

Article

Enhanced Wearable Single-Handed Control System Merging Inertial and Flex Sensors with Electrotactile Feedback

Unax Arregi * , Nicolás Medrano *  and Belén Calvo 

Aragon Institute for Engineering Research (I3A), University of Zaragoza, 50009 Zaragoza, Spain;
becalvo@unizar.es

* Correspondence: 951247@unizar.es (U.A.); nmedrano@unizar.es (N.M.); Tel.: +34-876553358 (N.M.)

Featured Application

A self-contained, single-hand wearable interface that combines continuous orientation and throttle inputs, embedded posture recognition, direct wireless communication, and spatially distributed electrotactile command signals. UAV teleoperation is used as an integration demonstrator, while the sensing, inference, communication, and feedback architecture is applicable to other remotely operated and robotic systems.

Abstract

Many wearable interface solutions use external computing systems—such as a computer—to infer actions from gestures, limiting feedback to a single vibrotactile or visual channel and rarely reporting on execution times or energy consumption. This paper presents the design and implementation of a wearable single-handed control system based on the Arduino Nano RP2040 Connect device, integrating an inertial measurement unit (IMU) using the Mahony orientation filter for precise attitude tracking and a set of piezo-resistive flex sensors for throttle mapping and gesture-based command execution. A TinyML neural network embedded on the Arduino device classifies hand gestures. The system also delivers multi-channel haptic feedback, creating tactile sensation via Transcutaneous Electrical Nerve Stimulation (TENS) electrodes. Operation with a real quadcopter shows concurrent four-axis continuous control, seven discrete commands, wireless transmission, and command-confirmation feedback. The embedded classifier achieves an accuracy of 0.935 and an F1-score of 0.934, with 1129 parameters and a mean inference time of 1.415 ms, matching results for the best reference classifier, while the complete control cycle remains below 8.031 ms, the sensing-to-transmission latency is below 33.1 ms, and the total glove consumption is 443.55 mW. These results establish a compact multimodal wearable platform whose control mappings can be adapted to other interactive applications such as the UAV teleoperation selected for validation in this work.



Academic Editors: Iwona Grobelna
and Christos Bouras

Received: 31 July 2026

Revised: 21 September 2026

Accepted: 23 September 2026

Published: 26 September 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: wearable human–machine interface; electrotactile feedback; IMU; dataglove; TinyML; gesture recognition; UAV teleoperation

1. Introduction

Human–machine interfaces (HMIs) provide a flexible input mode for controlling remote systems. Depending on the application, HMIs can take different forms, ranging from virtual reality panels for driving assistance [1], augmented reality systems combined with IoT devices for machine control (Computer Numerical Control, Magnetic Resonance Imaging) [2], brain–computer interfaces for quadcopter control [3], or more generic devices

using master–slave architectures, computers, or tablets for controlling robotic manipulators and small unmanned vehicles [4]. In these applications, the interface system is usually based on a computer, whose computing power allows for real-time data processing and classification using algorithms such as Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Markov Chains, K-Nearest Neighbor (k-NN), Random Forest, as well as Deep Learning Neural Network models like Convolutional Neural Networks, Recurrent Neural Networks or Long Short-Term Memory Networks, among others [5,6].

Unlike the controllers mentioned earlier, wearable HMIs consist of a small, user-controlled device that sends information directly to the controlled system or, more commonly, through a personal computer (PC) for processing. Specifically, smart-glove-based wearable HMIs consist of a portable interface capable of mapping the natural movement of the wrist and fingers to the corresponding control variables [7,8], being particularly appealing because they capture detailed movement information via a compact, wearable device. The inertial measurement units (IMUs) integrated into these devices can track wrist and hand orientation [8,9], while finger-mounted sensors can provide direct measurement of hand posture [10]. This combination offers a multimodal input space that can be naturally mapped to the degrees of freedom of complex systems such as pitch, roll, yaw, and throttle, in the case of an unmanned aerial vehicle (UAV), without the operator having to look away from a handheld transmitter. Additionally, robust gesture recognition can be maintained even when the number of active sensors on the gloves is reduced, helping to limit hardware complexity and cost [10].

Beyond continuous motion capture, a fully wearable interface must be able to identify and differentiate discrete commands from postural transitions and incidental movements. Classical classifiers and neural networks have been applied to gesture recognition in wearable and vision-based devices [8,11], using computers to deduce between interface signals and commands to the controlled system. A fully autonomous interface requires efficient and computationally compact inference algorithms, which rules out many of those presented in the literature. In this scenario, the application of TinyML to drastically reduce the complexity and size of neural models allows this inference to be executed directly on the wearable device microcontroller, avoiding the connection to a mainframe or to the cloud and providing a fixed local processing path.

One of the issues of wearable control systems that is typically overlooked is the sensory feedback to the user. For instance, teleoperation is fundamentally a closed-loop process: the user must continuously receive information about the state of the controlled system and the effects of its actuation in order to effectively modulate their commands. Visual feedback through a first-person-view (FPV) display addresses this, but requires additional devices and hardware such as mobile phones, laptops and other screen-based devices. Alternatively, haptic feedback systems, and particularly, those based on TENS, offer a non-invasive means of delivering tactile and proprioceptive signals directly to the skin surface [12–14] without the need for additional mechanical systems, as is the case with less compact and more energy-intensive vibrotactile stimulation [15,16]. In fact, TENS-based electrotactile stimulation has been shown to transmit graded intensity information, directional signals, and event notifications in various HMI applications [17], but its integration into some innovative applications such as UAV control gloves remains largely unexplored.

Reviewing the literature, existing human–UAV interfaces include different approaches such as wearable sensing, embedded gesture devices, immersive controllers, and camera-based recognition. Lee and Yu [8] combine a hand-mounted IMU, direct and gesture control modes, machine-learning recognition, and wrist vibrotactile feedback. Shin et al. [9] implement a Raspberry Pi Zero wearable interface that converts hand-pose gestures and

gesture sequences into flight commands. VR controllers provide natural motion mapping for UAV and UAV-manipulator teleoperation [18], while RGB-camera methods support large motion/posture command sets without a worn sensing glove [19]. Table 1 compares the main structural characteristics of these works with those in the proposed interface. Our proposal integrates the control of a device by combining commands from a manual gesture system based on flexible sensors and integrated posture inference with wireless command transmission and multi-channel electrotactile confirmation in a single portable microcontroller, comprising a simpler, low cost, but highly efficient functional solution.

Table 1. Architectural comparison of representative human–UAV interaction approaches.

System	Operator Input	Processing and Communication	Operator Feedback
Lee and Yu [8]	Hand-mounted IMU; direct and gesture modes	Machine-learning gesture recognition and drone command link	Wrist vibrotactile obstacle-distance cue
Shin et al. [9]	Wearable hand-pose gestures and gesture sequences	Raspberry Pi Zero gesture-to-command interface	Visual observation of vehicle response; no tactile channel
Yang et al. [18]	Dual VR controllers and natural hand motion	VR/host framework for UAV and manipulator commands	Immersive visual interaction; no wearable tactile channel
Yun et al. [19]	Egocentric RGB-camera motion and posture gestures	Lightweight vision recognition for mobile/edge execution	Visual command interface; no wearable tactile channel
This work	IMU orientation and four flex sensors	RP2040 fusion, MLP inference, and direct UDP transmission	Ten-route electrotactile command identification confirmation

Moreover, UAV applications provide a demanding benchmark for responsive interaction because stabilization, command transmission, and safety management must operate concurrently [20–22]. Optimization-based controllers offer a complementary safety layer. For example, the acceleration-level hierarchical quadratic programming framework of Xing et al. [23] unifies trajectory tracking, physical constraints, manipulability, and obstacle avoidance under task priorities. Due to the complexity of these algorithms, the control system relies on an industrial computer in charge of performing the calculations and sending them to the controlled system. Moreover, these devices receive references from the portable interface while retaining the ability to modify commands that might violate platform restrictions, which requires independent processing capabilities. Thus, to develop a control system capable of continuously managing and identifying hand orientation, finger flexion-based acceleration control, integrated discrete inference, direct wireless communication, and localized electrotactile confirmation, all operating simultaneously in a battery-powered handheld device without additional computing resources via a low-cost microcontroller, it is necessary to reduce the complexity of the required algorithms and memory resources so that all can be executed in real time.

This paper presents the design, implementation, and experimental validation of a wearable haptic in-feedback control system built around an Arduino Nano RP2040 Connect as a portable low-cost processing platform. Our specific contributions are as follows: (i) a self-contained multimodal wearable architecture that integrates inertial and flex sensing, embedded gesture recognition, wireless command transmission, and ten selectable electrotactile routes on one microcontroller; this solution eliminates the need for an external computer required by other wearable gloves, while keeping the entire device within a measured power budget of 443.55 mW; (ii) a dual-channel interaction scheme combining proportional hand-motion control with temporally confirmed discrete commands, which bounds continuous sensing-to-transmission latency to 33.1 ms and discrete command delivery to 455 ms, while deterministically rejecting transient and repeated activations;

(iii) quantitative characterization of classifier performance, stimulation electronics, and multi-rate embedded execution, yielding an accuracy of 0.935 and an F1-score of 0.934 on the hold-out set, equal to the performance of the best reference classifier (Radial Basis Function Kernel Support Vector Machine, RBF-SVM) and 3.3 percentage points above the threshold-tree and random-forest baselines, at a 1.415 ms mean inference time and a worst-case 8.031 ms control cycle, below the 9.62 ms IMU sampling period; and (iv) a real-UAV demonstrator prototype showing simultaneous operation of sensing, inference, communication, control mapping, and haptic command confirmation, managing four proportional axes, seven discrete actions and ten electrotactile routes concurrently on a single RP2040. The platform-level organization and measurements provide a reproducible basis for adapting the interface to other remote-control applications. To the authors' best knowledge, this work constitutes the first reported autonomous wearable prototype including this innovative approach.

2. Materials and Methods

2.1. System Overview

The proposed handheld interface consists of several interconnected modules: a multimode sensor measurement acquisition module; a module for acquiring, interpreting, and wirelessly transmitting control commands; a hardware–software block for electrotactile command feedback; and an integrated centralized controller. Figure 1 illustrates these blocks, displaying the data and information flow from the hand kinematics configuration and transmission of the output command packet, as well as from the accepted controller status to the operator. Note that the command-referenced feedback path is platform-independent and does not require vendor-specific vehicle telemetry.

In the selected application example, the control system translates physical hand movements into executable commands through sequential data processing. This process begins with a synchronous data acquisition at the operator interface layer via a six-axis IMU and a piezo-resistive flex sensor array embedded in a dataglove (Figure 2). The raw angular velocity, linear acceleration, and the analog output voltages from the flex sensors are acquired by an onboard Arduino Nano RP2040 Connect microcontroller (Arduino S.r.l., Monza, Italy). These data are used to identify, in parallel, two different types of commands:

1. Continuous Trajectory Control: A Mahony orientation filter [24] processes the telemetry information provided by the IMU module to compute pitch, roll and yaw angles, while the voltages provided by the flex sensors on the index and middle fingers are algorithmically mapped to continuously establish a throttle value.
2. Discrete Command Inference: The voltages acquired from the ring and little fingers feed an edge-deployed TinyML neural network classifier, mapping these inputs into a set of high-level symbolic flight-state commands, such as automated takeoff or landing.

The continuous variables and discrete commands are packed into an RF frame and sent to the platform under control. In parallel, according to the acknowledged command, the microcontroller sends the corresponding stimulus feedback by means of the electrotactile subsystem, thus providing to the operator an immediate indication of the command identified by the wearable interface.

