Real-Time Multiclass Human Activity Recognition Using Accelerometry on a Raspberry Pi

Alexander Diab-Liu
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Canada
alexdiabliu@gmail.com

***Abstract*—Human Activity Recognition (HAR) is a core building block for cyber-physical systems in biomedical engineering, enabling context-aware monitoring, rehabilitation, and assistive technologies. This work presents the design, implementation, and evaluation of a fully embedded, real-time multiclass HAR system running on a Raspberry Pi equipped with a Sense HAT accelerometer. 3-axis accelerometry data was collected across four target activities (sitting, walking, running, and turning on the spot clockwise) and trained on a linear Support Vector Machine (SVM) classifier with windowed statistical features. The pipeline transforms raw acceleration into overlapping temporal windows and extracts ten features per window, including axis-wise means and standard deviations, as well as magnitude statistics. A stratified 80/20 train–test split is used to evaluate generalization. The final SVM model achieves an overall accuracy of 94.32% on the held-out test set, with near-perfect classification across the four categories. The trained model was then deployed to the Raspberry Pi, where a lightweight real-time script performs rolling-window feature extraction and drives the Sense HAT LED matrix as an intuitive activity indicator. This project demonstrates that classical machine learning methods, when carefully engineered, are sufficient to deliver accurate and interpretable real-time HAR on resource-constrained embedded platforms.**

***Index Terms*—Cyber-physical systems, human activity recognition, Raspberry Pi, accelerometer, Sense HAT, support vector machine, embedded machine learning.**

I. Introduction

Cyber-physical systems (CPS) tightly couple computation with the physical world through observation, reasoning and actuation. In biomedical engineering, CPS underpin applications such as wearable health monitors, fall detection systems, gait analysis tools, and rehabilitation robots, where physiological or motion signals are transformed into context-aware decisions in real time.

Human Activity Recognition (HAR) using inertial sensors is a well-studied problem. However, many published systems either rely on offline analysis or assume access to powerful compute platforms. In contrast, practical biomedical systems often require on-body or near-body embedded hardware that is inexpensive, low power, and capable of running robust algorithms under tight resource constraints. This motivates the exploration of classical machine learning models and efficient feature engineering tailored to small embedded platforms.

The objective of this project was to develop a complete end-to-end HAR CPS that runs entirely on a Raspberry Pi with an attached Sense HAT. The system observes 3-axis accelerometer data, reasons about activity using a trained multiclass SVM, and acts by providing real-time visual feedback via the onboard LED matrix. The four classified activities are:

- Sitting (stationary, moderately slouched),
- Walking at a comfortable pace,
- Running/jogging,
- Turning on the spot clockwise.

The main contributions of this work are:

- A full end-to-end pipeline for data collection, labeling, feature extraction, model training, and real-time deployment on a Raspberry Pi.
- A compact window-based feature set that enables a linear SVM to achieve high multiclass accuracy while remaining computationally efficient.
- A real-time embedded implementation that demonstrates stable activity classification and intuitive LED-based feedback in a live setting.

II. Background and Related Work

Human Activity Recognition (HAR) has become a widely studied topic in wearable computing, rehabilitation engineering, and mobile health, driven by the proliferation of inertial sensors in smartphones and wearable devices. Among available sensing modalities, accelerometers remain the most commonly used due to their low power consumption, small form factor, and ability to capture whole-body movement without requiring external infrastructure. Classical accelerometer-based HAR pipelines generally follow a common structure: raw time-series data are segmented into short windows, transformed into statistical or frequency-domain features, and used as input to a supervised learning model. This paradigm has been used successfully across numerous applications, including fall detection, gait analysis, posture monitoring, occupational safety, and fitness tracking.

A substantial body of work has evaluated machine learning algorithms for HAR, ranging from simple models such as k-Nearest Neighbours and decision trees to more advanced techniques such as Support Vector Machines (SVMs), Random Forests, and Hidden Markov Models. Classical feature-based methods remain popular in embedded and real-time systems because they offer predictable computational requirements,

moderate memory usage, and interpretable behaviour. These advantages make them particularly suitable for cyber-physical systems (CPSs), where strict latency, power, and stability constraints limit the applicability of large neural networks or high-dimensional feature representations.

