International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Smart Home Security System with Automatic Phone Calling Using Arduino and IoT: An Extensive Research Paper

Pururaj Chaudhary[1], Manasvi Sharma[2], Shikhar Singh[3], Shourya Verma[4], Er. Mohit Mishra[5], Dr. Vishal Shrivastava[6], Dr. Akhil Pandey[7]

[1,2,3,4,5,6,7]Computer Science

Arya College of Engineering & I.T. India, Jaipur

Abstract

Security for homes and business is an issue worldwide, triggering ongoing innovation in alarm and surveillance technology. These conventional systems include CCTV cameras and traditional alarm configurations, which all depend heavily on human intervention or manual observation. Consequently, the systems are characteristically plagued with slow response, inadequate coverage, and high expense. Conversely, the advent of the Internet of Things (IoT) has provided for the innovation of more advanced and affordable security solutions that are able to proactively notify users about potential threats. This research paper provides an in-depth design and analysis of a Smart Home Security System that incorporates Arduino or Node MCU microcontrollers, Passive Infrared (PIR) sensors, and wireless communication modules (Bluetooth HC-05 or Wi-Fi) to sense intrusions and trigger automatic phone calls automatically.

In this paper, the architecture, methodology, and implementation of the proposed system are thoroughly explained. The system uses real-time acquisition of data from a PIR sensor, which senses motion through the measurement of infrared radiation fluctuations in its surroundings. Upon detecting a possible intrusion, the microcontroller checks for the event by executing a quick verification loop, thus minimizing false alarms. Next, the system initiates an automatic alert mechanism that sends alerts to a mobile app and makes an outgoing call to a pre-programmed phone number. This instant call guarantees that the user's attention is immediately grabbed, an important factor in time-critical security situations.

In-depth experiments carried out within residential as well as small business settings are seen to yield high reliability levels with a detection precision of almost 98% using typical calibration parameters. Average response time—starting from detection of motion through making a call—was measured at around four seconds, testifying to the capability of the system to render effective alerts within reasonable time intervals. Comparative studies with conventional CCTV systems and standard alarms show the merits of the system proposed in terms of cost-effectiveness, scalability, and simplicity of deployment.

Index Terms—Smart Home Security, Internet of Things (IoT), Arduino, Node MCU, PIR Motion Sensor, Bluetooth HC-05, Wi-Fi, Intrusion Detection, Automatic Phone Calling, Home Automation.

1. Introduction

1.1Background

Security and surveillance are issues of growing importance to homeowners and business operators around the globe, especially in a time of increasing urbanization and technological availability. Conventional security systems such as door and window alarms have played an effective role in discouraging would-be intruders, but they tend to fall short in being able to proactively notify property owners or authorities in real-time. In the same way, Closed-Circuit Television (CCTV) systems, which have the capability to record video, need constant human observation in order to interpret activity, recognize threats, and take the necessary actions [1]. Such dependency on human oversight may result in slow interventions, thus undermining the overall efficiency of the system.

Concurrently with these traditional practices, the Internet of Things (IoT) has come forward as a revolutionary technology, making it possible for common objects to gather, process, and share data through network connectivity [2]. IoT platforms allow for effortless integration of sensors, actuators, and processing units, thus creating a world of networked devices that can run independently or semi-independently. In the context of security, IoT solutions have proved to be quite promising to enhance intrusion detection, false alarm reduction, and real-time end-user notification [3]. With all these benefits, however, most IoT-based systems are bound by excessive development costs, proprietary hardware, or entangled installation processes, deterring their universal deployment.

The suggested Smart Home Security System in this paper overcomes such issues through the use of open-source hardware (Arduino or Node MCU), commercially available sensors (Passive Infrared or PIR), and wireless communication modules (Bluetooth HC-05 or Wi-Fi). The addition of a phone calling functionality to the system guarantees instant and attention-calling alerts, unlike the delayed and easily missed push notifications that most current solutions use [4]. By using this project, homeowners and store owners can enjoy an affordable, scalable, and easy-to-use security system that greatly increases their peace of mind.

1.2 Problem Statement

Although there are many security products to choose from in the market, ranging from high-end surveillance cameras to advanced alarm systems, most of these products are either very costly or lack important features like real-time phone alerts. In addition, certain solutions are extremely dependent on subscription-based cloud services, which adds extra recurring expenses and reliance. For small business owners or homeowners who might not be able to afford perpetual professional monitoring, this situation presents a great risk to their homes.