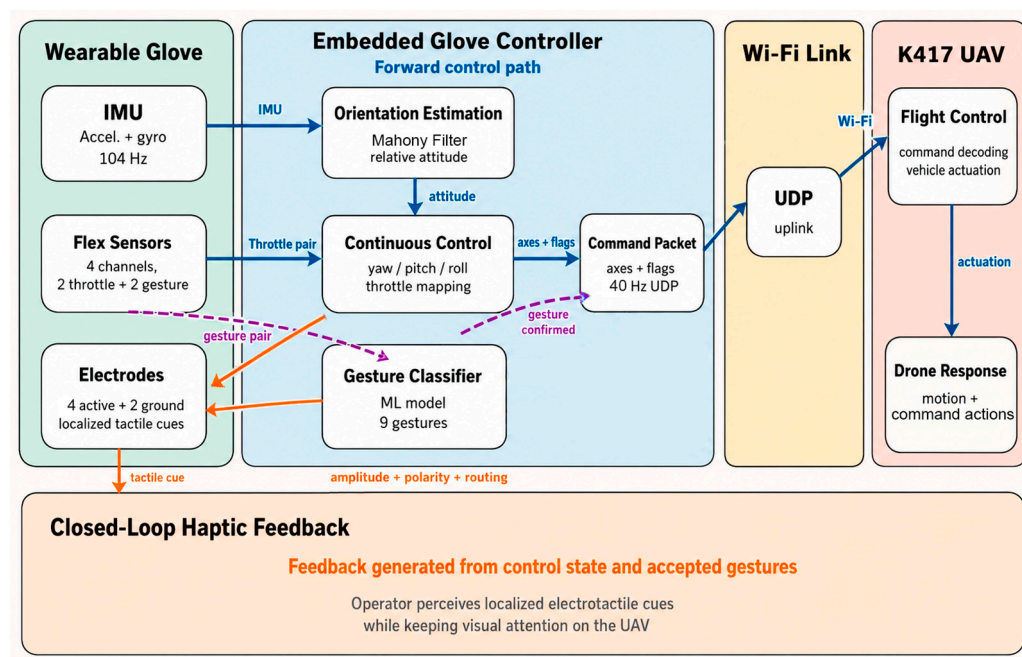


Figure 1. Block diagram of the proposed wearable teleoperation architecture. The figure separates the wearable sensing interface, embedded glove controller, wireless communication layer, and UAV execution stage, while also showing the reverse electrotactile feedback loop (orange arrows) used to return system-state cues to the operator.

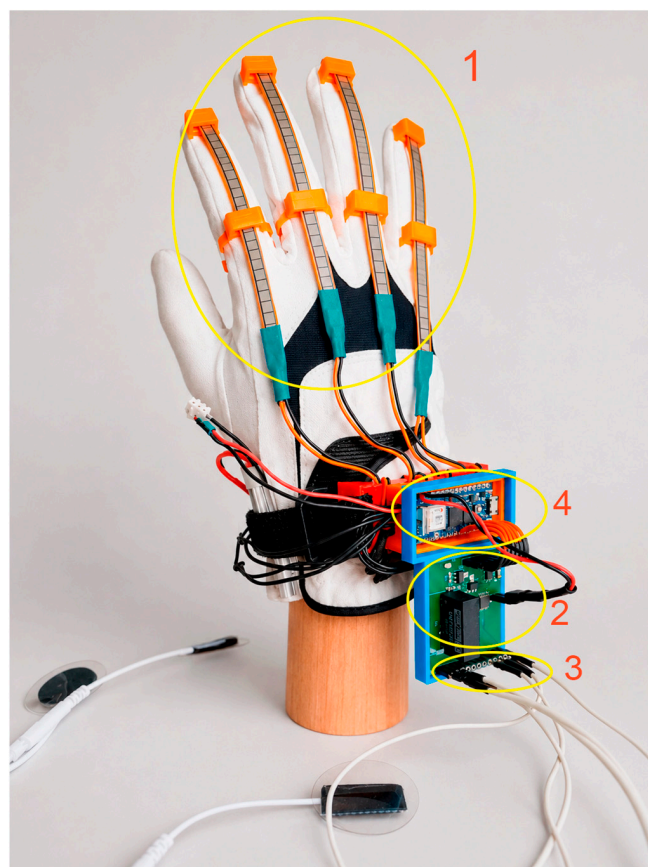


Figure 2. Assembled glove prototype. Flex sensors (1) are attached to the index, middle, ring, and little fingers; the stimulation circuit (2) and microcontroller module (4) are located on a wristband; the Arduino device is housed in a wrist-mounted enclosure; electrode connectors (3) are visible on the underside of the PCB.

2.2. Hardware Design

The complete system is assembled onto a single glove. It integrates four finger-mounted flex sensors, a microcontroller module including an IMU and an RF module, the electrotactile stimulation circuit, and the excitation electrodes, connected to its control circuit through an analog switch controlled by the microcontroller. The system presents a compact form factor that preserves natural hand dexterity. Figure 2 shows the main prototype components: (1) piezoresistive strips for measuring finger flexion, (2) electrotactile stimulation circuitry, (3) connectors for the electrodes, and (4) the main module that includes a six-axis measurement system, a radio frequency transmission system, and the microcontroller responsible for measurement acquisition, decision making, and haptic feedback.

2.2.1. Microcontroller: Arduino Nano RP2040 Connect

The central processing unit is an Arduino Nano RP2040 Connect [25], which hosts a Raspberry Pi RP2040 dual-core ARM Cortex-M0+ microcontroller running at 133 MHz with 264 kB of SRAM. The module operates at 3.3 V and provides four external 12-bit ADC channels. In this implementation, the Arduino digitizes the voltage values provided by the four flex-sensors, with a resolution of 10 bits (0–3.3 V range, ~3.2 mV voltage resolution, acquisition time < 30 μ s). Wireless connectivity is provided by an on-board U-blox NINA-W102 module [26], which integrates a dedicated dual-core ESP32 processor. This processor fully offloads RF management from the RP2040, preserving its computational time for IMU fusion, neural network inference and haptic control. The NINA-W102 supports both Wi-Fi (802.11 b/g/n at 2.4 GHz) and Bluetooth Low Energy (BLE); Wi-Fi was selected for this application due to its compatibility with the target drone access point protocol and the bandwidth.

For glove orientation, the Arduino module includes a STMicroelectronics LSM6DSOX six-axis IMU [27] (STMicroelectronics, Amsterdam, The Netherlands). The sensor contains a three-axis accelerometer and a three-axis gyroscope. The Arduino_LSM6DSOX library [28] configures the accelerometer for a ± 4 g range with a 0.122 mg resolution and the gyroscope for a ± 2000 dps range with a 70 mdps resolution, both with a default reading frequency of 104 Hz. The IMU data acquisition and quaternion update are adjusted to the IMU readout frequency, while the orientation commands are sent at a frequency of 40 Hz, a value imposed by the controlled device used in our application example.

The platform was selected after comparing embedded targets able to host, in a single wearable unit, the orientation filter, the TinyML classifier, the stimulation control, and the wireless link. Table 2 summarizes the candidates considered and the reasons for them to be discarded. Linux-class boards, such as the Raspberry Pi Zero used in similar gloves [9], offer high computing power, but provide no analog inputs, with an order of magnitude more power consumption, and introduce non-deterministic scheduling that conflicts with the fixed inter-sample window shown in Section 3.4. Cortex-M4F boards, such as the Arduino Nano 33 BLE Sense, add a hardware floating point, but provide only low-energy Bluetooth (BLE) radiofrequency connectivity. ESP32-S3 modules integrate both Wi-Fi and a floating-point unit, but their SAR ADC is markedly non-linear near the supply rails, which degrades the flex-sensor divider readings, and the radio stack shares the same core as that for the application. Higher-performance devices such as the Teensy 4.0 exceed the required computing power while integrating neither radio nor IMU, increasing board area, cost, and complexity regarding the connections to other components required for a handheld device.

The Arduino Nano RP2040 Connect was therefore selected because as the best candidate, combining, within a 45×18 mm module, an on-board six-axis IMU, a Wi-Fi/BLE radio module with an independent co-processor, suitable ADC channels, 264 kB of SRAM,

and a single 3.3 V supply, exhibiting compatibility with the rest of the electronics. Its main drawback is the absence of a floating-point unit in the Cortex-M0+ cores, so the quaternion update is executed with software-emulated floating-point arithmetic; the 3.494 ms measured for IMU reading and conditioning confirms that this time remains compatible with the update rate of the unit, while the RF offload to the NINA-W102 preserves the remaining computational headroom.

Table 2. Comparison of the candidate embedded platforms considered for the wearable controller.

Platform	Core/Clock	SRAM	On-Board IMU/Radio	Analog Inputs	Main Limitation for This Application
Arduino Nano RP2040 Connect (this work)	Dual Cortex-M0+, 133 MHz, no FPU	264 kB	LSM6DSOX/ Wi-Fi + BLE (NINA-W102 co-processor)	$4 \times 12\text{-bit}$	No hardware floating point; the orientation filter runs on emulated floating-point arithmetic
Arduino Nano 33 BLE Sense	Cortex-M4F, 64 MHz, FPU	256 kB	Yes/BLE only	$8 \times 12\text{-bit}$	No Wi-Fi; incompatible with the UDP access point of the target UAV
ESP32-S3 module	Dual Xtensa LX7, 240 MHz, FPU	512 kB	No/Wi-Fi + BLE	$20 \times 12\text{-bit}$	Non-linear SAR ADC near the rails; radio stack shares the application cores; external IMU required
Teensy 4.0 (i.MX RT1062)	Cortex-M7, 600 MHz, FPU	1 MB	No/No	$14 \times 12\text{-bit}$	Neither radio nor IMU integrated; added modules increase area, cost, and wiring
Raspberry Pi Zero 2 W [9]	Quad Cortex-A53, 1 GHz, Linux	512 MB	No/Wi-Fi + BLE	None	No ADC; watt-class consumption; non-deterministic scheduling breaks fixed-cycle encapsulation

2.2.2. Flex Sensor Array

Finger flexion is measured using four Spectra Symbol Flex Sensors [29], attached to each of the index, middle, ring, and little fingers via small retaining rings, as shown in Figure 2. Each sensor exhibits a resistance of approximately 7–13 k Ω at rest, which increases to at least double this value under full flexion. Each sensor is embedded into a voltage divider formed by a fixed resistor R_{fixed} , in series with the piezoresistive element R_{piezo} , biased at a 3.3 V supply voltage, so that the output voltage of the resistive divider is

$$V_{out} = 3.3 \cdot \frac{R_{fixed}}{R_{fixed} + R_{piezo}}, \text{ [V]} \quad (1)$$

where V_{out} is the voltage provided to the analog input of the microcontroller, R_{fixed} is the fixed resistor of the divider, R_{piezo} is the flexion-dependent resistance of the piezo-resistive element, and 3.3 V is the regulated supply voltage that biases the divider.

The fixed resistor for each sensor is chosen as the geometric mean of its measured minimum and maximum resistances, $R_{fixed} = \sqrt{R_{max} \cdot R_{min}}$, in order to maximize the dynamic range of the output voltage. The resulting analog voltage is sampled by the RP2040's ADC at a 10-bit resolution, yielding a digital value in the range of [0, 1023].

Two of the flex sensors (index and middle fingers) are reserved for continuous throttle control (Section 2.4), while the remaining two provide the two-dimensional input vector

for the neural network gesture classifier. This partition reduces interference between continuous and discrete control channels, while keeping the wearable layout simple.

2.2.3. Electrotactile Stimulation Circuit

The electrotactile stimulation subsystem is designed to generate referred tactile sensations on the palmar surface of the hand from electrodes placed exclusively on the dorsum, following the approach of Tanaka et al. [6]. Achieving selective stimulation of ten independent palmar zones requires the circuit to (i) supply milliampere-range currents at sufficient compliance voltage to overcome skin impedance ($\sim 1\text{ M}\Omega$ in dry contact); (ii) multiplex the stimulus across ten output channels, plus two independent ground electrodes (located on the median and the ulnar nerves); and (iii) switch stimulation polarity per channel, since anodic stimulation is effective for most zones, but cathodic stimulation is required for the distal electrodes [6]. For this reason, the hardware must provide both channel selection and polarity inversion under firmware control, so that each anatomical zone can later be calibrated with the most reliable configuration yielding the perceptual response.

Figure 3 shows the custom implemented circuit schematic, which is powered by a 7.4 V lithium polymer (LiPo) battery (1000 mAh) connected to the Arduino Vin pin. A voltage regulator connected directly to the Arduino Vin pin then provides the stable 5 V required by the stimulation circuitry. The high compliance voltage of 72 V required for cutaneous current delivery is generated by a NMT0572SC isolated DC–DC converter (Murata Manufacturing Co., Kyoto, Japan), also powered by this 5 V rail.

