Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Energy Allocation Between Computation and Wi-Fi Transmission in TinyML Federated Learning on IoT Devices: A Measured Crossover

MD ABDUS SHAKUR SIAM

Bangladesh University of Engineering and Technology https://orcid.org/0000-0003-4628-1832

Farag Azzedin

fazzedin@kfupm.edu.sa

King Fahd University of Petroleum and Minerals https://orcid.org/0000-0001-9712-439X

Research Article

Keywords: Computation and communication trade-off, duty cycling, energy efficiency, federated learning, Internet of Things, power measurement, TinyML, Wi-Fi

Posted Date: November 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11579488/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Energy Allocation Between Computation and Wi-Fi Transmission in TinyML Federated Learning on IoT Devices: A Measured Crossover

Md Abdus Shakur Siam and Farag Azzedin

***Abstract*—Energy models for federated learning (FL) on battery-powered Internet of Things (IoT) devices often assume that radio transmission costs far more than on-device computation. We test this assumption on a Wi-Fi microcontroller: an ESP32-S3 trains the output head of a keyword-spotting model and exchanges its updates over 2.4 GHz Wi-Fi, while an independent power monitor and hardware phase markers separate the energy of computation from that of communication. Across 105 runs, computation costs 1.00 mJ per local epoch with no detectable fixed overhead, and communication costs 7.4 to 9.7 mJ per client turn in two sessions, regardless of the number of epochs. The two are therefore equal at 7.4 to 9.7 local epochs. Training started immediately after a radio burst absorbs about 0.5 mJ of tail energy, which a short pause removes. Shannon-capacity and switching-energy models underestimate the measured energies by 95% to 99% at their nominal parameters, because they omit protocol overhead and memory activity, and neither contains the power drawn while the device is awake. We show that sleeping between rounds moves the crossover toward the ratio of exchange time to epoch time, which yields 11 to 19 epochs with sleep powers measured on the same board. In deep sleep the board draws 17.2 mW, but each wake-up, dominated by Wi-Fi re-association, costs 0.68 J, 35 times the marginal energy of a ten-epoch turn. From these results, we derive a measurement-calibrated procedure for selecting the number of local epochs and design guidelines for energy-constrained IoT clients.**

***Index Terms*—Computation and communication trade-off, duty cycling, energy efficiency, federated learning, Internet of Things, power measurement, TinyML, Wi-Fi.**

I. Introduction

FEDERATED learning (FL) allows wireless devices, from smartphones to battery-powered Internet of Things (IoT) nodes, to train a shared model without transmitting their raw data [1], [2]. Because the uplink is the scarce resource on smartphones, the prevailing recommendation is to perform more local epochs E between aggregation rounds, which reduces both the number of rounds and the traffic. The underlying premise predates FL: for wireless sensor nodes, transmitting one kilobit over 100 m was estimated to cost as much energy as executing three million instructions [3]. Much of the wireless FL literature builds on this premise. Some works select the number of local updates under a resource budget [4], [5]; others jointly allocate bandwidth, transmit power and local computation under analytical models [6]–[16]; and others compress, prune or quantize the updates [17]–[23] or study energy consumption in IoT settings [24], [25]. These models typically describe transmission through the Shannon capacity and computation through a switching energy proportional to $\kappa f^2 C$ [6]–[8], [16]. In one simulated setting, Tran et al. find that computation energy is much smaller than communication energy and conclude that devices should compute more and communicate less [6].

This premise is less secure on the microcontrollers (MCUs) that power most IoT nodes, such as environmental sensors, wearables and smart-home devices. Such devices have limited processing ability, memory, bandwidth and energy, which many FL methods do not account for [26]. An MCU trains on the same battery that powers its radio, it has no accelerator, and its SRAM is too small to train a full network [27]. It also draws substantial power merely by remaining awake, a term that the common models omit. The empirical evidence is mixed and uses different metrics: a study of wearable devices finds that computation consumes far more energy than the transmission of model updates [28], whereas MCU deployments over low-rate LoRa links find that exchanging the model can take hours per round [29]–[31], and a simulation study shows that duty-cycle limits and frame losses lengthen the rounds further [32]. Because these studies differ in radio, model and metric, they cannot locate the boundary between the computation and communication energy on a single platform.

This paper adopts an IoT system design perspective. A battery-powered IoT node with a fixed energy budget can spend it on computation (more local epochs) or on transmission (more rounds). We address three questions. At what E does the computation energy of one client turn equal its communication energy? How does this point move when the device sleeps between rounds or when the channel is congested? What does it imply for the number of rounds and the traffic of an FL job? We answer the first question by measurement on a single ESP32-S3 device that plays the role of every client. This choice is deliberate: it isolates the physical energy of one client turn from board-to-board variation and allows the energy terms to be measured precisely. The answers to the second and third questions combine these measurements, including measured sleep powers and wake-up energies, with an explicitly stated model.

To our knowledge, this is the first study to locate the crossover between the computation and communication energy

This research received no external funding.

Md Abdus Shakur Siam is with the Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh (e-mail: shakursiam@gmail.com).

Farag Azzedin is with the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia (corresponding author; e-mail: fazzedin@kfupm.edu.sa).

of an FL client on a Wi-Fi microcontroller with an independent power instrument, and to quantify how the awake baseline, the radio tail and sleep shift it. Its contributions are as follows.

1) We measure the computation and communication energy of a client turn on an ESP32-S3 over Wi-Fi with an independent power monitor, a fixed protocol and a phase-resolved attribution procedure (Algorithm 1), in a randomized campaign with six values of E and ten repetitions each. Computation grows linearly with E and communication is approximately constant, so the crossover lies near 9.7 epochs in the main campaign and near 7.4 in a second session, seven to nine times the 1.07 epochs that the combined Shannon and switching-energy model predicts at its nominal parameters.
2) We isolate the post-communication radio tail in a 45-run test, quantify the portion that a training burst absorbs, and show that a short pause removes it.
3) We derive a closed-form crossover that includes sleep power and a contention factor, and show that sleep moves the crossover monotonically toward the ratio of exchange time to epoch time (Proposition 1). We measure the board-level power of modem, light and deep sleep and the energy of waking up and re-associating with the access point over 20 cycles, and evaluate the model with these values.
4) We compare the measurements with two analytical models common in wireless FL, their combination, and a datasheet-current model, and identify the terms that each omits.
5) We decompose a client turn into protocol phases, relate convergence to rounds and transmitted bytes, and provide a measurement-calibrated procedure for selecting E (Algorithm 2). We distinguish the hardware crossover $E^{\star}$ from the learning optimum E_{opt} and state the status of every reported quantity.

Section II reviews related work, Section III presents the system model, and Section IV introduces the analytical baselines. Sections V to VIII describe the testbed, the measurements, the sleep and contention analysis, and the convergence study. Sections IX and X give design guidelines and outline the scope and future directions, and Section XI concludes the paper. Appendix A details the instrumentation, and Appendix B lists the status of each result.

II. Related Work

A. Analytical Energy Models for Wireless FL

Most energy-aware FL designs optimize an analytical model. Tran et al. trade learning time against device energy through a Pareto formulation [6]. Yang et al. minimize the total energy of FL under a latency constraint by selecting CPU frequencies and transmit powers [7], and Chen et al. select users, resource blocks and transmit powers to reduce the FL loss under delay and energy constraints [8]. Subsequent work extends these models to IoT, cell-free, UAV-assisted and mobile edge networks [10]–[12], [14], [15]. The same structure persists in recent designs: a personalized FL scheme for transformers computes client energy from a computation term proportional to the capacitance coefficient and the square of the CPU frequency, and from an upload term whose rate follows the Shannon capacity of an OFDMA subchannel [16]. These models describe computation as $\kappa f^2 C$ and communication through the Shannon capacity. In the works cited above, we found no term for the power that a device draws while awake and no comparison of the modeled energy with energy measured on a device. Wang et al. and Luo et al. treat the number of local updates as a control variable under a resource budget [4], [5]. Both built prototypes, with three Raspberry Pi devices and two laptops over Wi-Fi [4] and with twenty Raspberry Pi devices [5], respectively. Luo et al. state that they did not capture energy on their prototype because it is difficult to measure, and instead draw it from a normal distribution [5].

B. Training and FL on Microcontrollers

Memory is the principal constraint on MCU training. Lin et al. train convolutional networks within 256 KB of SRAM by skipping the gradients of less important layers [27]. A simpler approach freezes the feature extractor and trains only the last layer: TinyOL applies it online on an Arduino Nano 33 BLE Sense [33], and TinyFedTL applies it to federated transfer learning [34]. Llisterri Giménez et al. perform federated keyword spotting on three Arduino Nano 33 BLE Sense boards [35]. Both federated systems exchange weights with the server over a serial port, and TinyFedTL reports more than 30 s per transfer [34]. Neither system reports energy, and Llisterri Giménez et al. identify the energy and bandwidth trade-offs of the network interfaces of embedded devices as future work [35]. Ficco et al. combine on-board transfer learning with FL over MQTT on ESP8266, ESP32 and Arduino nodes and report the training time and the current drawn during training, measured with a USB digital multimeter [36]; they do not measure the energy of communication. Other studies run FL on MCUs over LoRa [29], [30] or simulate it [32]. Torres Sanchez et al. train an autoencoder on ESP32-S3 nodes for industrial anomaly detection, report between 0.57 and 5.96 J per training epoch depending on the machine, and estimate that exchanging the model over LoRaWAN at spreading factor 9 takes 18 to 20 h [31]. Federated TinyML has also been prototyped on MCU-based medical devices [37]. Our model split follows the last-layer approach, and its backbone is the depthwise separable CNN for keyword spotting on MCUs [38], which also serves as the keyword-spotting model of the MLPerf Tiny benchmark [39].

C. Measured Energy of Radios and FL

Measurements frequently reveal costs that models omit. Balasubramanian et al. measured 3G, GSM and Wi-Fi on mobile phones and found that 3G and GSM remain in high-power states after a transfer; in 3G, nearly 60% of the energy of a typical transfer is such tail energy [40]. For FL, Qiu et al. measured the power of NVIDIA Jetson clients with the built-in `tegrastats` utility but estimated communication energy from country-level download and upload speeds and the power of the router [41]. On MCUs, MLPerf Tiny measures inference energy with an external energy monitor [39], but it

TABLE I
KEY NOTATION.

Symbol	Meaning
E	Number of local epochs per client turn
K, D_k, B	Number of clients, local samples of client k, batch size
$\Delta E_{\mathrm{comp}}(E)$	Computation energy above baseline in one turn
ΔE_{comm}	Communication energy above baseline in one turn (download and upload)
α, c_0	Slope (mJ per epoch) and intercept of $\Delta E_{\mathrm{comp}}(E)$
P_{base}	Awake baseline power of the board
P_s, E_{wake}	Sleep power and wake-up energy
t_{ep}	Duration of one epoch
T_{comm}	Duration of the download and upload exchanges in one turn
$T_{\mathrm{active}}(E)$	Awake time of one turn, $T_{\mathrm{comm}} + t_{\mathrm{ep}}E$
T_r	Round period
$E_{\mathrm{turn}}(E, T_r)$	Gross energy of one client turn
$E^\star$, $E_0^\star$	Crossover; its value in standby
ρ	Time ratio $T_{\mathrm{comm}}/t_{\mathrm{ep}}$
$R(E)$, E_{opt}	Rounds to the target accuracy; energy-optimal E
κ_c	Airtime factor for channel contention
κ, f, C	Switching coefficient, CPU frequency and cycles (baseline model)
P_{tx}, M, W, SNR	Transmit power, payload, bandwidth and SNR (baseline model)

does not benchmark training or communication. Barral Vales et al. measured an ESP32-S2 development board during Wi-Fi ranging and found that it drew about 73 mA at 5 V between transmissions, with peaks of 454 mA, and that the module accounted for only about 25 µA of the 0.56 mA that the board, with its LED removed, drew in deep sleep [42]. The present work measures computation and Wi-Fi communication on one MCU with one independent instrument, accounts for the awake baseline and the radio tail, and compares the result with the analytical models above.

III. System Model

A. Protocol and Energy Accounting

A client holds D_k local samples. In each turn, it fetches a configuration, downloads the global model head (3,120 B), runs E epochs of mini-batch SGD with batch size $B = 16$, and uploads the updated head (3,120 B). Each of the three exchanges is a separate HTTP/1.1 request to the host that opens and closes its own TCP connection; no connection is reused across exchanges. The host aggregates the uploads with FedAvg. Table I summarizes the notation.

