



Real-Time Hand Gesture-Based Virtual Mouse System Using ESP32-CAM and OpenCV

Muhammad Farhan Alghifari Chaniago Saputro¹, Sugondo Hadiyoso^{2*}, Indrarini Dyah Irawati³, Achmad Rizal⁴

^{1,2,3}School of Applied Science, Telkom University, Bandung, Indonesia

⁴School of Electrical Engineering, Telkom University, Bandung, Indonesia

allghifari.farhan@gmail.com¹, sugondo@telkomuniversity.ac.id², indrarini@telkomuniversity.ac.id³

Abstract

This research develops a virtual mouse control system that uses real-time hand gesture recognition implemented on an ESP32-CAM-based Internet of Things (IoT) platform. By leveraging OpenCV for image processing, the system translates hand gestures into corresponding mouse actions, including cursor movement, clicking, and scrolling. The study evaluates system performance under different lighting conditions and Wi-Fi speeds. Results show that higher Wi-Fi speeds significantly reduce latency, enabling smoother real time gesture recognition and high definition video output, while lower speeds lead to noticeable delays and reduced accuracy. The system successfully enables remote cursor control through camera captured hand gestures, supporting functions such as left click, right click, scrolling, and dragging. In latency tests performed with an internet speed of approximately 60 Mbps, the system achieved an average delay of about 50 milliseconds. Under optimal lighting conditions with minimal background interference, it accurately tracked hand movements and recognized gestures such as pointing, clicking, dragging, and scrolling in real time, achieving an accuracy rate of 95%. Despite its lower resolution compared to conventional webcams, the ESP32-CAM proves to be an effective solution for virtual mouse control, particularly in scenarios where high-resolution imaging is unnecessary. Its IoT capabilities support remote operation, allowing users to control the virtual mouse from a distance as long as both the ESP32-CAM and the computer remain connected. Overall, the findings highlight the ESP32-CAM based IoT platform as a viable alternative for gesture based interaction in real applications, although further enhancements are needed to improve performance in challenging environments.

Keywords: hand gesture recognition; image processing; IoT; real-time; virtual mouse control

How to Cite: M. Farhan Alghifari Chaniago Saputro, S. Hadiyoso, and I. Dyah Irawati, "Real-Time Hand Gesture-Based Virtual Mouse System Using ESP32-CAM and OpenCV", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 10, no. 1, pp. 53 - 61, Feb. 2026.

Permalink/DOI: <https://doi.org/10.29207/resti.v10i1.6609>

Received: April 30, 2025

Accepted: January 1, 2026

Available Online: February 16, 2026

This is an open-access article under the [CC BY 4.0 License](#)

Published by [Ikatan Ahli Informatika Indonesia](#)

1. Introduction

Rapid advancements in technology have led to an increasing demand for more intuitive and efficient human-computer interaction (HCI) methods [1]. Traditional input devices, such as keyboards and mouse, have long been the standard for interacting with computers; however, these devices are often cumbersome and can limit the user's interaction and flexibility [1], [2]. The limitations of these devices have prompted researchers and engineers to explore alternative HCI systems that can provide natural, intuitive, and accessible interfaces [3]. One promising alternative is hand gesture-based interaction, where users control devices by detecting and interpreting hand movements [4]- [6]. This approach eliminates the need for physical contact with traditional input devices and

offers a more engaging and seamless experience [1], [4], [7], [8].

Hand-gesture recognition systems have evolved over the years, and their applications have expanded across various fields, including gaming, accessibility tools, robotics, and virtual reality [9], [10]. The core challenge in these systems is to capture hand movements and translate them into meaningful commands for the computer [11]. Earlier systems primarily relied on external cameras or specialized hardware, such as depth sensors or motion-tracking devices, to capture hand gestures [12], [13]. Although these technologies have shown significant promise, they often come with high costs, complexity, and substantial hardware requirements, making them less practical for widespread use in everyday environments.

In recent years, advances in computer vision and machine learning have enabled the development of more accessible and adaptable hand-gesture recognition systems. Among these developments, ESP32-CAM, an Internet of Things (IoT) device, has gained attention as a compact and capable platform for hand gesture-based control [14]. ESP32-CAM integrates a microcontroller with a built-in camera, offering a small form factor and sufficient processing capabilities that make it suitable for real-time applications in HCI [3], [15], [16]. This device presents an opportunity to create lightweight and easy-to-integrate systems for virtual hand gesture control, without the need for expensive external cameras or high-power computing resources [17].

Although several studies have explored the potential of hand gesture recognition systems using various sensors and devices, few have investigated the feasibility of using devices such as ESP32-CAM for this purpose. Commonly developed applications are still limited to the use of standalone cameras installed on a PC. Therefore, this research aims to fill this gap by evaluating the use of ESP32-CAM for virtual hand gesture mouse control. Specifically, this study explores the effectiveness of ESP32-CAM in recognizing and translating hand gestures into mouse actions such as pointing, clicking, and scrolling [1], [7], [18]. A comparative analysis will also be conducted between the ESP32-CAM-based system and more conventional webcam-based gesture recognition systems, focusing on performance factors, such as accuracy, responsiveness, ease of implementation, and system efficiency [18].

An essential aspect of this study is to investigate the practical application of ESP32-CAM in real-world environments. While much of the existing literature on hand gesture recognition has focused on controlled experimental setups, few studies have assessed the system's performance in dynamic everyday settings. Therefore, this study also explores factors such as lighting conditions, background noise, and the impact of network speed on the system's performance [15] [19]. One of the key factors that could influence the performance of the system is the Wi-Fi connection, as ESP32-CAM transmits video data to the laptop for processing via Wi-Fi. Faster Wi-Fi speeds are expected to reduce latency and improve the real-time processing of gestures, whereas slower connections can introduce delays and affect the accuracy of gesture recognition.

