

# Resource-Efficient Edge TinyML for Real-Time Occupancy Estimation and Energy-Aware Control in Smart Hostel Rooms

<sup>1</sup>Rachit Arora, <sup>2</sup>Saksham Kumar, <sup>3</sup>Rudra Singh

<sup>123</sup>Department of Information Technology, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India

\*\*\*

**Abstract** - Hostel and dormitory buildings waste significant electricity because fans, air-conditioners, and lights are left on fixed schedules or simply forgotten, whether or not anyone is in the room. This paper builds an edge-based Tiny Machine Learning (TinyML) system that infers real-time room occupancy from a low-cost, multi-sensor node (PIR motion, CO<sub>2</sub>, ambient sound, ambient light), with inference running entirely on a microcontroller so the system needs no live cloud connection. Four lightweight classifiers (decision tree, logistic regression, k-nearest-neighbours, and a single-hidden-layer MLP) are compared on a seeded synthetic sensor dataset reflecting patterns from the occupancy-sensing literature, evaluated over 30 repeated stratified splits with accuracy, F1-score, precision, recall, specificity, and false-negative rate reported as mean  $\pm$  SD with 95% confidence intervals, alongside confusion matrices and a five-way sensor-ablation study. The Tiny MLP is selected as the deployment model, reaching 93.7%  $\pm$  0.6% test accuracy; quantized to an 8-bit TensorFlow Lite Micro model, it yields a genuine, measured 2,280-byte deployment artifact. Flash and RAM figures (28,417 / 1,732 bytes) come from a real ESP32-S3 firmware build, not a host-machine estimate; latency and power remain projections from comparable published measurements. A week-long simulation suggests a simulated energy reduction of 40.5%  $\pm$  0.8% against an always-on baseline, but a sensitivity analysis shows this depends heavily on that baseline assumption, essentially disappearing against a more realistic, semi-adaptive schedule. The paper also covers hardware/deployment choices, privacy considerations, a novelty comparison against related TinyML occupancy work, and next steps for field validation.

**Key Words:** TinyML, edge computing, occupancy prediction, smart buildings, energy efficiency, embedded machine learning, smart hostel, IoT sensors

## 1. INTRODUCTION

### 1.1 Motivation and Problem Statement

Higher-education campuses run large hostel complexes housing hundreds to thousands of students, and once cooling, lighting, and appliance loads are added up, these blocks account for a sizeable share of campus electricity use [27]. Energy audits of hostel buildings tell a consistent

story: a large share of that consumption is avoidable, with fans, air-conditioners, and lights left running while the occupant is in class, at the mess hall, or away for the weekend, simply because switching off depends on someone remembering to do it rather than on the room enforcing it [12]. Most occupancy-aware building automation research targets commercial offices, but hostel rooms follow a different rhythm (class timetables, mess timings, curfews, exam schedules), rarely have a centralized building-management system to retrofit onto, and are someone's private living space, not an open office.

Existing approaches fall into two camps, neither of which fits well. A single passive-infrared (PIR) sensor into a threshold or timer controller is cheap but known to misread a stationary occupant as an empty room, since PIR detects motion, not presence [3]. More accurate machine-learning approaches instead assume raw sensor streams can be shipped to a cloud or on-premise server [5], [6]; scaled to a hostel with hundreds of rooms, that multiplies network load, creates a single point of failure, recurs in compute cost, and in some published systems relays audio or video continuously out of a private bedroom. This paper asks a narrower question: can occupancy be estimated accurately enough for practical energy control using only cheap, privacy-preserving sensors, motion counts, CO<sub>2</sub>, ambient sound level, and ambient light, none of which can capture identity or conversation content, paired with a classifier small enough to run entirely on the microcontroller reading the sensors? If so, no occupancy data ever has to leave the room.

### 1.2 Why Edge-Based TinyML

TinyML designs and deploys machine-learning models that fit inside the kilobyte-scale memory and milliwatt-scale power budgets of microcontrollers, typically via frameworks such as TensorFlow Lite for Microcontrollers [7], sitting at the device-level end of the broader edge-computing picture [25]. Recent TinyML deployments run quantized models in well under a millisecond on hardware costing a few hundred rupees, at single-digit milliwatts of active power [10], [11], [21]. This pays off in four ways for a hostel: inference happens locally, so a dropped Wi-Fi connection never disables the fan/light controller; latency stays low enough for responsive control; power draw is low enough to run for months on a small battery or be harvested from the circuit it switches; and no continuous

stream of raw sensor data has to go anywhere, which matters for what is, after all, a resident's own bedroom.

### 1.3 Contributions

This paper's contributions are fourfold. First, a complete edge-based system architecture for TinyML occupancy prediction and actuation, built around the retrofit constraints and occupancy rhythms of a shared hostel room. Second, a comparison of four microcontroller-appropriate classifiers on the axes that actually determine embedded feasibility, accuracy, inference latency, and memory footprint, showing the most accurate model is not necessarily the one worth deploying. Third, a week-long simulation, grounded in published occupancy-sensing patterns, quantifying the potential simulated energy reduction and comfort trade-offs of occupancy-aware control against always-on habits, together with a sensitivity analysis over every load-bearing assumption. Fourth, a treatment of the privacy, calibration, and retrofit challenges a real deployment would face, with a concrete field-validation agenda.

## 2. RELATED WORK

### 2.1 Occupancy Sensing and Machine Learning for Building Energy

Occupancy sensing has been approached through many modalities. PIR sensors remain the most widely deployed option because they are cheap, but a recent review documents their well-known blind spot, missing stationary occupants, and surveys machine-learning approaches that fuse PIR with other signals to patch it [3]. CO<sub>2</sub> concentration offers a complementary proxy, since exhaled CO<sub>2</sub> accumulates in an enclosed room, though it responds more slowly than motion sensing because gas concentration lags entry or exit [5], [23]. Fusing signals works: PIR with smart-meter load data through a CNN outperformed either signal alone in an office setting [6], and CO<sub>2</sub> fused with PIR via a hidden-Markov-style estimator cut PIR's false-vacancy problem [4]. Device-free radio-frequency sensing is a further option, WiFi channel-state information reported occupancy-detection rates above 94% [15], and passively captured WiFi/BLE frames have been used similarly [16], though both depend on occupants carrying networked devices, which a purely environmental sensor cluster sidesteps. On the modeling side, a comprehensive review of ML-based occupancy prediction concludes model choice should track deployment constraints, not just benchmark accuracy [1]; random forests beat regression baselines for the related problem of building energy-load prediction [24], favouring tree-based methods as an interpretable default, in line with the decision tree evaluated here; and a broad review of occupancy-based HVAC/IEQ control finds savings of roughly 10-40% relative to non-adaptive schedules are

typical, though the exact figure depends heavily on how wasteful the replaced baseline was [2], the range this paper's own estimate is measured against.

