Smart Home Automation Integration Using ESP32 for Houses with Conventional Wiring & Switch Systems

Vimukthi Pahasara Ginigal Godage

May 2024

Abstract

Smart home automation is a technology that has been widely used in the field of home automation. However, most of these systems necessitate extensive modifications to the existing wiring, switches, and sockets, thereby posing challenges for homeowners seeking an easy-to-install solution. This project aims to address this issue by designing a non-invasive Home Automation System based on the ESP32 platform, requiring minimal alteration to conventional household infrastructure. The proposed solution offers a user-friendly, cost-effective approach tailored to the needs of Sri Lankan households, addressing technical expertise, infrastructure limitations, and economic constraints while ensuring ease of use and flexibility for users. The system's combination of sensor and controller modules creates an efficient and adaptable home automation network.

Keywords— Home Automation, ESP32, Arduino, Over-the-Air web updater, WiFi, Bluetooth, Arduino IDE

1 Problem Statement

Smart Home Automation Systems have come a long way since the start of Internet of Things(IoT). There are many smart home automation systems in the market with many advanced features such as voice commands, integrated AIs, capability to connect using phones, computers, smart televisions, and even smart watches. Some of the popular Smart Home Systems are Smart Things by Samsung, Apple HomeKit, Amazon Alexa, Google Home and Lutron Caséta Wireless. However, there is a common problem with all of these systems. That is, they require a lot of modifications to wiring and require replacing switches, sockets, and bulbs for compatible ones when integrating to an already finished house with conventional wiring and switch system. This process requires a lot of expertise knowledge, modifications, time and money. Some of the modifications need for the aforementioned systems are discussed below.

- **SmartThings by Samsung**
  - ***Integration:*** SmartThings operates through a central hub that communicates with smart devices throughout the home using wireless protocols

such as Zigbee or Z-Wave. To integrate SmartThings into a conventional home, traditional switches must be replaced with smart switches or supplemented with smart plugs. This process involves linking each smart device to the hub for centralized control.

  - ***Difficulty:*** Integrating SmartThings can be challenging due to the need to replace multiple switches or add smart plugs, requiring significant labor and technical expertise in electrical work. Each switch replacement or plug addition requires careful attention to wiring configurations, potentially increasing the complexity and cost of the installation.

- **Apple HomeKit**
  - ***Integration:*** HomeKit relies on a central hub, typically an iPad or HomePod, to communicate with compatible smart devices using Bluetooth or Wi-Fi. Integration into a conventional home involves replacing existing devices with HomeKit-compatible alternatives or adding bridges to make non-compatible devices accessible via the HomeKit ecosystem.
  - ***Difficulty:*** While HomeKit integration doesn't require rewiring, it still presents challenges. Replacing conventional devices with HomeKit-compatible ones can be costly, and configuring bridges for non-compatible devices may require technical expertise. Furthermore, ensuring seamless communication between devices and the HomeKit hub demands meticulous setup and troubleshooting.
- **Amazon Alexa with Echo Devices**
  - ***Integration:*** Alexa functions as a voice-controlled interface for a variety of smart home devices, including lights, switches, plugs, and more. Integrating Alexa into a conventional home involves replacing traditional switches with Alexa-compatible smart switches or adding smart plugs to non-smart devices for voice control.
  - ***Difficulty:*** Integrating Alexa can be labor-intensive, as it requires replacing switches or adding smart plugs throughout the home. Each installation necessitates careful configuration and setup to ensure proper communication with the Alexa ecosystem. Moreover, compatibility issues may arise with certain devices, requiring additional troubleshooting and potentially increasing costs.
- **Google Home with Google Assistant**
  - ***Integration:*** Google Home offers voice control and centralized management of smart devices using Google Assistant. Integrating Google Home into a conventional home involves replacing existing switches with Google-compatible smart switches or adding smart plugs for voice-controlled operation.
  - ***Difficulty:*** Similar to Alexa, integrating Google Home requires replacing switches or adding plugs, which can be time-consuming and technically demanding. Configuring devices to communicate with Google Assistant and troubleshooting compatibility issues may further complicate the integration process, potentially increasing labor and costs.
- **Lutron Caséta Wireless**

- ***Integration:*** Lutron Caséta Wireless provides smart switches and dimmers that can replace traditional switches without requiring a neutral wire. These switches communicate with a central hub via Lutron's proprietary wireless protocol for centralized control.
- ***Difficulty:*** While Lutron Caséta Wireless offers a less invasive solution compared to systems that require rewiring, integration still presents challenges. Replacing switches with Lutron smart switches demands technical expertise in electrical work, and configuring the system for optimal performance requires meticulous setup and calibration. Additionally, the cost of Lutron-specific devices may be higher than other solutions, impacting the overall expense of integration.

In summary, the current market offers various Home Automation Systems catering to the increasing demand for smart, interconnected homes. However, most of these systems necessitate extensive modifications to the existing wiring, switches, and sockets, thereby posing challenges for homeowners seeking an easy-to-install solution. This project aims to address this issue by designing a non-invasive Home Automation System based on the ESP32 platform, requiring minimal alteration to conventional household infrastructure.

2 Problem Context

The above problem statement is going to be considered in the context of Sri Lanka from this point onward. [1] shows that 30% of the Sri Lankans have a only a vague idea and 22% has no idea about Smart Home Automation Systems. Most of the Sri Lankan households cannot afford the aforementioned Smart Home Systems as they are very expensive. Even if some one considered integrating a Smart Home System to a house that is in the planning stage the trend to integrate a Smart Home System into already built house with a conventional wiring and switch system is very low considering the cost , time and modifications that need to be made to the house. Some of the limitations are discussed in detail below.

- **Economic Constraints:** Many Sri Lankan households operate on limited budgets, making the high cost of current Smart Home Systems a significant barrier. This includes not only the initial purchase price but also the ongoing costs for maintenance and potential upgrades.
- **Technical Expertise:** There is a limited availability of skilled technicians who can handle the intricate installations required for these systems. Most households would find it challenging to access affordable and reliable technical support for the complex integration of smart devices.
- **Infrastructure Limitations:** Sri Lankan homes, especially older ones, often have outdated electrical systems that are not compatible with modern smart devices. Retrofitting these homes involves substantial rewiring and replacement of switches, sockets, and bulbs, which can be both disruptive and costly.
- **Market Availability:** The availability of smart home devices and accessories is limited in Sri Lanka. Importing these devices can increase costs due to shipping fees and import taxes, further reducing their accessibility.
- **Cultural and Behavioral Factors:** There is a general hesitation towards adopting new technologies, especially those that require significant changes to

existing home infrastructure. The perceived complexity and disruption associated with integrating Smart Home Systems deter many homeowners from considering them.