This study aims to provide a deeper understanding of the potential of compact low-cost IoT devices such as ESP32-CAM for enhancing human-computer interaction through hand gesture recognition. By exploring the strengths and limitations of using embedded hardware for virtual mouse control, this study provides valuable insights into how such systems can be integrated into everyday applications [8]. Whether in educational settings, accessibility tools, or DIY projects, the findings of this research could help broaden the use of gesture-based interfaces and provide

an efficient alternative to traditional input devices [1], [20].

2. Methods

This study adopts a mixed-method approach that integrates both hardware and software components to develop a real-time hand gesture recognition system based on an ESP32-CAM microcontroller [1], [20], [21]. The methodology is divided into two primary components: an embedded system using ESP32-CAM for video capture and streaming, and an image processing module implemented in Python using OpenCV for gesture recognition and virtual mouse control [4], [7].

The ESP32-CAM functions as the core hardware component, capturing continuous video of hand gestures and streaming footage through a web server hosted using its built-in Wi-Fi capability [17]. This live video stream is accessed on a laptop or PC connected to the same Wi-Fi network, serving as the processing unit for interpreting hand gestures [1].

On the software side, the streamed video was processed using Python and OpenCV library [6], [18]. The image-processing workflow involves detecting the presence of hands and identifying the finger positions within each video frame [20]. Specific hand gestures are then mapped to corresponding virtual mouse actions such as cursor movement and clicking [1], [7], [18]. The system is designed to interpret simple and intuitive hand gestures to wirelessly control an on-screen mouse in real-time [4], [21], [22].

To evaluate the performance of the system, several key metrics were considered, including the gesture recognition accuracy, responsiveness (latency), image quality, and overall usability. Tests were conducted under various lighting conditions and Wi-Fi speeds, to simulate real-world scenarios [15]. This approach allows for a comprehensive assessment of the ESP32-CAM's capability as a cost-effective solution for gesture-based human-computer interactions [1], [3].

2.1 System Overview

The core system consists of ESP32-CAM, which plays a critical role in capturing the video feed of the user's hand gestures. The ESP32-CAM is equipped with a built-in camera and microcontroller that processes the video feed locally before sending it to a connected device. To enable programming and data transmission between the ESP32-CAM and laptop, the microcontroller was connected via an FTDI module. This module provides the necessary interface for uploading control algorithms and transmitting processed data for further action. The system was designed to track specific hand gestures, such as pointing, clicking, and scrolling, by analyzing the visual input provided by the camera.

The OpenCV library running on the laptop was used to process the video feed from ESP32-CAM [8]. OpenCV

provides a robust suite of tools for image processing, enabling the system to identify and classify hand gestures based on predefined patterns [4]. The library's machine learning and computer vision algorithms allow real-time processing of the video feed, enabling the system to recognize hand movements with low latency. Once a gesture is recognized, the system translates it into corresponding mouse actions, such as moving the cursor, simulating a click, or scrolling through the content on the screen [8], [18].

The hand-gesture recognition process begins by detecting the position and movement of the user's hand within the video feed [22]. Using OpenCV's image-processing techniques, the system isolates the hand from the background, analyzes its contours, and identifies key features, such as the number of fingers extended or the position of the hand relative to the camera. These features are then mapped to specific actions, such as moving the mouse pointer (based on the position of the hand) or clicking (based on specific gestures, such as a fist or finger tap) [15].

2.2. Hardware Components

The hardware components used in this study are listed in Table 1. These components were carefully selected for their compatibility and ability to function efficiently within the proposed system. The central piece of the system is ESP32-CAM, a microcontroller with a built-in camera that captures real-time video feeds of the user's hand gestures [6]. This device processed the video data and sent it to a laptop for further analysis.

The FTDI module plays a vital role in programming the ESP32-CAM and facilitating data transmission between the microcontroller and the laptop. This ensures seamless communication and allows the uploading of gesture recognition algorithms to ESP32-CAM. Additionally, a micro-USB cable is required to power the ESP32-CAM and establish a connection between the ESP32-CAM and the laptop during programming and data transmission. Jumper cables are used to make various necessary connections, such as linking the FTDI module to the ESP32-CAM and other essential wiring for the circuit setup. Each component plays a specific role in ensuring that the system operates smoothly and effectively. Table 1 provides a detailed breakdown of the components and their respective quantities.

Table 1. Hardware Components

S.N.	Component	Quantity
1	ESP32-CAM	1
2	FTDI Module	1
3	Micro-USB Cable	1
4	Jumper Wires	10

2.3 Hardware Setup

The primary hardware components for this project included the ESP32-CAM, FTDI module, USB cables, and jumper wires. The ESP32-CAM serves as the microcontroller with a built-in camera, which is responsible for capturing real-time videos of the user's

hand gestures. It processed the video feed and transmitted it for further analysis on a connected laptop. The FTDI module plays an essential role in programming the ESP32-CAM and facilitating communication between the microcontroller and the laptop. It ensures proper data transmission and allows the uploading of gesture recognition algorithms to ESP32-CAM. USB cables were used to connect the FTDI module to the laptop, providing both power and data transfer capabilities. Jumper wires are utilized to make the necessary connections between the components, such as connecting the FTDI module to the ESP32-CAM for programming and data exchange.