We use two forms of energy accounting, because the choice affects the answer. *Marginal energy* is the energy above the awake baseline power P_{base} and quantifies what computation and communication add. For one turn,

$$\Delta E_{\mathrm{active}}(E) = \Delta E_{\mathrm{comp}}(E) + \Delta E_{\mathrm{comm}}, \tag{1}$$

$$\Delta E_{\mathrm{comp}}(E) = \alpha E + c_0, \tag{2}$$

where ΔE_{comm} is independent of E and covers the model download and upload. The configuration request is excluded; Section VI-B reports the effect of including it. *Gross awake energy* additionally counts P_{base} over the active time

$$T_{\mathrm{active}}(E) = T_{\mathrm{comm}} + t_{\mathrm{ep}}E, \tag{3}$$

where t_{ep} is the duration of one epoch. If the device sleeps for the remainder of a period T_r at power P_s and incurs a wake-up energy E_{wake}, the energy of one turn is

$$E_{\mathrm{turn}}(E, T_r) = P_{\mathrm{base}}T_{\mathrm{active}}(E) + \Delta E_{\mathrm{active}}(E) + P_s\big(T_r - T_{\mathrm{active}}(E)\big) + E_{\mathrm{wake}}. \tag{4}$$

The primary metric is the energy of one client turn; a round with K clients costs $K\,E_{\mathrm{turn}}$.

B. Crossover and Design Problem

The crossover $E^\star$ is the number of epochs at which computation and communication cost the same energy. In standby, the condition is $\Delta E_{\mathrm{comp}}(E^\star) = \Delta E_{\mathrm{comm}}$, which yields $E_0^\star = (\Delta E_{\mathrm{comm}} - c_0)/\alpha$. When the device sleeps between turns, the awake baseline during the active window contributes to both terms, and the crossover becomes

$$E^\star(P_s) = \frac{(P_{\mathrm{base}} - P_s)\,T_{\mathrm{comm}} + \Delta E_{\mathrm{comm}} - c_0}{(P_{\mathrm{base}} - P_s)\,t_{\mathrm{ep}} + \alpha}. \tag{5}$$

The following result characterizes how sleep moves the crossover.

Proposition 1: Let $\alpha > 0$, $t_{\mathrm{ep}} > 0$ and $0 \le P_s \le P_{\mathrm{base}}$, and define $\rho = T_{\mathrm{comm}}/t_{\mathrm{ep}}$ and $w = (P_{\mathrm{base}} - P_s)t_{\mathrm{ep}}/\big((P_{\mathrm{base}} - P_s)t_{\mathrm{ep}} + \alpha\big)$. Then

1) $E^\star(P_s) = (1 - w)\,E_0^\star + w\,\rho$, with $0 \le w < 1$;
2) $E^\star(P_s)$ lies between $E_0^\star$ and ρ;
3) as P_s decreases, $E^\star(P_s)$ moves monotonically toward ρ. It increases if $T_{\mathrm{comm}} > E_0^\star\,t_{\mathrm{ep}}$, decreases if $T_{\mathrm{comm}} < E_0^\star\,t_{\mathrm{ep}}$, and is constant if the two are equal.

Proof: Let $\Delta = P_{\mathrm{base}} - P_s \ge 0$. Since $\Delta T_{\mathrm{comm}} = \Delta t_{\mathrm{ep}}\,\rho$ and $\Delta E_{\mathrm{comm}} - c_0 = \alpha\,E_0^\star$, the numerator of (5) equals $\Delta t_{\mathrm{ep}}\,\rho + \alpha\,E_0^\star$. Dividing by the denominator $\Delta t_{\mathrm{ep}} + \alpha > 0$ gives part 1, and $0 \le w < 1$ because $\alpha > 0$. Part 2 follows because a convex combination lies between its two points. For part 3,

$$\frac{\partial E^\star}{\partial \Delta} = \frac{\alpha T_{\mathrm{comm}} - t_{\mathrm{ep}}(\Delta E_{\mathrm{comm}} - c_0)}{(\Delta t_{\mathrm{ep}} + \alpha)^2} = \frac{\alpha\,t_{\mathrm{ep}}\,(\rho - E_0^\star)}{(\Delta t_{\mathrm{ep}} + \alpha)^2}, \tag{6}$$

whose sign is that of $\rho - E_0^\star$, i.e., of $T_{\mathrm{comm}} - E_0^\star t_{\mathrm{ep}}$. Since Δ increases as P_s decreases, the claim follows. ∎

Proposition 1 explains the sleep results of Section VII. The weight w is the share of the awake baseline in the gross energy of one epoch. On our board, $P_{\mathrm{base}}t_{\mathrm{ep}}$ is approximately four times α. With the measured sleep powers of Section VII, $w = 0.80$ in light and deep sleep and $w = 0.70$ in modem sleep, so the crossover moves 70% to 80% of the way from $E_0^\star$ to ρ. Consequently, the choice of sleep state matters little for the crossover, whereas the exchange time matters considerably.

The design problem is to select E such that the energy required to reach a target accuracy is minimized:

$$E_{\mathrm{opt}} = \arg\min_{E}\ R(E)\,E_{\mathrm{turn}}(E, T_r), \tag{7}$$

where $R(E)$ is the number of rounds required. The crossover $E^\star$ is a property of the hardware and the protocol, whereas E_{opt} additionally depends on the data, the learning rate and the target. Because the two are distinct quantities, neither is used to validate the other.

C. Contention

When other devices share the channel, each exchange takes longer. We model this effect with an airtime factor $\kappa_c \geq 1$ on the communication energy, $\Delta E_{\mathrm{comm}}(\kappa_c) = \kappa_c \, \Delta E_{\mathrm{comm}}$. This is a modeling device; κ_c was not measured.

IV. Analytical Baselines

We compare the measurements with four estimates: two analytical models that are common in the wireless FL literature, their combination, and a model built from datasheet values. For each, we report a nominal value and a range obtained by sweeping the stated parameter bounds; errors are relative to the main campaign of Section VI ($\Delta E_{\mathrm{comm}} = 9.75\,\mathrm{mJ}$, $\alpha = 1.002\,\mathrm{mJ}$ per epoch and $E^\star = 9.74$). The model calculations are included in the released analysis scripts.

- **Shannon capacity:** The communication energy is $P_{\mathrm{tx}} \cdot 8M/\big(W \log_2(1 + \mathrm{SNR})\big)$ with $M = 2 \times 3{,}120\,\mathrm{B}$. The nominal point uses a bandwidth $W = 20\,\mathrm{MHz}$, an SNR of 20 dB and $P_{\mathrm{tx}} = 150\,\mathrm{mW}$; the range spans W from 20 to 40 MHz, SNR from 10 to 30 dB [6], [7], [10], [11] and P_{tx} from 100 to 200 mW [43]. The model yields $0.056\,\mathrm{mJ}$ per turn (range 0.013 to $0.144\,\mathrm{mJ}$), 99.4% below the measurement. As in the cited works, P_{tx} is the transmit power, whereas the measurement is the additional power drawn by the whole board; the comparison therefore shows what the model leaves out on such a device, not an error in the capacity formula.
- **Switching energy:** The compute energy per epoch is $\kappa f^2 C$ [6], [7], [9], [12], [13], [16] with $f = 240\,\mathrm{MHz}$, $C \approx 1.8 \times 10^6$ cycles (the epoch time multiplied by f) and $\kappa = 5 \times 10^{-28}$ (range 10^{-28} to 10^{-27}). It yields $0.052\,\mathrm{mJ}$ per epoch (range 0.010 to $0.105\,\mathrm{mJ}$), 94.8% below the measurement.
- **Shannon and switching combined:** The crossover is the Shannon communication energy divided by the switching compute energy, 1.07 epochs (range 0.12 to 13.8), 89.0% below the measurement at the nominal point. Because the range includes the measured value, the combined model is uninformative rather than contradicted.
- **Datasheet currents:** The communication term applies an effective radio power of $536\,\mathrm{mW}$ at 3.3 V (range 450 to 650 mW), chosen between the receive and transmit consumption listed in the datasheet [43], for an assumed radio-active time of 20 ms per turn (range 15 to 25 ms). It yields $10.72\,\mathrm{mJ}$ (range 6.75 to $16.25\,\mathrm{mJ}$), 10.0% above the measurement. The computation term multiplies an incremental CPU power of $132\,\mathrm{mW}$ (range 100 to 160 mW) by the epoch time and yields $1.001\,\mathrm{mJ}$ per epoch (range 0.76 to $1.21\,\mathrm{mJ}$). Because both factors of this term are taken from our measurement (Section V), the computation term is calibrated rather than predicted, and its agreement within 0.1% is not evidence for the model. The resulting crossover is 10.7 epochs (range 5.6 to 21.4), 9.9% above the measurement.

The estimates deviate from the measurement for different reasons (Fig. 1). The Shannon and switching models underestimate the energies by 95% to 99%. The Shannon model counts only the payload bits at the physical-layer rate and omits medium-access and CSMA overhead, headers, the HTTP exchange and the analog radio draw; the switching model omits memory stalls, SRAM and flash access and leakage. Their combination produces a crossover whose range spans two orders of magnitude. The datasheet model lies within 10% of the measured communication energy at its nominal point, but this agreement is largely coincidental: the model charges a high power for a short radio-active time, whereas the measured exchange lasts $125\,\mathrm{ms}$ (median) at a median of $67\,\mathrm{mW}$ above baseline. The model also contains no term for the $561\,\mathrm{mW}$ that the board draws while awake, which is the largest energy in the system, and its crossover range spans a factor of four. Relative to the second session (Section VI-B), its communication energy would be 46% too high.

V. Testbed and Method

A. Model Split

The model is a 12-class keyword-spotting network for Google Speech Commands v2 [44]. A frozen DS-CNN-S backbone [38], [39] with 23,552 parameters, trained in floating point for the 12 classes and quantized to INT8, produces a 64-dimensional embedding. Training the whole network in float32 would have to keep, for back-propagation, at least the input of each of its nine convolutional layers, eight of which are $25 \times 5 \times 64$ feature maps: about $0.26\,\mathrm{MB}$ per sample and $4.1\,\mathrm{MB}$ for a batch of 16, plus $0.19\,\mathrm{MB}$ for the weights and their gradients. This exceeds the $512\,\mathrm{KiB}$ of internal SRAM [27]. The 8 MB external PSRAM of our board could hold these buffers, but at a lower memory bandwidth and with a much larger computation per epoch; we did not evaluate full-network training. As in [33], [34], only the float32 linear head is trained; its weights, gradients, one batch of embeddings and the outputs occupy about $15\,\mathrm{KB}$ of internal SRAM. The head has $64 \times 12 + 12 = 780$ parameters (3,120 B), is trained on the device with SGD implemented in C++, and is the only part transmitted over the air.

The measurements isolate the cost of training this head. The device therefore does not run the backbone during a turn. Instead, the firmware generates the 100 input embeddings and labels of each client with a deterministic pseudo-random generator, seeded per client and re-seeded at the start of every epoch, so that every epoch processes the same 100 samples. The forward pass, the backward pass and the update execute the same sequence of operations for any input values, so the measured energy per epoch closely approximates that of training the head on cached embeddings, plus the small cost of the generator, which takes the place of reading the embeddings from memory. A node that recomputed the embeddings in every epoch would additionally pay the inference energy of the backbone. The backbone and real audio are used in the convergence study of Section VIII. The K federated clients in our experiments are virtual: one physical device plays all of them. Each virtual client holds 100 samples, the learning rate is 0.01, and the batch size is 16.