Support Vector Machines, in particular, have proven effective for HAR tasks that involve moderate numbers of activities and engineered features with good class separability. Linear SVMs maximize the margin between classes, leading to strong generalization even when training data are limited. Their inference cost reduces to a simple dot product with the learned weight vector, making them well-suited for deployment on microcontrollers and resource-constrained platforms. These properties have led to widespread adoption of SVMs in lightweight wearable systems, especially when interpretability and determinism are desirable.

In parallel, modern research has increasingly shifted toward deep learning approaches, such as convolutional and recurrent neural networks, which learn temporal and spectral features directly from raw sensor streams. While these models achieve state-of-the-art performance on large public datasets, they typically require substantial computational resources, large training datasets, and complex tuning procedures. Such requirements can be prohibitive for embedded CPS scenarios involving real-time processing on low-power devices like the Raspberry Pi or microcontroller-based wearables.

Unlike large HAR datasets that may contain dozens of activities, this project focuses on a small but meaningful subset of four: sitting, walking, running, and turning. These activities are directly relevant to mobility classification, posture assessment, and basic behavioural monitoring. The primary objective of this work is not to exceed the accuracy of deep learning systems on broad benchmarks, but rather to demonstrate that a carefully engineered linear SVM pipeline, combined with low-cost hardware and principled CPS design, is sufficient to deliver responsive, interpretable, and reliable real-time HAR. This approach highlights the continued relevance of classical machine learning techniques in embedded biomedical applications, particularly when transparency, efficiency, and deployability are prioritized.

III. System Design and Methodology

This section describes the complete pipeline used to design, train, and deploy the real-time Human Activity Recognition (HAR) cyber-physical system. The pipeline follows the Observe–Reason–Act framework, beginning with sensor data acquisition, progressing through feature extraction and SVM-based decision making, and concluding with LED-based feedback on the Raspberry Pi. Fig. **??** illustrates the overall data path from raw accelerometry to activity label and visual feedback.

A. Hardware Platform

All accelerometry data were collected using the Sense HAT inertial measurement unit (IMU) mounted on a Raspberry Pi 4. The Sense HAT integrates a 3-axis accelerometer capable of returning raw linear acceleration values along the x, y, and z axes at regular sampling intervals. Data were captured using the Python Sense HAT module, which provides real-time access to IMU data and simple access to the LED matrix and joystick.

The Sense HAT was placed on the wearer's torso, oriented upright and centered at the sternum. This position provides a stable reference frame for whole-body motion and is commonly used in wearable HAR research because it captures both upper-body posture and lower-body locomotion signatures without excessive rotational artifacts. Placement consistency was maintained throughout the data collection process to ensure reproducibility; the sensor was attached using the same strap and orientation for all recording sessions.

The IMU was sampled at an effective frequency of approximately 33.3 Hz, determined by the polling interval within the Python data collection script. The sampling rate was selected based on literature indicating that the majority of human motion-related accelerometry content occurs below 10 Hz [2]. Sampling at more than twice this upper bound satisfies the Nyquist criterion and ensures preservation of relevant frequency components while maintaining computational efficiency. Higher sampling rates were not pursued because they would increase storage and processing requirements without adding meaningful information for the relatively slow activities considered here.

B. Human Activity Protocol

To ensure consistent labeling and reproducible sequences, a structured protocol was followed for each of the four activities (sitting, walking, running, turning clockwise). For each activity, the wearer performed the motion continuously for approximately two minutes while standing or moving in an open indoor environment with minimal external disturbances. Two minutes per activity was chosen as a compromise between subject burden and the need for sufficient data to capture natural variability in gait and posture. Across four activities this yields roughly eight minutes of labelled data and, after windowing, hundreds of examples per class.

Activities were performed in the following order: sitting $\rightarrow$ walking $\rightarrow$ running $\rightarrow$ turning. Between activities, the wearer paused to reset posture and confirm the new activity label on the LED matrix, ensuring clear transitions between labelled segments. This sequential protocol simplified both data collection and later inspection of sensor traces for quality control.

Labeling was performed using the Sense HAT joystick. The up/down/left/right directions were mapped to activity labels 0–3, and pressing the center button toggled recording mode. The LED matrix displayed a distinct colour corresponding to the active label, allowing the subject to verify that label changes were registered correctly during collection. After recording, the resulting CSV files were briefly inspected in Python to ensure that label segments aligned with visible changes in acceleration patterns, providing an additional sanity check on the protocol.