### 2.2 TinyML and Edge AI

TinyML runs inference inside a microcontroller's kilobyte-scale memory, the innermost tier of the broader edge-computing hierarchy mapped out in [25]. TensorFlow Lite for Microcontrollers showed quantized networks can run on tens of kilobytes of RAM with no OS or dynamic allocation, setting the toolchain this paper builds on [7]; an early industry report argued offline inference is a first-class edge-AI requirement, not an optimization [8]. A systematic survey of 26 TinyML studies reports sub-millisecond inference latencies and singles out quantization as the most reliably effective way to shrink a model to a microcontroller's flash budget [9], and a 2025 study measured a three-to-four-times storage reduction from 8-bit post-training quantization with minimal accuracy loss [10]. Broader model-compression surveys agree quantization, pruning, and knowledge distillation are the three dominant technique families [20]; a 2025 study combining structured pruning with quantization reported roughly fifteen-fold size reductions with well under one point of accuracy loss [22], and a separate survey covers the deployment tooling that turns compressed models into firmware [21]. Closest to the present work, a 2026 study deployed a TinyML occupancy model directly on an ESP32 using CO<sub>2</sub>, temperature, humidity, and light sensors, reporting sub-millisecond on-device inference [11]; this paper follows a similar sensor-fusion, on-device-inference approach but targets the hostel-room energy-control problem specifically, adding a multi-model accuracy-versus-footprint comparison and an energy-reduction simulation.

### 2.3 Hostel and Campus Energy Studies, and Research Gap

Energy audits of hostel and university residential buildings keep pointing to the same culprit, lighting and fan/AC circuits left running in empty rooms, with occupancy-based switching recommended as the fix, though the audits diagnose the problem rather than propose a sensing technology [12]. Low-cost commodity sensors can support credible indoor-environment readings once carefully calibrated [26], and campus-scale energy-management reviews flag predictive, occupancy-aware control as a recurring recommendation [27]. Taken together, this literature establishes that multi-sensor fusion beats single-sensor occupancy estimation, occupancy-aware control can meaningfully cut energy use, and TinyML makes on-device inference practical, but it has not looked closely at hostels: occupancy and building-energy work has focused on offices and generic residential buildings, and TinyML-for-occupancy work has so far cared

more about single-room technical feasibility than a joint accuracy/latency/memory comparison for a budget-constrained retrofit setting. This paper closes that gap by pairing a hostel-specific system design with a systematic four-classifier comparison and a quantified, sensitivity-tested energy simulation.

## 2.4 Positioning Against Closely Related TinyML Occupancy Work

Table -1 makes this gap concrete against the studies this paper is closest to. The comparison is deliberately

narrow, not a ranking of which paper is "better," since each targets a different building type and question, but a check that this paper's combination, hostel-specific framing, a joint accuracy/latency/memory sweep across four classifier families, an explicit sensor-ablation study, and an energy-control simulation with a documented sensitivity analysis, is not a relabelling of an existing result. No prior study in this comparison combines all five columns; the closest, [11], shares the sensor-fusion-plus-on-device-inference approach but targets a generic room and reports neither a multi-model comparison, a sensor-ablation study, nor an energy-control simulation.

**Table -1:** Positioning against closely related TinyML and occupancy-sensing studies

| Study                           | Multi-sensor fusion            | Hostel/residential-specific       | Multi-model comparison | Sensor ablation study | Energy-control simulation      |
|---------------------------------|--------------------------------|-----------------------------------|------------------------|-----------------------|--------------------------------|
| This paper                      | Yes (PIR+CO2+sound+light+temp) | Yes (hostel room)                 | Yes (4 classifiers)    | Yes (5-way)           | Yes, with sensitivity analysis |
| Pandkar et al. [11], 2026       | Yes (CO2+temp+humidity+light)  | No (generic room)                 | No (single model)      | No                    | No                             |
| Wu et al. [6], 2024             | Yes (PIR+smart meter)          | No (office)                       | No (single CNN)        | No                    | No                             |
| Jiang et al. [4], 2018          | Yes (CO2+motion)               | No (office)                       | No                     | No                    | No                             |
| Bhushan et al. [10], 2025       | No (vision only)               | No (generic edge node)            | No                     | No                    | No                             |
| Zhang et al. [1], 2022 (review) | N/A (review)                   | No (offices/residential, general) | N/A (review)           | N/A                   | N/A                            |

Note: A check mark indicates the property is a central part of that study's contribution, not merely mentioned in passing.

## 3. SYSTEM ARCHITECTURE

### 3.1 Design Goals

Five design goals shape the architecture, drawn from Section 1 and the hostel energy-audit literature [12]. Per-room cost must stay low enough for a hostel-scale rollout, in keeping with the low-cost sensing philosophy for building monitoring [26]. Power draw must be low enough for a small battery, power bank, or circuit-harvested energy, matching milliwatt-scale TinyML sensor-node budgets [9], [10]. The system must be retrofit-friendly, needing no rewiring beyond an inline relay. It must keep working offline, so detection and actuation continue with no network connection, echoing the case for local inference as a first-class edge-sensing requirement [8]. And it must

build in privacy from the start: no audio, video, or identity-revealing signal is captured, and only an aggregate occupied/vacant state, never raw sensor values, leaves the room.

### 3.2 Sensor and Edge-Compute Layers

Fig. 1 shows the architecture. The in-room sensor cluster combines four low-cost modalities, none able to capture identity or conversation content, unlike camera- or microphone-recording-based systems. A PIR module supplies a fast, low-power motion signal, the modality identified as most widely deployed [3]. An NDIR CO2 sensor adds a slower, motion-independent presence proxy, covering PIR's blind spot when an occupant sits still [3], [5]. A MEMS microphone computes only an ambient sound-pressure level in decibels once per window; no waveform

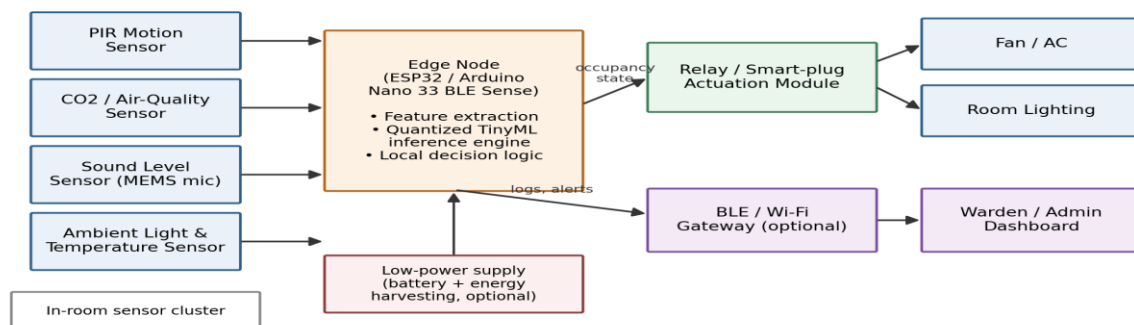
is stored or transmitted. An ambient light sensor paired with temperature/humidity adds weak but useful context, since occupied rooms show different lighting and micro-climate patterns, in line with the multi-sensor fusion shown to beat single-sensor estimation [4], [6], [11]. A microcontroller-class edge node, in the class of the ESP32-S3 or Arduino Nano 33 BLE Sense, hundreds of kilobytes of RAM, a few megabytes of flash, a few hundred to roughly a thousand rupees, with mature TinyML support [7], aggregates readings locally, matching the hardware class used in [11] and the categories surveyed in [9]. Every sampling window it extracts the feature vector (Section 4.1), runs the quantized classifier (Section 5.3), and applies the actuation logic (Section 4.3), all without a round-trip to a server.