- **Power Supply Reliability:** Frequent power outages and voltage fluctuations in Sri Lanka can affect the performance and reliability of smart home devices, which often depend on a stable power supply for optimal functioning.
- **Internet Connectivity:** While urban areas may have decent internet connectivity, many rural regions in Sri Lanka still suffer from poor internet infrastructure. Since most Smart Home Systems rely on constant internet access for control and updates, this poses a significant challenge.

This project is focused on mainly addressing the Economic Constraints, Technical Expertise and Infrastructure Limitations while giving a solution that looks traditional(without replacing the switched, sockets and modifying the wiring) in outward to lower the hesitation for trying out Smart Home Automation as a new technology.

3 Proposing Solution

The proposing solution for this problem is creating a Smart Home System that can be installed on top of the existing wiring, switches and sockets. Also, giving options to choose whether they would like to replace the existing bulbs with more advanced dimmable LED bulbs that can change the brightness. The difference between dimmable and non-dimmable LED bulbs as follows.

Dimmable and non-dimmable LED bulbs serve different purposes and applications, each with distinct characteristics. Dimmable LED bulbs, equipped with internal circuitry for adjustable brightness, can be used with compatible dimmer switches to create various lighting atmospheres and reduce energy consumption. These LED bulbs are ideal for residential areas like living rooms and bedrooms, as well as commercial spaces such as restaurants and offices, where flexible lighting is desirable. However, they are typically more expensive due to the additional technology required. In contrast, non-dimmable LED bulbs offer fixed brightness and are simpler and more robust since they lack the complex circuitry for dimming. They are cost-effective and suited for areas where consistent lighting is sufficient, such as kitchens, bathrooms, outdoor spaces, and industrial settings. Non-dimmable LED bulbs are designed to work with standard switches and are not compatible with dimmer switches, as using them together can cause flickering, buzzing, and reduced bulb lifespan [2].

As the initial step of the project, switching LED bulbs on/off depending on the presence of a human, change brightness of the space according to the luminosity in the space and the time of day, switching other electric appliances on and off, switching off the fan depending on the presence of a human and controlling fan speed remotely are the automations going to be implemented.

There will be two modules, one containing the sensors to get data from the environment such as presence of a human, luminosity of the space. The other module will be the controller module which will be installed near the wall switches, plug sockets to control the LED bulbs and other electric appliances. The controlling of the LED bulbs and electric appliances will be done through some parallel connections made to the wall switch or wall socket which can easily be done using the instructions provided. All the modules will be connected to a central Wi-Fi network. There will be no need for Wi-Fi repeaters as the sensor modules and the controller modules are connected in

a mesh topology. A message sent by the central router can be propagated to a larger distance by sending that messages through multiple sensor and controller modules. These features will remove or lessen the Technical Expertise and the Infrastructure Limitations constraints compared to the most of the Smart Home Automation Systems in the market.

Also, ESP32 is a easily accessible MCU with a reasonable price. The other sensors can also be procured for a reasonable price when manufacturing in bulk. These choices will help to lessen the Economic Constraints for Smart Home Automation System.

The user can remove the controller unit from the wall switch or wall socket it is connected to at any time without making any harm to the normal operation of the house's LED bulbs and electric appliances. Also there is a manual overriding from the smart phone application as well as the from a mechanical switch in the controller module to override the functionality of the controller unit and to switch back to traditional method. These features that enable the user to come back to the previous state without again going through a major process and being able to override the automated functionalities easily in case of module failure will give users peace of mind, which in turn may lead Sri Lankans to add Smart Home Automation to their houses without any worry.

In summary, the proposed smart home automation solution offers a user-friendly, cost-effective approach tailored to the needs of Sri Lankan households, addressing technical expertise, infrastructure limitations, and economic constraints while ensuring ease of use and flexibility for users.

4 Why ESP32?

The ESP32 is a powerful and versatile microcontroller unit (MCU) that offers a comprehensive set of features ideal for smart home automation systems. Its integrated Wi-Fi and Bluetooth, energy-efficient design, high performance, rich peripheral support, cost-effectiveness, robust security features, and extensive development resources make it an excellent choice for both simple and complex smart home applications. Some of these feature are detailed below[3].

- **Integrated Wi-Fi and Bluetooth Connectivity**
  The ESP32 features dual-mode Wi-Fi (802.11 b/g/n) and Bluetooth 4.2 (both classic and BLE). This dual connectivity allows the ESP32 to communicate with a wide range of smart home devices, including smartphones, sensors, and other IoT devices. The Wi-Fi functionality supports both Access Point (AP) and Station (STA) modes, which can be simultaneously active, enabling the ESP32 to serve as a network bridge or hub.
- **Low Power Consumption**
  The ESP32 is engineered for energy efficiency, making it suitable for battery-operated devices. It offers several power-saving modes, such as:
  - *Active Mode:* The device operates at full power, suitable for intensive tasks. Modem Sleep Mode: Wi-Fi and Bluetooth radios are disabled, but the CPU remains active, reducing power consumption while maintaining computational capabilities.
  - *Light Sleep Mode:* The CPU and peripherals are paused, and only the RTC (Real-Time Clock) and select peripherals are operational, significantly lowering power usage.

  - *Deep Sleep Mode:* The CPU and most of the RAM are powered down, leaving only the RTC and a small portion of RAM active, allowing the device to wake up via timers or external events.
  - *Ultra-Low Power Co-Processor:* This separate low-power processor can handle simple tasks like ADC conversions, sensor readings, and threshold checks while the main processor remains in deep sleep, further conserving energy.

- **High Performance**

  The ESP32's dual-core Tensilica LX6 microprocessor operates at clock speeds up to 240 MHz, providing ample processing power for complex tasks and real-time operations. It supports simultaneous execution of tasks, making it capable of handling multiple smart home functions, such as real-time data processing from various sensors and managing multiple device communications.

- **Rich Peripheral Support**

  The ESP32 is equipped with a diverse array of peripheral interfaces, enabling it to connect and control a wide variety of external components:

  - *GPIO:* Up to 36 General Purpose Input/Output pins for interfacing with buttons, LEDs, and other digital devices.
  - *ADC:* 18 channels of 12-bit Analog-to-Digital Converters for reading analog sensors.
  - *DAC:* 2 channels of 8-bit Digital-to-Analog Converters for generating analog signals.
  - *I2C:* 2 Inter-Integrated Circuit interfaces for communication with sensors, displays, and other peripherals.
  - *SPI:* 4 Serial Peripheral Interface buses for high-speed data transfer with devices like memory cards and displays.
  - *UART:* 3 Universal Asynchronous Receiver-Transmitter interfaces for serial communication.
  - *PWM:* 16 Pulse Width Modulation channels for controlling motors, LEDs, and other actuators.
  - *SDIO:* 2 Secure Digital Input Output interfaces for connecting SD cards and other high-speed data transfer devices.

