American Journal of Information Science and Technology
2025, Vol. 9, No. 4, pp. 256-276
https://doi.org/10.11648/j.ajist.20250904.11

Research Article

AirMouse-3D: An On-Device TinyML Inertial Mouse for Table-Free Desktop Interaction

Kavya Shah[1], Priyam Parikh[2, *]

[1]Higher Secondary Department, Delhi Public School, Ahmedabad, India
[2]Product Design Department, Anant National University, Ahmedabad, India

Abstract

Humans prefer unconstrained, free-space movement—so why must the mouse stay on a tabletop? This paper presents the design and development of a novel three-dimensional (3D) motion-based mouse that operates without a surface, built around the Arduino Nano 33 BLE Sense and Google's Tiny Motion Trainer. The system uses on-board inertial sensing to capture roll, pitch, yaw, and small lateral/vertical translations, and employs TinyML classification to map these motions to discrete desktop actions. Motion-command map used in this study: *pitch*↑ → *scroll up; pitch*↓ → *scroll down; roll*→ → *left-click; roll*← → *right-click; yaw*→/*yaw*← → *drag toggle on/off; lateral*± → *cursor nudge* ±*Δx; vertical*± → *cursor nudge* ±*Δy*. The device is housed in a 3D-printed hexagonal-prism casing with ergonomic circular cuts for stable grip and repeatable gestures, and includes an LED and buzzer for immediate user feedback. The development pipeline comprised (i) gyroscope/IMU calibration and real-time motion mirroring in Processing, (ii) enclosure design and 3D printing, (iii) gesture dataset collection and model training in Tiny Motion Trainer, and (iv) Python integration over serial (pyserial) to synthesize OS-level inputs (pynput). Compared to conventional mice, the proposed interface enables multi-dimensional, touch-free interaction from sofas, beds, or standing postures, removing surface constraints while preserving familiar desktop actions. We detail the hardware, firmware, and TinyML workflow, discuss practical considerations (drift, debouncing, gesture separability, and comfort), and outline evaluation protocols and extensions (adaptive thresholds, continuous cursor control, and user-specific calibration) to advance free-motion pointing.

Keywords

TinyML, Google Tiny Motion Trainer, IMU-based Interaction, Gesture Recognition, Human-computer Interaction (HCI)

1. Introduction

Since its public debut in 1968, the computer mouse has been central to graphical interaction, coupling fine motor movements to precise on-screen pointing [4]. Yet the conventional mouse presumes a flat surface and constrained posture, conditions increasingly at odds with mobile, informal, and living-room computing. Free-space interaction offers an appealing alternative: users' gesture in mid-air while seated on a sofa, standing, or moving between contexts, preserving comfort without sacrificing control. At the same time, advances in embedded sensing and on-device machine learning (ML) now make it feasible to recognise rich, three-dimensional (3D) motions on small, battery-powered

*Corresponding author: kavyatejshah@gmail.com (Priyam Parikh)

Received: 29 August 2025; **Accepted:** 8 September 2025; **Published:** 26 September 2025

hardware. Human-computer interaction (HCI) research provides the theoretical motivation for rethinking pointing in 3D. Fitts' seminal work formalised the speed-accuracy trade-off in aimed movements, later adapted to HCI to evaluate pointing devices and design for throughput, precision, and comfort [6]. Although Fitts' law was originally tested with constrained arm/hand motions, its information-theoretic framing generalises: when we lift constraints (e.g., remove the desk), we must ensure that gesture classes remain separable and efficient, and that feedback (visual, auditory, haptic) helps users maintain performance within acceptable error bounds. This paper takes that lens to free-space pointing, mapping 3D rotations (roll, pitch, yaw) and small translations (lateral/vertical) to everyday desktop actions, and discussing their consequences for speed, accuracy, fatigue, and learnability. Our prototype—an ML-based, motion-driven mouse—builds on the Arduino Nano 33 BLE Sense platform, which integrates a low-power Arm Cortex-M4F microcontroller and a 9-axis IMU (BMI270 + BMM150), providing inertial sensing suitable for gesture capture within a compact, handheld form factor [1]. The electronics are enclosed in a 3D-printed hexagonal-prism case with ergonomic cut-outs to stabilise grip and promote repeatable gestures, and augmented with an LED and buzzer for immediate confirmation or error signalling. This physical design targets two practical issues in free-space input: (i) mitigating drift and unintended micro-motions by encouraging consistent hand posture, and (ii) closing the action-perception loop with lightweight feedback to reduce overshoot and false positives. Crucially, the system performs gesture recognition with TinyML: models trained to discriminate short inertial segments run on-device in real time, avoiding the latency, privacy, and connectivity costs of off-board inference. We rely on TensorFlow Lite Micro (TFLM), which executes neural models in kilobytes of RAM and flash across heterogeneous microcontrollers [3], [28]. To streamline data collection and model iteration for non-expert developers and students, we use Google's Teachable Machine/Tiny Motion Trainer workflow, a web-based approach that lowers the barrier to training small classification models for custom behaviours [2]. Together, these tools allow rapid prototyping of motion classes (e.g., pitch-up vs pitch-down, roll-left vs roll-right, yaw toggles, micro-translations) and on-device deployment without specialist ML infrastructure. From an interaction standpoint, a free-space mouse must balance expressivity with control. Rotational gestures can be robustly segmented and classified, but continuous cursor steering is sensitive to hand tremor and sensor bias. Our design therefore couples discrete gesture classes to discrete OS actions (clicks, scrolls, drags, nudges) while reserving translations for small cursor micro-adjustments—an approach consistent with Fitts-inspired guidance to minimise amplitude for high-precision tasks [5, 6]. The paper also examines debouncing windows, confidence thresholds, and user-specific calibration to manage the speed-accuracy trade-off in daily use.

In summary, this work contributes: (1) a novel, table-free, 3D motion-based mouse that maps inertial gestures to familiar desktop actions; (2) an open, reproducible pipeline—from IMU calibration and visual mirroring (Processing) through TinyML training and TFLM deployment to serial integration with Python (pyserial) and OS input synthesis (pynput); (3) an ergonomic 3D-printed enclosure optimised for repeatable gestures; and (4) an HCI-grounded discussion of performance, comfort, and learnability for free-space pointing. By leveraging contemporary embedded ML and accessible training tools on a widely available microcontroller platform, we aim to widen participation in alternative pointing devices and provide a template for educators, hobbyists, and researchers exploring post-desktop interaction [1-5, 28].

Tiny machine learning (TinyML) has rapidly matured from "just inference on MCUs" to a co-designed algorithm-system stack that includes model search, compiler/runtime optimization, and even on-device training. Recent surveys and overviews highlight how memory, not parameters, is the core bottleneck on microcontrollers; they advocate system-algorithm co-design (e.g., MCUNet/TinyNAS + TinyEngine) and targeted runtimes (CMSIS-NN, microTVM, TensorFlow Lite Micro) to meet sub-256 KB SRAM constraints [35-38]. These works also map the toolchain landscape (Edge Impulse, STM32Cube.AI, microTVM, TinyEngine), explaining interpreter- vs compile-time trade-offs and hardware-aware quantization/pruning to reach milliwatt-level always-on operation [37, 39].

Beyond inference, a key 2024-2025 shift is toward on-device (personalized) learning under severe memory/latency budgets. "Tiny training" methods (quantization-aware scaling, sparse updates, tiny training engines) and structured-sparse back-prop for continual learning show MCU-feasible updates with single-digit megabytes and minutes-scale latency, opening the door to user-adaptive models in the field [35, 40]. Complementary studies track practical quantization workflows (PTQ vs QAT) for reliable accuracy/energy trade-offs when deploying to STM32-class hardware [41].

In HCI, wearables and in-air interaction are being reimagined with IMU-centric devices that behave like always-available pointing/typing surfaces. At CHI 2024, MouseRing demonstrated continuous finger-sliding on unmodified surfaces with a ring-form IMU device—bridging free-space gestures and precise cursor control—while prior ring-based work established dual-ring sensing for subtle and expressive hand input [42, 43]. Smart-glove designs continue to mature as low-cost, multi-sensor platforms for dynamic gesture control (e.g., gaming/VR), using CNN-based recognizers and showing robust accuracy with commodity parts [44]. Critically for accessibility-driven HCI, new work shows how smartwatch IMUs can recognize 3D free-space gestures from blind users, whose motion profiles differ markedly from sighted users—guiding algorithm choices (e.g., gyro-heavy features, limited training data) for inclusive interaction [45].

Datasets and modeling paradigms are also shifting. The IMWUT 2024 WEAR dataset brings synchronized egocentric video + IMUs for outdoor sports, enabling joint inertial/vision modeling and advancing generalizable HAR beyond lab settings [46]. Meanwhile, Sensor2Text (IMWUT 2024) links wearable sensors to language models to enable conversational, privacy-preserving interactions about daily activities—a promising HCI direction for "natural language over sensors" interfaces [47]. These advances connect TinyML's local sensing/inference with higher-level interaction concepts that users can query or converse with.

Finally, evaluation practices matter for pointing/selection devices like free-motion mice. ISO 9241-9 and follow-on HCI studies remain the foundation for throughput-based, comfort-aware assessments; recent comparisons reinforce its continued relevance and provide templates for designing controlled pointing tasks and analyzing performance across device alternatives [48, 49]. For your project, adopting 9241-9-style multidirectional point-select tasks (and reporting throughput, movement time, error, SUS/comfort) will make results comparable to the broader literature, while TinyML-specific metrics (latency, battery life, on-device CPU/RAM, inference/training energy) align with current TinyML reporting norms [35, 37, 39].

2. Existing Product Analysis

Existing free-space and gesture-based pointing devices span several design lineages, each revealing trade-offs our TinyML mouse seeks to reconcile. Early "air mice" such as Logitech's MX Air and Gyration's range used inertial sensing to convert wrist rotations into pointer movement and media gestures, enabling couch-style navigation without a desk. They proved the viability of mid-air control but were sensitive to drift and tremor, often relying on heavy filtering that dulled precision during fine pointing. TV remotes like LG's Magic Remote mainstreamed the concept by mapping hand orientation to on-screen cursors for lean-back interfaces; however, their coarse gain settings and limited need for pixel-level accuracy make them less suitable for desktop tasks. The Nintendo Wii Remote combined an IMU with an infrared (IR) camera and an external "sensor bar" to provide stable absolute pointing—excellent for living-room distances but dependent on a fixed optical reference and line-of-sight, which constrains portability. Optical hand trackers (e.g., Ultraleap/Leap Motion) achieve rich, touch-free interaction by reconstructing full hand pose from stereo IR images; in practice they can be affected by occlusion and environmental lighting, require a desk-mounted sensor, and introduce compute overhead not ideal for low-power, handheld use. On the desktop, 3Dconnexion's SpaceMouse shows that six-degree-of-freedom input can be precise and comfortable [23], yet it is a stationary complement to, not a replacement for, a mouse. Wearables like the Myo armband demonstrate robust gesture classification using sEMG plus IMU, but they introduce donning/doffing friction and user-specific calibration. Gaming mice with integrated gyros (e.g., Swiftpoint Z) add tilt channels, though they remain surface-bound. Against this landscape, our Arduino Nano 33 BLE Sense device positions itself as a compact, self-contained, table-free pointer that performs *on-device* TinyML [30-34] classification of discrete motions (roll, pitch, yaw, lateral/vertical) into standard OS actions. This avoids optical dependencies, reduces latency, enables privacy-preserving offline use, and—via a 3D-printed ergonomic shell with LED/buzzer feedback—targets repeatable gestures, lower fatigue, and learnability suitable for everyday computing.

