



Gesture-Controlled Embedded Interface for Sterile Medical Environments

Keshan Gunasekara ^{a*}

*^aDepartment of Electrical and Electronic Engineering, Sri Lanka Institute of Information Technology, Malabe
10115, Sri Lanka*

^{*}Email: keshangunasekara1789@gmail.com

Abstract

Maintaining sterility in surgical environments is critical, yet conventional input devices such as keyboards and mice pose contamination risks and are impractical during procedures. This paper presents the design and implementation of a compact, contactless gesture-controlled embedded system tailored for medical settings, enabling surgeons to interact with on-screen medical images and control connected devices without breaking sterile conditions. The system employs a Raspberry Pi Pico MCU (Microcontroller Unit) and three VL53L0X ToF (Time-of-Flight) sensors to recognize seven distinct hand gestures, including directional swipes and timed holds. Detected gestures are transmitted via USB (Universal Serial Bus) serial communication to a host computer, where a custom LabVIEW (Laboratory Virtual Instrument Engineering Workbench) interface interprets the input and executes corresponding actions such as image navigation, zoom, and device control. Visual feedback is provided through RGB (Red-Green-Blue) LEDs (Light-Emitting Diodes), enhancing usability and user confidence. In addition to software-based control, the system enables gesture-based actuation of a servo motor, demonstrating the capability to manipulate external devices directly from the embedded unit. The prototype demonstrated reliable gesture recognition, seamless integration with host applications, and precise servo motor control, validated through systematic functional testing. Achievements include accurate gesture detection, intuitive user feedback, robust communication between embedded hardware and software, and successful real-time device actuation. The prototype establishes a strong foundation for future enhancements, such as expanding the gesture set, improving classification algorithms, enabling wireless communication, and integrating a broader range of medical devices. Overall, this work demonstrates the feasibility and effectiveness of low-cost embedded

hardware for intuitive, real-time, and hygienic user interfaces in critical medical applications, paving the way for smarter and safer surgical environments

Keywords: Gesture Recognition; Sterile Medical Interface; Embedded System; Raspberry Pi Pico; Time-of-Flight Sensor; LabVIEW; Servo Motor Control; Contactless User Interface

* Corresponding author. Email address: keshangunasekara1789@gmail.com

1. Introduction

Embedded systems are specialized computing platforms designed to perform dedicated functions within larger electrical or mechanical systems. These systems typically incorporate an MCU (Microcontroller Unit) or microprocessor, along with sensors and actuators, to enable real-time monitoring, control, and automation in a variety of domains, including healthcare, automotive, and industrial automation [1], [2]. Their reliability, compactness, and low power consumption have made embedded systems indispensable in medical devices, supporting applications such as patient monitoring, drug delivery, and diagnostic imaging.

In surgical and other sterile medical environments, maintaining hygiene is of utmost importance. Surgeons and clinicians frequently need to interact with digital medical images such as X-rays, CT (Computed Tomography) scans, or MRI (Magnetic Resonance Imaging) during procedures. However, using traditional input devices like a mouse, keyboard, or touchscreen poses a contamination risk and can compromise sterile conditions [3]. Existing alternatives, such as voice recognition or foot pedals, may lack precision or reliability in fast-paced clinical settings, highlighting the need for a more practical, contactless method of interacting with on-screen content [4].

Gesture recognition has emerged as a promising approach for contactless human-computer interaction in medical environments. Gesture recognition enables systems to detect and interpret physical movements, typically of the hands or arms as input commands, allowing users to interact with digital systems without direct contact. This is especially valuable in the operating room, where maintaining sterility is critical [3], [4]. By using gesture-based controls, medical professionals can manipulate images, navigate records, or control devices in a more intuitive and hygienic manner [5].

Several technological approaches exist for implementing gesture recognition. Vision-based systems utilize cameras and image processing algorithms to analyze hand or body movements. While these systems can recognize complex gestures with high accuracy, they require significant computational resources, are sensitive to lighting conditions, and may raise privacy concerns [4], [6]. Sensor-based systems employ non-imaging sensors such as IR (Infrared), ultrasonic, or ToF (Time-of-Flight) sensors to detect changes in distance or motion. These solutions are compact, cost-effective, and robust against ambient light variations, making them suitable for embedded platforms in the operating room [7]. Wearable systems embed motion or flex sensors (such as accelerometers or gyroscopes) into gloves or wristbands to track limb movement. While precise, these require the user to wear a device, which may not be acceptable in sterile settings due to hygiene and comfort concerns [4], [8].

