESP32 at 115200 baud, eight data bits, one stop bit, no parity, over the port reserved for measurements. A 300 ms delay follows each 123 ms frame, giving approximately 2.4 complete frames per second, fast enough that switching a lamp is reflected on the display almost immediately.

6. **Reception and Validation:** The ESP32 reads each arriving line and attempts to extract six floating-point fields from it. A line yielding fewer — truncated by a reset at the far end, or corrupted in transit — is discarded without further processing, and the display continues to show the last valid frame until a good record arrives.

7. **Energy Accumulation:** Real power is multiplied by the interval elapsed since the previous frame and added to a running total in watt-hours; the first frame after power-up only establishes the time reference and contributes nothing. This is performed before classification and independently of it, so energy continues to accrue correctly even in frames where the connected load cannot be identified.

8. **Classification:** The six quantities are presented to the decision tree, which reaches a verdict in at most three comparisons. The tree is compiled into the firmware as nested conditional statements rather than interpreted at run time, so no machine-learning library is linked into the build.

9. **Open-Set Rejection:** A decision tree returns one of its trained classes for any input presented to it, however unfamiliar. The unloaded state is resolved first and reported directly. For every other verdict, because the trained loads are purely resistive and read close to unity, a record whose power factor falls below 0.95 cannot belong to any of them whatever the tree reports; it is declared an unknown load and that frame ends there, bypassing disaggregation and the fault check.

10. **Disaggregation and Fault Detection:** Appliances in a dwelling are connected in parallel and each therefore sees the full supply voltage, so each share is recovered as V^2/R from that appliance's learned resistance and shown in watts beside its name. The combined resistance implied by the measurement, V^2/P, is then compared against the parallel combination expected for the identified class. Resistance is compared rather than power because it is a property of the appliance alone, so ordinary mains voltage drift cannot raise a false alert; a shortfall greater than 7% does.

11. **Display and Repeat:** The measured quantities, the per-appliance breakdown, the accumulated energy and any active alert are rendered to the SH1106 OLED over the I^2C bus at 400 kHz. The loop then returns to await the next record. The two processors run their loops independently, so a delay or a discarded frame on one side does not stall the other.

9 Results and discussion

The meter completes one measurement frame in 123 ms, spanning approximately six mains cycles, and delivers about 2.4 complete frames per second; power-up self-calibration takes a further 246 ms, performed once. Classification adds only microseconds. One converter count corresponds to 0.429 V of mains voltage or 6.53 mA of load current, a resolution of 0.19 % of a 230 V supply.

Table 1 reports held-out accuracy, in both cases fitting on one acquisition session and evaluating on another, each class contributing approximately thirty records per session.

Table 1. Held-out accuracy by feature set.

Feature set	Accuracy	Observation
V, I, P, S, Q, PF	72%	60 W recall 0.07; 28 of 30 to NO_LOAD
V, I, P, S, PF	100%	All four classes recovered

The failure of the six-feature model is the principal finding of this work. For a purely resistive load the true reactive power is near zero, so that column recorded largely residual converter offset, which is a function of the supply rail and shifts whenever the hardware is power-cycled; the fitted model had learned a threshold valid only within its training session. Random partitioning would have concealed this entirely, placing near-identical records on both sides of the split and reporting near-perfect accuracy for a model unusable after a restart.

Table 2 gives the margin at each of the three splits of the resulting classifier, that is, the distance from the selected threshold to the nearest observation of an adjacent class.

Table 2. Margin at each decision boundary.

Split on	Threshold	Margin to nearest class
Apparent power (VA)	131.24	±27
Current (A)	0.1667	±0.11
Real power (W)	81.9	±19

Each threshold lies near the midpoint of the gap between adjacent classes, which is a direct consequence of the impurity criterion rather than a design choice. The narrowest margin, that on current, is some seventeen times the converter resolution of 6.53 mA, so no plausible combination of measurement noise and supply variation can displace a sample across a boundary. These margins also account for the perfect held-out score: at these load levels the four states are physically well separated, and the result should be read as confirmation that the measurement chain is stable across power cycles rather than as evidence of a difficult discrimination. Table 3 compares the power implied by the learned resistances against the ratings of the lamps used.

