

Design and Implementation of an IoT-Based Smart Home System for Motion-Triggered Object Detection and Automated Ventilation Control

Mayang Sari¹, Apri Siswanto^{2*}

^{1, *2} Universitas Islam Riau, Jl. Kaharuddin Nasution N0. 113, Pekanbaru, 28000, Indonesia

¹email : mayangsari559@student.uir.ac.id

^{*2}email: aprisiswanto@eng.uir.ac.id

(Article Received: 28 October 2026; Article Revised: 18 November 2026; Article Published: 1 December 2026;)

ABSTRACT – Internet of Things (IoT)-based smart homes enable the automation of home devices to improve comfort and security. This study designs and develops a smart home system in which passive infrared (PIR) sensors are used only to detect motion, while an ESP32-CAM and YOLOv3 are used for camera-based object detection and class labeling. A human-presence indication is therefore generated only when a PIR-triggered camera frame contains a person detected by YOLOv3; PIR motion alone is not interpreted as occupancy. The system also features MQ-2 sensor-based gas leak detection and automatic ventilation control using a servo motor as a safety response. Data and alerts are transmitted through Telegram, while Blynk is used for integrated monitoring. System development followed requirements analysis, design, implementation, functional testing, and corrective refinement. The dedicated YOLOv3 object test documented 11 confidence values ranging from 53% to 99% (descriptive mean 86.2%; median 94%); these values are detector confidence scores and not classification accuracy because no annotated ground-truth benchmark was used. In an additional 11-trial illuminance-oriented functional test, object detection succeeded in all six trials conducted at 624–950 lux and failed in all five trials conducted at 0–147 lux. The contribution of this work is an event-driven IoT architecture that coordinates motion sensing, visual verification, remote notification, and closed-loop gas mitigation, together with field-oriented evaluation of lighting effects on low-cost camera-based detection.

Keywords - Smart home, ESP32-CAM, YOLOv3, motion detection, IoT.

Rancang Bangun Sistem Smart Home Berbasis IoT untuk Deteksi Gerakan, Deteksi Objek, dan Pengendalian Ventilasi Otomatis

ABSTRAK – Smart home berbasis Internet of Things (IoT) memungkinkan otomasi perangkat rumah untuk meningkatkan kenyamanan dan keamanan. Penelitian ini merancang dan mengembangkan sistem smart home yang menggunakan sensor passive infrared (PIR) hanya untuk mendeteksi gerakan, sedangkan ESP32-CAM dan YOLOv3 digunakan untuk deteksi objek berbasis kamera dan pemberian label kelas. Indikasi keberadaan manusia hanya ditetapkan ketika citra yang dipicu oleh PIR menunjukkan objek manusia yang terdeteksi oleh YOLOv3; gerakan PIR saja tidak diinterpretasikan sebagai okupansi. Sistem juga dilengkapi deteksi indikasi kebocoran gas berbasis sensor MQ-2 dan pengendalian ventilasi otomatis menggunakan motor servo sebagai respons keselamatan. Data dan peringatan dikirim melalui Telegram, sedangkan Blynk digunakan untuk pemantauan terintegrasi. Pengembangan sistem mencakup analisis kebutuhan, perancangan, implementasi, pengujian fungsional, dan penyempurnaan korektif. Pengujian khusus YOLOv3 mendokumentasikan 11 confidence score dengan rentang 53%–99% (rata-rata deskriptif 86,2%; median 94%); nilai tersebut merupakan confidence score detektor dan bukan akurasi klasifikasi karena penelitian tidak menggunakan benchmark ground truth beranotasi. Pada pengujian fungsional tambahan berbasis pencahayaan sebanyak 11 percobaan, deteksi berhasil pada seluruh enam percobaan dengan 624–950 lux dan gagal pada seluruh lima percobaan dengan 0–147 lux. Kontribusi penelitian terletak pada arsitektur IoT berbasis kejadian yang mengoordinasikan sensor gerak, verifikasi visual, notifikasi jarak jauh, dan mitigasi gas secara closed-loop, serta evaluasi lapangan terhadap pengaruh pencahayaan pada deteksi berbasis kamera berbiaya rendah.

Kata Kunci – Smart home, ESP32-CAM, YOLOv3, deteksi gerakan, IoT.

1. INTRODUCTION

Advancements in information and communication technology have driven the

emergence of various automation systems applied in daily life. A rapidly growing implementation of this is the "smart home" concept—a residence where devices can be monitored and controlled via an

internet connection. This concept is closely linked to the Internet of Things (IoT), which enables physical devices to send and receive data in real-time, thereby supporting automation, monitoring, and decision-making based on environmental conditions [1-3].

In a residential context, security and safety are paramount, particularly when occupants are away or during the night when active monitoring is reduced [4]. Security risks, such as potential burglary, arise when the home is unoccupied, while safety risks include gas leaks caused by negligence—such as forgetting to turn off the stove—which can trigger fires [5]. Furthermore, remote monitoring presents a challenge for highly mobile families—such as parents who frequently work out of town—who require timely information regarding the home's status and activities within it [6, 7].

Research on smart homes generally focuses on specific functions, such as gas leak detection, motion detection, or camera-based monitoring. However, practical implementations often reveal limitations in system integration; for instance, gas leak detection may not trigger automatic mitigation measures, or security systems might provide notifications without accompanying visual documentation or clear object labels [8]. Environmental conditions, particularly lighting, may also affect the reliability of camera-based object detection and therefore need to be evaluated under realistic use conditions [9, 10]. These gaps motivate an integrated system that can detect events, obtain visual evidence, respond automatically, and present useful information to the user. In the proposed architecture, the PIR sensor is treated strictly as a motion trigger. It does not determine room occupancy by itself. When motion is detected, the ESP32 triggers the ESP32-CAM and the resulting stream is processed by YOLOv3; only a YOLOv3 person detection is used as visual evidence of human presence. For safety, the MQ-2 module triggers an alarm and servo-controlled ventilation when its reading exceeds the configured threshold. Event information and system status are transmitted through Blynk and Telegram for remote monitoring.

System testing was conducted in stages to verify the functionality of each component: testing the MQ-2 sensor with gas exposure from a lighter (using a reading threshold >1000 to trigger the alarm, notifications, and automatic ventilation opening); testing the PIR sensor and ESP32-CAM at distances of 1-4 meters to evaluate motion detection capabilities and image quality; testing live streaming for remote access; and testing YOLOv3 object detection to observe classification results and confidence levels. Continuous testing was also carried out over a seven-day period—covering morning, afternoon, evening, and night—to assess

the stability of the system's response under daily usage conditions. When detection failures occurred in low-light conditions, the evaluation was expanded to include light intensity (lux) measurements to understand the impact of lighting on object detection performance.