- **Cost-Effective**

  Despite its advanced features and capabilities, the ESP32 is available at a relatively low cost compared to other microcontrollers with similar performance. This affordability makes it an attractive option for large-scale deployment in smart home systems without compromising on functionality or quality.

- **Open-Source Development Environment**

  The ESP32 benefits from extensive support within the open-source community, with resources and development tools such as:

  - *ESP-IDF:* Espressif IoT Development Framework, a robust and flexible SDK for developing applications specifically for the ESP32.
  - *Arduino IDE:* Compatibility with the popular Arduino ecosystem, allowing developers to leverage a vast array of libraries and tools.

- *PlatformIO:* An open-source ecosystem for IoT development, supporting multiple development platforms and frameworks, including the ESP32.
- *Micropython:* A lean and efficient implementation of Python 3 for microcontrollers, enabling rapid development and prototyping.

- **Security Features**
  The ESP32 incorporates advanced security mechanisms to protect against unauthorized access and ensure data integrity:
  - *Secure Boot:* Ensures that only trusted firmware is executed by verifying the integrity of the firmware image before execution.
  - *Flash Encryption:* Encrypts the firmware and data stored in flash memory, protecting sensitive information from being read or tampered with.
  - *Hardware Accelerated Encryption:* Supports AES (Advanced Encryption Standard), SHA (Secure Hash Algorithm), RSA (Rivest–Shamir–Adleman), and ECC (Elliptic Curve Cryptography), providing fast and efficient cryptographic operations.

- **Scalability**
  The ESP32's versatility allows it to scale from simple to complex smart home automation tasks. Its powerful processor and extensive peripheral support enable it to manage everything from single-point controls, like a smart light switch, to comprehensive systems integrating multiple sensors, actuators, and user interfaces.

5 Literature Review

Over the years, the landscape of smart home technology has flourished, showcasing an array of innovative solutions. Among these, many systems have harnessed the power of Arduino and Bluetooth technology [4, 5], albeit encountering limitations such as communication ranges restricted to around 50 meters and interface constraints for home appliances and sensors. A notable example is a networked monitoring system [6] which utilized Bluetooth for local interfacing and internet-based remote monitoring via RTP and web-based GUI. While this system demonstrated versatility across various home automation applications, it required minor adjustments to optimize performance.

Diving deeper, an intriguing study explored Energy Management in Smart Homes using Bluetooth Low Energy [7]. By analyzing the impact of standby appliances and high-power loads during peak hours on energy consumption charges, researchers uncovered an efficient approach to curbing peak load demand and reducing electricity costs, all while enhancing user comfort.

In the realm of Smart Home Automation, a captivating project emerged, centered on light and temperature sensors, with Arduino serving as the master controller and Visual Basic as the programming language [8]. This sophisticated setup allowed for nuanced control of home appliances via voice commands, complemented by a robust security system leveraging Matlab GUI.

Transitioning to Wi-Fi-based solutions, a comprehensive automation system was designed to manage power and security within the home [9]. With capabilities to monitor lights, temperature, smoke, and motion, this system offered both local and remote control, albeit with specific web browser configurations and server IPs required for remote access.

A significant leap forward came with the integration of Arduino and Wi-Fi in a Smart Home Automation and Security System [10]. This cutting-edge system boasted an integrated web server and an intuitive Android app for remote device control. Its ability to automatically adjust lights and fans based on sensor inputs underscored its sophistication, eliminating the need for a dedicated server PC.

In parallel, a theoretical model delved into the intricate factors influencing user acceptance of smart homes [11], shedding light on elements like trust, awareness, enjoyment, and perceived risks. This insightful study emphasized the importance of technology education and awareness in mitigating potential risks and enhancing overall quality of life.

Beyond individual systems, modular approaches to home automation emerged, leveraging Wireless Networks, REST APIs, and Android apps [12]. While offering scalability, such modular designs often incurred higher costs and lacked comprehensive network security. Additionally, Raspberry Pi-based systems [13] showcased expanded functionalities, albeit at a higher cost, catering to diverse household needs.

In the realm of IoT, groundbreaking initiatives such as MQTT-based home automation using ESP8266 Node MCU [14] demonstrated remarkable efficiency in battery usage for remote monitoring and control. However, challenges such as the lack of customized GUI for remote access and cloud computing capabilities underscored the evolving nature of these technologies.

Furthermore, IoT-based environmental monitoring systems [15, 16] and smart security systems [17] showcased the integration of cutting-edge technology to enhance safety and comfort within homes. Despite their benefits, these systems faced challenges such as data storage limitations and sensor interfacing constraints, signaling areas for future innovation.

In addition, recent work has demonstrated a Home Automation system capable of real-time device control using web services, the Arduino board, and cloud infrastructure [18]. This innovative setup utilizes current sensors to monitor electricity consumption via GUI, with a web server-based notification system informing users of device statuses. However, challenges such as data storage limitations and introduced delays highlight areas for further refinement.

In conclusion, the evolution of smart home technology continues to captivate with its blend of innovation and practicality. Wi-Fi-based sensing systems, in particular, hold promise for their cost-effectiveness and convenient deployment [19, 20], paving the way for a future where homes are not just smart but truly intelligent.

6 SWOT Analysis of the Proposed Solution

1. **Strengths**

   - ***Non-invasive Installation:*** The proposed solution offers a non-invasive approach, requiring minimal alteration to conventional household infrastructure, thereby reducing installation time and costs.
   - ***Cost-effective Solution:*** By utilizing readily available and affordable components such as the ESP32 MCU and sensors, the project addresses economic constraints, making smart home automation more accessible to households with limited budgets.
   - ***User-friendly Design:*** The inclusion of manual overrides and easy-to-use smartphone applications enhances user experience, providing flexibility

and peace of mind to homeowners.

- ***Technical Expertise Reduction:*** The project aims to simplify installation and operation, reducing the need for specialized technical expertise, thus making it more accessible to a wider range of users.
- ***Mesh Network Connectivity:*** The use of mesh network topology ensures reliable communication between devices, even in areas with limited Wi-Fi coverage, enhancing system robustness and reliability.

2. **Weaknesses**

   - ***Limited Functionality:*** The initial scope of the project focuses on basic automation tasks such as controlling lights and appliances, potentially limiting its appeal to users seeking more advanced features.
   - ***Dependency on Wi-Fi:*** The reliance on Wi-Fi connectivity for communication between devices may pose challenges in areas with unstable or limited internet access.
   - ***Compatibility Issues:*** While the project aims to provide a versatile solution, compatibility with existing devices and systems may still present challenges, especially in older homes with outdated infrastructure.