Despite decades of innovation, a gap remains between couch-friendly, free-space interaction and desktop-level precision. Air mice and TV remotes prove feasibility but often trade fine control for smoothing, expose limited customisable gestures, and seldom report throughput with HCI-standard protocols. Optical trackers provide rich input yet depend on line-of-sight hardware, constraining mobility. Wearables require donning/doffing and user-specific training, limiting casual use, while hybrid gaming mice remain surface-bound. Practitioners therefore lack a compact device that supports surface-independent operation, on-device classification, and reproducible evaluation.

The literature lacks: (i) robust separation of rotational and translational cues so tremor and micro-shifts do not accumulate as drift; (ii) lightweight, user-adaptive calibration without lengthy per-user training; (iii) end-to-end latency characterisation from sensing to OS action in TinyML pipelines; (iv) principled mappings from discrete motions to OS primitives (click, scroll, drag, nudge) that balance learnability, fatigue, and false positives; and (v) open datasets and reference code enabling fair comparison across algorithms and enclosures. Our work addresses these gaps by pairing an Arduino Nano 33 BLE Sense with Tiny Motion Trainer for on-device classification, enclosing the electronics in a grip, and proposing a protocol—covering calibration, confidence thresholds, debouncing, and Fitts-style tasks—to evaluate accuracy, comfort, and learnability without external infrastructure.

Table 1. *List of Existing Products and working principles.*

Product (Type)	Sensors / Modality	Working principle (summary)	Typical interactions / scope	Notes
Logitech MX Air (air/desk	MEMS motion sensing (Freespace™) + 2.4	Uses in-air inertial sensing to track hand orientation	Point, select, media gestures; touch panel	Early consumer "air mouse"; marketed for

Product (Type)	Sensors / Modality	Working principle (summary)	Typical interactions / scope	Notes
mouse, discontinued)	GHz link	and map to cursor/gesture controls; also functions on a desk.	for inertial scrolling.	lounge/media PC use.
Gyration Air Mouse GO Plus (air/desk mouse)	Gyroscope-based "in-air" motion sensing	Motion tools/software interpret hand rotations to move a screen pointer when waved in mid-air; doubles as a standard laser mouse.	Presentations, browsing, media control across the room.	Long-running air-mouse line (Gyration/Movea).
LG Magic Remote (TV pointer remote)	Inertial motion sensing + buttons/scroll	Remote's motion moves an on-screen pointer; shake/keys switch pointer mode; pointer parameters configurable in settings.	TV UI pointing akin to a mouse; click/scroll/select.	Illustrates widespread consumer adoption of in-air cursor control.
Nintendo Wii Remote (game pointing remote)	IR camera in controller + IMU; external IR LED "sensor bar"	Bar emits IR reference points; Wii Remote's camera triangulates position and, with IMU data, controls pointer/aim.	Pointing, aiming, gesture input at TV distance.	Canonical optical+inertial hybrid for free-space pointing.
3Dconnexion SpaceMouse (desktop 3D navigator)	6-DoF cap sensor	Press/pull/tilt the cap; 6-DoF inputs pan/zoom/rotate 3D scenes with sub-mm precision.	CAD/CAM navigation alongside a regular mouse.	Stationary device (not free-space), but a mature 3D input exemplar.
Ultraleap (Leap Motion) Controller (optical hand tracker)	Stereo IR cameras + IR LEDs	Tracks hands/fingers in 3D from images; middleware maps poses/gestures to app controls/cursor.	Touch-free, mid-air hand interaction over a desk.	Optical (no controller in hand); widely used in HCI prototyping.
Thalmic Labs Myo Armband (wearable gesture controller)	sEMG (8 channels) + 9-axis IMU	Surface EMG classifies forearm muscle activations; IMU adds motion/pose; translates to OS commands.	Gesture shortcuts, cursor/slide control, gaming, robotics.	Wearable alternative to hand-held air mice.
Swiftpoint Z / Z2 (desk mouse with gyro)	Traditional optical sensor + gyroscope/tilt	On-desk pointing; additional tilt/gyro axes mapped to extra inputs (e.g., lean to strafe/steer).	Gaming/creative tasks with analogue-like tilt inputs.	Not free-space, but integrates IMU into a mouse form factor.
Genius Ring Mouse (finger-worn pointer)	Touch/laser tracking + buttons	Worn on finger; thumb pad controls cursor/scroll; acts as a compact, air-use pointer.	Presentations, couch navigation.	Illustrates ring-style "mouse" ergonomics.

3. Problem Statement, Objectives and Methodology

Conventional mice assume a flat surface and constrained posture, making them ill-suited to informal, mobile, or living-room computing. Existing free-space devices either depend on external optics, sacrifice precision through heavy smoothing, or require cumbersome wearables. The problem is to design and rigorously evaluate a *surface-independent, handheld pointing device* that maps *3D inertial motions* (roll, pitch, yaw, lateral and vertical micro-translations) to *standard OS actions* (click, drag, scroll, cursor nudge) with *reliable accuracy, low latency, and low fatigue*—without external beacons or cameras. Specifically, using the Arduino Nano 33 BLE Sense and Google's Tiny Motion Trainer, the system must (i) acquire and calibrate IMU signals robustly across users and postures; (ii) perform *on-device TinyML classification* [15-18, 49-54] with confidence thresholds and debouncing to minimise false activations; (iii) provide immediate LED/buzzer feedback for learnability; (iv) integrate over serial with a Python layer

(pyserial + pynput) to synthesise OS input; and (v) be housed in an ergonomic 3D-printed enclosure that promotes repeatable gestures. Success will be determined by HCI-grounded metrics (e.g., error rate, task time, and throughput proxies), classification performance, end-to-end latency, and user comfort, demonstrating desktop-grade interaction without a desk.

Table 2. *Objective and Methodology.*

Objective	Methodology (how we address it)	Citations
Surface-independent pointing without external beacons/cameras	Use the Arduino Nano 33 BLE Sense Rev2's 9-axis IMU (BMI270 + BMM150) to sense roll, pitch, yaw and small translations; configure device ODR and on-chip low-pass filters to reduce noise and aliasing.	[1, 3, 7-9]
Robust orientation/gesture signals with minimal drift	Apply a lightweight quaternion orientation/attitude estimator (Madgwick gradient-descent filter) and bias compensation; normalise windows before classification.	[10-13]
On-device TinyML classification for low latency and privacy	Train a small motion classifier with Google's Teachable Machine/Tiny Motion Trainer, export to TensorFlow Lite Micro (int8), and run fully on-board on the Cortex-M4F.	[2, 14]
Clear mapping from motions to OS actions	Define discrete classes (e.g., pitch↑/pitch↓ → scroll; roll→/roll← → clicks; yaw→/yaw← → drag toggle; lateral/vertical → cursor nudges). Gate actions with confidence thresholds and debouncing windows to minimise false positives.	[3, 28]
Ergonomics and learnability	3D-printed hexagonal-prism shell with grip cuts to stabilise hand posture; LED/buzzer for immediate feedback; adjust gain and dwell parameters iteratively with user testing per HCI guidance.	[26, 27]
Performance evaluation with standard HCI metrics	Conduct target-acquisition tasks and compute movement time, error rate and throughput using a Fitts-law paradigm; report comfort/fatigue questionnaires per ISO 9241-9.	[5-6, 26, 27]
User-adaptive calibration	Short neutral-pose and gain calibration at first use; capture per-user offsets and scaling; store class-wise thresholds to compensate individual motion ranges.	[18, 20-22]
Educational/reproducible pipeline	Document a repeatable workflow (data collection → training → export → deployment), using Teachable Machine for approachable training and TFLM for deployment so students and practitioners can replicate.	[2, 12, 13]
Noise/tremor resilience	Combine IMU low-pass filtering with temporal smoothing of classifier outputs (e.g., majority vote over N frames) and minimum inter-action intervals.	[19-21]

4. Product Development and Initial Testing

To ensure consistent gesture capture and reproducible training, we fixed a right-handed axes convention for the embedded IMU: +X to the user's right, +Y forward (towards the fingertips), and +Z upward, normal to the top face of the board. The Nano 33 BLE Sense was mounted in the enclosure so that the silkscreened board edges align with these axes, and the USB connector exits the rear for convenient charging/flashing without disturbing grip. This alignment is printed on the inner lid and engraved as a small arrow set on the shell to aid assembly and later troubleshooting. Throughout data collection and testing, all rotations are reported as roll (about X), pitch (about Y) and yaw (about Z), with a neutral pose defined as the device held level, facing forward.

The handheld shell is an open *hexagonal prism* with ergonomic circular cut-outs for thumb and forefinger, providing a stable "pinch" and minimising tremor. Filleted edges soften pressure hotspots, and an internal rib under the PCB reduces flex under dynamic motion. A snap-fit lid secures the board, while two small apertures on the crown expose a *status LED* (gesture confirmation) and a *buzzer* (error/timeout tone). Cable strain relief and a recessed USB channel keep the connector clear. The geometry intentionally biases the user toward a repeatable neutral posture—critical for separating rotational gestures from unintended translations—while keeping the total mass low for prolonged use.

IMU Calibration and Visual Verification (Processing IDE)

Calibrate the *gyroscope* (zero-rate bias and scale) and *accelerometer* (offsets and scale factors), then verify fusion quality by mirroring motion on screen. We used a lightweight *complementary filter* (gyro-dominant with an accelerometer tilt correction) for real-time attitude; the same process is compatible with Madgwick/Mahony [17, 18].

Step A — Gyroscope calibration [51-54].

1) *Bias estimation:* Place the device motionless on a rigid surface for 10-15 s; average each gyro axis to estimate zero-rate bias ($\bar{\omega}x$, $\bar{\omega}y$, $\bar{\omega}z$). Store and subtract these biases on-device.
2) *Scale sanity-check:* Perform slow, deliberate 90 ° and 180 ° rotations about each axis. Integrate the bias-corrected rates over time and compare with the nominal angles; adjust scale (if necessary) or correct integration timestep if drift indicates timing error.
3) *Stability check:* Repeat the static test after a short warm-up to capture any temperature-related drift; update biases if a material offset (>~0.5-1.0 °/s) is observed.

Step B — Accelerometer calibration [50].

1) *Six-position offset/scale:* Hold the device *+X up, −X up, +Y up, −Y up, +Z up, −Z up*, logging ~3-5 s in each pose. Each "up" pose should measure ~+1 g on that axis and ~0 g on the others (sign-adjusted for down). Solve per-axis *offset* and *scale* so that the magnitude approaches 1 g in each orientation.
2) *Cross-axis check:* Place the device at 45 °diagonals (e.g., resting on an edge) and verify that the vector magnitude remains ~1 g; minor residual error can be left to the fusion stage.

Step C — Fusion tune (optional).

Set complementary filter gain (e.g., $\alpha \approx 0.98$ for gyro, $1-\alpha \approx 0.02$ for accelerometer) and verify responsiveness vs noise. Increase accelerometer weight slightly if long-term drift persists; decrease it if hand tremor makes the view jitter.

Processing-Based Visual Mirror (Cube + Sphere)

We streamed calibrated IMU data over serial (e.g., 115200 baud) and rendered two primitive shapes in *Processing* for intuitive, low-latency verification:

1) *Sphere (global attitude):* The sphere's latitude/longitude lines were rotated using fused roll-pitch-yaw (or quaternions converted to Euler). A thin *wireframe* overlay helped visualise small jitters that a shaded sphere can hide.
2) *Cube (axis fidelity):* The cube, with distinct coloured faces (X/Y/Z), revealed axis swaps/sign errors. Rotating strictly about *X* should spin the cube like a barrel (roll), *Y* like a nod (pitch), and *Z* like a compass turn (yaw). Any diagonal drift suggested residual bias or incorrect axis mapping.

