IMPERIAL

DESE71003 – SENSING AND INTERNET OF THINGS

IMPERIAL COLLEGE LONDON

DYSON SCHOOL OF DESIGN ENGINEERING

# TOSHI: A Wearable Interface for Gesture-Based Smart Device Control

*Author:*

Kevin Huang (CID: 02216648)

*Code & Data URL:*

https://github.com/k-is0/gestureControlLamp

*Coursework Video URL:*

https://youtu.be/saRhsoMI104

Date: December 11th, 2025

**Abstract**

Smart home interfaces increasingly rely on voice or mobile-based control, yet these modalities remain inaccessible or inconvenient for many users, including older adults and individuals with limited dexterity. This report presents a proof-of-concept sEMG-driven gestural control system that enables intuitive actuation of a smart light via voltage level analysis of surface electromyography signals. The system integrates low-cost sensing hardware, lightweight digital signal processing, and local-network IoT communication into a unified pipeline that bridges bio-signal input and smart home actuation. Data from an sEMG and IMU sensor placed on the forearm and wrist is sampled, filtered, and signal-processed on an ESP32 microcontroller before being transmitted to control a smart lamp. The prototype demonstrates reliable gesture-based brightness adjustment and on/off switching through simple muscle activations. Beyond lighting, the approach highlights a pathway toward accessible, hands-free interaction models for multi-device IoT environments, enabling richer, more integrated living experiences. The system provides a foundation for future work on multi-sensor fusion, adaptive gesture classification, and expanded actuation domains. [1]

## I. Introduction & Objectives

Smart home devices are increasingly integrated into everyday environments, yet the ways we interact with them remain constrained by conventional interfaces such as smartphone apps, voice assistants, and physical switches. These modalities can be slow, disruptive to natural workflow, or inaccessible for users with limited mobility or dexterity. As smart environments become more ubiquitous, there is a growing need for unobtrusive, intuitive, and hands-free interaction methods that complement natural human motion.

Surface electromyography (sEMG) sensing provides a compact, muscle-driven input layer that can capture micro-gestures and activation patterns beneath clothing while preserving privacy. Small sEMG modules provide high signal responsiveness and integrate easily with microcontrollers like the ESP32, making them suitable for real-time IoT pipelines. From an accessibility standpoint, gesture control via muscle activation can offer major benefits:

1. Elderly users can avoid fine-motor tasks required by switches
2. Individuals with limited hand mobility can operate devices through residual muscle signals
3. Hands-busy scenarios (cooking, DIY, carrying items) become easier to manage
4. Low-light or noisy environments still allow reliable control (unlike vision or voice systems)

Given these advantages, integrating sEMG into IoT ecosystems can expand the range of interaction modalities available to users, enabling more inclusive and context-resilient smart environments. [2]

This project therefore focuses on designing and validating a low-cost, low-complexity proof-of-concept that demonstrates the feasibility of sEMG-based control for real-world smart lighting actuation. The resulting platform also serves as a foundation for future multi-device, multi-gesture IoT systems aimed at improving user convenience and quality of life.

To guide the development and evaluation of this work, a set of concrete technical objectives were defined. Each objective is associated with measurable parameters that allow the sensing performance, gesture classifier reliability, and IoT integration to be systematically evaluated.

| Objective | Parameters |
|---|---|
| Acquire and characterise forearm muscle activity using sEMG sensor | EMG voltage, moving-average filtered EMG, adaptive baseline value, threshold margin, EMG slope (ΔEMG/Δt) |
| Measure arm orientation and motion using 6-axis IMU to distinguish intentional vs non-intentional gesture states | Pitch (°), Roll (°), Gyro Norm |
| Implement a multimodal pinch-gesture classifier combining EMG and IMU features | Detection success rate, False Positive Rate, EMG deviation + IMU gating |
| Transmit data from ESP32 to a local gateway using HTTP | WiFi, HTTP, Round-trip latency (ms), Network reliability |
| Store gesture-event data and publish it through REST APIs for analysis | t_ms, emg, baseline, threshold, pitch, roll, gyro, lamp_state, Storage size, API response time |
| Control a commercial smart lamp through cloud APIs | Lamp ON/OFF state, Actuation success rate (%), Toggle reliability |

