ÇANKAYA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CENG 407

Innovative System Design and Development I

Project Report

IoT Integrated Accessible Smart Home System

Hasan Emre Usta - 202111031

Kayra Dalçık – 202111023

Ömer Altıntaş – 202111209

Gökay Çetinakdoğan - 202111050

Advisor: Faris Serdar Taşel

1. Home

Project Description

This project aims to design and implement an innovative home automation control system that leverages Internet of Things (IoT) technologies and computer vision for hands-free and accessible smart home interaction. This system is designed to provide independence and improved quality of life for individuals with motor disabilities, while also offering an innovative human-computer interaction paradigm for general users.

2. Introduction and Objectives

2.1 Project Overview

This project aims to design and implement an innovative home automation control system that leverages Internet of Things (IoT) technologies and computer vision for hands-free and accessible smart home interaction. This system is designed to provide independence and improved quality of life for individuals with motor disabilities, while also offering an innovative human-computer interaction paradigm for general users. The system enables users to control smart home devices through eye-tracking and gaze-based commands, eliminating the need for traditional physical interfaces such as switches or voice assistants.

2.2 Objectives

The main objectives of this project are:

- Develop a vision-based user interface capable of detecting and interpreting users' eye movements, gaze direction, and blinking patterns in real time.
- Integrate this interface with an IoT control hub that manages smart home devices via wireless communication.
- Create WiFi-enabled prototype devices (e.g., lights, fans, temperature controllers) that respond to gaze-based commands.
- Design an accessible, low-cost, and reliable smart home control solution suitable for individuals with limited mobility and elderly users.

2.3 Scope

The scope of this project includes:

- Implementation of a vision processing module on a Raspberry Pi 5 equipped with a camera module for eye-tracking using MediaPipe or similar Python libraries.
- Development of an IoT control hub using an ESP32 microcontroller that communicates with the Raspberry Pi over WiFi.
- Integration of prototype smart devices that can be wirelessly controlled through the ESP32.
- Evaluation of system performance in terms of response time, accuracy, and user accessibility.

2.4 Target Hardware

- Raspberry Pi 5 (with ip camera for image capture and processing)
- ESP32 microcontroller (for IoT control and communication)
- WiFi-enabled prototype devices (e.g., smart light, fan, temperature controller)
- Power supply units and safety enclosures for all modules

2.5 Constraints

The system design must consider the following constraints:

- Power: Each module should operate efficiently under limited power conditions, suitable for continuous home use.
- Cost: Components should be cost-effective to allow for affordable scalability and real-world deployment.
- Safety: Electrical connections and enclosures must comply with basic electrical safety standards to prevent hazards.
- Camera Privacy: Indicator when camera is active; no data recording or transmission
- Testing Protocol: Comprehensive safety testing before live demonstrations

3. Literature & Technology Review

This section presents a comprehensive literature review on smart home automation platforms, eye-tracking–based human–computer interaction, IoT communication protocols, computer vision algorithms, hardware platforms, and facial recognition technologies. The aim is to establish the scientific and technological foundation of the accessible smart home control system to be developed, to identify components that may be expanded in the future, and to highlight the innovative contributions of the project compared to existing systems.

3.1 Modern Smart Home Systems and Accessibility Challenges

Contemporary smart home technologies, human–computer interaction (HCI) research, and studies in the field of accessibility collectively form the multidisciplinary literature that underpins the development of an eye-tracking–based smart home control system. Although this body of work has been extensively explored in both industry and academia, studies that integrate eye tracking with IoT-based device control remain quite limited. Therefore, a thorough review of previous research is essential for revealing the novel aspects of the system. This section evaluates the existing literature across three main axes: smart home platforms, eye-tracking–based HCI research, and assistive technologies.

3.1.1 Modern Smart Home Systems and Their Limitations

Smart home systems have significantly advanced over the past decade, forming a large ecosystem through platforms such as Google Home, Amazon Alexa, Apple HomeKit, and Samsung SmartThings. These systems coordinate lighting, security, HVAC, energy management, and multimedia devices in an automated manner. Through protocols such

as Matter, Zigbee, WiFi, Thread, and Z-Wave, devices from different manufacturers can be controlled from a single centralized interface.
However, a commonly emphasized issue in the literature is that despite their technical capabilities, these systems fall short in terms of accessibility. Modern platforms typically rely on two primary interaction paradigms:

- Mobile application–based control

- Voice command–based control

Mobile applications depend on tapping icons or performing touch gestures on a screen, which requires fine motor skills—posing a significant challenge for individuals with neuromotor impairments. Voice-based control, on the other hand, is unreliable for users with speech impairments, neurological disorders, or those living in noisy environments. Consequently, the literature indicates that while modern smart home systems are technologically robust, they do not adequately accommodate all user groups. One of the most essential forms of control for individuals with disabilities-contactless interaction is almost entirely absent from these platforms. Notably, no commercial smart home system today provides a built-in mechanism for controlling devices through eye movements. This gap directly reinforces the innovative value of the proposed project.

3.2 Eye-Tracking in Human–Computer Interaction

Eye tracking holds a distinct position within the field of human–computer interaction (HCI) because eye movements are among the few behavioral indicators that directly and rapidly reflect user intent. In the literature, eye tracking is regarded not only as a sensory input modality but also as a fundamental component of the "natural interaction" paradigm. For users with limited motor abilities, eye movements can serve as a powerful and accessible alternative to traditional input methods such as touch, speech, keyboard use, or gesture-based control.

3.2.1 The Role of Eye Movements in Interaction

HCI literature evaluates eye movements through three primary functions: attention, intention, and direct control. The eyes naturally reveal where a person's attention is directed, providing a strong cue for systems attempting to infer user intent. In attention-based models, the system interprets what the user intends to do by analyzing the focus point on the screen. For example, studies in educational technologies often assess whether students are engaged with learning materials based on their gaze behavior.
In intention-based models, the eyes function as a pointing mechanism. Users can make selections or issue commands simply by looking at a target. The most well-known approach within this paradigm is the "gaze-as-input" model, in which eye movements serve as an input channel comparable to a mouse cursor. The literature demonstrates that this model is particularly effective for enabling computer access for individuals with motor impairments.
The third function, direct control, treats eye movements as explicit control signals. For instance, a leftward gaze can trigger a predefined action. This form of control is crucial for individuals who cannot physically access buttons or traditional interfaces. HCI studies emphasize that because eye movements impose minimal cognitive load, gaze-based control constitutes one of the most natural interaction methods available.

3.2.2 Blink-Based Interaction Models

Blinking is considered one of the earliest and most natural user signals in HCI. In blink-based systems, users trigger actions through single or double blinks. Blink-driven communication systems have historically been used for individuals with ALS, spinal cord injuries, or severe communication impairments. The foundational mechanism is relatively straightforward: the system monitors open/closed eye states and maps them to commands.

However, the literature highlights several limitations of blink-only systems. Prolonged use can cause eye fatigue or dryness, and involuntary blinks may lead to false positives, posing safety concerns. Moreover, blink frequency varies significantly across users. For these reasons, modern HCI research recommends using blinking as a supportive signal rather than the primary interaction method. In the proposed project, blinks may similarly be employed as auxiliary commands to reinforce gaze-direction-based control.