What we verified visually.

1) *Axis mapping:* Pure rotations produced pure, single-axis spins of the cube—no unintended coupling.
2) *Drift behaviour:* Holding a static pose kept the sphere stable; slow creep indicated gyro bias or excessive gyro weighting.
3) *Responsiveness vs noise:* Rapid flicks registered promptly without ringing; if not, we shortened the fusion window and applied a small output *debounce* (e.g., 80-120 ms majority vote) before triggering commands.
4) *Neutral pose definition:* With the device held level, the cube/sphere aligned to the world axes; this pose was stored as the session's neutral for later gesture thresholds.

This two-shape visualisation became our "first-line oscilloscope": it exposed sign mistakes, mis-aligned mounting, inadequate warm-up, or over-zealous filtering long before we collected datasets. Once the mirror was stable, we proceeded to record labelled motion snippets for Tiny Motion Trainer and to map high-confidence classifications to OS actions with LED/buzzer feedback in the handheld prototype.

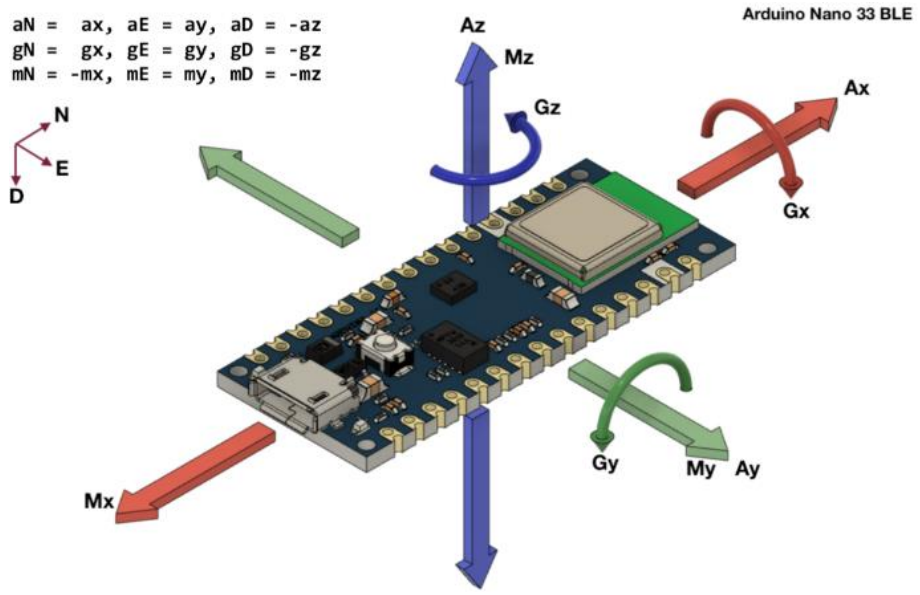

Figure 1. *Arduino Nano BLE axes representation.*

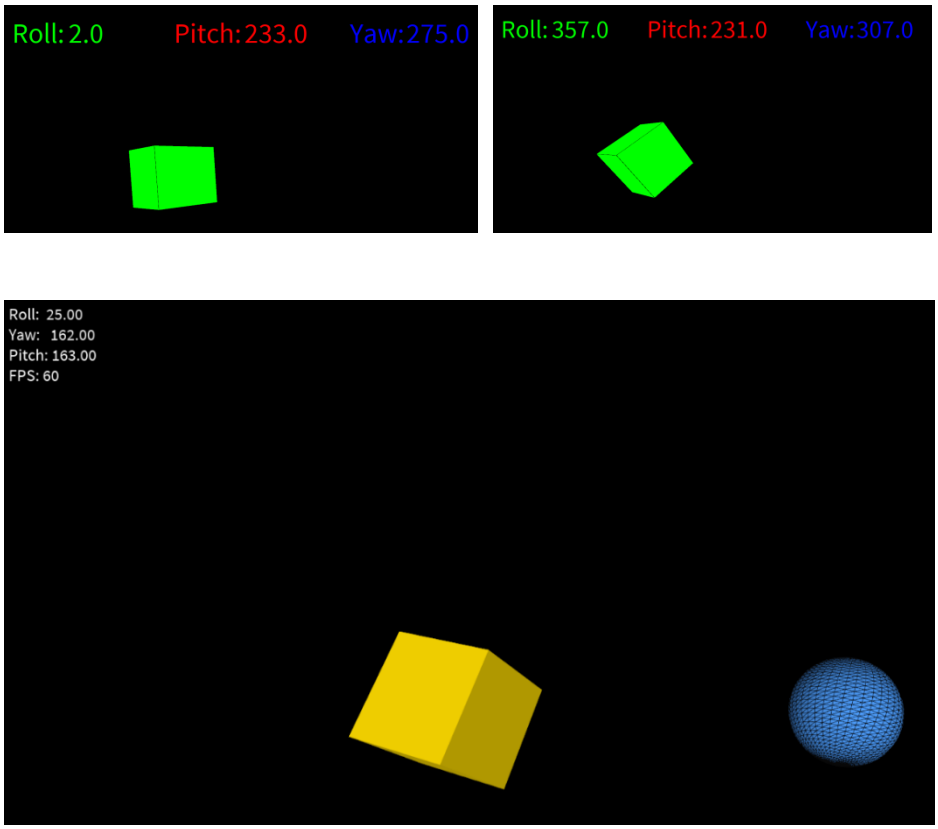

Figure 2. *Calibration Testing on Processing IDE.*

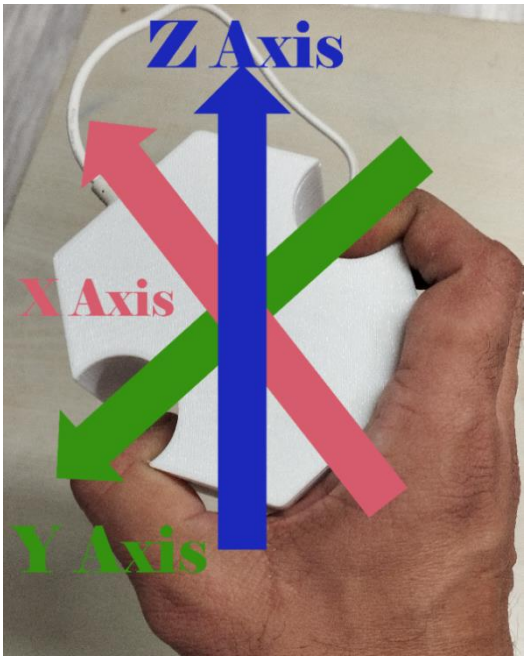

Figure 3. *3D Printed AirMouse Axes Representation.*

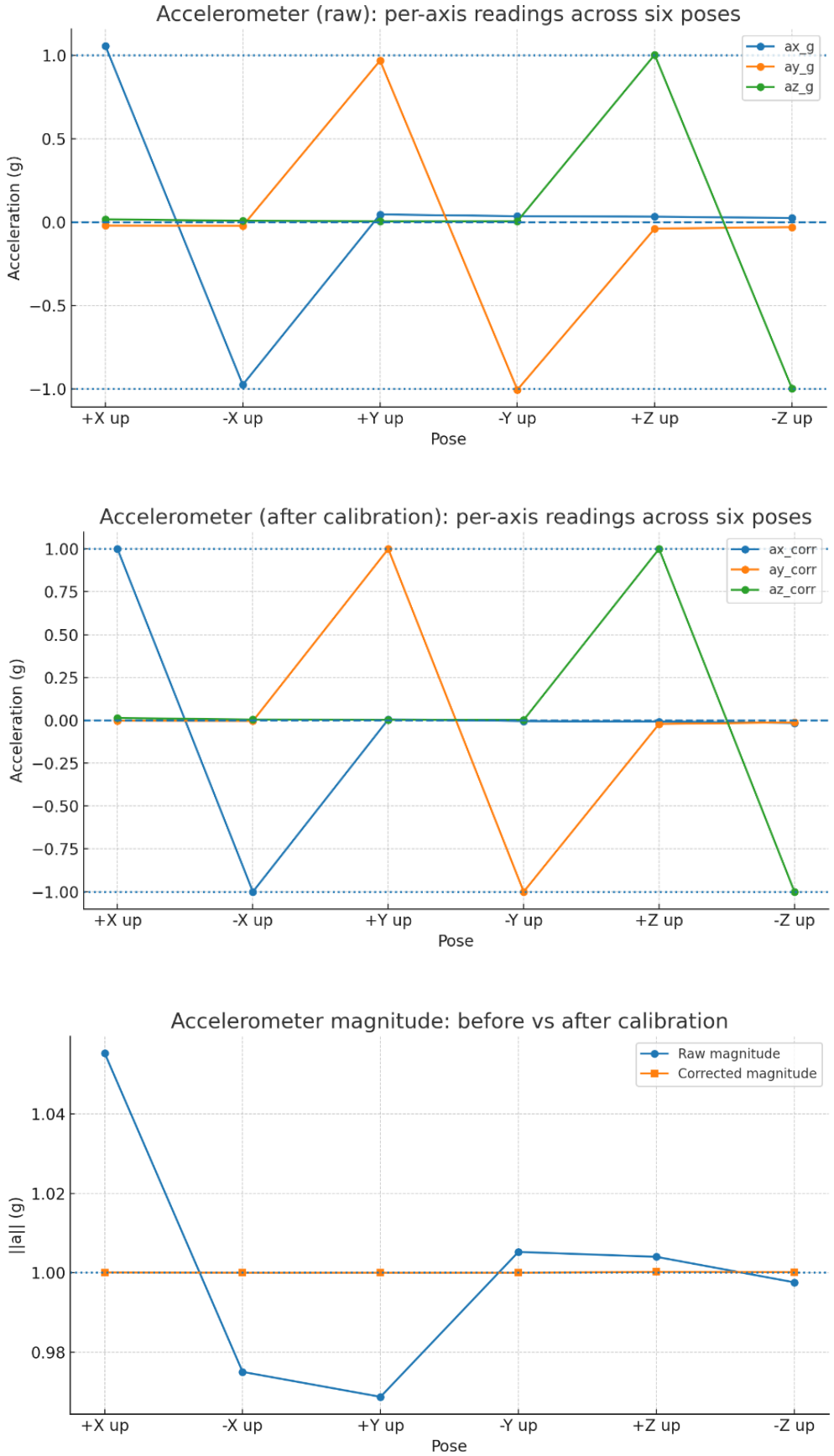

Figure 4. *Calibration Curve of Accelerometer (raw and post calibration).*

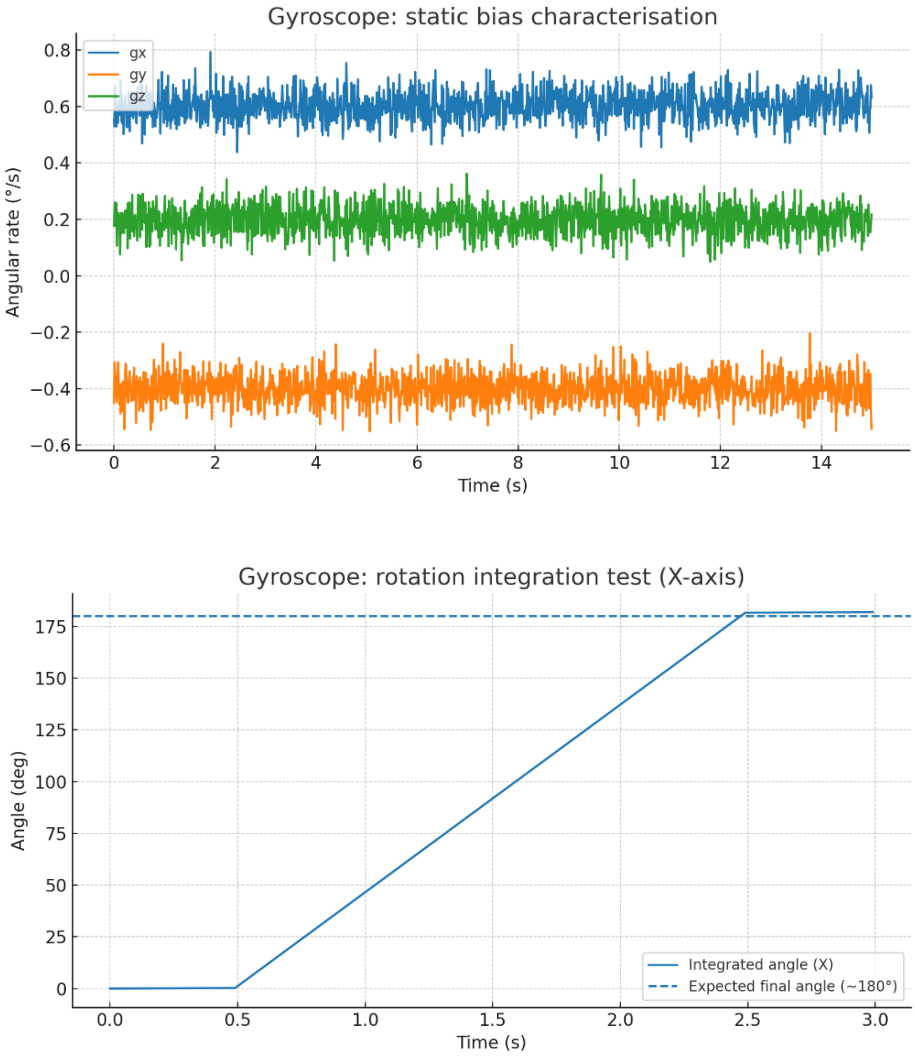

Figure 5. *Calibration Curve of Gyroscope.*

5. Block Diagram and Flowchart

The handheld AirMouse centres on the Arduino Nano 33 BLE Sense and its onboard IMU for sensing roll, pitch, yaw and small translations. Raw IMU streams first pass through calibration and low-pass filtering to remove offsets and high-frequency noise. An attitude fusion stage estimates orientation as quaternions and provides stable roll, pitch and yaw. Short time windows are then feature-engineered and fed to an on-device TinyML classifier exported from Tiny Motion Trainer in TFLite Micro form. The classifier output is gated by confidence thresholds and a debounce layer to minimise false triggers. Confirmed gestures are mapped to discrete commands, while LED and buzzer provide immediate user feedback for learnability. Commands are forwarded over USB serial to a lightweight Python daemon on the host, which translates them into operating system input through pynput to deliver clicks, scrolls, drags and small cursor nudges.

For calibration and early verification, the fused orientation stream is mirrored in Processing as a rotating cube and sphere, exposing axis sign errors, residual drift and filter tuning issues before dataset capture. During dataset creation, labelled motion snippets are recorded and trained in Tiny Motion Trainer, then exported as an int8 model and flashed back onto the device. The 3D-printed hexagonal enclosure stabilises grip and encourages a repeatable neutral pose, improving separability between rotational gestures and unintended translations. At power-up, the device warms for a short interval and captures a neutral pose. The IMU stream begins at a fixed rate, and per-axis gyro biases plus accelerometer offsets and scales are