AirSense
AI-Powered Air Mouse Using **TinyML**

H.W.T.P. Pasandul

Faculty of Information Technology
University of Moratuwa
Sri Lanka
pasandul.22@uom.lk

B.H. Sudantha

Faculty of Information Technology
University of Moratuwa
Sri Lanka
sudanthabh@uom.lk

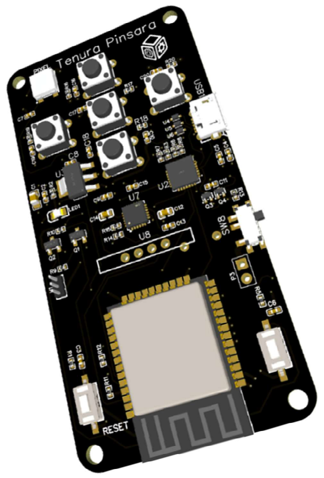
The 10th International Conference on Information Technology Research (ICITR 2025)
10th December 2025
Faculty of Information Technology
University of Moratuwa
Sri Lanka

Content

1. Introduction
2. Problem & Mortivation
3. System Architechture
4. TinyML Pipeline
5. Data Acquisition & Pre-processing
6. Neural Network Architecture
7. Model Optimization
8. Experimental Results
9. Discussion and Comparisons
10. Conclusion and further work
11. References

Introduction

AirSense is a smart, dual-mode controller that replaces traditional remotes,Air Mouses



PIXEL
AI Air Mouse
UP
LEFT
OK
RIGHT
DOWN
Mode: Air Mouse
Battery: 90%

The Problem & Motivation

The Context:
HCI (Human-Computer Interaction) is moving towards natural, contactless interfaces.

The Challenge:
Traditional gesture recognition relies on heavy Computer Vision or Cloud Computing.
Latency: Cloud dependence kills the "real-time" feel (acceptable latency is <20ms).
Privacy: Streaming sensor data to the cloud is a risk.
Energy: Constant Wi-Fi transmission drains batteries.

The Solution:
TinyML (Edge AI). Running the inference entirely on the microcontroller.