The wiring diagram in Figure 1 illustrates the interconnection of these components. The ESP32-CAM is connected to the FTDI module, which is, in turn, connected to the laptop. The camera on the ESP32-CAM captures the video feed of hand movements, and the FTDI module facilitates data flow between the microcontroller and the laptop. Once the video data are received, they are processed using OpenCV for hand gesture detection. This setup ensures that ESP32-CAM can capture and process hand gestures effectively, translating them into mouse actions in real time [23]. The use of jumper wires ensured that the components were correctly connected to enable smooth communication and functionality during the research process. The entire system, from the ESP32-CAM to the FTDI module and laptop, works together to enable the recognition and translation of hand gestures into meaningful computer commands, such as mouse movements and clicks [8].

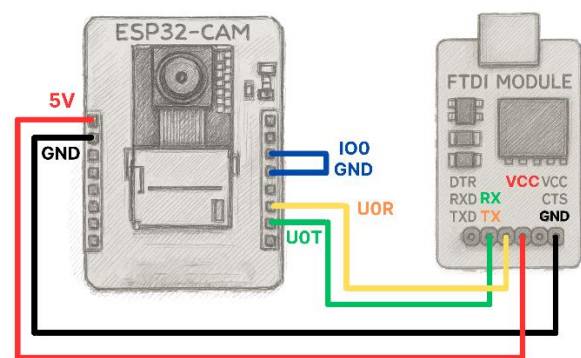


Figure 1. Wiring Diagram of ESP32-CAM and FTDI Module

2.4. System Block Diagram

The system processes the captured video data and identifies hand gestures using OpenCV [6]. The block diagram in Figure 2 provides an overview of the architecture of the system, illustrating the flow of data from the ESP32-CAM to the Wi-Fi module, which communicates with a laptop for gesture recognition. The ESP32-CAM, which acts as the primary input device, captures real-time videos of the user's hand gestures using its built-in camera. These video data are transmitted to the Wi-Fi module, which then sends the information to the laptop for further processing.

Once the data reached the laptop, they were processed using OpenCV, an open-source computer vision library. OpenCV analyzes the video feed to detect and recognize specific hand gestures such as pointing, clicking, and scrolling [7], [18]. The recognized gestures are then translated into corresponding mouse actions such as moving the cursor, simulating clicks, or scrolling through content [4], [24].

The system architecture ensures that each component works in harmony to efficiently process video data. The ESP32-CAM and FTDI modules handle video capture and data transmission, whereas the Wi-Fi module facilitates communication between the ESP32-CAM and laptop. Once the video feed reached the laptop, OpenCV performed gesture recognition and provided the output as mouse movements or clicks.

This flow of data from input to processing and output is crucial for a real-time hand gesture recognition system, ensuring smooth interaction with the computer without the need for traditional input devices [7], [8]. The modular design of the system also allows for potential scalability, enabling additional gestures or input methods to be incorporated in future iterations.

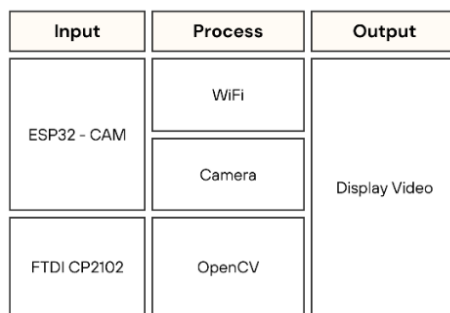


Figure 2. System Block Diagram

2.5 Workflow Diagram of the Application

The workflow diagram in Figure 3 outlines the steps involved in the hand gesture mouse control application using ESP32-CAM. The process begins with the start step, where ESP32-CAM is initially connected to the system. The next step checks for an active Wi-Fi connection, ensuring that ESP32-CAM is properly linked to the network [21]. If the connection is offline, the system stops and waits for a proper connection, thereby ensuring that the device can properly transmit data to the laptop. Once the Wi-Fi connection is established, ESP32-CAM is turned on and starts capturing the video feed of the user's hand movements.

The Hand Detection step follows, in which the system uses OpenCV to process the video data and detect hand gestures in real time. If hand gestures are detected, the system displays the video to the user. At this stage, the system continuously analyzes hand movements and translates them into mouse actions, such as cursor movement, clicking, or scrolling. The process continues until the application is finished, marking the end of the hand gesture recognition process.

This workflow ensures that the system operates smoothly by checking connectivity, capturing videos, detecting gestures, and providing real-time feedback to the user [25]. This highlights the key steps required for ESP32-CAM to function effectively in controlling a virtual mouse through hand gestures [5]. The diagram provides a clear overview of the sequential steps and decision points that constitute the hand-gesture-based control system.

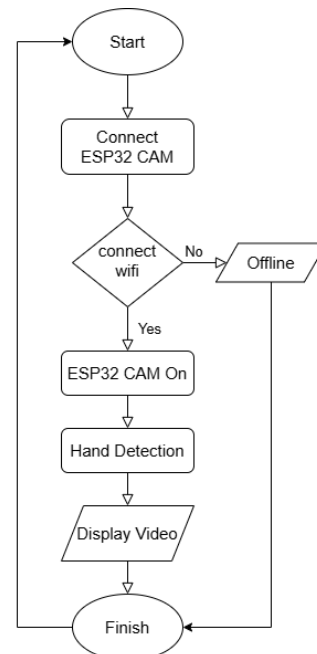


Figure 3. Flowchart of the system

2.6 Hand Gesture Recognition Algorithm

The hand-gesture recognition algorithm starts with the start step, where the system begins by detecting the user's hand [20]. The first step was to show the hand to the camera and initiate the gesture recognition process. The system then evaluates the position of the hand and number of fingers raised to identify the corresponding gesture. The flowchart shows the multiple decision points, each corresponding to a different hand gesture.