3.2.3 Gaze-Based Selection and Dwell-Time Interaction

The most mature gaze-based interaction model in the literature is gaze-based selection, in which gaze direction is used to control a cursor or select items in an interface. This approach is widely applied in screen-based interaction systems. A user looks at an interface element (e.g., a button), and if the gaze remains fixed on the element for a specified duration, a selection is executed. This mechanism is referred to as dwell-time selection.

HCI research demonstrates that dwell-time values are critical for both system accuracy and user satisfaction.

- If dwell time is too short, unintentional glances may be interpreted as selections.

- If it is too long, interaction becomes slow and cumbersome.

Thus, optimizing dwell time is one of the essential design parameters in gaze-controlled systems.

Gaze-based selection is highly suitable for smart home automation settings as well. Users can simply look at a particular object—such as a lamp icon or the image of a physical device—to communicate their intent. This makes gaze-based interaction particularly valuable in situations where physical contact is impossible or undesirable.

3.2.4 Head Pose, Eye Fatigue, and Real-World Constraints

HCI literature emphasizes that eye-tracking performance must be evaluated not only under ideal laboratory conditions but also in real-world environments. Factors such as changes in head orientation, variations in distance from the camera, the use of eyeglasses, lighting conditions, and user fatigue significantly influence system accuracy.

Eye fatigue is particularly problematic during extended use. For this reason, modern interaction design research suggests employing short interaction cycles and providing users with periodic rest opportunities in gaze-based systems. Eye fatigue is especially prominent in applications requiring continuous cursor control. In contrast, smart home environments, which involve momentary interactions, are naturally well-suited for gaze-based control.

Adaptation to head movement has long been a technical challenge; however, modern systems employing mesh-based facial tracking have substantially mitigated this issue. Even so, HCI studies continue to stress the importance of ergonomically positioning the camera relative to the user for optimal performance.

3.2.5 Eye Tracking for Smart Home Interaction: Research Gap

Although the eye-tracking literature is extensive, studies integrating eye tracking directly with IoT-based smart home control remain scarce. Most existing research focuses on:

- cursor control in screen-based interfaces,
- augmentative and alternative communication (AAC) devices,
- wheelchair navigation systems,
- educational and vision-health applications.

In the domain of smart home automation, gaze-based control has not yet become an established paradigm. Therefore, the proposed project:

- targets a combination that is largely missing in the literature (HCI + IoT + Eye Tracking),
- offers a natural interaction modality for individuals with motor impairments,
- introduces a novel interaction layer to smart home ecosystems.

3.3 Assistive Technologies for People with Disabilities

Assistive technologies developed for individuals who are unable to use traditional human–computer interaction methods due to motor impairments, speech disorders, neurological conditions, or physical disabilities have long been a critical research area in the literature. These technologies aim to help users achieve independence in daily life, communicate effectively, and interact with their environment. Accordingly, the assistive technology literature encompasses a broad conceptual foundation spanning both technological innovation and user experience.
Systems designed for individuals with disabilities often integrate sensing technologies, sensor-based solutions, biological signal analysis, and computer vision techniques. However, a common conclusion in the literature is that no single assistive technology is universally ideal for all users. For this reason, eye tracking has gained increasing attention due to its advantages such as low cost, natural interaction, contactless operation, and high accessibility.
This section examines the assistive technology literature through three primary categories: systems based on biological signals, motion- and gesture-based systems, and computer-vision–based contactless interaction systems.

3.3.1 EEG, EMG, and Biological Signal-Based Assistive Technologies

Technologies based on EEG (Electroencephalography) and EMG (Electromyography) are among the most advanced tools developed for enabling communication and device control in individuals who have experienced significant loss of motor function. EEG-based brain–

computer interfaces (BCIs) interpret neural activity and translate it into commands. The literature highlights that such systems play a critical role as communication tools for individuals with severe motor neuron diseases such as ALS.
Despite offering high accuracy, EEG-based solutions present several practical limitations. Electrode placement requires expertise, the systems are costly, neural signals are highly sensitive to noise, and users must wear electrodes on the scalp for extended periods, making daily use inconvenient.
EMG-based systems derive commands from muscle activation; however, they are ineffective for users who have entirely lost muscle function. Furthermore, involuntary muscle contractions may generate false-positive commands. Consequently, the literature generally characterizes biological-signal–based technologies as suitable primarily for laboratory environments rather than practical everyday use for a broad user population.

3.3.2 Motion, Gesture, and Wearable-Based Assistive Systems

Motion sensors, accelerometers, wearable embedded sensors, and glove-based devices represent another major group of assistive technologies frequently explored for enabling interaction in individuals with disabilities. These systems rely on gesture recognition or the extraction of commands from bodily movements. For instance, glove-based systems can detect finger flexion to enable robotic arm control or basic computer interactions.
However, the fundamental limitation of such systems is their dependence on motor movement. For individuals with spinal cord injuries or those experiencing significant loss of muscle tone, these devices become difficult or impossible to use. Additionally, wearable technologies may cause discomfort during prolonged use, require frequent calibration, and pose challenges in adapting to daily routines—issues widely reported in the literature.
Therefore, although motion-gesture systems may be effective for individuals with mild impairments, they fail to offer sufficient accessibility for users with more severe motor limitations. This highlights the growing need for contactless, low-effort interaction methods.

3.3.3 Voice-Controlled Assistive Technologies and Their Limitations

Voice recognition systems—such as Google Assistant, Siri, and Alexa—are among the most widely adopted assistive tools in both commercial and academic contexts. Users can control devices simply by issuing voice commands. However, voice control presents significant limitations in accessibility literature.
Individuals with speech disorders often cannot use these systems reliably. Performance decreases substantially in noisy environments, and some neurological conditions disrupt speech rhythm or articulation, leading to recognition errors. Additionally, privacy concerns prevent many users from adopting voice-activated technologies. Therefore, voice-based control is not considered a universally accessible solution in assistive technology research.

3.3.4 Computer Vision-Based Assistive Systems

4 In recent years, computer-vision–based assistive technologies have gained rapid prominence in the literature. Camera-based systems offer substantial advantages because they can infer user intent without requiring physical contact. Facial expressions, head orientation, eye openness, and pupil movements can all serve as indicators of user intention.

5 Expression-based systems interpret gestures such as smiling or raising eyebrows as commands; however, these require strong facial muscle control and therefore are not suitable for all users. Head-orientation–based control may be effective for simple robotic or cursor-control tasks, but prolonged use may lead to neck fatigue.
6 Within this context, eye tracking stands out as the most effective and user-friendly

method among computer-vision–based systems. Eye movements are more isolated from involuntary motor activity, faster to execute, and more natural for individuals with limited motor abilities, making them especially suitable for accessible interaction.

3.3.5 Why Eye Tracking Stands Out Among Assistive Technologies

The literature identifies four key characteristics that make eye tracking unique within assistive technology:

- **Natural interaction:** Eye movements require no additional cognitive or physical effort.
- **Contactless operation:** No muscle force or physical interaction is needed; micro-movements suffice.
- **High accessibility:** Effective even for individuals with severe motor impairments.
- **Low cost:** Camera + software solutions are significantly cheaper than EEG/EMG-based systems.

