No Cloud, No Problem: A Real-Time HAR Insole with On-Device Inference

Ruben Schlonsak[1,3(✉)], Jiabao Yu[1,2], Hans-Christian Jetter[4], and Denys J. C. Matthies[1,3]

[1] Technical University of Applied Sciences Lübeck, Lübeck, Germany
ruben.schlonsak@th-luebeck.de
[2] East China University of Science and Technology, Shanghai, China
[3] Fraunhofer IMTE, Lübeck, Germany
[4] University of Lübeck, Lübeck, Germany

Abstract. We present an ankle-mounted insole artifact enabling real-time recognition of seven daily activities. Classification is performed on-chip on a dual-core ESP32-S3 microcontroller without requiring any cloud or smartphone connection. Our proposed program design allows for multi-threading, treating sensing (SensorTask), inference (ModelTask), and wireless output (BLETask) as three periodic tasks within FreeRTOS. The SensorTask acquires six pressure channels and a six-axis IMU sampling at 20 Hz. The ModelTask dequeues the sample and executes a post-training-quantized one-dimensional convolutional network in 54 ms. The BLETask transmits the predicted class on change, keeping the radio duty cycle below 7%. Across eight subjects, the system reaches 92.8% leave-one-subject-out accuracy with a worst-case end-to-end latency of up to 555 ms. The mean current increases by 2.24 mA above a 100 mA sensor baseline, allowing for nearly 5 h of operation on a 500 mAh cell. Stability is underpinned by a zero-loss of sensor frames during two hours of continuous streaming with a minimal task jitter below 2 ms. To our knowledge, this is the first insole that reports a complete timing–energy–accuracy triad for entirely local inference on an ESP32-class microcontroller. The results demonstrate that careful task scheduling, rather than network compression alone, is sufficient for achieving reliable edge intelligence in resource-constrained wearables.

Keywords: Human Activity Recognition · Edge AI · Smart Insole · Real-Time Inference · FreeRTOS · TinyML

1 Introduction

In human activity recognition (HAR), foot interfaces, such as insoles, have migrated from research labs to commercial gait analysis and rehabilitation products [21]. Most systems still offload sensor streams to a smartphone or cloud service, incurring 200 ms to 400 ms network latency, high radio power, and

Ö. Durmaz Incel et al. (Eds.): iWOAR 2025, LNCS 16292, pp. 324–342, 2026.
https://doi.org/10.1007/978-3-032-13312-0_19

possible privacy leakage [18,31]. Recent studies have demonstrated that convolutional and recurrent networks can be efficiently compressed using post-training quantization and executed in real-time on microcontrollers, such as the ESP32 [2,11,29]. Nevertheless, firmware design remains an under-examined bottleneck. Inference is often placed inside a single polling loop. When Bluetooth advertising or Wi-Fi interrupts occur, periodic sensing deadlines slip, queues overflow, and the reported accuracy cannot be reproduced in the field. Very few published wearables quantify task jitter or verify that their schedulers remain deterministic when radio traffic is present (Fig. 1).

Fig. 1. Our sensor prototype: The ESP32 S3 module with the 6-axis IMU and the battery placed in a 3D-printed case that is strapped around the ankle, while six capacitive sensors are embedded in the connected insole, which was part of former work [29].

A real-time operating system (RTOS) provides a principled remedy. FreeRTOS version 11.1 extends symmetric multiprocessing and memory-protection features to commodity microcontrollers [15], yet prior insole projects that report timing figures usually rely on more powerful system-on-chip or external co-processors [8,14]. To the best of our knowledge, no earlier work combines timing, energy, and accuracy for a FreeRTOS-based insole that performs all inference on an ESP32-class device [24]. Symmetric multiprocessing allows one kernel instance to balance work across both cores; however, published examples discuss how tasks are pinned or how per-core utilization is measured. Earlier studies also omit worst-case execution time budgets and do not show that queue depths are sufficient to prevent frame loss during continuous walking [16]. These gaps obscure reproducibility and cloud reliability.

This paper addresses that gap by treating edge inference as just another periodic task in a real-time schedule:

1. Deterministic FreeRTOS schedule. We design a three-task pipeline that includes sensor acquisition, 1-D CNN inference, and Bluetooth Low Energy (BLE) reporting, all of which are pinned to the two cores of an ESP32-S3. Worst-case execution times (WCET) and semaphore-protected queues guarantee no sample loss at 20 Hz.
2. Compact edge model. A post-training-quantized 1-D CNN (304 kB flash, 86 kB RAM) classifies seven daily activities in 54 ms on the device while maintaining 92.8% leave-one-subject-out accuracy.
3. Comprehensive evaluation. We publish end-to-end latency, per-task jitter distributions, energy overhead (+2.24 mA on a 500 mAh Li-Po), and memory footprint, offering the first timing-energy-accuracy triad for a FreeRTOS HAR insole on the ESP32-S3.

By demonstrating that strict task-level scheduling can deliver deterministic sensing alongside neural inference without compromising battery life or accuracy, this work advances the design of privacy-preserving, low-power wearables that rely solely on local intelligence.