If the index finger is raised, the system interprets it as a left-click gesture [20]. This triggered the mouse to perform a left-click action [18]. If the middle finger was raised, it was recognized as a right-click action, prompting the system to simulate a right-click action. If five fingers are raised, the system detects this gesture as a drag mouse action, which is used to drag objects on the screen.

The flowchart further indicates that if both the middle and index fingers are raised simultaneously, this is recognized as the cursor movement gesture, which controls the movement of the mouse pointer on the screen. This gesture allows the user to move the cursor to different locations.

Once the system recognizes these gestures, the corresponding actions are processed and rendered. The rendering step involves updating the display with

appropriate mouse actions such as cursor movement, clicking, or dragging. After performing these actions, the system displays the result to the user, showing the updated state of the screen based on the hand gestures.

This process continues until the finish point is reached, marking the completion of the hand gesture recognition and mouse control tasks [26]. Figure 4 presents a flowchart that effectively illustrates how the system interprets hand gestures and translates them into meaningful mouse actions, enabling users to interact seamlessly with a computer using only hand gestures [27].

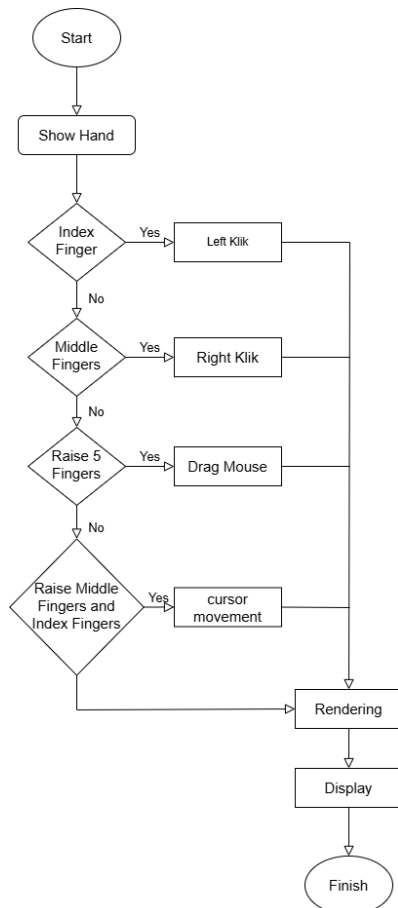


Figure 4. Flowchart for Hand Gesture Mouse Control

2.7 Implementation Steps

The implementation process began by uploading firmware to the ESP32-CAM using the Arduino IDE. The device was configured to automatically connect to a Wi-Fi network and to stream video via an internal HTTP server. After successful programming, the ESP32-CAM was powered and placed in a stable position to capture the hand gestures. The video stream, accessible through a specific URL endpoint (e.g., /cam-hi.jpg), served as the input source for the image processing system on the host computer [6], [17].

On the software side, two Python scripts were developed to handle the image processing and gesture-based mouse control. The first script, *deteksitangan.py*, was responsible for detecting and analyzing hand

landmarks using the MediaPipe library and extracting positional data for each finger. The second script, *virtualmouse.py*, captures the video stream from ESP32-CAM over HTTP, processes the images using OpenCV, and determines hand gestures based on landmark configurations. These gestures were then mapped to the corresponding mouse actions using Autopy and PyAutogui. For example, a cursor movement is triggered when the index and middle fingers are raised, while left- or right-click actions are executed depending on which single finger is extended. The two scripts operate in tandem to form a modular system that is easy to maintain and expand.

To run the system, the Python environment was configured using several essential libraries, including cv2, mediapipe, numpy, time, math, requests, and urllib. The application fetches frames from the ESP32-CAM in real time, applies image-processing techniques to detect hand positions, interprets gestures, and simulates mouse inputs accordingly. This process runs continuously, providing smooth and contactless control of the mouse pointer. For better stability, additional functions such as movement smoothing and frame rate monitoring were also implemented. The system is lightweight, runs locally on a standard laptop, and requires only a stable Wi-Fi connection shared with an ESP32-CAM device.

3. Results and Discussions

The system performance was evaluated through a series of experiments to assess its accuracy, responsiveness, image quality, and usability under diverse conditions. Tests were conducted in both controlled environments and dynamic real condition settings, where key variables, including lighting, background interference, and Wi-Fi speed, were systematically adjusted to measure their impact on the system performance.

3.1 Test Results

The system was tested for its accuracy and responsiveness for the detection of hand gestures. Figure 5 illustrates the real-time detection of hand gestures [5]. In the image, the system detects and tracks the position of the hand using the 3D coordinates of the hand landmarks displayed on the terminal. The pink dots represent hand landmarks, and the green frame highlights the detected hand, providing a clear visualization of the hand's detection process. This result highlights the system's ability to track hand gestures accurately and provide feedback through coordinates [4].

The quality of the video feed captured by ESP32-CAM was assessed by analyzing the resolution, clarity, and ability of the system to track hand movements. ESP32-CAM was capable of producing clear video feeds under ideal lighting conditions. The resolution of the captured video was 640×480 pixels, which is sufficient for recognizing simple hand gestures such as pointing, clicking, and scrolling.