Consequently, eye-tracking technologies are frequently described as “the most practical and sustainable control method” for individuals with physical disabilities.

3.3.6 Research Gap: Assistive Eye-Tracking for Smart Homes

Despite the breadth of assistive technology research, studies integrating eye tracking with IoT-based smart home control remain extremely limited. Most existing research focuses on:

- screen-based cursor interaction,
- augmentative and alternative communication devices,
- wheelchair navigation systems,
- rehabilitation applications.

Within the context of smart home automation, gaze-based control remains a relatively unexplored domain. Therefore, the proposed project:

- addresses an underdeveloped intersection in the literature (HCI + IoT + Eye Tracking),
- offers a highly innovative and accessibility-focused contribution,
- introduces a new gaze-driven control paradigm capable of managing multiple IoT devices.

3.4 Computer Vision Algorithms and Software Frameworks

In the smart home automation system to be developed, the core component of user interaction is the processing of camera-captured images through computer vision techniques and the extraction of meaningful eye-movement signals from these images. Therefore, understanding the historical progression of computer vision research, examining the operational principles of the algorithms employed, and evaluating the behavior of different methods on embedded hardware platforms are critically important for the project. The following subsections provide a broad overview ranging from classical approaches to modern deep-learning–based solutions, along with a detailed discussion of facial recognition technologies that may be integrated into the system in future stages.

3.4.1 Classical Computer Vision Approaches

Among the earliest methods used for eye detection in the field of computer vision are traditional image-processing–based models. One of the most well-known examples is the Haar Cascade classifier family, which is based on the Viola–Jones framework. Haar features employ simple rectangular filters that capture basic intensity variations in the image, and with the help of the AdaBoost algorithm, a cascade of classifiers enables fast detection of eyes or faces. This method was widely used in early devices with limited computational power.

However, Haar Cascade methods have several significant limitations. Because the features used by the algorithm do not adequately represent the complex structure of the human eye, detection accuracy degrades substantially under variable lighting conditions, partially closed eyelids, or changes in face orientation. Furthermore, these models can only identify the general location of the eyes—they cannot estimate iris position or gaze direction. For this reason, classical techniques are insufficient for today’s real-time, interaction-intensive applications.

3.4.2 Landmark-Based Face and Eye Analysis

Following classical approaches, a major milestone in the literature was the development of landmark-based models capable of identifying specific anatomical points on the face. The 68-point facial landmark model implemented in Dlib is one of the most prominent examples. This model extracts critical landmarks corresponding to the eyes, nose, mouth, and jawline, enabling various geometric computations.

For instance, the Eye Aspect Ratio (EAR) metric is widely used to detect whether the eyes are open or closed, making it effective for blink detection or attention analysis. However, because EAR does not provide direct information about the pupil position, gaze direction can only be inferred indirectly from landmark geometry—and such estimations often suffer from high error rates.

Additionally, the performance of landmark-based models on embedded systems is a major concern. These models typically run at around 8–10 FPS on devices such as Raspberry Pi, which can negatively impact user experience in real-time control systems. Thus, while landmark-based approaches hold academic value, they fall short of meeting the responsiveness requirements of the proposed system.

3.4.3 MediaPipe Face Mesh and Iris Tracking

Modern and high-accuracy solutions in eye tracking became possible with the development of deep-learning–based face and iris models. One of the most powerful tools in this domain is MediaPipe Face Mesh, which, despite its computational efficiency, is capable of generating 468 three-dimensional facial landmarks. One of its subcomponents detects the iris center using four high-precision points, enabling direct estimation of gaze direction.

A key factor behind MediaPipe’s success is the BlazeFace detector used for face detection, which operates very efficiently on CPUs. The subsequent mesh regression model generates a stable 3D facial structure that remains robust even under head movements. MediaPipe can achieve 12–25 FPS on Raspberry Pi, making it one of the most suitable solutions for real-time gaze tracking on embedded platforms.

Thanks to its architecture, MediaPipe can simultaneously track blinks, eye openness, iris movement, gaze direction, and head pose. For this reason, it stands out as a strong candidate for the core user-interaction module in this project.

3.4.4 Deep Learning–Based Gaze Estimation

The most accurate gaze-tracking methods in the literature are deep-learning–based models that directly estimate gaze direction. Systems such as Gaze360 and iTracker process both eye and face images to predict three-dimensional gaze vectors. These models rely on large convolutional neural networks (CNNs) and require substantial computational resources. Thus, while they can be executed on GPU-enabled platforms such as Jetson Nano, they are unsuitable for real-time operation on CPU-only embedded systems like Raspberry Pi.

Although these models serve as important accuracy benchmarks in gaze-tracking research, limitations related to power consumption, processing load, and latency make them impractical for the current stage of the project. However, with future enhancements to system architecture—such as cloud-based processing or the use of more powerful edge devices—they remain potential candidates for advanced system upgrades.

3.4.5 Real-World Performance Considerations

The performance of eye-tracking algorithms is determined not only by theoretical accuracy but also by adaptability to real-world conditions. Common real-world challenges highlighted in the literature include variable lighting conditions, eyeglass reflections, changes in the distance between the user and the camera, camera-angle constraints, and user fatigue.

Modern models such as MediaPipe incorporate mechanisms such as mesh stabilization, temporal smoothing, and illumination robustness to mitigate these challenges. These capabilities enable MediaPipe to deliver significantly more stable and resilient performance compared to classical approaches.

3.5 Face Recognition Systems

Although facial recognition technologies are not currently utilized within the scope of the project, they may be incorporated into the system in future stages. Therefore, it is important to review the fundamental approaches and algorithms discussed in the literature.

Haar Cascade Face Detection

This is one of the most classical methods used for face detection. However, it is not stable under modern and variable lighting conditions.

Dlib Face Recognition

This method enables user authentication by generating 128-dimensional facial embeddings. It can run on Raspberry Pi, but its performance is relatively slow.

Deep Learning–Based Face Recognition (FaceNet, ArcFace)

These models represent the state-of-the-art in modern facial recognition. Due to their high accuracy, they have become the standard in contemporary face recognition systems. However, they require GPU acceleration and are therefore difficult to deploy on embedded systems for edge computation.

Facial recognition can be used in the future to support features such as user authentication, personalized profile settings, child safety mechanisms, and automatic system login. For this reason, it presents a significant opportunity for expansion in the future work section of the project.

3.6 IoT Communication Protocols and Smart Home Messaging

For smart home systems to operate reliably and seamlessly, not only the performance of hardware components but also the communication protocols connecting these components play a crucial role. In IoT architectures, communication significantly influences system responsiveness, scalability, fault tolerance, and reliability. As a result, the literature includes an extensive body of research analyzing the strengths and weaknesses of various communication protocols in detail. Particularly in real-time interaction scenarios—such as gaze-based control systems—the selection of an appropriate communication protocol is a critical factor that directly shapes the user experience.

Most IoT protocols are optimized for specific requirements; some aim for low power consumption, while others are designed for high bandwidth or low latency. Therefore, the literature does not designate any single protocol as an "ideal solution"—each protocol offers distinct advantages depending on context. The following subsections examine modern IoT communication methods, their role within smart home environments, and common use cases in the literature through a technical and neutral perspective.