2 Related Work

2.1 Smart Insoles for Activity Recognition

Wearable sensor-equipped insoles have been developed that collect foot sensor data and usually transmit it for expensive data processing to external devices with higher computational power, such as to a smartphone [28], smart glass [5], tablet computer [17], PC [7] etc. Early smart-insole systems focused primarily on sensing and wireless streaming, leaving feature extraction and classification to off-body devices. These systems typically stream plantar pressure or motion data via Bluetooth or similar wireless links to a smartphone or cloud, where algorithms analyze gait metrics or detect specific events. Early examples include CapSoles [17], a battery-powered insole prototype that submits 6 CapSense values to a computer, where the data is processed with machine learning, such as

to identify the wearer and six different terrains. Another recent example is presented by Wang et al.; a self-powered insole equipped with 22 pressure sensors and solar energy harvesting, which sends real-time gait data via Bluetooth to a smartphone for analysis [28]. Their system uses a machine-learning model on the phone to recognize eight motion states (sitting, standing, running, etc.) from the transmitted pressure patterns. This approach improves user posture and provides early warnings for foot or neurological conditions by offloading the computational inference to the phone. Martini et al. developed a wireless insole with 16 optoelectronic force sensors, focusing on real-time gait event detection [16]. The insole's onboard microcontroller samples the sensors at a rate of 100 Hz. It transmits the raw data via an ultra-wideband link to a remote processor (National Instruments SOM), which calculates gait phase variables and center of pressure in real-time.

Typical HAR capabilities with insole- or shoe-placed sensors include body posture, ambulation activity, gait abnormalities, and terrain [12]. Prior research projects have shown rich detection capabilities with smart footwear prototypes, while the data is usually wirelessly streamed to an external device that has richer computational power. This laid the groundwork for our approach, which aims to integrate the entire HAR pipeline directly into the insole prototype itself.

2.2 Edge-AI Inference on MCUs

Deep learning algorithms have proven high performance in HAR, while light models are now fit to work "on the edge" [1]. This trend in "TinyML" has enabled running neural network inference directly on resource-constrained microcontrollers, such as the ARM Cortex-M series (e.g., STM32, ESP32, nRF52). Key techniques include model compression (quantization, pruning), efficient architecture design, and optimized inference runtimes that contend with tight memory (tens of kB RAM) and limited CPU speed [13,25]. Likewise, optimized models for sensor data have been demonstrated to run in real-time on microcontroller-class wearables. Shakerian et al. present a chest-mounted IMU sensor with an integrated CNN that performs HAR inference locally on a low-cost MCU, removing the need for any external phone or computer during operation [25]. The 1D convolutional network in their device classifies activities from accelerometer signals on board with high accuracy and low latency. The idea of using CNN on accelerometer data is not new, but has already been proposed at iWOAR 2018 [19]. Meanwhile, researchers have successfully deployed similar architectures, such as recurrent neural networks on microcontrollers. Di Leo et al. implemented an LSTM-based gesture recognition on an STM32L4 MCU, achieving >90% accuracy while maintaining real-time performance [3]. These examples illustrate the growing capability for on-device HAR in wearables, thanks to TinyML toolkits (e.g. TensorFlow Lite Micro) and efficient model design. Running inference at the edge brings well-known benefits: it avoids wireless latency, reduces battery drain from constant radio use, and preserves user privacy by keeping personal sensor data local [31]. Very recent work is even pushing toward on-device

learning – for instance, Zuo et al. propose an edge AI platform that can incrementally train or adapt an activity model on a device (a smartphone in their case) without cloud interaction [31]. Shalby et al. likewise introduce a TinyML method for on-device learning of new human activities on a resource-constrained MCU (STM32), highlighting the trend toward personalization on the edge [26]. Recently, Peretti et al. [22] investigated energy efficiency of feature selection for HAR running on MCU. The study effectively shows that with around two dozen thoughtfully selected features, energy consumption during feature extraction on a typical low-power smartwatch platform can be minimized without significantly sacrificing HAR accuracy. This makes a strong case for adopting energy-aware feature selection in resource-constrained wearable ML applications. All literature in its entirety show that contemporary wearable devices can perform non-trivial HAR inference locally on microcontroller hardware, provided the features and models are carefully optimized. Our work builds on this foundation by not only deploying a quantized CNN in a foot-worn MCU but also by architecting the firmware to meet real-time scheduling demands in a multi-task wearable system.

2.3 RTOS Scheduling in Wearables

Resource-constrained wearables often rely on a real-time operating system (RTOS), such as FreeRTOS or Zephyr, to schedule concurrent tasks (including sensor sampling, wireless communication, and inference) under timing and energy constraints [23]. An RTOS provides preemptive multitasking, priority-based scheduling, and inter-task synchronization, which are crucial for reliable performance in wearables, medical devices, and other latency-sensitive embedded systems [10]. FreeRTOS, a widely used open-source RTOS, facilitates the allocation of high priority to critical tasks, such as sensor reading, ensuring their execution on schedule. Concurrently, lower-priority tasks, such as data transmission over Bluetooth, are subject to delay or preemption as needed [30]. This design prevents slow peripherals or network interrupts from blocking the main sensor loop. Zhang et al. illustrate this in a multi-sensor wearable for Parkinson's disease monitoring, where they adopted FreeRTOS to ensure that data acquisition at 100 Hz would not be disrupted by other processes, such as writing to an SD card or handling Wi-Fi communication [30]. As a result, the device maintains stable sampling and avoids missing data frames, even when network traffic is present.