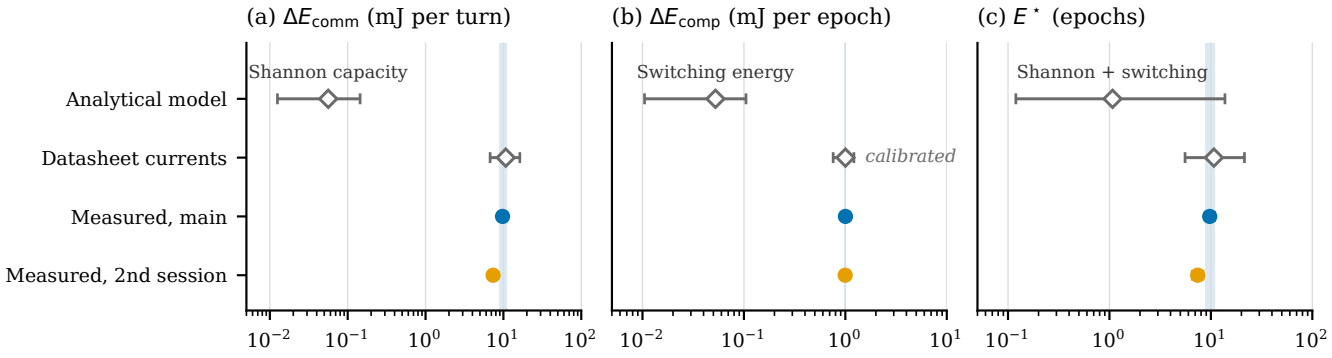
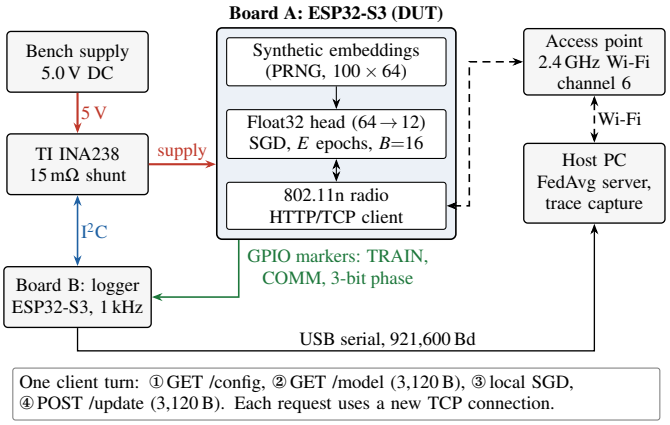


Fig. 1. Analytical models against the measurement, on logarithmic axes. Open diamonds are nominal model values with the range obtained from the parameter bounds of Section IV; the top row shows the analytical model named above its marker. The computation term of the datasheet model is calibrated to the measured plateau power and epoch time. Filled circles are measurements: main campaign (blue, 95% bootstrap interval shaded) and second session (amber).

(a) Energy measurement on the device



(b) Convergence study (simulation)

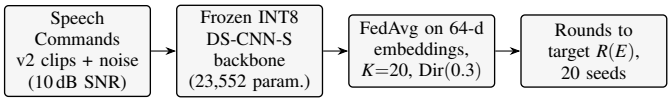


Fig. 2. System overview. (a) Energy measurement: the ESP32-S3 device under test (Board A) trains the float32 head on synthetic 64-dimensional embeddings and exchanges it with a FedAvg host over 2.4 GHz Wi-Fi. An INA238 on its 5 V supply, read by a second ESP32-S3 (Board B) at 1 kHz, measures its power, and GPIO markers label the phases of each turn. (b) Convergence study: the frozen INT8 backbone computes embeddings of noisy Speech Commands clips, which feed a FedAvg simulation that yields the rounds to target $R(E)$.

B. Measurement

The device under test is a generic ESP32-S3 development board with two USB-C ports, built around an ESP32-S3-WROOM-1-N16R8 module: two Xtensa LX7 cores at 240 MHz, 512 KiB of internal SRAM, 16 MB of flash, 8 MB of octal PSRAM and an 802.11 b/g/n radio. The firmware of all campaigns, including the sleep campaign of Section VII, was built with the Arduino PSRAM option disabled, and the training code keeps all its buffers in internal SRAM. A TI INA238 with a 15 mΩ (1%) high-side shunt is placed in the supply line, and a second ESP32-S3 reads it at 1 kHz and streams the samples at 921,600 baud. The device raises GPIO markers during training and during HTTP exchanges, and a 3-bit phase bus labels the sub-phases, so that every sample is tagged as training, communication or idle. Energy is the time integral of power. The board is supplied with 5 V DC at its input, which feeds the on-board 3.3 V regulator; all reported powers therefore refer to the board at its 5 V input. Appendix A describes the wiring and isolation, and Fig. 2 summarizes the testbed.

All 105 runs use one firmware build, one board pair, one access point and fixed cables. The access point is a phone hotspot on channel 6 at a distance of 0.5 m from the board; the received signal strength, read at the start and end of the campaign, was −40 to −41 dBm. Each run includes 30 s of idle before the turn and 30 s after it. Algorithm 1 specifies the energy attribution and the acceptance test applied to every trace. The baseline power of a run is the median power from 5 s to 25 s of each idle period, averaged over the two periods, and the energy above baseline is the sum of $(P - P_{\text{base}})\,\Delta t$ over the samples of a phase, with negative terms retained. A run is accepted only if it has no dropped samples, strictly increasing timestamps, the expected number of markers and a baseline drift below 2% between the two idle periods. Rejected attempts were retained and repeated; twenty attempts were rejected, mostly because the training marker never appeared. No sample was repaired or interpolated.

We conducted two campaigns. The *main campaign* comprises 60 runs, with $E \in \{1, 2, 5, 8, 10, 20\}$, ten repetitions each, a randomized run order and a 60 ms pause between the phases of a turn. The *tail test* comprises 45 runs, with $E \in \{1, 5, 20\}$, pauses of 0, 60 and 200 ms and five repetitions per cell. It was executed in five blocks, each of which traversed all nine cells in a fixed order.

C. Power Profile

Figure 3 shows one run of the main campaign. With modem sleep disabled, the awake idle board draws $P_{\text{base}} = 561.1 \pm 1.8$ mW over the 105 runs, and the baseline of individual runs ranges from 557.8 to 569.7 mW. In the run shown, the baseline is 560.5 mW and the SGD plateau is 691.5 mW, i.e., 131 mW above baseline, for 152 ms. Across the campaign,

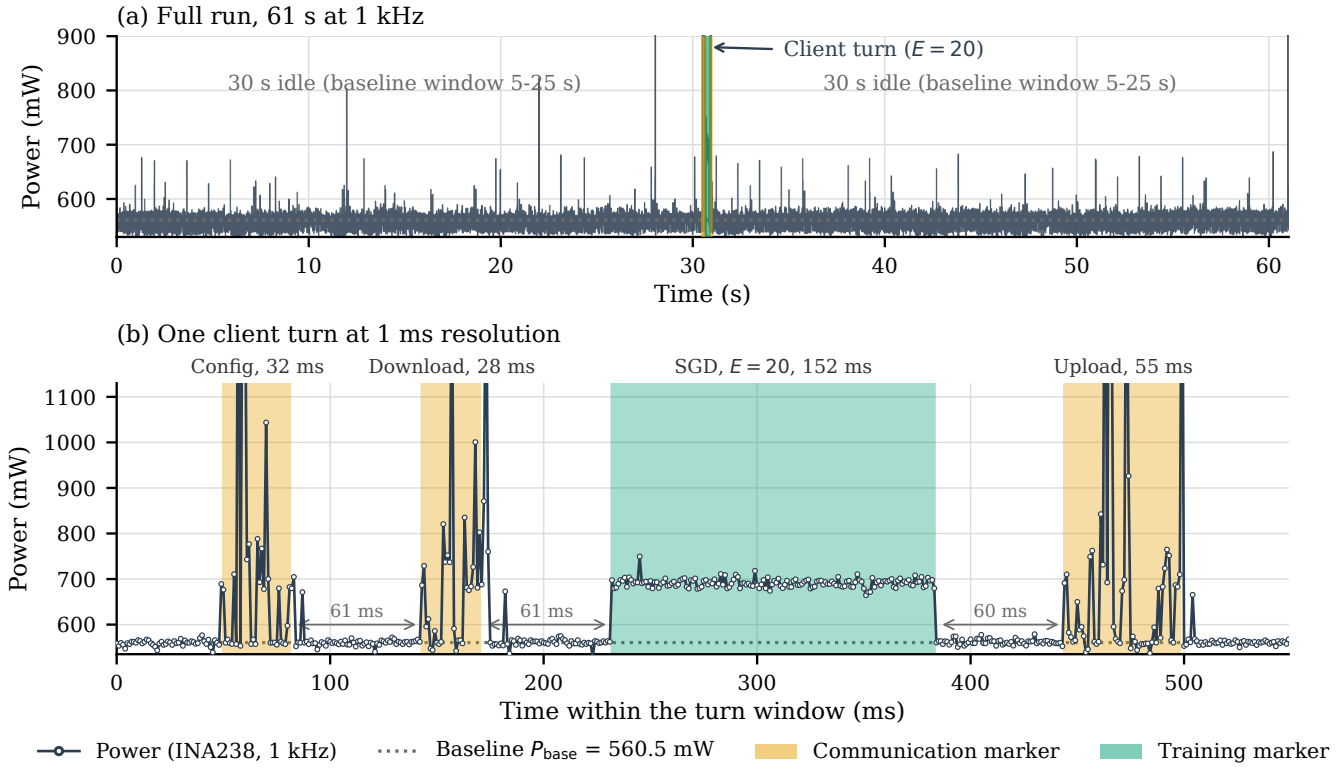


Fig. 3. Power trace of one main-campaign run ($E = 20$) at 1 kHz. (a) The full 61 s run, with 30 s of idle before and after the turn. (b) One client turn: configuration request, model download, SGD, model upload and the 60 ms pauses between them.

TABLE II
MEASURED MARGINAL ENERGY PER CLIENT TURN IN THE MAIN CAMPAIGN ($N = 10$ PER REGIME). VALUES ARE THE MEAN ± THE STANDARD DEVIATION. THE RATIO IS THE MEAN COMPUTATION ENERGY OVER THE MEAN COMMUNICATION ENERGY.

E	T_{train} (ms)	ΔE_{comp} (mJ)	ΔE_{comm} (mJ)	Median comm.	Ratio
1	7.5 ± 0.5	0.99 ± 0.06	8.5 ± 3.2	7.5	0.12
2	15.1 ± 0.3	2.00 ± 0.06	9.4 ± 2.9	9.0	0.21
5	37.9 ± 0.3	5.01 ± 0.10	10.6 ± 6.9	8.5	0.47
8	60.7 ± 0.5	8.01 ± 0.19	11.3 ± 5.0	9.6	0.71
10	75.8 ± 0.4	9.98 ± 0.25	7.9 ± 2.2	7.5	1.27
20	151.6 ± 0.5	20.04 ± 0.37	10.8 ± 4.3	11.1	1.86

one epoch takes 7.58 ms and costs 1.00 mJ above baseline, corresponding to 132 mW. Individual samples in the radio phases reach approximately 1 W. The 60 ms pauses allow the radio tail to decay before training starts (Section VI-C).

VI. MEASURED CROSSOVER

A. Energy per Turn

Table II and Fig. 4 report the measured energy per client turn in the main campaign. Computation grows almost linearly with E and has a small spread. The mean communication energy per value of E ranges from 7.9 to 11.3 mJ without a systematic trend; a linear fit of the communication energy on E has a slope of 0.07 ± 0.09 mJ per epoch.

A least-squares fit of the 60 runs gives

$$\Delta E_{\text{comp}}(E) = 1.002\,E - 0.010 \text{ mJ}, \qquad R^2 = 0.999, \quad (8)$$

with 95% bootstrap intervals of $[0.991, 1.014]$ for the slope and $[-0.070, +0.048]$ for the intercept; the intercept is therefore zero to within approximately 0.07 mJ. The slope agrees with the product of the plateau power above baseline and the epoch time, $132\,\text{mW} \times 7.58\,\text{ms} = 1.00\,\text{mJ}$. Because the communication energy shows no trend in E, we represent it by its mean over the 60 runs, $\Delta E_{\text{comm}} = 9.75$ mJ, with a 95% bootstrap interval of $[8.74, 10.91]$.

Communication energy is considerably more variable than computation energy: its standard deviation is 2 to 7 mJ per regime, compared with 0.06 to 0.4 mJ for computation. The variability originates in the radio exchange. Most exchanges take 84 to 163 ms (the middle half of all 105 runs), but some stall (Fig. 5a). In 6 of the 105 runs the exchange took more than 500 ms, and the longest took 2.8 s. A stall adds energy because the radio remains active, which is why the mean (9.75 mJ) exceeds the median (8.85 mJ). Duration alone does not explain the spread, however: across all runs, the correlation between exchange time and communication energy is only $r = 0.31$ (0.56 when the six stalls are excluded).