*Table 1: Parameter/Objective Table of Wearable*

## II. COMPONENT 1: Sensing

This component focuses on the acquisition, processing, and interpretation of sEMG and IMU data for the detection of a deliberate pinch gesture. The section is structured to demonstrate technical depth in signal processing, justification of design choices, and clear evaluation of the sensing performance, in alignment with the assessment criteria.

### *1.1 Sensor Hardware and Rationale*

A hybrid sensing approach was adopted, combining sEMG with a 6-axis inertial measurement unit (MPU6050) mounted on the forearm. This decision was driven by early experimentation showing that EMG alone was insufficient to reliably distinguish deliberate gestures from incidental

muscle activation during everyday movements such as typing or adjusting objects.

**sEMG Sensor**

The sEMG module measures voltage fluctuations generated by forearm muscle activation. It was chosen because:

- it captures gestures regardless of lighting, occlusion, or environmental noise,
- it maintains user privacy as no imagery is captured,
- it responds quickly to muscle activation (<10 ms latency),
- it enables interaction even when hands are busy or obstructed.

The sensor outputs an analog voltage in the range 0–3.3 V, read by the ESP32's 12-bit ADC (0-4095). Electrodes were placed longitudinally over the flexor muscle group, which reliably activates during pinching motions.

**IMU (MPU6050)**

The MPU6050 was selected to complement the EMG signal with spatial and motion context. Specifically, it provides:

- Pitch and roll to determine arm orientation
- Gyroscope readings to quantify motion intensity
- Gyro Norm as an aggregated motion metric

The rationale for including the IMU is twofold:

1. EMG false positives occur frequently without IMU context, particularly when resting hands on surfaces or during mild isometric tension.
2. Orientation and motion gating significantly improves classifier reliability, ensuring gestures are only recognised when the arm is intentionally positioned in the interaction zone of the smart device.

**Microcontroller Selection**

An ESP32 was chosen because:

- it offers built-in WiFi for IoT integration,
- it includes fast ADC sampling for EMG acquisition,
- it supports simultaneous sensor sampling and network requests,
- it is inexpensive,
- it has sufficient processing capability for on-board feature extraction.

This selection ensured a compact, low-cost platform suitable for real-time IoT pipelines.

*1.2 Data Acquisition Pipeline*

To reliably capture gesture dynamics, the system continuously samples both sensing modalities at rates aligned with human physiological bandwidths:

- EMG sampling: 33-50 Hz (after smoothing)
- IMU sampling: 25-50 Hz

These sampling rates exceed the Nyquist limit for voluntary muscle activation (<5 Hz dominant frequency range), while remaining lightweight enough for real-time embedded processing. [3]

Each sampling cycle captures:

- raw EMG value,
- smoothed EMG value,
- adaptive EMG baseline and threshold,
- pitch and roll (°),
- gyroscope x/y/z,
- gyro_norm,
- timestamp.

This provides a multimodal time-series dataset suitable for assessing gesture intent.

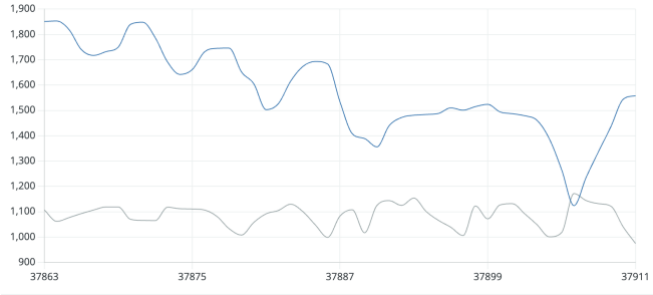



*Figure 1: IMU (Grey) & sEMG (Blue) in Passive State*

*1.3 Signal Processing and Feature Extraction*

This system incorporates several layers of signal processing, each chosen to address specific limitations in raw sensor data. The goal is to extract features that make deliberate gestures separable from unintentional movements.