Another example by Luna-Perejón et al. is an ankle-worn device for recording a fall detection dataset. They report using FreeRTOS in the firmware to "correctly process the information and transmit it without losing data" at 50 Hz from a 3-axis accelerometer [14]. Recent surveys of wearable systems emphasize the need for rigorous scheduling to ensure reproducible performance in real-world deployments [10]. While an RTOS greatly enhances reliability, it also adds overhead and complexity. To address this, some work has explored lightweight scheduling alternatives. Kos et al. (2025) argued that complete RTOS solutions (FreeRTOS, ChibiOS, NuttX, etc.), with their rich features (preemptive multitasking, IPC, etc.), can introduce unnecessary overhead in simple wearable

biofeedback applications [10]. To address this issue, they developed a custom cooperative scheduler that executes tasks in a fixed cyclic order within a timer interrupt, eliminating the need for context switching except at predetermined points. Their minimalist scheduler achieved a context-switch overhead of only 3 µs, compared to 5–15 µs for FreeRTOS on the same Cortex-M microcontroller [10].

3 System Design

This section details the design and implementation of the edge AI insole, covering the hardware platform, model architecture, firmware design, and the end-to-end deployment pipeline. The system is designed to meet the stringent requirements of low power and low latency operation on a resource-constrained microcontroller.

(a) (b) (c)

Fig. 2. Hardware prototype of the smart insole system. (a) Wearable unit showing the external enclosure and attachment to the insole. (b) Internal view of the electronics, including microcontroller, IMU, battery, and connectors. (c) Internal view of the insole with embedded capacitive pressure sensors and wiring.

3.1 Hardware Platform

The system is built upon the Waveshare ESP32-S3-LCD-1.28, a compact development board featuring a dual-core Xtensa® LX7 processor operating at 240 MHz. The board is equipped with 2 MB of PSRAM, 16 MB of flash memory, and integrated 2.4 GHz Wi-Fi and Bluetooth 5 (LE) connectivity. For sensor data acquisition, the platform features an onboard QMI8658 6-axis Inertial Measurement Unit (IMU), which provides data from a 3-axis accelerometer and a 3-axis gyroscope. This is augmented by six external capacitive sensors connected to GPIO pins strategically placed to capture pressure distribution across the insole. The developed prototype is shown in Fig. 2.

3.2 Data Acquisition and Pre-processing

A dataset was collected from eight adult participants performing seven everyday activities: running, walking, climbing stairs, descending stairs, sitting, standing, and lying (weight: 60–100 kg, height: 170–190 cm). The 12-dimensional sensor data (6-axis IMU and six pressure channels). We sample at 20 Hz because plantar–pressure and foot-IMU spectra contain negligible energy above 10 Hz; Elstub et al. report <3% peak-force error when decimating to 20 Hz [4]. For segmentation size, we used a 0.5 s window (10 frames) that spans a full stance or swing sub-phase of the 1 s gait cycle, providing sufficient context while capping worst-case latency below 0.6 s; Nazari et al. found 0.5 s to be the optimum for CNN-based gait HAR (99.95% F1) [20]. To mitigate the risk of overfitting [27], a leave-one-subject-out split was employed during the training process. To prepare the data for the neural network, a sliding window technique was employed. A window size of 0.5 s (10 consecutive samples) was chosen to ensure low system latency. The window moves with a stride of one sample, creating overlapping frames that enhance the temporal resolution of the input data, resulting in more robust classification.

3.3 Model Architecture and Training

Model Selection. Considering the resource constraints of the target MCU, a one-dimensional convolutional neural network (1D-CNN) was selected. This architecture offers a strong balance between classification performance and computational efficiency for time-series data, making it more suitable for embedded deployment than more complex models such as LSTMs or hybrid networks, which typically have higher memory and computational demands. The 1D-CNN architecture was implemented using the Keras API in TensorFlow and consists of the architecture described in Fig. 3. The model was trained on a desktop PC using the Adam optimizer and the sparse categorical cross-entropy loss function.

Model Optimization and Deployment. To deploy the model on the ESP32-S3, an end-to-end pipeline was established:

- **Quantization:** The trained Keras model was converted to the TensorFlow Lite (TFLite) format. Post-training quantization was applied, converting the model's 32-bit floating-point weights to 8-bit integers. This reduced the model's storage footprint by approximately 65% while the input and output layers were kept as float32 to maintain precision.
- **Conversion to C Array:** The quantized TFLite model file was converted into a C header file using the xxd tool. This transforms the model into a static unsigned char array that can be directly compiled into the firmware.
- **Firmware Integration:** The model was integrated into the firmware using the TensorFlow Lite for Microcontrollers (TFLM) library. A MicroInterpreter was instantiated, and a tensor arena, a statically allocated memory region for

model operations, was defined. To minimize binary size, only the TFLM operations required by the model (e.g., Conv1D, Dense, Softmax) were registered using a MicroMutableOpResolver.