3. **Opportunities**

   - ***Market Demand:*** With growing interest in smart home technologies, there is a significant opportunity to capture market share, especially among households seeking affordable and easy-to-install solutions.
   - ***Expansion of Features:*** The project could expand its functionality in the future to include additional automation tasks, integration with voice assistants, or compatibility with a wider range of smart devices, thereby enhancing its appeal to users.
   - ***Partnerships and Collaborations:*** Collaborating with local manufacturers or service providers could help expand distribution channels and improve accessibility to the target market.

4. **Threats**

   - ***Competition:*** The smart home automation market is highly competitive, with established players offering a wide range of solutions. The project faces the threat of competition from existing systems with greater brand recognition and advanced features.
   - ***Regulatory and Compliance Issues:*** Compliance with local regulations and standards, particularly regarding electrical safety and data privacy, could pose challenges and delays in product development and deployment.
   - ***Technological Obsolescence:*** Rapid advancements in technology could render the proposed solution outdated or less competitive over time if not regularly updated or adapted to incorporate new features and capabilities.

7 Project Implementation

7.1 Architecture

There are two types of modules in the system. First one is a sensor module and the second one is a controller module. Sensor modules are placed within such a way that the sensors can pickup the information(presence of humans and illuminance of the space) from the room, lobby, kitchen or any other space in the house. If there are more than one LED light bulb, fan and socket outlets in the space the sensor module is installed, that one sensor module can be used to control those multiple light bulbs, fans and socket outlets. Sensor modules may or may not be installed near a power outlet. In case the sensor module is installed in a place where it is difficult to get a connection with a power outlet, solar energy in case installed in outdoor setting or rechargeable batteries in case a indoor setting can be used to power the sensor module. But it is recommended to have a power connection from the main supply to have a stable operation without interruptions.

Controller modules are installed in proximity(very near) to the wall switches, wall sockets and only other controller mechanism in place. There are some parallel connections made to the wall switch or wall socket to be able to control the LED light bulbs and other appliances according to the messages from the sensor module and other controller modules. Controller modules are always near the wall switched or wall sockets ensuring a proper power supply from the main supply. It is very important for controller modules to have a proper power supply than sensor modules. In case of a sensor module failure the control module can give the users manual control from the traditional wall switches and wall sockets.

The sensor modules and the controller modules are connected in a mesh [21] Wi-Fi network with a Wi-Fi router(to get internet access and access to a computer when available). This mesh connectivity ensures the messages passing down without any loss even if the transmitting module and the receiving modules are far away with each other. But for this to work there must be other other module(sensor or control) between those to module that can carry the message in a chain.

A computer connected to the Wi-Fi network is needed to setup the system for the first time or to change the configurations. Configurations include what sensors are in what room, what sensor is assigned to the control of which LED bulb or appliance, any custom logic for controlling the LED bulbs or appliances that differ from the already defined one. Once the configuration is created and the firmware for each of the sensor module and the control module is uploaded via OTA(Over the Air) updates, the users can use the mobile app for control the LED bulbs through smart phone or make the system go in to manual control.

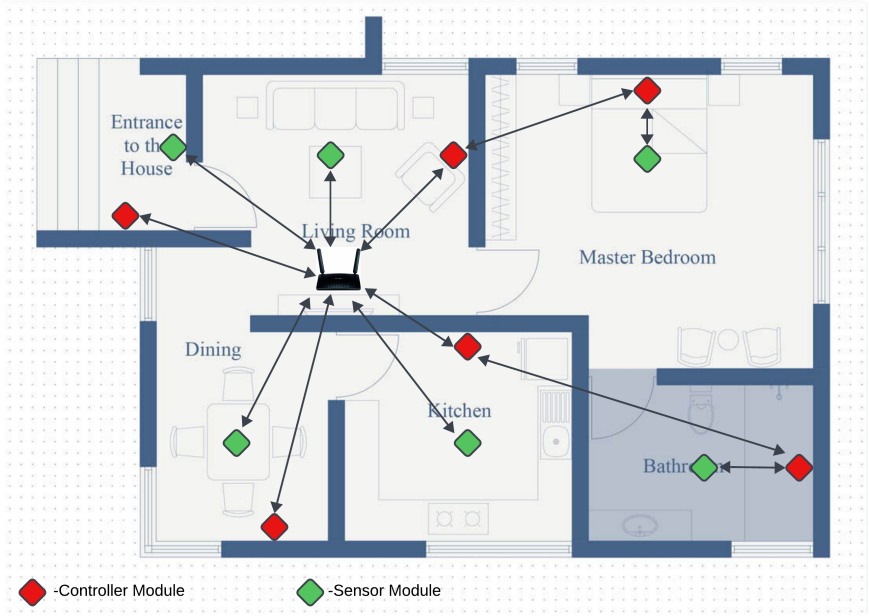


Figure 1: Architecture of the Smart Home Automation System

As can be seen from the figure 1, even though the network is in mesh topology it is not a complete mesh where each node is connected to all the other possible nodes. It is connected in a partial mesh network[21]. This is to ensure that the message passing is kept to a minimum in order to save the energy. If the network is in a complete mesh topology there are a lot of redundant message passing and may lead to performance issues(because the MCU will be busy identifying redundant messages) and energy waste due to unnecessary message passing.

In conclusion, the system's combination of sensor and controller modules creates an efficient and adaptable home automation network. The sensor modules, strategically placed to detect human presence and illuminance, offer versatile control over multiple devices, while controller modules ensure stable power supply and manual control options. The use of a partial mesh Wi-Fi network enhances communication reliability and energy efficiency, minimizing redundant message passing. Initial setup and configurations are managed through a computer, with ongoing control available via a mobile app. This design ensures a seamless, energy-conscious, and user-friendly solution for modern home automation needs.

7.2 Sensor Module

The sensor module of the initial design consist of a human presence detection sensor, a illuminance sensor and a indicator LED for notifying the user of the status. As the human presence detection sensor a mmWave radar sensor is used. The use of a camera poses privacy problems as well as data processing problems. Analyzing the video feed from the camera to detect human presence is not a feat the ESP32 MCU can

achieve. Therefore it is necessary to have a computer connected to the network to do the processing. The next most important problem is the privacy and safety concerns with using a camera to detect the human presence. This becomes most prominent in places such as bedrooms and bathrooms. Therefore to detect the human presence HLK-LD2450 sensor is used.