### 3.3 Actuation, Connectivity, and Power

The occupancy decision drives a relay or smart-plug module wired inline with the room's fan/AC and lighting

circuits, following occupant-centric actuation strategies documented for HVAC and lighting [2], echoing a 15-month smart-plug study where programmable switching alone produced measurable conservation gains [18]. Because this switches live household current, installation is assumed to happen under qualified institute electrical staff, not students (Section 6.2). An optional BLE/Wi-Fi gateway periodically relays aggregate telemetry, occupancy-state transitions and estimated consumption, never raw sensor values, to a warden dashboard, following the edge-AI principle that only decisions need leave the device [8], [10]; since this layer is optional, a Wi-Fi dropout only degrades remote reporting; local control keeps running. The edge node and sensors run from a small battery, optionally topped up by energy harvested from the controlled circuit, in keeping with harvesting techniques for self-powered indoor IoT [17], keeping the sensing subsystem's own standby draw ( $\sim 1.2$  W in the Section 5 simulation) small next to the loads it manages, matching single-digit-milliwatt active power and sub-5-mW sleep currents reported for comparable microcontrollers [9], [10].

**Figure 1. Edge-based TinyML occupancy-sensing and control architecture**



**Fig -1:** Edge-based TinyML occupancy-sensing and control architecture: sensors feed a microcontroller edge node that performs on-device inference and drives local actuation, with an optional gateway for aggregate telemetry.

**Table -2:** Indicative hardware bill of materials for one room's sensor-and-actuation node

| Component                | Representative part             | Function                               | Approx. unit cost (INR) | Active power       |
|--------------------------|---------------------------------|----------------------------------------|-------------------------|--------------------|
| PIR motion sensor        | HC-SR501 class                  | Coarse motion detection                | 60-90                   | $\sim 65$ mW       |
| CO2 / air-quality sensor | NDIR module (e.g. MH-Z19 class) | Presence proxy independent of motion   | 900-1,200               | $\sim 150$ -200 mW |
| MEMS microphone          | I2S digital mic class           | Ambient sound level only, no recording | 100-150                 | $\sim 5$ mW        |
| Light +                  | LDR + digital                   | Ambient light and                      | 150-200                 | $\sim 5$ mW        |

| Component                   | Representative part                        | Function                              | Approx. unit cost (INR) | Active power                  |
|-----------------------------|--------------------------------------------|---------------------------------------|-------------------------|-------------------------------|
| temperature/humidity sensor | temp/RH module                             | micro-climate context                 |                         |                               |
| Microcontroller (edge node) | ESP32-S3 / Arduino Nano 33 BLE Sense class | Feature extraction + TinyML inference | 600-1,400               | 80-240 mW active; <5 mW sleep |
| Relay / smart-plug actuator | 5V single-channel relay module             | Switches fan/light circuit            | 80-120                  | ~80 mW (coil)                 |
| Enclosure, wiring, misc.    | -                                          | Housing and connections               | 150-250                 | -                             |
| Estimated total per room    | -                                          | -                                     | ~2,000-3,400            | -                             |

Note: Costs and power figures are indicative engineering estimates based on typical listings for comparable component classes at the time of writing, not verified retail quotations, and are meant to give an order-of-magnitude sense of feasibility rather than serve as a procurement quote.

## 4. METHODOLOGY

### 4.1 Simulated Sensor Dataset

Instrumenting live hostel rooms fell outside this study's scope, so model development ran on a synthetic dataset built to reflect published patterns, not physically measured logs. The generative procedure is implemented in a single, seeded, version-controlled module (datagen.py) shared by both the model-comparison and energy-simulation datasets. Five-minute-window feature vectors, PIR count, CO<sub>2</sub>, sound level, light level, temperature, hour of day, and a weekday/weekend flag, were generated over 21 simulated days (6,048 rows) using a NumPy Generator seeded at 42. For every window, a deterministic base occupancy probability is assigned from hour and weekday/weekend flag (0.10 on weekdays 9:00-17:00; 0.55 at 17:00-20:00; 0.95 at 22:00-7:00; 0.40 otherwise on weekdays; 0.75 on weekends outside 10:00-12:00 and 0.45 within it), perturbed by Gaussian noise (sd=0.07), clipped to [0.02, 0.98], and drawn via Bernoulli to fix ground-truth occupancy. Conditioned on that state, each sensor feature is drawn independently from the distribution in Table -3, calibrated qualitatively against the PIR/CO<sub>2</sub> literature [3], [4], [5]; the deliberate overlap between occupied/vacant distributions reflects that no single sensor is a perfect proxy. A further 3% random label flip emulates sensor glitches. The held-out week used for the energy simulation (Section 4.5) is generated by the identical procedure with a different seed (7), so no training-seed artifact leaks into the energy figures. Every parameter above is stated here and in Table -3 so the dataset is reproducible from this text alone; it remains explicitly a simulation input, not a claim about measured hostel conditions (Section 6.3).

### 4.2 Candidate Models, Training, and Evaluation Protocol

Four classifier families were tested: a shallow decision tree (depth 4) for interpretability and near-zero inference cost; multinomial logistic regression as a minimal-footprint linear baseline; k-nearest-neighbours (k=5), since instance-based methods are sometimes proposed despite their storage cost; and a single-hidden-layer MLP with eight hidden units ("Tiny MLP"), the smallest practical neural-network classifier deployable via TensorFlow Lite for Microcontrollers [7]. All four trained on identical feature vectors, isolating model family from input representation. A single 75/25 split cannot distinguish a genuinely stable model from one that drew an easy test fold, so evaluation instead uses 30 independent repeated stratified 75/25 splits (seeds 0-29, ~1,512-window test set each). Features for logistic regression, k-NN, and the Tiny MLP were standardized per split using training-fold statistics only, refit each time so no test-fold information leaks; the decision tree used unscaled features. For every split and model, accuracy, precision, recall, F1, specificity, and false-negative rate were computed, and the 30 per-split values summarized as mean, SD, and a 95% t-distribution confidence interval (Table -4, Section 5.1), with pooled confusion matrices also reported (Section 5.1). Inference latency is the average single-sample prediction time over 200 repetitions on the evaluation machine, a relative proxy across model families, not an absolute microcontroller figure (Section 5.4, VI-C). A separate five-way ablation (Section 5.2) retrains the Tiny MLP on progressively larger feature subsets, PIR alone; PIR+CO<sub>2</sub>; PIR+CO<sub>2</sub>+light; the full seven-feature set; and all five physical sensors with time features removed, each averaged over 15 splits, to check whether the hour/weekend features, also inputs to the synthetic label generator, inflate the reported accuracy.

### 4.3 On-Device Inference Pipeline and Actuation Logic

Fig. 2 shows the six-stage pipeline run every sampling window: raw sensor sampling; feature extraction; on-device normalization using fixed mean/SD constants baked into firmware; quantized model inference; conversion to a binary occupied/vacant decision; and actuation plus an optional telemetry log entry, all inside the microcontroller's own memory with no network round-trip. To keep momentary sensor noise from causing relay chatter, a real deployment should require a prediction to hold steady across two consecutive windows before switching state, a standard debounce/hysteresis rule. This hysteresis is left out of the raw per-window results in Section 5, so those numbers are a conservative, worst-case view of classification quality; its effect on the smoothed control signal is left for the field validation in Section 7.