Given the constraints of surgical environments, sensor-based gesture recognition using ToF sensors offers a balanced solution. ToF sensors such as the VL53L0X provide high accuracy, small size, and consistent performance regardless of ambient light [7]. For the MCU, platforms like Raspberry Pi Pico are favored for their compactness, sufficient processing power, affordability, and ease of development. Communication between the embedded unit and the host computer is typically handled via a USB (Universal Serial Bus) serial interface, which offers simplicity and reliability during the prototyping phase.

The system developed in this paper leverages these technological choices to create a compact, real-time gesture-controlled embedded system for sterile medical environments. The design employs three VL53L0X ToF sensors and a Raspberry Pi Pico MCU to recognize seven distinct hand gestures, including directional swipes and timed holds. These gestures are transmitted to a host computer, where a LabVIEW (Laboratory Virtual Instrument Engineering Workbench) interface interprets them for image navigation, zoom control, and direct actuation of a connected servo motor. The addition of servo motor control enables users not only to interact with software applications but also to physically manipulate connected devices through gestures, a capability with significant potential for hands-free operation of medical instruments and other peripherals in the operating room [9]. Visual feedback is provided to the user via RGB (Red-Green-Blue) LED (Light-Emitting Diode) indicators, further enhancing usability and user confidence.

The prototype was carefully engineered to address the practical challenges of real-world clinical use. The sensors are arranged in a triangular configuration to maximize gesture coverage while minimizing hardware complexity. The microcontroller's firmware, developed in C++ using the Arduino IDE (Integrated Development Environment), manages sensor initialization, gesture detection, servo actuation, and serial communication. On the host side, the LabVIEW interface, supplemented by AHK (AutoHotKey) scripts, translates gesture codes into application commands, enabling seamless integration with medical imaging software and other computer-based tools. The system's modular architecture allows for future expansion, such as the addition of more sensors for finer gesture resolution, wireless communication via Bluetooth, or integration with a broader range of medical devices.

Extensive testing of the prototype confirmed its ability to reliably recognize both direction-based and time-based gestures, with minimal delay between gesture detection and action execution. The system provided robust performance under typical lighting conditions and demonstrated resilience to moderate variations in hand positioning and movement speed. Visual feedback through RGB LEDs and the LabVIEW interface helped users quickly learn and trust the system's responses. Some limitations were identified, such as reduced accuracy during rapid hand movements or suboptimal sensor alignment, which highlight opportunities for further refinement in sensor calibration and gesture classification algorithms.

By explicitly enabling gesture-based control of both digital interfaces and physical devices such as a servo motor, this prototype demonstrates how low-cost embedded hardware and robust sensor technology can be leveraged to create intuitive, real-time gesture interfaces for critical medical applications. The developed prototype not only addresses the immediate need for sterile, contactless image navigation but also establishes a flexible platform for future enhancements, including wireless operation, expanded gesture sets, and integration with additional medical devices. This approach contributes to the advancement of smarter, safer, and more efficient surgical environments.

2. Materials and methods

This chapter outlines the methodology adopted for the design and implementation of the gesture-controlled embedded interface, as depicted in Figure 1 (System Overview Diagram). The methodology is structured into four main sections: system design, basic system operation, hardware implementation, and software implementation. Each section describes the integration and interaction of sensors, microcontroller, servo control, and the LabVIEW-based user interface to ensure robust, real-time, and hygienic hands-free control in sterile medical environments.

2.1. System Design

The system is organized into three primary units: the Component Unit, the Microcontroller Unit, and the LabVIEW Unit. Figure 1 illustrates the architecture and data flow among these units.

The Component Unit comprises three VL53L0X ToF sensors arranged to maximize gesture detection coverage, two RGB LEDs for intuitive visual feedback, and a servo motor for device actuation. All components are mounted on custom acrylic boards for stability and optimal alignment. For the final prototype, a custom-designed PCB (Printed Circuit Board) was developed to integrate all sensors, LEDs, the servo interface, and microcontroller connections in a compact, robust, and reliable layout. The modular design of the PCB allows for straightforward integration of future sensors or peripherals, supporting scalability and adaptability for additional features.

The Microcontroller Unit is built around the Raspberry Pi Pico Microcontroller, which leverages its dual-core architecture for parallel task execution. During setup, the microcontroller defines constants, initializes GPIOs, and configures the ToF sensors, with built-in error detection routines to ensure reliable startup. After initialization, the workload is divided between the two cores: Core 0 is dedicated to gesture recognition and system communication, while Core 1 exclusively handles servo motor control. Core 0 continuously measures distances from the ToF sensors, identifies hand movements and holds, determines the correct gesture, updates the indicator LEDs, and sends gesture or data codes to the LabVIEW host via USB serial communication. Core 1 operates independently, listening for servo angle commands sent from LabVIEW, converting these to PWM (Pulse Width Modulation) signals, and actuating the servo accordingly. This separation of responsibilities ensures efficient, real-time operation without interference between gesture processing and hardware actuation and enables the system to respond instantly to both gesture inputs and servo commands.