B. Crossover

The standby crossover is $E_0^\star = (\Delta E_{\text{comm}} - c_0)/\alpha$. Table III reports it under several accounting choices. The first row counts only the energy of the two model exchanges, as delimited by the markers, and gives $E^\star = 9.74$ epochs. The second row additionally charges to communication the 0.49 mJ of radio tail energy that a training burst absorbs when it starts

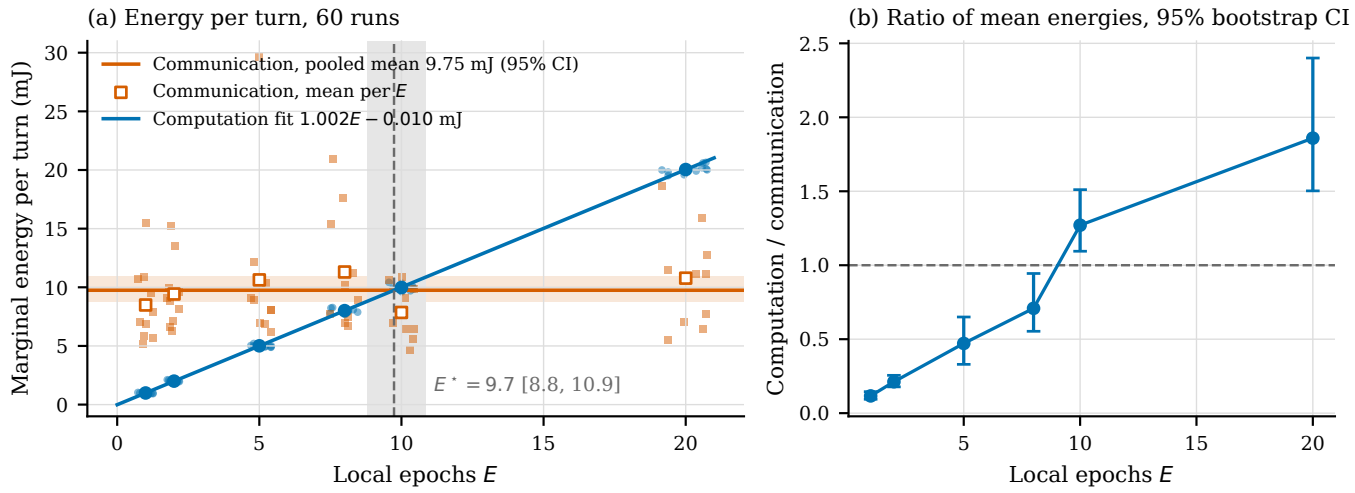


Fig. 4. Measured marginal energy per client turn in the main campaign. (a) Each small marker is one run. The computation energy grows linearly with E (blue line: least-squares fit). The communication energy shows no trend in E and is represented by its pooled mean over the 60 runs (orange line) with its 95% bootstrap interval (orange band); open squares are the means for each E. The grey band is the 95% bootstrap interval of $E^{\star}$. (b) Ratio of the mean computation energy to the mean communication energy for each E, with 95% bootstrap intervals; the ratio crosses one between $E = 8$ and $E = 10$.

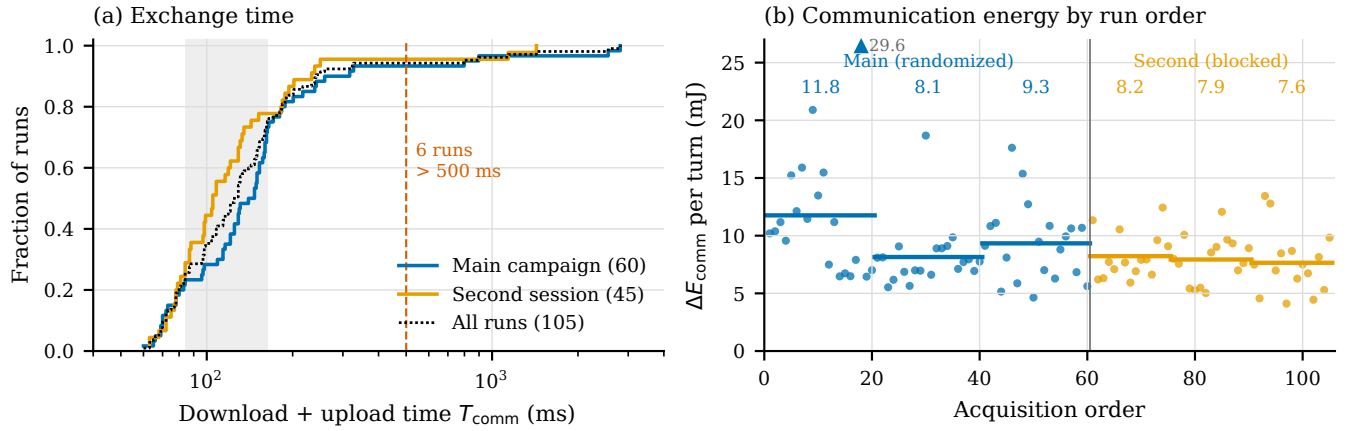


Fig. 5. Communication diagnostics for all 105 runs. (a) Cumulative distribution of the download and upload time; the shaded band is the middle half of the runs. (b) Communication energy per turn in acquisition order. Horizontal bars and numbers give the mean of each third of a campaign. One main-campaign run (29.6 mJ) is off scale and marked at the top edge.

immediately after a radio burst, as measured in Section VI-C, and gives $E^{\star} = 10.23$ epochs. The configuration request costs 3.4 mJ on average; counting it as communication raises $E^{\star}$ to 13.1 epochs. Below the crossover, communication accounts for most of the marginal energy, and above it computation does; at $E = 20$, computation contributes 65% of the marginal energy of a turn.

The main campaign is not the only measurement. The tail test contains 15 runs at the same 60 ms setting with $E \in \{1, 5, 20\}$, which give $\Delta E_{\mathrm{comm}} = 7.37$ mJ and $E^{\star} = 7.42$ epochs. For the same values of E, the main campaign gives 9.97 mJ, 2.6 mJ more (95% bootstrap interval $[0.7, 4.7]$ mJ; permutation test, $p = 0.05$). This difference is not a fixed offset between sessions (Fig. 5b). In the main campaign, the mean communication energy of the first, second and last third of the runs is 11.8, 8.1 and 9.3 mJ (Spearman correlation with run order -0.23), whereas over all 45 runs of the second session it is 8.2, 7.9 and 7.6 mJ. Because the main campaign was randomized, this drift is distributed over all values of E and is unlikely to create or hide a trend in E. Retries and the load

TABLE III
ESTIMATES OF THE MARGINAL CROSSOVER $E^{\star}$ UNDER DIFFERENT ACCOUNTING CHOICES, WITH 95% BOOTSTRAP INTERVALS. "RADIO TAIL" ADDS THE 0.49 MJ OF POST-BURST ENERGY MEASURED IN SECTION VI-C; ITS INTERVAL RESAMPLES THE MAIN CAMPAIGN AND THE TAIL TEST JOINTLY.

Accounting	ΔE_{comm} (mJ)	$E^{\star}$ (epochs)	95% interval
Model exchanges only (markers)	9.75	9.74	[8.79, 10.85]
Model exchanges + radio tail	10.24	10.23	[9.20, 11.39]
Median communication energy	8.85	8.84	n/a
Configuration request included	13.12	13.1	[11.9, 14.4]
Second session (60 ms, $N = 15$)	7.37	7.42	[6.54, 8.36]

Algorithm 1 Phase-resolved energy attribution and run acceptance

Require: Trace $\{(t_n, P_n, g_n^{\mathrm{tr}}, g_n^{\mathrm{co}}, d_n)\}$ at $\Delta t = 1\,\mathrm{ms}$: power, training and communication marker lines, dropped-sample count; expected training time $\hat{T}(E)$

$\mathcal{T}_1, \ldots, \mathcal{T}_J \leftarrow$ pulses of g^{tr}; $\mathcal{C}_1, \ldots, \mathcal{C}_S \leftarrow$ pulses of g^{co}, in time order
if $\sum_n d_n > 0$ **or** t not strictly increasing **or** $J \neq 1$ **or** $S < 3$ **or** $|\mathcal{T}_1|\Delta t \notin [0.6, 1.5]\,\hat{T}(E)$ **then**
  return REJECT
end if
$n_a \leftarrow$ first marker sample; $n_b \leftarrow$ last marker sample
$b_1 \leftarrow \mathrm{median}\{P_n : 5\,\mathrm{s} \le t_n - t_0 < 25\,\mathrm{s}\}$
$b_2 \leftarrow \mathrm{median}\{P_n : 5\,\mathrm{s} \le t_n - t_{n_b} < 25\,\mathrm{s}\}$
if $t_{n_a} - t_0 < 25\,\mathrm{s}$ **or** $|b_2 - b_1|/b_1 > 2\%$ **then**
  return REJECT
end if
$P_{\mathrm{base}} \leftarrow (b_1 + b_2)/2$
$\Delta E_{\mathrm{comp}} \leftarrow \sum_{n \in \mathcal{T}_1} (P_n - P_{\mathrm{base}})\Delta t$ ▷ negative terms kept
$\Delta E_{\mathrm{cfg}}, \Delta E_{\mathrm{dl}}, \Delta E_{\mathrm{ul}} \leftarrow \sum_{n \in \mathcal{C}_s} (P_n - P_{\mathrm{base}})\Delta t$ for $s = 1, 2, 3$
$\Delta E_{\mathrm{comm}} \leftarrow \Delta E_{\mathrm{dl}} + \Delta E_{\mathrm{ul}}$; $T_{\mathrm{comm}} \leftarrow (|\mathcal{C}_2| + |\mathcal{C}_3|)\Delta t$
return ACCEPT and P_{base}, ΔE_{comp}, ΔE_{cfg}, ΔE_{comm}, T_{comm}

of the phone hotspot were not logged, so the change in radio cost within and between sessions cannot be attributed to a specific cause. Because the communication energies are right-skewed, the evidence for this difference is moderate; since it nevertheless exceeds the tail effect of the next subsection, we report the crossover as a range of 7.4 to 9.7 epochs, with the main campaign, which has the most runs, at the upper end. The computation side is stable: the tail test gives a slope of $0.996\,\mathrm{mJ}$ per epoch at $60\,\mathrm{ms}$, in agreement with 1.002.

C. The Radio Tail

After a radio burst, the power requires a short time to return to the baseline. Cellular radios are known for long tails that can account for most of the energy of a transfer [40]. The Wi-Fi tail observed here is much shorter, but if training starts immediately, the end of the tail falls within the training window. The tail test examines this effect by comparing back-to-back phases (0 ms) with pauses of 60 ms and 200 ms.

Figure 6 presents the result. The fitted slope is 1.020, 0.996 and $1.006\,\mathrm{mJ}$ per epoch for delays of 0, 60 and 200 ms, so the delay does not affect the per-epoch cost. Without a pause, however, the intercept is $+0.48\,\mathrm{mJ}$, compared with $-0.01\,\mathrm{mJ}$ with a 60 ms pause. The difference is $0.49\,\mathrm{mJ}$, with a 95% interval of $[0.18, 0.82]$, and a permutation test that shuffles the delay labels within each E gives $p = 0.04$. A pause of 200 ms yields the same result as 60 ms; their intercepts differ by $0.03\,\mathrm{mJ}$, with an interval of $[-0.07, 0.14]$. The evidence is moderate, because each cell contains only five runs and the excess energy at 0 ms varies across E: it is 0.66, 0.43 and $1.02\,\mathrm{mJ}$ at $E = 1$, 5 and 20, respectively.

The traces reveal the origin of the excess energy. Figure 7 shows the mean power above baseline in the 60 ms pause after each phase. The pauses after the model download, the configuration request and the upload contain 0.75, 0.76 and $0.59\,\mathrm{mJ}$, most of it within the first 20 ms, whereas the pause after training contains only $0.11\,\mathrm{mJ}$. A radio burst thus leaves approximately 0.6 to $0.8\,\mathrm{mJ}$ behind, whereas a training burst leaves little. In the five back-to-back runs at $E = 20$, the first 10 ms of training draw on average 171 mW above baseline, compared with 130 to 131 mW when training starts after a pause. These observations are consistent with the excess of 0.5 to $1\,\mathrm{mJ}$ in the fit.