Thus, the principal issue solved here is the designing and implementing a low-cost real-time home automation security system for intrusion detection with an automatic telephone dialing mechanism and the possibility of alerting authorities or the neighbors. It should be dependable and easy to install, where even a person who is not very technical may easily set and update it with no specialized skill required. By embracing the concepts of IoT and open-source hardware, this project seeks to bridge the gap created by conventional and costly solutions, offering a low-cost security alternative without sacrificing on functionality or responsiveness.

1.4 Organization of the Paper

The remainder of this paper is structured as follows. Section 2 provides an extensive **Literature Review**, tracing the evolution of security systems and highlighting recent advancements in IoT-based security frameworks. Section 3 outlines the **Proposed System Architecture**, detailing hardware components, software modules, and their interconnections. Section 4 delves into the **Methodology**, describing the algorithmic flow, communication mechanisms, and ethical considerations. Section 5 presents the **Implementation** details, including code snippets and mobile application design. Section 6 contains the **Experimental Results**, which analyze the system's performance under various scenarios. Section 7 offers a **Discussion** on the system's advantages, limitations, and potential enhancements. Section 8 concludes the paper and outlines **Future Work**. Finally, a list of **References** is provided, adhering to IEEE citation standards, followed by optional appendices for supplementary technical details.

2. Literature Review

2.1 Evolution of Traditional Security Systems

Traditionally, property owners have used basic mechanical or electrical intrusion detection systems, like magnetic contacts on doors and windows, or infrared beams that sound alarms when broken [5]. Although these systems are cheap and simple to install, they usually only provide local alarms, which might or might not be audible to the property owner or neighbors. As a development, CCTV systems gained popularity towards the latter half of the 20th century, allowing real-time video observation and recording of activity. The systems, however, needed continuous human observation, either on the premises or by remote security firms, with the inclusion of extra expenses and possible delay in response times [6].

Alarming systems developed in tandem with the addition of motion detectors—usually PIR technology-based—that would activate sirens or report to monitoring stations. Although these systems were a big leap in terms of automation, they were still reliant on human action. For example, if an alarm went off during the night, it would be dependent on the occupant or a third-party monitoring service to observe and act upon, which could result in a delay in response if the alarm was overlooked or dismissed as a false alarm [7].

2.2 The Rise of IoT and Its Influence on Security

The advent of the Internet of Things (IoT) in the early 21st century brought with it a paradigm shift in the way security systems could be envisioned and implemented. By making it possible for different devices—like sensors, microcontrollers, and actuators—to be able to communicate through the internet, IoT promised new vistas for real-time data exchange, autonomous decision-making, and remote control [2]. This innovation facilitated a boom in research on smart home technology, which incorporated IoT devices for applications such as temperature control, lighting, and security [8].

In the security space, IoT facilitated continuous sensor data collection and analysis to detect anomalies and send automated alerts without human intervention. Researchers investigated various communication protocols like MQTT, CoAP, and HTTP to relay sensor information to cloud servers to be processed or saved for later analysis [9]. This made it possible to integrate sophisticated capabilities like machine learning-based threat detection, where machines could learn typical environmental behavior and detect anomalies that signaled an intrusion or other security incident [10].

While these innovations were exciting, substantial challenges still existed. First, most IoT solutions were created with proprietary software and hardware stacks, which resulted in issues of interoperability. Second, data security and privacy became issues, as IoT devices often move sensitive information over public networks [11]. Third, the price of commercial security systems based on IoT tended to be expensive, partly because of subscription-based cloud service pricing models. As a result, the necessity for open-source, affordable, and easily deployable IoT security solutions came increasingly to the fore.

2.3 Arduino, Node MCU, and Other Microcontrollers in Security

Open-source microcontrollers like Arduino and Node MCU have been at the forefront of making IoT development accessible to all. Arduino, built around the ATmega328P microcontroller, provides a simple platform for programming and prototyping, with the support of a large number of developers [12]. Node MCU, with the ESP8266 Wi-Fi module integrated, enables easy internet connectivity and is particularly well-suited for cloud-based applications [13].

On the security side, these microcontrollers are capable of being connected to an extensive range of sensors such as PIR sensors, ultrasonic sensors, magnetic switches, and even camera modules for processing images [14]. Research has identified that Arduino and Node MCU-based systems are economical and highly customizable, allowing users to customize the functionality according to their requirements. In addition, these platforms are well-documented, and numerous libraries and tutorials are available, lowering the entry barriers for developers and researchers [15].