This testing approach is relevant when compared to the study by Danang et al. [9], which developed an IoT-based home security prototype using the NodeMCU ESP8266, ESP32-CAM, and a combination of sensors with automatic Telegram notifications, using the Kembang Sari Regency housing complex as a case study and the waterfall development method [11]. While Danang et al. emphasized device integration and photo transmission via Telegram, the present study extends the architecture by adding MQ-2 gas/smoke sensing, a NodeMCU ESP32, and YOLOv3-based visual object detection. The present evaluation also includes a seven-day stability test and an analysis of illumination (lux), which were not primary focuses of the comparative study.

One relevant study is that conducted by Raharja et al. [12], titled "Web-Based IoT Smart Home Prototype using NodeMCU ESP8266, Servo Motor, and DHT11 Temperature Sensor," which focuses on developing a smart home system for the remote control of household appliances via a website. The system utilizes the NodeMCU ESP8266, a DHT11 temperature sensor, and supporting components such as relays, LED lights, DC fans, and servo motors. Implementation was carried out using the Arduino IDE and Microsoft Visual Studio. The results indicate that the prototype operates effectively at internet speeds of approximately 100 kbps, with temperature data updates occurring every 1-2 seconds and actuator response times ranging from 1 to 3 seconds. The system also provides low-latency notifications (around 2 seconds) when the temperature exceeds a set threshold. That study employed the prototyping method [13]. In contrast, the current study utilizes the waterfall method, integrates an ESP32-CAM, employs the NodeMCU ESP32, and implements YOLOv3 for object detection.

Research by Musyafa [14], titled "Smart Home Design with Internet of Things (IoT) Concept using NodeMCU ESP8266 via Telegram Bot," emphasizes controlling home electronic devices using a Telegram Bot as both an input medium and a notification tool. A NodeMCU ESP8266 microcontroller is used to control devices such as porch lights, bedroom lights, and fans via relay modules. The results show that the system can be successfully implemented and allows for device control via text commands, while security is maintained through restricted user access [15]. This study differs by incorporating gas leak detection (MQ-2), motion detection (PIR), servo-based

automatic ventilation control, ESP32-CAM integration, the use of NodeMCU ESP32, and object detection using YOLOv3.

Research by Maulana et al. [16], titled "Internet of Things-Based Smart Home System Prototype," aims to build a smart home prototype to enhance efficiency, comfort, and security for residents. The system was developed to be controlled via an Android application, allowing users to manage home devices more conveniently. The research employed the waterfall methodology, encompassing requirements analysis, design, implementation, testing, and maintenance. The results demonstrated the successful creation of a functional prototype utilizing components such as the ESP8266 and various sensors, including the MQ-2. This study differs from the aforementioned research by incorporating a PIR sensor, utilizing the NodeMCU ESP32, integrating the ESP32-CAM, and implementing YOLOv3 for object detection and enhanced visual-based security.

Research by [17], titled "Design and Development of an IoT-Based Smart Home System for Boarding Houses," developed a system emphasizing remote electrical control, smoke detection, energy monitoring, and room temperature and humidity monitoring. The system utilized IoT concepts to connect devices and services via the internet, employing an Android application as the interface for control and monitoring. Although it integrated key aspects such as smoke detection and environmental monitoring, that study did not emphasize camera-based security or object classification. In contrast, the current study integrates the ESP32-CAM and implements YOLOv3 for object classification, while also utilizing a PIR sensor to trigger motion-based detection.

In general, previous studies indicate that IoT-based smart home systems have been widely implemented for device control, notifications, and certain security aspects. However, there remains room for development, particularly regarding the integration of motion detection linked to cameras and automated object classification, as well as gas leak detection coupled with immediate automated mitigation actions. Addressing these gaps, this study proposes a more integrated system featuring ESP32-CAM support, dual notifications via Blynk and Telegram, and YOLOv3-based object detection to bolster security and safety.

The novelty of this work is not a new object-detection algorithm or a new gas sensor. Instead, the technical contribution lies in the event-driven coordination of existing components within a unified sensing-to-action architecture. Specifically, PIR motion events initiate camera acquisition and

YOLOv3 processing through MQTT rather than being interpreted directly as occupancy; MQ-2 events initiate a closed-loop safety response consisting of alarm activation, automatic ventilation actuation, and remote notification; and Telegram and Blynk provide complementary remote-information channels. The scientific contribution is the field-oriented characterization of this integrated architecture over a seven-day usage period, including additional illuminance (lux) observations to identify the operational effect of lighting on low-cost camera-based object detection. Accordingly, the study contributes system architecture and empirical operational evidence rather than a new YOLO model.

2. METHOD

The method employed in this study to develop the Internet of Things (IoT)-based smart home prototype is the waterfall method. The waterfall method is an early System Development Life Cycle (SDLC) approach that emphasizes sequential and systematic development stages [18, 19]. The stages considered in this study were requirements analysis, system design, implementation, testing, and maintenance/corrective refinement. Because the work evaluates a research prototype rather than a long-term production deployment, the maintenance stage was limited to post-test corrective refinement and diagnosis of observed failures. In particular, low-light detection failures identified during field testing motivated additional illuminance (lux) measurements, while communication logic included repeated MQTT reconnection attempts and periodic NTP synchronization. Long-term preventive maintenance and component replacement were outside the scope of the present experiment. The workflow is illustrated in Figure 1.

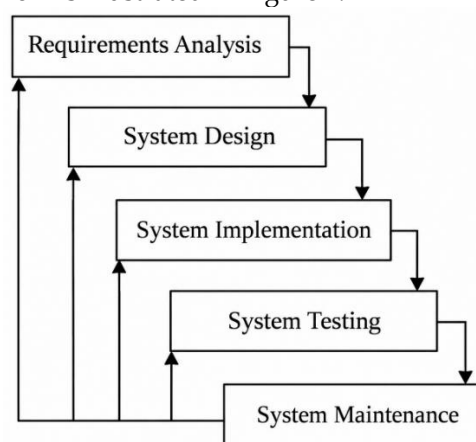


Figure 1. Waterfall method

A. Requirements Analysis

The requirements analysis phase aims to clearly identify and formulate system requirements,

encompassing both functional and non-functional aspects [20, 21]. This phase involves gathering relevant information regarding the smart home system to be developed, including the selection of the microcontroller, sensors, actuators, and the communication and notification media to be employed. The functional requirements analyzed include motion detection using a PIR sensor, gas leak detection using an MQ-2 sensor, alarm activation and automatic ventilation opening via a servo motor, image capture and live streaming using the ESP32-CAM, notification transmission to Telegram and Blynk, and object classification using YOLOv3 based on the camera's video stream. Meanwhile, non-functional requirements include Wi-Fi connection reliability, rapid system response, and the system's ability to operate stably in remote monitoring scenarios [22].