TABLE IV
PHASES OF ONE CLIENT TURN FROM THE PHASE BUS. TIMES ARE THE MEDIAN WITH THE MIDDLE HALF OF THE RUNS IN BRACKETS, FOR ALL 105 RUNS. ENERGIES ARE THE MEAN ± THE STANDARD DEVIATION ABOVE BASELINE.

Phase	Protocol state	Time (ms)	Energy (mJ)
Config request	HTTP GET, configuration	41 [28, 69]	3.3 ± 2.5
Download	HTTP GET, model head	37 [27, 57]	3.6 ± 1.7
Training	Local SGD (E epochs)	$7.58E$	$1.00E$
Upload	HTTP POST, update	85 [46, 115]	5.4 ± 3.0
Download and upload		125 [84, 163]	9.0 ± 3.7

Two consequences follow. First, a measurement in which training starts immediately after a radio burst overstates the computation energy by approximately $0.5\,\mathrm{mJ}$, which is half an epoch at $E = 1$. Second, the tail is a genuine communication cost that is incurred after every burst. Charging the full post-burst energy after both the download and the upload ($0.75 + 0.59 = 1.34\,\mathrm{mJ}$, Fig. 7) to communication would raise $E^\star$ by approximately 1.3 epochs; the second row of Table III, which charges only the absorbed $0.49\,\mathrm{mJ}$, is therefore conservative. The pause is a measurement device: deployed firmware need not wait for the tail, but the tail energy is incurred in either case.

D. Link-Layer Breakdown

The phase bus shows how the time of an exchange is distributed. Table IV lists the result for all 105 runs. The payload itself is small: at the 802.11n rates that a link at $-40\,\mathrm{dBm}$ supports, transferring $3{,}120\,\mathrm{B}$ takes well under a millisecond, and even at the lowest OFDM rate of 6 Mb/s it takes about 4 ms. Nevertheless, the download takes approximately 37 ms and the upload approximately 85 ms (medians), so most of the time is spent in the protocol and the radio rather than in transmitting payload bits. This observation also helps to explain the failure of the Shannon model. The upload costs more than the download, consistent with its longer duration and with the radio transmitting, and the configuration request, which carries almost no data, costs as much as the model download. The communication time of a turn is 125 ms at the median, with a long tail caused by stalls. The sleep analysis below uses T_{comm} from 84 to 163 ms, the middle half of the runs.

VII. SLEEP AND CHANNEL CONTENTION

The sleep powers and wake-up energies in this section were measured on the same board and at the same 5 V input as P_{base}; channel contention was not measured.

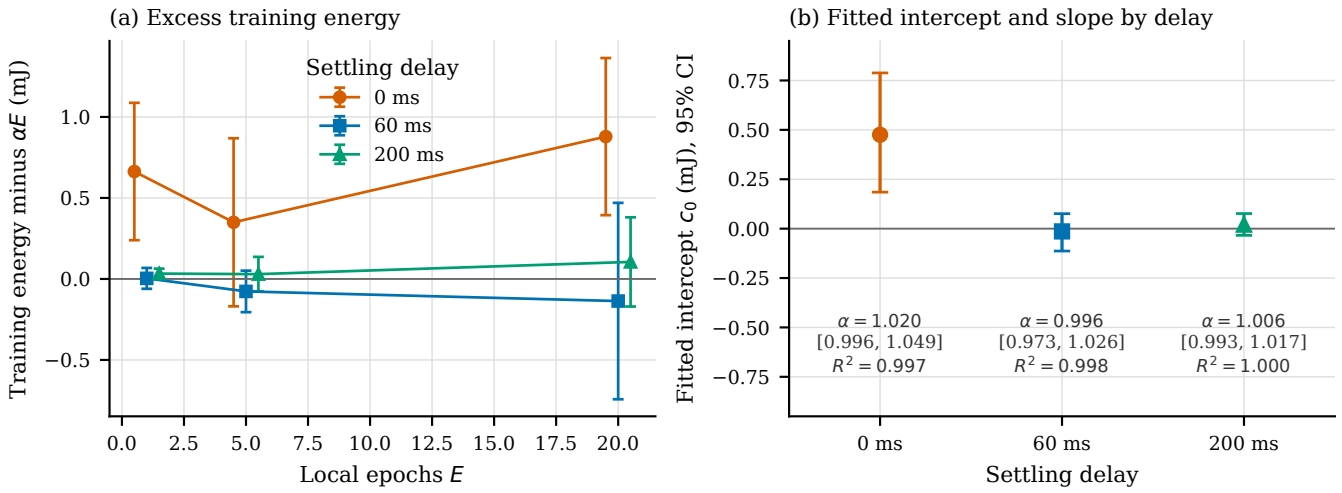


Fig. 6. Tail test, 15 runs per settling delay (five each at $E = 1$, 5 and 20). (a) Training energy above the line αE, with $\alpha = 1.002$, for three settling delays; bars show one standard deviation. (b) Fitted intercept c_0 of training energy against E for each delay, with its 95% bootstrap interval; the fitted slope α, its interval and R^2 are printed below each point.

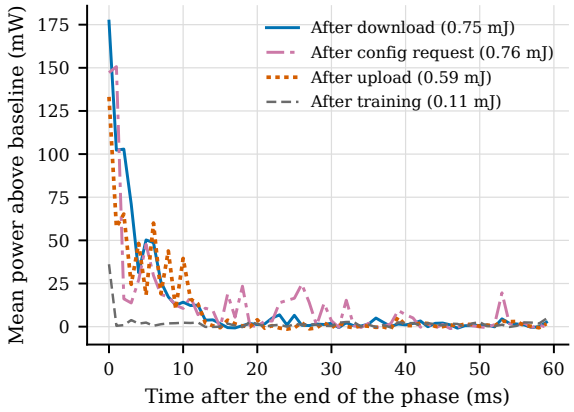


Fig. 7. Mean power above baseline after the end of each phase, taken from the 60 ms pauses of 74 of the 75 runs with this setting; one main-campaign run ($E = 5$) is excluded because its communication marker shows four pulses instead of three, so its pauses cannot be assigned to phases unambiguously. The numbers in the legend are the energy in the first 60 ms.

A. Measured Sleep Power and Wake-Up Energy

A dedicated campaign measured the board in four states with the instrument of Section V. Each of 20 cycles holds the board for 30 s in each of the following states: active standby, the condition of P_{base}; modem sleep, in which the station stays associated and the radio sleeps between beacons (Wi-Fi power save `WIFI_PS_MIN_MODEM`) while the CPU idles at 240 MHz; light sleep, entered by the firmware with a timer wake-up; and deep sleep with a timer wake-up. Before light and deep sleep the firmware stops Wi-Fi, and after each wake-up it re-associates with the access point and obtains an address through DHCP. The phase bus delimits every state with short pulses and holds a dedicated code from the first instruction after a deep-sleep wake-up, and from a light-sleep wake-up, until an address is obtained; the marker lines are low throughout the sleep windows, so the pull-down resistors draw no current there. The sleep powers are plateau means with 0.5 s trimmed from both ends of each window. The wake-up energy is the gross energy from the instant at which the power leaves the deep-sleep floor, when the timer expires, until the address is obtained. For this campaign the INA238 uses its ± 40.96 mV range (83 µA per least significant bit). All 20 cycles passed the quality gates of Appendix A, without dropped samples or failed reconnections.

Figure 8 shows one cycle, and Table V lists the means. Modem sleep reduces the board power from 561 to 254 mW (95% bootstrap interval $[251, 257]$), light sleep to 35.8 mW (7.03 mA at 5.09 V) and deep sleep to 17.16 mW (3.37 mA at 5.10 V), with cycle-to-cycle standard deviations of 6.8, 0.18 and 0.02 mW. These are board-level values. The datasheet lists chip-level currents of 240 µA in light sleep and 7 to 8 µA in deep sleep [43]. We attribute the remaining several milliamperes to the regulator, LEDs and USB circuitry of the development board, which a purpose-built node would remove; we did not measure these components separately. On an ESP32-S2 development board, Barral Vales et al. similarly found that the USB-UART bridge and the regulator, not the module, drew most of the deep-sleep current [42]. The INA238 shunt offset (± 1.5 µV typical, ± 5 µV maximum) bounds the offset error of every power at ± 0.5 mW typical and ± 1.7 mW maximum, i.e., 3% and 10% of the deep-sleep power.

Waking up is expensive. A deep-sleep wake-up costs 684 mJ on average (95% interval $[647, 727]$ mJ, median 633 mJ) and lasts 1.27 s. Booting takes only 55 ms; the remaining 1.22 s are spent starting Wi-Fi, scanning, associating and obtaining an address, at a mean power of 537 mW with peaks above 1 W (Fig. 8b). These steps are communication, but they occur once per wake-up and are not part of T_{comm}, so we include them in E_{wake}. Re-association after light sleep costs 757 mJ on average (median 629 mJ); one of the 20 reconnections took 3.6 s, which raises the mean. Light sleep therefore saves

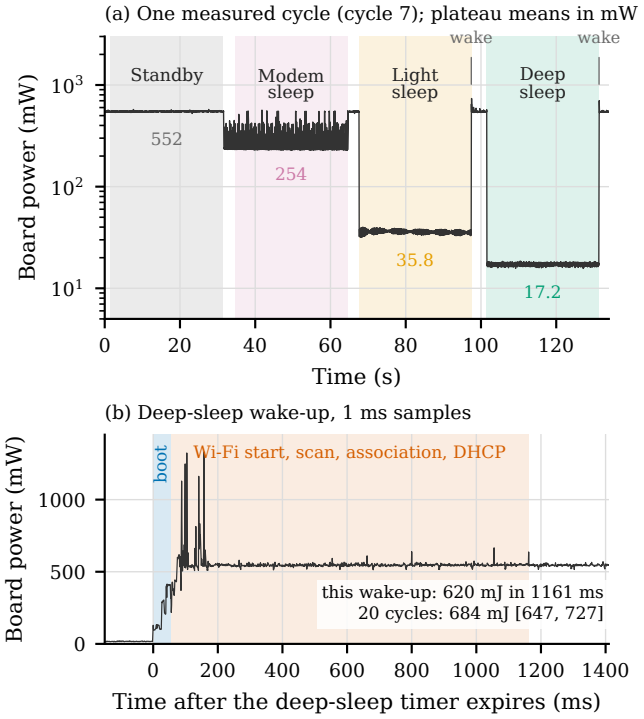


Fig. 8. Measured board power in the sleep campaign. (a) One of the 20 cycles (10 ms means, logarithmic axis) with the plateau means over all cycles. The standby level (552 mW) is the mean of this session and lies below the P_{base} of the FL campaigns (561.1 mW, Table V) because most standby periods of this session followed a deep-sleep wake-up (Section VII-A). (b) A deep-sleep wake-up at 1 ms resolution: boot, followed by Wi-Fi start, scan, association and DHCP.

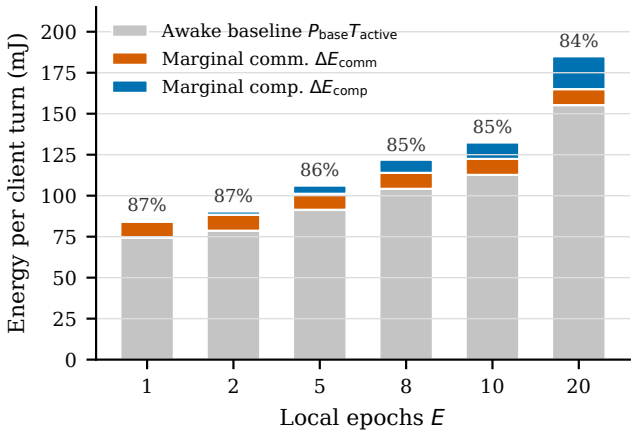


Fig. 9. Gross awake energy of one client turn in standby, from the measured fits with $T_{\text{comm}} = 125$ ms. The configuration request is not included. Numbers above the bars are the share of the awake baseline $P_{\text{base}}T_{\text{active}}$.