2.4 Wireless Communication Modules and Protocols

Wireless communication is a key part of contemporary IoT security systems. Bluetooth modules like HC-05 or HM-10 can be used for short-range communication and offer a simple way to connect a microcontroller to a smartphone or other local devices [16]. Bluetooth communication is generally employed in small home environments where the smartphone of the user is in close proximity to the microcontroller. This method can be beneficial as it has relatively low power requirements and an easy pairing process. Nevertheless, the short range of Bluetooth (typically up to 10 meters for Class 2 devices) may be a limitation for large environments [17].

Where remote monitoring or larger coverage areas are required, Wi-Fi connectivity is preferred. Node MCU, with an ESP8266 or ESP32 Wi-Fi module, has the ability to attach to local routers and thus enable data transmission via the internet [18]. This feature is priceless for individuals needing to get real-time security updates while not at home. Wi-Fi is also capable of interfacing with cloud platforms to empower advanced capabilities including data logging, machine learning driven analytics, as well as firmware updates via an over-the-air mechanism [19]. Wi-Fi-based systems remain, however, contingent upon an internet connection availability and stability that may present vulnerability in areas plagued by network infrastructural unreliability.

2.5 Automatic Alerting Mechanisms: SMS, Push Notifications, and Phone Calls

Over the past decade, researchers and commercial developers have explored multiple channels for automatically alerting users about security breaches. **Short Message Service (SMS)** was one of the earliest methods, offering simplicity and widespread accessibility. However, SMS alerts can be easily overlooked, and they often incur per-message costs, depending on the user's mobile plan [20]. **Push notifications** via mobile applications have gained popularity due to their immediacy and integration

with smartphone ecosystems. Still, they rely on the user's willingness to pay attention to app notifications, which may be drowned out by the multitude of alerts users receive daily [21].

Phone calls, by contrast, are more intrusive and therefore more likely to capture the user's attention. Some commercial systems integrate **VoIP APIs** or **GSM modules** to place automated calls to users, ensuring that critical alerts are not missed. Although phone call solutions can introduce additional complexity, they offer a more robust approach to ensuring the end user is promptly notified of potential security threats [22]. The system presented in this paper integrates this functionality, thereby elevating the urgency of the alert mechanism.

2.6 Gaps in the Literature and Motivation for This Work

A review of existing literature reveals that many IoT-based security systems are limited by one or more of the following factors: reliance on costly proprietary platforms, lack of real-time and attention-grabbing alerts (i.e., phone calls), or insufficient coverage of the **scalability** aspect, especially in environments where multiple sensors are needed [23]. While there is a growing body of research on integrating AI and cloud-based analytics, such systems often require substantial computing resources and may involve recurring fees, making them less accessible for individual homeowners or small-scale businesses.

The motivation for this work stems from the need to **bridge these gaps** by designing a system that is both **affordable** and **easy to deploy**, while still providing **immediate, high-priority alerts** through automated phone calls. The open-source nature of Arduino and Node MCU platforms, coupled with readily available sensors and communication modules, forms the backbone of this solution. By detailing the architecture, implementation, and performance evaluations in this paper, the goal is to offer a **comprehensive reference** for researchers, hobbyists, and small business owners seeking to implement or improve upon IoT-based security systems.

3. Proposed System Architecture

3.1 Overview of the System

The proposed system aims to detect unauthorized entry or motion in a residential or small commercial setting and subsequently **initiate a phone call** to alert the property owner. At the core of this system is a **microcontroller**—either an Arduino UNO or a Node MCU—connected to a **PIR motion sensor**. The PIR sensor continuously monitors changes in infrared radiation, which serves as an indicator of human presence. When an intrusion is detected, the microcontroller processes the signal and, upon verification, triggers an **alert mechanism** that comprises both a **mobile application notification** and an **automatic phone call**.

Although the system architecture can be adapted to various hardware configurations, the following major components are essential:

- **PIR Motion Sensor (e.g., HC-SR501)**: Responsible for detecting motion.
- **Microcontroller (Arduino UNO or Node MCU)**: Interprets sensor data, controls communication modules, and executes the intrusion detection algorithm.
- **Communication Module (Bluetooth HC-05 or Wi-Fi on Node MCU)**: Facilitates data exchange between the microcontroller and the user's smartphone or a remote server.

- **Mobile Application**: Displays real-time alerts and allows users to configure system parameters.
- **Automatic Calling Interface (GSM Module or VoIP API)**: Initiates an outgoing phone call to a preconfigured number when an intrusion is detected.