The LabVIEW Unit on the host computer serves as the main user interface and system controller. It receives gesture codes from the microcontroller, interprets them, and executes corresponding commands for device or application control. LabVIEW also allows the user to set the servo angle, which is sent to the microcontroller for immediate actuation. The interface manages indicator LEDs, command execution, error messages, and can trigger external scripts or system commands as needed. The user interface is designed to be intuitive, providing function selection, real-time gesture and servo feedback, and robust error handling. LabVIEW can also interface with external applications via AutoHotKey scripts or the system command prompt, allowing seamless integration with medical imaging software or other tools.

This modular and parallelized architecture, as shown in Figure 1, ensures robust, real-time gesture recognition, intuitive feedback, and seamless control of both digital applications and physical devices. The design is also future proof for expansion, supporting additional sensors, wireless communication modules, or further hardware integration as needed.

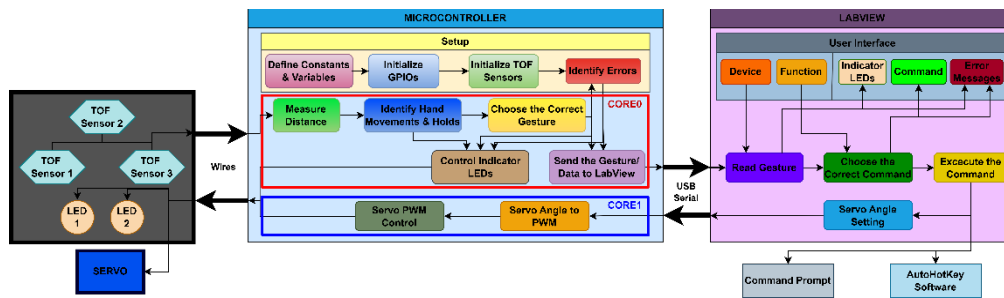


Figure 1: System Overview Diagram

2.2. System Operation

The system's functional flow begins with the Component Unit, where the ToF sensors continuously measure distances to detect hand presence and movement. The sequence and timing of sensor activations are analyzed to distinguish between directional swipes and timed hold gestures. The RGB LEDs provide immediate feedback, indicating gesture detection status and hold durations with distinct colors. The servo motor is also physically connected to the microcontroller and is actuated based on commands received from the host computer.

Within the Microcontroller Unit, Core 0 manages all gesture-related processing. It polls the ToF sensors, identifies hand movements and holds, determines the correct gesture, updates the indicator LEDs, and sends gestures or data codes to the LabVIEW Unit via USB serial communication. Core 1 operates independently, focusing solely on servo motor control. It listens to servo angle commands sent from LabVIEW, converts these to PWM signals, and actuates the servo accordingly. This dual-core approach ensures that gesture recognition and servo actuation run in parallel, maintaining real-time responsiveness and system reliability.

The LabVIEW Unit acts as the main user interface. It reads gesture codes from the microcontroller, interprets them, and executes the corresponding commands for device or application control. For servo actuation, LabVIEW allows the user to set the desired servo angle, which is transmitted to the microcontroller for immediate action. The interface also manages indicator LEDs, command execution, error messages, and can trigger external scripts (such as AutoHotKey) or system commands as needed.

2.3. Hardware Implementation

The hardware prototype consists of the Raspberry Pi Pico microcontroller, three VL53L0X ToF sensors, two RGB LEDs, and a servo motor, all mounted on custom acrylic boards. The sensors are positioned in a triangular configuration for optimal gesture detection, with careful wiring for power, ground, I²C communication, and individual shutdown control. The LEDs are connected to GPIO pins for independent color management, and the servo is attached to a dedicated GPIO pin configured for PWM output, allowing precise angle control based on received commands. The system is powered via USB, which also supports serial data transfer to the host computer.

To assist users in understanding system states and gesture feedback, the RGB LEDs are programmed to display specific color patterns during operation. These visual cues represent various events such as gesture detection,

validation, errors, and hold durations. A summary of the LED indicator states, and their corresponding meanings are provided in Table 1.

Table 1: LED Indicator States and Corresponding Messages

Colour	LED 1	LED 2
Blue	Gesture detection in progress	1-second hold
Green	Valid gesture detected	2-second hold
Red	Invalid gesture detected	3-second hold
Both LEDs illuminate red: critical error within the embedded unit.		