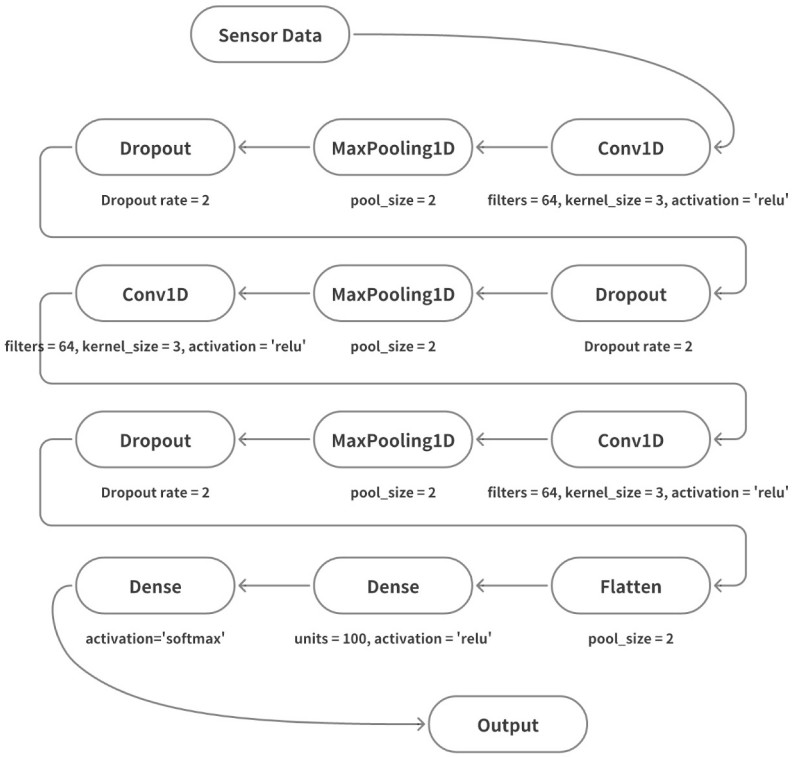


Fig. 3. One-dimensional convolutional neural network (1D-CNN) architecture for human-activity recognition from smart-insole sensor data. The model ingests a frame of 10 time steps × 12 sensor channels, processes it through three identical convolutional blocks each consisting of a 64-filter Conv1D layer (kernel = 3, ReLU), a MaxPooling1D layer (pool = 2), and dropout (rate = 0.2), and then flattens the feature maps. A fully connected layer with 100 ReLU units precedes a seven-unit softmax layer that outputs the class-probability distribution.

3.4 Real-Time Task Scheduling with FreeRTOS

To manage concurrent operations and prevent task interference, a multi-tasking architecture was implemented using FreeRTOS, leveraging the ESP32-S3's dual-core capabilities. The three-task scheduler is summarized in Algorithm 1.

Algorithm 1 Minimal RTOS Task Layout

```
    Shared: dataQueue, modelReady, modelResult
 1: procedure SensorTask(every 50 ms)
 2:     frame ← ReadSensors()                          ▷ ADC+IMU+touch
 3:     Enqueue(dataQueue, frame)
 4: end procedure
 5: procedure ModelTask
 6:     while true do
 7:         window ← DequeueN(dataQueue, W)
 8:         modelResult ← Infer(window)
 9:         Give(modelReady)
10:     end while
11: end procedure
12: procedure BleTask
13:     while true do
14:         Take(modelReady)
15:         if connected then
16:             NotifyBLE(modelResult)
17:         end if
18:     end while
19: end procedure
```

Task Design. The following three primary tasks were implemented: 1) the acquisition of sensor data, 2) the utilization of the developed model for predictions, and 3) the transmission of the latest classification results via Bluetooth Low Energy, as described in Table 1.

Table 1. FreeRTOS task breakdown.

Task	Core	Priority	Avg. rate	Function
`sensorTask`	0	2	20 Hz	Acquire IMU & pressure data
`modelTask`	1	1	$\approx$ 20 Hz	Assemble 0.5 s windows & invoke CNN
`bleTask`	1	1	on demand	Advertise latest prediction

A FreeRTOS queue is used to safely pass data from the high-frequency sensorTask to the lower-frequency modelTask, preventing data loss and ensuring FIFO processing. A binary semaphore synchronizes the modelTask and bleTask, ensuring that BLE transmissions occur only when a new prediction is available, thereby preventing redundant transmissions and conserving energy. This partitioned, priority-based scheduling strategy isolates time-critical sensing from computationally intensive inference and non-deterministic BLE communication, ensuring system stability and real-time performance (Fig. 4).

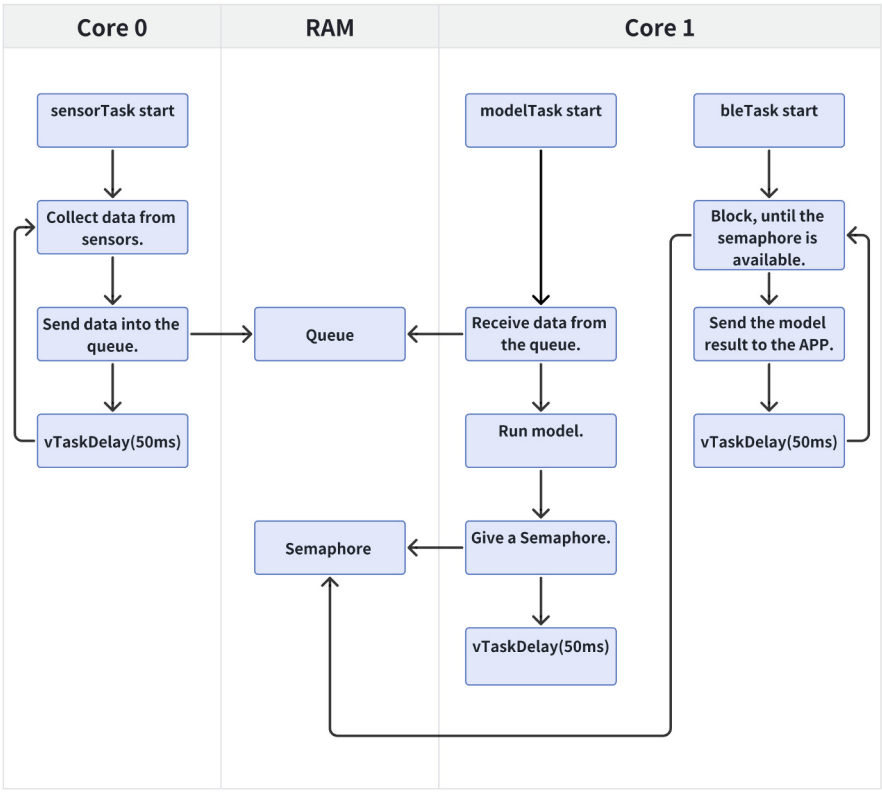


Fig. 4. Task-level schedule on the dual-core ESP32-S3.