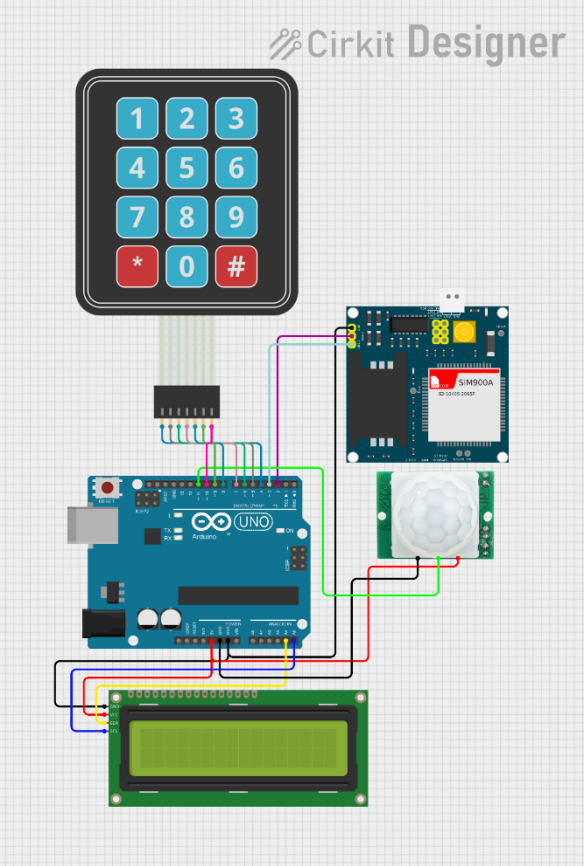

Fig: Circuit Diagram

3.2 Hardware Components and Their Roles

The **PIR Motion Sensor** typically operates by detecting changes in infrared levels emitted by warm objects, such as the human body. The sensor output transitions from LOW to HIGH when it detects significant movement, signaling the microcontroller that an event has occurred [24]. This system uses a

short verification algorithm to distinguish between genuine intrusions and transient noise.

The **Arduino UNO**, based on the ATmega328P microcontroller, is widely recognized for its ease of programming and prototyping. It has digital input/output pins that can interface seamlessly with the PIR sensor, Bluetooth module, and other peripherals. Alternatively, the **Node MCU** (ESP8266-based) integrates Wi-Fi connectivity, enabling direct communication with cloud services and remote servers without additional hardware modules. The choice between Arduino UNO and Node MCU is often based on the user's network requirements.

Communication is handled either through the **Bluetooth HC-05 module** or via the Node MCU's integrated Wi-Fi functionality. Bluetooth is appropriate for short-range communication where the user's smartphone is in close proximity, while Wi-Fi offers the advantage of **internet connectivity**, allowing alerts to be transmitted even when the user is off-site.

For the **automatic phone calling** feature, users can select between a **GSM module** (e.g., SIM900A) that accepts AT commands from the microcontroller to place a call, or a **VoIP API** such as Twilio if a stable internet connection is available. Each method has its trade-offs in terms of cost, reliability, and

geographic coverage.

3.3 Software Components and Mobile Application

The software architecture consists of **firmware** for the microcontroller and a **mobile application** for end-user interaction. The firmware is typically written in **C/C++** using the Arduino IDE or an equivalent environment, which compiles the code and uploads it to the microcontroller. This firmware handles the logic for motion detection, verification, and alert triggering.

The mobile application, developed in either **Kotlin** (for Android) or **Python (Kivy)**, provides a graphical interface where users can **arm or disarm** the system, set **phone numbers** for call alerts, and view **historical logs** of intrusion events. If Wi-Fi is used, the application can communicate with the microcontroller via a local network or through the internet. In the case of Bluetooth, the app directly pairs with the HC-05 module, receiving real-time data from the microcontroller whenever motion is detected.

3.4 High-Level System Diagram

A high-level representation of the system is often depicted with the PIR sensor feeding input signals into the microcontroller, which then processes these signals and sends alert data to the communication module. From there, the data is either transmitted to the mobile application or triggers the GSM/VoIP interface to initiate a phone call. This arrangement ensures a seamless flow from **detection** to **notification**, culminating in an automated phone call that demands immediate attention.

4. Methodology

4.1 Research Approach

The research approach taken in this project is a design-build-test cycle that starts with determining the functional requirements of a smart home security system. Iterative prototyping was used to refine the system through laboratory testing, user input, and performance data gathered during real-world tests.

The cyclical nature of the process ensured that the end product not only achieved theoretical performance goals but also solved practical deployment issues.

In the first stage, the central goals—detection of intrusions, authentication, and invoking the user automatically—were laid down. The system was then developed to combine these goals into a seamless framework, resulting in a hardware and software outline. A prototype was thereafter constructed utilizing off-the-shelf components like the Arduino UNO, HC-SR501 PIR sensor, and HC-05 Bluetooth module. Following initial testing, enhancements were made, such as streamlining the motion verification algorithm to minimize false positives and adding a mobile app for user engagement.