### 4.4 Quantization and On-Device Memory Footprint Verification

An "under 300 bytes" footprint claim derived only from the Tiny MLP's raw float32 parameter count is not the same thing as a complete deployed footprint, since it never went through quantization. This section instead reports a genuine, reproducible quantization-and-measurement pipeline. A TensorFlow/Keras model matching the Tiny MLP architecture (Dense(8, ReLU) then Dense(1, sigmoid), 73 parameters) was trained on the same standardized features and split, converted with TensorFlow Lite's post-training full-integer quantization (200-sample representative-dataset calibration) to an int8 model, following the 8-bit approach shown to typically cut storage three-to-four-fold with minimal accuracy loss [10]; quantized test accuracy (93.25%) was within 0.14 points of the float32 counterpart (93.39%). Quantization is one of the standard model-compression techniques alongside pruning and knowledge distillation [20]; a 2025 study combining pruning with quantization reported roughly fifteen-fold size reductions [22], suggesting headroom if a larger hidden layer were ever needed.

Four genuinely measured quantities follow, reported together in Table -7 (Section 5.4). Raw quantized parameter storage (int8 weights plus int32 biases) is 100 bytes; the complete .tflite flatbuffer actually shipped in firmware is 2,280 bytes, with the gap almost entirely FlatBuffer schema/operator overhead (~91% of the file) rather than weights, an instructive finding in its own right for a model this small. This flatbuffer was loaded into a real TensorFlow Lite for Microcontrollers interpreter (op resolver: FULLY\_CONNECTED, FULLY\_CONNECTED, LOGISTIC) built from source, using TFLM's RecordingMicroAllocator on a host x86\_64 build: AllocateTensors() succeeded with an arena of exactly 1,520 bytes and failed below 1,504 bytes. Because TFLM's

memory planner sizes the arena from tensor shapes, dtypes, and the operator graph rather than host CPU ISA, this figure was expected to transfer to the ESP32-S3 target, and that expectation was checked directly: the identical flatbuffer, resolver, and a 1,536-byte arena were built into a minimal ESP-IDF v5.3.1 application, cross-compiled with Espressif's xtensa-esp32s3-elf toolchain, and linked into a complete firmware image. The linker-reported figures are the fourth measured quantity: the TFLM runtime plus the two reachable kernels plus the model occupy 27,753 bytes of flash and 196 bytes of static RAM; adding application glue and the declared arena brings the TinyML-attributable total to 28,417 bytes flash and 1,732 bytes static RAM, of which 1,536 bytes is the arena, confirming the host-measured 1,520-byte minimum almost exactly on the real target. This is a small slice of the complete bootable firmware image (227,964 bytes), dominated by the mandatory ESP-IDF/FreeRTOS framework any application on this target must link against, not by the model. The firmware was not flashed to physical hardware, since no board was available; what is reported is the compiled-and-linked image for the real target architecture and toolchain, not a host-machine estimate.

### 4.5 Simulated Energy-Reduction Estimate and Sensitivity-Analysis Design

Every result here and in Section 5.5 is called a simulated energy reduction, never an "energy saving," to keep the assumption trail attached to the number. An independent seven-day dataset was generated with the same procedure as Section 4.1 (seed 7), and the Tiny MLP predicted occupancy for every window. Two policies were compared over identical ground-truth occupancy: an always-on baseline, the headline condition and the one most consistent with the hostel energy-audit literature's description of common practice [12], leaving fan and lights on all day once first switched on; and an occupancy-aware policy that energizes the circuit only when the model predicts occupancy, plus a constant sensor-node standby draw. Fan and lighting power were assumed at 70 W and 18 W (a ceiling fan/AC fan-mode load and an LED circuit), standby at 1.2 W.

Because this figure is potentially the most vulnerable claim in the paper, every load-bearing assumption was swept rather than fixed. Occupancy-pattern sensitivity reran the simulation across 20 independently seeded weeks. Fan power was swept 50-100 W, lighting power 10-25 W, and tariff INR 6-10/kWh. False-negative sensitivity artificially flipped 0-30% of correctly-predicted-occupied windows to vacant, to check whether the reduction estimate and comfort cost move together. Baseline-policy sensitivity, the single most consequential assumption, replaced the always-on baseline with an "evening-only" semi-adaptive timer (17:00-23:00 and 22:00-07:00 on, off otherwise) that some hostels already use, to test how much

of the headline reduction is attributable to the model rather than to baseline wastefulness. All five sweeps are reported in Section 5.5 and Table -8.

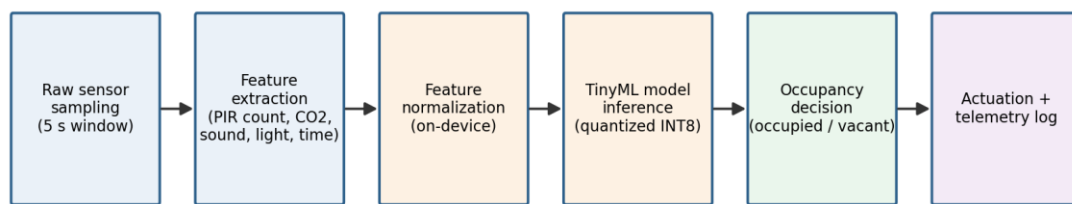
**Table -3:** Simulated per-feature statistics by ground-truth occupancy state, informed by patterns reported in [3]-[5]

| Feature                   | Sensor            | Occupied        | Vacant           |
|---------------------------|-------------------|-----------------|------------------|
| PIR events / 5 min window | PIR motion sensor | $\lambda = 1.8$ | $\lambda = 0.30$ |
| CO2 concentration (ppm)   | NDIR CO2 sensor   | 750 (90)        | 600 (70)         |

| Feature                  | Sensor             | Occupied   | Vacant     |
|--------------------------|--------------------|------------|------------|
| Ambient sound level (dB) | MEMS microphone    | 38 (6)     | 33 (6)     |
| Ambient light (lux)      | Light sensor       | 80 (40)    | 20 (15)    |
| Temperature (°C)         | Temperature sensor | 27.2 (1.0) | 26.8 (1.0) |

Note: PIR event count follows a Poisson distribution; all other features follow a Normal distribution reported as mean (standard deviation).