4 Evaluation

The system was evaluated using a structured framework that assessed classification performance, task scheduling behavior, and resource consumption (including latency, energy, and memory) on both a PC and the target embedded device.

Person-Independent Accuracy (Leave-One-Subject-Out Cross-Validation)

To assess generalization, a Leave-One-Subject-Out Cross-Validation (LOSOCV) was performed. The model achieved an average accuracy of 92.8%. The corresponding confusion matrix is presented in Fig. 5 While performance was strong for most activities, some confusion was observed between 'sitting' and 'standing'. This is attributable to the similarity in foot pressure and orientation in these static states, making it difficult to distinguish them without more diverse training data that covers a range of body weights and postures.

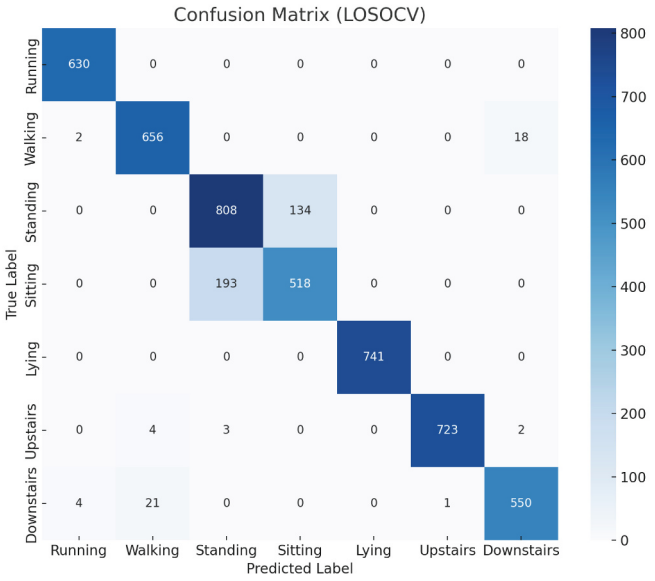


Fig. 5. Normalised confusion matrix for leave-one-subject-out cross-validation (LOSOCV).

On-Device Performance

After deploying the quantized model to the ESP32-S3, the model's real-world performance was tested with new participants not included in the training set. The system achieved an average accuracy of 92.76%, demonstrating that the post-training quantization and on-device execution had a negligible impact on classification performance. The misclassification patterns (e.g., 'sitting' vs. 'standing') were consistent with the PC-based evaluation.

4.1 System Performance Metrics

The FreeRTOS task scheduling was verified by monitoring serial output logs. The logs confirmed that the sensorTask executed consistently at 20 Hz without being disrupted by other tasks. The modelTask was triggered correctly after 10 sensor readings, and the bleTask was only activated after the modelTask completed inference, validating the effectiveness of the queue and semaphore synchronization mechanisms.

Inference Latency. The time taken for a single inference cycle on the ESP32-S3 was measured by logging timestamps before and after the *interpreter->Invoke()* call.

- Inference Latency: The average time to run the 1D-CNN model was 54 ms.
- End-to-End Latency: The total time from the start of data collection for a window to the final classification is approximately 554 ms (500 ms for data acquisition + 55 ms for inference). This is well within the acceptable range for real-time HAR applications.

As shown in Fig. 6, our Edge-AI smart-insole solution achieves an end-to-end inference latency of 55 ms, outperforming typical cloud-based inference (up to 300 ms) [6]. Due to higher computation power, modern smartphone-based inference can surpass this latency.

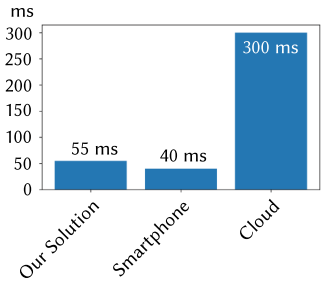
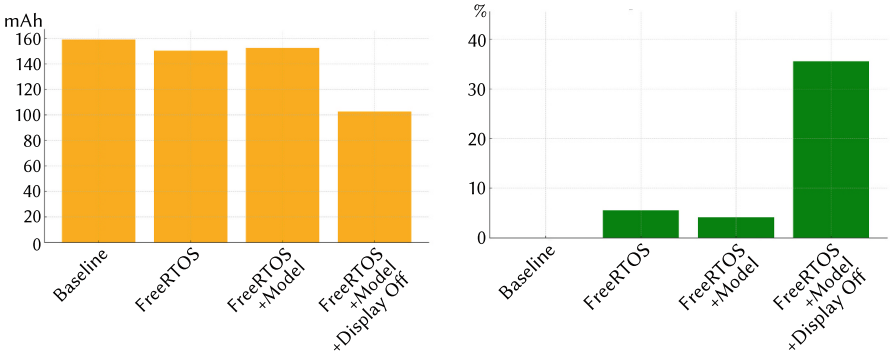


Fig. 6. End-to-end inference latency for our Edge-AI smart-insole solution (55ms), typical smartphone inference (up to 40ms), and cloud server inference (up to 300ms)

Energy Consumption. Energy consumption was measured using a Nordic Power Profiler Kit II (PPK2). The power consumption measurements are as follows: Baseline (Sensors + BLE, no scheduling) at 159.06 mA; with FreeRTOS Scheduling (no inference) at 150.24 mA, reducing consumption by preventing redundant BLE transmissions; and Full System (Scheduling + Inference) at 152.48 mA. By disconnecting the display from the microcontroller, we can save an additional 50 mA of power, resulting in a total of 102.48 mA. The results are presented in Figs. 9 and 10.