C. Data Collection and Labeling

Data collection was performed using a custom Python script running on the Raspberry Pi. The program continuously polls the Sense HAT joystick for user input and the accelerometer for motion data. The joystick is used to both control recording and switch between labelled activities. For each recorded sample, the script logs a timestamp, the three accelerometer components, and an integer label to a CSV file.[1]

Four labels are defined:

0 Sitting (blue LED)
1 Walking (red LED)
2 Running (green LED)
3 Turning clockwise (yellow LED)

For each activity, the subject performed the motion continuously for two minutes while the system logged raw acceleration. Across four CSV files (one per label), the final dataset contains approximately 16 000 rows and five columns (timestamp, x, y, z, label) before cleaning. This corresponds to about 8 minutes of recorded behaviour, which is sufficient to train a lightweight classifier while keeping the overall dataset manageable for manual inspection and iteration.

Missing values are removed (although none were present in practice), and the index is reset, resulting in a clean dataset of 16 000 usable samples. Out-of-protocol samples at the very beginning and end of each file (pre-transition or post-transition noise) were trimmed to ensure that windows contained predominantly single-activity behaviour.

D. Windowing and Feature Extraction

Directly feeding individual accelerometer samples to a classifier would ignore temporal structure and be extremely sensitive to noise, which was observed with initial model proposals. Instead, a sliding-window approach was adopted to aggregate short segments of motion into fixed-length examples. For offline model training, the following configuration is used:

- Window size: 50 samples,
- Step size (stride): 10 samples.

At a sampling interval of roughly 30 ms, a 50-sample window corresponds to approximately 1.5 s of motion, with 0.3 s overlap between consecutive windows. This duration is long enough to capture multiple gait cycles during walking and running, while still being short enough to provide reasonably responsive predictions. A smaller window would reduce latency but yield noisier feature estimates; a substantially larger window would improve smoothness at the cost of slower response and fewer training examples. The chosen 50/10 configuration provides a practical trade-off between latency, statistical stability, and dataset size. To ensure balanced representation across classes, windowing is applied separately within each label subset.

For each window, we compute the following ten features:

- Means of each axis: μ_x, μ_y, μ_z,
- Standard deviations of each axis: σ_x, σ_y, σ_z,
- Magnitude statistics: mean magnitude μ_{mag}, standard deviation σ_{mag}, maximum magnitude $m_{\max}$, and minimum magnitude $m_{\min}$,

where magnitude is defined as

$$\text{mag} = \sqrt{x^2 + y^2 + z^2}.$$

These features capture both the overall posture (through mean acceleration) and the intensity and variability of motion (through standard deviations and magnitude statistics). The windowed dataset produced by this pipeline consists of 1 584 samples, each with ten features and an associated class label. Simple exploratory plots (histograms and scatter plots) reveal clear clustering of windows by label in this feature space, motivating the use of a linear classifier.

E. Train–Test Split and Class Balance

The windowed feature matrix is split into training and test sets using a stratified 80/20 partition:

- Train set: 1 276 windows (319 per class),
- Test set: 320 windows (80 per class).

Random stratification ensures that all four activities are equally represented in both sets, which simplifies interpretation of per-class performance and avoids biasing the model toward majority classes. Because each class contributes the same number of windows, macro-averaged metrics (which treat all classes equally) and weighted metrics (which weight by class frequency) are expected to be similar. This symmetry simplifies comparison of evaluation measures in Sec. IV.

F. SVM Model and Training Pipeline

A linear multiclass Support Vector Machine (SVM) was chosen due to its computational efficiency, interpretability, and strong performance in high-dimensional HAR tasks. The full processing pipeline is summarized in Fig. **??** and can be described as:

1) Raw accelerometer samples
2) Sliding-window segmentation
3) Ten-dimensional statistical feature vector
4) Normalization using `StandardScaler`
5) Linear SVM (one-vs-rest multiclass decomposition)
6) Predicted label and LED output via the Sense HAT

Support Vector Machines are margin-based classifiers that learn a separating hyperplane by maximizing the geometric margin between classes, ensuring that the decision boundary is as far as possible from the nearest training samples, known as support vectors. For a linear classifier of the form $f(x) = w^T x + b$, the margin is defined as $M = 2/\|w\|$, and maximizing M is equivalent to minimizing $\frac{1}{2}\|w\|^2$. Because real-world data are rarely linearly separable, slack variables ξ_i are introduced to allow margin violations, yielding the soft-margin objective $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$, where C controls the penalty for misclassified or within-margin points. The associated empirical loss is the hinge loss, $\ell(y, f(x)) =$

[1]The data collection script uses a small debounce interval on joystick events to avoid spurious label changes and samples data at roughly 30 ms intervals while recording.