**Figure 2. On-device inference pipeline executed every sampling window**



**Fig -2:** Six-stage on-device inference pipeline executed by the edge node every sampling window.

## 5. RESULTS AND EVALUATION

### 5.1 Model Comparison and Error Analysis

Table -4 and Fig. 4 report accuracy, F1, specificity, false-negative rate, latency, and footprint, each a mean +/- SD over 30 repeated splits (95% CI for accuracy). All four models land within a two-point band, 92.2% to 93.7% mean accuracy, confirming the engineered feature set carries most of the discriminative signal regardless of model family [1]. The Tiny MLP reaches the highest mean accuracy (93.7% +/- 0.6%, 95% CI [93.5%, 93.9%]) and F1 (94.5% +/- 0.5%), with a confidence interval that does not

overlap the decision tree's; k-nearest-neighbours, which led on a single split in an earlier version of this evaluation, sits third, and still has to store the entire training set (est. 124.03 KB) against the Tiny MLP's real, measured 2.23 KB quantized flatbuffer. Table -5 pools confusion-matrix cells across all 30 splits (45,360 predictions/model). False-positive counts are broadly similar across models (1,296-1,504), but false negatives, the error that costs comfort by leaving an occupied room's fan or lights off, diverge more: the Tiny MLP's 1,436 pooled false negatives is the lowest of the four, below the decision tree's 2,043, logistic regression's 1,915, and k-NN's 1,835, the more decision-relevant property behind its selection (Section 5.3).

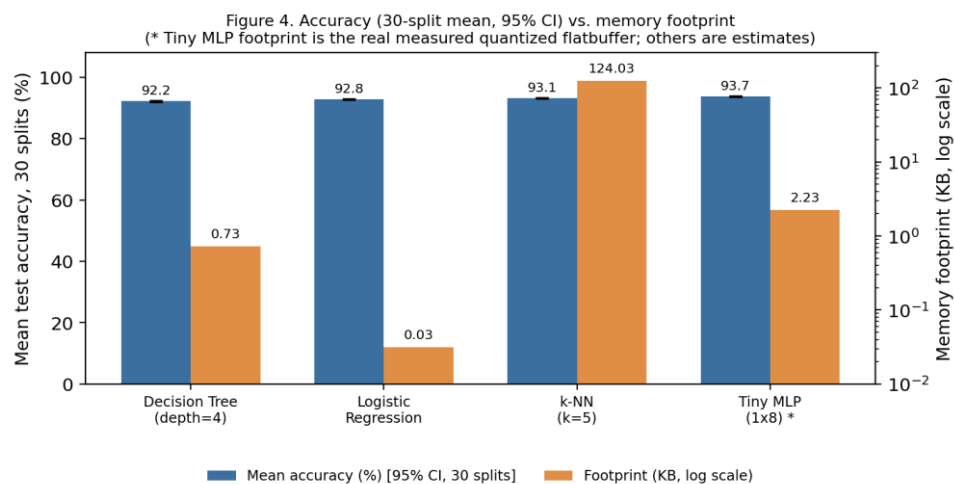
**Table -4:** Multi-seed classification performance across 30 repeated splits (mean +/- SD; 95% CI for accuracy)

| Model         | Accuracy (95% CI) | F1-score   | Specificity | FNR   | Latency (ms) | Footprint (KB) |
|---------------|-------------------|------------|-------------|-------|--------------|----------------|
| Decision Tree | 92.18% +/-        | 93.08% +/- | 92.26%      | 7.88% | 0.08         | 0.73 (est.)    |

| Model                      | Accuracy (95% CI)               | F1-score         | Specificity | FNR   | Latency (ms) | Footprint (KB)  |
|----------------------------|---------------------------------|------------------|-------------|-------|--------------|-----------------|
| (depth=4)                  | 0.75% [91.90, 92.46]            | 0.68%            |             |       |              |                 |
| Logistic Regression        | 92.82% +/- 0.62% [92.59, 93.05] | 93.64% +/- 0.56% | 93.09%      | 7.39% | 0.08         | 0.03 (est.)     |
| k-Nearest Neighbours (k=5) | 93.10% +/- 0.53% [92.90, 93.29] | 93.90% +/- 0.47% | 93.33%      | 7.08% | 0.97         | 124.03 (est.)   |
| Tiny MLP (1x8) — selected  | 93.73% +/- 0.57% [93.52, 93.94] | 94.51% +/- 0.50% | 92.76%      | 5.54% | 0.10         | 2.23 (measured) |

**Table -5:** Pooled confusion-matrix cells across 30 splits (45,360 predictions per model)

| Model                      | True Negatives | False Positives | False Negatives | True Positives |
|----------------------------|----------------|-----------------|-----------------|----------------|
| Decision Tree (depth=4)    | 17,936         | 1,504           | 2,043           | 23,877         |
| Logistic Regression        | 18,096         | 1,344           | 1,915           | 24,005         |
| k-Nearest Neighbours (k=5) | 18,144         | 1,296           | 1,835           | 24,085         |
| Tiny MLP (1x8) — selected  | 18,033         | 1,407           | 1,436           | 24,484         |



**Fig -4:** Mean accuracy (with 95% CI error bars, 30 repeated splits) vs. memory footprint across the four candidate models (footprint axis on log scale; Tiny MLP footprint is the real measured quantized flatbuffer, the other three are estimates).

## 5.2 Sensor Ablation Study

Table -6 reports the five-way ablation (Section 4.2), each averaged over 15 splits. PIR alone reaches only 78.4% accuracy, confirming that motion counts alone miss

stationary occupants [3]; adding CO<sub>2</sub> lifts accuracy to 86.9%, and adding light closes most of the remaining gap, to 93.2%. The key comparison is between the full seven-feature set (93.72% +/- 0.44%) and the same five physical sensors with time-of-day features removed (92.94% +/-

0.52%): a gap of only 0.78 points. Since hour-of-day and weekend flag are also inputs to the synthetic label generator, a much larger gap would have suggested the model was exploiting that shared schedule rather than reading the sensors; instead the physical sensors alone recover essentially all the accuracy, with time features contributing a small, bounded increment.

### 5.3 Selected Deployment Model

A microcontroller in the ESP32/Nano 33 BLE Sense class typically offers on the order of a few hundred kilobytes of usable RAM shared with firmware and drivers,

so k-NN's 124.03 KB eats a substantial fraction of that budget and only worsens as the training set grows. The decision tree and logistic regression have smaller estimated footprints (0.73 KB, 0.03 KB) but trail the Tiny MLP in accuracy and, more importantly, false-negative rate. Weighing Tables IV and V against embedded feasibility, this paper selects the Tiny MLP: it reaches the highest mean accuracy and F1, the lowest pooled false-negative count, and, once quantized, a real footprint orders of magnitude smaller than its closest accuracy competitor, illustrating a broader TinyML principle: the most accurate model in an offline comparison is not automatically the one worth deploying [9], [10].

**Table -6:** Five-way sensor ablation, Tiny MLP, averaged over 15 repeated splits

| Feature configuration                 | # Features | Accuracy (mean +/- SD) | F1-score (mean +/- SD) | FNR (mean) |
|---------------------------------------|------------|------------------------|------------------------|------------|
| PIR only                              | 1          | 78.41% +/- 0.73%       | 81.56% +/- 0.59%       | 16.43%     |
| PIR + CO2                             | 2          | 86.92% +/- 0.74%       | 88.53% +/- 0.67%       | 11.64%     |
| PIR + CO2 + light                     | 3          | 93.15% +/- 0.49%       | 93.96% +/- 0.43%       | 6.81%      |
| All 5 sensors, no time features       | 5          | 92.94% +/- 0.52%       | 93.81% +/- 0.46%       | 6.37%      |
| All sensors + time (full feature set) | 7          | 93.72% +/- 0.44%       | 94.50% +/- 0.39%       | 5.51%      |

Note: Hour-of-day and weekend flag are also inputs to the synthetic dataset's own occupancy-probability schedule (Section 4.1); the gap between the last two rows is the check for schedule leakage through those two features.