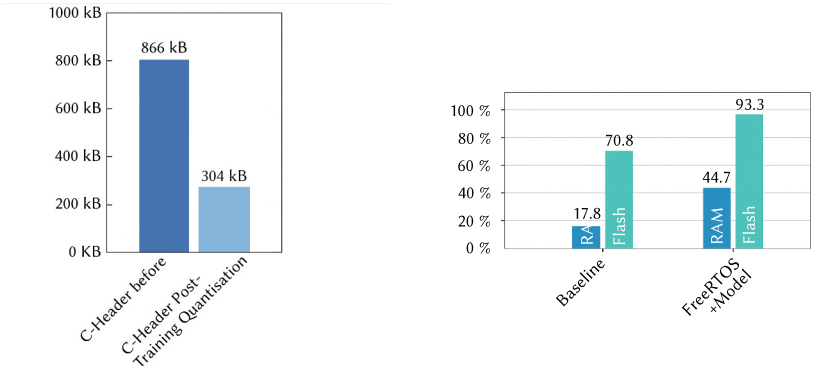
As illustrated in Fig. 7, disabling the display yields the most energy savings, reducing the average current consumption by 35.6% compared to the baseline. In contrast, integrating FreeRTOS task scheduling and ML inference introduces only minor variations of less than 6%.

Memory Usage. The impact of quantization and deployment on memory was analyzed using PlatformIO. Post-training quantization reduced the model's C header file size from 866 KB to 304 KB (a 64.9% reduction). Deploying the model and TFLM engine increased the firmware's total resource usage. Static RAM usage increased by 86.36 KB, and Flash usage increased by 1213.77 KB compared to the baseline firmware (only sensing). The final firmware (sensing + model) occupied 93.3% of the available Flash and 44.7% of the RAM, confirming that the system fits within the microcontroller's constraints (Fig. 8).



(a) Average current per configuration. (b) Percentage reduction vs. baseline.

Fig. 7. Comparison of absolute and relative energy impact for each firmware version.



(a) Impact of model quantization. (b) Comparison of memory usage

Fig. 8. Comparison of absolute and relative energy impact for each firmware version.

5 Discussion and Limitations

The three-task FreeRTOS schedule delivered the deterministic behavior that real-time sensing demands. Across two hours of continuous operation, the insole did not miss a single twenty-hertz sample, and the measured sensor-loop jitter stayed within two milliseconds. Fixing each task to a specific core and moving data only through queues proved sufficient to isolate radio interrupts and other asynchronous events.

The end-to-end delay combines the half-second sensor window, a 54 ms inference pass, and trivial queue overhead, yielding a worst-case latency of approximately 555 milliseconds. For rehabilitation feedback, this margin sits well below the one-second threshold that therapists consider acceptable. It would, however, be tight for applications such as powered exoskeleton control, which require sub-half-second responses.

Energy measurements indicate that the convolutional network contributes only 2.24 mA to a sensor-only baseline of 100 mA, maintaining an average draw of below 110 mA. A 500 mAh lithium-polymer cell, therefore, supports more than 4 h of operation. Radio bursts reach 150 mA, yet the class-change filter keeps the Bluetooth transmitter active less than seven percent of the time, so the impact on mean current is modest.

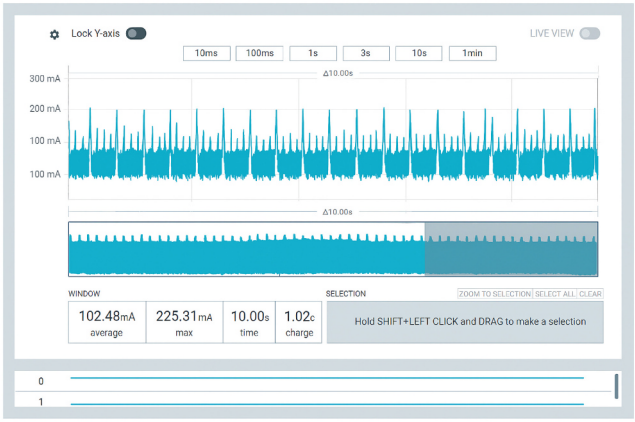
(a) Baseline: BLE @ 20 Hz, no RTOS or ML.

(b) FreeRTOS scheduler, BLE only.

Fig. 9. Current profiles for the baseline and FreeRTOS-only configurations.

Memory headroom is generous. The quantized network consumes 304 kB of flash and allocates 86 kB of RAM, leaving ample space for firmware updates or additional models. Static allocation and the absence of dynamic memory calls eliminate the risk of fragmentation during long runs.

(c) FreeRTOS + CNN inference (0.5 s windows).



(d) Same as (c) with LCD turned off.

Fig. 10. Current profiles for configurations including CNN inference. Disabling the display saves about 50 mA.

Together, these results confirm the central premise of the work: edge inference performs well when treated as a periodic real-time task with a clearly defined worst-case execution time. The scheduler, not the network, is the primary lever for reliable field performance.