To further enhance reliability, compactness, and ease of integration, a custom-designed PCB was developed for the unit. This PCB consolidates all sensor, LED, servo, and microcontroller connections into a robust, streamlined layout, reducing wiring complexity and improving signal integrity. The custom PCB ensures consistent electrical performance, minimizes noise, and increases the mechanical durability of the system, which are key factors for real-world deployment in medical environments. The PCB layout was carefully designed to optimize component placement and routing, supporting both current requirements and future scalability.

The modular acrylic enclosure not only provides mechanical stability but also allows for easy adjustments during prototyping and testing. Colored acrylic was selected to make components visible, aiding in troubleshooting and offering a modern, professional appearance. The enclosure is assembled using laser-cut panels and spacers, ensuring secure housing and easy access for upgrades or repairs.

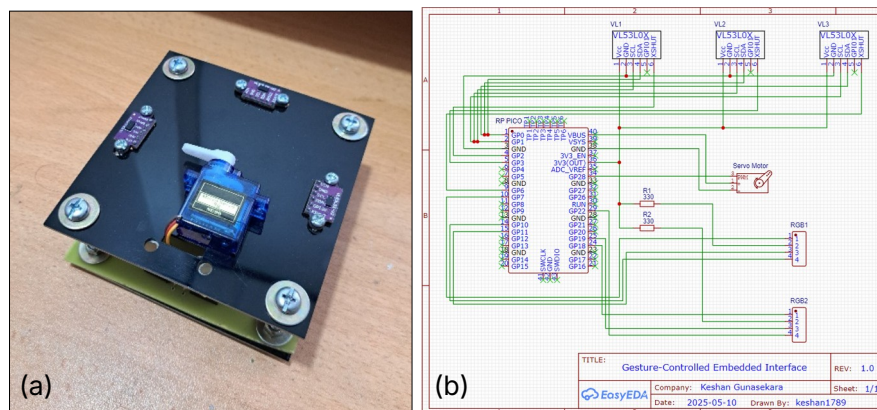


Figure 2: (a) Final prototype of the gesture-controlled embedded interface; (b) Circuit Diagram

2.4. Software Implementation

The software is divided between the embedded programming of the Raspberry Pi Pico and the LabVIEW interface on the host computer. The Pico is programmed using the Arduino and relevant libraries. Core 0 is responsible for sensor initialization, gesture detection logic, LED feedback, and serial communication. It continuously polls the VL53L0X ToF sensors, processes the activation sequence and timing, identifies gestures, and updates the RGB LEDs to provide immediate visual feedback. Detected gesture codes are sent to the host computer via USB serial communication.

Core 1 runs independently and is dedicated to servo control. It monitors the serial port for angle commands sent from LabVIEW and generates the corresponding PWM signal to actuate the servo motor to the specified position. This dual-core approach allows gesture recognition and servo actuation to occur in parallel, ensuring both functions remain responsive and do not interfere with each other.

On the host side, LabVIEW manages serial communication with the Pico, receiving gesture codes and sending servo angle commands. The user interface is designed for clarity and ease of use, allowing function selection, displaying gesture status, and providing control of the servo angle. Visual indicators on the interface further enhance usability: four green LEDs represent the detected swipe direction (up, down, left, right), while a slider in the center of the interface displays the duration of hold gestures. A red LED beside the direction indicators signals invalid gesture attempts, and a prominent red LED at the top of the interface indicates system-level errors such as sensor disconnections or communication failures. For gesture-based navigation, LabVIEW can trigger application commands or AutoHotKey scripts, enabling seamless integration with imaging software and other applications. For servo control, LabVIEW transmits the desired angle directly to the microcontroller, enabling precise and real-time hardware control.

The LabVIEW control logic handles serial data acquisition, command interpretation, and feedback mechanisms. It ensures only new or changed commands are executed, filters out redundant actions, and provides visual indicators such as LED patterns and sliders that mirror the physical unit. Error messages and system status updates are prominently displayed to facilitate troubleshooting and ensure reliability.