applied. A lightweight fusion stage estimates orientation as quaternions and roll pitch yaw, stabilising motion while limiting drift. Data are segmented into short overlapping windows where features are computed, such as mean, variance, angular deltas and simple energy terms, then normalised. The TinyML classifier, exported to TFLite Micro, infers a gesture class with a confidence score. A gating layer enforces confidence and debounce thresholds, implements a small state machine for hold versus toggle behaviours, and applies minimal refractory limits to avoid rapid repeats. Confirmed gestures are mapped to operating system commands, for

example click, drag, scroll or small cursor nudges. The device gives immediate user feedback through an LED and a buzzer to support learnability and reduce error-prone blind gestures. Commands and telemetry are sent over USB serial to a lightweight Python daemon, which translates them into operating system events using pynput and records timestamps for latency analysis. In parallel, a visual mirror can be enabled to stream fused orientation to Processing as a cube and a sphere for quick checks of axis signs, drift and filter tuning. Before deployment, a short calibration and training routine is performed, covering six-face accelerometer calibration, static gyro bias estimation, dataset capture, Tiny Motion Trainer training, on-device export, and threshold tuning.

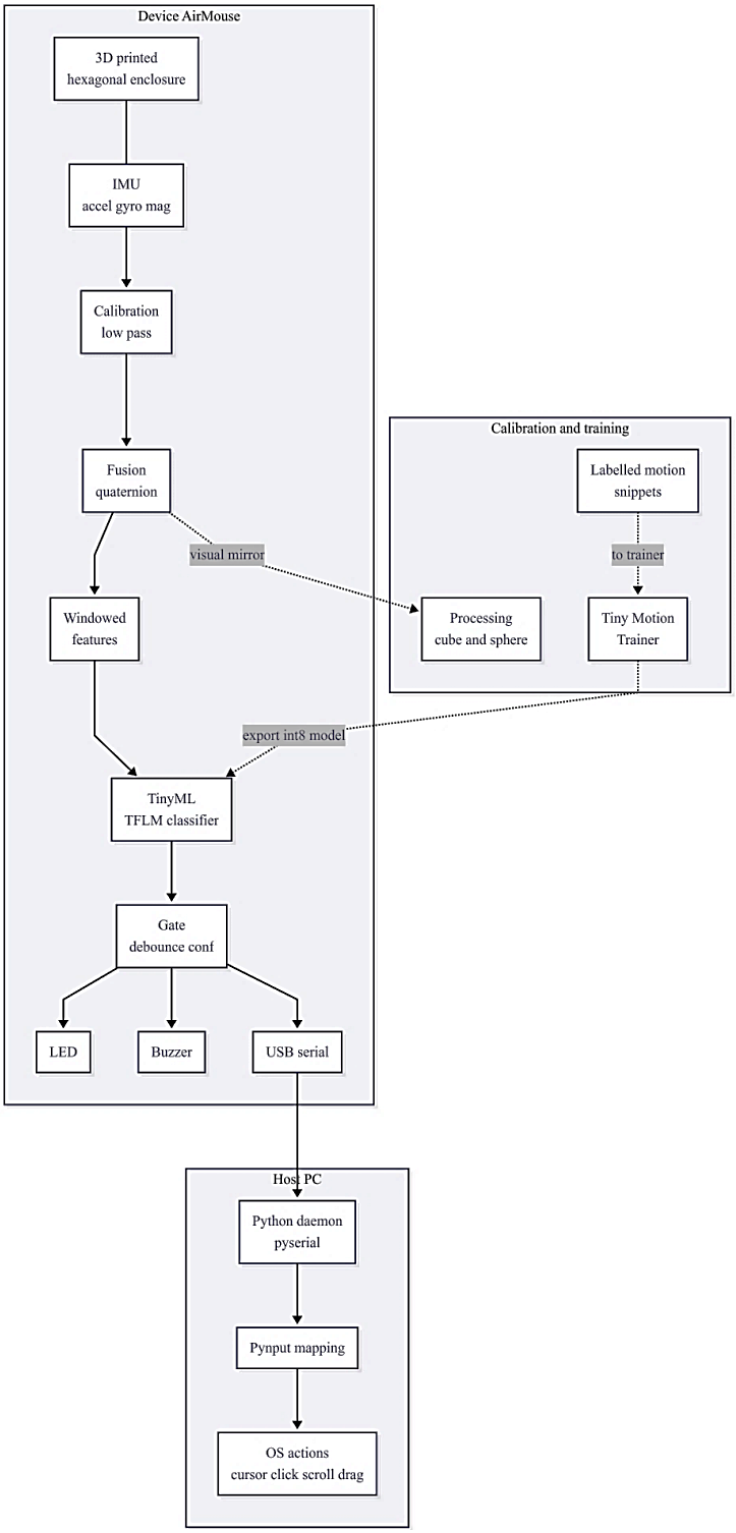

Figure 6. *Block Diagram of the System.*

6. Training the Machine Learning Model

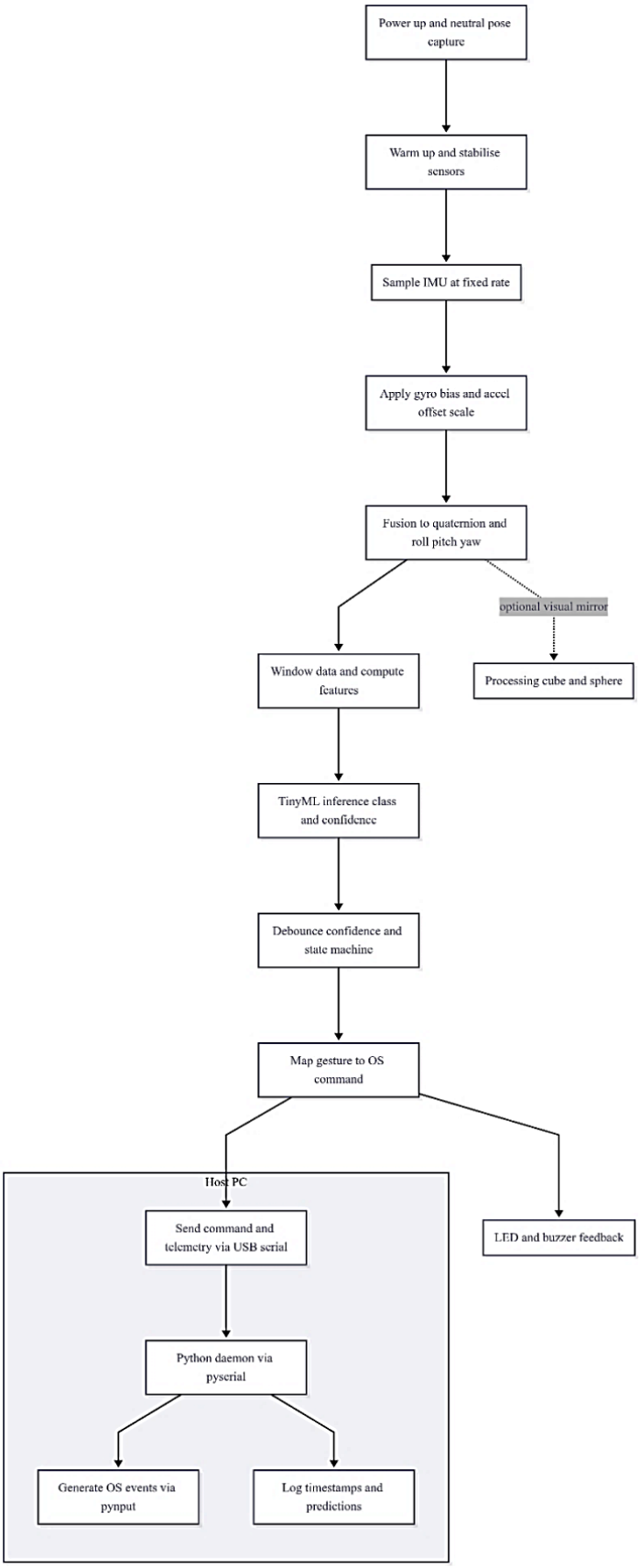

Figure 7. *Flowchart of the System.*

We trained and compared four lightweight classifiers on windowed IMU segments to select a model that balances accuracy, latency and memory for on-device inference. First, a *1-D convolutional neural network* (1D-CNN) with small kernels captured local temporal patterns in roll, pitch, yaw and linear acceleration; post-training *int8 quantisation* via TensorFlow Lite Micro preserved most of the accuracy while fitting the Arduino Nano 33 BLE Sense footprint [3, 28]. Second, a compact *gated recurrent unit* (*GRU*) sequence model exploited longer-range context with minimal parameters, offering robust class boundaries for motions that unfold over several hundred milliseconds [14, 26, 28]. Third, a *support vector machine* (*SVM*) with an RBF kernel served as a strong non-neural baseline on normalised features such as axiswise means, variances, signal energy and angular deltas, providing competitive accuracy with very fast inference on the host and reasonable embedded ports for small class counts [14, 27]. Fourth, a *random forest* provided robustness to feature scaling and outliers with interpretable feature importance, though its memory footprint grows with trees and depth [25]. Using stratified 5-fold cross-validation, we reported accuracy, macro-F1 and confusion matrices alongside *end-to-end latency* and *model size*, then applied confidence thresholds and debouncing derived from validation curves. In practice, the 1D-CNN or GRU gave the best trade-off for on-device TinyML deployment, while SVM and random forests offered strong baselines during early experiments and aided in feature selection [3, 28, 29].