Several weaknesses remain. The dataset contains recordings from only eight healthy volunteers wearing similar shoes, so the model may not generalize to atypical footwear, pathological gait, or significant variations in body mass. The network recognizes seven broad activities and cannot yet distinguish fine-grained states such as stair climbing or uneven terrain. The inference rate is fixed at one window every 500 ms, regardless of walking speed; an adaptive window tied to

step cadence could reduce latency and energy consumption. Baseline consumption is dominated by sensors and an always-on microcontroller, indicating that deeper sleep modes and duty-cycled sensing are necessary for prolonged outdoor wear. Finally, the design stores fixed weights; any personalization must occur off-device, followed by a firmware update, which undercuts the goal of complete privacy and autonomy. Future work should enlarge the dataset, test adaptive timing and low-power modes, and explore on-device continual learning that fits inside the existing memory budget.

6 Conclusion and Outlook

This work demonstrates that real-time human-activity recognition can be performed entirely within a shoe when inference is treated as an ordinary periodic task. A simple three-task FreeRTOS schedule on an ESP32-S3 meets every 20 Hz sampling deadline, while a post-training, quantized, one-dimensional convolutional network delivers 92.8% leave-one-subject-out accuracy in 54 milliseconds. The total latency from the first sample to the class label remains below 555 milliseconds, and the complete pipeline draws only 2.4 milliamps more than sensor-only operation, allowing for nearly 5 h of uptime on a modest 500mAh cell. The fundamental lesson that emerges is that the design of the scheduler, rather than the architecture of the model, often proves to be the predominant factor in achieving reliable edge intelligence. By assigning tasks to cores, determining worst-case execution times, and utilizing queues for all hand-offs, the insole circumvents the missed samples and jitter that often afflict bare-metal solutions. As memory and energy remain available, future research may investigate adaptive window sizes, deeper sleep modes, and on-device continual learning. These enhancements would propel the platform from a robust laboratory prototype to a practical, privacy-preserving instrument for routine gait monitoring and rehabilitation support.

Edge-AI, as proposed, enables power-efficient, real-time, and privacy-preserving HAR. It can also address challenges [9] like data handling and reducing annotation effort by processing sensor data directly on-device. To advance the state-of-the-art, one would explore adaptive sampling and inference rates tied to gait dynamics to further reduce latency and power consumption. In future, integrating on-device continual learning would allow the system to personalize models locally, enhancing privacy and performance. Finally, deploying deeper sleep modes and duty-cycled sensing could significantly extend battery life, paving the way for all-day wearable use in rehabilitation and gait monitoring scenarios.

Acknowledgments. This publication is a result of the research funded by the Federal Ministry for Economic Affairs and Climate Protection of Germany (KK5646401RH4) and by the Federal Ministry of Research, Technology and Space of Germany (03DPC0711A).

References

1. Agarwal, P., Alam, M.: A lightweight deep learning model for human activity recognition on edge devices. Procedia Comput. Sci. **167**, 2364–2373 (2020). https://doi.org/10.1016/j.procs.2020.03.289
2. Daghero, F., et al.: Human activity recognition on microcontrollers with quantized and adaptive deep neural networks. ACM Trans. Embed. Comput. Syst. **21**(4) (2022). https://doi.org/10.1145/3542819
3. Di Leo, K., Biagetti, G., Falaschetti, L., Crippa, P.: Microcontroller implementation of LSTM neural networks for dynamic hand gesture recognition. Sensors **25**(12) (2025). https://doi.org/10.3390/s25123831. https://www.mdpi.com/1424-8220/25/12/3831
4. Elstub, L., Grohowski, L., Wolf, D., Owen, M., Noehren, B., Zelik, K.: Effect of pressure insole sampling frequency on peak force accuracy during running. bioRxiv (2022). https://doi.org/10.1101/2022.05.18.492523
5. Elvitigala, D.S., Matthies, D.J.C., Weerasinghe, C., Nanayakkara, S.: Gymsoles++: combining google glass with smart insoles to improve body posture when performing squats. In: Proceedings of the 14th PErvasive Technologies Related to Assistive Environments Conference, pp. 48–54 (2021). https://doi.org/10.1145/3453892.3453898
6. Ferrari, P., Sisinni, E., Brandão, D., Rocha, M.: Evaluation of communication latency in industrial IoT applications. In: 2017 IEEE International Workshop on Measurement and Networking (M&N), pp. 1–6 (2017). https://doi.org/10.1109/IWMN.2017.8078359
7. Gabrecht, M.T., Wang, H., Matthies, D.J.C.: Pneushoe: a pneumatic smart shoe for activity recognition, terrain identification, and weight estimation. In: Proceedings of the 8th International Workshop on Sensor-Based Activity Recognition and Artificial Intelligence (iWOAR), pp. 1–5 (2023). https://doi.org/10.1145/3615834.3615853
8. Khandakar, A., et al.: Design and implementation of a complete wearable smart insole solution to measure plantar pressure and temperature (2022). https://arxiv.org/abs/2206.07779
9. Kirsten, K., et al.: The supervised learning dilemma: Lessons learned from a study in-the-wild. In: International Workshop on Sensor-Based Activity Recognition and Artificial Intelligence, pp. 181–195. Springer (2024). https://doi.org/10.1007/978-3-031-80856-2_12
10. Kos, A., et al.: Lightweight periodic scheduler in wearable devices for real-time biofeedback systems in sports and physical rehabilitation. Appl. Sci. **15**(12) (2025). https://doi.org/10.3390/app15126405. https://www.mdpi.com/2076-3417/15/12/6405
11. Lattanzi, E., Calisti, L., Contoli, C.: Are transformers a useful tool for tiny devices in human activity recognition? In: Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence, ICAAI 2024, pp. 339–344. Association for Computing Machinery, New York (2025). https://doi.org/10.1145/3704137.3704171
12. Li, X., Matthies, D.J.C.: Shoetect: detecting body posture, ambulation activity, gait abnormalities, and terrain with multisensory smart footwear. In: Proceedings of the 7th International Workshop on Sensor-based Activity Recognition and Artificial Intelligence (iWOAR), pp. 1–10 (2022). https://doi.org/10.1145/3558884.3558904