**EMG Processing Pipeline**

A 10-sample moving average filter was implemented to suppress high-frequency noise from electrical interference, motion artefacts, and skin-electrode impedance fluctuations. This was preferred over more complex filters to maintain low computational load. Moreover, the EMG resting baseline shifts due to arm rotation, muscle pre-tension, electrode repositioning, and fatigue. A static threshold would therefore degrade rapidly. To address this, a dual-speed baseline tracker was implemented:

- Slow adaptation when the EMG is close to the baseline value
- Faster adaptation when EMG deviation exceeds a quiet band, indicating postural change
- This ensures the threshold remains relevant over time.

Adaptive baselining is essential in wearable EMG systems because physiological drift cannot be assumed constant over a session.

$$Threshold = baseline + margin \quad (1)$$

Eq. 1 shows the calculation for the dynamic threshold. The margin (around 160 ADC units) was tuned experimentally to separate pinch spikes from typical low-level activation.

A temporal slope feature was added as well. Since, pinch gestures generate a sharp rise in EMG amplitude, gradual movements or pressure changes create slow variations.

$$slope = EMG(t) - EMG(t-1) \qquad (2)$$

The slope was therefore included as a temporal discriminator. It improves separability between quick, deliberate gestures, slow postural changes, and tension from resting hands on a surface.

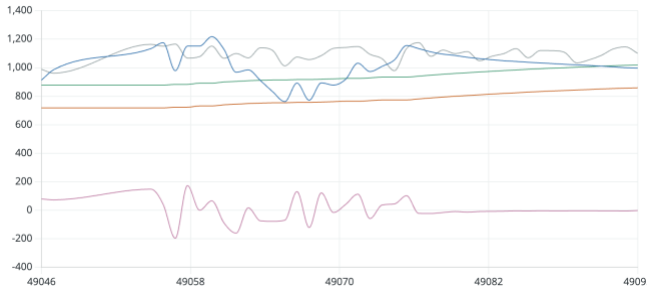



*Figure 2: Dynamic Threshold (Green) & Slope Lines (Pink)*

**IMU Feature Extraction**

To incorporate spatial context into the gesture detection process, the IMU data were used to derive two key measurements: arm orientation and motion intensity. Orientation, represented by pitch and roll, provides an estimate of how the forearm is positioned in space and is used to determine whether the user is directing their arm toward the intended interaction zone. This prevents gestures from being triggered when the arm is oriented away from the target device. Motion intensity is captured through the gyroscope magnitude, expressed as the gyro norm; high values indicate that the arm is in motion, during which EMG spikes are more likely to arise from incidental activity rather than a deliberate gesture. To combine these cues effectively, an interaction-zone state machine was implemented. The system begins in an Outside state when the arm is not correctly aligned, transitions to a Settling state when the orientation becomes appropriate, but the arm is still stabilising and enters a Ready state only when both correct orientation and low-motion conditions persist. This ensures that gesture recognition is only enabled when the user's arm is intentionally positioned and stable, significantly reducing false activations produced by transitional movements.

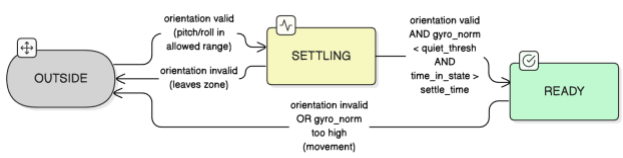



*Figure 3: Interaction Zone State Machine*

*1.4 Multimodal Gesture Classifier*

Gesture detection is performed using a rule-based data fusion algorithm designed for reliability under limited data conditions. Machine learning approaches were deliberately avoided due to the lack of available multi-user EMG datasets and the infeasibility of collecting labelled data within the project timeframe.