B. System Design

The system design phase is conducted to devise a solution that meets the established requirements, ensuring the developed system addresses the research problem [23, 24]. In this phase, the researcher develops the system architecture and workflow using various modeling forms, including block diagrams, flowcharts, use case diagrams, sequence diagrams, and activity diagrams, as well as hardware design and component placement [25].

1. Block Diagram

A block diagram is a system representation that depicts primary functions as blocks and the relationships between them via connecting lines [26, 27]. In this study, system inputs originate from the PIR sensor, MQ-2 sensor, and servo actuator status. Processing is performed by the NodeMCU ESP32, ESP32-CAM, and the YOLOv3 object detection module. System outputs consist of an alarm (buzzer) and notifications via Telegram and Blynk.

The system workflow illustrated in the block diagram is as follows:

- When the PIR sensor detects motion, the ESP32 sends a trigger signal to the ESP32-CAM via MQTT.
- The ESP32-CAM then transmits a live streaming link for processing by YOLOv3, allowing detected objects to be classified (e.g., humans, animals, or inanimate objects). The detection results are subsequently sent to Telegram.
- When the MQ-2 sensor detects a gas leak exceeding the threshold, the system activates an alarm and moves the servo to a 90° angle to automatically open the ventilation. Notifications regarding the gas status are sent via Blynk (and can also be extended to Telegram as needed). The

system block diagram is shown in Figure 2.

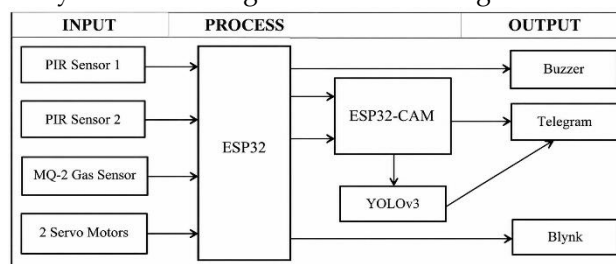


Figure 2. Diagram block

2. ESP32 Smart Home Flowchart

Flowcharts are used to describe system workflow systematically so that it is easy to understand [28]. On the ESP32, the program flow begins with Serial initialization, pin configuration, servo, and WiFiManager for the network connection process. Once the device is connected to Wi-Fi, the system connects to Blynk and the MQTT broker. If the MQTT connection fails, the system will try again until successful. In the main loop, the ESP32 reads the MQ-2 sensor value and the status of the PIR 1 and PIR 2 sensors, then sends the data to Blynk. If the PIR detects movement, the ESP32 sends a trigger message via MQTT and checks the system time; The alarm is only active when movement occurs outside the occupant's activity hours. On the gas module, if the MQ-2 value is above the threshold (≥ 1000), the alarm activates and the servo moves to 90°; if the value returns below the threshold, the alarm stops and the servo returns to the 0° position. Additionally, the system updates the time via NTP periodically and processes incoming messages via MQTT callback. The ESP32 flowchart is shown in Figure 3.

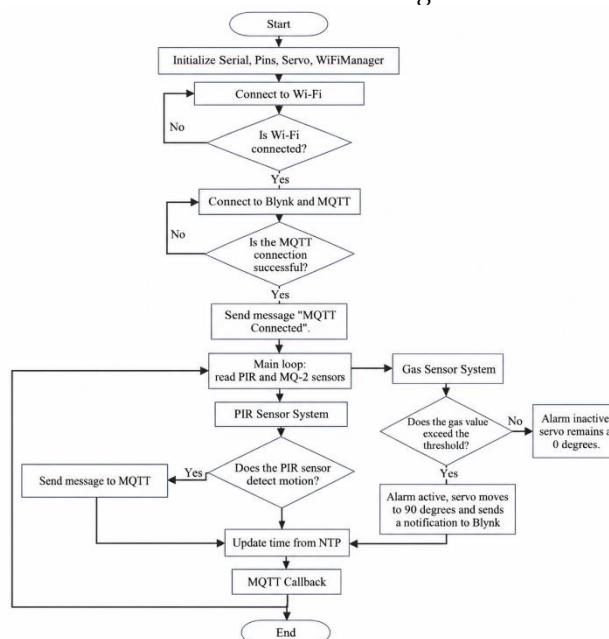


Figure 3. Smart home flowchart

3. ESP32-CAM Flowchart

For the ESP32-CAM, the workflow begins with

program initialization, NTP time synchronization, and the establishment of an MQTT connection. The system then monitors incoming MQTT messages—such as PIR motion detection commands, image capture requests, live streaming link transmissions, or object detection processing triggers. Upon receiving a trigger from the ESP32 (e.g., PIR motion detection), the ESP32-CAM captures an image and/or prepares a live stream according to the system scenario, subsequently sending the relevant information and visual data to Telegram. The system also handles streaming requests from the HTTP server; when a request is received, the ESP32-CAM transmits the stream for monitoring purposes, which is then processed using YOLOv3. The ESP32-CAM flowchart is shown in Figure 4.

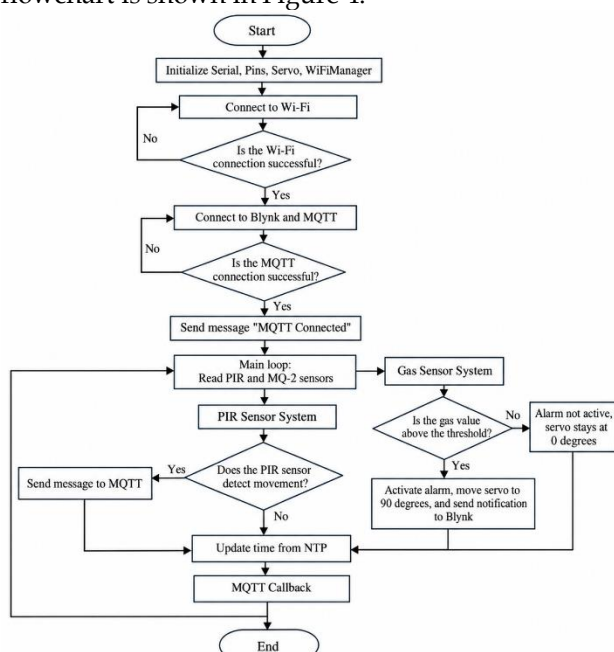


Figure 4. ESP32 cam flowchart

4. Flowchart of Object Detection Code Using YOLOv3 and MQTT

The YOLOv3 module operates on the processing side (server/computer), receiving the video feed from the ESP32-CAM. The initial stage involves setting up the ESP32-CAM streaming URL and configuring the MQTT broker (address, port, and topics for receiving commands and transmitting detection results). The system also prepares a storage directory to document detection results. Subsequently, the YOLOv3 model is loaded, and the MQTT client is initialized. Once the connection is established, the system opens the video stream and performs object detection. If an object is detected, the system saves the relevant frame and transmits a summary of the detection results via MQTT. After the detection process concludes, the results are forwarded to Telegram (via the ESP32-CAM, in accordance with the system workflow), and

the homeowner can access the live streaming link for real-time monitoring. The flowchart for the YOLOv3 module is shown in Figure 5.