$\max(0,\, 1 - y(w^T x + b))$, which penalizes incorrect and low-confidence predictions. Although SVMs are inherently binary classifiers, they can be extended to multiclass problems using either one-vs-rest, where one classifier is trained per class, or one-vs-one, where a classifier is trained for every pair of classes. After training, prediction is performed using the sign of the decision function, $y' = \text{sign}(w^T x + b)$, which determines on which side of the hyperplane a sample lies.

Training was performed on a PC using scikit-learn. The model pipeline consisted of:

- `StandardScaler` to standardize each feature to zero mean and unit variance,
- `SVC` with:
  - Linear kernel,
  - Regularization parameter $C = 1.0$,
  - One-vs-rest multiclass strategy,
  - Hinge loss via the primal optimization problem.

Extensive hyperparameter optimization was not the focus of this project; instead the standard choice $C = 1.0$ was adopted, as it already yielded high accuracy on the validation split and maintained good margin properties. Alternative models such as k-Nearest Neighbours and Linear Discriminant Analysis were considered conceptually, but a linear SVM offered a more attractive balance of robustness to outliers, interpretability, and predictable inference cost on the embedded platform.

The trained pipeline (scaler + SVM classifier) was serialized using `joblib` and transferred back to the Raspberry Pi for real-time inference. This separation of offline training and embedded deployment follows a standard CPS pattern in which computationally expensive learning is performed off-device, while the embedded node executes only the lightweight inference steps.

G. Validation Method

Model validation employed several metrics:

- Overall accuracy on the held-out test set,
- Per-class precision, recall, and F1-score,
- A confusion matrix to visualize misclassification patterns.

Precision and recall were computed as

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \qquad \text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

where TP_i, FP_i, and FN_i denote the number of true positives, false positives, and false negatives for class i, respectively. Macro-averaged scores were obtained by averaging the per-class metrics, giving equal weight to each activity. Weighted averages were also reported for completeness, although they coincide closely with macro values due to the balanced class distribution. Misclassifications were examined to evaluate separability between dynamic activities such as walking and running, and this analysis informed design choices such as window size and the need for additional sensor modalities.

H. Real-Time Embedded Deployment

For live operation, a separate Python script runs on the Raspberry Pi and loads the previously trained and serialized SVM pipeline. This script performs the following tasks in a continuous loop:

- Reads raw accelerometer values from the Sense HAT,
- Maintains rolling buffers for x, y, and z values,
- Computes the same set of ten features once the buffer reaches the configured window size,
- Calls the SVM pipeline's `predict` method to obtain an activity label,
- Maps the predicted label to a corresponding LED colour and updates the LED matrix.

The real-time script uses a slightly smaller window size (20 samples) and a sampling interval of approximately 50 ms, yielding a window duration of about 1 s. This configuration offers a good trade-off between responsiveness and stability of predictions: shorter windows reduce latency but are more sensitive to noise, whereas longer windows smooth predictions but respond more slowly to activity changes. A joystick-controlled state machine allows the user to start and pause classification and to cleanly exit the program. When paused, the LED matrix displays a white screen to indicate idle status. A compact bash script was set up to run the program automatically on boot using `systemd`, eliminating the need for remote login and providing a fully self-contained embedded CPS.

IV. Results and Discussion

A. Offline Evaluation

The trained SVM model was evaluated on the held-out test set of 320 windows (80 per activity). The model achieved an overall accuracy of 94.32%, indicating that the ten-dimensional statistical feature space provides a sufficiently discriminative representation for the four targeted activities. This confirms the hypothesis that simple second-order statistics derived from accelerometer signals are adequate for separating locomotor and non-locomotor behaviours under controlled conditions.