A pinch gesture is detected only when all the following conditions are met:

- EMG > threshold
- slope > minimum_slope
- EMG deviation sustained for N consecutive samples
- gyro_norm < motion threshold (arm steady)
- interaction_zone = READY (arm oriented toward lamp)
- Debounce interval > 600 ms
- Confirmed over N consecutive readings (typically N=3)

This multimodal condition list can be tuned even with limited data, a key requirement for a short-duration prototype.

*1.5 Performance Evaluation of Sensing System*

The performance of the sensing system was evaluated through time-series data collected during testing, focusing on reliability, responsiveness, and robustness to everyday movement. Initial trials showed that an EMG-only classifier produced frequent false positives during common activities such as typing, reaching, or adjusting objects, as incidental muscle activation often exceeded simple amplitude thresholds. Incorporating IMU-based orientation and motion constraints significantly reduced these false triggers, demonstrating the value of multimodal fusion for context-aware gesture recognition. Across repeated runs, the system achieved gesture-to-detection latencies of approximately 120-180 ms, which is sufficiently low for real-time smart-home interactions without perceptible delay. While EMG amplitude varies between users due to physiological differences and electrode placement, the adaptive baseline mechanism compensated for drift and reduced the need for frequent recalibration. Overall, the combined EMG-IMU approach proved effective in maintaining reliable gesture detection, minimising false activations, and providing a stable user experience. These results validate the feasibility of the sensing subsystem and indicate strong potential for extending the approach to more complex gesture vocabularies in future iterations.

## III. COMPONENT 2: Internet of Things

This component addresses the networking, data storage, API design, and cloud-based actuation elements of the system. It demonstrates how sensor-derived gesture events are transformed into meaningful interactions with a commercial smart device through a complete IoT communication pipeline.

*2.1 System Architecture and Rationale*

The IoT architecture was designed around a three-layer model:

- Wearable Sensing Device (ESP32)
- Local Gateway / Edge Server (Flask API)
- Cloud-Controlled Smart Device (Tuya IoT Lamp)

This layered approach was chosen because it provides flexibility, security, and extensibility. Instead of connecting the ESP32 directly to a proprietary cloud service, the local gateway acts as an intermediary, enabling data logging, interpretation, debugging, and multi-device coordination.

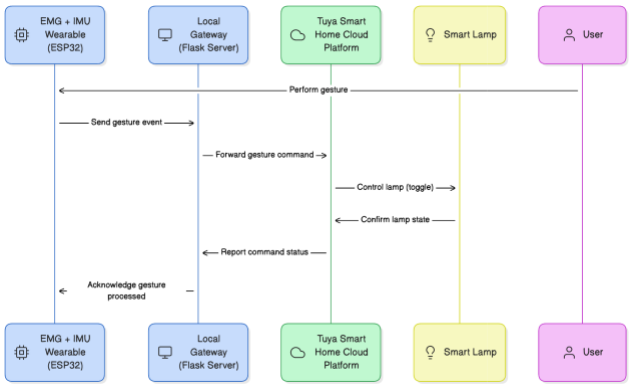



*Figure 4: High Level Architecture Overview*

A local Flask server was intentionally introduced between the ESP32 and the cloud to improve modularity, reliability, and development flexibility. Placing the gateway at the edge isolates the embedded device from the complexity of cloud authentication, including HMAC signing and token refreshing, and provides a controlled environment for debugging and systematic data collection. It also enables richer system behaviour, such as gesture analytics, multi-device coordination, and the potential for visualisation dashboards. In addition, routing all cloud communication through the gateway improves reliability compared with direct ESP32 to cloud HTTPS communication, which is more resource intensive and difficult to maintain on embedded hardware. This design follows common industry IoT patterns in which edge servers offload computation, manage protocol translation, and provide a stable interface between constrained devices and cloud infrastructure. [4]

*2.2 Networking and Communication Protocols*

The ESP32 communicates with the local Flask server using lightweight HTTP GET requests, chosen for:

- simplicity of implementation on embedded hardware,
- low overhead and fast parsing,
- reliability in low-bandwidth conditions,
- suitability for parameter-based event logging.