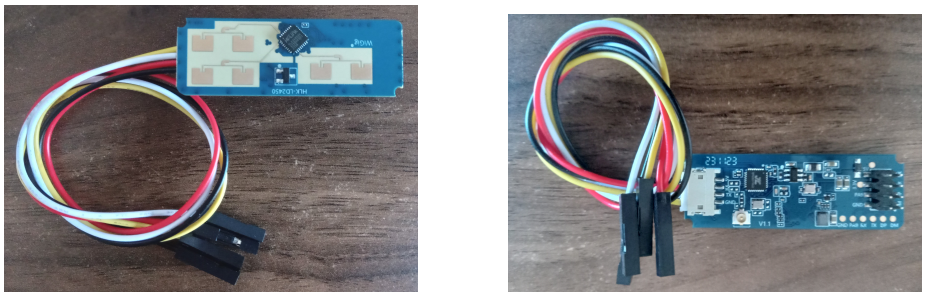
- **HLK-LD2450**

  The HLK-LD2450 sensor is a sophisticated device designed for precise motion detection and illuminance measurement, optimized for integration into smart home automation systems. This sensor utilizes advanced passive infrared (PIR) technology to detect the presence of humans by sensing variations in infrared radiation levels caused by movement within its detection field. Its highly sensitive PIR sensor can accurately detect motion within a range of up to 5 meters, making it suitable for monitoring various indoor environments such as rooms, lobbies, and kitchens.

  In addition to motion detection, the HLK-LD2450 features an integrated illuminance sensor capable of measuring ambient light levels with high precision. This dual-function capability allows the sensor to provide comprehensive environmental data, enabling automated control of lighting systems based on occupancy and natural light availability, thereby enhancing energy efficiency.

  The sensor operates on a low-power consumption model, which can be powered via a direct connection to the main supply, solar energy for outdoor settings, or rechargeable batteries for indoor applications without nearby power outlets. For optimal performance and uninterrupted operation, a stable power connection from the main electrical supply is recommended.

  Connectivity is facilitated through Wi-Fi, supporting integration into a mesh network configuration [22–24]. This ensures robust communication with controller modules, enabling seamless relay of sensor data for automated control of connected devices. The HLK-LD2450 also supports over-the-air (OTA) firmware updates, ensuring the sensor remains up-to-date with the latest features and improvements.

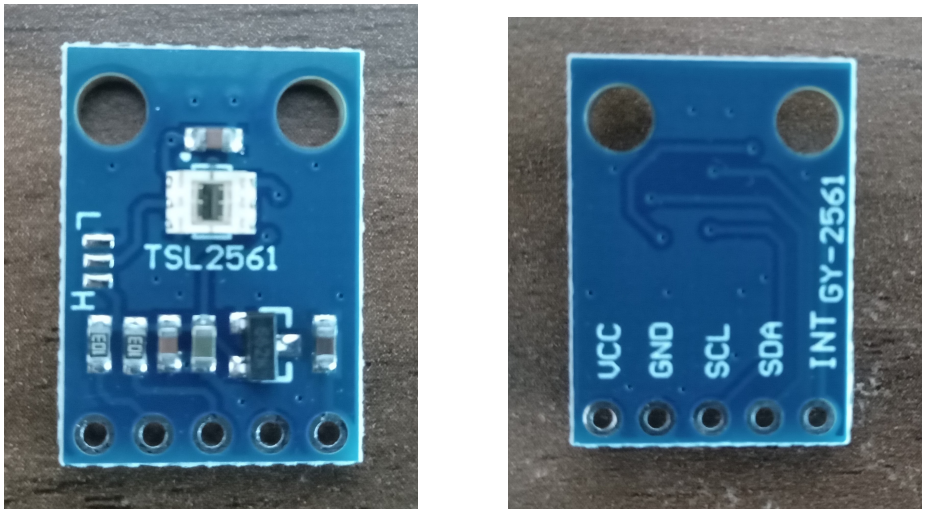


(a) Front (b) Back

Figure 2: HLK-LD2450 human motion trajectory tracking radar module

For the illuminance sensor TSL2561 is used. TSL2561 is a easily accessible sensor used for most of the DIY projects that need a illuminance sensor and a budget friendly sensor.

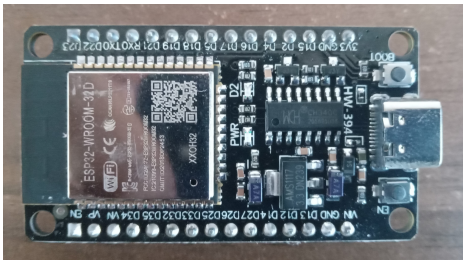
- **TSL2561**
  The TSL2561 is a high-precision digital luminosity (light intensity) sensor, which integrates a broadband photodiode (visible plus infrared) and an infrared-responding photodiode on a single CMOS IC. The sensor communicates over I2C or SMBus interfaces, offering high flexibility in its use within different systems. The TSL2561's dual-diode architecture allows it to perform sophisticated calculations to derive a highly accurate lux value, compensating for the presence of infrared light, which can otherwise distort readings in conventional photodiodes. This feature makes it highly suitable for ambient light sensing in consumer electronics, display backlighting, and industrial applications. The sensor boasts a dynamic range of 0.1 to 40,000 lux with adjustable gain settings to accommodate low and high light conditions. Additionally, the TSL2561 offers a programmable interrupt feature for low-power applications, allowing the host processor to sleep until a significant change in light level is detected.



(a) Front (b) Back

Figure 3: TSL2561 luminosity sensor

For prototyping, the the ESP32 DEVKIT V1 will be used. The ESP32 DevKit V1 is essential for prototyping designs due to its integrated components and user-friendly features. It combines the ESP32 microcontroller, with Wi-Fi and Bluetooth, and essential support components like voltage regulators and a USB-to-serial converter. This integration simplifies development by providing all necessary hardware in one package. With multiple GPIO pins and compatibility with environments like Arduino IDE and ESP-IDF, it enables easy connection



and straightforward coding. Built-in wireless connectivity allows rapid IoT application prototyping. Extensive community support and documentation further reduce the learning curve, making the ESP32 DevKit V1 a cost-effective, efficient solution for developing connected devices.

Circuit diagram of the initial sensor module is given below.