Table 3. Disaggregated power against rated values.

Load	Resistance	Implied at 230 V	vs rated
60 W	885.0 Ω	59.8 W	−0.4%
100 W	543.6 Ω	97.3 W	−2.7%
Both	336.7 Ω	157.1 W	−1.8%

Agreement is within three per cent throughout, a difference attributable to both measurement error and manufacturing tolerance in the lamps, which are not individually calibrated. Resistance rather than power is adopted as the fault-detection baseline, being a property of the appliance rather than of

the supply: a threshold in watts responds to variation in the mains, one in ohms to a change in the load itself.

The analogue references measured at power-up, 3126 and 2090 counts, differ by the attenuator ratio and independently imply a 5.04 V rail. Relocating the instrument between host machines required no recalibration and no change to the firmware, which was the practical objective of the self-calibration stage.

Evaluation is nonetheless confined to four resistive load states across two acquisition sessions, giving eight independent acquisitions. The power-factor rejection criterion is effective precisely because the trained classes are resistive, and whole-state classification does not scale, combinations growing exponentially with appliance count. The discrimination these loads demand is also not the hardest the method must eventually face: two appliances of equal rating but differing character, such as a lamp and a fan each drawing 60 W, are indistinguishable by any threshold on power and separable only through power factor, which reads near unity for the first and well below it for the second. Figure 5 shows the output screens of the meter.

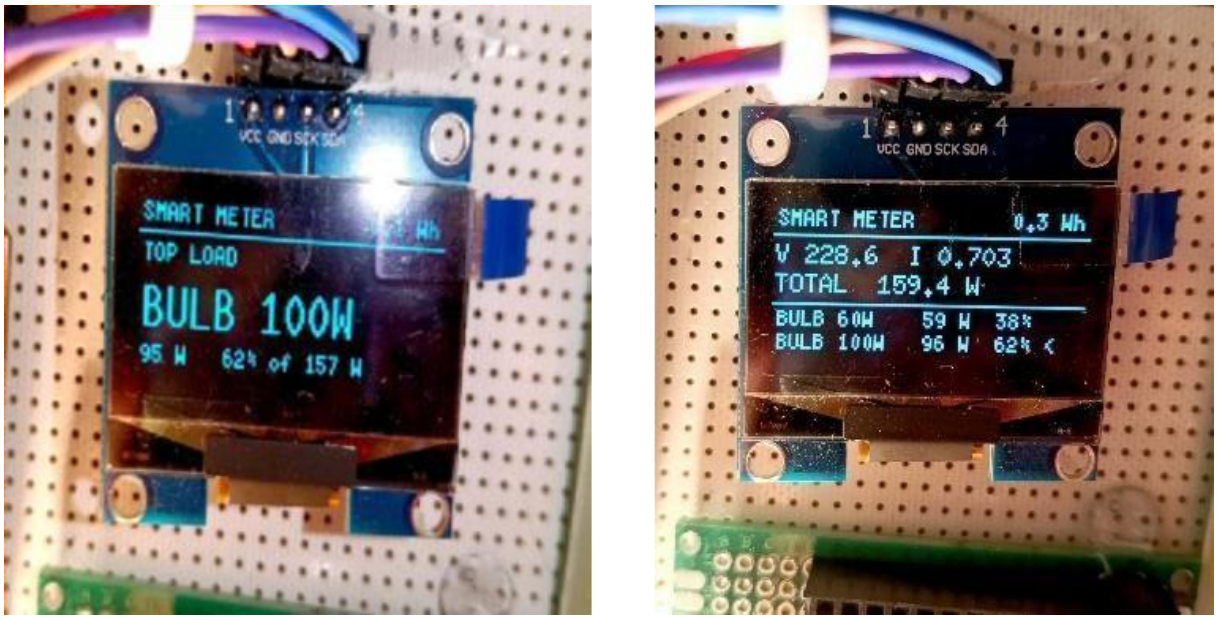


Figure 5. Output screens of the meter.