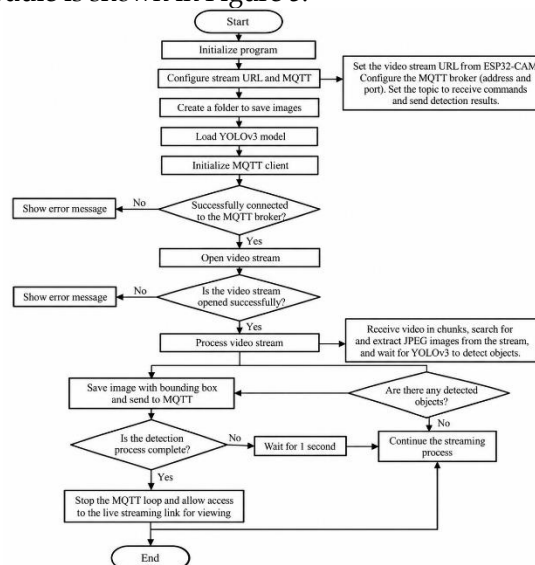


Figure 5. Object detection with YOLOv3 & MQTT

5. Use Case Diagram

A use case diagram illustrates the interactions between actors and the system to achieve specific functions, thereby aiding in the understanding of functional requirements and the relationships between features [29]. In this study, the primary actor is the homeowner. System interactions include monitoring notifications, receiving photos or detection results, and accessing live streaming. When the PIR sensor detects motion, data is sent to Blynk, and a trigger message is transmitted via MQTT to command the ESP32-CAM to capture an image and stream the feed to YOLOv3. Object detection results are then forwarded to Telegram. When the MQ-2 sensor detects a gas leak, an alarm activates, a servo opens the ventilation, and a notification is sent via Blynk (and optionally via Telegram, as per the design). The use case diagram is shown in Figure 6.

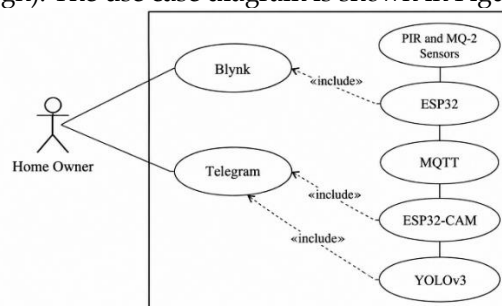


Figure 6. Use case diagram

6. Sequence Diagram

A sequence diagram illustrates the sequence of message exchanges between objects or systems over a specific timeframe. This diagram helps clarify the communication flow between devices and

services. In this system, when the PIR sensor detects movement, the ESP32 sends a trigger message to the ESP32-CAM. The ESP32-CAM then transmits a live streaming URL to the YOLOv3 module via MQTT. YOLOv3 processes the video, classifies objects, and sends the detection results back via MQTT. The ESP32-CAM forwards these results to Telegram. For the gas leak scenario, when the MQ-2 sensor detects a value exceeding the threshold, the ESP32 activates an alarm, operates a servo to open the ventilation, and sends a notification to Blynk. The sequence diagram is shown in Figure 7.

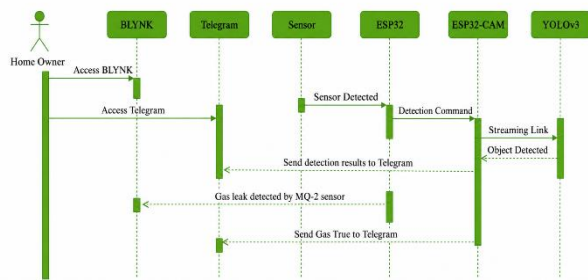


Figure 7. Smart Home Gas Detection Sequence Diagram

7. Activity Diagram

An activity diagram is a UML diagram that illustrates the flow of activities and decisions within a system process [30]. Two activity diagrams were created for this study:

a. PIR sensor activity diagram

The PIR sensor detects motion. If motion occurs outside the occupant's active hours, the alarm is activated; if it occurs during active hours, the alarm remains inactive. Each motion detection triggers data transmission from the ESP32 to MQTT, which is then forwarded to the ESP32-CAM; the ESP32-CAM then prepares a stream for YOLOv3. The object detection results are sent back and forwarded as a notification to Telegram. The PIR activity diagram is shown in Figure 8.

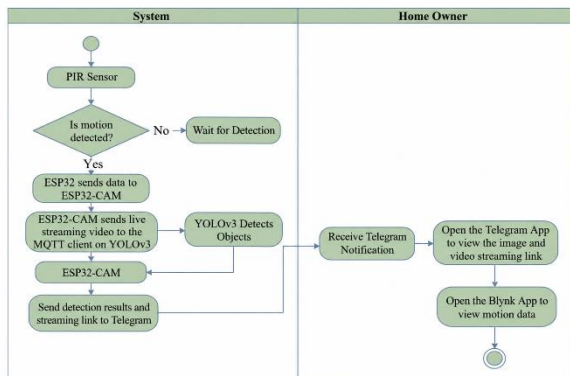


Figure 8. PIR sensor activity diagram

b. MQ-2 sensor activity diagram

The MQ-2 sensor monitors for gas leaks. When the gas level exceeds the threshold, the alarm activates and the servo moves to 90° to open the vent.

The ESP32 then sends a notification to Blynk (and can also be configured to send notifications via Telegram if required). When the gas level returns to normal, the alarm stops and the servo returns to 0°. The MQ-2 activity diagram is shown in Figure 9.