3.6.1 MQTT and the Publish–Subscribe Model in IoT Architectures

MQTT is widely recognized in the literature as one of the most commonly used lightweight communication protocols for IoT applications. It is built on a publish–subscribe architecture, wherein devices communicate not directly with each other but through a broker. This model provides scalable and low-latency communication, especially in scenarios where many devices continuously exchange data over the same network.

Academic studies frequently highlight the following characteristics of MQTT:

- extremely small packet size,
- stable performance under unreliable network conditions,
- a natural fit with event-driven systems,
- low computational overhead on embedded devices.

Due to these advantages, MQTT has been widely adopted in sensor networks, home automation systems, environmental monitoring, and real-time control mechanisms. However, the literature also notes certain limitations: security configurations must be handled carefully, broker dependency may create a single point of failure in rare cases, and QoS settings require proper management in large distributed environments. Thus, while MQTT is powerful and efficient for lightweight applications, it demands careful configuration in high-security industrial settings.

3.6.2 HTTP REST: Ubiquity and Accessibility with Latency Limitations

HTTP REST remains an important option in IoT projects due to its broad compatibility. REST-based systems are platform-independent, easy to implement, and supported across virtually all device ecosystems. Therefore, REST is often considered a strong choice for IoT scenarios in which data transfer is infrequent, periodic, or of low criticality.

However, the request–response nature of HTTP REST introduces substantial latency. Establishing a new connection for each request, the overhead associated with large HTTP headers, and the lack of built-in publish–subscribe functionality make REST less suitable for real-time interaction environments.

As such, academic studies indicate that REST is typically preferred not in high-frequency or instantaneous control mechanisms but rather in applications requiring larger data transfers or infrequent communication.

3.6.3 Zigbee, Z-Wave, and Thread in Low-Power Smart Home Networks

Zigbee, Z-Wave, and Thread are widely used in commercial smart home systems, primarily due to their energy efficiency and robust mesh networking capabilities. Systems such as Philips Hue, IKEA TRÅDFRI, and SmartThings have relied on these protocols for years. Their low energy consumption enables battery-powered sensors to operate for extended periods.

The mesh network architecture allows devices to function as both endpoints and routers, enabling stable communication across large spaces. This makes Zigbee and Thread particularly advantageous in multi-story homes or wide residential areas.

Nevertheless, the literature identifies several limitations. Zigbee and Z-Wave typically require specialized gateways or dongles, which may increase development costs. Their data transmission speeds are relatively low, making them unsuitable for high-frequency or high-bandwidth applications. As a result, while these protocols are ideal for low-power sensor networks, they are less effective in systems requiring high throughput or low latency.

3.6.4 Matter and the Evolution of Smart Home Interoperability

Matter is a next-generation standard developed by Google, Apple, Amazon, and the Connectivity Standards Alliance (CSA), receiving growing attention in the literature. Its primary goals include enhancing cross-device compatibility, ensuring manufacturer independence, and reducing fragmentation within the smart home ecosystem.

When combined with Thread, Matter supports low-power operation, secure communication, and stable mesh networking. However, academic sources note that Matter is still in its maturation phase and may require advanced technical expertise during development. Limited hardware support can also increase prototyping costs.

Thus, while Matter is considered a key technology for the future of smart home environments, it remains in an early adoption stage within research and educational contexts.

3.6.5 WiFi-Based Communication in Embedded IoT Projects

WiFi is one of the most widely used wireless communication methods in academic and prototyping environments. This is primarily due to its high bandwidth, lack of additional hardware requirements, natural compatibility with embedded platforms, and flexibility during development. Devices such as Raspberry Pi and ESP32, which include built-in WiFi support, further reinforce its widespread use.

WiFi provides stable performance on local networks; however, the literature emphasizes that latency fluctuations may occur under heavy network traffic. Additionally, compared to low-power protocols like Zigbee and Thread, WiFi consumes significantly more energy.

Thus, while WiFi is advantageous for rapid prototyping and high-interaction applications, it is less suitable for battery-powered IoT systems with strict power constraints.

3.6.6 MQTT + WiFi in Literature: Use Cases, Performance, and Limitations

Literature surveys indicate that the combination of MQTT and WiFi is widely preferred in projects requiring low latency, fast data transmission, and embedded-platform compatibility. The ease of integrating this combination with low-cost embedded systems such as Raspberry Pi and ESP32 has made it a popular choice in academic research.

Studies highlight several benefits of this architecture:

- millisecond-level latency,
- simple development ecosystem,
- small packet sizes,
- high stability in local networks,
- modular publish–subscribe structure.

However, the literature also acknowledges that other protocols may provide advantages depending on the application. For instance, Zigbee and Thread offer excellent low-power performance for sensor networks, while Matter is beneficial for long-term device interoperability in large systems. REST-based communication is more appropriate for large data transfers and scenarios where real-time interaction is not required.

Thus, while MQTT + WiFi is recognized as a widely used and high-performing solution, IoT research demonstrates that no single protocol is universally optimal. Protocol selection must be guided by application requirements, energy constraints, and the interaction model.

3.7 Hardware Platforms and Processing Considerations

Selecting a hardware platform for smart home systems is not merely an engineering-oriented decision; it requires a multidimensional evaluation that considers the system's operating model, computational capacity, power consumption, real-time performance requirements, and the accessibility needs of the target user group. A gaze-tracking–based control system necessitates the seamless integration of components such as a camera sensor, image-processing unit, wireless communication modules, and IoT device controllers. Therefore, the literature identifies the comparison of hardware platforms—based on processing capability, ecosystem support, and extensibility—as a critical step in system design.
The following subsections provide a comprehensive examination of commonly used components in this field, including embedded computers, microcontrollers, camera sensors, GPU accelerators, and energy–thermal management modules.

3.7.1 Raspberry Pi as a Vision-Centric Embedded Computing Platform

Raspberry Pi is one of the most widely used platforms in computer vision and IoT projects due to its low cost and extensive community support. Its multi-core ARM Cortex-A72 processor, compatibility with the Python ecosystem, and native support for libraries such as OpenCV and MediaPipe make it an ideal prototyping environment for image-processing tasks.
Academic studies highlight several advantages of Raspberry Pi:

- Ability to perform image processing and real-time tasks concurrently
- Platform-independent Python support enabling rapid development
- Tight integration with camera modules
- Low power consumption (5–7 W)
- Broad compatibility with various sensors and peripherals

Raspberry Pi includes a limited GPU capable of running lightweight CPU-optimized deep learning models. While this restricts the execution of large CNN architectures, it provides adequate performance for optimized models such as MediaPipe Face Mesh. The literature reports stable performance between 12–30 FPS for image processing tasks, which is sufficient for gaze-direction–based interaction.
However, constraints such as limited memory, thermal throttling under heavy computational loads, and inadequate processing capability for large-scale deep learning models are frequently discussed limitations. Thus, Raspberry Pi is generally preferred for lightweight to moderate vision workloads.