### 5.4 Memory Footprint, Host-Measured and On-Target

Table -7 collects every memory figure from the pipeline in Section 4.4, each labeled by how it was obtained rather than presented as one undifferentiated number. Raw int8 storage (100 bytes) and the complete .tflite flatbuffer (2,280 bytes) come from TensorFlow Lite's converter and analyzer; the 1,520-byte host-build minimum arena comes from a genuine TFLM RecordingMicroAllocator run; the remaining rows come from an actual ESP-IDF v5.3.1 build cross-compiled for the esp32s3 target and linked into a complete firmware image, not a host-machine estimate. Together the TinyML component occupies 28,417 bytes of flash and 1,732 bytes of static RAM on the real target, of which 1,536 bytes is the declared tensor arena, lining up almost exactly with the 1,520-byte host-measured minimum, a direct check on that expectation rather than an untested assumption. Against the complete bootable firmware image (227,964 bytes), the TinyML component is a small slice, around 17% of flash and 2.6% of used static RAM, the remainder being mandatory ESP-IDF/FreeRTOS framework overhead any application on this target must carry. This firmware was compiled and linked for the real target but not flashed to physical hardware, since no board was available; latency and active/sleep power therefore

remain projections from comparable published on-device measurements [9], [10], [11], not measurements from this study's own hardware (Section 6.3).

### 5.5 Simulated Energy-Reduction Results and Sensitivity Analysis

Averaged over 20 independently seeded occupancy-pattern weeks (Section 4.5), the headline simulated energy reduction is 40.5% +/- 0.8% (range 38.6-42.4%), deliberately termed a simulated energy reduction rather than an energy saving throughout, and one that sits at the upper end of, but within, the roughly 10-40% range commonly reported for occupancy-based building control [2]. Fig. 3 shows one representative simulated week (seed 0): the baseline consumes an estimated 14.78 kWh against 8.78 kWh occupancy-aware, a 40.6% reduction, with daily reductions from about 24% to 50% and the smallest reductions on the two weekend days, when the simulated schedule keeps the room in use for more of the day. Table -8 reports the five sensitivity sweeps. The reduction is essentially insensitive to fan power (39.8-40.5% across 50-100 W) and lighting power (40.1-40.3% across 10-25 W), and tariff leaves the percentage unchanged at 40.2% while scaling only the INR figure (INR 35.65-59.42/week across INR 6-10/kWh). Two sweeps matter more. Inflating the

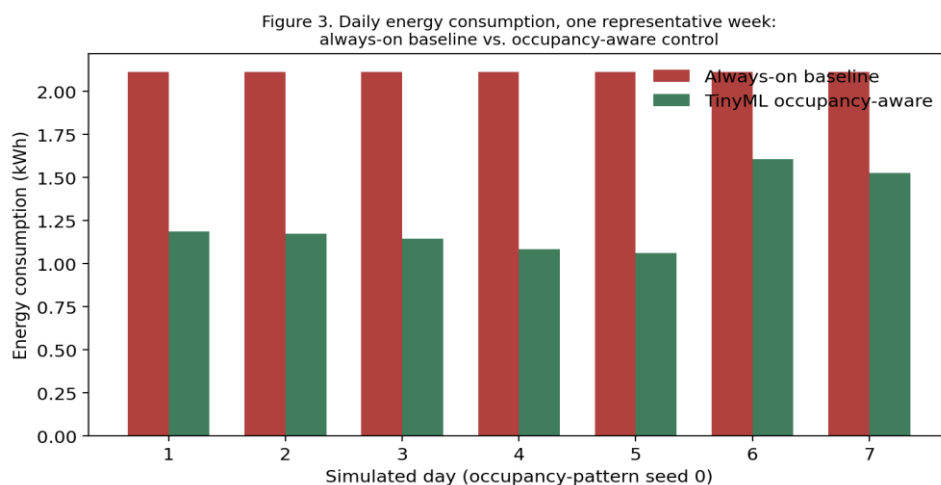
model's false-negative rate by 0-30% paradoxically raises the naive reduction estimate from 40.2% to 56.8%, since a model that misses more occupancy simply leaves devices off more, while missed-occupancy itself rises from 3.1% to 19.6%, a comfort cost, not a genuine efficiency gain, if the percentage is read in isolation. Most consequentially, replacing the always-on baseline with the semi-adaptive

"evening-only" schedule collapses the reduction from 40.5% to -1.9%, essentially zero. The headline figure is therefore not a property of the model in isolation; it is a property of the model measured against a specific, genuinely wasteful baseline, and should be read as an upper bound that depends on how adaptive a given hostel's existing habits already are.

**Table -7:** Quantized-model memory footprint: host-measured and real ESP32-S3 target build

| Quantity                                                              | Value                      | Measurement basis                               |
|-----------------------------------------------------------------------|----------------------------|-------------------------------------------------|
| Raw int8 weights + int32 biases (persistent)                          | 100 bytes                  | Measured (TFLite converter)                     |
| Complete .tflite flatbuffer (deployed model file)                     | 2,280 bytes                | Measured (TFLite converter)                     |
| TFLM tensor arena, host-build minimum                                 | 1,520 bytes                | Measured (RecordingMicroAllocator, host x86_64) |
| ESP32-S3 firmware: TFLM runtime + reachable kernels (flash)           | 27,753 bytes               | Measured (esp32s3 target build)                 |
| ESP32-S3 firmware: TFLM static globals + declared arena (RAM)         | 1,732 bytes                | Measured (esp32s3 target build)                 |
| ESP32-S3 firmware: application glue code (flash)                      | 664 bytes                  | Measured (esp32s3 target build)                 |
| TinyML-attributable total (flash / static RAM)                        | 28,417 bytes / 1,732 bytes | Measured (esp32s3 target build)                 |
| Whole bootable firmware image (reference; ESP-IDF framework overhead) | 227,964 bytes              | Measured (esp32s3 target build)                 |

Note: Rows below the arena line come from an actual ESP-IDF v5.3.1 build cross-compiled for the esp32s3 target (idf.py size-components), not a host-machine estimate; the firmware was not flashed to physical hardware, since no board was available.



**Fig -3:** Simulated daily energy consumption over one representative week: always-on baseline vs. TinyML occupancy-aware control.