13. Lin, J., Chen, W.M., Lin, Y., Cohn, J., Gan, C., Han, S.: Mcunet: tiny deep learning on IoT devices (2020). https://arxiv.org/abs/2007.10319
14. Luna-Perejón, F., Salvador-Domínguez, B., Perez-Pe.ña, F., Rodríguez Corral, J.M., Escobar-Linero, E., Morgado-Estévez, A.: Correction: Luna-perejón et al. smart shoe insole based on polydimethylsiloxane composite capacitive sensors. sensors 2023, 23, 1298. Sensors **25**(11) (2025). https://doi.org/10.3390/s25113560. https://www.mdpi.com/1424-8220/25/11/3560
15. Maintainers, F.: Symmetric multiprocessing support in freertos kernel v11.1.0 (2024). https://www.freertos.org/symmetric-multiprocessing-introduction.html. Accessed 15 July 2025
16. Martini, E., et al.: Pressure-sensitive insoles for real-time gait-related applications. Sensors **20**(5) (2020). https://doi.org/10.3390/s20051448. https://www.mdpi.com/1424-8220/20/5/1448
17. Matthies, D.J.C., Roumen, T., Kuijper, A., Urban, B.: Capsoles: who is walking on what kind of floor? In: Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 1–14 (2017). https://doi.org/10.1145/3098279.3098545
18. Mazilu, S., et al.: Online detection of freezing of gait with smartphones and machine learning techniques. In: 2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops, pp. 123–130 (2012). https://doi.org/10.4108/icst.pervasivehealth.2012.248680
19. Nair, N., Thomas, C., Jayagopi, D.B.: Human activity recognition using temporal convolutional network. In: Proceedings of the 5th international Workshop on Sensor-based Activity Recognition and Interaction, pp. 1–8 (2018). https://doi.org/10.1145/3266157.3266221
20. Nazari, F., Shajari, A., Nahavandi, D., Mohajer, N.: Optimum signal duration for human activity recognition based on deep convolutional neural networks (2024). https://arxiv.org/abs/2406.11164
21. Park, B., Kim, M., Jung, D., Kim, J., Mun, K.R.: Smart insole-based abnormal gait identification: deep sequential networks and feature ablation study. Digit Health **11**, 20552076251333000 (2025)
22. Peretti, S., Contoli, C., Lattanzi, E.: An experimental study on the energy efficiency of feature selection for human activity recognition with wrist-worn devices. In: International Workshop on Sensor-Based Activity Recognition and Artificial Intelligence (iWOAR), pp. 40–54. Springer (2024). https://doi.org/10.1007/978-3-031-80856-2_3
23. Promwad Team: Choosing an rtos: Freertos vs. zephyr vs. threadx vs. mbed os (2025). https://promwad.com/news/choosing-rtos-freertos-zephyr-threadx-comparison. Accessed 15 July 2025
24. Santos, V.M., Gomes, B.B., Neto, M.A., Amaro, A.M.: A systematic review of insole sensor technology: recent studies and future directions. Appl. Sci. **14**(14) (2024). https://doi.org/10.3390/app14146085. https://www.mdpi.com/2076-3417/14/14/6085
25. Shakerian, A., Douet, V., Shoaraye Nejati, A., Landry, R.: Real-time sensor-embedded neural network for human activity recognition. Sensors **23**(19) (2023). https://doi.org/10.3390/s23198127. https://www.mdpi.com/1424-8220/23/19/8127
26. Shalby, H.H.Y., Roveri, M.: Dendron: enhancing human activity recognition with on-device tinyml learning. In: 2025 IEEE Symposium on Computational Intelligence on Engineering/Cyber Physical Systems (CIES), pp. 1–8. IEEE (2025). https://doi.org/10.1109/cies64955.2025.11007628

27. Tello, A., Degeler, V., Lazovik, A.: Too good to be true: accuracy overestimation in (re)current practices for human activity recognition. In: 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), pp. 511–517. IEEE (2024). https://doi.org/10.1109/percomworkshops59983.2024.10503465
28. Wang, Q., et al.: A wireless, self-powered smart insole for gait monitoring and recognition via nonlinear synergistic pressure sensing. Sci. Adv. **11**(16), eadu1598 (2025). https://doi.org/10.1126/sciadv.adu1598
29. Willnow, P., Sternitzke, M., Schlonsak, R., Gabrecht, M., Matthies, D.J.C.: Surfsole: demonstrating real-time surface identification via capacitive sensing with neural networks. In: International Workshop on Sensor-Based Activity Recognition and Artificial Intelligence (iWOAR), pp. 251–259. Springer (2024). https://doi.org/10.1007/978-3-031-80856-2_16
30. Zhang, H., Li, C., Liu, W., Wang, J., Zhou, J., Wang, S.: A multi-sensor wearable system for the quantitative assessment of Parkinson's disease. Sensors **20**(21) (2020). https://doi.org/10.3390/s20216146. https://www.mdpi.com/1424-8220/20/21/6146
31. Zuo, J., Arvanitakis, G., Ndhlovu, M., Hacid, H.: Magneto: edge AI for human activity recognition – privacy and personalization (2024). https://arxiv.org/abs/2402.07180