3.7.2 ESP32 as a Low-Cost Wireless IoT Control Unit

ESP32 is one of the most widely adopted microcontrollers in the IoT ecosystem and is frequently employed in academic literature for wireless sensor networks, smart home devices, and event-driven control systems. With its low power consumption, built-in WiFi/Bluetooth support, and fast GPIO access, it is ideal for real-time device control.
Key advantages highlighted in research include:

- Stable communication over WiFi and strong compatibility with MQTT

- Millisecond-level response time
- Low energy consumption (~0.2–1 W)
- Extensive open-source development support
- Dual-core processor enabling multitasking

Although ESP32 lacks the processing power required for image processing or heavy data analysis, it performs exceptionally well in tasks involving device control, sensor acquisition, and network communication. As a result, many system architectures in the literature adopt hybrid designs such as Pi + ESP32 or Jetson + ESP32.

3.7.3 Camera Systems for Eye Tracking Applications

Camera selection is a critical hardware design decision for eye-tracking systems. Academic research indicates that camera parameters including resolution, FPS, lens characteristics, and infrared (IR) support directly affect algorithmic performance.

Common camera types used in gaze-tracking applications include:

RGB Cameras:

Raspberry Pi Camera Modules, USB webcams, and low-latency CSI cameras are frequently used. Under adequate illumination, RGB cameras provide reliable performance for iris tracking.

Infrared (IR) Cameras:

IR cameras deliver more stable results in low-light environments and offer clearer visibility of the pupil under IR illumination, making them suitable for high-accuracy eye-tracking systems. However, they require more complex hardware and are generally more expensive.

High-FPS Cameras:

Cameras operating at 120 FPS or higher allow the tracking of rapid eye movements; however, they significantly increase the computational load on embedded devices. For this reason, high-FPS cameras are mainly used in academic research, whereas mid-range FPS (e.g., 30+) is sufficient for real-time prototyping.

The literature indicates that RGB cameras exceeding 30 FPS provide adequate performance for low-latency gaze-based interaction, making Raspberry Pi's camera ecosystem a commonly preferred option.

3.7.4 Alternative Embedded AI Platforms: Jetson Nano, Coral TPU, OpenMV

Several alternative embedded AI platforms have been studied in the literature for running more computationally demanding models:

Jetson Nano:

NVIDIA's GPU-accelerated platform can run CNN-based deep learning models with high performance. It supports advanced gaze-estimation models such as Gaze360 and iTracker. However, its relatively high power consumption (10–20 W), cost, and need for active cooling make it less ideal for small-scale prototypes.

Google Coral Edge TPU:

The Edge TPU accelerates TensorFlow Lite models via dedicated hardware. Although highly efficient, its compatibility is limited to specific model architectures, and some MediaPipe components are not supported.

OpenMV Cam:

With low power consumption and MicroPython support, OpenMV is suitable for basic image-processing tasks; however, it lacks the computational capacity required for robust eye-tracking applications.
Despite their strengths, these platforms are often outweighed by the superior cost–performance balance of Raspberry Pi for gaze tracking + microcontroller-based IoT control. Thus, most academic implementations adopt a Raspberry Pi front-end combined with a microcontroller back-end.

3.7.5 Power Consumption, Thermal Performance, and Practical Constraints

Power and thermal considerations are essential in embedded systems, especially for continuous-vision applications such as gaze tracking, where image processing runs constantly.
Key constraints highlighted in the literature include:

- Raspberry Pi can exceed 80°C under heavy image-processing workloads
- Thermal throttling may reduce FPS and degrade performance
- USB cameras add additional power demands
- ESP32 consumes minimal power but requires more energy during WiFi transmission

Therefore, cooling solutions such as fans, heatsinks, and resolution/FPS optimization strategies are widely employed in academic projects.

3.7.6 Hardware Selection in Literature: General Patterns and Use Cases (Neutral)

A review of the literature reveals consistent patterns in hardware selection for eye tracking, IoT control, and accessibility-focused systems:

- Embedded computers (Raspberry Pi, Jetson Nano) for image processing
- Microcontrollers (ESP32, STM32) for device-level control
- Low-latency RGB or IR cameras as sensing components
- WiFi-based communication for rapid prototyping
- GPU accelerators for large deep learning models
- Zigbee/Thread devices for low-power, energy-critical applications

This diversity demonstrates that different hardware platforms offer advantages in different contexts. The literature acknowledges that no single platform is universally optimal; rather, system architecture, performance requirements, and cost considerations collectively shape hardware selection.

3.8 Architectural Alternatives

In smart home automation systems, the architectural structure plays a decisive role in the system's performance, reliability, latency tolerance, and scalability. For a model that requires gaze-tracking–based user interaction, architectural selection becomes even more critical because commands generated from eye movements must be processed, interpreted, and transmitted to IoT devices within milliseconds. Therefore, the literature identifies various architectural approaches depending on where embedded image processing, IoT communication layers, and control mechanisms are executed. This section examines three primary architectures frequently discussed in the literature for smart home systems- edge processing, cloud-based processing, and hybrid/integrated architectures evaluated in terms of their technical foundations, advantages, limitations, and use cases.

3.8.1 Edge Processing Architectures

In edge processing architectures, all image processing, decision-making, user interaction analysis, and control logic are executed directly on local devices. This approach is particularly emphasized in the literature for embedded vision applications and systems requiring real-time performance.

Low Latency and Real-Time Performance

One of the most important requirements of gaze-based systems is converting gaze-direction changes into IoT device commands within milliseconds. The literature demonstrates that edge-based solutions provide significantly lower latency compared to cloud-based systems because the image is processed locally, eliminating or minimizing network delay. For this reason, edge processing is the most commonly preferred architecture in real-time HCI applications.

Privacy and Data Locality

Keeping image data on the local device offers a major advantage in terms of user privacy. For applications involving sensitive biometric data such as eyes, face, and facial expressions, edge processing is widely recommended in academic and ethical research.

Limitations and Computational Constraints

Despite its advantages, edge processing is limited by processing power, memory capacity, and thermal constraints. GPU-enabled devices such as Jetson Nano can overcome these limitations, but at a higher cost. Systems operating on Raspberry Pi are limited to lightweight models. Therefore, edge processing is most suitable for optimized and lightweight vision models.

3.8.2 Cloud-Based Architectures

Cloud processing refers to an architecture in which image-processing tasks are offloaded to powerful remote servers. The local device streams camera data and receives processed results from the cloud. This approach is valuable for running large deep-learning models that cannot be executed on embedded hardware.

GPU-Accelerated Model Deployment

Cloud architectures allow running heavy models such as iTracker or Gaze360 with high accuracy on GPU-equipped servers. These models are too computationally demanding for embedded devices but operate effectively in the cloud.

Scalability and Centralized Control

Cloud-based systems enable centralized control of many devices. This architecture is widely used in large smart-building deployments and intensive sensor networks.

Limitations: Latency, Privacy, and Connectivity

The main disadvantages of cloud architectures include:

- network latency,

- dependence on internet connectivity,

- privacy risks associated with transmitting user image data.

The literature reports internet latency between **40–200 ms**, which is unacceptable for real-time gaze-controlled systems. Therefore, cloud-based architectures are generally unsuitable for eye-tracking applications requiring instantaneous feedback.

3.8.3 Hybrid and Distributed Architectures

Hybrid architectures distribute image processing and decision-making tasks between edge devices and cloud servers. This design is popular in research aiming for both low latency and high accuracy.

On-Device Preprocessing + Cloud-Level Deep Analysis

In this approach:

- basic tasks (face detection, landmark extraction, low-level processing) run on the device,

- high-accuracy operations (deep gaze estimation) are offloaded to the cloud.