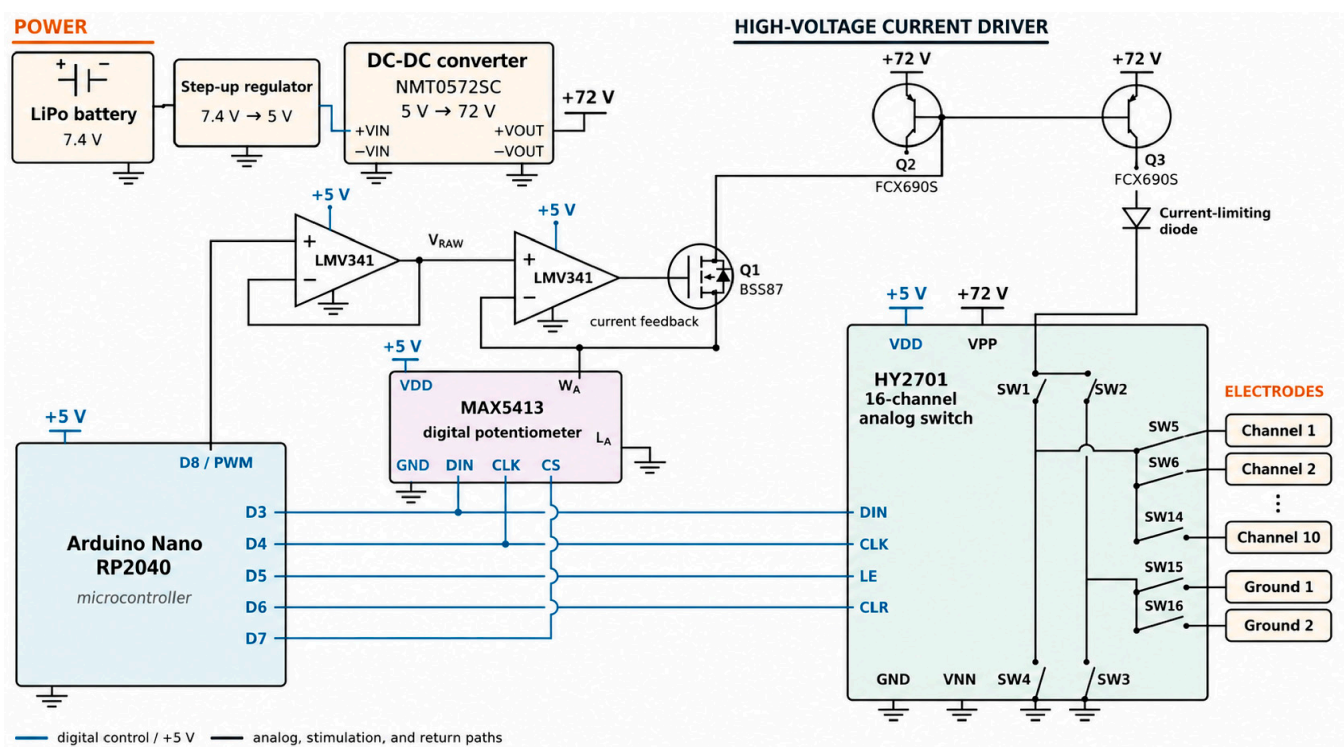


Figure 3. Schematic of the electrotactile stimulation circuit. Key blocks from left to right: power input and regulation, current-source stage (MAX5413 digital potentiometer, LMV341 op-amps, BSS87 FET transistors, FCX705 BJT transistors in current mirror configuration), HV2701 high-voltage multiplexer with an H-bridge for polarity switching, and a safety diode clamp.

Differently from the method used in [6], current intensity is controlled by a MAX5413 digital potentiometer (0–10 k Ω , 256 steps) (Analog Devices, Wilmington, MA, USA), configured as a variable resistor R_{WA-LA} , whose programming word is sent through Serial Peripheral Interface (SPI) protocol from the microcontroller, which also provides the stimuli

signals— V_{RAW} after a LMV341 (Texas Instruments, Dallas, TX, USA) voltage follower configuration—detailed in Section 2.6. The voltage-controlled current source, composed of an LMV341 operational amplifier with a BSS87 MOSFET (Nexperia, Nijmegen, Netherlands) feedback transistor at its output, provides the suitable controllable current V_{RAW}/R_{WA-LA} to a pair of FCX705 BJT transistors (Diodes, Inc., Plano, TX, USA) in a current mirror configuration with a 72 V supply voltage, which then deliver the current to the stimulation electrodes through a current-limiting diode.

The relationship between the digital word $P \in [0, 255]$ and the output current follows a hyperbolic law $I \propto 1/P$, linearized via $y = 1/I = aP + b$. This firmware-level linearization allows the rest of the control logic to operate directly in milliampere units, decoupled from the analog circuit details. Before the haptic subsystem can be used as a closed-loop feedback channel, the relationship between the digital potentiometer setting and the current stimulus actually delivered by the circuit must be experimentally characterized. For this, a load $R_L = 1 \text{ k}\Omega$ is connected at the female headers by replacing the electrodes, measuring the output voltage V_L for different MAX5413 digital potentiometer values. The current is then obtained as $I = V_L/R_L$ (Figure 4). The measured response confirms the expected hyperbolic trend $I \propto 1/P$. The current-limiting diode begins to clamp the output at approximately 3.8 mA, which is adequate to provide tactile feedback.

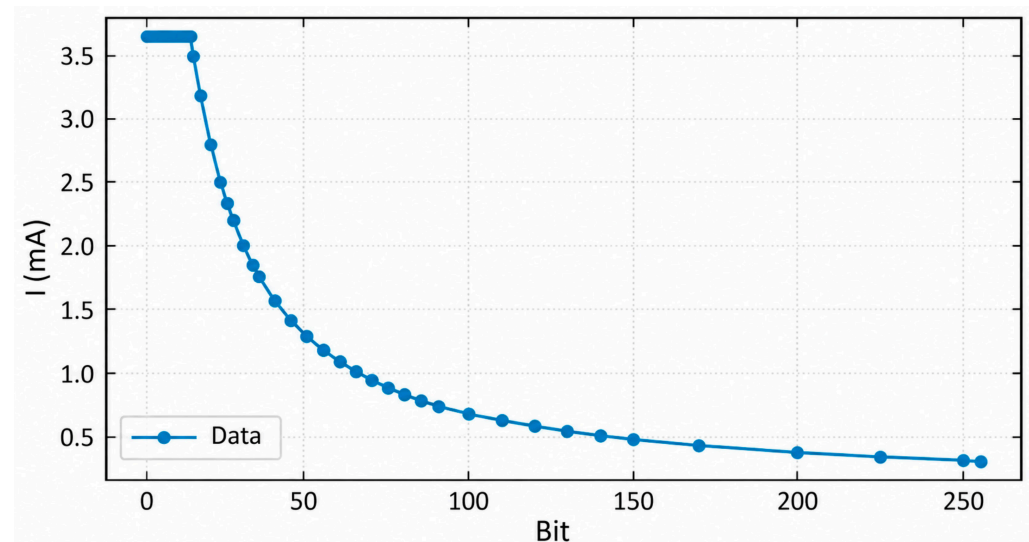


Figure 4. Measured current as a function of the MAX5413 digital potentiometer setting. The response follows the expected hyperbolic behavior, while the E-452 protection diode (Figure 3) limits the maximum current to 3.8 mA.

Then, the current is properly routed by the high-voltage analog switch to the suitable stimulation electrodes [30]. This device multiplexes the single current source across the ten electrode channels and two ground paths and implements an H-bridge to dynamically invert stimulation polarity under firmware control. The HV2701 high voltage multiplexer (Microchip Technology, Chandler, AZ, USA) switches between channels in approximately 10 μs , enabling various techniques for simultaneous stimulation of more than one channel within a single 10 ms control cycle.

2.2.4. Haptic Stimulation Circuit Design and Electrode Configuration

All stimulation circuit components are implemented on a custom two-layer PCB, as seen in Figure 5. In this Figure, the different components that form the module, respectively, LMV341 OPAMP (1), MAX5413 potentiometer (2), MOSFET and BJT transistors (3), current limiting diode (4), HV2701 analog switch (5), and DC-DC NMT0572SC converter (6), are marked. The female connector headers located on the left side of the board are connected

to the microcontroller's control pins, while those on the right side carry the haptic signals to different parts of the hand.

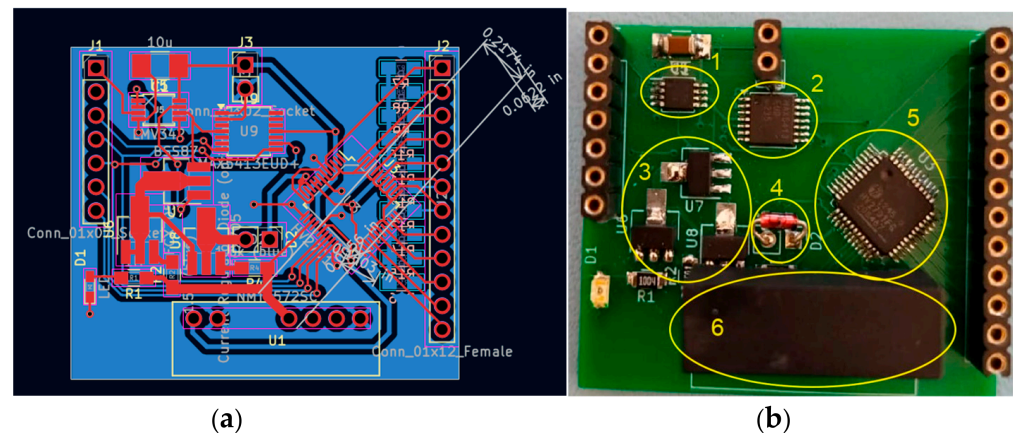


Figure 5. Custom-printed circuit board for the electrotactile stimulation circuit: (a) layout; (b) fabricated and assembled board.

From the output headers on the right side of the PCB, 10 correspond to the active electrodes (one per target palmar zone), and the other two are the independent ground electrodes placed on the dorsum of the hand, attached with standard medical-grade adhesive TENS pads. The final placement of the electrodes on the dorsal side of the hand is shown in Figure 6, along with the channel numbers, so that the haptic system works according to the channel definitions on the microcontroller firmware.

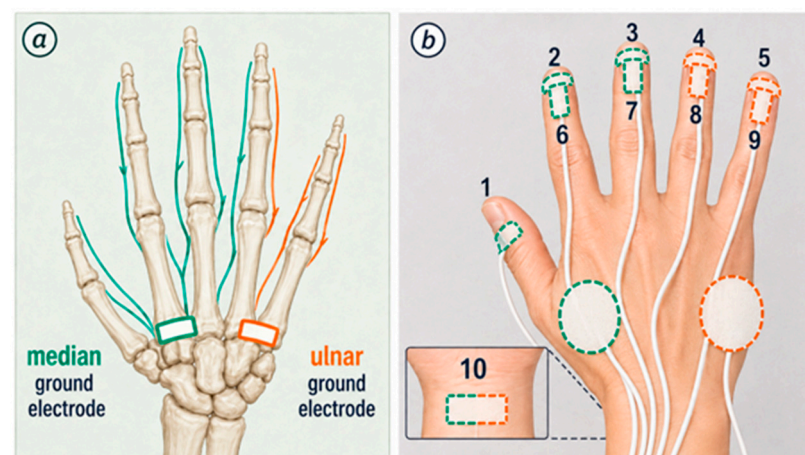


Figure 6. Final electrode configuration: (a) two ground electrodes are used to avoid median/ulnar interference; (b) electrode placement and corresponding channel number for each one (see Table 4).

2.3. Orientation Estimation

Accurate real-time estimation of the glove orientation is the core of the continuous-control pipeline. A basic implementation based on direct integration of the Euler angle requires time integration of two gyroscope axes, which introduces unbounded angular drift in one of the angles. To overcome this limitation, a nonlinear complementary filter, proposed by Mahony et al. [24], is implemented. The filter formulates the estimation problem as a deterministic observer kinematics system directly on the special orthogonal group $SO(3)$, exploiting the Lie group geometry to decouple gyroscope measurements from the reconstructed attitude error while remaining computationally efficient for the RP2040 hardware.

2.3.1. Orientation Representation: Unit Quaternions

Although Euler angles (yaw ψ , pitch θ , roll φ) provide an intuitive representation of orientation, they suffer from a well-known singularity called gimbal lock: when two rotation axes align, each orientation update step requires computationally expensive trigonometric functions. To overcome this issue, system orientation is based on unitary quaternions, an algebraic representation that avoids these singularities and updates orientation through simple arithmetic operations only. A unitary quaternion q is defined as

$$q = (q_0, q_1, q_2, q_3), \quad \|q\| = 1 \quad (2)$$

where q_0 is the scalar part, associated with the rotation angle, and (q_1, q_2, q_3) is the vector part, associated with the rotation axis. When Euler angles are needed to transmit orientation to the control environment, conversion is applied as a post-processing step, so all internal filter computations remain singularity-free.

2.3.2. Accelerometer–Gyroscope Fusion

The gyroscope provides an estimation of the angular velocity ω , whose time integration tracks orientation with high short-term precision. However, any static bias or slow drift in gyroscope readings is accumulated in the integration process, generating an orientation error that grows without limit over time. The accelerometer, in contrast, measures the total acceleration of the sensor; when the glove is not subject to appreciable external accelerations, this measurement is dominated by the Earth's gravitational field, whose direction is known in the inertial frame. This provides a drift-free absolute inclination reference, but it is noisy during fast hand movements.