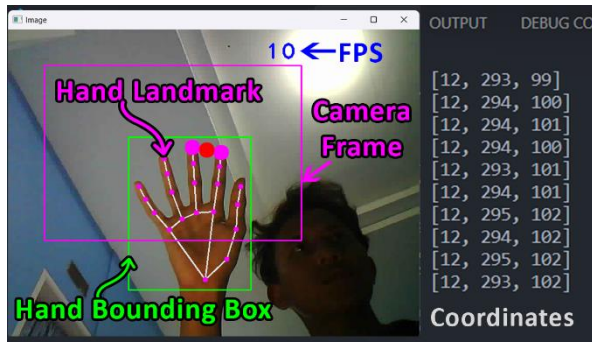


Figure 5. Hand Gesture Detection with Coordinates

Additionally, Figure 6 demonstrates the system's real-time hand gesture processing, with frames per second (fps) displayed in the top-left corner of the video feed. The fps serves as a critical performance metric, reflecting the processing speed and the ESP32-CAM's efficiency in handling video data. A higher fps indicates greater responsiveness, which is essential for real applications that require seamless real time tracking and control. In this case, the system achieved a strong performance, confirming that ESP32-CAM can process gestures efficiently without perceptible delays.

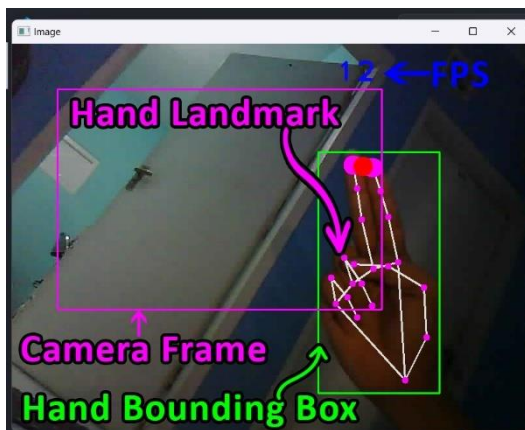


Figure 6. Real-Time Hand Gesture Detection with FPS Display

Figure 6 illustrates the system's ability to track hand landmarks and provide real time feedback. In the first image, the system accurately followed the hand across different positions and dynamically updated the corresponding 3D coordinates. These coordinates, displayed in the terminal, offer valuable insights into the internal processing of the system, revealing how hand positions and movements are precisely mapped.

The real-time video feed in Figure 6, complete with fps overlay, provides quantitative evidence of the processing efficiency of the system. Analysis of the fps data confirms stable performance in gesture recognition, achieving the necessary speed accuracy balance for practical implementation in virtual mouse applications, where responsiveness directly correlates with usability [4].

The system demonstrated very low latency under high-speed Wi-Fi conditions, with an average delay of approximately 50 ms [20], [5]. This low latency indicates that the system can process hand gestures in

real time without noticeable delays, making it suitable for interactive applications [18]. However, when tested under low speed Wi-Fi conditions (5 Mbps), the latency increased significantly, with average delays of approximately 200 ms. This increase in latency was particularly noticeable during fast hand movements, when the system struggled to maintain smooth control of the cursor, especially during high-speed gestures such as scrolling.

Figure 7 provides empirical evidence of how network quality and lighting affect system performance by comparing high-speed (>20 Mbps) and low speed Wi-Fi conditions under adequate lighting. The observed 31 fps (displayed in the upper left corner of the frame) confirms that faster internet speeds enable smoother video processing, lower latency, and consequently, more accurate gesture detection and execution.

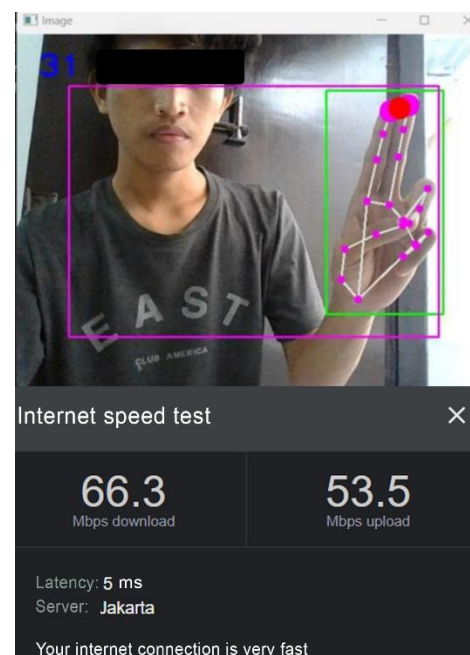


Figure 7. Hand Gesture Detection with High Internet Speed

When operating on slower Wi-Fi (~5 Mbps, Figure 8), the system's performance drops to 12 fps, causing a visible lag during rapid gestures. The comparison also highlights how better lighting (Figure 7 vs. 6) enhances detection accuracy by improving hand background contrast and reducing noise. These results demonstrate that both network speed and lighting quality are crucial for optimal gesture recognition performance.

Furthermore, one of the key advantages identified in this study is the ability of the ESP32-CAM to support remote hand gesture recognition, provided that both the ESP32-CAM and the controlling device (e.g., laptop or PC) are connected to the same Wi-Fi network (SSID) [4], [7]. This feature allows users to perform hand gestures from a distance without the need for a direct wired connection between the devices [7], [18]. It adds flexibility to the system, making it suitable for use in environments such as classrooms, smart homes, and public presentations, where remote and contactless

interactions are advantageous. This capability enhanced the overall usability and application range of the ESP32-CAM based virtual mouse system in real-world scenarios.