A confusion matrix provides a more detailed view of model performance. Sitting (label 0) and turning clockwise (label 3) achieve perfect classification with 100% precision and recall. These activities have highly distinctive signatures: sitting exhibits near-zero variance across all axes, while deliberate turning produces asymmetric rotational acceleration patterns that differ substantially from the translational motion present in walking or running.

Walking (label 1) maintains perfect recall (100%) but shows reduced precision (81%), with most false positives originating from running windows. Conversely, running (label 2) attains high precision but a lower recall of 78%. These errors follow a clear trend: both activities represent cyclical lower-limb gait, and their statistical windows overlap when walking speed increases or running cadence decreases. Transitional behaviour—such as accelerating into a run or decelerating into a walk—further blurs these boundaries, producing windows that lie near the SVM decision hyperplanes.

Macro-averaged precision, recall, and F1-score for the four-class problem are all approximately 0.93, reflecting balanced

performance across classes. Weighted scores remain similar, due to the intentionally balanced dataset. These results demonstrate that while a linear separator is sufficient for the primary modes, the walking–running continuum represents a fundamental ambiguity that may require richer dynamics (e.g., frequency-domain features or gyroscope data) for more definitive separation.

B. Real-Time Performance

When deployed on the Raspberry Pi, the system maintains real-time inference capability without dropping samples or introducing noticeable latency. With a 20-sample sliding window and a sampling period of approximately 50 ms, each prediction incorporates around one second of temporal context. This latency is acceptable for feedback-oriented applications, such as activity visualization or basic monitoring, and aligns with common design constraints of embedded CPS systems.

Qualitative evaluation shows that LED colour output tracks the user's movements smoothly and consistently:

- The LED matrix remains a stable blue during sitting, reflecting the near-stationary accelerometer profile.
- When the user begins walking, the display transitions to red within one window cycle. As gait becomes more vigorous, the system consistently classifies windows as running, corresponding to a green LED output.
- Turning behaviour produces a distinct and stable yellow indication, even when performed at varying angular velocities.

Occasional momentary misclassifications between walking and running appear during speed transitions, matching the patterns observed in offline results. Importantly, these misclassifications do not propagate or destabilize the system: once gait settles into a consistent pattern, the classifier rapidly converges to the correct state. This illustrates a desirable CPS property: predictive stability under transient disturbances.

C. Discussion

The combined offline and real-time results highlight several key insights about HAR on low-cost embedded hardware. First, classical machine learning methods remain highly competitive when paired with carefully engineered features. Despite relying solely on statistical descriptors, the linear SVM achieves performance comparable to more computationally expensive models reported in lightweight HAR literature. This is especially relevant for CPS settings, where resource-constrained devices must balance accuracy with interpretability and energy efficiency.

Second, the system's performance underscores the influence of human biomechanics on classification difficulty. Activities with distinct kinematic signatures (e.g., sitting, turning) are easily separable, whereas gait activities inhabit a continuum. Improving separability may require integrating temporal models (e.g., HMMs, RNNs) or richer sensor modalities (e.g., gyroscope, magnetometer), but at the cost of increased computational load.

Finally, the strong alignment between offline metrics and embedded performance indicates that the chosen window size, feature set, and model complexity form a well-calibrated design trade-off. The model is simple enough for real-time deployment yet expressive enough to capture meaningful distinctions in acceleration patterns.

D. Limitations

Although effective as a proof-of-concept, the system has several limitations:

- **Single-subject dataset:** All data were collected from one individual, limiting generalizability across body types, gait patterns, and sensor placements.
- **Environmental homogeneity:** Data were recorded indoors on flat ground, without variations such as slopes, uneven terrain, or external disturbances.
- **Restricted activity set:** Only four activities were modeled. Real-world movement involves composite behaviours (e.g., fidgeting, transitions, multitasking) that violate the assumption of one dominant activity per window.
- **Sensor modality limitations:** Accelerometer data alone may not capture subtleties in rotational motion or posture changes that could improve classification of similar activities.
- **Feature-level simplicity:** Statistical features ignore temporal structure within a window. Incorporating frequency-domain or waveform-shape descriptors could enhance separability, especially for gait-related tasks.

Despite these limitations, the system demonstrates that meaningful activity recognition can be achieved with minimal hardware and modest feature engineering, making it an attractive baseline for more advanced CPS implementations.