The Mahony filter exploits the complementary nature of these two sensors: the gyroscope delivers an accurate and dynamic short-term estimate, while the accelerometer continuously corrects long-term drift. To quantify how much the estimated orientation deviates from the real one, a vector error $\mathbf{e}$ is defined as the cross-product between the normalized gravitational acceleration measured by the accelerometer and the gravitational direction predicted by the current quaternion in the sensor frame [24]:

$$\mathbf{e} = \hat{\mathbf{a}} \times \mathbf{v} \quad (3)$$

where $\hat{\mathbf{a}} = \mathbf{a} / \|\mathbf{a}\|$ is the normalized acceleration, and $\mathbf{v}$ is the estimated gravity direction obtained by rotating $[0, 0, 1]^T$ with the current quaternion:

$$\mathbf{v} = \begin{pmatrix} 2(q_1q_3 - q_0q_2) \\ 2(q_0q_1 + q_2q_3) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (4)$$

The cross-product yields a vector whose direction is the correction axis and whose magnitude is proportional to the attitude error angle. When both vectors coincide, the error is zero, and the filter introduces no correction. This formulation avoids a full algebraic attitude reconstruction, computing the error directly in the sensor frame without passing through Euler angles.

2.3.3. Proportional–Integral Control and Quaternion Propagation

The error vector $\mathbf{e}$ is used to correct the angular velocity before integration via a classical proportional–integral (PI) controller. The constants K_p and K_i are the filters' gain parameters, and they determine the response intensity to the orientation error. The proportional action, scaled by a gain K_p , reacts immediately to the present error; the integral action, scaled by a gain K_i , accumulates the error over time and compensates the static

gyroscope bias. The integral error term $\mathbf{e}_{int}$ and the corrected angular velocity $\boldsymbol{\omega}_c$ are updated each cycle using the gyroscopes newest measurement $\boldsymbol{\omega}$:

$$\mathbf{e}_{int}(t_k) = \mathbf{e}_{int}(t_{k-1}) + K_i \mathbf{e} \Delta t \quad (5)$$

$$\boldsymbol{\omega}_c(t_k) = \boldsymbol{\omega}(t_k) + K_p \mathbf{e}_k + \mathbf{e}_{int}(t_k) \quad (6)$$

The corrected angular velocity $\boldsymbol{\omega}_c$ propagates the quaternion via the unit quaternion differential kinematics. Discretised with a Euler step of size Δt , and denoting the components of $\boldsymbol{\omega}_c$ as (g_x, g_y, g_z) , the four quaternion components are updated as

$$\begin{aligned} q_0 &\leftarrow q_0 - \frac{\Delta t}{2} (q_1 g_x + q_2 g_y + q_3 g_z), \\ q_1 &\leftarrow q_1 + \frac{\Delta t}{2} (q_0 g_x + q_2 g_z - q_3 g_y), \\ q_2 &\leftarrow q_2 + \frac{\Delta t}{2} (q_0 g_y - q_1 g_z + q_3 g_x), \\ q_3 &\leftarrow q_3 + \frac{\Delta t}{2} (q_0 g_z + q_1 g_y - q_2 g_x) \end{aligned} \quad (7)$$

These operations consist entirely of floating-point additions and multiplications with no trigonometric functions, making the algorithm suitable for the RP2040 microcontroller. After each iteration, the quaternion is renormalized so that the unit-norm constraint is preserved, thus preventing numerical error accumulation:

$$\mathbf{q} \leftarrow \frac{\mathbf{q}}{\|\mathbf{q}\|} \quad (8)$$

The stability of this correction scheme was formally demonstrated in [24] via a Lyapunov analysis. Using the trace of the attitude error matrix and the estimated bias magnitude as a candidate function, its time derivative is shown to be negative semi-definite, guaranteeing almost-global convergence of the orientation error to zero and estimated bias towards its true value. The filter is implemented with proportional gain $K_p = 3.5$ and integral gain $K_i = 0.03$, experimentally determined to provide fast convergence, while disregarding the acceleration disturbances that arise during hand movement.

2.3.4. Glove Positioning System Calibration

Upon power-on, before entering steady-state estimation, the system executes a static calibration phase. With the glove held at rest, $N = 100$ gyroscope samples are acquired, and their average value is computed as the initial gyroscope offset:

$$\mathbf{b}_\omega = \frac{1}{N} \sum_{i=1}^N \boldsymbol{\omega}_i \quad (9)$$

where $\mathbf{b}_\omega$ is the three-component gyroscope bias vector estimated during the static phase, and $\boldsymbol{\omega}_i$ is the i -th raw triaxial angular-velocity sample acquired at the 104 Hz gyroscope output data rate while the glove remains motionless.

After obtaining this value, every angular velocity sample is corrected by subtracting $\mathbf{b}_\omega$ before feeding it to the filter. This pre-subtraction reduces the initial error observed by the PI integral term, substantially accelerating the filter's convergence to steady state.

A second practical issue arises from the fact that the physical orientation of the glove at power-on is arbitrary: without correction, the control environment would always initialize at a different attitude, depending on how the user puts on the glove. To address this issue, once calibration is completed, the resulting quaternion is captured as a neutral-state reference, and its conjugate is stored:

$$\mathbf{q}_{off} = (q_0, -q_1, -q_2, -q_3) \quad (10)$$

The orientation transmitted to the control environment is then not the raw $\mathbf{q}$ but the relative orientation with respect to that selected reference, as obtained by the quaternion product:

$$\mathbf{q}_{\text{rel}} = \mathbf{q}_{\text{off}} \otimes \mathbf{q} \quad (11)$$

This operation ensures that the system always starts from a user-defined neutral posture, regardless of the initial physical orientation of the glove, greatly simplifying the startup procedure and making the device robust to variations in the glove position between use sessions. Additionally, the user can trigger a reference reset at any time during operation, allowing the neutral posture to be dynamically redefined to accommodate changes in arm position or after long sessions.

2.4. Flex Sensor Mapping

The four flex sensors embedded in the glove can be used to encode two different sets of instructions. As previously mentioned, each flex sensor is connected to a resistive divider fed with 3.3 V, so that along with the resistance change from the sensors, the output voltage of each divider also changes. In the selected application example, the ring- and little-finger sensor readings are used as the input vector for a discrete gesture classifier, while the index- and middle-finger sensors are dedicated to continuous throttle control of the UAV's vertical axis. This separation was chosen to decouple the throttle input from the set of gesture commands and reduce ambiguity between postures that differ only in the degree of index- or middle-finger flexion, allowing the pilot to modulate altitude continuously without triggering a discrete command.

The raw ADC readings from both throttle sensors are normalized to $[0, 1]$ using the per-sensor calibration limits obtained at startup:

$$v_{\text{norm}} = \frac{v_{\text{raw}} - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}} \quad (12)$$

where v_{min} and v_{max} are the ADC values corresponding to full extension and full flexion, respectively, identified during the calibration phase. The throttle command is then computed as the difference between the two normalized readings, centered on their average and scaled to the UAV command range. Since different UAVs can have different ranges for their minimum and maximum throttle levels, exact values for our model will be described later.

2.5. Gesture Recognition

Posture identification for the ring and little fingers is implemented with a compact MLP deployed on the RP2040. Section 3.3 compares this model with different classifiers as axis-aligned threshold tree, k-NN, RBF-SVM, and Random Forest problems, thus characterizing the trade-off between accuracy and computing resource consumption of the proposed embedded design.

2.5.1. Input Features and Classes

The artificial neural network selected for command identification receives the two normalized ADC readings from the ring- and little-finger sensors. Compared to classifiers using all four fingers [10], the two-dimensional input reduces model size and inference cost, while preserving the two channels from the index and middle fingers for encoding continuous variables. Using the two discrete inputs, nine posture classes are defined, assigning seven to discrete-value actions and two to neutral or unassigned states. Figure 7 shows the postures and class identifiers.

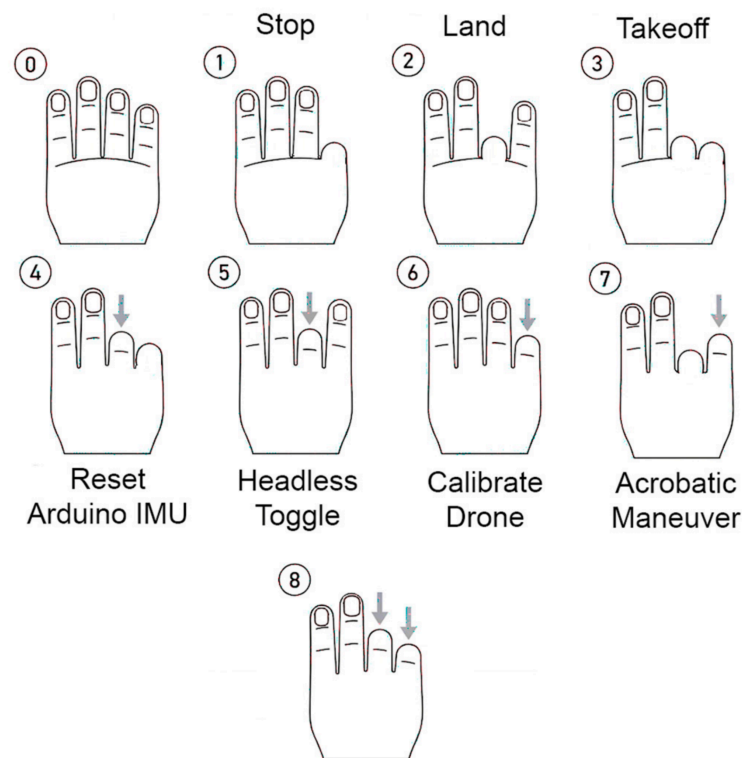


Figure 7. Finger positions that form the dataset and their corresponding identification numbers used on the classifier. The arrows indicate semi-flexed fingers. All classes except for 0 and 8 trigger drone-specific control actions when using the dataglove as a control device. For this dataset, only the ring and pinky fingers are considered.

2.5.2. Dataset Collection

Posture samples were performed by the same user over several days, with 50–54 observations in each of nine classes. The acquisition file stores the two ADC readings and their corresponding class label. An 80/20 stratified random split was used, with model selection restricted to the 368-sample training partition and final evaluation on 92 test samples. This protocol characterizes intra-operator performance under day-to-day variations in equipment placement and finger position. Normalization parameters were fitted on the training data and applied unchanged to the test data.

2.5.3. Architecture Selection and Training

The proposed model is a multilayer perceptron (MLP) with two inputs, two hidden layers, and nine outputs. Architecture selection used scikit-learn grid search with five-fold stratified cross-validation on the training partition. The candidate hidden-layer configurations were 40×20 , 64×32 , 128×64 , and 128×128 , combined with three L2 regularization values and three initial learning rates. The F1-score was the selection metric.

The best training-partition result used hidden layers of 40 and 20 neurons, L2 regularization of 10^{-5} , and an initial learning rate of 3×10^{-3} , with a mean cross-validated F1-score = 0.983. An equivalent network was subsequently trained and exported with TensorFlow Lite for embedded inference. Table 3 summarizes the proposed neural network architecture. Test results and comparisons with other classifiers are reported in Section 3.3. The three best candidates are compared with our selection for the hold-out partition, as described in Section 3.3.

Table 3. Final MLP architecture selected for the training partition.

Layer	Output Size	Parameters
Input	2	—
Dense 1 (ReLU)	40	120
Dense 2 (ReLU)	20	820
Output (Softmax)	9	189
Total	—	1129

2.5.4. On-Device Deployment and Confirmation Logic

The trained model is quantized using TensorFlow Lite [31] from the 3.11.3 version of Python-native 32-bit floating-point numerical representation to a microcontroller-compatible 8-bit fixed point, reducing the model size for flash memory by a factor of four, while introducing negligible accuracy loss. The quantized model is exported as a C header file and integrated into the Arduino firmware via the Eloquent TinyML library version 2.1.3 [32]. Measured on-device inference time is approximately 1.4 ms per prediction.

In the implementation deployed in this work, the microcontroller updates the classifier's output state every 80 ms. When the sensor readings identify a new class, a 350 ms logical confirmation window is initiated. Due to the classifier's state update time, four additional consecutive readings are taken within this window, so the estimated class is considered correct after five consecutive identical predictions (the initial prediction plus the four confirmation readings). Otherwise, the confirmation timer resets, waiting for the classifier to identify a new command. The time associated with the gesture identification phase, plus the configuration and transmission of RF packets to the controlled system at its nominal frequency of 40 Hz, means that the time elapsed from the start of a new gesture command, its confirmation window, and the transmission process requires approximately 400 ms. The subsequent blocking time configuration, designed to prevent command duplication, limits the number of instructions the system can transmit per second, with a maximum of two instructions for a minimum blocking period of 100 ms. In the system verification phase, presented in Section 3.5, a blocking time of 550 ms was established, limiting the number of commands to one per second to allow for proper visualization of the system's evolution. Note that this time between instructions has been specifically selected to ensure the stability of the controlled system's behavior, although it could be reduced by decreasing both the logical confirmation and blocking times. Finally, repeating the same command requires returning to the neutral class, which eliminates the hold-to-repeat function. Gestures released before confirmation, or before an active confirmation period expires, are intentionally rejected; therefore, the design prioritizes the deterministic suppression of transient and repeated activations over the omission of short gestures.