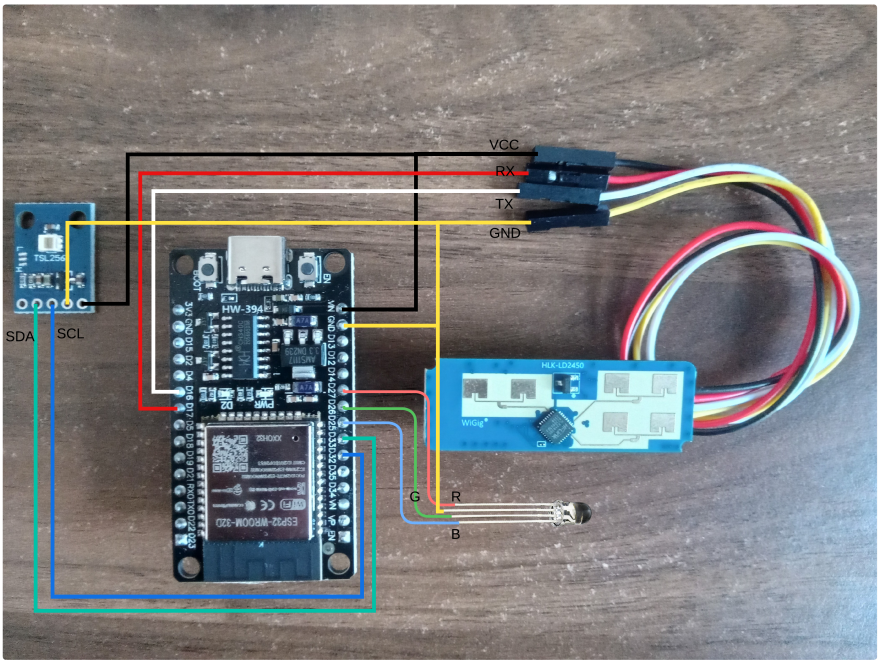


Figure 4: Circuit diagram of the sensor module

HLK-LD2450 is connected to communicate through UART with the ESP32. The RX of the sensor is connected to TX(D17) of the ESP32 board and the TX of the sensor is connected to RX(D16) of the ESP32. The TSL2561 is connected to communicated through I^2C with the ESP32. The serial data line(SDA) is connected to the D33 of the ESP32 and the serial clock line(SCL) is connected to the D32 of the ESP32. The RGB LED's red, green, blue terminals are connected to D27, D26, D25 respectively.

An RGB LED combines red, green, and blue light-emitting diodes in a single package. By adjusting the intensity of each diode, it produces a wide spectrum of colors through additive color mixing. Control is typically achieved using pulse width modulation (PWM) to vary the brightness of each color component.

The importance of using an RGB LED is its capability to create a lot of colors depending on the duty cycle of the PWM signal. In ESP32, a resolution of 0 to 255 for the duty cycle can be archived. These different colors can indicate different status rather than using a single LED per each status. This allows us to add more status and remove status as the project code evolves [25].

7.3 Controller Module

Control module is the module that is responsible for switching on/off LED bulbs, adjust brightness of the LED bubs, on/off fan, adjust speed of the fan and on/off other electric appliances based on the data from the sensor modules or instructions from the mobile application.. There are multiple configurations for the controller module. The first one is the controller module for controlling the switches for low-powered appliances such as LED bulbs, fans. The second configuration of the controller module controls the switches for high-power appliances. The third configuration of the controller module controls the wall sockets.

The circuit diagram of the first configuration of the controller module is given below. It consists of a relay to enable or disable the manual override, SSR array to switch on off the connected appliance and to enable or disable the dimmer circuit, and a dimmer circuit to control the power delivered to the appliance.

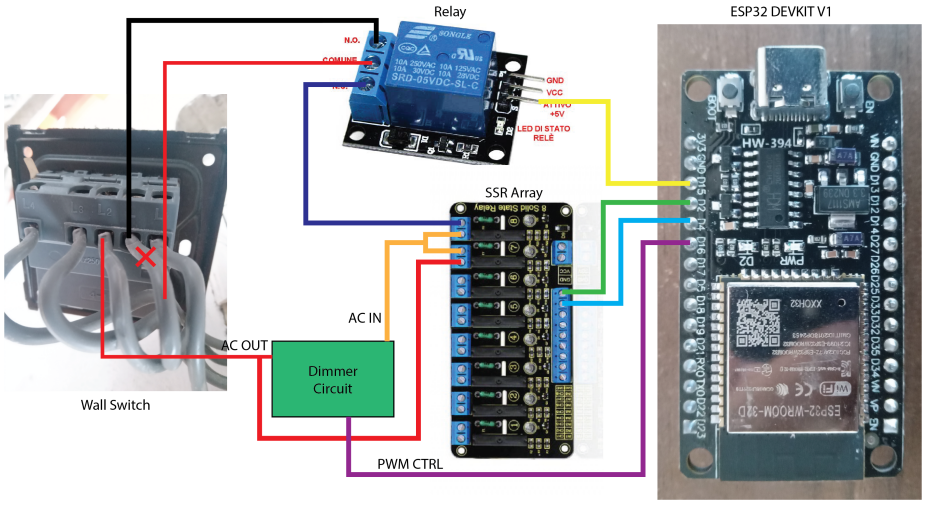


Figure 5: Circuit diagram of the first configuration of the controller module

As can be seen from the diagram the live wire that goes in to the wall switch is disconnected from the wall switch and connected to the common terminal of the relay. The relay used here is a dual channel relay. The normally closed terminal is connected back to the place where the live wire goes in the wall switch. The normally open terminal is connected to one of the SSR(Solid State Relay)(let us name this SSR as SSR-1) in the array. The out terminal of that SSR is connected to another SSR(let's name this SSR as SSR-2) and the AC_IN terminal of the dimmer circuit. Then the AC_OUT terminal and the out terminal of the SSR-2 is connected together and connected to the live wire for the appliance that need to be controller as seen in the circuit diagram. When automatic controls are enabled, the D15 will be at a HIGH voltage level, activating the relay connecting SSR-1 to the live wire. By activating SSR-1, the appliance can be switched on. To activate SSR-1, the D2 GPIO should output a HIGH voltage level(or 1 state). By activating SSR-2, the current can bypass the dimmer circuit. To activate SSR-2, the D4 GPIO should output a HIGH level(or

1 state). The importance of bypassing the dimmer circuit comes when users are not replacing conventional LED bulbs with dimmable LED bulbs, conventional LED bulbs can be driven without any flickering [2]. The dimmer circuit used for the prototype is a TRIAC based dimmer circuit that use phase angle controlling for variable power supplying to the load. The circuit is described in this article, [26].

The circuit diagram of the second configuration of the controller module is given below. It consist of two relay to enable or disable the manual override and to switch on/off the appliance.