V. Conclusion and Future Work

This paper presented the full design, implementation, and evaluation of a real-time multiclass Human Activity Recognition (HAR) system deployed on a Raspberry Pi and Sense HAT platform. By leveraging window-based feature extraction and a linear Support Vector Machine (SVM), the system demonstrated that classical machine learning techniques remain highly effective for embedded cyber-physical systems that operate under strict constraints on latency, computational resources, and energy consumption. The resulting model achieved 94.32% overall accuracy across four activities, and its behaviour in real-time testing closely mirrored offline performance, indicating a well-calibrated match between algorithmic complexity and hardware capability. This consistency reflects one of the core goals of CPS engineering: ensuring predictable, stable behaviour when transitioning from controlled offline evaluation to real-world deployment.

Beyond accuracy, this work highlights several broader insights relevant to biomedical and wearable computing. First, interpretable feature-based models provide clear advantages in transparency and debuggability, especially when deployed on devices where behaviour must be understood and trusted

by clinicians, researchers, or end users. Second, the use of low-cost hardware demonstrates the potential for accessible and scalable monitoring tools that do not rely on expensive instrumentation or high-power processors. This reinforces the value of lightweight HAR systems for applications such as mobility monitoring, patient rehabilitation, remote assessment in low-resource settings, and real-time safety feedback for older adults or individuals with mobility impairments.

Looking ahead, there are several promising avenues for future work:

- **Multi-user data collection**: Expanding the dataset to include multiple participants of varying ages, body types, and gait patterns would enable robust subject-independent models and allow formal generalization testing.
- **Sensor fusion**: Combining accelerometer data with gyroscopes, magnetometers, or external wearable sensors could strengthen classification of rotational or transitional movements and improve separability between walking and running.
- **Expanded activity catalog**: Incorporating clinically significant actions—such as sit-to-stand transitions, stair ambulation, near-falls, or tremor events—would broaden the system's utility for biomedical monitoring and rehabilitation.
- **Temporal modeling**: Exploring models that explicitly encode temporal structure, such as Hidden Markov Models, LSTMs, or sequence-based SVMs, may improve recognition of activities with gradual transitions or rhythmic patterns.
- **Integration into larger CPS frameworks**: Embedding the HAR classifier into closed-loop assistive systems, telehealth platforms, or activity-adaptive feedback controllers would transform the prototype into a truly functional component of a complete cyber-physical system.

In conclusion, this project demonstrates that even modest hardware, paired with thoughtfully designed classical machine learning pipelines, can provide accurate, real-time, and interpretable activity recognition suitable for embedded biomedical applications. The system serves as a strong foundation for more advanced HAR architectures and underscores the enduring relevance of lightweight, well-engineered CPS designs in an era increasingly dominated by deep learning. The approaches developed here lay the groundwork for scalable, reliable, and clinically meaningful activity recognition systems capable of extending into future deployments across health monitoring, assistive technologies, and wearable computing.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] D. Arvidsson, J. Fridolfsson, and M. Börjesson, "Measurement of physical activity in clinical practice using accelerometers," Wiley Online Library, https://onlinelibrary.wiley.com/doi/10.1111/joim.12908 (accessed Dec. 2, 2025).

[3] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[4] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[5] N. Lane et al., "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[6] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, pp. 1–25, 2016.

SUPPLEMENTAL INFORMATION

This section provides additional material that supports the main text, including a full UML description of the cyber-physical system, data exploration plots, evaluation figures, and implementation details.

A. UML Diagrams of the CPS

Figures S1–S3 show the Unified Modeling Language (UML) representation of the Human Activity Recognition (HAR) cyber-physical system implemented on the Raspberry Pi and Sense HAT platform.