2.6. Haptic Feedback

The haptic subsystem maps the controller state computed by the glove to electro-tactile stimulation on the dorsum of the hand. Continuous variables and discrete events are assigned to different anatomical locations so that the signals can be interpreted without adding a visual display. The haptic layer has two functions: magnitude-dependent indication during continuous control and confirmation that a discrete gesture has been acknowledged by the microcontroller.

At the firmware level, the stimulation controller selects the active electrode channel through the HV2701 multiplexer according to the identified command, applies the polarity determined during perceptual calibration, and modulates the stimulation intensity through the digital potentiometer described in Section 2.2.3. The output pin does not encode the

stimulus amplitude directly; instead, it acts as a gate that enables or disables the current source. The perceived intensity is set by the MAX5413 configuration word P , where lower values correspond to lower resistance and therefore, higher stimulation current. The relationship between the programming word and the delivered current is experimentally characterized in Section 3.1.

The stimuli signals are generated using two different temporal patterns. The first one is the burst signal shown in Figure 8. A burst is a short group of square pulses, with each pulse remaining active for 50 ms for a period of 150 ms. This pattern is used for discrete events because it is perceived as a short, localized tap or button-like confirmation. A single pulse confirms standard accepted actions, such as takeoff, motor stop, camera tilt, or zero-orientation capture, whereas three pulses identify more specific actions such as arming an acrobatic maneuver or toggling headless mode. In this way, the haptic system provides the user with precise information about the identified command: the number of pulses indicates the type of confirmation, while its anatomical location is provided by the selected electrode path.

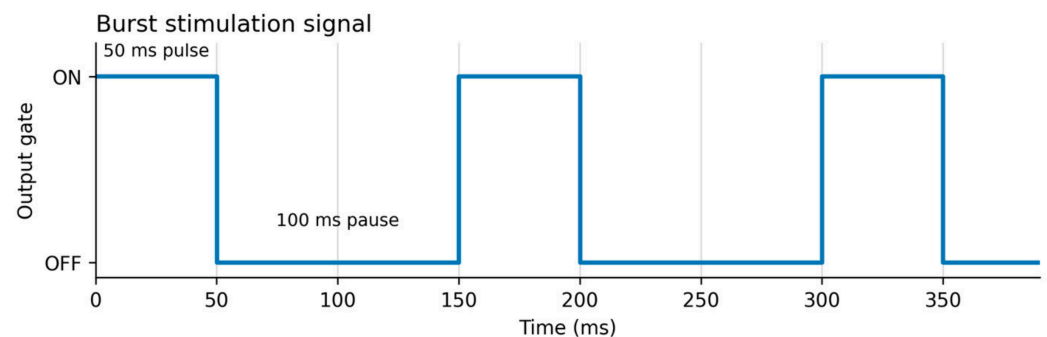


Figure 8. Simulated burst stimulation signal. The firmware generates pulses of 50 ms in width every 150 ms. This signal evokes a brief localized tap-like sensation and is used to confirm discrete commands.

The second pattern is the pulse train shown in Figure 9. The default train operates at 100 Hz, corresponding to a 10 ms period, with a pulse width of 400 μ s. Unlike the burst, this signal is not intended to mark a single event. It is used while a continuous flight axis remains away from its neutral value. The resulting sensation is perceived as a sustained localized tingling or vibration-like stimulus. Its strength is varied through the potentiometer setting: larger control deviations are mapped to lower P values, producing higher current and therefore, a more intense sensation.

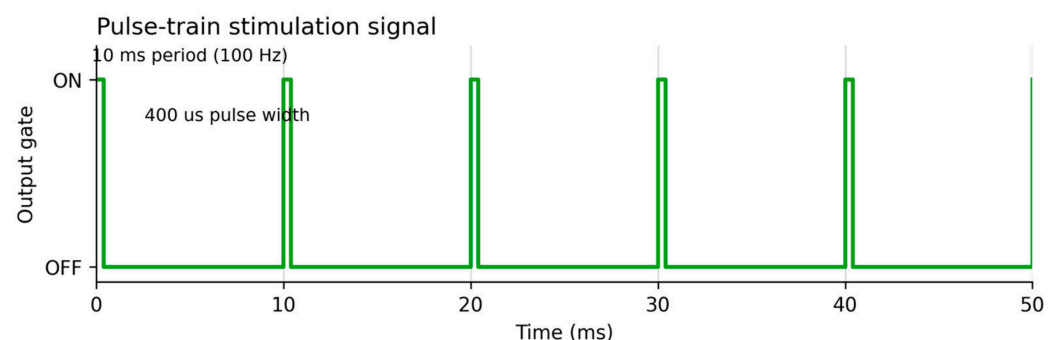


Figure 9. Simulated pulse-train stimulation signal. The firmware repeats narrow 400 μ s pulses at a nominal frequency of 100 Hz. This signal evokes a sustained tingling or vibration-like sensation whose intensity represents the magnitude of a continuous flight-axis deviation.

When no command has been accepted and all flight axes remain close to neutral, the HV2701 output matrix is cleared, and the stimulation output is kept low, so no tactile signal is generated. Once an accepted discrete command occurs, the firmware switches briefly to burst mode and produces one or three localized taps, depending on the type of event. During continuous control, the firmware instead uses the 100 Hz pulse train and updates the stimulation intensity every 10 ms. For each axis, the absolute deviation from the neutral stick value is mapped inversely to that axis-specific potentiometer interval. Thus, small deviations produce weak or no stimulation, while larger deviations generate a stronger sustained sensation at the corresponding hand zone. The final electrode routes and calibrated potentiometer intervals depend on the single-user perceptual validation; the demonstrator-specific channel map is reported later in the Section 3.

2.7. Communication and Drone Integration

To validate the complete wearable interface in a realistic control scenario, the proposed dataglove was integrated with a low-cost commercial quadcopter, a Karuisc K417. This platform requires simultaneous control of the four standard flight degrees of freedom, while also supporting auxiliary commands such as takeoff, landing, calibration, and acrobatic maneuvers. It therefore provides a demonstrator in which the inertial estimator, flex-sensor interface, embedded gesture classifier, wireless link, and the haptic feedback subsystem operate concurrently in the same real-time pipeline.

2.7.1. Wireless Command Interface

The selected drone provides its own access point and is controlled by means of a Wi-Fi User Datagram Protocol (UDP). To enable direct glove-to-drone communication, the fields of the control packets sent to and from the drone were empirically identified so that the microcontroller firmware was able to replicate their structure, i.e., sending the combined values of the four continuous flight axes as discrete command and mode flags. The four continuous axes are encoded as unsigned bytes, with 0x80 representing the neutral hover state and the values 0x28 and 0xDC defining the minimum and maximum command magnitudes accepted by the aircraft. Additional bytes are used to request discrete actions such as takeoff, landing, motor stop, IMU calibration, headless-mode switching, and acrobatic maneuvers. Protocol notes and source code are available in a public project repository [33].

2.7.2. Embedded Control Pipeline

All sensing, inference, haptic generation, packet construction, and Wi-Fi transmission are executed locally on the Arduino Nano RP2040 Connect (Figure 2, number 4), without any host computer in the loop. The full embedded controller is scheduled as a multi-rate pipeline: IMU acquisition and orientation update target the 104 Hz sensor cadence; gesture inference is scheduled every 80 ms (12.5 Hz); control packets are scheduled every 25 ms (40 Hz). At each transmission event, the dataglove orientation provided by the IMU is converted to yaw, pitch, and roll command bytes after applying the firmware-defined 8-degree dead zones. Nonlinear exponential shaping improves resolution around the neutral position, and angular saturation limits the mapped range to 45 degrees. The two flex sensors assigned to the throttle are combined, subjected to a 0.12 normalized dead zone before conversion to the drone byte range [0x28, 0xDC], with 0x80 as the neutral value. At the 40 Hz control rate, this filter has an analytical time constant of approximately 0.20 s and reaches 90% of a step change in approximately 0.45 s. Therefore, the throttle value is computed using the ADC values obtained for each of the two fingers:

$$T = \text{clip}(0x80 + k \cdot (v_{\text{index}} - v_{\text{middle}} - d_0), 0x28, 0xDC) \quad (13)$$

where k is a sensitivity scaling factor, and d_0 is the zero-throttle offset measured at rest during calibration. The $\text{clip}(\cdot)$ function enforces the safe command byte range accepted by the drone protocol.

To suppress abrupt transients caused by finger tremor or rapid posture changes, the throttle signal is smoothed with a first-order exponential moving average (EMA) filter:

$$T_f[n] = \alpha \cdot T[n] + (1 - \alpha) \cdot T_f[n - 1] \quad (14)$$

with a smoothing coefficient $\alpha = 0.12$. This final throttle value T_f is tuned during the demonstration trials to reduce hover-command oscillation while preserving a usable response to altitude inputs. The filtered value T_f is packed into the UAV control frame and transmitted at 40 Hz. In parallel, the confirmed gesture class provided by the two remaining flex sensors updates the discrete-command field to request the associated action.

This scheduling strategy keeps the continuous and discrete control channels synchronized while preserving enough computational headroom for simultaneous haptic rendering, whose timing performance is quantified in Section 3.4. The complete firmware and auxiliary documentation used for this integration are available in the same previously mentioned repository [33].

3. Experimental Results

The results are organized by subsystem and evidence source. Sections 3.1 and 3.2 characterize the stimulation electronics and electrotactile calibration; Section 3.3 reports gesture-classification performance and baseline comparisons; Section 3.4 reports embedded execution time and scheduler-derived response estimates; and Section 3.5 evaluates the complete architecture through a real-UAV demonstrator.

3.1. Electrical Characterization of the Stimulation Circuit

For firmware implementation, the current curve in Figure 4 is linearized by the variable substitution $y = 1/I$. Over the operating interval, the transformed measurements are described by the linear model $y = aP + b$ shown in Figure 10.

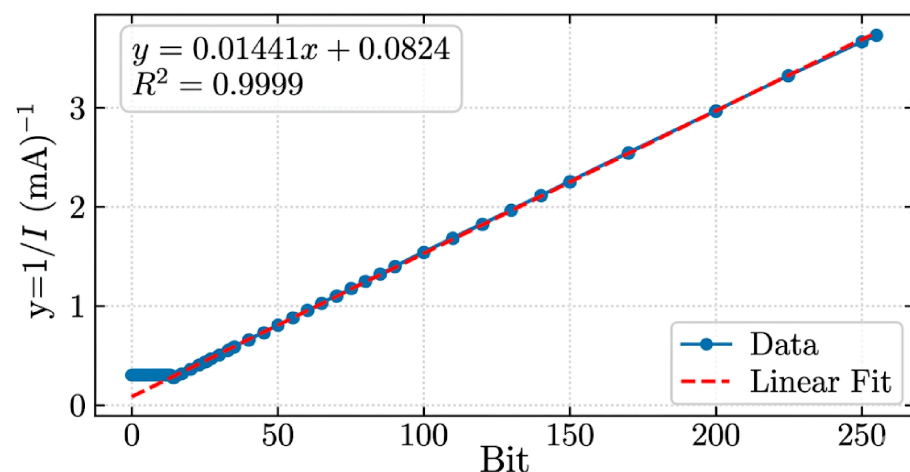


Figure 10. Linear regression after transforming the current curve with $y = 1/I$. The high coefficient of determination confirms that the linearized model is valid over the usable operating range.

Therefore, for the operating interval, the current I delivered by the circuit can be approximated as

$$I(P) = \frac{1}{0.01441 P + 0.0824} \text{mA}, \quad P \in [15, 255] \quad (15)$$

For the values of $P < 15$, the current will be approximately the same as for $P = 15$. By using this expression, the actual current sent to the haptic channels for each value of P is defined.

3.2. Electrotactile Calibration and Channel Mapping

Electrotactile calibration is performed with the user who controls the demonstrator UAV. The standardized procedure follows six steps: (1) inspect the skin and reproduce the electrode placement shown in Figure 11; (2) initialize the potentiometer at its highest-resistance setting, with stimulation disabled; (3) select one channel and apply a 50 ms burst while decreasing resistance stepwise until a clear, but comfortable, sensation is reported; (4) record the detection level and perceived palmar location; (5) reverse polarity and repeat when localization is absent or differs from the target; and (6) retain the lowest comfortable setting that produces the intended location, subject to the 3.8 mA hardware limit. Recalibration each time a user wears the glove compensates for changes in electrode contact, placement, skin condition, and individual sensitivity.