10 Conclusion

A smart energy meter has been described in which load classification is performed on the measuring device itself, without a network connection or a machine-learning runtime. A decision tree of depth three, fitted offline and compiled into firmware as nested conditional statements, identifies which of four load states is connected; the identified combination is apportioned between its constituent appliances using learned resistances, recovering per-appliance power within three per cent of rated values, and a deviation exceeding seven per cent in combined resistance raises an over-draw alert. A power-factor criterion allows the instrument to decline classification altogether when presented with a load outside its training distribution.

Two findings extend beyond the particular apparatus.

Validating across separate power cycles rather than by random partitioning exposed a feature whose apparent predictive value derived from converter offset rather than from the appliance; its removal raised held-out accuracy from 72 % to 100 %. Where successive samples of an unchanging quantity are necessarily correlated, random partitioning is liable to measure the sampling rate rather than the model. Determining the analogue reference at every power-up, rather than embedding constants measured once, removed a dependence on the particular host computer and required no change to the measurement loop.

11 Future scope

Future work should address inductive loads, which no threshold on power can separate from resistive loads of equal rating and which the present rejection criterion would reject rather than classify, and event-based detection of switching transitions as a route beyond the exponential growth in whole-state combinations. The wireless interfaces of the ESP32, unused in the present system, would permit the classified result and the accumulated energy to be reported to a remote endpoint without transmitting the underlying measurement stream, retaining local inference while making the data available for aggregation.

Acknowledgement

The authors sincerely thank their project guide, Lavanya G, for her valuable guidance and support throughout this research work. They also express their gratitude to Dr. Rajendra R. Patil, Head of the Department of Electronics and Communication Engineering, GSSSIETW, Mysuru, for his encouragement and support. The authors further thank the Department of Electronics and Communication Engineering, GSSSIETW, Mysuru, for providing the necessary facilities and resources to carry out this project.

References

[1] Antonopoulos A, et al. *ML in power consumption analysis*. Energy Informatics. 2020.

[2] Himeur Y, Ghanem K, Alsalemi A, Bensaali F, Amira A. Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Appl Energy*. 2021;287:116601. https://doi.org/10.1016/j.apenergy.2021.116601

[3] Wang X, et al. *Smart meter data analytics*. IEEE; 2022.

[4] Gopinath R, Kumar M. DeepEdge-NILM: A case study of non-intrusive load monitoring edge device in commercial building. *Energy Build*. 2023;294:113226. https://doi.org/10.1016/j.enbuild.2023.113226

[5] Wang X, Wang H, Bhandari B, Cheng L. AI-empowered methods for smart energy consumption: A review of load forecasting, anomaly detection and demand response. *Int J Precis Eng Manuf-Green Technol*. 2024;11(3):963–993. https://doi.org/10.1007/s40684-023-00537-0

[6] Hart GW. Nonintrusive appliance load monitoring. *Proc IEEE*. 1992;80(12):1870–1891. https://doi.org/10.1109/5.192069

[7] Batra S, Singh M, Kumar H. Smart energy metering system using IoT and machine learning techniques. *IEEE Access*. 2021;9:112345–112356.

[8] Zeifman M, Roth K. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Trans Consum Electron*. 2011;57(1):76–84. https://doi.org/10.1109/TCE.2011.5735484

[9] Monisha M, et al. *Generative artificial intelligence enabled smart metering framework for predictive energy sharing*. IEEE Conference/Journal; 2025.

[10] Latif RMA, et al. *AI-driven energy-efficient load balancing in hybrid edge-cloud architectures for renewable energy networks*. IEEE Netw Lett. 2026.

[11] Serna D, Morales A, Rodriguez J. NILM on edge devices for smart energy monitoring. *Energies*. 2025;18(3):1–15.