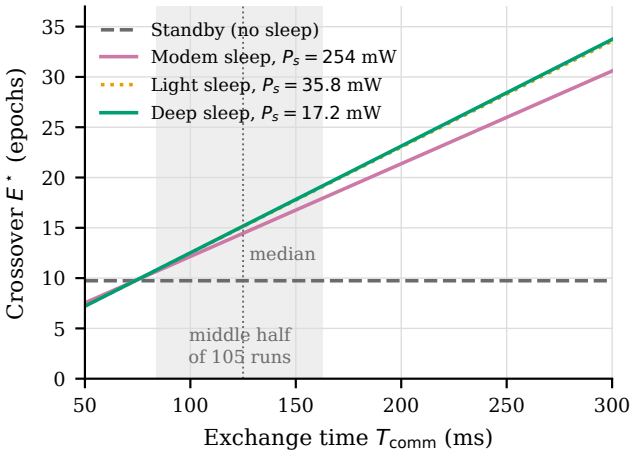


Fig. 10. Crossover from Eq. (5) against the exchange time, with measured α, c_0, ΔE_{comm}, P_{base}, t_{ep} and board-level sleep powers ($\kappa_c = 1$). The light-sleep curve almost coincides with the deep-sleep curve. The shaded band is the middle half of the measured exchange times. The curves combine measured inputs through Eq. (5).

sleep power but not wake-up energy in our setup, because our firmware stops Wi-Fi before light sleep; the default build of the framework that we use, Arduino-ESP32 3.3.6, does not enable the automatic light sleep that keeps the association alive between beacons. Entering either sleep state costs 4.3 mJ. A deep-sleep wake-up thus costs as much as 1.2 s at the awake baseline, almost four times the 180 mJ that we had assumed before this campaign on the basis of a 0.32 s reconnection at P_{base}, and 35 times the marginal energy of a turn with $E = 10$.

During this campaign, the standby power was 561.6 mW in the first cycle, which followed a cold start, and 551 mW on average in the 19 cycles that followed a deep-sleep wake-up. We use the P_{base} of the main campaign in all models; the sleep powers do not depend on it.

B. Sleep Moves the Crossover

In standby, the awake baseline is incurred whether or not the device performs work, so it cancels in the marginal comparison. If the device sleeps between turns, the baseline during the active window becomes a real cost, and both a longer exchange and every additional epoch cost more. One epoch takes 7.58 ms at 561 mW, i.e., 4.3 mJ, more than four times its marginal 1.00 mJ. Figure 9 shows the magnitude of this term: the awake baseline over the active window accounts for 84% to 87% of the gross energy of a turn for every E tested.

Evaluating Eq. (5) with the measured sleep powers gives Table V and Fig. 10. The crossover rises from 9.7 to between 11 and 19 epochs once sleep is possible, and the curves for light and deep sleep almost coincide, while modem sleep lies slightly lower (10.7 to 18.0). Both observations follow from Proposition 1: with w between 0.70 and 0.80, the crossover moves most of the way from $E_0^\star = 9.74$ toward $\rho = T_{\text{comm}}/t_{\text{ep}}$, which is 11.1, 16.5 and 21.5 for $T_{\text{comm}} = 84$, 125 and 163 ms. The exchange time therefore matters much more than the sleep state, which is why the stalls described in Section VI-A (Fig. 5a) are relevant to the designer. The threshold of part 3 of the proposition is $E_0^\star t_{\text{ep}} = 73.9$ ms. Only 14 of the 105 measured exchanges were shorter, so for most turns sleep raises the crossover.

Two further observations follow from the measured values. In standby, a 60 s turn costs approximately 33.7 J regardless of E, so idling dominates. In deep sleep, the same turn costs

TABLE V
MEASURED BOARD-LEVEL SLEEP POWER P_s AND GROSS WAKE-UP ENERGY E_{wake} (MEANS OVER 20 CYCLES), CROSSOVER FROM EQ. (5) AT THE QUARTILES OF THE MEASURED EXCHANGE TIME, AND ENERGY PER TURN AT $T_r = 60$ S USING $T_{\mathrm{comm}} = 125$ MS. LIGHT AND DEEP SLEEP INCLUDE RE-ASSOCIATION WITH THE ACCESS POINT.

State	P_s (mW)	E_{wake} (mJ)	$E^\star$ for $T_{\mathrm{comm}} =$		E_{turn} (J)		
			84 ms	163 ms	E=1	E=5	E=20
Standby	561.1	0	9.7	9.7	33.68	33.68	33.70
Modem sleep	253.9	0	10.7	18.0	15.29	15.30	15.35
Light sleep	35.8	757	10.8	19.1	2.99	3.01	3.08
Deep sleep	17.16	684	10.8	19.2	1.80	1.82	1.89

approximately 1.8 J, of which the sleep floor contributes about 55%, the wake-up about 37% and the active window, including all computation and the model exchange, only 5% to 10% for E from 1 to 20. Modem sleep, which keeps the association and needs no wake-up, still costs 15.3 J because the CPU stays awake. On this board, light and deep sleep therefore remove most of the idle energy, deep sleep the most, and in either state every saved round saves a wake-up of about 0.7 J. The value of a saved round thus depends on the sleep regime, which is a power-management decision rather than a learning hyperparameter.

C. Contention

With the airtime factor of Section III, the standby crossover is $E^\star(\kappa_c) = (\kappa_c \cdot 9.748 + 0.0098)/1.002$, which gives 9.7, 14.6, 19.5 and 29.2 for $\kappa_c = 1$, 1.5, 2 and 3. If the device sleeps and contention also stretches the awake exchange time, the effect is stronger: with the measured deep-sleep power and $T_{\mathrm{comm}} = 125\kappa_c$ ms, the crossover grows from 15.2 at $\kappa_c = 1$ to 45.5 at $\kappa_c = 3$. A congested channel lengthens each exchange and raises the crossover, so additional local work per round becomes more favorable; the magnitude of this effect depends on the channel load, which was not varied.

VIII. CONVERGENCE AND ROUNDS TO TARGET

A. Setup

On easy tasks, every setting reaches the target within one round, which conceals the trade-off; we therefore use a harder simulated task. The simulation trains the same 780-parameter head with FedAvg on embeddings from the frozen backbone of Section V, computed from clips mixed with background noise at an SNR of 10 dB. The training pool contains 200 clips for each of the ten keywords and the unknown class; the silence class is excluded. There are $K = 20$ clients with 100 disjoint clips each, Dirichlet label skew with concentration 0.3, batch size 16, up to 30 rounds and 20 random seeds. The 1,650 held-out clips are split per class into a validation half, used only for selecting the learning rate, and a test half, used for every reported number. For each E, the learning rate is the value in $\{0.003, 0.01, 0.03, 0.1, 0.3, 1.0\}$ that minimizes the mean number of rounds to 75% validation accuracy; the selected rates are 0.3 for $E \leq 2$ and 1.0 otherwise. Rounds to 75% test accuracy are fractional and are obtained by linear interpolation between rounds, starting from the accuracy of the initial model. All 20 seeds reach the target for every E, and the final test accuracy after 30 rounds is 78.7% to 79.4%.

TABLE VI
ROUNDS AND ENERGY PER CLIENT TO REACH 75% TEST ACCURACY IN THE $K = 20$ NON-IID SIMULATION (MEAN OVER 20 SEEDS, 95% BOOTSTRAP INTERVAL FOR $R(E)$). MARGINAL AND GROSS ENERGIES USE THE MEASURED FIT. THE LAST COLUMN USES THE MEASURED DEEP-SLEEP VALUES $P_s = 17.16$ MW AND $E_{\mathrm{wake}} = 684$ MJ WITH $T_r = 60$ S. BOLD MARKS THE MINIMUM OF EACH COLUMN.

E	η	$R(E)$	Traffic (kB)	Marginal (J)	Gross (J)	Deep sleep (J)
1	0.3	5.47 [4.93, 6.08]	34.1	0.059	0.465	9.82
2	0.3	3.18 [2.83, 3.54]	19.9	0.037	0.288	5.73
5	1.0	2.06 [1.82, 2.36]	12.9	0.030	0.219	3.75
8	1.0	1.61 [1.41, 1.81]	10.0	0.029	0.196	2.94
10	1.0	1.33 [1.19, 1.48]	8.3	**0.026**	**0.177**	2.46
20	1.0	1.28 [1.16, 1.40]	8.0	0.038	0.236	**2.42**

B. Rounds, Traffic and Energy

Table VI lists the simulated rounds $R(E)$ together with the energies derived from them. We combine $R(E)$ with the fitted model of Eq. (2), the main-campaign ΔE_{comm} and $T_{\mathrm{active}} = 125 + 7.58E$ ms, and, for the last column, with the measured deep-sleep power and wake-up energy of Section VII. One client transfers $2 \times 3{,}120 = 6{,}240$ B per round, so the traffic and the number of channel accesses to the target scale with $R(E)$. From $E = 1$ to $E = 20$, the traffic per client decreases from 34.1 kB to 8.0 kB, and the number of rounds, each comprising a download and an upload, decreases from 5.5 to 1.3.

C. Crossover and Learning Optimum

For marginal and gross energy, the energy to reach the target is minimized at $E = 10$ (Table VI, Fig. 11). With the measured deep-sleep values, the minimum moves to $E = 20$, but $E = 10$ is practically tied with it. Table VII examines how robust these optima are. They are broad rather than sharp: individual seeds favor values from $E = 2$ to $E = 20$, and the accounting mainly changes the cost of departing from the optimum. Because each deep-sleep round carries a 0.68 J wake-up, which exceeds the active energy of a turn, sleep strongly penalizes too little local work and tolerates too much, consistent with the higher crossover of Section VII.

The shape of $R(E)$ explains the optimum. Rounds decrease rapidly up to $E = 10$ and then approach a floor, with $R(20)$ only 4% below $R(10)$, while the compute energy per turn keeps growing linearly. Part of this floor is imposed by the 75% target, which several seeds reach within the first round at large E. Stricter targets, which almost no seed reaches within one round, leave all three optima unchanged, and so do wake-up energies across and beyond the measured interval (Table VIIb). Only the idealized sleep values that we assumed before the sleep campaign, which shrink the fixed cost of a round, return the deep-sleep optimum to $E = 10$. On the measured board, the sleep floor (about 1.03 J per 60 s round) and the wake-up are fixed per-round costs that push the optimum toward more local work.

That the learning optimum lies near the marginal crossover is a property of this task and constitutes evidence for neither

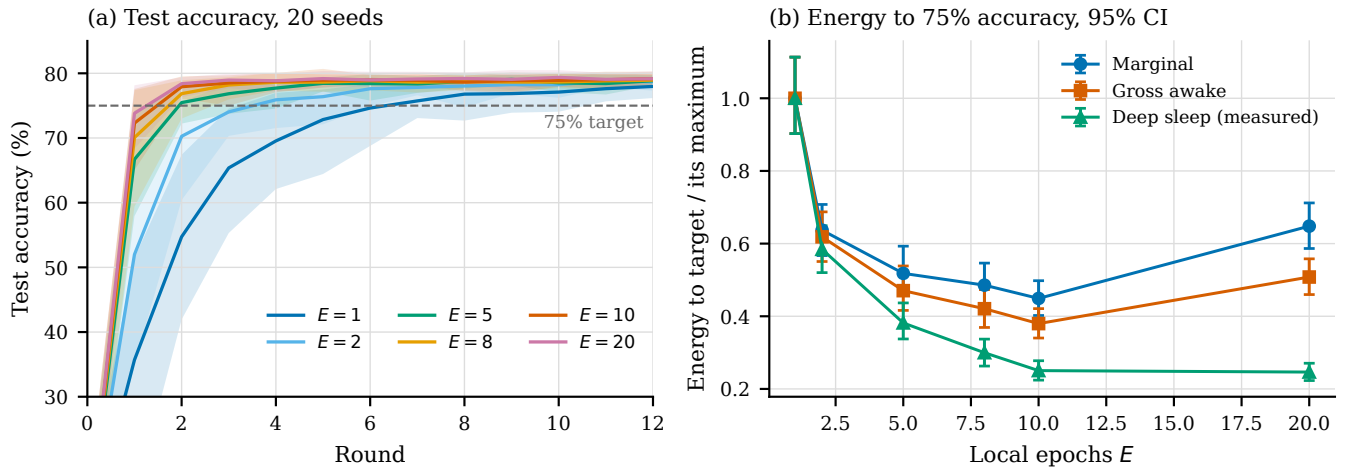


Fig. 11. Convergence in the $K = 20$ non-IID simulation. (a) Test accuracy per round, mean and 2.5 to 97.5 percentile range over 20 seeds. (b) Energy to 75% test accuracy for each accounting, divided by its own maximum, with 95% bootstrap intervals over seeds. The deep-sleep values use the measured sleep power and wake-up energy.