Table 4. Results of the perceptual calibration performed on one of the authors of this article. For each channel, the perceived sensation is recorded for each polarity. In the last column, the selected polarity is determined to stimulate the corresponding target zone.

Channel	Target Zone	Anodic *	Cathodic *	Selected Polarity
1	Thumb	Dorsal	Valid	Cathodic
2	Index distal	Valid	Dorsal	Anodic
3	Middle distal	Valid	Dorsal	Anodic
4	Ring distal	Valid	Valid	Both
5	Little distal	Valid	Dorsal	Anodic
6	Index proximal	Valid	Dorsal	Anodic
7	Middle proximal	Valid	50/50	Anodic
8	Ring proximal	Valid	50/50	Anodic
9	Little proximal	Valid	50/50	Anodic
10	Palm	None	Valid	Cathodic

* Note: Results from preliminary single-operator calibration. “Valid” denotes sensation at the intended palmar zone; “Dorsal” denotes sensation confined to the dorsum; “50/50” denotes sensation distributed between dorsal and palmar regions; and “None” denotes no perceived sensation.

The resulting channel configuration is stored as a session-specific mapping between each command, stimulation polarity, anatomical route, and intensity setting. This procedure preserves the common command feedback while allowing the electrical settings to be adapted.

The channel polarity configuration is summarized in Table 4. Single-user calibration shows that anodic stimulation is the most reliable option for the majority of channels. Specifically, channels 2 to 9 are successfully projected to their intended palmar zones with anodic stimulation, whereas cathodic stimulation often shifts the perceived sensation toward the dorsal side or generates an ambiguous localization. Otherwise, channel 1, corresponding to the thumb in our prototype configuration, and channel 10, corresponding to the palm, require cathodic stimulation to evoke the intended referred stimulus, while channel 4 is the only case in which both polarities provide valid stimulus location. These observations directly determine the channel-polarity map applied in the haptic feedback

module discussed in Section 2.6. The channel-wise procedure converts placement- and user-dependent perception into a reproducible set of operating parameters for each user session.

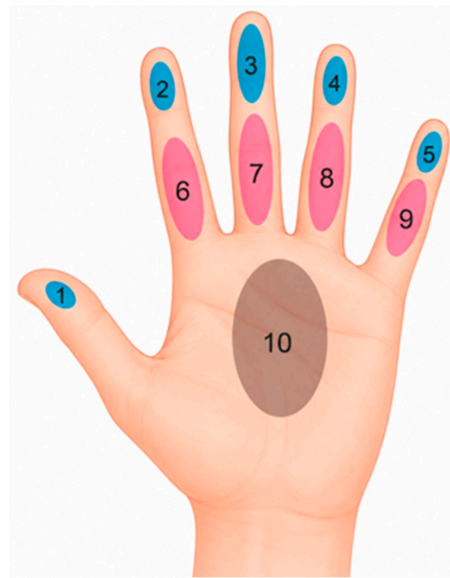


Figure 11. Target palmar zones associated with the ten active stimulation channels used in the perceptual validation study (see Table 4).

For the selected user, the suitable average current that provides tactile sensation, with no current-conducting gel, is 3.38 mA, within the configurable current range. Based on this calibration, a final demonstrator-specific haptic mapping has been defined for the application of UAV teleoperation. Table 5 presents the anatomical zones assigned to continuous control variables and discrete gesture classes, together with the corresponding selected anatomical zone for each one, and the range of potentiometer programming words used during the tests. Note that the specific value of P depends on several different considerations, such as skin condition, electrode placement or individual sensitivity.

Table 5. Calibrated haptic mapping for continuous axes and discrete command confirmation.

Control Variable/Gesture Class	Anatomical Zone	Type	Stimulation
Continuous control channels			
Yaw (rotation)	Index	Proximal anodic	Pulse train, $P \in [16, 25]$
Pitch (forward/backward)	Middle	Proximal anodic	Pulse train, $P \in [23, 28]$
Roll (lateral displacement)	Ring	Proximal anodic	Pulse train, $P \in [28, 35]$
Throttle	Palm	Cathodic	Pulse train, $P \in [15, 20]$
Discrete gesture confirmations			
Class 0: Unassigned	—	—	No stimulation
Class 1: Stop motors	Palm	Cathodic	1 pulse, $P = 18$
Class 2: Landing	Middle	Proximal anodic	1 pulse, $P = 25$
Class 3: Takeoff	Index	Proximal anodic	1 pulse, $P = 20$
Class 4: Reset Arduino IMU	Ring	Proximal anodic	1 pulse, $P = 30$
Class 5: Headless mode ON/OFF	Middle	Proximal anodic	3 pulses, $P = 25$
Class 6: Calibrate drone IMU	Ring	Proximal anodic	3 pulses, $P = 30$
Class 7: Acrobatic maneuver	Palm	Cathodic	3 pulses, $P = 18$
Class 8: Unassigned	—	—	No stimulation

3.3. Gesture Classification and Baseline Comparison

Gesture-classification evaluation is organized in three stages. First, model selection is performed on the 368-sample training partition by five-fold stratified cross-validation, using F1-score as the selection metric. Second, the selected MLP and four baseline classifiers are evaluated on the fixed 92-sample hold-out (test set) partition, comparing the results. Third, the selected MLP performance is tested at classification level and response speed for the reduced version of the model executed on the RP2040. Table 6 shows the dataset characteristics, selected architecture, and headline evaluation results.

Table 6. Summary of the dataset, selected MLP architecture, and evaluation results.

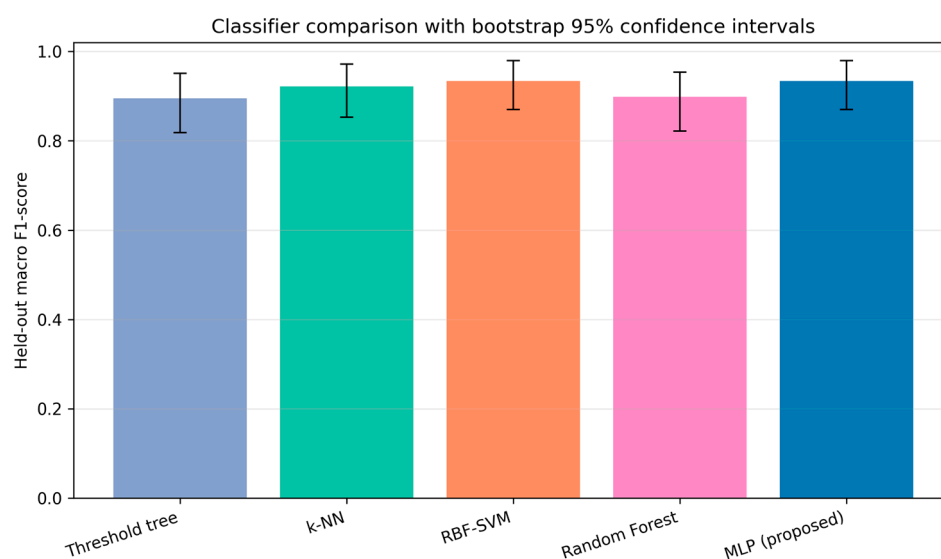
Metric	Value
Number of gesture classes	9
Dataset size	460 samples; nine classes; one operator
Training/test split	368/92 samples (80%/20%, stratified)
Selected hidden-layer sizes	40, 20
Cross-validated F1 score	0.9834
Hold-out accuracy	0.9348
Hold-out F1 score	0.9335
Measured RP2040 inference time	1.415 ms mean per-window maximum (0.974–2.847 ms)

Table 7 compares the selected MLP with the threshold-tree, k-NN, RBF-SVM, and Random Forest classifiers under the same training/test partition. The results presented in Table 7 correspond to the best values achieved while varying different parameters for each classification model: depth and minimum leaf size in threshold tree; neighbor count and weighting in k-NN; C and gamma in RBF-SVM; and tree count, depth, and leaf size in Random Forest. Figure 12 visualizes the achieved F1-score and its 95% confidence interval for the five classifiers compared. The RBF-SVM and MLP give the same test error values, yielding an accuracy of 0.935 and an F1-score of 0.934. k-NN followed with a 0.924 accuracy and a 0.921 F1-score, whereas the threshold tree and Random Forest models each achieved 0.902 accuracy. The overlapping confidence intervals indicate comparable predictive performance at this sample size.

Because a two-hidden-layer perceptron is not in itself a novel classifier, the four hidden-layer configurations explored during the grid search of Section 2.5.3 were retrained and evaluated on the same partition, with identical normalization and regularization settings. Although they span more than an order of magnitude in capacity (1129 trainable parameters for 40×20 , 2569 for 64×32 , 9225 for 128×64 and 18,057 for 128×128), the four networks provide identical predictions on the 92-sample hold-out set, misclassifying the same six patterns and therefore, sharing the same accuracy (0.935), F1-score (0.934), and bootstrap confidence intervals. Additional capacity provides no measurable benefit in a two-dimensional, nine-class problem whose decision boundaries are already resolved by the smallest candidate, and the residual error is attributable to overlap between neighboring postures rather than to limited model expressiveness. The 40×20 configuration is therefore selected because it reaches this common performance ceiling with one-sixteenth of the parameters of the largest candidate, which keeps the quantized model and its tensor arena within a few kilobytes of RP2040 SRAM and the measured on-device inference time at 1.415 ms, far below the 9.62 ms inter-sample window.

Table 7. Classifier comparison on the fixed 92-sample hold-out set. Confidence intervals are paired bootstrap percentile intervals; execution times correspond to PC implementation.

Classifier	Accuracy (95% CI)	F1 Score (95% CI)	Average Time (PC, μ s/Sample)	Serialized Size (KB)
Threshold tree	0.902 (0.837–0.957)	0.894 (0.818–0.950)	1.14	4.99
k-NN	0.924 (0.870–0.967)	0.921 (0.852–0.971)	13.52	13.98
RBF-SVM	0.935 (0.880–0.978)	0.934 (0.869–0.979)	13.14	11.77
Random Forest	0.902 (0.837–0.957)	0.898 (0.822–0.953)	61.78	443.70
MLP (this work)	0.935 (0.880–0.978)	0.934 (0.869–0.979)	2.72	37.74

**Figure 12.** F1-score with paired bootstrap 95% confidence intervals for the test patterns (holdout) set. The comparison characterizes the predictive-performance range of the candidate models on the fixed 92-sample test partition.

The selected MLP correctly classifies 86 of the 92 hold-out samples. Table 8 reports the number of samples, precision, recall and F1 for every class. Figure 13 presents the corresponding confusion matrix, with true classes arranged by row and predicted classes by column. Classes 0, 1, 3, and 5 achieved a recall of 1.000; the lowest recall was 0.800 for class 6. Class-wise F1 ranged from 0.889 to 1.000.

Table 8. Class-wise performance of the selected MLP on the 92-sample test set. Assigned functions follow the interface mapping in Table 5.

Class	Assigned Function	Number of Samples	Precision	Recall	F1
0	Neutral/Unassigned	10	0.909	1.000	0.952
1	Stop motors	11	0.917	1.000	0.957
2	Landing	10	0.900	0.900	0.900
3	Takeoff	10	1.000	1.000	1.000
4	Reset orientation	10	0.900	0.900	0.900
5	Headless mode toggle	11	0.917	1.000	0.957
6	IMU calibration	10	1.000	0.800	0.889
7	Acrobatic maneuver	10	0.900	0.900	0.900
8	Unassigned	10	1.000	0.900	0.947

The six classification errors shown in the confusion matrix in Figure 13 are distributed across five confusion paths: 2 to 7, 4 to 1, 6 to 0, 6 to 4, 7 to 2, and 8 to 5. The reciprocal 2/7 confusion identifies the clearest pairwise overlap, while class 6 accounts for two errors and the lowest class-wise recall. No other off-diagonal cell contains more than one sample. These values describe individual posture classifications. At command level, however, the developed algorithm requires five consecutive identical, sufficiently confident predictions before transmission so that isolated errors do not generate commands.