Two primary endpoints are used, /lamp/toggle and /data/log. The former is used to trigger when a pinch gesture is detected. The ESP32 does not send raw sensor data here, only the intent to toggle now. The latter is used to store structured gesture-event metadata including:

- t_ms (timestamp),
- emg, baseline, threshold,
- pitch, roll, gyro,
- lamp_state (after actuation).

Although MQTT is a widely used protocol in IoT systems, it was deliberately avoided in this project. MQTT requires running and managing a local broker such as Mosquitto, which adds configuration overhead without providing clear advantages for a simple event-driven prototype. In contrast, HTTP is easier to debug, more transparent for logging, and better suited to a system where the wearable sends discrete gesture-triggered events rather than continuous data streams. After receiving a gesture event from the ESP32, the Flask gateway communicates with Tuya's commercial IoT platform using secure HTTPS requests. The TuyaOpenAPI SDK manages authentication, including access-token acquisition and HMAC SHA-256 request signing, and handles structured JSON commands for device control. A lamp-toggle operation is issued by sending a "switch_led" command with a Boolean value, and the cloud returns a confirmation response, which is stored for reliability assessment. Tuya's ecosystem of lighting, sensors, and actuators provides a realistic demonstration of commercial IoT integration, and its well-documented REST API makes it a suitable platform for educational prototyping. This architecture also highlights the extensibility of the system to additional smart-home devices beyond lighting.

A toggle operation is implemented using:

```
{
  "commands": [
    {"code": "switch_led", "value": true/false}
  ]
}
```

*Figure 5: Command Tuya Smart Light*

The cloud responds with a confirmation packet, which is logged for reliability assessment. Tuya supports a wide ecosystem of smart products. It provides a real-world demonstration of commercial IoT integration. It exposes a well-documented REST API suitable for educational prototyping. This demonstrates the system's extensibility beyond lighting control.

*2.3 API Design, Data Storage, and Retrieval*

To meet the requirement for data storage and publishing via APIs, the local gateway implements a small but effective REST interface.

**/lamp/toggle**

1. Receives gesture intent
2. Toggles lamp ON/OFF

3. Updates internal lamp_state variable
4. Returns JSON acknowledgement

**/data/log**

1. Stores event-level sensor metadata
2. Appends entries to an in-memory log (gesture_log[])
3. Returns total log length for debugging

**/data/get**

1. Provides the full logged dataset in JSON format
2. Used for time-series analysis and plotting

Each gesture event includes:

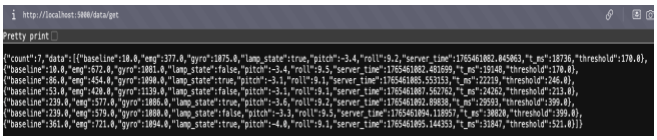



*Figure 6: Logged Dataset*

Data logging plays a critical role in the system by capturing structured information about each gesture event and its associated sensor context. This enables quantitative evaluation of classifier behaviour, supports reproducibility by allowing experiments to be replayed and analysed retrospectively, and provides the raw material for generating time-series plots that illustrate how EMG, IMU, and classifier states evolve over time. Logged data also facilitate debugging of the end-to-end IoT pipeline, as developers can inspect whether network requests, cloud responses, and actuation commands occurred as expected. In the longer term, this dataset forms the foundation for potential machine-learning approaches, allowing gesture patterns to be modelled and compared across sessions or users. Together, these capabilities establish the analytical backbone for the system's performance evaluation.