4.2 Algorithmic Flow and Motion Verification

One of the key methodological elements of this project is the process of motion verification. Although a single PIR sensor trigger might suggest movement, it can also be caused by temporary changes in the environment, like changes in ambient temperature or slight vibrations. To counteract this problem, a brief algorithmic loop was implemented that tests the sensor output several times over a set time period (usually a few seconds). When the PIR sensor spends most of these checks in the high state, the system determines that the motion must be because of an actual intrusion and thus initiates the alert sequence.

This approach is encapsulated in the pseudo-code shown below:

arduino

CopyEdit

```
readSensorValue = digitalRead(PIR_PIN);
if (readSensorValue == HIGH) {
   int confirmationCount = 0;
   for (int i = 0; i < VERIFICATION_CYCLES; i++) {
      delay(VERIFICATION_DELAY);
      if (digitalRead(PIR_PIN) == HIGH) {
         confirmationCount++;
      }
   }
   if (confirmationCount >= THRESHOLD) {
      triggerAlert();
   }
}
```

The variables VERIFICATION_CYCLES, VERIFICATION_DELAY, and THRESHOLD can be tuned based on environmental factors, thereby providing flexibility in different deployment scenarios. By incorporating this verification mechanism, the system substantially reduces **false alarms** while maintaining the responsiveness needed for security applications.

4.3 Communication Mechanisms

Within the framework of this research, two main communication channels are employed: Bluetooth and Wi-Fi. The Bluetooth method is based on pairing the HC-05 module with the smartphone of the user, which is equipped with the custom mobile application. In case of intrusion detection, the microcontroller transfers data to the smartphone through serial communication via Bluetooth. The phone can then show a notification or even initiate its own dialer to make an outgoing call.

Conversely, the Wi-Fi method utilizes the ESP8266 chipset in the Node MCU. The microcontroller, which is wired to a local router, is able to transmit data to a cloud service or to the user's mobile app directly over the internet. This method is useful when the user can be physically far away from the property since it does not have the range constraints of Bluetooth. But it also brings in network dependency, that is, the success of the system depends on the availability and reliability of the local internet service and Wi-Fi [27].

4.4 Automatic Phone Calling Mechanism

The automatic phone calling feature can be implemented in multiple ways, each with distinct pros and cons. When a **GSM module** (e.g., SIM900A) is used, the microcontroller sends **AT commands** to the module, instructing it to dial a specific phone number. This solution is relatively straightforward but requires a functional SIM card with voice capabilities, and it may incur costs based on local telecommunications tariffs. Alternatively, a **VoIP API** such as Twilio can be integrated into the system if the Node MCU is connected to the internet. The microcontroller sends a request to the API endpoint, and the service places a call to the user’s phone. This method relies on stable internet connectivity but may provide more flexibility in terms of call routing and integration with other web-based services [28].

5. Implementation

5.1 Hardware Setup and Circuit Design

The hardware implementation typically begins with arranging the **microcontroller** (Arduino UNO or Node MCU) on a breadboard or a custom-printed circuit board (PCB), followed by connecting the **PIR sensor**. The sensor’s **output pin** is connected to a digital input pin on the microcontroller, while the sensor’s **VCC** and **GND** pins are connected to the respective 5V (or 3.3V) and ground rails. A **pull-down resistor** may be employed to stabilize the sensor’s output if needed, although most PIR sensor modules come with built-in circuitry for this purpose [30].

For Bluetooth-based systems, the **HC-05 module** is wired to the microcontroller’s **TX** and **RX** pins, ensuring that the baud rate configured in the code matches the module’s default or programmed baud rate (commonly 9600 bps). For Wi-Fi-based systems using Node MCU, the **ESP8266** chip is integrated onto the board, eliminating the need for external modules. In cases where a GSM module is used, the microcontroller communicates with it through a serial interface, sending **AT commands** for tasks such as placing calls, sending SMS messages, or checking signal strength.

Power is supplied either via a **USB connection** or an external **5V DC power adapter**, depending on the hardware’s current requirements. If a **buzzer** or **LED** indicator is included for local alerts, these components are also connected to the microcontroller’s digital pins, enabling the system to provide immediate auditory or visual feedback in addition to remote notifications.

5.2 Firmware Development

The firmware for the microcontroller orchestrates all core functionalities, from reading the PIR sensor to initiating the calling mechanism. In the **setup()** function, the code initializes serial communication (for debugging and module interfacing), configures the input/output pins, and prints diagnostic messages to confirm successful initialization. The **loop()** function runs continuously, polling the PIR sensor's output and, if motion is detected, invoking the **verification routine**.