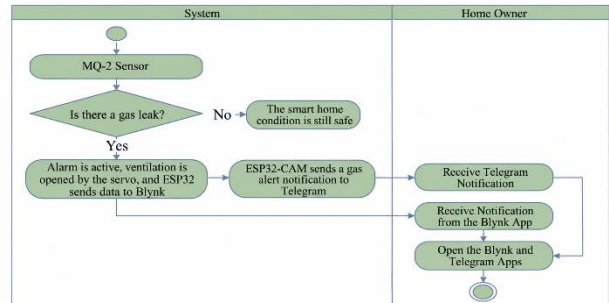


Figure 9. MQ2 sensor activity diagram

8. Smart Home System Design

The smart home system design in this study encompasses several areas. Sensors are installed in three main zones: the living room, the kitchen, and the garage. The system utilizes two PIR sensors, an MQ-2 sensor, two servo motors, a buzzer, and two processing units—the NodeMCU ESP32 and the ESP32-CAM. The MQ-2 sensor is placed in the kitchen to detect gas leaks. A servo motor is installed on the kitchen vent to automatically open it when a gas leak is detected. PIR Sensor 1 is mounted at the front door to monitor entry and exit activity, while PIR Sensor 2 is installed in the garage to enhance security in that area. A buzzer is placed inside the house to provide alerts in the event of a gas leak or movement outside of active hours. The component placement design is illustrated in Figure 10.



Figure 10. Smart home design

9. Hardware Design

Hardware design is conducted to ensure that all components can be integrated and operate according to the system's functions [31]. In this study, the design was created using the Fritzing application. The components used include an ESP32, an ESP32-CAM, two PIR sensors, an MQ-2 sensor, a buzzer, two servo motors, and a power supply/battery. The hardware design is shown in Figure 11.

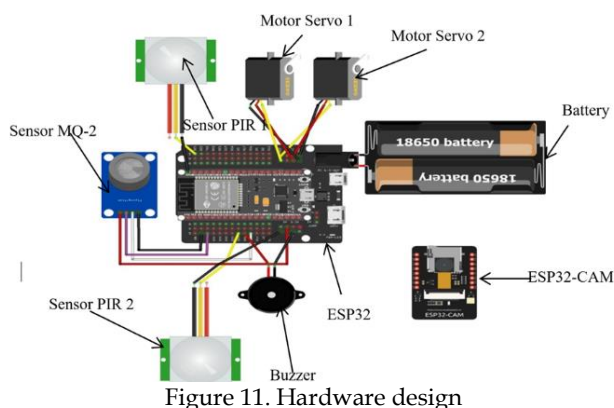


Figure 11. Hardware design

C. Implementation and Experimental Configuration

The implemented prototype consisted of a NodeMCU ESP32 as the main controller, an ESP32-CAM equipped with an OV2640 2 MP camera, two passive infrared (PIR) sensors, one MQ-2 gas and smoke sensor, two servo motors, a buzzer, a 12 V/3 A power supply, and a battery backup system. The sensing and actuation devices were distributed across three functional zones: the front-door/living-room area, kitchen, and garage. PIR Sensor 1 was installed in the front-door area to detect human motion, while PIR Sensor 2 monitored motion in the garage. The MQ-2 sensor was installed in the kitchen to monitor the presence of smoke or combustible gases, with a servo motor used to actuate the ventilation mechanism when the predefined threshold condition was reached. The ESP32-CAM provided visual input for the YOLOv3-based object detection module, which was used to identify detected objects and complement the motion information obtained from the PIR sensors. Therefore, PIR sensor activation was interpreted as motion detection rather than direct confirmation of room occupancy. Occupancy-related information was inferred from the combination of motion sensing and camera-based object detection. For reproducibility, the specific NodeMCU ESP32 board revision, PIR sensor model, servo motor model, sensor interface configuration, operating voltage, and relevant electrical connections should be explicitly reported according to the components used in the experimental prototype.

The embedded firmware was developed in C++ using the Arduino IDE, while the YOLOv3-based object detection module was implemented in Python and executed on a computer running Windows 10. The Arduino environment was used to program the ESP32-based controller for sensor acquisition, actuator control, and communication between the IoT components, whereas Python was used to perform image processing and object detection on image frames obtained from the ESP32-CAM. The YOLOv3

detector processed the captured images to identify objects relevant to occupancy-related monitoring and complemented the motion information provided by the PIR sensors. The available source report specifies the general software environments used in the prototype but does not document the exact Arduino IDE version, ESP32 board package and library versions, Python and image-processing framework versions, YOLOv3 configuration and weight files, dataset or class-name source, input image resolution, confidence threshold, Intersection over Union (IoU) threshold, non-maximum suppression (NMS) parameters, or detailed specifications of the processing computer. These implementation parameters should therefore be documented in future replication studies to improve reproducibility and facilitate direct performance comparison across different hardware and software environments.

Communication among the system components was implemented using an MQTT-based publish/subscribe mechanism. The ESP32 functioned as the primary publisher, transmitting event data generated by the PIR motion sensors and the MQ-2 gas/smoke sensor to the MQTT broker. The broker then distributed the corresponding messages to subscribed components. Upon receiving a relevant trigger, the ESP32-CAM captured an image and provided access to its live video stream. The captured visual data were subsequently processed by the YOLOv3-based detection module to identify objects associated with occupancy-related events. Telegram was used as the notification interface to deliver event messages, captured images, and, when required, a fallback link to the live camera stream, while Blynk provided real-time visualization of sensor readings and system status. For remote access, the locally hosted ESP32-CAM video stream was exposed through an ngrok tunnel, allowing the stream to be accessed outside the local network. The source report describes the overall communication architecture but does not document the specific MQTT broker implementation, broker version, host address, communication port, topic structure, Quality of Service (QoS) level, authentication or TLS configuration, Wi-Fi/network configuration, or ngrok version. Consequently, these parameters should be reported explicitly in future implementations or replication studies to improve system reproducibility and enable a more complete evaluation of communication reliability and security.

The experimental evaluation comprised: (1) seven MQ-2 functional trials using lighter-gas exposure with a sensor reading threshold of ≥ 1000 to activate the buzzer and move the ventilation servo to 90 degrees; (2) PIR and ESP32-CAM tests at 1-4 m; (3) external live-stream access through ngrok; (4)

motion-triggered YOLOv3 object detection; (5) a seven-day field test covering multiple daily time periods; and (6) additional lux measurements after low-light failures were observed. In the implemented error-handling logic, a failed YOLOv3 request could be retried for up to five attempts; if detection still failed, the system sent an error message and live-stream link through Telegram so that the user could inspect the event manually. The PIR workflow also used a delay of approximately 25 s to 1 min before the next motion event to reduce repeated notifications. The YOLOv3 values reported in this study are detector confidence scores. No annotated ground-truth benchmark set was collected in the original experiment; therefore, precision, recall, F1-score, mAP, and a confusion matrix are not reported, and the present evaluation should be interpreted as a functional system evaluation rather than a model-accuracy benchmark.