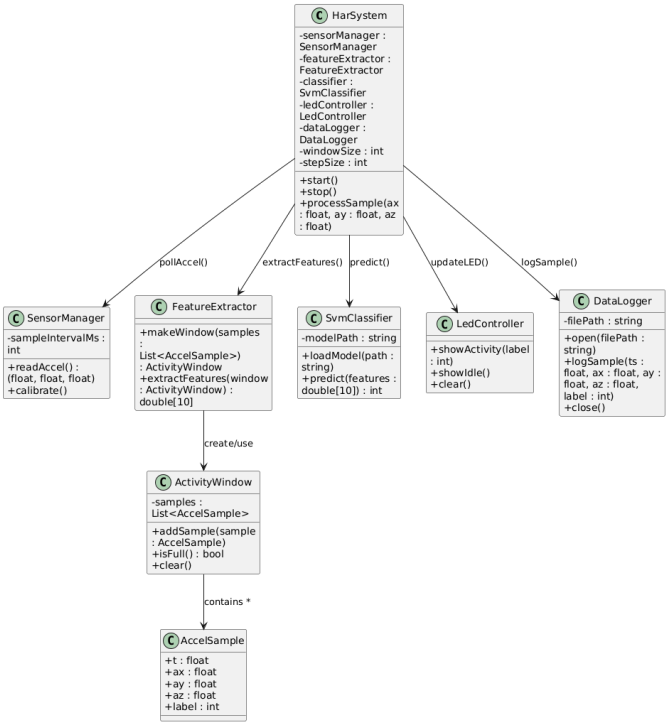

Fig. 1. Class diagram illustrating the core modules of the embedded HAR system and their relationships.

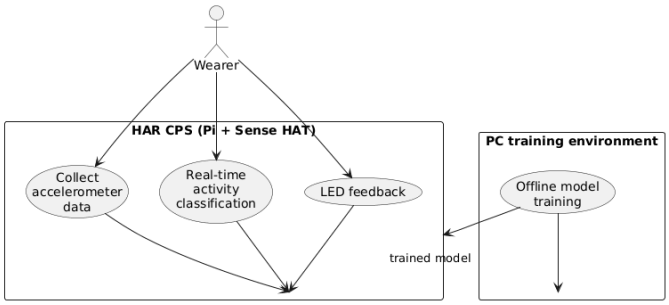

Fig. 2. High-level CPS relationship diagram linking the wearer, the HAR system, and the external PC used for offline model training

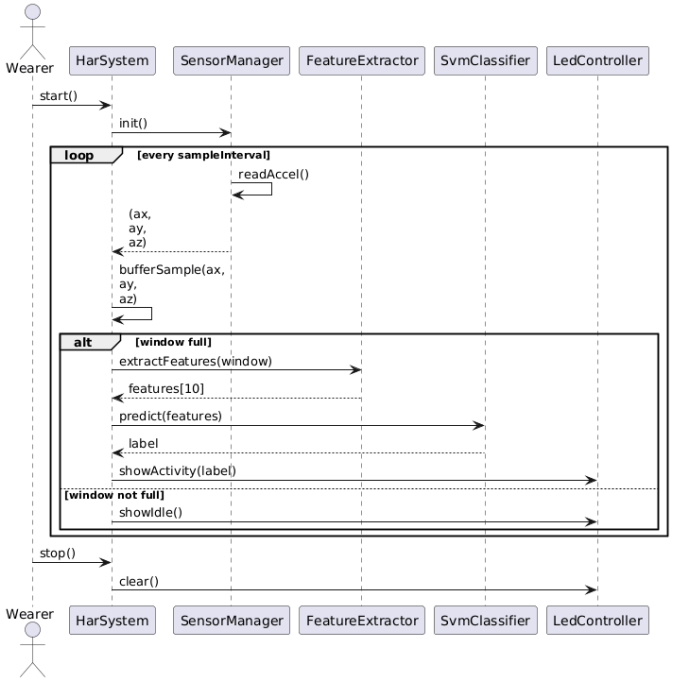

Fig. 3. Real-time HAR sequence diagram from sensing to classification and LED output.

B. Data Exploration and Feature Space Visualization

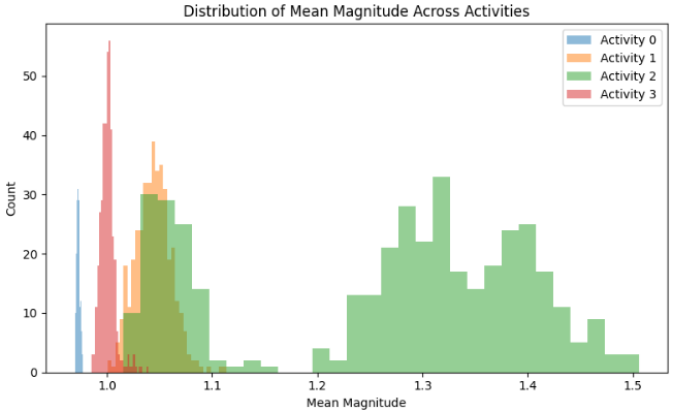

Fig. 4. Example histograms of raw accelerometer magnitudes for the four activities. The distributions highlight clear separation between sedentary (sitting) and dynamic (walking/running/turning) behaviour.