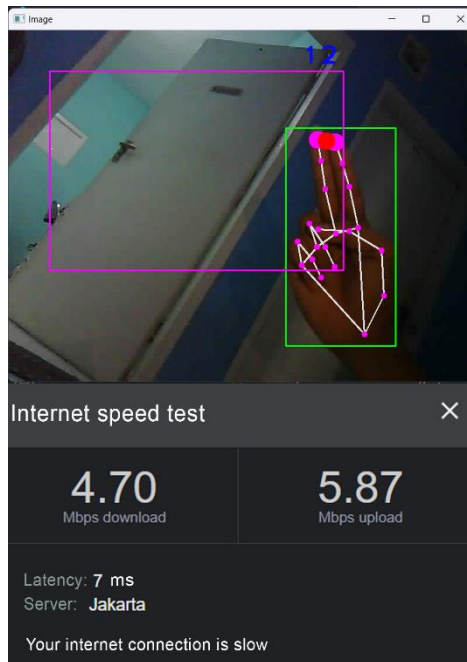


Figure 8. Hand Gesture Detection with Low Internet Speed

This image demonstrates the system's real-time processing of hand gestures, with the fps value displayed on the top-left corner of the video frame, demonstrating the system's capability to process videos efficiently.

Under optimal conditions, where the lighting was consistent and there was minimal background interference, the system achieved a high accuracy rate of 95% [20]. ESP32-CAM effectively recognized gestures such as pointing and left clicking with near perfect accuracy. However, the accuracy decreased under non ideal conditions, such as when the background was cluttered or when lighting was poor. Under low light conditions, the system's accuracy dropped to approximately 85%, and in environments with significant background noise, the accuracy was further reduced to approximately 80%. These findings highlight the critical importance of lighting and background conditions in achieving optimal gesture recognition performance.

Responsiveness and latency were evaluated by measuring the time taken by the system to recognize a gesture and translate it into a corresponding mouse action (e.g., moving the cursor or clicking) [20]. The time delay was measured from the moment the user performed the gesture, when the corresponding action was registered on the screen.

3.2 Discussions

Performance tests confirm the ESP32-CAM system's 95% gesture recognition accuracy under ideal conditions (stable lighting and clean background).

While limited by resolution, its real time tracking capability for basic gestures (pointing, clicking, scrolling) makes it particularly suitable for cost sensitive, educational, and DIY applications, where high end performance is unnecessary [7].

However, the performance of the system is significantly influenced by environmental conditions, particularly by lighting. Under optimal lighting conditions, where the user's hand was well lit and clearly visible, ESP32-CAM performed excellently with fast response times and high accuracy. The system successfully translated hand gestures into the corresponding mouse actions without any noticeable delay. However, when the lighting worsened, especially in low-light environments, the performance of the system deteriorated. The video feed became grainy, making it difficult for the system to accurately detect and track hand contours. The lower resolution and limited processing capabilities of ESP32-CAM in low light conditions caused a decrease in recognition accuracy and an increase in latency, as the system struggled to identify hand landmarks and movements. These findings are consistent with the limitations observed in webcam-based systems, which struggle in poorly lit settings [18].

In real scenarios, where users may not have full control over lighting, this reliance on good lighting conditions may limit the system's effectiveness. To address this issue, future iterations of the system could benefit from additional lighting solutions such as built-in infrared lighting or other illumination methods, as well as advanced image enhancement techniques to improve hand gesture recognition in low-light environments. Additionally, the system can be improved by incorporating machine learning algorithms that adapt to different lighting conditions and improve the recognition of gestures over time.

Background interference also had a noticeable impact on system performance. In environments with minimal background clutter, the system performs well and accurately detects and tracks hand gestures. However, in more dynamic or cluttered settings, the system struggled to distinguish the hand from the background, leading to misdetection or failure to recognize gestures. This challenge is particularly noticeable when the hand is close to or blends in with the background, making it difficult for ESP32-CAM to differentiate between the hand and the surroundings. This highlights the need for a system that is more robust to background interference.

One potential solution to this problem could be the implementation of background subtraction techniques or more advanced machine-learning models capable of better differentiating the hand from the background, even in complex environments. These enhancements could help improve the accuracy of the system in dynamic and varied settings, where background noise and clutter are more prominent [4].

Furthermore, the performance of the system is significantly influenced by the speed and stability of the Wi-Fi connection. Wi-Fi speed plays a critical role in determining the responsiveness of the system and overall performance. High speed Wi-Fi (over 20 Mbps) enables smooth, real time gesture recognition with minimal latency and a high quality video output. In contrast, slower Wi-Fi connections (below 10 Mbps) introduced noticeable delays, pixelation, and frame drops, which negatively affected the system's ability to process gestures quickly. This dependence on a fast and stable Wi-Fi network limits the system's performance in environments with poor network infrastructure, where latency and network instability can disrupt user experience.

This finding suggests that future improvements could focus on optimizing data transmission methods to reduce reliance on Wi-Fi speed. Incorporating additional hardware enhancements, such as faster Wi-Fi modules or alternative communication protocols, can help mitigate the effects of slow network speed. Additionally, exploring offline processing methods or edge computing solutions could help reduce the dependence of the system on a continuous network connection and improve performance in more challenging network environments.

In conclusion, while the ESP32-CAM based gesture recognition system demonstrates promising results in controlled environments, it faces challenges when operating under suboptimal conditions, particularly in low-light settings, cluttered backgrounds, and slow Wi-Fi speeds [5]. Despite these limitations, the system offers a cost effective solution for hand gesture recognition, making it suitable for specific applications in which high-resolution video and real-time performance are not essential [7], [18]. Future developments should focus on enhancing the robustness of the system to environmental factors, improving its performance under low light conditions, and optimizing its reliance on network connectivity to make it more adaptable for real world applications.

4. Conclusions

This research successfully demonstrated the feasibility of using ESP32-CAM, a microcontroller with an integrated camera, for real-time hand gesture recognition and virtual mouse control. This study explored the system's ability to detect and translate hand gestures into corresponding mouse actions, such as pointing, clicking, and scrolling. The results indicate that ESP32-CAM is capable of performing well in controlled environments, where factors such as lighting and background conditions are optimal. Under these conditions, the system achieved high accuracy, low latency, and smooth performance, demonstrating its potential as an alternative to traditional input devices, such as keyboards and mouse.