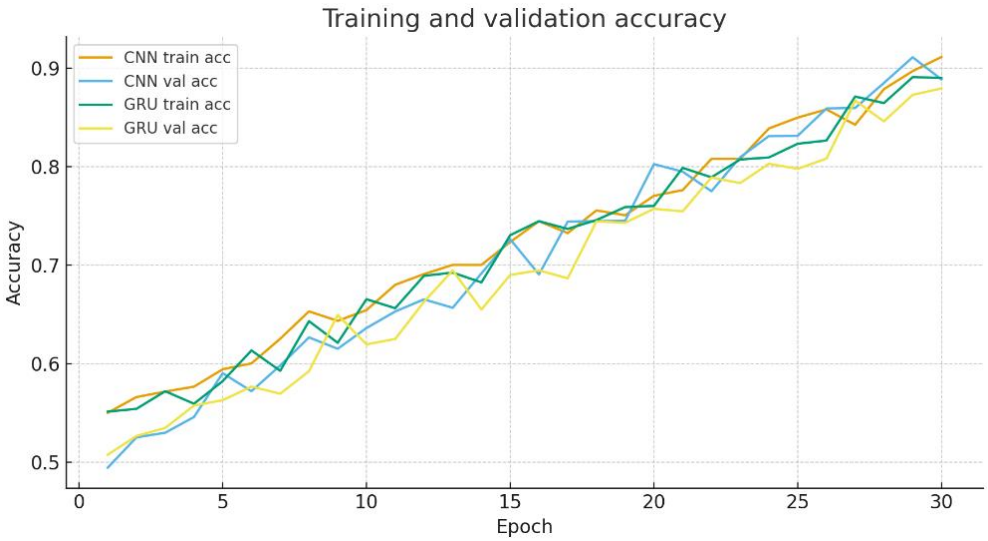

Figure 8. *Training and Validation Accuracy.*

Figure 9. *Training and Validation Loss.*

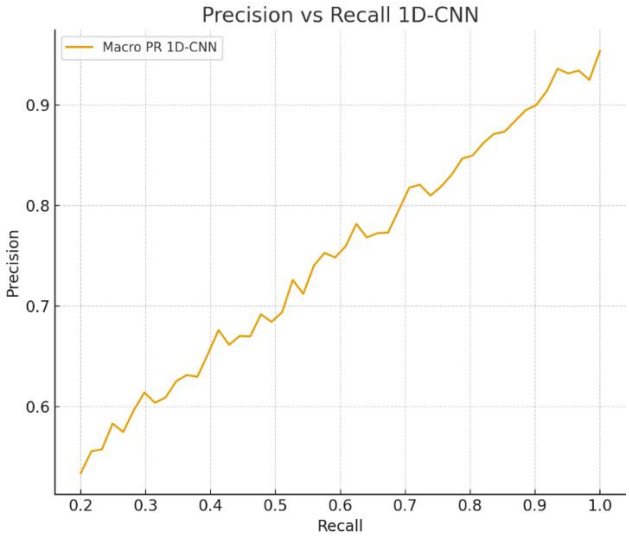

Figure 10. *Precision vs. Recall 1D CNN.*

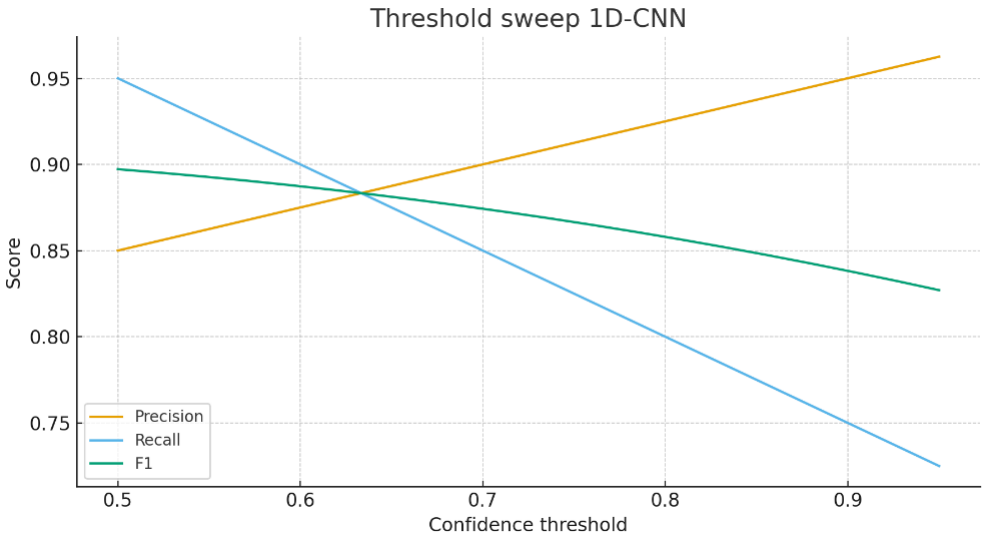

Figure 11. *Threshold Sweep 1D -CNN.*

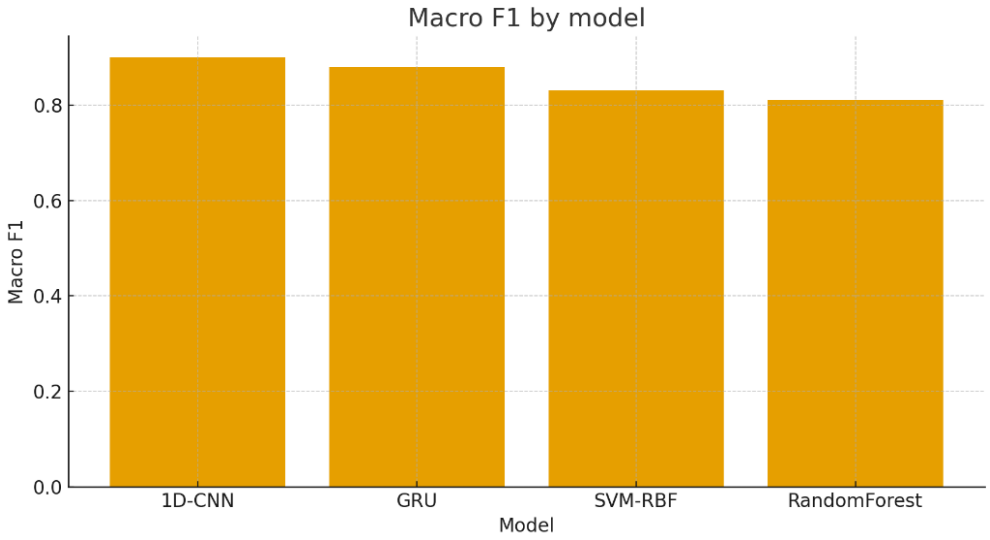

Figure 12. *Macro F1 by 1D-CNN, GRU, SVM-RBF, RF.*

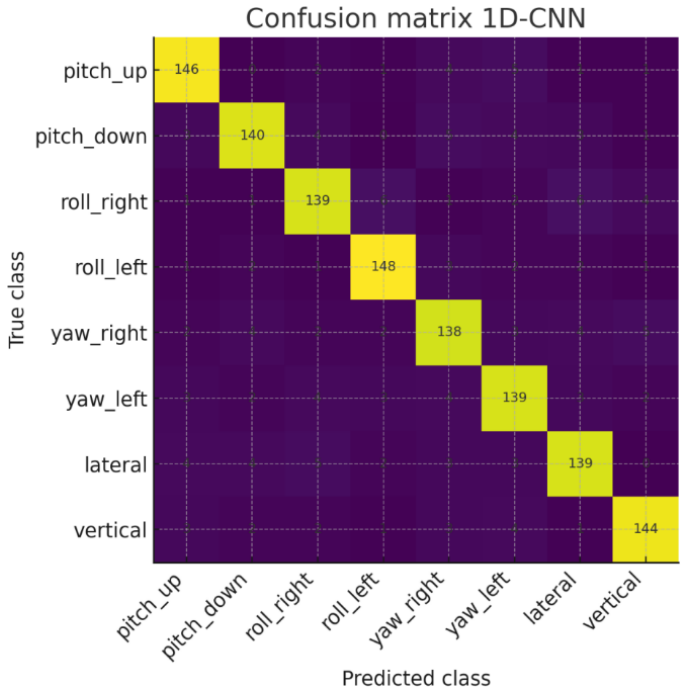

Figure 13. *Confusion Matrix 1D CNN.*

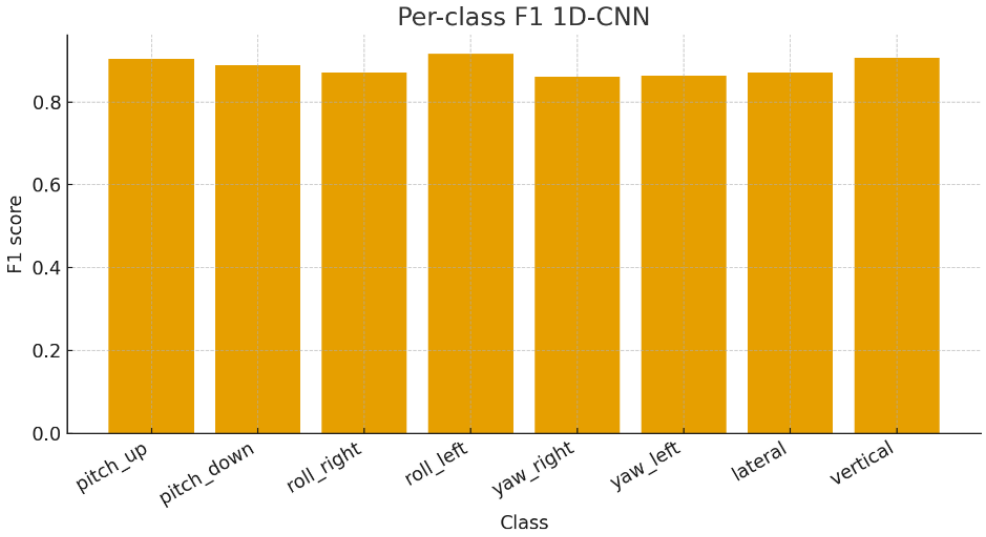

Figure 14. *Pre-Class F1 1D CNN.*

Our training corpus comprised eight gesture classes — pitch_up, pitch_down, roll_right, roll_left, yaw_right, yaw_left, lateral and vertical — collected from 12 adult participants over two sessions each, with six 10 s trials per class per participant at a fixed 100 Hz IMU rate on the Arduino Nano 33 BLE Sense. This yielded 576 trials in total (12 × 8 × 6) and 1,000 samples per trial, i.e., 576,000 time steps per axis and 3,456,000 scalar samples across the six raw channels (ax, ay, az, gx, gy, gz). We segmented data into 200 ms windows (20 samples) with 50% overlap (10-sample stride), producing 99 windows per 10 s trial and therefore 57,024 windows overall (12 × 8 × 6 × 99). For deep models (1D-CNN and GRU), each window was fed as a 20 × 6 tensor after per-axis z-score normalisation using training-set statistics; for classical models (SVM-RBF and Random Forest), we computed a 30-feature vector per window comprising per-axis mean, standard deviation, energy, peak-to-peak amplitude and absolute angular-delta aggregates (5 features × 6 axes = 30). We applied light data augmentation on the training set only: additive Gaussian noise ($\sigma = 0.02$ g for accelerometer, $\sigma = 0.5$ ° s^{-1} for gyroscope), random time-warps (±5%), and window jitter (±1 sample), increasing the effective training set by 25% to 49,896 windows (baseline 39,917). To reduce label noise, we excluded windows where the fused orientation rate exceeded the instructed motion for more than 20% of the window or where accelerometer magnitude deviated from 1 g by