TABLE VII

ROBUSTNESS OF THE ENERGY-OPTIMAL E IN THE $K = 20$ NON-IID SIMULATION. (A) AT THE 75% TARGET: OPTIMUM OF THE MEAN ENERGY, NUMBER OF SEEDS WHOSE OWN OPTIMUM IS $E = 2/5/8/10/20$, AND ENERGY OF $E = 5$ AND $E = 20$ RELATIVE TO $E = 10$. (B) OPTIMUM FOR MARGINAL, GROSS AND DEEP-SLEEP ENERGY UNDER STRICTER TARGETS, RECOMPUTED FROM THE SAVED ACCURACY CURVES WITHOUT RETRAINING, AND UNDER OTHER DEEP-SLEEP PARAMETERS: THE ENDS OF THE 95% INTERVAL OF THE MEASURED E_{wake}, TWICE ITS MEAN, AND THE IDEALIZED VALUES THAT WE ASSUMED BEFORE THE SLEEP CAMPAIGN ($P_s = 0.08$ MW, $E_{\text{wake}} = 180$ MJ). $R < 1$ IS THE NUMBER OF SEEDS AT $E = 10$ AND $E = 20$ THAT REACH THE TARGET WITHIN THE FIRST ROUND.

(a) By accounting, 75% target Accounting	E_{opt}	Per-seed optima	E=5	E=20
Marginal	10	2/6/5/7/0	+15%	+44%
Gross	10	2/3/6/8/1	+24%	+34%
Deep sleep	20	0/1/4/8/7	+52%	−1.6%

(b) Sensitivity Variation	R<1	Marginal	Gross	Deep
Target 75%	6, 7	10	10	20
Target 76%	2, 6	10	10	20
Target 77%	2, 3	10	10	20
Target 78%	0, 1	10	10	20
$E_{\text{wake}} = 647$, 727 mJ	–	–	–	20
$E_{\text{wake}} = 1367$ mJ	–	–	–	20
Idealized sleep (assumed)	–	–	–	10

quantity; on an easier task, the same hardware would yield a different E_{opt}. The crossover alone therefore does not determine the best setting: the radio, the sleep schedule and the data all contribute.

D. Measurement-Calibrated Choice of E

Algorithm 2 turns the model into a procedure that a designer can apply to a new device or radio. Its first line requires only a short measurement campaign, such as that of Section V, and line 4 requires $R(E)$ from a simulation or a pilot FL run of the target task. The procedure returns both the crossover and the energy-optimal E, so that the two remain distinct.

Algorithm 2 Measurement-calibrated choice of local epochs

Require: Candidate set $\mathcal{E}$; sleep state (P_s, E_{wake}); round period T_r; target accuracy

1: Measure N turns per E for a few values of E with an external monitor and phase markers; fit α, c_0 by least squares; estimate P_{base}, t_{ep}, ΔE_{comm} and T_{comm}
2: Compute $E^\star(P_s)$ from Eq. (5) and $\rho = T_{\text{comm}}/t_{\text{ep}}$
3: **for** each $E \in \mathcal{E}$ **do**
4: Obtain $R(E)$ for the target accuracy
5: $J(E) \leftarrow R(E)\, E_{\text{turn}}(E, T_r)$ from Eq. (4)
6: **end for**
7: **return** $E^\star(P_s)$ and $E_{\text{opt}} = \arg\min_{E \in \mathcal{E}} J(E)$

IX. DESIGN GUIDELINES FOR IOT DEPLOYMENTS

1) **Do not assume that communication dominates on an MCU:** On this platform, the marginal energies are equal between approximately 7 and 10 epochs. Values of E from 5 to 10 lie below or near that point, so communication remains the larger marginal term over most of that range, but computation is not negligible.
2) **Account for the cost of being awake:** At $E = 1$, the awake baseline over a turn is approximately 74 mJ, whereas the marginal energy is approximately 11 mJ. Sleep scheduling can therefore matter more than tuning E.
3) **Treat the round period and the sleep state as design variables:** In deep sleep, a round is expensive because of the wake-up and reconnection, which cost 0.68 J on our board, more than the active energy of a turn; in standby, it is inexpensive. Modem sleep keeps the association but leaves the CPU awake, and saves only about half of the idle power. Select E with the sleep model of the deployment, using Eqs. (4) and (7) and Algorithm 2.
4) **Reduce the cost of reconnecting:** Re-association and address assignment take about 1.2 s after every deep-

sleep wake-up. Static addressing, fast reconnection with stored channel and access-point information, or a sleep mode that keeps the association could save more energy per round on our board than the entire computation of a turn, which costs about 20 mJ at $E = 20$, the largest value tested; we did not measure these techniques.

5) **Account for protocol overhead, not only payload:** The payload occupies at most a few milliseconds of airtime, yet an exchange takes tens of milliseconds. Compression and quantization, including quantization-aware training [23], reduce the bits per update; on a link such as ours, fewer connections or a persistent connection address the larger cost.
6) **Plan for stalls:** A few exchanges take hundreds of milliseconds or more, add energy and raise the mean, so time-out and retry policies matter for the energy budget.
7) **Expect contention to favor more local work:** If contention stretches each exchange by κ_c, the crossover grows approximately in proportion (Section VII).
8) **Measure with an independent instrument, hardware markers and a settling gap:** Analytical and datasheet models omit the awake baseline, the protocol overhead and the radio tail (Section IV), and training that starts immediately after a radio burst absorbs part of the tail (Section VI-C).

Other radios fit the same framework by replacing ΔE_{comm} and T_{comm}. A slow low-power wide-area link would push $E^\star$ much higher, and a short-range burst radio would pull it lower. This expectation is consistent with the mixed findings in the literature [28], [29], [31], but neither radio was measured here.

X. Scope and Future Directions

The study isolates the energy of one client turn under controlled conditions. This choice defines its scope and points to several extensions.

Platforms and workloads: We measured one board, radio, task and payload, with one device acting as all K clients, and the training uses synthetic embeddings, which represents a node that caches its embeddings (Section V-A). Recomputing the embeddings in every epoch or training a larger model would raise α and lower the crossover. Algorithm 2 and the released scripts allow the results to be recalibrated for other boards, radios, models and multi-device deployments.

Radio conditions: The communication energy varies between turns and between the two sessions, and we report the crossover as a range for this reason. Logging retries, channel load and signal strength, and varying the access point and the number of competing stations, would explain this variation and turn the contention model of Section VII into a measurement. A larger randomized tail test would also sharpen the tail estimate ($p \approx 0.04$ with five runs per cell), which the tail profiles of Fig. 7 already support independently.

Sleep and convergence: The sleep powers are board-level values of a development board with one access point, and the energy of FL with sleep combines these measured parameters through Eqs. (4) and (5). The convergence study is simulated for one task and one level of label skew, with larger learning rates than on the device; the energy per epoch does not depend on the learning rate. Measuring complete FL rounds with sleep on a purpose-built node, evaluating automatic light sleep and fast reconnection, and running the convergence study on devices are the natural next steps.

XI. Conclusion

The premise that radio transmission dominates the energy of a federated learning client did not hold on the Wi-Fi microcontroller that we measured, a class of device that powers many battery-operated IoT nodes. With an independent power monitor and hardware phase markers, we found that the energy of local computation reaches that of a model exchange after fewer than ten local epochs, and that the exact crossover differs between measurement sessions. The relevant communication cost is not the payload but the protocol exchange, the stalls and the radio tail, none of which appears in Shannon-capacity models, and the dominant term in the gross awake energy of a turn is the power drawn while the device is awake, which none of the common models contains. As a consequence, the number of local epochs cannot be selected from a communication model alone. When the device sleeps between rounds, the crossover is governed mainly by the ratio of exchange time to epoch time, so reducing the duration and the number of connections is at least as valuable as reducing the number of bits. Measured on the same board, deep sleep reduces the idle power to 17 mW, but each wake-up and reconnection costs 0.68 J, more than the computation and the model exchange of a turn together. In our convergence study, the energy-optimal number of epochs is $E = 10$ for marginal and gross energy, and with deep sleep $E = 10$ and $E = 20$ are practically tied; the cost of deviating from the optimum depends on the sleep regime, which makes power management a design variable of FL rather than an implementation detail. For battery-powered IoT deployments, the practical levers are therefore the sleep state, the reconnection procedure and the number of connections per round, together with the number of local epochs. The proposed procedure separates the hardware crossover from the learning optimum and requires only a short instrumented campaign on the target device. Section X outlines how these results can be extended to other platforms, radio conditions and complete FL deployments with sleep.

Appendix A
Instrumentation Details

Sensing: An INA238 with a 15 mΩ (1%) shunt is placed on the high side of the device supply. The ADC range is ±163.84 mV, and the device operates in continuous mode with bus-voltage and shunt-voltage conversion times of 280 µs each, a die-temperature conversion time of 50 µs and no averaging, so a complete conversion cycle (610 µs) fits within the 1 ms sampling interval. A second ESP32-S3 reads current and bus voltage at 1 kHz and transmits them over a USB serial link at 921,600 baud. Power is computed as $P = V_{\mathrm{bus}} I$, and energy is the sum over 1 ms samples between marker edges.

Markers: GPIO 10 marks training and GPIO 11 marks communication. A 3-bit phase bus on GPIO 12, 13 and 14

labels the sub-phases and is read by the logger. Each line has a $100\,\mathrm{k}\Omega$ pull-down resistor. Figure 12 shows the complete wiring.

Isolation: USB power is disconnected from the device during measurement, because a connected host would back-power the board and bypass the shunt. The device, the INA238 and the logger share a common ground.

Supply: The bench supply provides $5.0\,\mathrm{V}$ DC to the board input, which feeds the on-board $3.3\,\mathrm{V}$ regulator. The multimeter read $5.04\,\mathrm{V}$ before and after the campaign, and the logged bus voltage averaged $5.03\,\mathrm{V}$, with dips during radio and training bursts.

Quality gates and rejected runs: A run is accepted only if it has zero dropped samples, strictly increasing timestamps, the expected marker count and a baseline drift below 2%. The 105 accepted runs had a mean drift of 0.24% and a maximum drift of 0.84%. Twenty attempts were rejected and repeated, and their traces are retained. Nineteen of them showed no training marker, which the bench log attributes to a serial reset when the port was opened or to the phone hotspot entering sleep; these causes were not tested. One attempt had 99 dropped samples and an invalid first timestamp. The accepted traces were not edited.

Sleep campaign: The sleep campaign uses the same wiring and the $\pm 40.96\,\mathrm{mV}$ ADC range ($1.25\,\mu\mathrm{V}$, i.e., $83\,\mu\mathrm{A}$ per least significant bit, full scale $2.73\,\mathrm{A}$) with otherwise identical settings. At boot, the device raises each of the five marker lines alone for $300\,\mathrm{ms}$, and the logger confirmed all five. The run comprised 20 cycles in one continuous recording of 2.79 million samples; it had no dropped samples, no gap longer than $2\,\mathrm{ms}$, no failed reconnection, and the expected number of delimiters for every state. Two shorter pilot runs, used to verify the firmware, are retained but not analyzed. The Wi-Fi power-save mode was set through the ESP-IDF call `esp_wifi_set_ps` and read back in every cycle, because in Arduino-ESP32 3.3.6 the function `WiFi.setSleep` applies the previously requested mode; the FL campaigns are not affected, because the driver applies the requested mode (power save off) when the connection is established.

Integration near the baseline: Energy is the sum of $(P - P_{\mathrm{base}}) \cdot 1\,\mathrm{ms}$ over a window, with negative terms retained. Clamping negative terms to zero would rectify baseline noise and introduce a positive bias that grows with the window length; on idle samples held out from the baseline estimate, this bias is $2.8\,\mathrm{mW}$ on average. It affects the communication windows, which are close to the baseline and long when the exchange stalls, but hardly the training windows, whose power lies about $130\,\mathrm{mW}$ above the baseline. With clamping, the mean communication energy of the main campaign would be 10.16 instead of $9.75\,\mathrm{mJ}$, and the crossover 10.15 instead of 9.74 epochs. An independent implementation of Algorithm 1 accepts all 105 runs, reproduces their baselines to within $0.005\,\mathrm{mW}$ and their training energies to within $0.005\,\mathrm{mJ}$, and reproduces the communication energies of 102 runs to within $0.002\,\mathrm{mJ}$. The reported communication energies take the window edges from the phase bus; in the other three runs, these differ from the GPIO edges by one or two samples, which changes the main-campaign mean by $0.014\,\mathrm{mJ}$ and $E^\star$ by 0.01 epochs.