However, this study also highlights some critical limitations of the ESP32-CAM-based system,

particularly in non-ideal environments. The accuracy of gesture recognition was significantly affected by poor lighting conditions and complex backgrounds, where the system struggled to distinguish between the hand and the surroundings. In such scenarios, the system performance decreases, with a notable increase in latency and a reduction in image quality. These challenges underscore the need for improved system robustness to effectively handle dynamic environments. Future iterations of the system could benefit from additional lighting solutions, background subtraction techniques, and enhanced image-processing algorithms to mitigate the effects of suboptimal lighting and complex backgrounds. Another important finding from this research is the significant role played by Wi-Fi speed in determining the responsiveness and overall performance of the system. High-speed Wi-Fi connections result in smooth gesture recognition and minimal latency, enabling real-time interaction with the virtual mouse. In contrast, slower Wi-Fi connections introduce noticeable delays and reduce the quality of the video feed, leading to less accurate gesture recognition and slower response times. These findings emphasize that for the ESP32-CAM-based gesture recognition system to perform optimally, a fast and stable network connection is essential. This reliance on the Wi-Fi speed could pose limitations in environments with poor network infrastructure, where latency could become a disruptive factor for real-time applications.

In future work, several opportunities will exist to improve the system's performance and expand its capabilities. One potential area of improvement is machine learning. By integrating machine-learning models into the system, it can learn to recognize a wider range of hand gestures, adapt to different lighting conditions, and improve its accuracy over time. Machine-learning models can also help enhance the robustness of the system in environments with varying backgrounds or complex settings. This approach enables the system to handle dynamic environments better, making it more suitable for real-world applications where lighting and background conditions are not always ideal. The integration of more advanced sensors, such as depth sensors and infrared cameras, can also provide more detailed and accurate hand gesture recognition, particularly for more complex interactions. These sensors can capture depth information and enable the system to detect hand gestures more accurately, even in cluttered or poorly lit environments.

Acknowledgements

The authors gratefully acknowledge the support and resources provided by the Center of Excellence in Green Technology (CoE GT), Telkom University, which were instrumental in conducting this research. The facilities and research environment provided by CoE GT significantly contributed to the successful completion of this study.