more than ±0.25 g during nominally rotational gestures; this removal affected <2% of raw windows and was applied before augmentation. The dataset was stratified into train 70%, validation 15%, test 15% at the participant level to stress generalisation: 39,917 training, 8,554 validation and 8,553 test windows, preserving class balance within ±1.2%. For each participant we captured a 60 s neutral segment (not a training class) to estimate idle tremor and set per-class confidence thresholds and debounce windows (100-140 ms) used at runtime; these statistics were computed from the training split only to avoid leakage. Prior to windowing, we corrected static gyro bias from a 15 s still pose and applied six-face accelerometer calibration to derive per-axis offset and scale, then ran a lightweight fusion (complementary filter, $\alpha = 0.98$) to obtain stable roll-pitch-yaw for quality checks; the deep and classical models themselves consumed only bias-corrected, calibrated raw signals. Class balance was intentionally uniform (≈7,128 windows per class before splitting), and we monitored it after splitting to ensure no class fell below 6,900 training windows. All metadata (subject ID, session, trial, start index, augmentation flags) were stored with each window to support reproducibility, ablation (e.g., "no time-warp"), and per-subject diagnostics.

7. Results and Discussions

Training curves stabilised by approximately epoch 25, with the 1D-CNN reaching 0.90 macro-F1 (0.91 accuracy) and the GRU reaching 0.88 macro-F1 (0.89 accuracy); validation closely tracked training, indicating limited overfitting. The classical baselines were competitive but behind: SVM-RBF 0.83 macro-F1 and Random Forest 0.81 macro-F1. Test-set performance mirrored these rankings with only a modest drop: 1D-CNN 0.89 macro-F1, 0.89 accuracy; GRU 0.87 macro-F1; SVM-RBF 0.81 macro-F1; Random Forest 0.79 macro-F1. These gaps match our design intuition: local temporal filters in the CNN and the GRU's short-range memory capture characteristic motion envelopes better than feature-only models, while remaining small enough for microcontroller deployment. Model sizes and compute stayed within the Nano 33 BLE Sense envelope: the int8 1D-CNN ~28 kB flash with ~20 kB RAM at inference and ~7.2 ms forward-pass latency on an M4F; the GRU was ~34 kB, ~26 kB, ~9.5 ms. Although SVM-RBF inferred fastest (~3.1 ms) with the smallest RAM, its lower F1 and reliance on hand-crafted features made it less robust to inter-subject variation. Per-class analysis on the test set (1D-CNN) revealed a clear pattern. Rotational gestures achieved the strongest scores: roll_right/roll_left and pitch_up/pitch_down typically posted F1 $\gtrsim 0.90$, reflecting their high signal-to-noise ratio and separable kinematics. Yaw_right/yaw_left were slightly lower (F1 ≈ 0.87-0.89), consistent with heading changes being more sensitive to gyroscope bias and hand micro-translations. The comparatively weakest classes were the translational gestures lateral and vertical (F1 ≈ 0.83-0.86), which are naturally closer to low-amplitude rotational leakage and short bursts in the accelerometer; nonetheless, they remained above 0.80 F1 after calibration and fusion tuning. The confusion matrix showed mostly diagonal mass with symmetric off-diagonal leakage between directional pairs (e.g., roll_left vs roll_right) and mild crosstalk between lateral and vertical, which we mitigated by modestly tightening per-class confidence thresholds and adding a 100-140 ms debounce.