This reduces cloud load and minimizes data transmission.

Adaptive Offloading Strategies

The literature describes adaptive systems that dynamically offload tasks based on CPU/GPU load, balancing energy consumption and performance.

Increased Complexity and Reliability Concerns

Hybrid architectures increase system complexity because two separate computing environments must work reliably together.
If the edge component fails, cloud processing is disrupted, and vice versa.
Additionally, hybrid systems usually depend on continuous internet access.

3.8.4 Privacy, Latency, and Energy Trade-Offs in Architecture Selection

Architecture selection is evaluated not only by performance but also by:

Privacy:
Biometric data (eye, face) is extremely sensitive.
Edge processing is the preferred solution in academic literature.

Latency:
Gaze-based systems require millisecond-level responsiveness.
Edge architectures provide the lowest latency.

Energy consumption:
Streaming video to the cloud significantly increases power usage.
WiFi video transmission is one of the major constraints in hybrid architectures.

User experience:
Any interaction delay breaks the sense of direct control, degrading usability.
Thus, real-time HCI literature overwhelmingly favors edge processing.

3.8.5 Architectural Trends in Literature: General Patterns

The literature reveals consistent trends across HCI and smart home research:

- If real-time performance is essential → **edge processing**

- If maximum accuracy is needed → **cloud or GPU-accelerated models**

- If energy efficiency is required → **Zigbee/Thread low-power networks**

- If the system spans many devices → **hybrid architectures**

- If privacy is a priority → **local/edge-only processing**

These trends support the conclusion that **each architecture is optimized for specific scenarios**, and no single model is ideal for all systems.
Final architectural decisions must be based on application needs.

3.9 General Evaluation

This literature review has comprehensively examined the multidisciplinary research domain situated at the intersection of smart home automation, human–computer interaction, eye tracking, IoT communication protocols, computer vision algorithms, and accessibility technologies. Existing studies demonstrate that although smart home platforms have matured technically, they still present significant gaps in terms of accessibility. While systems such as Google Home, Alexa, and SmartThings possess extensive device ecosystems, user interaction is predominantly based on voice commands or mobile applications—both of which create substantial usability barriers for individuals with limited motor abilities.

The human–computer interaction literature identifies eye movements as a natural, rapid, and low-effort interaction modality, emphasizing that gaze-based systems offer high accessibility, particularly for users with motor impairments. However, the integration of eye tracking with IoT-based device control has been explored only to a limited extent in the existing research. This reveals a clear need for a new interaction paradigm within smart home systems.
Computer vision research spans a wide range of techniques—from classical methods to deep-learning models—for gaze tracking, and demonstrates that lightweight models such as MediaPipe Face Mesh can achieve real-time performance on embedded devices. The hardware literature similarly indicates that platforms such as Raspberry Pi, ESP32, and IR/RGB cameras are widely used for low-cost prototyping, while cloud and hybrid architectures are considered for scenarios requiring more computationally intensive models. Research on IoT protocols further shows that technologies such as MQTT, Zigbee, Thread, and Matter each provide distinct advantages and limitations depending on the application context.
Overall, the literature suggests that although the technical foundation of smart home automation is well established, there remains a critical gap in accessibility and alternative interaction methods. Transforming eye movements into a natural control signal and integrating this signal within an IoT ecosystem represents a novel contribution both technically and socially. Therefore, the proposed project directly targets the shortcomings identified in current research and offers an innovative solution aimed at enhancing accessibility in smart home systems.

4. System Requirements

4.1 Overview

The IoT Integrated Accessible Smart Home System enables users to control smart home devices through eye-tracking gestures such as looking left/right and short/long blinks. Manual command execution is also supported as an alternative control method.

4.2 Scope

The system includes:

- Video input via an IP camera

The system receives a real-time video stream from an IP camera or a smartphone connected over the local network.

- Eye-tracking and gesture recognition

The system analyzes eye movements and blink patterns from the video stream to detect user gestures.

- Smart home integrations (e.g., smart bulbs, smart plugs, Smart TVs)

Detected gestures are translated into commands that control supported smart home devices.

4.3 Target Users

- Users with limited hand mobility

Provides hands-free interaction for users who have difficulty using traditional input devices.

- Users unable to use voice assistants

Offers a non-verbal control method for users who cannot or prefer not to use voice commands.

- Anyone needing an alternative human–computer interaction method

Can be used by anyone seeking a different or more accessible interaction approach.

4.4 System Architecture

4.4.1 Image Capture Layer

- Reads video stream from an IP camera.

Continuously captures video data from a configurable IP camera source.

- Processes frames on Raspberry Pi using OpenCV.

Each video frame is processed locally on the Raspberry Pi using OpenCV for efficiency.

4.4.2 Eye-Tracking & Gesture Layer

- Detects face, eyes, and iris using MediaPipe Face Mesh.

Uses MediaPipe Face Mesh to accurately locate facial features and eye landmarks.

- Recognizes gaze direction and blink gestures.

Analyzes eye position and blink duration to identify user intentions.

- Produces high-level events such as LOOK_LEFT, LOOK_RIGHT, SHORT_BLINK, LONG_BLINK.

Converts low-level detections into meaningful system events.

4.4.3 Smart Home Control Layer

- Maps gesture events to navigation or device-control actions.

Associates detected gestures with specific user interface or device commands.

- Communicates with smart devices over LAN.

Sends control messages to smart devices using local network communication.

- Supports manual commands as a fallback control method.

Allows manual input when gesture-based control is unavailable or unsuitable.

4.5 Hardware Requirements

- Raspberry Pi 5 with network connectivity

Acts as the main processing unit for video analysis and device control.

- IP camera or smartphone streaming over LAN

Serves as the video input source for eye-tracking.

- Smart devices such as:

➢ Smart bulbs

➢ Smart plugs

➢ Smart TV

4.6 Software Requirements

- Raspberry Pi OS

Provides a stable and optimized operating system for the Raspberry Pi.

- Python 3 environment

Used for implementing all core system components.

- Core libraries:

Essential libraries required for system functionality.

➢ OpenCV

➢ MediaPipe

➢ NumPy

- Optional IoT libraries (MQTT, REST/SDK depending on device brand)

➢ python-kasa (for TP-Link Kasa smart bulbs/plugs over LAN)

4.7 Functional Requirements

4.7.1 Camera Input

- Reads video from a configurable IP camera stream.
- Displays an error message if the connection fails.

4.7.2 Face & Eye Detection

- Detects a single user’s face and extracts eye/iris landmarks.

4.7.3 Gaze Direction

- Classifies gaze into LEFT, RIGHT, CENTER, UP, or DOWN.
- Generates events when a gaze direction remains stable.

4.7.4 Blink Detection

- Recognizes short blinks as quick-action gestures.
- Recognizes long blinks as confirmation gestures.

4.7.5 Debug Logging

- Periodically logs FPS, gaze state, and gesture events.

4.7.6 Optional GUI

- Displays the camera feed with overlay information.
- Supports exit via keyboard.

4.7.7 Smart Home Interaction

- Converts gestures into actions such as navigation or toggling devices.
- Sends device commands via LAN protocols (MQTT or HTTP).
  - For TP-Link Kasa devices, commands can be sent using python-kasa.
- Smart TV control may include channel switching, volume adjustment, or power toggling.