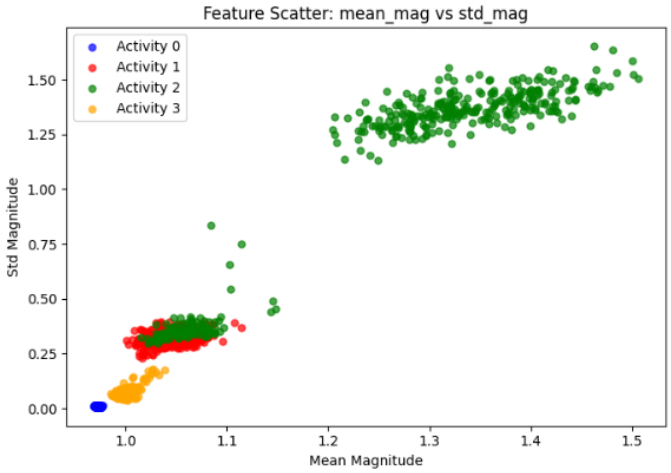

Fig. 5. Scatter plot of two representative features (e.g., mean magnitude vs. standard deviation of magnitude) for the windowed dataset. Clustering of samples by activity label motivates the use of a linear SVM in this feature space.

C. Evaluation Figures

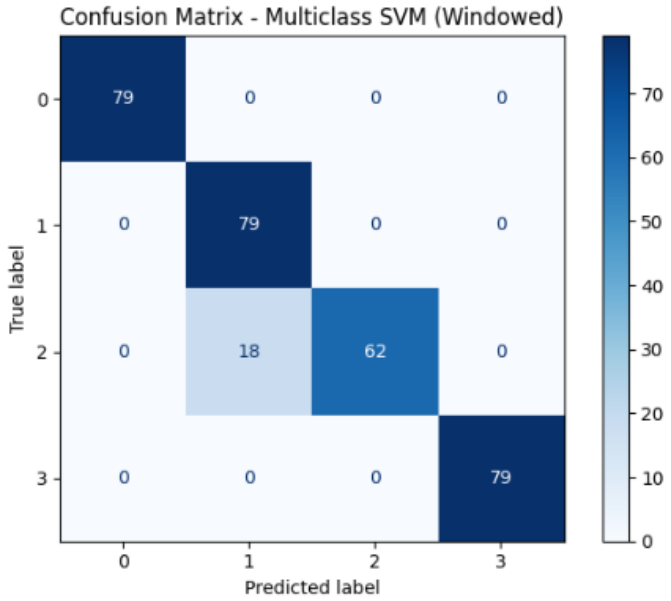

Fig. 6. Confusion matrix for the four-class SVM on the held-out test set (320 windows). The model achieves 94.32% overall accuracy, with perfect classification of sitting and turning, and occasional confusion between walking and running.

D. Implementation Details and Source Code

The complete implementation is provided on GitHub with the Python source code and supporting scripts:

- `collect_data.py` — Raspberry Pi data collection script for logging timestamped (x, y, z) accelerometer samples and joystick-based activity labels to CSV.
- `train_svm_har.py` — Offline training script that loads the collected CSV data, performs windowing and feature extraction, splits the dataset into training and test sets, trains the scikit-learn `StandardScaler` + `LinearSVC` pipeline, and exports the fitted model using `joblib`.
- `live_classifier.py` — Real-time classification script that runs on the Raspberry Pi, loads the trained SVM model, maintains rolling accelerometer buffers, computes the 10-dimensional feature vector per window, predicts the current activity, and drives the Sense HAT LED matrix.
- `har_service.sh` (optional) — Shell script or `systemd` service configuration used to launch the live classifier automatically on boot, enabling a fully self-contained embedded CPS.

All scripts were implemented in Python 3 using NumPy, pandas, scikit-learn, and the official Sense HAT library. Together, they provide a fully reproducible pipeline from raw sensor data to trained model and embedded deployment.