3. RESULT

A. System Implementation

The smart home system was successfully implemented by integrating the NodeMCU ESP32, ESP32-CAM, PIR sensor, MQ-2 sensor, ventilation servo, and alarm (buzzer), along with Telegram and Blynk notification services. All components were assembled into a single system capable of monitoring, detection, and automated actions based on home security and safety scenarios. The hardware implementation results are shown in Figure 12.



Figure 12. Hardware implementation

B. Gas Leak Detection Testing (MQ-2)

Testing of the MQ-2 sensor was conducted by exposing it to gas from a lighter. The system utilizes a threshold value of ≥ 1000 to indicate a hazardous condition. The results demonstrate that the MQ-2 sensor remained active and responsive throughout all trials (seven tests). When the gas level exceeded the threshold, the alarm activated and the servo opened the vent to 90° ; once levels returned to normal, the servo reverted to 0° . The alarm duration

ranged from 5 to 8 seconds, indicating the system's capability to consistently provide warnings and automatic mitigation. Further details regarding the gas leak detection test can be seen in Figure 13.

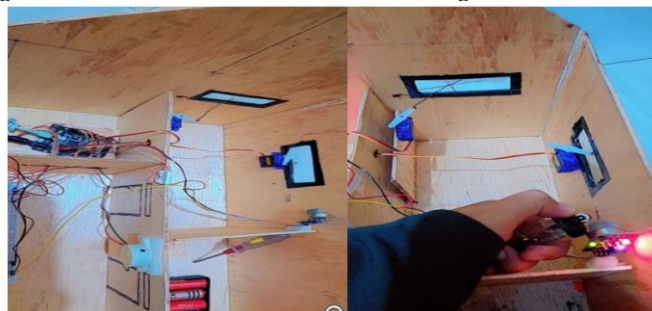


Figure 13. Gas leak detection test

C. Motion Detection (PIR) and Camera (ESP32-CAM) Testing

Testing of the PIR sensor and ESP32-CAM was conducted at distances ranging from 1 to 4 meters. The results indicate that the PIR sensor successfully detected movement across the entire test range (1-4 meters). The ESP32-CAM successfully captured images whenever the PIR sensor triggered a detection, and notifications were successfully sent via Telegram. A primary limitation was image quality under low-light conditions, which affected image clarity and the vision-based detection process.

D. Remote Access Testing via Live Streaming

The live streaming system was successfully deployed using ngrok, enabling users to access the streaming link from outside the local network. This feature serves as a backup monitoring method in the event that object detection fails due to environmental conditions. Remote access testing via live streaming is illustrated in Figure 14.

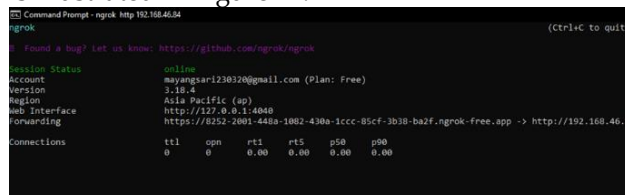


Figure 14. Live streaming test of the movement

E. Object Detection Testing using YOLOv3

YOLOv3 was tested using video input from the ESP32-CAM after a PIR motion trigger. The dedicated object test documented 11 prediction confidence values: bird 94% and 96%, cat 73%, bicycle 99%, horse 99%, cow 92%, human 96%, mobile phone 85%, dog 98%, motorcycle 63%, and a second cow detection at 53%. Across these 11 values, the descriptive mean confidence was 86.2% and the median was 94%, with an observed range of 53%-99%. These statistics summarize the detector's reported confidence outputs only and must not be interpreted as classification accuracy. Because the

experiment did not use independently annotated ground-truth frames, model-level metrics such as precision, recall, F1-score, mAP, and a confusion matrix cannot be calculated from the available data. Functionally, the system transmitted the predicted label, confidence score, text, and/or image through Telegram, thereby providing more specific event information than a PIR-only motion notification. Table 1 presents representative detections from this test.

Table 1. Representative YOLOv3 detections and confidence scores

Label and confidence score	YOLOv3 detection results
Bird 94% and bird 96%	
Cat 73%	
Bicycle 99%	
Horse 99%	
Cow 92%	
Human 96% and mobile phone 85%	
Dog 98%	

F. 7-Day Field Test (Motion-Triggered Visual Detection)

The test was conducted over a 7-day period during specific time slots (morning, midday, afternoon, and night for Monday–Friday; night and morning for Saturday–Sunday). Key findings:

- The system successfully detected movement via PIR sensors and sent notifications.
- The alarm activated primarily outside the residents' active hours, indicating that the time-based logic function was operating correctly.
- YOLOv3 detection behavior varied with environmental conditions. The seven-day comparison reported maximum individual confidence scores of 98% on Monday, 99% on Tuesday, 93% on Wednesday, 96% on Thursday, and 96% on Friday. During the weekend monitoring periods, the highest reported confidence scores were 86% on Saturday and 75% on Sunday. These are event-level confidence scores rather than accuracy measures. Detection failures were repeatedly documented during low-light morning, evening, and night periods, while higher-confidence detections were more common when illumination was adequate.
- The field results also showed that the alert chain remained operational when visual detection failed: PIR-triggered events could still activate the scheduled alarm logic and Telegram could provide an error message together with a live-stream link for manual verification. This fallback behavior is important because it separates motion-event detection from the success or failure of the camera-based classifier and motivated the controlled follow-up observations of illuminance reported in Section G.

Table 2. Examples from the seven-day motion triggered visual detection test

Detection time	PIR sensor status and detection	YOLOv3 label / score	Alarm status and duration	YOLOv3 detection results sent to Telegram
Morning 07:04	Motion detected by front PIR; resident detected	Human 94% and 65%, bottle 87% and mobile phone 89%	Inactive	
Afternoon 14:27	Motion detected by garage PIR; outside active hours	Human 98% and Human 70%	Active for 3 seconds	

G. Analysis of Lighting Effects (Lux Test)

Because several camera-based detections failed under dim conditions, an additional two-day illuminance-oriented functional test was conducted using a lux-meter application. Eleven trials were documented in total. Detection succeeded in six trials at measured illuminance values of 624, 665, 731, 880, 929, and 950 lux, and each successful event was reported as a human detection in Telegram. Detection

failed in five trials at 0, 0, 0, 102, and 147 lux, after which the system issued an error notification and provided the live-stream fallback. Thus, the overall functional detection success in this small follow-up series was 6/11 (54.5%). This percentage is a trial-level system outcome for the illuminance experiment and is not a YOLOv3 classification-accuracy metric.