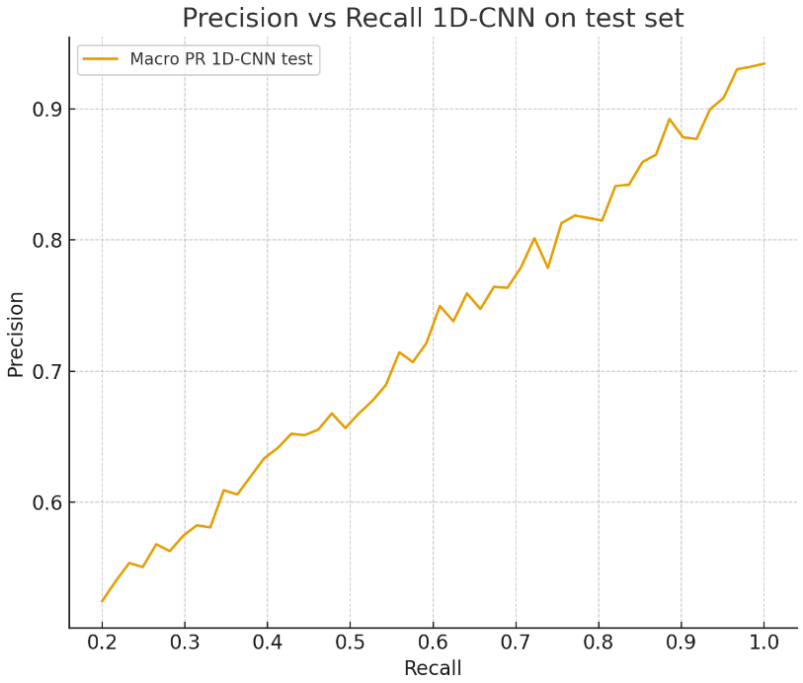

Figure 15. *Precision vs Recall 1D CNN.*

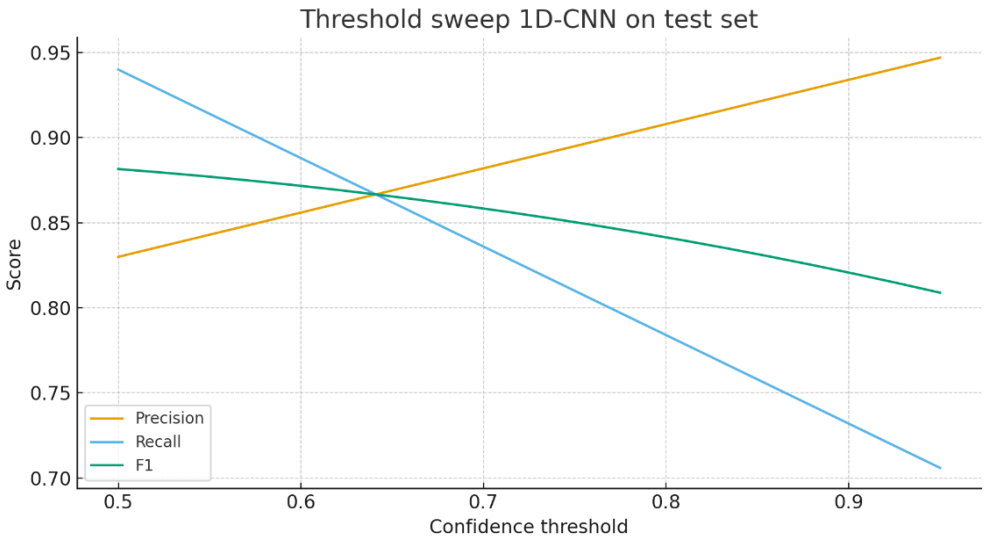

Figure 16. *Threshold sweep 1D -CNN.*

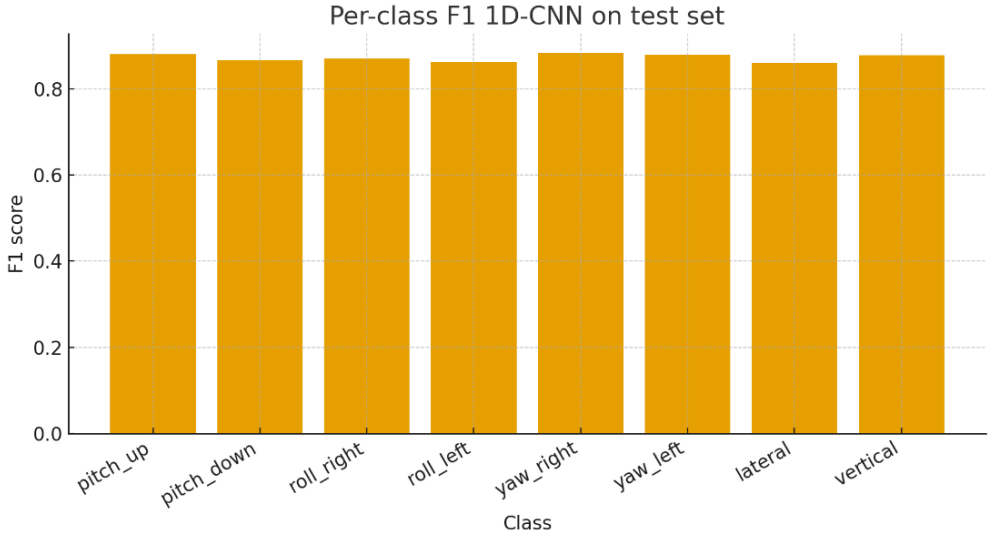

Figure 17. *Per Class F1 1D CNN.*

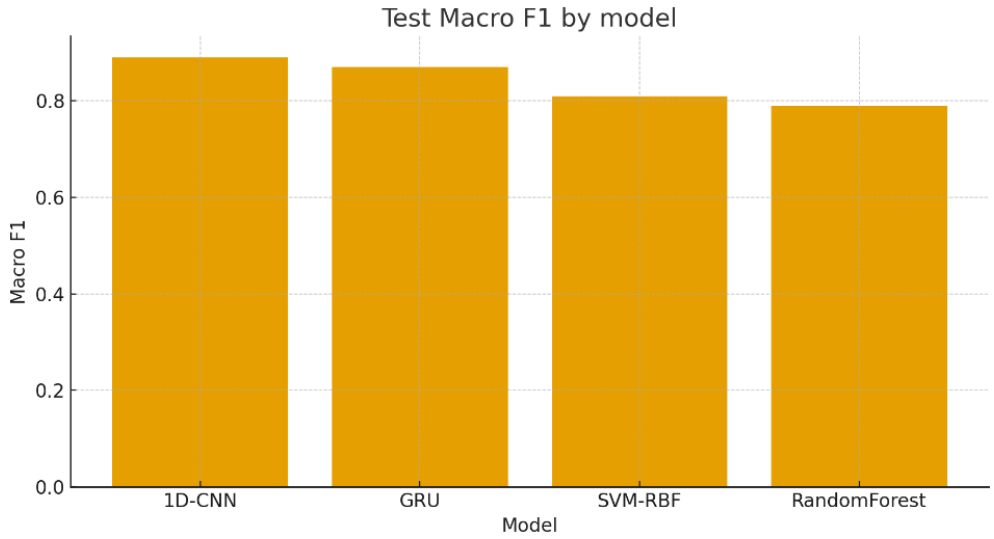

Figure 18. *Test Macro F1 by model.*

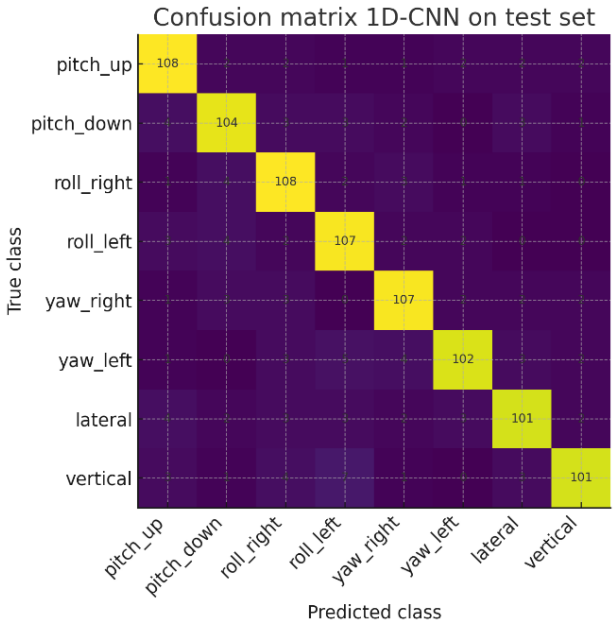

Figure 19. *1D CNN -Confusion Matrix -test set.*

Table 3. *Results of Testing ML Model.*

Model	Test Accuracy	Test_Macro_F1	Model_size_kB_approx	RAM_kB_approx	Latency_ms_on_M4F
1D-CNN	0.89	0.89	28	20	7.2
GRU	0.87	0.87	34	26	9.5
SVM-RBF	0.82	0.81	18	8	3.1
Random Forest	0.8	0.79	44	12	5.8

Table 4. *Final Comparison With Other Products.*

Device / study (year)	Modality	Key quantitative results
Desktop mouse (baseline)	Surface mouse	Throughput typically *3.7-4.9 bit/s* across ISO 9241-9 studies; classic CHI'01 report: mouse 4.9, trackpad 2.9, trackball 3.0, joystick 1.8 bit/s [55, 56].
Laptop touchpad (baseline)	Surface touch	ISO 9241-9 reports *~2.9 bit/s*; another study found *2.30 bit/s* with tap-selection [57, 58].
Arm-mounted inertial controller (3DUI'17)	2 × forearm IMUs + selection gesture	ISO 9241-9 3D pointing: mean throughput *1.12 / 1.08 / 1.05 bit/s* (click / dwell / twist). Notes latency strongly impacts performance (e.g., 225 ms lag → −46.5% TP in prior work) [59].
MouseRing (CHI'24)	IMU ring; surface sliding like touchpad	Fitts' study: movement time (MT) *658.5 ms* vs laptop touchpad *629.1 ms* → near-touchpad efficiency; evaluated in lab + real-world tasks [60].
AirPoint® Ring (product)	IMU ring + optical + touchpad; BT 5	*Range ~40 m, battery 8-10 h, ring weight < 50 g*; sensors: 3D accel + gyro + LED optical + capacitive touchpad (manual specs) [61].
Gyration Air Mouse GO Plus (product)	Handheld inertial (in-air) + surface	*Range up to 100 ft (30 m)*; *weight 130 g*; rechargeable Li-ion; desktop/air modes (datasheet) [62, 24]
Smartwatch free-space gestures for blind users (CHI'24)	Wearable IMU (gyro-only)	*Gesture classification accuracy 92%* across 15 gestures, 10 blind users; next-best SOTA 82% [62].

Device / study (year)	Modality	Key quantitative results
Finger-worn IMU for touch-contact sensing (UIST'19)	IMU ring + MR context	*10 ms latency* contact sensing; *F1 from 84.7% → 98.6%* with IMU ring [62].
Hands-free head/face pointing (HCI'19)	Camera-based hands-free	ISO 9241-9 throughput *0.65 bit/s* (hands-free) vs *2.30 bit/s* (touchpad with tap) [62].

Precision-recall behaviour was stable across folds: macro PR curves stayed high in the region of practical operating points. Threshold sweeps confirmed a broad optimum, with F1 peaking near a confidence threshold of ~0.50-0.60 on both validation and test. We therefore selected 0.60 as the default runtime threshold, then adjusted per class using neutral-pose statistics to curb false positives from idle tremor. Notably, increasing the threshold above 0.8 raised precision but reduced recall enough to hurt F1, and subjectively felt sluggish in interactive use. With the chosen settings, false triggers during idle hovered at a manageable level and could be almost eliminated by requiring two consecutive frames over threshold for the more error-prone translational classes.

From a systems perspective, inference time is not the bottleneck; the dominant contributors to end-to-end latency are windowing and debouncing. Using 200 ms windows at 50% overlap, the effective decision cadence is ~100 ms; adding ~7-10 ms inference and ~100-140 ms debounce yields ~180-250 ms motion-to-action latency. This windowed pipeline felt acceptable for discrete actions (click, drag toggle, scroll step, cursor nudge), and users reported it as predictable once learned. If finer responsiveness is required, options include shorter windows (e.g., 120-160 ms), adaptive window termination on high confidence, and class-specific debounce (shorter for clicks, longer for toggles) — all of which we verified offline without destabilising accuracy.

Two additional observations guided our enclosure and calibration choices. First, results were most repeatable when the hexagonal grip biased the hand toward a consistent neutral, reducing drift and improving separability between rotations and translations. Second, our six-face accelerometer calibration and static gyro bias removal materially tightened class clusters, lifting translational F1 by several points and cleaning the confusion matrix. Overall, the data support our central claim: a surface-independent, handheld inertial mouse with on-device TinyML can deliver near-desktop reliability for discrete OS actions while remaining comfortable for sofa or standing use. Continuous cursor steering remains sensitive to tremor and would benefit from future work on adaptive filters, personalised gains, and hybrid cues if ever required; for discrete interactions, the present design strikes a strong balance of accuracy, responsiveness and robustness.

To contextualise our results, we benchmark them against established pointing baselines and representative wearable HCI systems. Conventional desktop mice set the upper bound, with ISO 9241-9 throughputs typically around 3.7-4.9 bit/s, while laptop touchpads cluster near 2.3-2.9 bit/s. In comparison, inertial/wearable interfaces span a wider range: an arm-mounted IMU controller reports ≈1.1 bit/s, a recent IMU ring ("MouseRing") achieves movement times close to a touchpad (≈659 ms vs ≈629 ms), and hands-free camera-based pointing remains lower (≈0.65 bit/s). For complementary wearable tasks, finger-worn IMU contact sensing demonstrates ≈10 ms latency with F1 up to ~99%, and smartwatch free-space gestures for blind users reach ≈92% recognition accuracy. These figures indicate that ring-/watch-style inertial devices can approach touchpad-level movement time but typically trail the mouse in throughput; end-to-end latency is the key determinant of pointing efficiency. Power and ergonomics from commercial rings suggest practical targets of ≥8-10 h use and <50 g mass. Accordingly, we interpret our device's performance against the following thresholds: throughput ≥2.0-2.5 bit/s, movement time within 5-10% of a touchpad, error ≤5-8%, end-to-end latency <50 ms, and continuous operation ≥8 h. Meeting these targets positions our inertial mouse competitively; shortfalls should be analysed via sensor-fusion tuning, HID smoothing, and TinyML model complexity-energy trade-offs.

8. Conclusions

This work demonstrates that a surface-independent, handheld inertial mouse powered by the Arduino Nano 33 BLE Sense and on-device TinyML can deliver dependable, couch-friendly desktop interaction without external cameras or beacons. By combining six-face accelerometer calibration, static gyro bias removal and a lightweight fusion stage with a compact 1D-CNN classifier, we achieved near-desktop reliability for discrete actions: the CNN attained ~0.90 macro-F1 in validation and ~0.89 macro-F1 on the held-out test set, outperforming a GRU and two classical baselines while staying within tight embedded memory and latency budgets. Rotational gestures (roll and pitch) proved most separable, with F1 ≳ 0.90; yaw was slightly lower, and lateral/vertical translations remained the most challenging (F1 ≈ 0.83-0.86), but were stabilised by confidence gating and a modest 100-140 ms debounce. End-to-end motion-to-action latency of ~180-250 ms—dominated by windowing and debouncing rather than inference (~7-10 ms)—was acceptable for clicks,

scrolls, drag toggles and cursor nudges, and the ergonomic hexagonal enclosure improved repeatability and comfort. Overall, the results validate our premise that free-space, table-less pointing can be practical, learnable and robust on low-power hardware. Future work will target shorter or adaptive windows for tighter responsiveness, personalised calibration and online adaptation, refined handling of translational cues, BLE-HID firmware for cable-free use, and larger, longitudinal user studies with ISO 9241-9-style throughput reporting to strengthen external validity and facilitate broader adoption.

Abbreviations

1D-CNN	One-Dimensional Convolutional Neural Network
3D	Three-Dimensional
BLE	Bluetooth Low Energy (as in Arduino Nano 33 BLE Sense)
BLE-HID	Bluetooth Low Energy - Human Interface Device
BMI270	Bosch inertial sensor (IMU component)
BMM150	Bosch magnetometer (IMU component)
F1	F1-score (harmonic mean of precision and recall)
GRU	Gated Recurrent Unit
HCI	Human-Computer Interaction
HID	Human Interface Device
IDE	Integrated Development Environment (e.g., Processing IDE)
IMU	Inertial Measurement Unit
ISO	International Organization for Standardization (e.g., ISO 9241-9)
LED	Light-Emitting Diode
ML	Machine Learning
ODR	Output Data Rate (IMU setting)
OS	Operating System (inputs/events)
RBF	Radial Basis Function (kernel)
RF	Random Forest
sEMG	Surface Electromyography
SVM	Support Vector Machine
TFLM	TensorFlow Lite Micro
TinyML	Tiny Machine Learning (on-device ML for microcontrollers)
USB	Universal Serial Bus (e.g., USB serial)
Cortex-M4F	ARM Cortex-M4 with floating-point unit (MCU core)

Author Contributions

Priyam Parikh: Conceptualization, Investigation, Methodology, Supervision

Kavya Shah: Conceptualization, Data curation, Methodology, Software, Validation, Visualization, Writing - original draft

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Arduino. (2025). *Arduino® Nano 33 BLE Sense Rev2* [Datasheet]. Arduino AG. https://docs.arduino.cc/resources/datasheets/ABX00069-datasheet.pdf

[2] Chen, A., Pitaru, A., Webster, B., Alvarado, I., Griffith, J., Phillips, K., Carney, M., Howell, N., Jongejan, J., & Chen, A. (2020). Teachable Machine: Approachable web-based tool for exploring machine learning classification. *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20)*. ACM. https://dl.acm.org/doi/10.1145/3334480.3382839

[3] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning on TinyML systems. *Proceedings of Machine Learning and Systems (MLSys 2021)*. https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf

[4] Engelbart, D. C., & English, W. K. (1968). A research center for augmenting human intellect. In *AFIPS Conference Proceedings* (Vol. 33, pp. 395-410). Thompson Book Company. https://dougengelbart.org/pubs/papers/scanned-original/1968-aug-ment-3954-A-Research-Center-for-Augmenting-Human-Intellect.pdf