Challenge: "Unofficial" Hardware Support

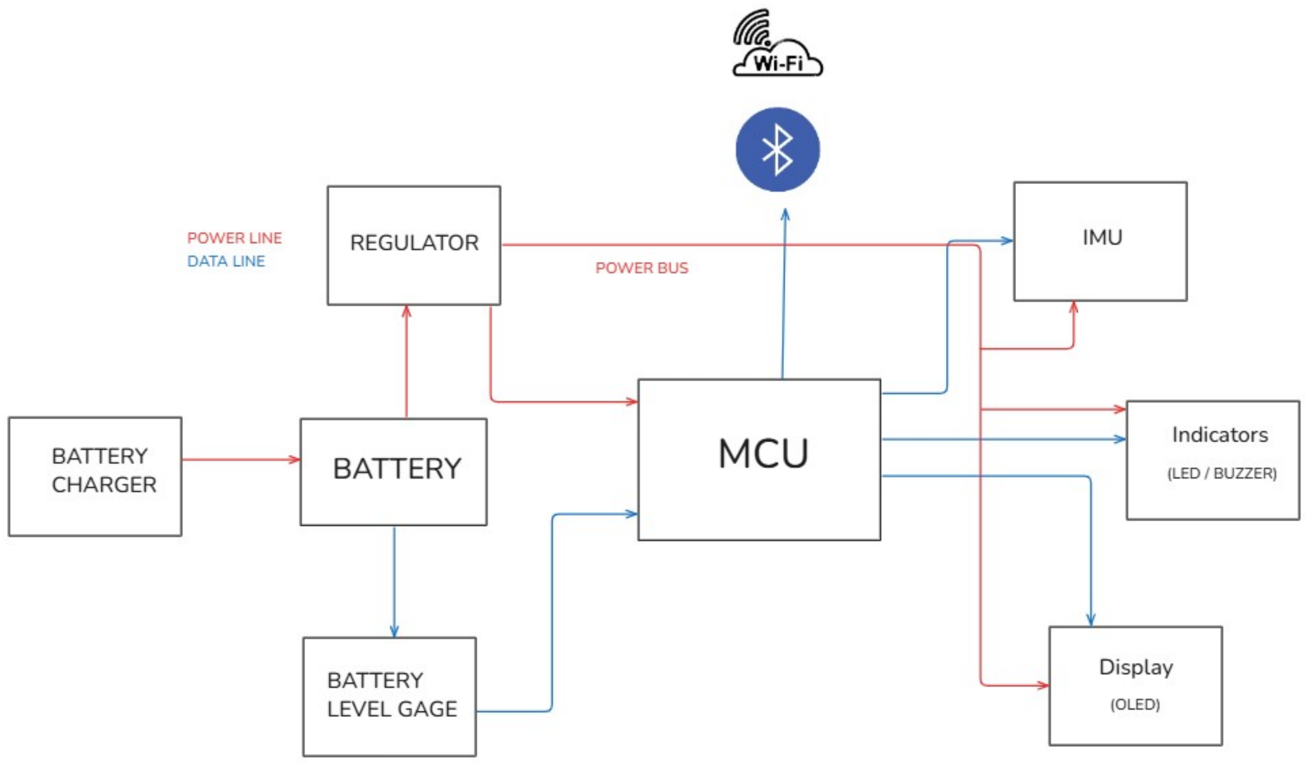
EDGE IMPULSE

The ESP32-WROOM-32D is not officially supported by Edge Impulse.

Solution: We wrote a custom Data Forwarding protocol over Serial to bridge the MPU6050 data into the ingestion engine.

This proves TinyML is accessible on custom hardware, not just dev-kits.

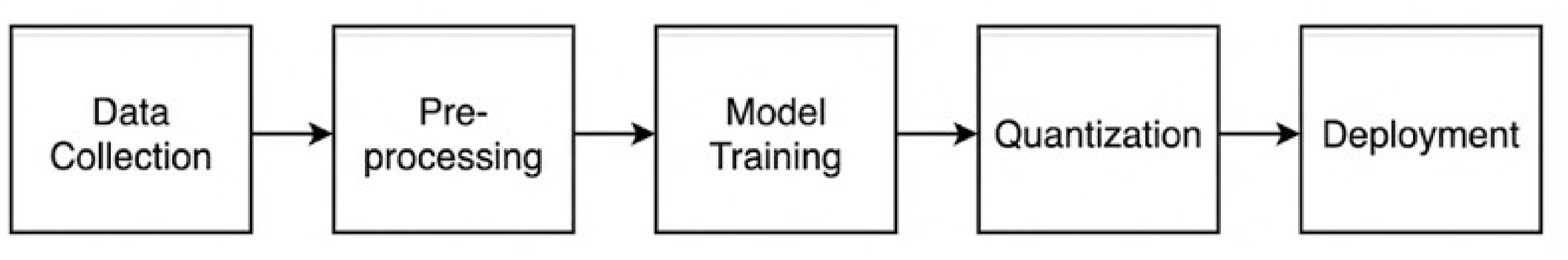
System Architecture (High Level)



MCU: ESP32-WROOM-32D (240MHz, Dual Core).
Sensor: MPU6050 (6-axis IMU).

The TinyML Pipeline

EDGE IMPULSE



The end-to-end workflow from raw sensor data to C++ inference library.

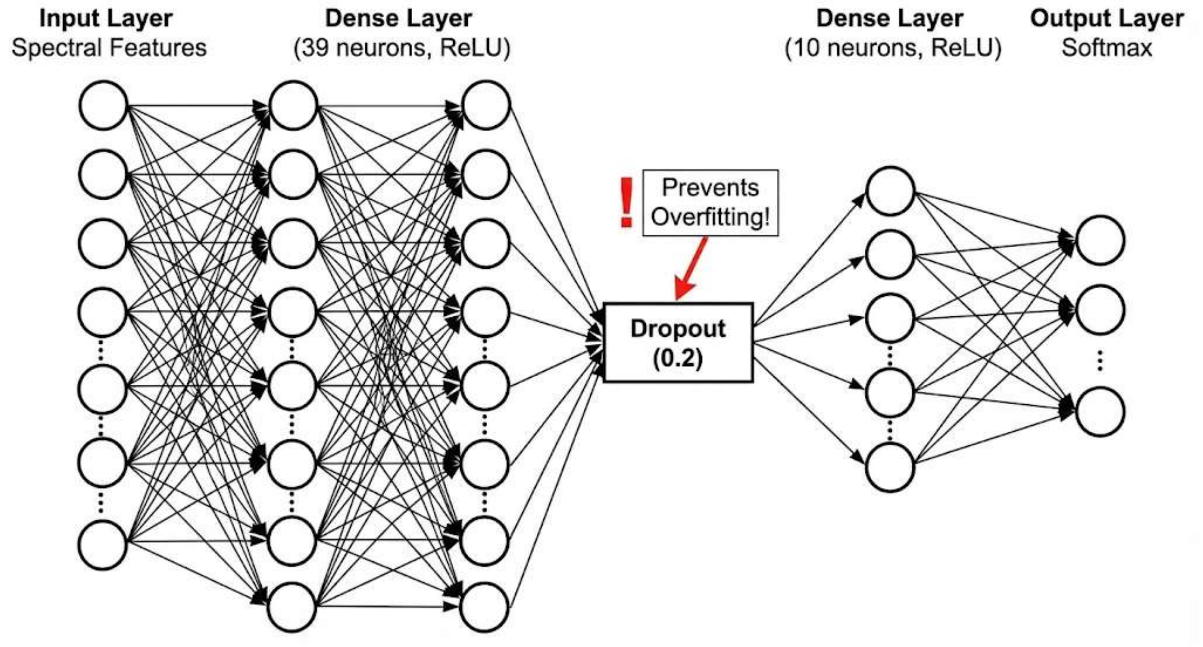
Protocol: Serial communication stream → Edge Impulse CLI → Ingestion.