- a. Within the observed test points, all six trials conducted at 624-950 lux produced a successful detection, whereas all five trials conducted at 0-147 lux resulted in failure. The source report also documented a failed trial at 102 lux even after flashlight assistance from approximately 1 m, while moving the flashlight closer increased the measured illumination to 929 lux and restored detection.
- b. These observations provide quantitative evidence that image illumination was an important operational constraint for the low-cost ESP32-CAM/OV2640 visual pipeline. However, because the illuminance values were obtained from a small post-hoc functional test rather than a randomized or calibrated imaging experiment, the results should not be interpreted as establishing a universal lux threshold for YOLOv3.
- c. Accordingly, the lighting test is used here to characterize system robustness under the tested deployment conditions, not to estimate detector accuracy. A future controlled study should hold distance, camera settings, object pose, and background constant while varying illuminance systematically and scoring detections against annotated ground truth.

4. DISCUSSION

A. Interpretation of Motion-Triggered Detection

The results should be interpreted as a two-stage event-verification process. The PIR sensor detects changes in infrared radiation associated with movement; therefore, a PIR trigger represents motion, not confirmed occupancy. The camera and YOLOv3 provide the second stage by assigning object labels to the triggered visual stream. A person label can be used as visual evidence of human presence at that event, but the current architecture does not continuously estimate whether a room remains occupied after motion ceases. This distinction avoids conflating motion sensing with occupancy estimation.

B. Comparison with Previous Studies and Research Contribution

Compared with Danang et al. [9], which combined NodeMCU ESP8266, ESP32-CAM, sensors, and Telegram notifications, the present system adds an event-driven MQTT path from PIR sensing to camera

acquisition and YOLOv3-based visual labeling, together with an MQ-2-triggered automatic ventilation response. Raharja et al. [12] focused on web-based appliance control and temperature monitoring, while Musyafa [14] emphasized Telegram-based device control. Maulana et al. [16] included MQ-2 sensing in a smart-home prototype but did not integrate the same motion-triggered visual-detection chain, and the boarding-house system in [17] emphasized electrical, smoke, energy, temperature, and humidity monitoring rather than camera-based object detection. Thus, the contribution of the present study is best characterized as system-level orchestration and field evaluation, not the invention of a new sensor or deep-learning algorithm.

C. Implications, Impact, and Limitations

Practically, the event-driven architecture can make a low-cost smart-home alert more informative by coupling a motion trigger with visual evidence and by linking a gas indication to an immediate actuator response. The dual use of Telegram and Blynk provides complementary channels for event notification and status monitoring, while the retry-and-live-stream fallback allows the user to inspect an event even when YOLOv3 fails to return a usable detection. The illuminance follow-up further showed a clear operational separation in this small dataset: all six trials at 624-950 lux succeeded, whereas all five trials at 0-147 lux failed. This pattern is useful for identifying low-light imaging as a deployment risk, but it is not evidence of a general decision threshold because the test was small and not experimentally controlled. More broadly, the findings are limited by the prototype-scale evaluation. The reported YOLOv3 percentages are confidence scores rather than accuracy metrics; no annotated ground-truth dataset was used; the PIR tests covered only 1-4 m; the MQ-2 evaluation used lighter-gas exposure and a device-specific threshold rather than calibrated gas concentration; camera performance degraded under low illumination; and remote functions depend on Wi-Fi, MQTT, and external network services. These limitations mean that the study demonstrates functional feasibility but does not establish generalized detector accuracy, gas-sensing calibration, or long-term reliability.

D. Future Research

Future work should construct an annotated test set collected from the actual deployment environment and report precision, recall, F1-score, mAP, and a confusion matrix at clearly stated confidence and IoU thresholds. Additional work should compare newer lightweight object detectors, evaluate latency and packet loss under different network conditions, and test privacy- and security-preserving

communication. Hardware improvements should include cameras or illumination support suitable for low-light conditions, longer and more diverse PIR tests, and calibrated MQ-2 experiments using known gas concentrations and controlled environmental conditions. Longer-term field deployment is also needed to quantify uptime, false alarms, energy consumption, maintenance frequency, and the durability of the ventilation actuator.

5. CONCLUSION

Based on the system design, implementation, and testing performed, the following conclusions can be drawn first, a smart home prototype was successfully designed and built to detect motion using PIR sensors and to perform camera-based object detection using an ESP32-CAM and YOLOv3. PIR activation is interpreted as a motion event, while a YOLOv3 person detection provides visual evidence of human presence for that event. The reported YOLOv3 percentages are confidence scores and are not claimed as classification accuracy. Next, the system implements a scheduling mechanism for PIR-triggered alerts to identify motion events outside the residents' usual activity hours. PIR Sensor 1 is activated from 09:00-13:00 and 22:00-06:00, while PIR Sensor 2 is activated from 05:00-15:00 and follows the same night schedule as PIR Sensor 1 (22:00-06:00). On Saturdays and Sundays, the sensors are activated only during the night-to-morning period (22:00-05:00) for the tested monitoring scenario. Last, the system sends alerts through Telegram and Blynk when a motion event or an MQ-2 threshold event is detected, and the gas-response logic activates the alarm and opens the ventilation servo automatically. The seven-day field evaluation showed that visual-detection confidence and success varied with environmental lighting. In the additional 11-trial illuminance test, all six trials at 624-950 lux produced detections, whereas all five trials at 0-147 lux failed. These outcomes reinforce low illumination as an important operational limitation of the ESP32-CAM/OV2640 pipeline while remaining distinct from model-accuracy metrics, which require annotated ground truth and formal evaluation in future work.