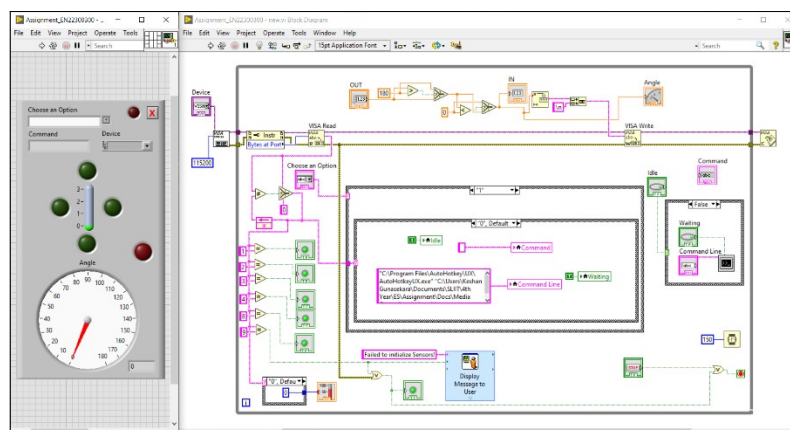


Figure 3: LabVIEW user interface and control block diagram

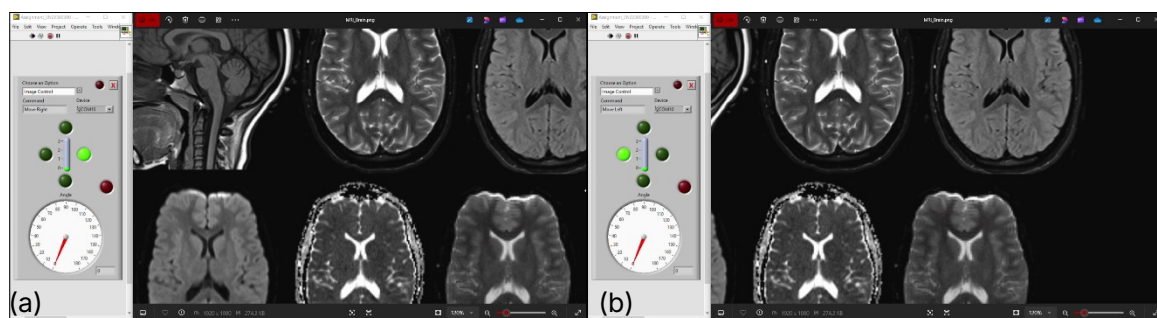
This comprehensive software methodology ensures robust, real-time gesture recognition and reliable control of both digital applications and physical devices. The modular design also supports future enhancements, such as wireless communication or expanded gesture sets, making the system adaptable for evolving requirements in sterile medical environments. Figures illustrating the LabVIEW user interface and the control block diagram are provided below to demonstrate the software's visual design and logical flow, in Figure 3.

3. Results

To assess the effectiveness of the developed gesture-controlled embedded interface, a series of experiments and functional tests were conducted. Following the completion of system integration and software development, the prototype was systematically tested to assess its accuracy, responsiveness, and reliability in both gesture recognition and servo motor control. The evaluation focused on real-time operation of the system, including the effectiveness of the LabVIEW user interface, the accuracy of gesture classification, the responsiveness of servo actuation, and the clarity of user feedback. Representative screenshots and control diagrams are included to illustrate the system's operation and to provide visual evidence of its performance during typical use scenarios.

The effectiveness and intuitiveness of the gesture-controlled embedded interface were first evaluated by observing the real-time visual feedback provided by the system's indicator LEDs during various operational states. The LED indicator system accurately reflected different gesture events and system conditions, including detection progress, valid and invalid inputs, hold durations, and critical errors. During testing, these visual cues consistently provided clear and immediate feedback, allowing users to interpret system responses confidently and interact with the prototype intuitively. The indicators significantly enhanced usability by confirming gesture recognition and highlighting system statuses without requiring additional displays or prompts.

Following the evaluation of LED feedback, the system's real-time integration with the LabVIEW user interface was tested during typical image navigation tasks. The Figure 4 below shows the LabVIEW interface in action as various gestures were performed to control on-screen images, such as zooming in and out, and moving the image up, down, left, or right. Each gesture is accurately detected by the system and immediately reflected in the interface, with corresponding commands executed for image manipulation. In addition, the screenshots also capture how the interface responds to invalid gesture inputs and sensor failure, clearly displaying error messages and status indicators. These results demonstrate not only the seamless translation of hand gestures into practical control actions but also the system's robustness in handling faults and providing immediate user feedback.