Data Acquisition & Pre-processing

- Dataset: 200 Samples (Small dataset strategy).
- Classes: Volume Up, Volume Down, Play/Pause, Idle.
- Window Size: 10000ms (10 second).
- Sampling Frequency: 100Hz.
- Feature Extraction (DSP):Raw accelerometer/gyroscope data is noisy.

- We utilized Spectral Analysis to extract frequency domain features before feeding them into the Neural Network.
- This reduces the dimensionality and helps distinguish the dynamics of the gesture (e.g., the speed of the hand raise).

Neural Network Architecture

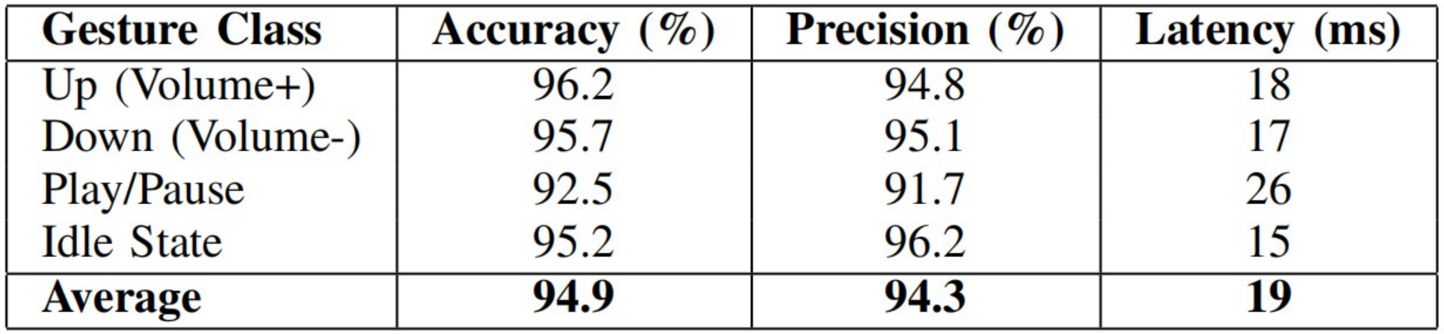


Quantization: Converted the model from Float32 to Int8.
Result: Model size reduced to just 15 KB, allowing it to fit easily into the ESP32's SRAM.
We take the raw signal from the X, Y, and Z axes and perform an FFT (Fast Fourier Transform).
For each axis, we extract the RMS (intensity), the Peak Frequency (dominant speed), and the Spectral Power across several frequency bins.

Model Optimization & Quantization

- The Constraint: Flash memory and RAM are expensive on embedded devices.
- Quantization:Converted weights from Float32 (4 bytes) to Int8 (1 byte).
- Impact: Reduced model size to ~15 KB.
- Accuracy Loss: Negligible (<1%) due to the distinct nature of the spatial gestures.