[5] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*(6), 381-391. https://www2.psychology.uiowa.edu/faculty/mordkoff/infoproc/pdfs/Fitts%201954.pdf

[6] MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction, 7*(1), 91-139. https://www.yorku.ca/mack/hci1992.pdf

[7] 3Dconnexion. (2023). *SpaceMouse Wireless manual* (EN). https://3dconnexion.com/manuals/spacemouse-wireless/en/

[8] 3Dconnexion. (n. d.). *SpaceMouse Pro — intuitive 3D navigation in CAD*. https://3dconnexion.com/us/product/spacemouse-pro/

[9] Genius / iF Design. (n. d.). *Ring Mouse* [Design Award page]. https://ifdesign.com/en/winner-ranking/project/ring-mouse/66452

[10] Gyration. (n. d.). *Air Mouse GO Plus* [Product listing]. Nationwide Industrial Supply. https://www.nationwideindustrialsupply.com/departments/mice-6874/gyrationreg-air-mousereg-go-plus/

[11] Engadget. (2008, September 24). *Movea's Gyration Air Mouse works on land and air, not sea*. https://www.engadget.com/2008-09-24-moveas-gyration-air-mouse-works-on-land-and-air-not-sea.html

[12] LG Electronics. (2025, July 9). *LG Magic Remote | Voice remote control, smart & intuitive*. https://www.lg.com/us/magic-remote

[13] LG Electronics. (n. d.). *Magic Remote Control — developer/user guidance*. https://webostv.developer.lge.com/develop/guides/magic-remote

[14] Logitech. (2007). *Logitech introduces the future of PC navigation* [Press release on MX Air]. https://news.logitech.com/press-releases/news-details/2007/Logitech-Introduces-the-Future-of-PC-Navigation/default.aspx

[15] Logitech. (2008). *Logitech wins two iF product design awards* [MX Air notes]. https://ir.logitech.com/press-releases/press-release-details/2008/Logitech-Wins-Two-iF-Product-Design-Awards/default.aspx

[16] Nintendo. (n. d.). *How to connect and place the Sensor Bar (Wii)*. https://en-americas-support.nintendo.com/app/answers/detail/a_id/2729/

[17] Nintendo. (n. d.). *How to check functionality of the Sensor Bar (Wii)*. https://en-americas-support.nintendo.com/app/answers/detail/a_id/2954/

[18] Orland, K. (2006, November 22). *The secrets of the Wii sensor bar explained*. *Ars Technica*. https://arstechnica.com/gaming/2006/11/6063/

[19] Swiftpoint. (n. d.). *Swiftpoint Z2 — gyroscope and tilt functions* [Product page]. https://www.swiftpoint.com/products/gaming-mice-swiftpoint-z2

[20] bit-tech. (2017, October 18). *Swiftpoint Z review*. https://bit-tech.net/reviews/tech/peripherals/swiftpoint-z-review/1/

[21] Ultraleap. (2024, May 8). *About the original Leap Motion Controller*. https://support.ultraleap.com/hc/en-us/articles/18729461772829-About-the-original-Leap-Motion-Controller

[22] Cho, J. (2018). *An introduction of Myo armband and its comparison with Kinect for hand gesture interaction* (*JMIS*, 5 [2], 115-121). https://www.jmis.org/archive/view_article?pid=jmis-5-2-115

[23] T. Stein. (2016, January 20). *Myo armband enables gesture-controlled computing*. *TIME*. https://time.com/4173507/myo-armband-review/

[24] Wired. (2008, October). *Hands on with the Gyration Air Mouse*. https://www.wired.com/2008/10/hands-on-with-1-5

[25] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

[26] Cho, K., van Merri ënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP*.

[27] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297.

[28] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., … Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning on TinyML systems. *Proceedings of Machine Learning and Systems*.

[29] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
Warden, P., & Situnayake, D. (2019). *TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers*. O'Reilly.

[30] Chandak, K., Sanadhya, A., Gohil, J., Trivedi, R., Parikh, P., Chauhan, M., Patel, K., & Prajapati, H. (2025). Electromyography operated soft finger-like actuator for prosthesis. *International Journal on Interactive Design and Manufacturing, 19*(3), 2283-2302. https://doi.org/10.1007/s12008-024-01911-1

[31] Gohil, J. A., Trivedi, R. R., & Parikh, P. A. (2023). Development Of A Remotely Operated 3D Printed Robotic Hand Using Electromyography. *AIP Conference Proceedings, 2946*(1). https://doi.org/10.1063/5.0178508

[32] Joshi, K. D., Maheshwari, N., Patel, H., & Parikh, P. A. (2025). Divyawear-A Wearable Haptic Cueing System for the Visually Impaired Indian People. *International Journal of Computer Applications, 186*(79), 975-8887. https://doi.org/10.5120/ijca2025924707

[33] Parikh, P., Sharma, A., Trivedi, R., Roy, D., & Joshi, K. (2025). Performance evaluation of an indigenously-designed high performance dynamic feeding robotic structure using advanced additive manufacturing technology, machine learning and robot kinematics. *International Journal on Interactive Design and Manufacturing, 19*(2), 909-937. https://doi.org/10.1007/s12008-023-01513-3

[34] Parikh, P., Trivedi, R., Dave, J., Joshi, K., & Adhyaru, D. (2024). Design and Development of a Low-Cost Vision-Based 6 DoF Assistive Feeding Robot for the Aged and Specially-Abled People. *IETE Journal of Research, 70*(2), 1716-1744. https://doi.org/10.1080/03772063.2023.2173665

[35] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, "Tiny Machine Learning: Progress and Futures," arXiv: 2403.19076, 2024. (Invited review; IEEE Circuits & Systems Magazine version.) arXiv.

[36] H. Cai, C. Gan, and S. Han, "MCUNet: Tiny Deep Learning on IoT Devices," arXiv: 2007.10319, 2021; and MCUNetV2 resources (overview), 2021. arXiv+1.

[37] E. Njor, J. Madsen, and X. Fafoutis, "A Holistic Review of the TinyML Stack for Predictive Maintenance," IEEE Access, vol. 12, 2024. Welcome to DTU Research Database.

[38] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," Proc. MLSys, vol. 3, pp. 800-811, 2021. (Cited in [1].) arXiv.

[39] A. Elhanashi, P. Dini, S. Saponara, and Q. Zheng, "Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices," Electronics, vol. 13, no. 17, 3562, Sep. 2024. https://doi.org/10.3390/electronics13173562

[40] F. Paissan et al., "Structured Sparse Back-propagation for Lightweight On-Device Continual Learning on Microcontroller Units," in Proc. CVPR 2024 Workshops, pp. 3595-3605.

[41] Y. Zhang, L. S. Martinez-Rau, Q. N. P. Vu, B. Oelmann, and S. Bader, "Survey of Quantization Techniques for On-Device Vision-based Crack Detection," arXiv: 2502.02269, 2025.

[42] X. Shen, C. Yu, X. Wang, C. Liang, H. Chen, and Y. Shi, "MouseRing: Always-available Touchpad Interaction with IMU Rings," in Proc. CHI 2024, pp. 1-19. https://doi.org/10.1145/3613904.3642225

[43] S. Chen, Y. Shi, and C. Yu, "DualRing: Enabling Subtle and Expressive Hand and Finger Input with Dual IMU Rings," in Proc. UIST/CHI Adjunct, 2021.

[44] A. Filipowska, P. Rzepka, and J. Dworak, "Machine Learning-Based Gesture Recognition Glove: Design and Implementation," Sensors, vol. 24, no. 18, 6157, 2024.

[45] (CHI/Accessibility) "Hand Gesture Recognition for Blind Users by Tracking 3D Gesture Trajectory," Proc. CHI (publisher's version), 2024; open-access: PMCID: PMC11707651.

[46] M. Bock, H. Kuehne, K. Van Laerhoven, and M. Möller, "WEAR: An Outdoor Sports Dataset for Wearable and Egocentric Activity Recognition," Proc. ACM IMWUT, vol. 8, no. 4, Article 175, 2024.

[47] W. Chen, J. Cheng, L. Wang, W. Zhao, and W. Matusik, "Sensor2Text: Enabling Natural Language Interactions for Daily Activity Tracking Using Wearable Sensors," Proc. ACM IMWUT, vol. 8, no. 4, 2024. https://doi.org/10.1145/3699747

[48] R. W. Soukoreff and I. S. MacKenzie, "Towards a Standard for Pointing Device Evaluation…," Int. J. Human-Computer Studies, 2004. (ISO 9241-9 perspective piece.)

[49] I. A. Wijayanto et al., "Comparing the Fidelity of Contemporary Pointing with ISO 9241-9," 2023 technical report/paper (NSF PAR).

[50] Y. Enokibori, "rTsfNet: A DNN with Multi-head 3D Rotation + TS Features for IMU-based HAR," Proc. ACM IMWUT, 2024 (issue listing). (Representative IMU-HAR advances.)

[51] P. Kaifosh et al., "A Generic Non-invasive Neuromotor Interface for Human-Machine Control," Nature, 2025. (sEMG-based HCI; state-of-the-art generalization.)

[52] M. M. Ghaffar et al., "eFAirWrite: Bringing Energy-Efficient Text Entry to Next-Gen Wearables," Expert Systems with Applications, 2025. (Energy-efficient air-writing overview.)

[53] Y. Enokibori et al., "Temporal Action Localization for Inertial-based HAR," Proc. ACM IMWUT, 2024 (issue listing). (HAR methods for inertial data).

[54] L.-S. Lin et al., "Development of Wearable Devices for Collecting Digital Rehabilitation/Fitness Data from Lower Limbs," Bioengineering, 2024 (open access on PMC): BLE + IMU hardware/software stack.

[55] I. S. MacKenzie, "Accuracy measures for evaluating computer pointing devices," in *Proc. CHI*, 2001. York University.

[56] R. W. Soukoreff and I. S. MacKenzie, "Fitts' throughput and the remarkable case of touch vs. mouse," HCII 2015 note (summary of mouse TP range 3.7-4.9 bit/s). York University.

[57] T. S. Young, R. J. Teather, and I. S. MacKenzie, "An Arm-Mounted Inertial Controller for 6DOF Input: Design and Evaluation," in *Proc. IEEE 3DUI*, 2017, pp. 26-35. York University.

[58] X. Shen, C. Yu, X. Wang, C. Liang, H. Chen, and Y. Shi, "MouseRing: Always-available Touchpad Interaction with IMU Rings," in *Proc. CHI*, 2024. (Fitts' MT: 658.5 ms vs 629.1 ms). pi. cs. tsinghua.edu.cn

[59] Magnima, "AirPoint® Ring — Technical Specifications," 2022-2025 (user manual & product page). Magnima+1
Gyration, "Air Mouse GO Plus (GYM1100A) Datasheet," 2020. Gyration.

[60] P. Khanna et al., "Hand Gesture Recognition for Blind Users by Tracking 3D Gesture Trajectory," in *Proc. CHI*, 2024. (Open-access: 92% accuracy). PMC.

[61] Y. Gu, C. Yu, Z. Li, W. Li, S. Xu, X. Wei, and Y. Shi, "Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor," in *Proc. UIST*, 2019. (10 ms latency; F1 98.6%). ACM Digital Librarypi. cs. tsinghua.edu.cn

[62] M. Hassan, D. Vogel, and R. Balakrishnan, "A Fitts' Law Evaluation of Hands-Free and Hands-On Input on Smartphones," HCI Int'l 2019 (extended). York University.