4.7.8 Manual Command Mode

- Allows users to issue commands manually through text input.
- Useful for configuration or accessibility alternatives.

4.8 Non-Functional Requirements

- Provides real-time processing on Raspberry Pi.
- Recovers gracefully from camera disconnections.
- Allows calibration of gesture sensitivity.
- Processes video locally for privacy.
- Minimizes false gesture triggers.

4.9 Assumptions

- Adequate lighting is present for stable eye tracking.
- Only one user appears on camera.
- The user can perform basic gaze and blink gestures.

4.10 Constraints

- Optimized for Raspberry Pi 5 hardware.
- Accuracy may degrade under extreme lighting.
- Tested primarily with mobile IP camera applications.

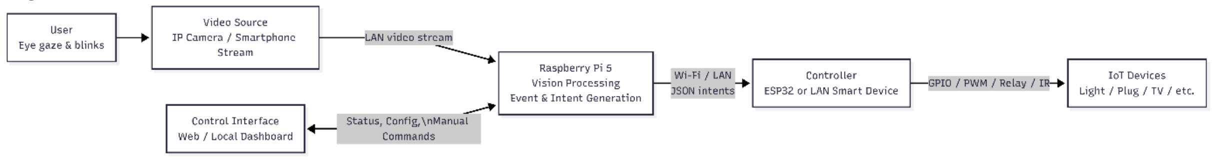
5. System Architecture

5.1 Architectural Overview

The IoT Integrated Accessible Smart Home System is designed as a **layered and modular architecture** that enables hands-free interaction with smart home devices using **eye-gaze and blink-based gestures**. The system integrates real-time computer vision, local decision-making, and network-based IoT control within a single coherent architecture.
At the center of the system is a **Raspberry Pi 5**, which functions as an **edge computing and control hub**. It continuously processes a video stream obtained from a camera source, detects gaze and blink gestures, converts these gestures into **high-level control intents**, and distributes these intents over the local network to device controllers or smart devices. A **control interface (UI)** is included in the architecture to allow visual feedback, configuration, and gesture-based navigation.
The architecture explicitly separates **visual perception**, **intent generation**, **user interface logic**, and **device actuation**, which improves scalability, maintainability, and system clarity.

5.2 Hardware Block Architecture

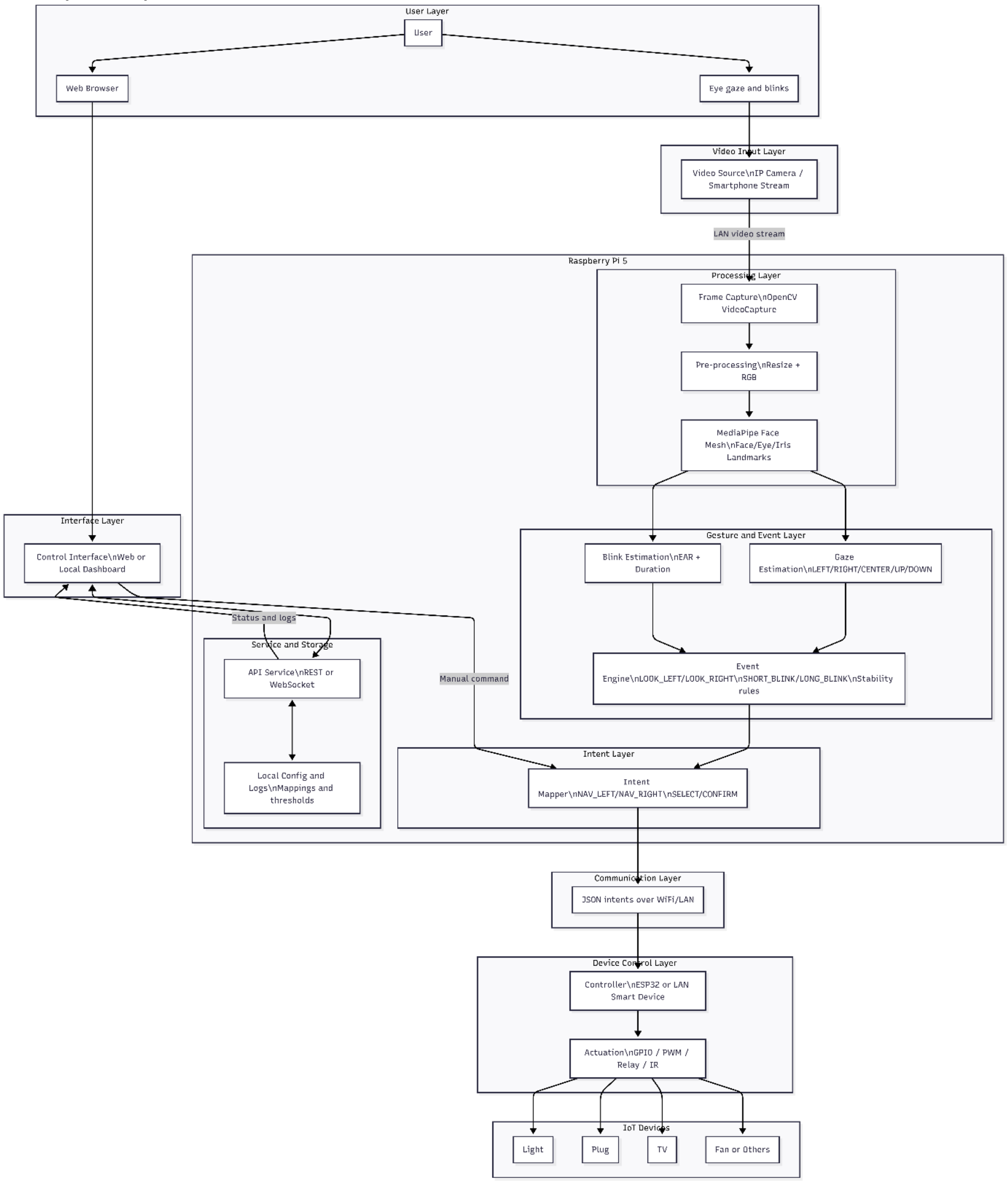
The following block diagram illustrates the main hardware components of the system and the high-level data and control flow between them.



Key architectural notes:

- The camera is treated as a **generic video source**, not tied to a specific hardware interface.
- The Raspberry Pi focuses on perception and decision-making.
- Physical device control is delegated to dedicated controllers (e.g., ESP32), ensuring electrical safety and modularity.
- The UI communicates with the same intent pipeline used by gaze-based control.