**Table -8:** Energy-reduction sensitivity analysis across five swept assumptions

| Sweep                     | Range tested               | Resulting simulated reduction     | Note                                                                                                      |
|---------------------------|----------------------------|-----------------------------------|-----------------------------------------------------------------------------------------------------------|
| Occupancy pattern         | 20 seeded weeks            | 38.6%-42.4% (mean 40.5% +/- 0.8%) | Headline figure; modest spread across plausible weeks                                                     |
| Fan power                 | 50-100 W                   | 39.8%-40.5%                       | Essentially insensitive to fan wattage                                                                    |
| Lighting power            | 10-25 W                    | 40.1%-40.3%                       | Essentially insensitive to lighting wattage                                                               |
| Electricity tariff        | INR 6-10/kWh               | 40.2% (unchanged)                 | Tariff scales the INR saving, not the % reduction                                                         |
| Added false-negative rate | +0-30%                     | 40.2% -> 56.8%                    | Reduction % rises as missed occupancy rises, 3.1% -> 19.6%: a comfort cost, not a genuine efficiency gain |
| Baseline policy           | always-on vs. evening-only | 40.5% vs. -1.9%                   | The single most consequential assumption: a realistic semi-adaptive baseline erases the reduction         |

## 5.6 Comfort Trade-off and Illustrative Cost-Benefit Estimate

Averaged over the same 20 simulated weeks, the deployed model missed true occupancy in 3.1% +/- 0.3% of windows and switched on for a vacant room in 3.3% +/- 0.3%, both from raw, non-hysteresis output; the two-window debounce rule (Section 4.3) should cut spurious activations further, at a small latency cost, though quantifying that precisely needs the field data in Section 7. As an order-of-magnitude illustration, not a financial projection: an assumed tariff of INR 8/kWh applied to the headline weekly reduction of roughly 6.0 kWh/room works out to about INR 48/week, or roughly INR 2,150-2,160 over a 45-week occupied year, against the INR 2,000-3,400 hardware cost in Table -2, a payback of roughly eleven to nineteen months, and only under a genuinely wasteful always-on baseline (Section 5.5); a hostel already on a semi-adaptive schedule should expect a far longer payback, or none. This leans heavily on the assumed baseline, tariff, and device power, none checked against a specific hostel's metered billing data.

## 6. DISCUSSION

### 6.1 Energy-Accuracy-Footprint Trade-offs

Section 5.1 illustrate a recurring TinyML lesson: once several model families reach broadly similar accuracy on a well-engineered feature set, the deployment decision should hinge on memory and latency budget and on which error type actually costs comfort, not on chasing the last

fraction of a point of raw accuracy [1], [9]. Evaluating across 30 splits rather than a single split changed this trade-off rather than merely confirming it: the Tiny MLP does not trade accuracy away for a smaller footprint, it wins on mean accuracy, F1, and pooled false-negative count against all three other candidates, while its real, quantized footprint (2,280 bytes) is still roughly 56 times smaller than k-NN's estimated 124.03 KB. A sharp trade-off does remain against k-NN specifically on inference cost, 0.97 ms versus the Tiny MLP's 0.10 ms, a gap that would only widen on a constrained microcontroller as the stored training set grows. The model worth deploying is the one that wins jointly on accuracy, the costliest error type, latency, and real measured footprint, not the one that tops a single accuracy column [9], [10].

### 6.2 Privacy, Ethics, and Deployment Challenges

A hostel room is a private residence, so sensing was deliberately limited to signals that cannot reconstruct identity, conversation content, or activity detail: PIR counts, CO2, an aggregate sound level instead of an audio waveform, and ambient light/temperature. Processing happens entirely on-device, following the principle that inference should occur where data is sensed [7], [8], and the optional gateway relays only aggregate state transitions and energy figures, never raw traces detailed enough to reconstruct a resident's schedule. Smart-home surveys point to inadequate disclosure and weak dashboard access control as recurring sources of resident distrust [19], so any real deployment needs clear disclosure of what is sensed, how long it is kept, who can

access it, and an opt-out or manual override. Several practical challenges also remain: NDIR CO<sub>2</sub> sensors drift and need periodic calibration [5], adding fleet-wide maintenance burden; the architecture assumes a single-occupant room, and shared rooms would need per-bed sensing or occupant-counting instead of binary classification [6] (Section 7); relay installation on a live circuit is an electrical-safety matter for qualified electricians, not students, adding cost on top of Table -2; and keeping batteries maintained across hundreds of nodes is a real operations problem that favours energy harvesting where feasible [9].

### 6.3 Limitations

The central limitation is that both datasets are synthetic, generated to qualitatively match patterns in the cited literature, not measured from instrumented hostel rooms as in the hardware-validated deployment in [11]; the reported figures are a simulation-based feasibility estimate, not a physical field trial, and Section 5.5's sensitivity analysis showed directly how much the headline energy figure depends on the baseline assumption. The energy model assumes constant power draw whenever a device is on, with no AC compressor cycling, thermal inertia, or seasonal cooling shifts, and compares only two hand-chosen baseline policies rather than surveying real switching habits. On hardware, Section 5.4 narrows but does not remove this limitation: flash and static RAM are now genuine figures from a real ESP-IDF build cross-compiled for the ESP32-S3 target, but latency and active/sleep power remain projections from comparable published measurements [9], [10], [11], since the firmware was never flashed to physical hardware. Every one of these simplifications, the synthetic data most of all, is a target for the field validation in Section 7.

### 7. FUTURE WORK

Building on Section 6.3, four directions are recommended. Most importantly, a pilot field deployment across instrumented hostel rooms should collect real sensor logs, check the simulated accuracy and energy-reduction estimates against metered ground truth, and measure actual on-device latency and power on physical microcontroller hardware instead of the projected figures used here (Section 5.4), following the methodology already demonstrated for a single-room ESP32 occupancy node in [11]. Second, the binary occupied/vacant classification should extend to multi-occupant counting for shared rooms, likely needing higher-resolution sensing and a regression or ordinal formulation, in the spirit of the PIR-and-smart-meter approach in [6], alongside an on-device continual- or federated-learning scheme that lets each room's node adapt to its resident's habits without centralizing raw data, drawing on privacy-preserving aggregation for federated IoT [13]. Third, the on/off

actuation logic could give way to a comfort-aware controller that modulates fan speed or setpoint in proportion to a confidence score, improving the comfort-energy trade-off in Section 5.6. Fourth, low-resolution mmWave radar, recently shown capable of robust people counting without identifiable imagery [14], is worth investigating as a further privacy-preserving sensing modality, alongside a formal techno-economic study using real metered billing data to replace the illustrative estimate in Section 5.6 with a validated payback analysis across a full hostel block.

### 8. CONCLUSION

This paper set out a resource-efficient edge TinyML framework for real-time occupancy estimation and energy-aware control in student hostel rooms, motivated by the well-documented tendency of hostel occupants to leave fans, air-conditioners, and lights running while they are away [12]. It proposed a privacy-preserving, multi-sensor architecture combining PIR motion, CO<sub>2</sub>, ambient sound level, and ambient light, and compared four microcontroller-appropriate classifiers over 30 repeated stratified splits, reporting accuracy, F1, specificity, and false-negative rate as mean  $\pm$  SD with 95% confidence intervals, pooled confusion matrices, and a five-way sensor-ablation study that ruled out schedule leakage through time-of-day features. The Tiny MLP came out as the preferred deployment model on the joint basis of accuracy, false-negative rate, and footprint, reaching 93.7%  $\pm$  0.6% mean test accuracy; converted to a real 8-bit-quantized TensorFlow Lite Micro model and cross-compiled into an actual ESP32-S3 firmware build, it occupies a genuinely measured 28,417 bytes of flash and 1,732 bytes of static RAM, a small fraction of a complete bootable firmware image, correcting an earlier, unquantized "under 300 bytes" estimate. A week-long simulation, grounded in the published literature, suggested a simulated energy reduction of 40.5%  $\pm$  0.8% against an always-on baseline, while a sensitivity analysis across occupancy patterns, device power, tariff, false-negative rate, and baseline policy showed this figure depends heavily on the last assumption: against a more realistic, already-semi-adaptive baseline, the simulated reduction essentially disappears. The deployed model missed genuine occupancy in 3.1%  $\pm$  0.3% of intervals. These results are simulation-based, and flash/RAM figures aside, not yet field-validated on physical hardware, as this paper has said throughout rather than only in a limitations paragraph; the natural next step, outlined in Section 7, is a pilot deployment measuring these same quantities directly on instrumented hostel rooms and physical microcontroller hardware.