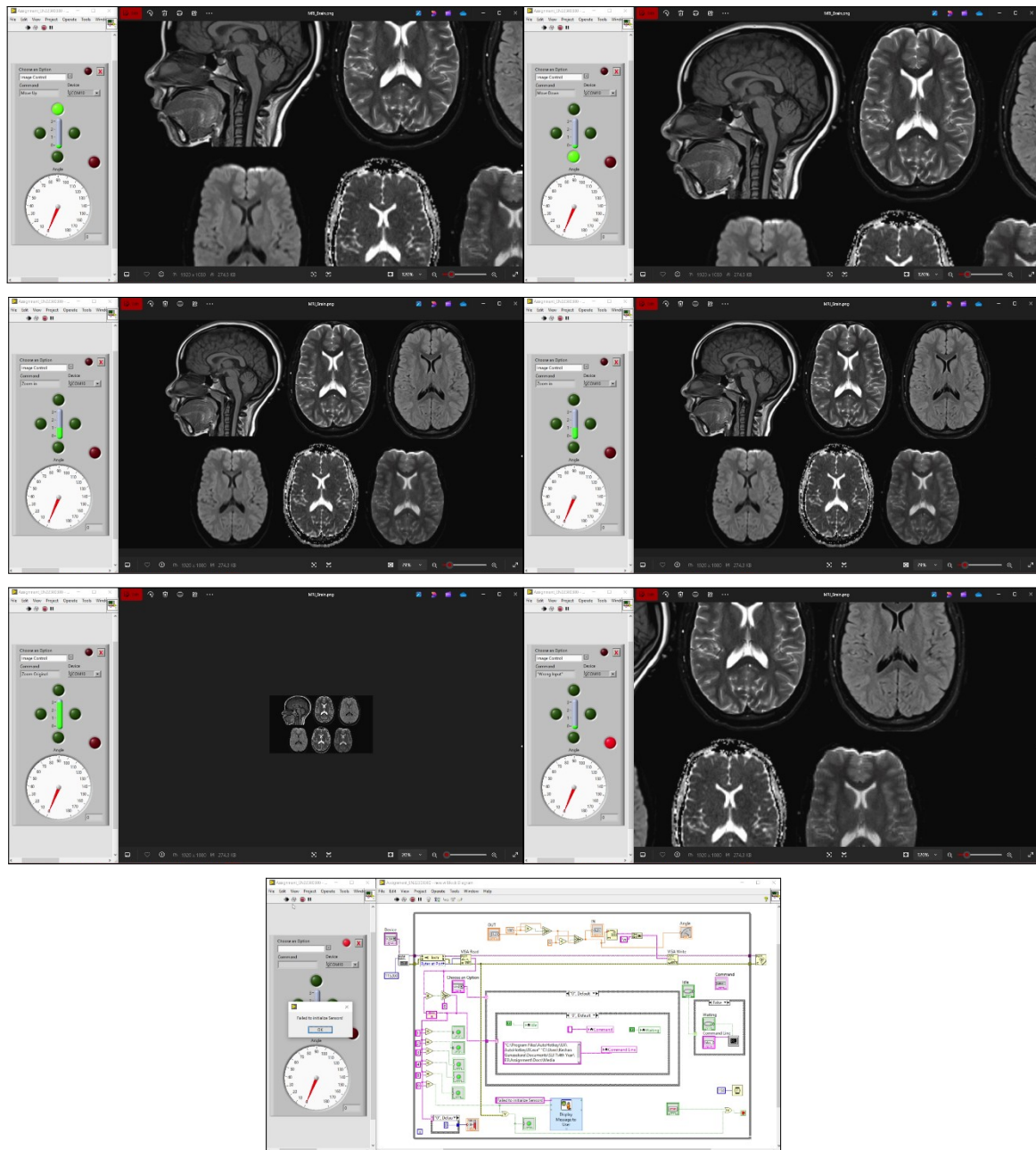


Figure 4: LabVIEW response for each image navigation task: (a) Move right, (b) Move left, (c) Move up, (d) Move down, (e) Zoom-in, (f) Zoom-out, (g) Original zoom ratio, (h) Invalid gesture, and (i) Sensor failure

To evaluate the accuracy and responsiveness of servo motor control, the system was tested by setting the servo to three distinct angles: 20°, 80°, and 180°. For each target angle, both the physical position of the servo on the prototype and the corresponding LabVIEW interface display were documented. The following Figure 5 demonstrates that when an angle command is issued from LabVIEW, the servo promptly moves to the specified position, and the interface confirms the action in real time. This validates the reliable integration between the software and hardware components and highlights the system’s capability for precise, gesture-driven actuation of external devices.



Figure 5: Prototype Servo and LabVIEW response Servo motor controls: (a) 20°, (b) 80°, and (c) 180°

In addition to visual feedback and interface responsiveness, the gesture recognition accuracy was quantitatively evaluated. A structured testing procedure was conducted using the final prototype, where each of the seven supported gestures (four directional swipes and three timed holds) was performed 40 times under controlled conditions. The outcomes were documented to determine the system's recognition accuracy for each gesture type. The results are presented in Table 2 below.

Table 2: Gesture Detection Accuracy

Gesture	Accuracy (%)	Observations
Swipe Up	85	Overall Stable
Swipe Down	85	Overall Stable
Swipe Left	70	Unstable with fast/incomplete gestures
Swipe Right	75	Unstable with fast/incomplete gestures
1s Hold	90	Minor errors in early release
2s Hold	95	Accurate with strong LED feedback
3s Hold	95	Accurate with strong LED feedback

To further assess usability, informal user testing was conducted with five participants, each unfamiliar with the system. Participants were asked to perform all seven gestures in sequence, and their performance was monitored across three trials. Results showed that the average time to correctly complete all gestures decreased from 3.2 minutes to 1.5 minutes, demonstrating a shallow learning curve and effective visual feedback through the LED indicators. Users also reported a high level of confidence in the system's responses, particularly for hold gestures due to the color-coded duration feedback.