References

- [1] M. V. A, "Hand Gesture Recognition System as Virtual Mouse for HCI," *International Journal for Research in Applied Science and Engineering Technology*, vol. IX, no. 8, pp. 1098-1101, 2021. <https://doi.org/10.22214/ijraset.2021.37503>
- [2] B. Šumak, M. Špindler, M. Debeljak, M. Heričko and M. Pušnik, "An empirical evaluation of a hands-free computer interaction for users with motor disabilities," *Journal of Biomedical Informatics*, vol. XCVI, no. 1, p. 103249, 2019. <https://doi.org/10.1016/j.jbi.2019.103249>
- [3] M. MK and M. MS, "Virtual Mouse And Keyboard For Computer Interaction," *international journal of engineering technology and management sciences*, vol. VI, no. 4, pp. 205-213, 2022. <https://doi.org/10.46647/ijetms.2022.v06i04.0036>
- [4] G. Bhole, D. Bhingare, R. Bhise, S. Bhegade, S. Bhokare and A. Bhosle, "System Control using Hand Gesture," in *2023 International Conference for Advancement in Technology, ICONAT 2023*, Goa, 2023. <https://doi.org/10.1109/ICONAT57137.2023.10080493>
- [5] S. S. Chakravarthi, B. N. K. Rao, N. P. Challa, R. Ranjana and A. Rai, "Gesture Recognition for Enhancing Human Computer Interaction," *Journal of Scientific and Industrial Research*, vol. LXXXII, no. 4, 2023. <https://doi.org/10.56042/jsir.v82i04.72387>
- [6] D. L. Falak, M. Khandait, S. M. Koteswar, M. Upari and U. Singh, "Virtual Mouse Using Hand Gestures," *International Research Journal of Modernization in Engineering Technology and Science*, vol. IV, no. 11, pp. 81-85, 2022. <https://doi.org/10.56726/irjmets30967>
- [7] R. Annachhatre, M. Tamakuwala, P. Shinde, A. Jain and P. V. Kulkarni, "Virtual Mouse Using Hand Gesture Recognition - A Systematic Literature Review," in *2022 IEEE International Conference on Blockchain and Distributed Systems Security, ICBDS 2022*, Pune, 2022. <https://doi.org/10.1109/ICBDS53701.2022.9935944>
- [8] N. Meghana, K. S. Lakshmi, M. N. L. Tejasree, K. Srujana and N. Ashok, "Gesture-Based Human-Computer Interaction," *EPRA International Journal of Research & Development (IJRD)*, vol. VIII, no. 10, pp. 237-241, 2023. <https://doi.org/10.36713/epra14757>
- [9] C. Panagiotou, E. Faliagka, C. P. Antonopoulos and N. Voros, "Multidisciplinary ML Techniques on Gesture Recognition for People with Disabilities in a Smart Home Environment," *AI*, vol. VI, no. 1, pp. 1-17, 2025. <https://doi.org/10.3390/ai6010017>
- [10] V. Chang, R. O. Eniola, L. Golightly and Q. A. Xu, "An Exploration into Human-Computer Interaction: Hand Gesture Recognition Management in a Challenging Environment," *SN Computer Science*, vol. IV, p. 441, 2023. <https://doi.org/10.1007/s42979-023-01751-y>
- [11] S. Tirpude, D. Naidu, P. Rajkarne, S. Sarile, N. Saraf and R. Maheshwari, "Realtime Hand Gesture Recognition System for Human Computer Interaction," *International Journal of Next-Generation Computing*, vol. XIV, no. 1, pp. 264-269, 2023. <https://doi.org/10.47164/ijngc.v14i1.1097>
- [12] E. Fifi, E. Castrillón, G. Stöttinger, M. P. Cuellar and D. P. Morales, "Hand Gesture Recognition on Edge Devices: Sensor Technologies, Algorithms, and Processing Hardware," *Sensors*, vol. XXV, no. 6, p. 1687, 2025. <https://doi.org/10.3390/s25061687>
- [13] M. M. Rahman, A. Uzzaman, F. Khatun, M. Aktaruzzaman and N. Siddique, "A comparative study of advanced technologies and methods in hand gesture analysis and recognition systems," *Expert Systems with Applications*, vol. CCLXVI, p. 125929, 2025. <https://doi.org/10.1016/j.eswa.2024.125929>
- [14] V. L. Bhale, N. L. Sumnure, H. S. Maske and D. S. S. Patil, "Robot Car Using ESP32 Cam," *International Journal of Advanced Research in Science, Communication and Technology (IJARST)*, vol. III, no. 1, pp. 494-497, 2023. <https://doi.org/10.48175/IJARST-11280>
- [15] V. V. T. Reddy, T. Dhyanchand, G. V. Krishna and S. Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition," in *Proceedings of 2020 IEEE-HYDCON International Conference on Engineering in the 4th Industrial Revolution, HYDCON 2020*, Hyderabad, 2020. <https://doi.org/10.1109/HYDCON48903.2020.9242677>
- [16] S. S. S, R. B. S. and R. A. R, "Virtual mouse using hand gestures by skin recognition," *Journal of Population Therapeutics and Clinical Pharmacology*, vol. XXX, no. 7, pp. 251-258, 2023. <https://doi.org/10.47750/jptcp.2023.30.07.030>
- [17] M. Mishra, A. Gupta, V. Mehta and R. Ranjan, "Virtual Mouse Input Control using Hand Gestures," in *7th International Conference on Trends in Electronics and Informatics, ICOEI 2023 - Proceedings*, Tirunelveli, 2023. <https://doi.org/10.1109/ICOEI56765.2023.10125668>
- [18] S. Ranjan, S. Anand, V. Gontia, A. Chandra, V. Sharan and S. SG, "Virtual Mouse Control using Hand Gesture," *International Journal of Engineering Research in Computer Science and Engineering*, vol. IX, no. 11, pp. 10-13, 2022. <https://doi.org/10.36647/ijerces/09.11.art006>
- [19] D. S. Tran, N. H. Ho, H. J. Yang, S. H. Kim and G. S. Lee, "Real-time virtual mouse system using RGB-D images and fingertip detection," *Multimedia Tools and Applications*, vol. LXXX, no. 7, p. 10473-10490, 2021. <https://doi.org/10.1007/s11042-020-10156-5>
- [20] J. D. Asmoro, A. T. Wibowo and M. Ridwan, "Virtual Mouse With Hand Gesture Recognition Based on Hand Landmark Model for Pointing Device," *Jurnal Teknologi dan Sistem Informasi*, vol. IX, no. 2, pp. 261-268, 2023. <https://doi.org/10.33330/jurteksi.v9i2.2073>
- [21] M. M. Sahiwala, S. R. Singh, D. Pawar, D. D. Jakasaniya and K. M. Patel, "Virtual Mouse using Coordinate Mapping," in *2nd International Conference on Sustainable Computing and Data Communication Systems, ICSCDS 2023 - Proceedings*, Erode, 2023. <https://doi.org/10.1109/ICSCDS56580.2023.10104598>
- [22] T. K. Selvi, S. Sasirekha, M. Manikandan, M. O. Solomon and M. Vignesh, "Virtual Mouse using OpenCV and VNC," *Journal of Information Technology and Digital World*, vol. V, no. 2, pp. 169-179, 2023. <https://doi.org/10.36548/jitdw.2023.2.007>
- [23] S. Wattamwar, D. Ramgude and S. More, "Hand Gesture Based Virtual Mouse Using AI," *Interantional Journal of Scientific Research in Engineering and Management*, vol. VIII, no. 1, pp. 1-3, 2024. <https://doi.org/10.55041/ijserm28234>
- [24] C. Saritha, Y. Phanitha and A. R. Krishna, "Virtual Mouse Operations Using Webcam," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. X, no. 3, pp. 650-653, 2023. <https://doi.org/10.32628/ijrsrset23103152>
- [25] S. Srivastava, R. Tiwari, A. Garg, Rishu and R. Tiwari, "Design and Development of Gesture Recognition Based Virtual Mouse System," in *2023 International Conference on Computational Intelligence, Communication Technology and Networking, CICTN 2023*, Ghaziabad, 2023. <https://doi.org/10.1109/CICTN57981.2023.10140390>
- [26] Shukla, D. D. Katiyar and M. G. Goel, "Gesture Recognition-based AI Virtual Mouse," *International Journal for Research in Applied Science and Engineering Technology*, vol. X, no. 3, pp. 1583-1588, 2022. <https://doi.org/10.22214/ijraset.2022.40937>
- [27] R. Shukla, "AI Virtual Mouse Using Hand Gesture Recognition," *International Journal of Research Publication and Reviews*, vol. III, no. 10, pp. 168-173, 2022. <https://doi.org/10.55248/gengpi.2022.3.10.13>