## REFERENCES

- 1) W. Zhang, Y. Wu, and J. K. Calautit, "A review on occupancy prediction through machine learning for enhancing energy efficiency, air quality and thermal comfort in the built environment," *Renewable and Sustainable Energy Reviews*, vol. 167, p. 112704, 2022.
- 2) P. Anand, D. Cheong, and C. Sekhar, "A review of occupancy-based building energy and IEQ controls and its future post-COVID," *Science of The Total Environment*, vol. 804, p. 150249, 2022.
- 3) A. Shokrollahi, J. A. Persson, R. Malekian, A. Sarkheyli-Hägele, and F. Karlsson, "Passive infrared sensor-based occupancy monitoring in smart buildings: A review of methodologies and machine learning approaches," *Sensors*, vol. 24, no. 5, art. 1533, 2024.
- 4) C. Jiang, Z. Chen, L. C. Png, K. Bekiroglu, S. Srinivasan, and R. Su, "Building occupancy detection from carbon-dioxide and motion sensors," in *Proc. 15th Int. Conf. Control, Automation, Robotics and Vision (ICARCV)*, 2018.
- 5) J. Kim, J. Bang, A. Choi, H. J. Moon, and M. Sung, "Estimation of occupancy using IoT sensors and a carbon dioxide-based machine learning model with ventilation system and differential pressure data," *Sensors*, vol. 23, no. 2, art. 585, 2023.
- 6) Y. Wu, S. Chen, Y. Jin, H. Xu, X. Zhou, X. Wang, A. Chong, J. Li, and D. Yan, "Novel occupancy detection method based on convolutional neural network model using PIR sensor and smart meter data," *Advanced Engineering Informatics*, vol. 62, p. 102589, 2024.
- 7) R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow Lite Micro: Embedded machine learning on TinyML systems," in *Proc. Machine Learning and Systems (MLSys)*, vol. 3, 2021. arXiv:2010.08678.
- 8) S. Soro, "TinyML for Ubiquitous Edge AI," MITRE Technical Report, Sep. 2020. arXiv:2102.01255.
- 9) S. Heydari and Q. H. Mahmoud, "Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions," *Sensors*, vol. 25, no. 10, art. 3191, 2025.
- 10) C. M. Bhushan, P. Koppuravuri, N. Prasanthi, F. Gazi, M. M. Hussain, M. Abdussami, A. A. Devi, and J. Faizi, "Deploying TinyML for energy-efficient object detection and communication in low-power edge AI systems," *Scientific Reports*, vol. 15, art. 44299, 2025.
- 11) M. Pandkar, S. Nambiar, A. Sinha, and P. Sharma, "Optimizing room occupancy estimation on the edge: A TinyML and sensor network approach," *Results in Engineering*, vol. 29, p. 109107, 2026.
- 12) B. S. Madhusudan, S. Vandavasi, B. S. Nataraja, and G. Gopi, "Auditing and analysis of energy consumption of a hostel building," *Current Journal of Applied Science and Technology*, vol. 39, no. 25, pp. 8–18, 2020.
- 13) E. Dritsas and M. Trigka, "Federated learning for IoT: A survey of techniques, challenges, and applications," *Journal of Sensor and Actuator Networks*, vol. 14, no. 1, art. 9, 2025.
- 14) N. Zhang, H. Li, A. Zahid, Y. Tian, and W. Li, "A robust mmWave radar framework for accurate people counting and motion classification," *Sensors*, vol. 26, no. 4, art. 1289, 2026.
- 15) M. Salman, Y.-D. Seo, and Y. Noh, "WiFi-enabled occupancy monitoring in smart buildings with a self-adaptive mechanism," in *Proc. 38th ACM/SIGAPP Symp. Applied Computing (SAC)*, Tallinn, Estonia, 2023, pp. 759–762.
- 16) E. Longo, A. E. C. Redondi, and M. Cesana, "Accurate occupancy estimation with WiFi and Bluetooth/BLE packet capture," *Computer Networks*, vol. 163, p. 106876, 2019.
- 17) H. Xiao, N. Qi, Y. Yin, S. Yu, X. Sun, G. Xuan, J. Liu, S. Xiao, Y. Li, and Y. Li, "Investigation of self-powered IoT sensor nodes for harvesting hybrid indoor ambient light and heat energy," *Sensors*, vol. 23, no. 8, art. 3796, 2023.
- 18) J. Oh, "IoT-based smart plug for residential energy conservation: An empirical study based on 15 months' monitoring," *Energies*, vol. 13, no. 15, art. 4035, 2020.
- 19) M. K. Kuyucu, Ş. Bahtiyar, and G. İnce, "Security and privacy in the smart home: A survey of issues and mitigation strategies," in *Proc. 4th Int. Conf. Computer Science and Engineering (UBMK)*, 2019.
- 20) Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, art. 60, 2023.
- 21) R. Cordova-Cardenas, D. Amor, and Á. Gutiérrez, "Edge AI in practice: A survey and deployment framework for neural networks on embedded systems," *Electronics*, vol. 14, no. 24, art. 4877, 2025.
- 22) S. Makenali, B. Rokh, and A. Azarpeyvand, "Integrating pruning with quantization for efficient deep neural networks compression," arXiv:2509.04244, 2025.
- 23) X. Lu, Z. Pang, Y. Fu, and Z. O'Neill, "Advances in research and applications of CO2-based demand-controlled ventilation in commercial buildings: A critical review of control strategies and performance evaluation," *Building and Environment*, vol. 223, p. 109455, 2022.

- 24) Z. Wang, Y. Wang, R. Zeng, R. S. Srinivasan, and S. Ahrentzen, "Random forest based hourly building energy prediction," *Energy and Buildings*, vol. 171, pp. 11–25, 2018.
- 25) L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven Internet of Things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, art. 174, 2023.
- 26) N. V. S. R. Nalakurthi, I. Abimbola, T. Ahmed, I. Anton, K. Riaz, Q. Ibrahim, A. Banerjee, A. Tiwari, and S. Gharbia, "Challenges and opportunities in calibrating low-cost environmental sensors," *Sensors*, vol. 24, no. 11, art. 3650, 2024.
- 27) N. I. Jasim, S. S. Gunasekaran, N. AlDahoul, A. N. Ahmed, A. El-Shafie, M. Sherif, and M. A. Mahmoud, "Toward sustainable campus energy management: A comprehensive review of energy management, predictive algorithms, and recommendations," *Energy Nexus*, vol. 18, art. 100435, 2025.