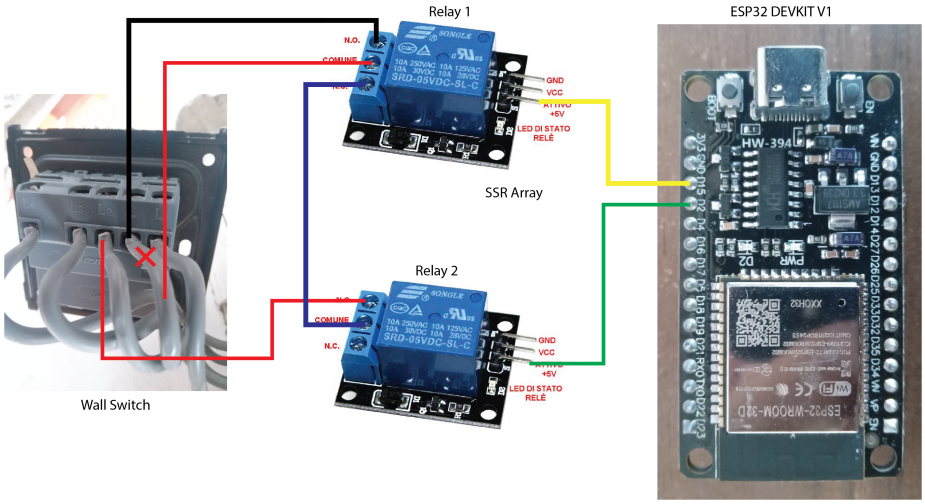


Figure 6: Circuit diagram of the second configuration of the controller module

The difference between this controller module configuration and the first configuration is that this configuration does not have a dimmer circuit and does not use SSR(Solid State Relay) for switching on/off the appliance. The reason this configuration does not have a dimmer circuit is that limiting the power for high-power appliances may harm the appliances and the circuit for such power-limiting circuit suitable for handling high power requires a lot of large components which in turn require a lot of space for the circuitry. The reason for using mechanical relays instead of SSRs is that mechanical relays can handle high powers and mechanical relays that can handle high power are relatively cheap compared to SSRs that can handle high power. The same as the first configuration, by enabling the Relay 1 (by making the D15 GPIO voltage level HIGH) the control can be taken from manual control to automatic. By enabling the Relay 2(by making the D2 GPIO voltage level HIGH) the appliance can be switch on/off according to the data from sensors or instructions from the mobile application.

In both configuration 1 (figure 5 and configuration 2(figure 6, the state 0 in GPIO D15 means that the control of the appliances goes back to the manual control. Therefore, setting this GPIO to voltage level LOW(state 0) by using the instructions from the mobile application can give the user the manual control. Also, if something happens to the ESP32 board, by introducing an external pull-down resistor for the D15 GPIO, manual control will be enabled automatically.

The circuit diagram of the third configuration of the controller module is given below. It consist of a relay to switch off the appliance and a current measuring sensor used to sense the amount of current drawn by the appliance.

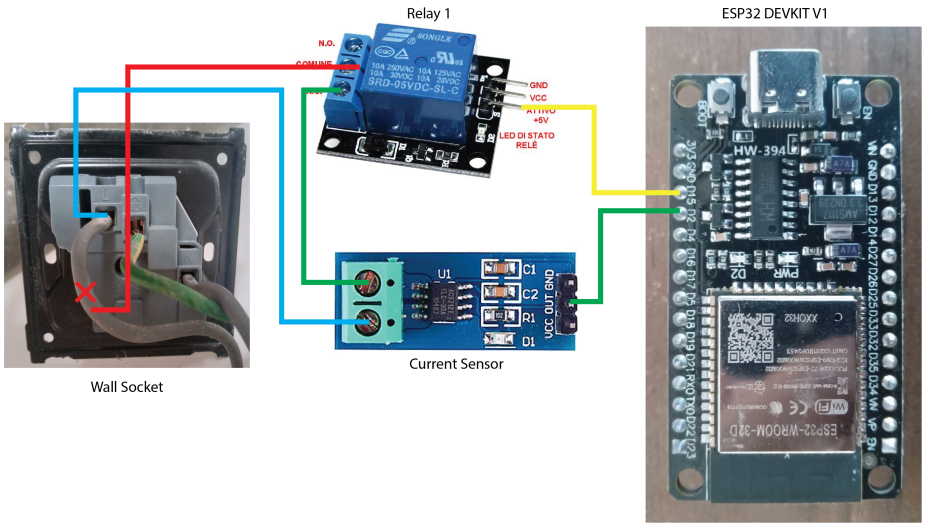


Figure 7: Circuit diagram of the third configuration of the controller module

This configuration consists of a mechanical relay and a current sensor. The current sensor used for the prototype is ACS712. This configuration is different from the previous two in functionality that this configuration controls the wall sockets instead of wall switches. Since the switch mechanism is hidden inside the casing in most wall sockets, this configuration is only able to switch an electrical appliance connected to the socket. The importance of the current sensor comes from a scenario such as the user has forgotten to switch off the electric iron and left the room. The current sensor will detect the current being drawn while the room is without a human presence. In such a case, this controller will cut the power for the socket.

Please note that in all the configurations up to now, the GPIOs of the ESP32 are directly connected to the Relays and the SSRs. This is because in the prototype, a relay module is used that contains the relay and the driver IC. For SSRs also, a module of SSR array is used. For production level circuits, additional components are needed as instead of modules, individual components will be soldered onto the PCB.

7.4 Configuration & Functionality

When a user first attempts to configure the sensor modules and the controller modules, these modules will not contain any code with logic for automation. First the user has to connect all the modules to the WiFi network. To get the the modules connected to the WiFi network, the modules need the WiFi SSID and the password. To configure the SSID and the password for a module, Bluetooth technology is used.

In this prototype, Arduino Language and Arduino IDE are used to program the ESP32 MCU. Using the Arduino language and IDE for ESP32 prototyping offers sev-

eral advantages, particularly for beginners and those aiming for quick development. The simple syntax, extensive libraries, and user-friendly interface make it accessible and easy to use. The Arduino IDE, with its straightforward interface, one-click code upload, and built-in serial monitor, enhances the development experience. A vast community and comprehensive documentation provide ample support, while cross-platform compatibility allows flexibility in choosing the development environment. These factors facilitate rapid prototyping and iterative testing. However, there are notable disadvantages. The Arduino environment may introduce performance overhead and less optimized code compared to professional-grade tools like ESP-IDF. It also limits access to the ESP32's advanced features and offers basic debugging tools. As projects grow in complexity, managing them in the Arduino IDE can become cumbersome due to limited project management features and modularity. Dependency management can also pose challenges with potential library conflicts. Additionally, the Arduino IDE lacks advanced features like integrated version control, refactoring tools, and comprehensive code analysis. Despite these limitations, the Arduino ecosystem remains a valuable starting point for many, though more advanced users might eventually transition to sophisticated environments for enhanced control and optimization.

ESP32 are capable of connectivity via WiFi as well as Bluetooth. The following code snippet is used to configure the SSID and password for sensor and controll modules [27]. The Preferences library is used to store the serial number values of the module, the module type, the SSID, and the password in the flash memory of the ESP32 [28, 29].