These results validate that the gesture-controlled interface performs reliably across different users and the used scenarios, meeting the intended goals of contactless interaction and user-friendliness in sterile environments.

4. Discussion

The development and evaluation of the gesture-controlled embedded interface revealed a number of important insights regarding its functionality, performance, and suitability for use in sterile medical environments. Overall, the system achieved its core objective of providing an intuitive, contactless method for interacting with digital content and external devices, such as servo-actuated tools, without breaching sterility. Through both functional testing and user evaluations, the system demonstrated reliable gesture recognition, real-time responsiveness, and user-friendly feedback mechanisms, establishing its potential for integration into surgical workflows where hygiene and precision are paramount.

Quantitative testing showed in Table 2 indicates that vertical swipe gestures were the most stable and accurate, achieving recognition rates of 85% for upward swipes and 85% for downward swipes. These results can be attributed to the positioning of Sensor 2, which was centrally aligned on the hardware platform and thus provided consistent spatial coverage for vertically oriented hand motions. The system exhibited lower accuracy with horizontal gestures, specifically 70% for left swipes and 75% for right swipes. These gestures relied more heavily on the outer sensors, which introduced variation in detection due to inconsistent hand paths and entry angles. Participants performing horizontal gestures often encountered issues when the motion was either too fast or not centered properly over the sensor array, indicating a need for more robust spatial coverage and filtering to handle diverse user movements.

Hold gestures were found to be the most reliable, with 1-second, 2-second, and 3-second holds achieving 90%, 95%, and 95% accuracy, respectively. These high success rates stemmed from the simplicity of the gesture, requiring the user to remain stationary over a single sensor, and the addition of intuitive LED-based visual feedback, which used a progressive color scheme to indicate hold duration in real time. This direct mapping between visual cue and gesture progress helped users adjust their interaction on-the-fly, significantly reducing the chance of error. Notably, even participants unfamiliar with the system adapted quickly to the hold gestures, highlighting their natural and low-effort usability.

The informal usability testing conducted with five participants further emphasized the system's intuitive learning curve. Users who had never interacted with the prototype were able to learn and perform all seven gestures with increasing speed and confidence. The average time to complete all gestures dropped from 3.2 minutes to 1.5 minutes after just three guided trials, demonstrating that the feedback mechanisms and gesture design were effective in guiding users toward consistent behavior. Most participants reported that the LED feedback made the system feel responsive and trustworthy, especially during hold gestures where timing feedback is critical.

Despite these positive outcomes, several limitations were identified during testing. Horizontal gestures remained somewhat unstable, particularly when executed too rapidly or at a shallow angle relative to the sensor layout. These misclassifications suggest that gesture detection can be affected by variability in human motion, and that additional preprocessing, such as temporal smoothing, trajectory filtering, or gesture confirmation thresholds may be necessary to improve consistency. Another issue was gesture overlaps during rapid successive inputs; if a second gesture was attempted within approximately 500 milliseconds of the first, the system occasionally failed to reset its internal state, leading to dropped or ambiguous detections. This indicates the need for gesture debouncing logic to separate gesture events cleanly, particularly in high-tempo environments such as operating rooms.

The current system architecture relies on USB-based serial communication between the Raspberry Pi Pico and the host computer. While this setup is simple and reliable for development and testing, it limits physical deployment options in clinical settings where wireless operation is more practical and less obtrusive. Introducing Bluetooth or Wi-Fi communication in future versions would significantly improve mobility and ease of installation, enabling seamless integration into various medical workspaces without cable constraints. Moreover, although LabVIEW was ideal for rapid prototyping and demonstration, its licensing model and platform dependence make it less suitable for real-world clinical deployment. Transitioning to a lightweight, cross-platform interface, potentially developed in Python, or through a mobile application would enhance usability, portability, and compatibility with other medical systems.

An additional area of potential expansion is gesture complexity and versatility. The current system supports seven gestures, which are sufficient for basic navigation and control. However, more advanced applications, such as multi-step tool control, multi-user collaboration, or 3D data manipulation may require a broader gesture vocabulary and finer spatial resolution. This could be achieved by integrating more ToF sensors or using hybrid sensor systems that combine ToF with IMUs or capacitive sensing for multi-modal gesture recognition. These improvements would enhance the system's adaptability to different use cases and user preferences, further aligning it with the needs of modern surgical environments.