Appendix B
Status of Results

- *Measured:* P_{base}, ΔE_{comp} (head training on synthetic embeddings), ΔE_{comm}, the phase times and energies, the tail energy and the tail profile, from 105 runs under one protocol; the board-level modem-, light- and deep-sleep powers, the wake-up and re-association energies and times, and the sleep-entry energies, from 20 cycles of a separate sleep campaign.
- *Derived from measurements:* α, c_0, $E^\star$, the intervals, the crossover and energy per turn for each sleep state (Table V), and the marginal, gross and deep-sleep energies in Table VI.
- *Assumed:* κ_c, the round period $T_r = 60\,\mathrm{s}$, and the parameters of the analytical and datasheet models. The contention results are model-only.
- *Simulated:* the rounds $R(E)$ in Table VI and the robustness analysis of Section VIII.

Data and Code Availability

The 105 raw power traces, the raw trace of the sleep campaign, the run summaries, the analysis and figure scripts and the firmware of both boards are available at https://github.com/shakursiam/fl-energy-crossover.

Acknowledgment

The authors used Claude (Anthropic) to assist in checking the data analysis and plotting scripts and in editing the language and layout of this article. The authors designed and performed the experiments, reviewed every script and every edit, verified all reported numbers against the raw measurements, and take full responsibility for the content of this article.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. PMLR, vol. 54, 2017, pp. 1273–1282.

[2] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[4] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[5] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[6] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 1387–1395.

[7] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2021.

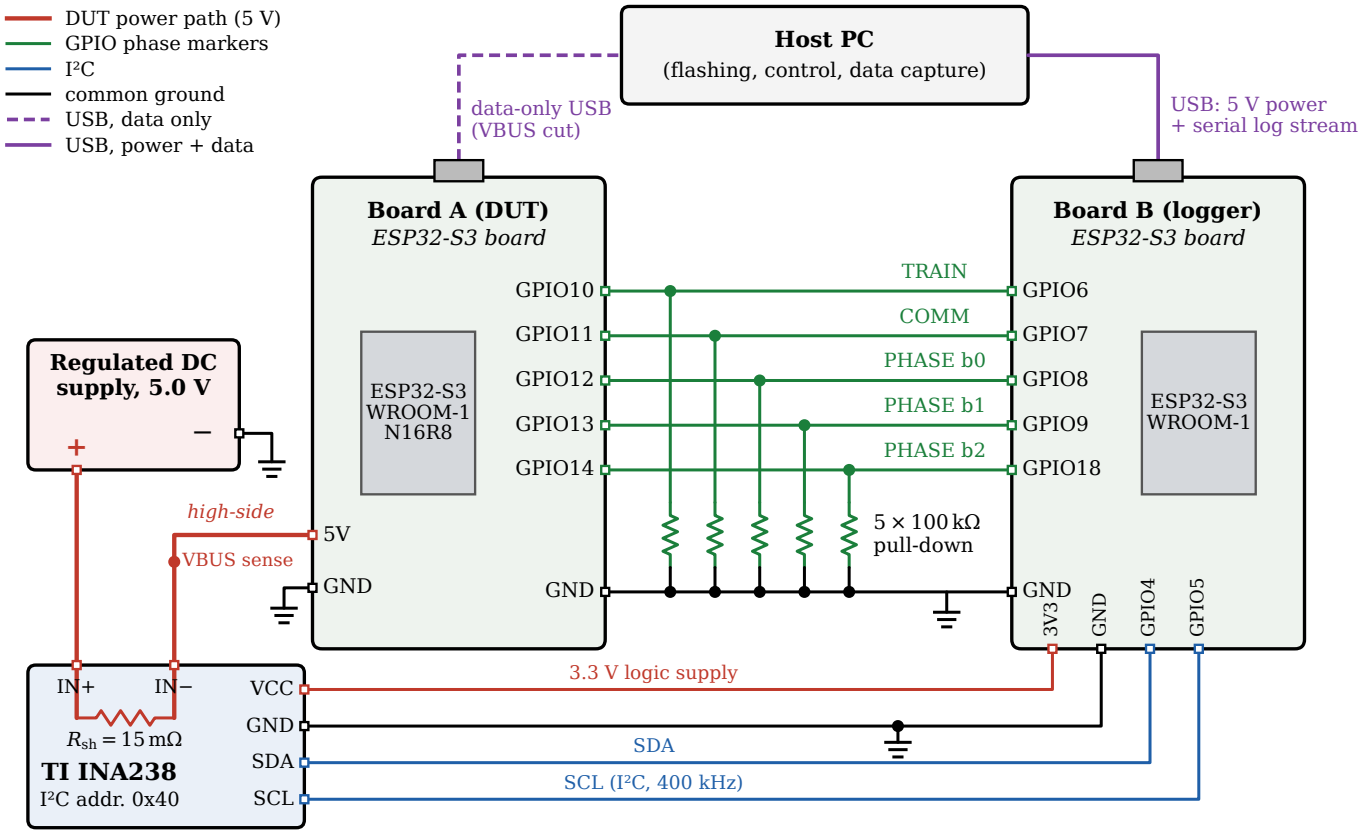


Fig. 12. Wiring of the testbed. The device under test (Board A) is powered at 5.0 V through the 15 mΩ high-side shunt of a TI INA238, which Board B reads over I²C at 400 kHz. Five GPIO lines, each tied to ground through a 100 kΩ pull-down, carry the TRAIN, COMM and 3-bit phase markers from Board A to Board B. The USB cable of Board A carries data only, so that all of its supply current passes through the shunt.

[8] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[9] X. Zhou, J. Zhao, H. Han, and C. Guet, "Joint optimization of energy consumption and completion time in federated learning," in *Proceedings of the IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 1005–1017.

[10] H. Chen, S. Huang, D. Zhang, M. Xiao, M. Skoglund, and H. V. Poor, "Federated learning over wireless IoT networks with optimized communication and resources," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 592–16 605, 2022.

[11] T. Zhao, X. Chen, Q. Sun, and J. Zhang, "Energy-efficient federated learning over cell-free IoT networks: Modeling and optimization," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 17 436–17 449, 2023.

[12] Q.-V. Pham, M. Le, T. Huynh-The, Z. Han, and W.-J. Hwang, "Energy-efficient federated learning over UAV-enabled wireless powered communications," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4977–4990, 2022.

[13] J. Ren, W. Ni, and H. Tian, "Toward communication-learning trade-off for federated learning at the network edge," *IEEE Communications Letters*, vol. 26, no. 8, pp. 1858–1862, 2022.

[14] X. Fan, Y. Chen, M. Liu, S. Sun, Z. Liu, K. Xu, and Z. Li, "UAV-enabled federated learning in dynamic environments: Efficiency and security trade-off," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 5, pp. 6993–7006, 2024.

[15] Z. Ju, F. Shen, and T. Wei, "BEFL: Balancing energy consumption in federated learning for mobile edge IoT," *arXiv preprint arXiv:2412.03950*, 2024.

[16] M. Hamood, A. Albaseer, H. El-Sallabi, M. Abdallah, A. Al-Fuqaha, and B. Hamdaoui, "Personalized federated learning with adaptive transformer pruning and hypernetwork-driven personalization in wireless networks," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 4, pp. 1–16, 2026.

[17] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2020.

[18] P. Prakash, J. Ding, R. Chen, X. Qin, M. Shu, Q. Cui, Y. Guo, and M. Pan, "IoT device friendly and communication-efficient federated learning via joint model pruning and quantization," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 638–13 650, 2022.

[19] Z. Pan, Y. Li, Z. Guan, M. Liang, A. Li, J. Wang, and F. Kou, "RFCSC: Communication efficient reinforcement federated learning with dynamic client selection and adaptive gradient compression," *Neurocomputing*, vol. 612, p. 128672, 2025.

[20] J. Xiao, Y. Chen, C. Yang, K. Li, K. Xie, X. Li, K. Li, and W. Liang, "Communication-efficient federated learning based on a two-stage adaptive basis matrix for gradient compression," *IEEE Transactions on Mobile Computing*, vol. 25, no. 10, pp. 18 403–18 420, 2026.

[21] Y. Wang, L. Lin, and J. Chen, "Communication-efficient adaptive federated learning," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, ser. PMLR, vol. 162, 2022, pp. 6292–6320.

[22] G. Paragliola, "Evaluation of the trade-off between performance and communication costs in federated learning scenario," *Future Generation Computer Systems*, vol. 136, pp. 282–293, 2022.

[23] X. Fang, L. Chen, H. Yin, X. Chen, and W. Wang, "Communication efficient federated learning with quantization-aware training design," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 4, pp. 45–59, 2026.

[24] N. Tekin, A. Acar, A. Aris, A. S. Uluagac, and V. C. Gungor, "Energy consumption of on-device machine learning models for IoT intrusion detection," *Internet of Things*, vol. 21, p. 100670, 2023.

[25] A. Sabovic, J. Fontaine, E. De Poorter, and J. Famaey, "Energy-aware TinyML model selection on zero energy devices," *Internet of Things*, vol. 30, p. 101488, 2025.

[26] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on

federated learning for resource-constrained IoT devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2022.
[27] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256KB memory," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 22 941–22 954, 2022.
[28] S. Rupaak, G. Khekare, Y. Kumar, and G. Soni, "Quantifying energy and accuracy trade-offs of federated learning on wearable health devices," *Frontiers in Big Data*, vol. 9, p. 1769948, 2026.
[29] N. Llisterri Giménez, J. Miquel Solé, and F. Freitag, "Embedded federated learning over a LoRa mesh network," *Pervasive and Mobile Computing*, vol. 93, p. 101819, 2023.
[30] O. Torres Sanchez, G. Borges, D. Raposo, A. Rodrigues, F. Boavida, and J. Sá Silva, "Enhancing the performance of industrial IoT LoRaWAN-enabled federated learning frameworks: A case study," *Internet of Things*, vol. 32, p. 101632, 2025.
[31] O. Torres Sanchez, A. Rodrigues, G. Borges, D. Raposo, F. Boavida, and J. Sá Silva, "A framework for tiny federated learning in resource-constrained IIoT environments: A case study," *IEEE Internet of Things Journal*, vol. 13, no. 9, pp. 19 076–19 089, 2026.
[32] A. Singh and S. S. Borkotoky, "Federated learning over LoRa networks: Simulator design and performance evaluation," *arXiv preprint arXiv:2508.10574*, 2025.
[33] H. Ren, D. Anicic, and T. A. Runkler, "TinyOL: TinyML with online-learning on microcontrollers," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.
[34] K. Kopparapu and E. Lin, "TinyFedTL: Federated transfer learning on tiny devices," *arXiv preprint arXiv:2110.01107*, 2021.
[35] N. Llisterri Giménez, M. Monfort Grau, R. Pueyo Centelles, and F. Freitag, "On-device training of machine learning models on microcontrollers with federated learning," *Electronics*, vol. 11, no. 4, p. 573, 2022.
[36] M. Ficco, A. Guerriero, E. Milite, F. Palmieri, R. Pietrantuono, and S. Russo, "Federated learning for IoT devices: Enhancing TinyML with on-board training," *Information Fusion*, vol. 104, p. 102189, 2024.
[37] K. Ibrar, P. Fusco, G. P. Rimoli, F. Palmieri, and M. Ficco, "Cross-modal federated TinyML for MCU-based Internet of Medical Things," *IEEE Journal of Biomedical and Health Informatics*, 2026, early access.
[38] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello Edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2018.
[39] C. Banbury, V. J. Reddi, P. Torelli, N. Jeffries, C. Kiraly, J. Holleman *et al.*, "MLPerf Tiny benchmark," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.
[40] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference (IMC)*, 2009, pp. 280–293.
[41] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. B. Gusmao, Y. Gao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning," *Journal of Machine Learning Research*, vol. 24, pp. 1–23, 2023.
[42] V. Barral Vales, O. Campos Fernández, T. Domínguez-Bolaño, C. J. Escudero, and J. A. García-Naya, "Fine time measurement for the Internet of Things: A practical approach using ESP32," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 305–18 318, 2022.
[43] *ESP32-S3 Series Datasheet*, Espressif Systems, 2026, version 2.2. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf
[44] P. Warden, "Speech Commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.