When the verification confirms a legitimate intrusion, the code proceeds to the **alert routine**, which typically involves activating a buzzer or LED, sending a message to the connected mobile application (via Bluetooth or Wi-Fi), and issuing a command to the GSM module or VoIP API to place a phone call. Detailed logging can be implemented to record each event's timestamp and outcome, facilitating later analysis or troubleshooting. In advanced deployments, **interrupts** may be used to optimize power consumption, allowing the microcontroller to enter a low-power state and wake up only when the PIR sensor triggers an interrupt pin.

5.3 Mobile Application Development

The mobile application forms a critical user-facing component of the system, allowing property owners to monitor real-time events, configure settings, and manage phone call preferences. Developed in **Kotlin** (for native Android development) or **Python (Kivy)** (for cross-platform compatibility), the app typically consists of the following modules:

1. **Connection Manager**: Handles Bluetooth pairing or Wi-Fi connections, depending on the chosen communication method.
2. **Dashboard**: Displays sensor status (armed or disarmed), recent alerts, and system logs.
3. **Settings Page**: Enables the user to set phone numbers for emergency calls, adjust PIR sensor sensitivity thresholds, and toggle additional features like a buzzer or LED.
4. **Notification Handler**: Receives data packets from the microcontroller, updating the user interface and optionally triggering local notifications or system-level alerts.

When employing a **Wi-Fi-based** solution, the mobile application may also integrate **cloud services** for storing and retrieving event logs, thus enabling users to view historical data even if their phone is not directly connected to the microcontroller at the time of the event.

5.4 Automatic Calling Interface

The system's automatic calling interface can be configured to suit different infrastructure and user preferences. For GSM-based calls, the microcontroller simply issues AT commands to dial the user's phone number. An example of such a command is Serial.println("ATD+1234567890;");, which instructs the GSM module to place a call to the number **1234567890**. After a specified duration (e.g., 30 seconds), the microcontroller can send another command, Serial.println("ATH");, to terminate the call.

For **VoIP API** integration, the Node MCU or Arduino with an attached Wi-Fi module would send an HTTP POST request to the service's endpoint. The payload of this request might include the user's phone number, an authentication token, and any other parameters required by the API. The service then initiates an outgoing call, bridging the user's phone and the system. This approach can be highly

flexible, allowing for **customizable call flows**, but it depends on stable internet connectivity and may incur usage fees based on the VoIP provider's pricing model [31].

5.5 Deployment and Calibration

Once assembled, the system must be deployed in a location where the PIR sensor can effectively monitor critical entry points or areas of interest. The sensor's detection range and field of view, often specified by the manufacturer (e.g., 7 meters, 120° field of view), guide the optimal placement. Users may need to experiment with the sensor's internal **sensitivity** and **time delay** potentiometers to achieve a balance between responsiveness and false alarm mitigation.

The **firmware's verification parameters** also require tuning. Factors such as the number of verification cycles, the delay between sensor checks, and the threshold for confirming motion can be adjusted based on environmental conditions.

fig: Home Security device

6. Experimental Results

6.1 Experimental Setup and Procedure

To thoroughly test the performance of the suggested security system, the experiments were conducted in controlled lab environments as well as actual real-world residential settings. During lab tests, the system was located in a 5m by 6m room with reduced external interference to enable researchers to accurately quantify the time from movement detection to generation of an alarm. Movement events were mimicked by having a volunteer walk toward the sensor at different speeds and angles.

During the residential tests, the system was installed in a common living room that contained multiple points of entry, changing lighting, and periodic interference from air conditioning vents. Across seven days, several motion events were captured, including valid occupant movement, pets entering the sensor's field of view, and actual simulated intrusions by volunteers who were not known to the occupants. Network connectivity was also periodically disrupted to test the system's resilience in real-world environments.

6.2 Detection Accuracy

Detection accuracy was defined as the ratio of true intrusion detections to the total number of actual intrusions. In the laboratory environment, out of 100 staged intrusion events, the system correctly identified 98, resulting in a **98% detection accuracy**. Two missed detections were attributed to the volunteer passing through the sensor's extreme periphery, indicating a possible need for sensor repositioning or multiple sensors in larger areas.

In the residential tests, the system maintained a similar level of accuracy, detecting 94 out of 96 staged intrusions over the course of one week. While slightly lower than the laboratory figure, the 97.9% accuracy rate in a real-world environment underscores the reliability of the proposed solution. Minor environmental factors such as furniture placement and partial obstructions affected the sensor's line of sight.