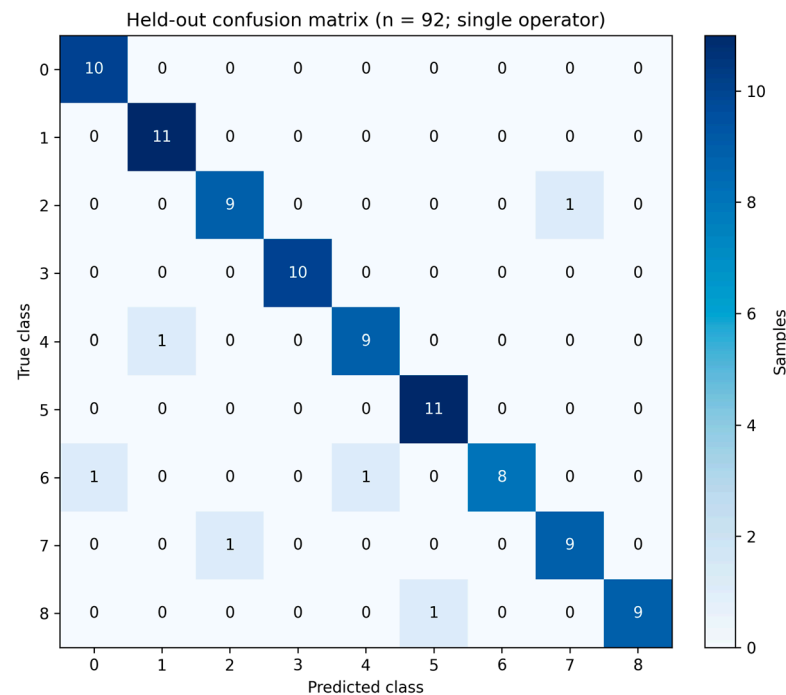


Figure 13. Confusion matrix of the selected 2-40-20-9 MLP for the test patterns ($n = 92$). Class identifiers and assigned functions are defined in Table 8; rows show true classes, and columns show predicted classes.

3.4. Embedded Execution Time and Latency Bounds

Embedded execution time was measured and averaged over 21 different samples. Within each execution sample, the firmware recorded the largest duration observed for every processing stage and for the complete control cycle. Table 9 reports the mean of these values, together with the complete-cycle statistics.

Concerning the MLP, the average maximum inference time executed on the RP2040 is 1.415 ms, with a 0.974–2.847 ms range across the 21 timing windows measured. Note that the largest recorded inference-stage only covers 3.6% of the 80 ms classification schedule fixed in this work (Section 2.7.2) and remains below the 9.62 ms IMU update interval, thus not impacting the system's execution flow.

The worst-case execution time for a full dataglove status monitoring and actuation process is 8.031 ms. Note that for the time limit of 9.62 ms imposed by the IMU reading period (104 Hz), in which all steps must be executed to maintain system synchronization, the worst-case measured cycle is of 83.5%, thus providing a time margin of 1.589 ms. The embedded sensing, inference, communication, and haptic tasks, therefore, are completed within the fastest scheduled update interval in every monitoring window.

Table 9 shows the average execution time of each of the processes running in the data glove's logic module, as well as the maximum execution time of the overall process obtained within the set of 21 measurements taken. This time is less than the state update period of the system under control, which, in the case of the proposed example, i.e., a quadcopter, is

25 ms, corresponding to its update frequency of 40 Hz. With these time values, the data glove is able to send the device the updated information from the proportional output sensors (IMU for orientation, flex sensors for speed) in a maximum time of 33.1 ms in the worst case, assuming that the measurement cycle is completed immediately after the last transmission cycle, accumulating (8.013 ms + 25 ms, Table 9).

Table 9. Embedded execution-time statistics from 21 dataglove operation processes. Each value for a complete cycle is measured independently per window before being averaged.

Quantity	Statistic	Time (ms)
IMU read and Mahony filter	Mean of per-window stage maxima	3.494
ADC acquisition	Mean of per-window stage maxima	0.083
Neural-Network inference	Mean of per-window stage maxima	1.415
CPU control logic	Mean of per-window stage maxima	0.212
Wi-Fi transmission	Mean of per-window stage maxima	2.115
Haptic update	Mean of per-window stage maxima	0.403
Other operations	Mean of per-window stage maxima	0.039
Stage-wise total	Stage-wise mean maxima (arithmetic sum, not an observed cycle)	7.761
Complete control cycle	Mean of per-window complete-cycle maxima	6.880
Complete control cycle	Observed execution time range (cycle) for the 21 processes	6.219–8.031
Complete control cycle	Largest observed execution time	8.031

Regarding the identification and transmission of discrete commands, obtained through the inference process executed in the neural network embedded in the glove's microcontroller, from the signals provided by the flex sensors (Figure 7), the update time has been set at 80 ms, with a subsequent logical confirmation process consisting of four consecutive identical inferences. This results in a confirmation time of approximately 350 ms before the command is sent via RF to the controlled device. Thus, the maximum time between transmissions of two commands is approximately 455 ms, which includes the initial inference time (80 ms) and four confirmation inferences (350 ms), plus an additional 25 ms in case the confirmation time ends immediately after a transmission period, requiring a wait for the next transmission.

Concerning the configuration of the electrotactile feedback, the update period required for configuring the stimulation signals and their corresponding routing is estimated at 10 ms, to which must be added the maximum configuration time for the haptic stage electronics, 0.460 ms. The total time is short enough to be covered by the estimates of the other processes (such as the logical confirmation of discrete commands), so the haptic stimulation process does not affect the system's dynamics.

3.5. Power Consumption

To estimate the power consumption of the proposed system, the following four modules that conform the full system have been considered:

- **Sensing:** This includes the inertial measurement unit (IMU) and the four flex sensors tied to the glove fingers. Regarding the operation of these flex sensors, the maximum power consumption was estimated for each resistive divider, comprising the flex sensor (with a nominal resistance of 500 Ω , the minimum value in its range) in series with a fixed 300 Ω resistor (Section 2.2.2).

- **Signal conditioning:** This is composed of the elements that provide the current signals to stimulate the haptic module, i.e., amplifiers, a digital potentiometer, and both BJT/FET transistors.
- **Power management:** This includes the voltage regulation, DC-DC converter, and 72 V analog switch operation. The switch power consumption includes the power due to the zero-load input current plus continuous operation with a 3.8 mA output (Section 2.3.3), delivering a burst signal 50% of the time and a pulse train the rest of the time (Section 2.6), with its nominal efficiency of 85%. Note that these values are overstated, as they assume the system is constantly stimulating the nerve endings in the hand at the maximum current level.
- **System control:** This module consists of the microcontroller responsible for resource management, inference, command transmission, and haptic feedback, as well as the Wi-Fi module. Power consumption has been estimated assuming a transmission time (Wi-Fi on) of 10%.

Table 10 shows the estimated maximum consumption values for each of these modules, as well as the total consumption. Due to the lack of a standard performance evaluation framework for these devices, it is difficult to compare power consumption with data reported in the literature; however, the maximum power consumption of the developed system falls within the range of other smart gloves (see [15]), even when accounting for the processing and communication stages involved in its operation. Specifically, based on the data in Table 10, if the proposed system is powered by a 7.4 V, 1000 mAh LiPo battery, its operating life would reach a minimum of 16.5 h.

Table 10. Power consumption of the dataglove distributed by subsystem.

Subsystem	Elements	Power (mW)
Sensing	IMU, flex sensors	18.30
Conditioning	Amplifiers, digital potentiometer, transistors	6.00
Power management	Regulator, DC-DC converter, analog switch	310.25
System control	Microcontroller, Wi-Fi module	109.00
Total		443.55

3.6. Closed-Loop Flight Control Demonstration

As a system-level demonstrator, the developed wearable interface is applied to control a Karuisc K417 quadcopter. The experiment is designed to concurrently execute all the processing stages: inertial orientation sensing, flex-based throttle, posture recognition, temporal command confirmation, direct UDP transmission, and electrotactile command feedback. The selection of a UAV application was due to the fact that it combines proportional multi-axis control with discrete events in a compact and visually verifiable task. During operation, IMU estimates generated continuous yaw, pitch, and roll command bytes; index- and middle-finger flexion are converted in throttle values; and ring- and little-finger postures determine high-level commands. Thus, microcontroller-embedded mappings provide four proportional channels and seven discrete actions, while the electrotactile signals indicate the command state recognized by the glove. Takeoff, landing, directional maneuvers, throttle modulation, and simultaneous continuous control were observed in the indoor demonstration [33]. Table 11 summarizes the measured and analytically estimated performance of the UAV demonstrator. The scheduler-derived estimates are useful across target platforms because they characterize the wearable interface up to packet transmission. The recorded demonstration shows consistent command-to-motion corre-

spondence and a continuous concurrent operation of the sensing, classification, wireless, and haptic subsystems.

Table 11. Interface-level performance of the UAV teleoperation demonstrator. Analytical estimates are derived from the measured execution maximum and fixed firmware schedules.

Metric	Evidence Basis	Result
Complete embedded cycle	Measured over 21 monitoring windows	6.880 ms mean for-window maximums; 8.031 ms largest observed cycle
Continuous sensing-to-packet response	Analytical estimate from measured maximum cycle plus 25 ms packet schedule (worst case)	≤ 33.1 ms
Discrete command-to-RF transmission time	Analytical estimate from 80 ms inference, 350 ms confirmation, and 25 ms packet schedule	455 ms
False-command discarding	Deterministic firmware behavior	Command confirmation requires five consecutive identical, sufficiently confident predictions
Short-gesture rejection	Deterministic firmware behavior	Gestures released before the 100 ms confirmation point are rejected by design
Repeated-command suppression	Deterministic firmware behavior	555 ms quantized lockout; a command repetition requires a previous neutral gesture
Throttle smoothing	Analytical estimate for $\alpha = 0.12$ at 40 Hz	Approximately 0.20 s time constant and 0.45 s to reach 90% of a step
Gesture classification	Measured on 92 test samples	0.935 accuracy; 0.934 F1-score
Haptic feedback status update	Maximum estimated of 10 ms for signal shape selection and feedback path plus 0.460 ms for timing configuration	Below 10.46 ms after command acceptance
Control and feedback coverage	Implemented and observed in the UAV demonstrator	Four proportional axes, seven discrete actions, and ten electrotactile routes operated concurrently

4. Discussion

The combined measurements show that the architecture meets its embedded real-time requirements. The largest observed measurement cycle, 8.031 ms, is below the 9.62 ms interval defined by the IMU reading period; therefore, a measurement process can be encapsulated between two consecutive measurements of the inertial sensor. Including a 25 ms time associated with the period between two data transmissions required by the controlled device in the proposed example, the upper time limit from continuous sensor measurement to the corresponding RF transmission of the associated actuation is 33.1 ms. Discrete commands require time confirmation, resulting in an estimated time from gesture identification to radio frequency transmission of the corresponding command of 455 ms.

Electrical characterization demonstrates that the excitation for the haptic module can be controlled, providing different feedback current profiles for different commands. The calibration results further show that polarity and intensity can be adjusted independently for each anatomical route. Considering calibration as an individual per-user, per-donning configuration step accommodates the effects associated of electrode position, contact, skin condition, and individual sensitivity [6,8], while preserving a common ten-channel feedback vocabulary.

The gesture-recognition results establish an effective accuracy–resources consumption balance for embedded operation. The discrete command identification provided by the selected MLP architecture, after being discretized and reduced for integration as an inference module in the microcontroller chosen as the dataglove control device, provides results equivalent to the best test accuracy and F1-score values observed in comparison with the reference classification methods used, with a maximum execution time of only 2847 ms.

The architectural comparison in Table 1 shows how the proposed system occupies a distinct design point. Used as an example application in quadcopter control, the proposed system maintains the usability of IMU-based controllers in [2,3], while integrating continuous and discrete commands into a single fully portable device: instructions to modify orientation and speed in real time based on the combination of IMU and flex sensor readings, gesture inference for the execution of complex discrete commands, transmission of commands by UDP protocol and management of a ten-path electrotactile user feedback system, all using a single RP2040 module as the control device. Unlike the VR and RGB-camera approaches in [9,10], operation does not depend on an immersive controller or an external camera view. The application command map can therefore be modified on the microcontroller firmware according to different conditions or applications, while the sensing, inference, scheduling, communication, and tactile-feedback layers remain unchanged. For safety-critical deployment, wearable commands can serve as references to a downstream constrained controller. For example, the acceleration-level hierarchical quadratic programming framework of Xing et al. [23] integrates trajectory tracking, physical constraints, manipulability, and obstacle avoidance under prioritized tasks. This downstream layer complements, rather than forms part of, the wearable-interface contribution.

Several limitations qualify the scope of these results. First, the reported execution-time budget is specific to the microcontroller used. The stage-wise figures in Table 9 depend on the computing resources, the integrated peripherals, and the scheduling behavior of the platform, so they should be read as platform-specific rather than as intrinsic properties of the architecture. Exporting the firmware to a different microcontroller family, or to a Linux-class board on which a complete cycle cannot be deterministically encapsulated between two consecutive IMU samples, would require the timing characterization of Section 3.4 to be repeated. Second, only feed-forward perceptrons operating on two static flex features were evaluated. Architectures that exploit the temporal dimension, such as temporal convolutional networks and recurrent models of the Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) families, were left outside the comparison because the present command vocabulary consists of static postures and because their memory and latency footprints are not compatible with the 80 ms classification slot. Extending the interface to dynamic gestures would require revisiting this choice, as well as the 8-bit quantization scheme, whose negligible accuracy loss has been verified only for the selected network. Third, the classifier, the electrotactile calibration, and the perceptual mapping were characterized for a single operator, so inter-subject variability in skin impedance, electrode placement, and finger geometry remains uncharacterized; whether the calibration procedure scales to a wider population without per-user retraining is an open question that only a multi-user study can answer.