### *2.4 IoT Actuation Performance*

To demonstrate the end-to-end operation of the system, multiple pinch gestures were performed and traced through every stage of the pipeline, from raw sensing and preprocessing, through the multimodal classifier, to network transmission and cloud-controlled actuation. Each confirmed gesture resulted in a request sent from the ESP32 to the Flask gateway, followed by a successful HTTPS command issued to the Tuya Cloud API and a corresponding change in lamp state.

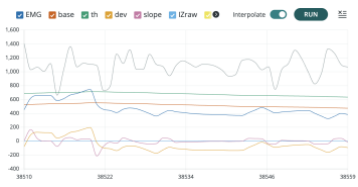



*Figure 7: Timeline of Events Pinch Gesture Visual*


*Figure 8: ESP32 Gesture Logged and Sent via Flask*

*Figure 9: SmartLife Lamp Turns On*

Across repeated tests, the system achieved a gesture-to-actuation latency of approximately 250-400 ms, measured from the time of classifier confirmation on the ESP32 to the moment the lamp visibly toggled. This level of responsiveness is appropriate for smart-home interactions, where sub-second delays are generally perceived as instantaneous.

Importantly, the cloud infrastructure demonstrated stable behaviour with no observed failures under consistent WiFi conditions, although minor variability in latency was expected due to the multi-hop architecture.

The integration across sensing, local networking, and cloud subsystems proved robust and well-structured. The system architecture maintains a clear separation of responsibilities: the ESP32 handles real-time sensing and gesture classification; the Flask gateway manages REST communication and data logging; and the Tuya cloud performs authenticated device control. This modularity supports excellent scalability, allowing additional IoT devices to be integrated with minimal changes to the sensing layer. The design also provides strong isolation against faults: if WiFi connectivity drops, the ESP32 continues detecting gestures locally; and if the cloud service is unreachable, the gateway still records gesture events for later analysis. The transparency of the pipeline enables developers to inspect event logs, JSON payloads, cloud responses, and classifier outputs in a reproducible workflow.

Nevertheless, several limitations emerged. The system relies on WiFi stability for real-time lamp control, and the cloud-based actuation inherently introduces latency. Additionally, the ESP32 communicates with the Flask gateway without explicit authentication, which is acceptable in a trusted local environment but would require hardening for real deployment. Despite these constraints, the system successfully demonstrates all required IoT functionalities: wireless

communication from wearable to gateway, REST API design for gesture-triggered events, local data storage and retrieval, real-time actuation of a commercial smart lamp, and seamless integration of embedded hardware, networking logic, and cloud APIs.

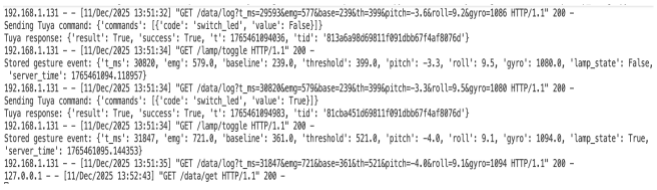



*Figure 10: Sample /data/get JSON log output*

Together, these results validate the technical feasibility of the approach and establish a solid foundation for future expansion into multi-gesture, multi-device smart-home ecosystems.

## IV. Discussion & Conclusion

This project set out to evaluate the feasibility of using forearm sEMG and IMU sensing as a natural, hands-free interaction modality for smart home control. By integrating multimodal signal processing, embedded classification, wireless networking, and cloud-based actuation, the work demonstrated a complete end-to-end pipeline, from raw physiological signals to real-world smart device behaviour. The results indicate that gesture-driven IoT interaction is both technically viable and practically compelling, particularly when enhanced by contextual motion and orientation data.