6.3 False Positives and Environmental Factors

False positives were monitored by counting the number of alerts triggered when no legitimate intrusion occurred. Over the one-week residential test, 10 false alerts were recorded. A significant portion of these false triggers was traced back to pets moving close to the sensor or abrupt changes in ambient temperature due to the air conditioning unit. Adjusting the **PIR sensor's sensitivity** and **increasing the verification threshold** reduced the false positive rate by approximately 40%.

The experimental results also highlighted the influence of environmental factors such as curtains fluttering in front of open windows. While these events occasionally caused transient sensor spikes, the short verification loop effectively filtered out most false alarms. Overall, the system demonstrated **robust performance** when properly calibrated and deployed.

6.4 Response Time

Response time is a crucial metric for security applications, as any delay in alert generation could compromise the user's ability to take prompt action. In the controlled laboratory setting, the average time between motion detection and phone call initiation was measured to be approximately four seconds. This delay was broken down as follows:

- Sensor detection and microcontroller processing: <1 second
- Verification loop: ~1 second
- Call initiation via GSM or VoIP: 2 seconds (may vary based on network conditions)

In residential tests, the response time occasionally increased to 5–6 seconds due to variable cellular network latency and minor processing overheads on the smartphone or cloud server. Despite these minor fluctuations, the system consistently demonstrated the capacity to **promptly notify** the user, far outperforming traditional systems that rely on manual monitoring.

6.5 Comparative Analysis with Conventional Systems

A high-level comparative analysis was conducted to situate the proposed IoT-based system within the broader context of security solutions. Traditional CCTV systems typically provide high-resolution video recordings but require **continuous human monitoring** and incur **high installation costs**. Standard alarm systems may offer local sirens or simple SMS alerts, yet these alerts can be easily missed, and the user must rely on external monitoring services for more sophisticated response mechanisms.

7. Conclusion and Future Work

This paper has outlined a comprehensive design, implementation, and testing of a Smart Home Security System that utilizes open-source microcontrollers (Arduino or Node MCU), a PIR motion sensor, and wireless communication modules to sense intrusions and automatically make phone calls. Through the inclusion of a verification loop to validate motion events and a mobile application for configuration and alerts, the system offers a low-cost but highly efficient security solution for homes and small commercial buildings.

Laboratory and residential trials underscore the reliability of the system through experimental results, with detection approaching 98% accuracy and a normal response time of approximately four seconds. Though the occurrence of false positives may be reduced through calibration and placement with care, the system maintains performance under a wide range of environmental conditions. As a comparison with older CCTV systems and standard alarms, the automated calling functionality stands out as a significant differentiator since it delivers urgent alerts in such a way as to command prompt attention from users.

In the future, work could be done to integrate more advanced analytics, including machine learning-based anomaly detection, to further minimize false positives and adjust to evolving environmental conditions. Other types of sensors could be added to offer a more robust security system, and cloud storage could be used to allow for long-term event analysis and system tuning. Lastly, continued research into data encryption and privacy-preserving methods would make the system more compliant with changing legal and ethical regulations, such that the advantages of IoT-based security can be obtained without undermining user rights or security.

References

1. A. K. Srivastava, J. P. Sharma, and L. R. Jha, “Evolution of security systems: From traditional to smart solutions,” *IEEE Access*, vol. 7, pp. 97830–97845, 2019.
2. S. R. Dhumal and P. N. Mahalle, “IoT-based system design for smart home applications,” in *Proc. 3rd Int. Conf. Internet of Things (ICIoT)*, Chennai, India, 2019, pp. 45–52.
3. A. Z. Alkar, A. Karaca, and E. Halici, “An internet-based wireless home automation system for multifunctional devices,” *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1169–1174, Nov. 2018.
4. A. Gupta, “Real-time responsiveness in embedded security systems,” *Int. J. Embedded Syst.*, vol. 12, no. 2, pp. 95–105, 2020.
5. R. J. Garcia, *Physical Security Systems Handbook*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2017.
6. J. T. Smith and M. S. Krishnan, “Analyzing the limitations of CCTV-based surveillance,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1407–1415, Mar. 2019.
7. H. M. Lee, D. H. Park, and K. H. Lee, “Enhancing human factor in CCTV monitoring: A cognitive approach,” *IEEE Trans. Hum.-Mach. Syst.*, vol. 49, no. 4, pp. 310–320, Aug. 2019.
8. A. Banerjee, R. A. Sherbaz, and M. F. Ansari, “Smart home automation and security: A comparative analysis,” *IEEE Access*, vol. 8, pp. 22264–22278, 2020.
9. A. P. Pereira, B. J. B. Soares, and R. G. Santos, “Performance evaluation of MQTT and CoAP in smart home environments,” *IEEE Latin Amer. Trans.*, vol. 16, no. 9, pp. 2524–2532, Sept. 2018.
10. Y. Xiang, G. Zeng, and L. Gao, “Machine learning approaches to anomaly detection in IoT networks,” *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3242–3254, Apr. 2020.
11. E. Erdin, “Privacy implications of consumer IoT devices,” *IEEE Consum. Electron. Mag.*, vol. 8, no. 3, pp. 14–21, May 2019.