Overall, the system offers a compelling foundation for contactless, real-time interaction in sterile settings. Its dual-core architecture ensures smooth parallel execution of gesture recognition and servo control, while its user interface delivers responsive, intuitive feedback that accelerates user proficiency. The challenges identified, such as horizontal gesture instability, rapid gesture overlap, and reliance on wired communication present clear, actionable paths for enhancement. With further refinement, this system has the potential to evolve into a highly effective tool for supporting sterile, intelligent, and user-centric workflows in healthcare.

5. Conclusion

This paper presented the successful design and implementation of a compact, low-cost, gesture-controlled embedded interface intended for sterile medical environments. Leveraging a Raspberry Pi Pico microcontroller and VL53L0X Time-of-Flight sensors, the system enabled intuitive, contactless interaction through a set of seven distinct hand gestures, including directional swipes and timed holds. These gestures were effectively mapped to digital commands for image navigation and physical actuation via a servo motor, with the control system managed through a custom LabVIEW interface. The dual-core processing capability of the microcontroller allowed for real-time parallel execution of gesture recognition and servo control tasks, ensuring responsiveness and system stability during operation. Visual feedback provided by RGB LEDs played a key role in enhancing user confidence, conveying real-time gesture validation, hold durations, and system status indicators in a simple and intuitive format.

The system was evaluated through functional testing and informal user trials, demonstrating high accuracy in gesture detection, particularly for vertical and hold gestures, and a short learning curve among first-time users. Although horizontal swipe gestures exhibited slightly lower reliability, the system's overall performance validated

its effectiveness and practicality for sterile environments. Limitations such as sensitivity to fast gestures, gesture overlap, and the use of wired communication highlight valuable areas for improvement. Future iterations will focus on enhancing gesture classification using filtering techniques, introducing wireless connectivity for increased flexibility, and transitioning to a more portable, platform-independent user interface. With its robust performance, ease of use, and scalability, the developed prototype provides a solid foundation for future advancements in contactless control systems for surgical and clinical applications, contributing toward smarter, safer, and more hygienic medical workspaces.

References

- [1] W. Cramblitt. "Biomedical Applications of Embedded Systems." Internet: <https://www.skillsoverpaper.com/blog/biomedical-applications%20of%20embedded%20systems>, Jul., 2023 [Apr. 25, 2025].
- [2] D. Joel. "Advancements in embedded systems: Revolutionizing medical devices." Internet: <https://www.linkedin.com/pulse/advancements-embedded-systems-revolutionizing-medical-dinoo-joel-ssvoc>, Mar. 22, 2024 [Apr. 25, 2025].
- [3] M. G. Jacob, J. P. Wachs and R. A. Packer. (2013, Jun.). "Hand-gesture-based sterile interface for the operating room using contextual cues for the navigation of radiological images." *Journal of the American Medical Informatics Association*. [On-line]. 20(e1), pp. 183-6. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC3715344/>. [Apr. 26, 2025].
- [4] K. Manjula, L. Vyshnavi, R. Thanushree and M. V. Siri. (2024, May). "A Gesture Based Tool for Sterile Browsing of Radiology Images." *International Journal of Advanced Research in Computer and Communication Engineering*. [On-line]. 13(5), pp. 486-92. Available: <https://ijarcce.com/wp-content/uploads/2024/05/IJARCCE.2024.13574.pdf>. [May 01, 2025].
- [5] B. J. Park, T. Jang, J. W. Choi, and N. Kim. (2016, Feb.). "Gesture-controlled interface for contactless control of various computer programs with a hooking-based keyboard and mouse-mapping technique in the operating room." *Computational and Mathematical Methods in Medicine*. [On-line]. 2016(1), p. 5170379. Available: <https://onlinelibrary.wiley.com/doi/10.1155/2016/5170379>. [May 01, 2025].
- [6] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith and J. Handler. (2006, Jan.). "A real-time hand gesture interface for medical visualization." *Advances in Intelligent and Soft Computing*. [On-line]. 36, pp.153-62. Available: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f99df4fb2395d02f1ae6506302de976c79546061>. [May 08, 2025].
- [7] STMicroelectronics, "Time-of-flight (ToF) ranging sensor," Internet: <https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html>. [May. 08, 2025].

- [8] L. A. Schwarz, A. Bigdelou and N. Navab. (2011) "Learning gestures for customizable human–computer interaction in the operating room." *Medical image computing and computer-assisted intervention*. [On-line]. 14, pp. 129-36. Available: <https://pubmed.ncbi.nlm.nih.gov/22003609/>. [May 08, 2025].
- [9] muftawu. "Servo motor control using hand gesture with computer vision." Internet: <https://www.hackster.io/muftawu/servo-motor-control-using-hand-gesture-with-computer-vision-b3edd4>, Oct. 12, 2021 [May 10, 2025]