5. Conclusions

This paper presents the development and testing of a prototype for an autonomous control device capable of interpreting commands from hand dynamics without requiring an additional computing system nor external monitoring and verification components. The prototype, powered by a LiPo battery, consists of a data glove that includes a six-axis inertial measurement unit, a set of four flexible sensors attached to the fingers, a radio-frequency

module, and an RP2040 microcontroller responsible for managing measurement acquisition, command interpretation, and transmission to the controlled device. Additionally, a haptic feedback circuit included in the design transmits information to the user about the recognized commands through current signals applied to appropriate points on the hand, emulating different types of tactile stimuli, comprising a non-visual confirmation.

To test the system's flexibility, a set of commands was coded, including both continuous responses based on the glove's spatial position, determined by IMU measurements and the flexion of two of the flex sensors, and a discrete instruction set based on the position of the remaining two fingers and their flex sensors. The interpretation of these instructions is provided by an artificial intelligence module based on a multilayer perceptron embedded in the microcontroller. An analysis based on several performance parameters shows that the reduced MLP model implemented using TinyML achieves similar results to those for other common techniques such as k-NN or SVM, among others, using a very limited number of hardware resources and with inference times suitable for the chosen tasks.

The execution time in the microcontroller's firmware for all coded processes, including measurement acquisition, inference of continuous commands, configuration of the information packet for transmission via UDP protocol, and other operations, is less than the 9.62 ms period between consecutive readings of the IMU module, allowing the processes to be encapsulated in defined time periods. To verify the functionality of the developed device, it was applied to control the behavior of a UAV. For this purpose, all systems were implemented with a firmware that includes the process of acquiring sensor measurements to control the UAV's dynamic movement, as well as inferring discrete, high-level commands available to the UAV, which are sent to the drone via its own Wi-Fi UDP protocol. Command recognition is based on a confirmation process consisting of the consecutive acquisition of five measurements providing identical inference value within a period of up to 455 ms, a period of time that can be modified according to the requirements of the target application. The haptic stimulation system integrated into the glove provides precise confirmation stimuli, which must be calibrated for each user in terms of both intensity (signal current) and hand position for proper operation.

These results establish a flexible multimodal autonomous platform whose control mappings can be adapted to different interactive application systems, adjusting both the set of coded commands and the relationship between these and the stimulus signals to the target problem, as well as optimizing the architecture of the AI model used for inference. Future work towards an end-user wearable control device will require a detailed multi-user characterization and analysis.

Author Contributions: Conceptualization, N.M. and B.C.; methodology, U.A. and N.M.; software, U.A.; formal analysis, U.A. and B.C.; investigation, U.A., B.C. and N.M.; resources, U.A. and N.M.; data curation, U.A.; writing—original draft preparation, U.A.; writing—review and editing, N.M., B.C. and U.A.; supervision, N.M. and B.C.; funding acquisition, B.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Agencia Estatal de Investigacion, Ministerio de Ciencia Innovacion, grant PID2022-138785OB-I00 (MCIN/AEI/10.13039/501100011033/FEDER, UE). The work of Unax Arregi was supported by the Aragon Institute of Engineering Research (I3A) through its own funding program.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Institutional Review Board (or Ethics Committee) of University of Zaragoza/Aragon Institute for Engineering Research I3A (protocol code PID2022-138785OB-I00 and date of approval 22 January 2026).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The firmware, raw two-feature gesture dataset, model-selection scripts, drone communication protocol notes, and supplementary implementation material are available in the public repository [33]. The dataset includes the ADC features and class labels used for the reported system evaluation.

Acknowledgments: The authors would like to acknowledge the Servicio General de Apoyo a la Investigacion—SAI, Universidad de Zaragoza, and Óscar Lasarte for their invaluable support in the circuit design and assembly processes.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IMU	Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
HMI	Human–Machine Interface
PCB	Printed Circuit Board
ADC	Analog-to-Digital Converter
BLE	Bluetooth Low Energy
FPV	First-Person View
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
EMA	Exponential Moving Average
TENS	Transcutaneous Electrical Nerve Stimulation
LSB	Least Significant Bit
PI	Proportional–Integral

References

1. Suzuki, S.; Wu, Y.; Kumaghai, T.; Masuda, M.; Takahashi, K.; Hashimoto, N. A Prototype Study on External Human Machine Interface (HMI) for Automated Bus-Preliminary Testing for Effectiveness Verification Using Virtual Reality (VR). In Proceedings of the 2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Orlando, FL, USA, 16–21 March 2024. [\[CrossRef\]](#)
2. Ajikumar, A.; Moghaddam, M. IoT-Integrated Augmented Reality for Spatially Aware Human–Machine Interaction for IEEE Transactions on Human-Machine Systems. *IEEE Trans. Hum.-Mach. Syst.* **2026**, *56*, 723–732. [\[CrossRef\]](#)
3. Yan, N.; Wang, C.; Tao, Y.; Li, J.; Zhang, K.; Chen, T.; Yuan, Z.; Yan, X.; Wan, G. Quadcopter Control System Using a Hybrid BCI Based on Off-Line Optimization and Enhanced Human-Machine Interaction. *IEEE Access* **2020**, *8*, 1160–1172. [\[CrossRef\]](#)
4. Young, S.; Peschel, J. Review of Human–Machine Interfaces for Small Unmanned Systems With Robotic Manipulators. *IEEE Trans. Hum.-Mach. Syst.* **2020**, *50*, 131–143. [\[CrossRef\]](#)
5. Matias, S.; Rodriguez, E. Human-Machine Interfaces a Dual Perspective on Innovation and Complexity. In Proceedings of the 2025 9th International Conference on Robotics and Automation Sciences, Osaka, Japan, 27–29 June 2025. [\[CrossRef\]](#)
6. Zheng, M.; Crouch, M.; Eggleston, M. Surface Electromyography as a Natural Human–Machine Interface: A Review. *IEEE Sens. J.* **2022**, *22*, 9198–9214. [\[CrossRef\]](#)
7. Yeh, S.-C.; Wu, E.H.-K.; Lee, Y.-R.; Vaitheeshwari, R.; Chang, C.-W. User Experience of Virtual-Reality Interactive Interfaces: A Comparison between Hand Gesture Recognition and Joystick Control for XRSPACE MANOVA. *Appl. Sci.* **2022**, *12*, 12230. [\[CrossRef\]](#)
8. Lee, J.-W.; Yu, K.-H. Wearable Drone Controller: Machine Learning-Based Hand Gesture Recognition and Vibrotactile Feedback. *Sensors* **2023**, *23*, 2666. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Shin, S.-Y.; Kang, Y.-W.; Kim, Y.-G. Hand Gesture-based Wearable Human-Drone Interface for Intuitive Movement Control. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019; pp. 1–6. [\[CrossRef\]](#)
10. Piskozub, J.; Strumillo, P. Reducing the Number of Sensors in the Data Glove for Recognition of Static Hand Gestures. *Appl. Sci.* **2022**, *12*, 7388. [\[CrossRef\]](#)
11. Hikawa, H.; Ichikawa, Y.; Ito, H.; Maeda, Y. Dynamic Gesture Recognition System with Gesture Spotting Based on Self-Organizing Maps. *Appl. Sci.* **2021**, *11*, 1933. [\[CrossRef\]](#)

12. Tanaka, Y.; Shen, A.; Kong, A.; Lopes, P. Full-hand Electro-Tactile Feedback without Obstructing Palmar Side of Hand. In Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI'23), Hamburg, Germany, 23–28 April 2023. [\[CrossRef\]](#)
13. Culbertson, H.; Schorr, S.B.; Okamura, A.M. Haptics: The Present and Future of Artificial Touch Sensation. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 385–409. [\[CrossRef\]](#)
14. Kourtesis, P.; Argelaguet, F.; Vizcay, S.; Marchal, M.; Pacchierotti, C. Electrotactile Feedback Applications for Hand and Arm Interactions: A Systematic Review, Meta-Analysis, and Future Directions. *IEEE Trans. Haptics* **2022**, *15*, 479–496. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Ozioko, O.; Dahiya, R. Smart Tactile Gloves for Haptic Interaction, Communication, and Rehabilitation. *Adv. Intell. Syst.* **2021**, *4*, 2100091. [\[CrossRef\]](#)
16. Fang, Y.; Guo, W.; Sheng, X. Toward a Wireless Wearable System for Bidirectional Human-Machine Interface With Gesture Recognition and Vibration Feedback. *IEEE Sens. J.* **2022**, *22*, 9462–9472. [\[CrossRef\]](#)
17. Kajimoto, H. Electro-tactile Display: Principle and Hardware. In *Pervasive Haptics*; Springer: Tokyo, Japan, 2016; pp. 79–96.
18. Yang, Z.; Tomita, K.; Kamimura, A. VR-Based Teleoperation of UAV-Manipulator Systems: From Single-UAV Control to Dual-UAV Cooperative Manipulation. *Appl. Sci.* **2025**, *15*, 11086. [\[CrossRef\]](#)
19. Yun, G.; Kwak, H.; Kim, D.H. Single-Handed Gesture Recognition with RGB Camera for Drone Motion Control. *Appl. Sci.* **2024**, *14*, 10230. [\[CrossRef\]](#)
20. Shakhathreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634. [\[CrossRef\]](#)
21. Hassanalian, M.; Abdelkefi, A. Classifications, applications, and design challenges of drones: A review. *Prog. Aerosp. Sci.* **2017**, *91*, 99–131. [\[CrossRef\]](#)
22. Tezza, D.; Andujar, M. The State-of-the-Art of Human–Drone Interaction: A Survey. *IEEE Access* **2019**, *7*, 167438–167454. [\[CrossRef\]](#)
23. Xing, H.; Xu, Y.; Chen, J.; Li, W.; Liu, Y.; Xie, Y.; Ding, L. Multi-Objective Optimization with Priority-Based Execution for Mobile Manipulators: An HQP Approach Integrating Adaptive Motion Planning and Obstacle Avoidance. *Robot. Auton. Syst.* **2026**, *201*, 105457. [\[CrossRef\]](#)
24. Mahony, R.; Hamel, T.; Pflimlin, J.-M. Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Trans. Autom. Control* **2008**, *53*, 1203–1218. [\[CrossRef\]](#)
25. Arduino. Arduino Nano RP2040 Connect. Available online: <https://docs.arduino.cc/hardware/nano-rp2040-connect/> (accessed on 28 July 2026).
26. U-blox. NINA-W10 Series Open CPU. Available online: <https://www.u-blox.com/en/product/nina-w10-series-open-cpu> (accessed on 28 July 2026).
27. STMicroelectronics. LSM6DSOX: 6-Axis IMU (Inertial Measurement Unit) with Embedded AI: Always-On 3-Axis Accelerometer and 3-Axis Gyroscope; Datasheet DS12814, Rev. 4; STMicroelectronics: Geneva, Switzerland, 2024. Available online: <https://www.st.com/resource/en/datasheet/lsm6dsox.pdf> (accessed on 15 August 2026).
28. Arduino. Arduino_LSM6DSOX Library. Available online: https://github.com/arduino-libraries/Arduino_LSM6DSOX (accessed on 28 July 2026).
29. Spectra Symbol. SpectraFlex Resistive Flex Sensors. Available online: <https://www.spectrasymbol.com/resistive-flex-sensors/spectraflex-flex-sensors> (accessed on 28 July 2026).
30. Microchip Technology. HV2601/HV2701 Low-Charge Injection, 16-Channel, High-Voltage Analog Switch Data Sheet. Available online: <https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/HV2601-HV2701-Low-Charge-Injection-16-Channel-HV-Analog-Switch-Data-Sheet-20005391E.pdf> (accessed on 28 July 2026).
31. Google. TensorFlow Lite. Available online: <https://www.tensorflow.org/lite/> (accessed on 28 July 2026).
32. Eloquent Arduino. EloquentTinyML Library. Available online: <https://www.arduino-libraries.info/libraries/eloquent-tiny-ml> (accessed on 28 July 2026).
33. Arregi, U. Haptic Stimulation and Drone Control; GitHub Repository. Available online: <https://github.com/unax4/Haptic-stimulation-and-drone-control> (accessed on 28 July 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.