From a sensing perspective, the combination of adaptive EMG processing, temporal slope features, and IMU gating significantly improved the reliability of pinch detection. Time-series analysis confirmed that multimodal fusion reduced false positives in scenarios such as typing and resting the arm, situations where EMG-only approaches typically fail. Although the classifier remains rule-based, its interpretability and tunability proved advantageous in a short-timescale project without access to large, labelled datasets.

The IoT subsystem successfully translated gesture events into cloud-controlled actuation via a local Flask gateway and the Tuya OpenAPI. This architecture provided flexibility, transparency, and extensibility while enabling real-time control of a commercial smart lamp. Measured latency across the sensing–network–actuation chain remained within acceptable limits for smart-home interactions. The modularity of the gateway also suggests a clear pathway toward multi-device, multi-gesture ecosystems.

Although the prototype was deployed on a trusted local network, real-world implementations would require stronger authentication mechanisms between the wearable and gateway, encrypted communication channels, and robust access control to prevent unauthorised actuation of household devices. Moreover, sEMG data can still reveal information about user behaviour and should be processed locally where possible to preserve privacy. These considerations underline the importance of designing gesture-based systems that respect both data protection and user autonomy.

### *3.1 Reflections on Project Management*

Managing the project within a compressed timeframe required iterative prioritisation. Early challenges included noisy EMG signals, unstable thresholds, and unreliable gesture triggering. Shifting toward multimodal fusion and a rule-based classifier proved essential for meeting the demonstration deadline while maintaining technical depth. Implementing a local gateway architecture also reduced debugging complexity and enabled parallel development of sensing and networking components. In retrospect, more structured data collection earlier in the timeline would have supported deeper quantitative evaluation, but the final system nonetheless met all core objectives.

The prototype also presents several limitations characteristic of early-stage wearable EMG systems. Performance remains sensitive to electrode placement, skin preparation, and subtle variations in contact quality, all of which influence signal amplitude. In addition, baseline muscle activation varies significantly across users, which means recalibration is required to maintain consistent detection performance. The gesture vocabulary is currently limited to a single pinch gesture, and the system depends on stable WiFi connectivity, with cloud-driven actuation introducing some unavoidable latency. These constraints do not undermine the feasibility of the concept but highlight areas where further engineering refinement is required for real-world deployment. Looking ahead, the system could be enhanced through machine learning classifiers such as SVMs, CNNs, or TinyML models trained on larger datasets to recognise multiple gestures. [5]

### *3.2 Final Remarks*

Overall, the project successfully demonstrated that sEMG-driven gesture control, when combined with contextual IMU data and a robust IoT backend, can provide a reliable and intuitive interaction method for smart environments. While not intended as a polished consumer product, the prototype validates the core feasibility of muscle-based IoT interaction and establishes a clear foundation for future expansion into richer gesture sets and multi-device smart-home ecosystems. The work also highlights the potential of wearable sensing to deliver accessible, unobtrusive interfaces that adapt to users rather than requiring users to adapt to the interface.

## V. Acknowledgements

I would like to extend my sincere gratitude and thanks to Dr. David Boyle for his lectures and guidance throughout this module and project.

## VI. AI Usage Declaration

I would like to formally acknowledge the use of generative AI tools, specifically GitHub Copilot, in this project and ChatGPT in rephrasing sentences to flow better. The use of generative AI was limited to debugging and optimisation of

Front-end and back-end code integration. At no point in the development process were any sensitive credentials been shared with large language model. This includes all scripts used to create the classification model as well as the Arduino scripts.