5.3 Layered System Architecture


User Layer
User
Web Browser
Eye gaze and blinks
Video Input Layer
Video Source\nIP Camera / Smartphone Stream
LAN video stream
Raspberry Pi 5
Processing Layer
Frame Capture\nOpenCV VideoCapture
Pre-processing\nResize + RGB
MediaPipe Face Mesh\nFace/Eye/Iris Landmarks
Interface Layer
Control Interface\nWeb or Local Dashboard
Gesture and Event Layer
Blink Estimation\nEAR + Duration
Gaze Estimation\nLEFT/RIGHT/CENTER/UP/DOWN
Status and logs
Service and Storage
API Service\nREST or WebSocket
Local Config and Logs\nMappings and thresholds
Manual command
Event Engine\nLOOK_LEFT/LOOK_RIGHT\nSHORT_BLINK/LONG_BLINK\nStability rules
Intent Layer
Intent Mapper\nNAV_LEFT/NAV_RIGHT\nSELECT/CONFIRM
Communication Layer
JSON intents over WiFi/LAN
Device Control Layer
Controller\nESP32 or LAN Smart Device
Actuation\nGPIO / PWM / Relay / IR
IoT Devices
Light
Plug
TV
Fan or Others

This diagram illustrates the layered structure of the system, showing how user input is captured, processed on the Raspberry Pi, translated into gesture events and high-level intents, and finally delivered to IoT device controllers. The architecture highlights the separation of perception, decision-making, user interface, communication, and actuation layers.

5.3.1 User Layer

The User Layer represents the interaction point between the human user and the system.
Users interact primarily through **eye-gaze direction** and **blink gestures**, which serve as non-verbal control inputs. This interaction model is especially suitable for users with limited hand mobility or those who require alternative interaction methods.
The system also provides visual feedback to the user through the control interface and the physical response of the controlled devices.

5.3.2 Video Input Layer
The Video Input Layer is responsible for acquiring real-time visual data.

- Video frames are obtained from a **configurable IP camera or smartphone camera stream** over the local network.
- Frames are captured on the Raspberry Pi using OpenCV-based video capture and resized to a fixed resolution for stable processing.
- The architecture abstracts the camera as a video source, allowing different camera types to be used without changing the rest of the system.

If the video stream becomes unavailable, the system detects the failure and handles it according to predefined error-handling policies.

5.3.3 Processing Layer (Raspberry Pi 5)
The Processing Layer performs all computationally intensive tasks required for gesture recognition and control.
Key responsibilities include:

- Preprocessing of incoming frames (resizing, color space conversion),
- Extraction of face, eye, and iris landmarks using **MediaPipe Face Mesh**,
- Runtime performance monitoring (e.g., FPS measurements and logs).

All processing is executed locally on the Raspberry Pi to minimize latency, preserve user privacy, and avoid dependency on external services.

5.3.4 Gesture Detection and Event Generation Layer
This layer converts raw visual landmarks into meaningful interaction events.
Eye-Gaze Detection

- The relative position of the iris within the eye region is used to classify gaze direction as LEFT, RIGHT, CENTER, UP, or DOWN.
- A center tolerance region is applied to suppress minor, unintentional eye movements.

Blink Detection

- Eye Aspect Ratio (EAR) is used to determine eye-open and eye-closed states.
- Blink duration is analyzed to distinguish between **short blinks** and **long blinks**, enabling different semantic meanings.

Stability Mechanisms

- Gaze-based events are emitted only if the detected gaze direction remains stable for a predefined duration.
- These mechanisms significantly reduce false positives and unintended commands.

The output of this layer consists of discrete events such as LOOK_LEFT, LOOK_RIGHT, SHORT_BLINK, and LONG_BLINK.

5.3.5 Intent Mapping Layer
The Intent Mapping Layer translates detected events into **device-agnostic control intents**. Rather than binding gestures directly to specific devices, the system generates abstract intents such as:

- NAV_LEFT, NAV_RIGHT
- SELECT

- CONFIRM or TOGGLE

This abstraction allows the same gesture set to be reused across different devices and user interfaces and enables flexible system expansion.

5.3.6 User Interface Layer

The User Interface Layer provides visual feedback and interaction management.
The interface is designed to:

- Display system status, detected gestures, and active intents,
- Present navigable UI elements that can be controlled using gaze (navigation) and blinks (selection/confirmation),
- Allow configuration and calibration of system parameters,
- Support manual command input as a fallback interaction method.

The UI consumes the same intents generated by the system, ensuring consistency between gesture-based control and manual interaction.

5.3.7 Communication Layer

The Communication Layer handles the transfer of intents and status information over the local network.

- Control messages are encoded as **JSON**.
- Communication occurs over **Wi-Fi or Ethernet (LAN)**.
- The architecture is transport-agnostic, allowing different network protocols to be used without altering the core system design.

5.3.8 IoT Device Control Layer

The IoT Device Control Layer executes control intents on physical devices.

- **ESP32-based controllers** or LAN-enabled smart devices receive intents from the Raspberry Pi.
- Controllers translate intents into device-specific actions using GPIO, PWM, relay switching, or infrared transmission.
- This design avoids direct high-voltage switching from the Raspberry Pi and improves system safety.

Supported device categories include lighting systems, smart plugs, televisions, and other household appliances.

5.4. End-to-End Control Flow

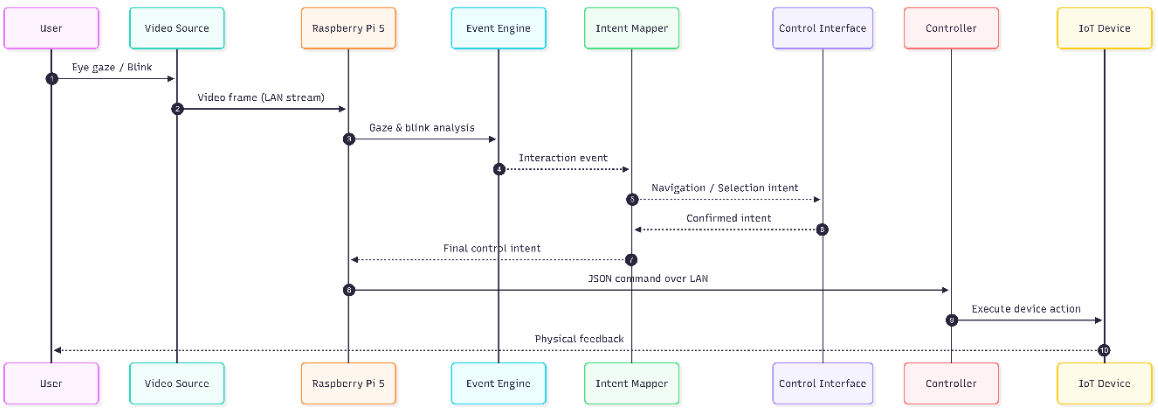


Figure 3: System Sequence Diagram.

This diagram illustrates that the process starts with the user gazing or blinking at the video source (camera); the captured image data is then transmitted to the Raspberry Pi 5 for processing. The Raspberry Pi analyzes this data ('Gaze & blink analysis') and identifies the user's interaction intent (navigation or selection) via the Event Engine and Intent Mapper modules. Once validated by the Control Interface, this intent is converted into a final control command and sent via the Raspberry Pi to the Controller unit in JSON format. In the final stage, the Controller processes this command to execute the physical action on the target IoT Device, completing the loop by providing physical feedback (such as a light turning on or a sound) to the user indicating that the operation is complete.

5.5 Architectural Rationale

This architecture:

- Clearly separates perception, decision-making, interface logic, and actuation,
- Supports both gesture-based and interface-based control paths,
- Is scalable to multiple devices and controllers,
- Aligns with real-time constraints and privacy considerations,
- Is suitable for incremental implementation across project phases without requiring architectural changes.