Experimental Results



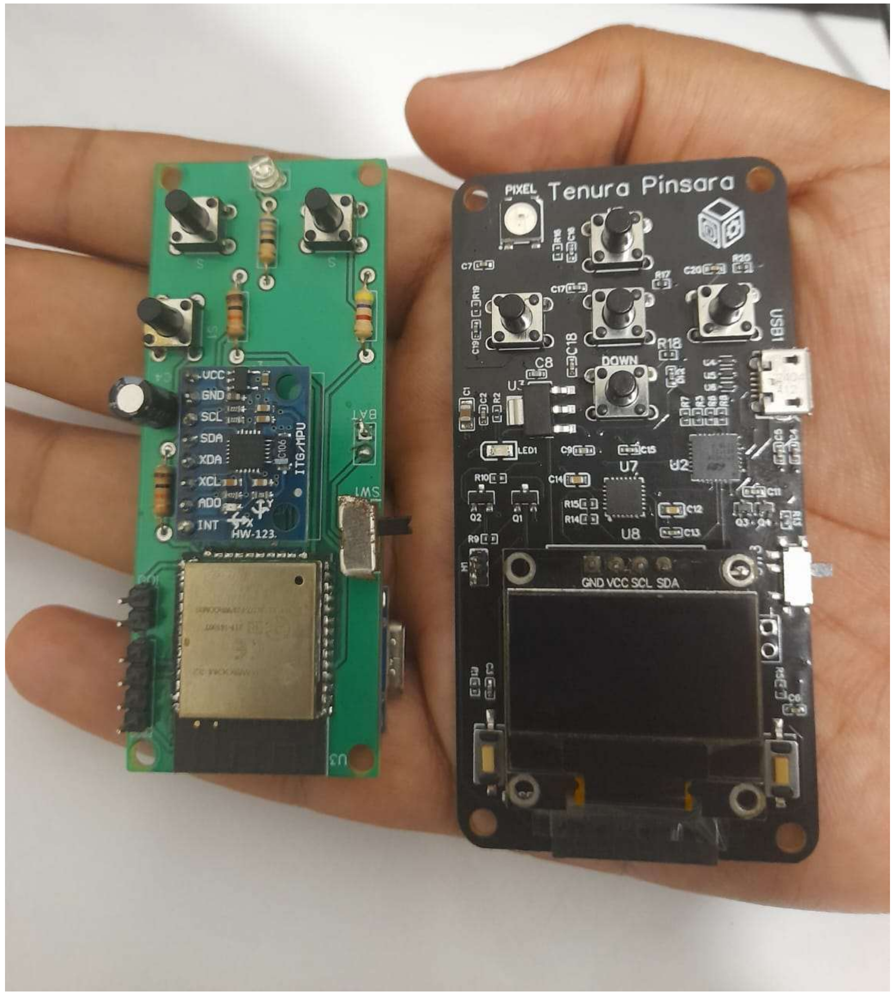
Gesture Class	**Accuracy (%)**	**Precision (%)**	**Latency (ms)**
Up (Volume+)	96.2	94.8	18
Down (Volume-)	95.7	95.1	17
Play/Pause	92.5	91.7	26
Idle State	95.2	96.2	15
Average	**94.9**	**94.3**	**19**

We achieved 94.9% accuracy.

Inference time is only 19ms, which is faster than the human perception of lag.

Up/Down gestures (crucial for volume) achieved >95% precision.

Our Implimentation



Discussion & Comparison

Existing Air Mice: Usually simple gyroscope mapping (no ML).

Existing Gesture Controllers: Often require camera systems or heavy processors.

Key Achievement: We proved that complex, context-aware interaction is possible on Low-Cost ($5 range) hardware without cloud connectivity.

Future Work & Conclusion

On-Device Learning: Implementing transfer learning on the ESP32 to allow users to record their own custom gestures.

Replace the ESP32 (General Purpose) with a custom TinyML Accelerator.

Conclusion: AirSense democratizes gesture control by optimizing the ML pipeline for the extreme edge.

References

- [1] P. Warden and D. Situnayake, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, O'Reilly Media, 2020.
- [2] V. J. Reddi et al., "Widening access to applied machine learning with TinyML," Harvard Data Science Review, Special Issue 3, 2021.
- [3] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," Proceedings of Machine Learning and Systems, vol. 3, pp. 800-811, 2021.
- [4] H. I. Fawaz et al., "Deep learning for time series classification: a review," Data Mining and Knowledge Discovery, vol. 33, no. 4, pp. 917-963, 2019.
- [5] Espressif Systems, "ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet," v3.3, 2023.
- [6] Edge Impulse, "Edge Impulse Documentation: Acquisition to Deployment," 2024. [Online]

Thank You
Q & A