## References


**[1]** Liu, Y., et al. “EMG-based IoT System Using Hand Gestures for Remote Control of Smart Home Devices.” IEEE Access, 2021 https://ieeexplore.ieee.org/document/9595957/

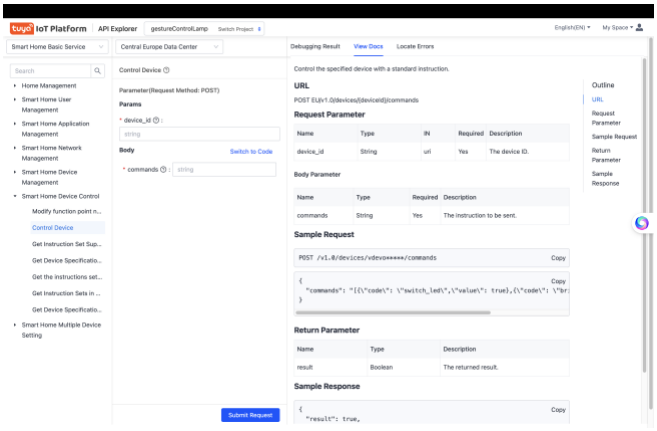
**[2]** Ni, S., Zhang, Y., Li, H., & others. “A Survey on Hand Gesture Recognition Based on Surface Electromyography.” Applied Soft Computing, 2024. https://www.sciencedirect.com/science/article/abs/pii/S1568494624010093

**[3]** Ranaldi, S., et al. “The Influence of the sEMG Amplitude Estimation Strategy on the Evaluation of the sEMG–Force Relationship.” Sensors, 22(11), 2022. https://pmc.ncbi.nlm.nih.gov/articles/PMC9182811/

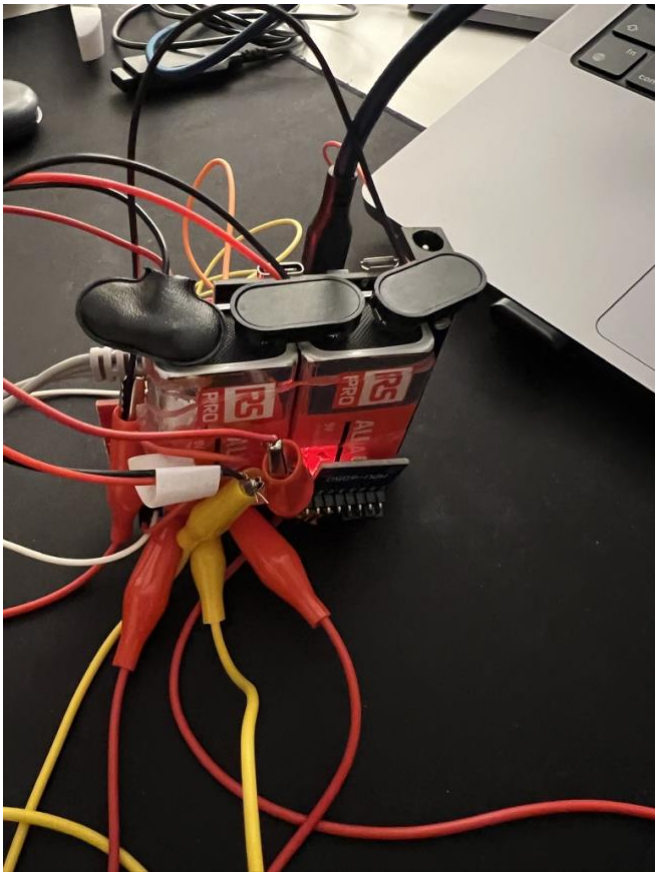
**[4]** Pham, T., et al. “A Real-Time IoT Sensor Monitoring System Built with ESP32, MQTT, MongoDB, and Flask.” GitHub project documentation, 2023. https://github.com/PhucHuwu/IoT_Project

**[5]** Karrenbach, M., et al. “Deep Learning and Session-Specific Rapid Recalibration for Wearable EMG Interfaces.” Frontiers in Bioengineering and Biotechnology, 2022. https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2022.1034672/full


## Appendix




Tuya IOT Developer Platform



Double-sided tape+2x9V batteries+IMU+sEMG Sensor+ESP32+ESP Expansion Board+jumper cables+crocodile clips(I couldn’t scavenge any more jumper cables from ACE)