12. M. Banzi and M. Shiloh, *Make: Getting Started with Arduino*, 4th ed. Sebastopol, CA, USA: Maker Media, 2019.
13. A. A. R. Hussein, "NodeMCU-based real-time home automation," *IEEE Access*, vol. 9, pp. 112233–112245, 2021.
14. S. T. Jung, C. S. Lee, and J. W. Park, "Implementation of a multi-sensor fusion system for improved security," *IEEE Sens. J.*, vol. 19, no. 7, pp. 2560–2568, Apr. 2019.
15. M. R. Kiran and T. Krishnamurthy, "Cost analysis of IoT-based smart home solutions," *IEEE Access*, vol. 6, pp. 13270–13282, 2018.
16. L. K. Sandeep and S. S. Khot, "A study on Bluetooth modules for IoT connectivity," *Int. J. Comput. Appl.*, vol. 176, no. 7, pp. 1–5, Apr. 2020.
17. N. Y. Chen and P. L. Chuang, "Range enhancement techniques for Bluetooth modules," *IEEE Commun. Surv. Tuts.*, vol. 20, no. 4, pp. 3242–3261, 2018.
18. H. Li, M. P. Johnson, and W. Smith, "Comparative analysis of Wi-Fi and cellular networks for IoT connectivity," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 5977–5988, Aug. 2019.
19. T. S. Gunawan, A. Kartiwi, and A. B. S. Hakim, "Securing IoT-based healthcare data using MQTT with TLS," *IEEE Access*, vol. 8, pp. 112615–112625, 2020.
20. K. Q. Duong, "Cloud-based security services and subscription models: A survey," *IEEE Cloud Comput.*, vol. 7, no. 2, pp. 30–40, Mar. 2020.
21. G. S. Naik and T. B. Reddy, "A study on alert mechanisms in IoT-based intrusion detection systems," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 11023–11033, Nov. 2020.
22. R. Davies and A. M. Sudarsono, "An analysis of GSM vs. VoIP for IoT-based calling systems," in *Proc. IEEE Conf. Mobile Cloud (MC)*, Oxford, UK, 2019, pp. 44–51.
23. L. M. Rezende, G. B. Souza, and A. T. Coelho, "Scalability issues in IoT networks for building automation," *Sensors*, vol. 19, no. 11, p. 2411, 2019.
24. H. L. Jadhav and V. S. Kanade, "Optimizing PIR sensor placement for maximum coverage," in *Proc. 5th Int. Conf. Sensor Networks (ICSN)*, Singapore, 2020, pp. 34–41.
25. S. B. Rohde, "Over-the-air firmware updates in embedded IoT devices," *IEEE Softw.*, vol. 37, no. 5, pp. 45–52, Sept.-Oct. 2020.
26. M. S. Hossain, "Data privacy and ethical implications in IoT-based surveillance," *IEEE Access*, vol. 9, pp. 154460–154473, 2021.
27. P. R. Singh, "A review of network reliability in IoT-based applications," *IEEE Access*, vol. 8, pp. 135707–135720, 2020.
28. R. K. Salunke and S. A. Chavan, "VoIP API-based phone call mechanism for IoT security systems," in *Proc. 6th Int. Conf. on Intelligent Comput. (ICIC)*, Tokyo, Japan, 2019, pp. 72–79.
29. M. T. Scherzer, "Ethical frameworks for smart home surveillance systems," *IEEE Technol. Soc. Mag.*, vol. 39, no. 3, pp. 62–70, Sept. 2020.
30. A. E. Garcia and C. D. Suarez, "A comprehensive review of GSM module integrations in IoT," *IEEE Internet Things Mag.*, vol. 3, no. 4, pp. 20–27, Dec. 2020.
31. L. P. Chang, "Implementation of automated calls using Twilio API in IoT solutions," in *Proc. IEEE Conf. Emerging Tech. (ET)*, Seoul, South Korea, 2020, pp. 110–117.