BIBLIOGRAPHY

- [1] D. Susilo, C. Sari, and G. W. Krisna, "Sistem Kendali Lampu Pada Smart Home Berbasis IOT (Internet of Things)," *Jurnal ELECTRA: Electrical Engineering Articles*, vol. 2, pp. 23-30, 2021.
- [2] M. Rifaldi and A. Siswanto, "Development of a Voice-Controlled Smart Home Prototype Using NodeMCU and Internet of Things (IoT) Technology," *Digital System and Computing*, vol. 1, pp. 1-5, 2025.
- [3] R. Kumar, S. Shrivastav, A. Chauhan, A. Kumar, and A. Yadav, "IoT based smart home automation system," in *Recent Trends in Intelligent Computing and Communication*, ed: CRC Press, 2025, pp. 302-306.
- [4] A. Vrančić, H. Zadravec, and T. Orehovalčki, "The role of smart homes in providing care for older adults: A systematic literature review from 2010 to 2023," *Smart Cities*, vol. 7, pp. 1502-1550, 2024.
- [5] S. Safie and R. Khairil, "Regulatory, technical, and safety considerations for UAV-based inspection in chemical process plants: A systematic review of current practice and future directions," *Transportation Research Interdisciplinary Perspectives*, vol. 30, p. 101343, 2025.
- [6] E. Ok, "A Detailed Review of the Progress in Home Automation Systems," 2025.
- [7] S. M. Hashemi and P. C. Hung, "Comprehensive Guide to Home Robot Security: From System Analysis to Solution Recommendation," *Journal of Information Assurance & Security*, vol. 20, 2025.
- [8] E. A. Kadir, A. Siswanto, and A. Yulian, "Home Monitoring System Based on Cloud Computing Technology and Object Sensor," in *Proceedings of the Second International Conference on the Future of ASEAN (ICoFA) 2017-Volume 2: Science and Technology*, 2018, pp. 963-972.
- [9] D. Danang, E. Fredyan, and I. S. Suasana, "Prototype Alat Keamanan Rumah Internet Of Things (Iot) Berbasis Nodemcu Esp8266 Dengan Esp32 Cam Dan Kombinasi Sensor Menggunakan Telegram," *Jurnal Universal Technic*, vol. 1, pp. 1-16, 2022.
- [10] N. Salsabila, A. Siswanto, and L. Bayuaji, "Design of a smart home door security system with face detection and smart bell using esp32-cam," in *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, 2025, pp. 124-129.
- [11] A. A. Adenowo and B. A. Adenowo, "Software engineering methodologies: a review of the waterfall model and object-oriented approach," *International Journal of Scientific & Engineering Research*, vol. 4, pp. 427-434, 2013.
- [12] R. U. M. Raharja, A. Pudoli, and D. Kusumaningsih, "Prototype smart home berbasis IoT dengan nodemcu esp8266, motor servo dan sensor suhu dht11 berbasis web," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 5, pp. 265-274, 2022.

- [13] K.-H. Grote, C. Beyer, C. Birke, and A. Tenbusch, "Advances in Integrating Rapid Prototyping and Three-Dimensional Digitizing into Design Methodology and Product Development Processes," *Journal of Integrated Design and Process Science*, 2024.
- [14] A. Musyafa, "Yulianti," "Perancangan Smart Home Dengan Konsep Internet Of Things (Iot) Menggunakan NodeMCU ESP8266 Via Telegram Bot," *Log*, J. Ilmu Komput. dan Pendidik, vol. 1, pp. 1470-1477, 2023.
- [15] D. Amos, A. Lesea, and R. Richter, *FPGA-based prototyping methodology manual: Best practices in design-for-prototyping*: Happy About, 2011.
- [16] M. Maulana, A. Noor, and A. Supriyanto, "PURWARUPA SISTEM RUMAH PINTAR BERBASIS INTERNET OF THINGS," *Politeknik Negeri Tanah Laut*, 2022.
- [17] G. P. Putra, Y. Divayana, and P. Rahardjo, "Rancang Bangun Sistem Smart Home Pada Rumah Kos Berbasis Internet of Things," *Jurnal SPEKTRUM*, vol. 9, 2022.
- [18] A. A. Wahid, "Analisis metode waterfall untuk pengembangan sistem informasi," *J. Ilmu-ilmu Inform. dan Manaj. STMIK*, no. November, vol. 1, pp. 1-5, 2020.
- [19] R. Davila-Campos, M. Mora, P. Y. Reyes-Delgado, S. Galván-Cruz, and G. C. Lopez-Torres, "Design and Evaluation of AgileBPM SDLC—an Agile SDLC for Business Process Management Systems," *IEEE Access*, 2025.
- [20] A. A. Aziiza and A. N. Fadhillah, "Analisis Metode Identifikasi dan Verifikasi Kebutuhan Non Fungsional," *Applied Technology and Computing Science Journal*, vol. 3, pp. 13-21, 2020.
- [21] A. Shabbir, A. Deraman, M. N. B. Hassan, K. U. Sarker, and S. Kamal, "Prioritizing Non-Functional Requirements and Influencing Factors for API Quality Framework: An Industry Approach," *International Journal of Advanced Computer Science & Applications*, vol. 16, 2025.
- [22] M. F. Ghafur, A. Wahyudin, and A. Anisyah, "Sistem Pemindaian Boarding Pass Mobile dan Dashboard Monitoring Terintegrasi untuk Optimasi Operasional Bandara," *Jurnal Komputer Teknologi Informasi Sistem Informasi (JUKTISI)*, vol. 4, pp. 929-941, 2025.
- [23] S. Supiyandi, M. Zen, C. Rizal, and M. Eka, "Perancangan Sistem Informasi Desa Tomuan Holbung Menggunakan Metode Waterfall," *JURIKOM (Jurnal Ris. Komputer)*, vol. 9, p. 274, 2022.
- [24] I. E. Lauritto, P. M. Madrid, A. V. Gabirondo Sigura, L. D. Fernández Gimenez, J. T. Molas Giménez, and J. O. Angelini, "Model-Based Systems Engineering Applied to Design and Development of Biomaterials Devices," in *Congreso Argentino de Bioingeniería*, 2025, pp. 40-54.
- [25] H. Ö. Özsoy, "Using Flowcharts and State Diagrams as Tools For Industrial Design of Smart Products," *Sanat ve Tasarım Dergisi*, pp. 205-233, 2025.
- [26] R. I. Rosihan, "Analisis Sistem Reliability Pada Mesin Extruder 90 Proses Extrusion Pada Pt. X Dengan Pendekatan Reliability Block Diagram," *Jurnal Rekavasi*, vol. 7, pp. 59-67, 2019.
- [27] D. Chen, Y. Feng, X. Li, M. Qu, P. Luo, and L. Meng, "Interpreting core forms of urban morphology linked to urban functions with explainable graph neural network," *arXiv preprint arXiv:2502.16210*, 2025.
- [28] M. Rahmadan and C. E. Gunawan, "Perancangan data flow diagram aplikasi tabungan sampah PT Pusri Palembang," in *Prosiding Seminar Nasional Mini Riset Mahasiswa*, 2024.
- [29] M. A. Taufan, D. S. Rusdianto, and M. T. Ananta, "Pengembangan Sistem Otomatisasi Use Case Diagram berdasarkan Skenario Sistem menggunakan Metode POS Tagger Stanford NLP," *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, vol. 6, pp. 3733-3740, 2022.
- [30] R. Destriana, S. M. Husain, N. Handayani, and A. T. P. Siswanto, *Diagram UML Dalam Membuat Aplikasi Android Firebase" Studi Kasus Aplikasi Bank Sampah*: Deepublish, 2022.
- [31] A. Siswanto and F. A. Alfyandi, "Door Lock System Based on Android Fingerprint Application Using Keypad and Telegram Notifications," in *International conference on smart computing and cyber security: strategic foresight, security challenges and innovation*, 2023, pp. 631-643.