Listing 1: Arduino code for configuring the WiFi SSID and password via Bluetooth

```
#include <WiFi.h>
#include "String.h"
#include "BluetoothSerial.h"
#include <Preferences.h>

//Indicator LED
#define PIN_RED    27 // GPIO27
#define PIN_GREEN  26 // GPIO26
#define PIN_BLUE   25 // GPIO25

BluetoothSerial SerialBT;
Preferences prefs;
WiFiClient  client;

typedef struct {
  char serial_num[20];
  uint8_t type_of_module;
  uint8_t ssid;
  uint8_t pwd;
} info_t;

String ssid_pass;
char *SSID1 = "POCO";
char *PASS = "9004652173";
```

```
String ssid;
String pass;

bool flag = false;

void error_indication(){
  analogWrite(PIN_RED, 255);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 0);
}

void bluetooth_mode_indication(){
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 255);
}

void wifi_mode_indication(){
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 255);
  analogWrite(PIN_BLUE, 0);
}

void setup()
{
  pinMode(PIN_RED,   OUTPUT);
  pinMode(PIN_GREEN, OUTPUT);
  pinMode(PIN_BLUE,  OUTPUT);

  prefs.begin("module_info", false);
  uint32_t serial = prefs.getUInt("serial_num", 0);
  uint8_t type = prefs.getUChar("type", 2);
  String ssid = prefs.getString("ssid", "");
  String pwd = prefs.getString("pwd", "");

  if (serial == 0 || tyep == 2){
    Serial.println("Important information are missing
----from the module. Please contact support.")
    error_indication();
  }else{
    Serial.begin(115200);
    Serial.println("Welcome to ESP32 Home Automation System");

    if (ssid == "" || pwd == ""){
      Serial.println("WiFi network not setup. Please setup
------using bluetooth");
      ThingSpeak.begin(client);
      if(!SerialBT.begin("ESP32"))
      {
```

```
        Serial.println("An error occurred initializing
--------Bluetooth");
        error_indication();
      }else{
        flag = true;
        bluetooth_mode_indication();
      }
    }else{
      WiFi.begin(SSID1, PASS);

      Serial.print("Connecting");
      while (WiFi.status() != WL_CONNECTED)
      {
        delay(500);
        Serial.print(".");
      }
      Serial.println();
      Serial.print("Connected, IP address: ");
      Serial.println(WiFi.localIP());
      wifi_mode_indication();
    }
  }
}

void loop()
{
  while(flag == true)
  {
    while(SerialBT.available())
    {
      ssid_pass = SerialBT.readString();
      Serial.println(ssid_pass);

      for (int i = 0; i < ssid_pass.length(); i++)
      {
        if (ssid_pass.substring(i, i+1) == ",")
        {
          ssid = ssid_pass.substring(0, i);
          pass = ssid_pass.substring(i+1);
          prefs.putString("ssid", ssid);
          prefs.putString("pwd", pwd);
          Serial.print("SSID =");Serial.println(ssid);
          Serial.print("Password =");Serial.println(pass);
          delay(2000);
          flag = false;

          int n1 = ssid.length();
          char char_array1[n1 + 1];
          strcpy(char_array1, ssid.c_str());
```

```
            int n2 = pass.length();
            char char_array2[n2 + 1];
            strcpy(char_array2, pass.c_str());

            WiFi.begin(char_array1, char_array2);

            Serial.print("Connecting");
            while (WiFi.status() != WL_CONNECTED)
            {
              delay(500);
              Serial.print(".");
            }
            Serial.println();
            Serial.print("Connected, IP address: ");
            Serial.println(WiFi.localIP());
            wifi_mode_indication();
            break;
          }
        }
      }
    }
}
```

The above code is modified to have a OTA(Over-the-Air) web updater so that after the installation also the firmware can be updated and bug fixed. Using ESP32 Over-the-Air (OTA) updates offers significant advantages, including the convenience of remote updates, reduced downtime, and cost efficiency by eliminating the need for physical access and manual maintenance. OTA updates enhance security through timely patches and consistent version control, while also providing flexibility for feature updates and bug fixes. This improves user experience by seamlessly delivering new functionalities. However, OTA updates come with challenges such as implementation complexity, reliability concerns due to potential interruptions, and network dependency. Security risks must be managed to prevent unauthorized access and data exposure, requiring robust encryption and secure update servers. Additionally, resource constraints like processing overhead and storage requirements can impact device performance. Effective version control and update coordination are essential, especially in large-scale deployments, to ensure consistency and reliability. Despite these challenges, the benefits of OTA updates make them a valuable tool for maintaining and enhancing ESP32-based devices.

The following code snippet shows the initial code in all the sensors and control modules, out of the box along with the OTA web updater [30, 31].

Listing 2: Arduino code for configuring the WiFi SSID and password via Bluetooth with OTA web updater

```
#include <WiFi.h>
#include "String.h"
#include "BluetoothSerial.h"
#include <Preferences.h>
#include <WiFiClient.h>
#include <WebServer.h>
```

```
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";

WebServer server(80);

/*
 * Login page
 */
const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
        "<tr>"
            "<td colspan=2>"
                "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                "<br>"
            "</td>"
            "<br>"
            "<br>"
        "</tr>"
        "<td>Username:</td>"
        "<td><input type='text' size=25 name='userid'><br></td>"
        "</tr>"
        "<br>"
        "<br>"
        "<tr>"
            "<td>Password:</td>"
            "<td><input type='Password' size=25 name='pwd'><br></td>"
            "<br>"
            "<br>"
        "</tr>"
        "<tr>"
            "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
        "</tr>"
    "</table>"
"</form>"
"<script>"
    "function check(form)"
    "{"
    "if(form.userid.value=='admin' && form.pwd.value=='admin')"
    "{"
    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";
```

```
/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'>"
"</script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
    "<input type='file' name='update'>"
         "<input type='submit' value='Update'>"
     "</form>"
 "<div id='prg'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  "$.ajax({"
  "url: '/update',"
  "type: 'POST',"
  "data: data,"
  "contentType: false,"
  "processData:false,"
  "xhr: function() {"
  "var xhr = new window.XMLHttpRequest();"
  "xhr.upload.addEventListener('progress', function(evt) {"
  "if (evt.lengthComputable) {"
  "var per = evt.loaded / evt.total;"
  "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
  "}"
  "}, false);"
  "return xhr;"
  "},"
  "success:function(d, s) {"
  "console.log('success!')"
 "},"
 "error: function (a, b, c) {"
 "}"
 "});"
 "});"
 "</script>";

//Indicator LED
#define PIN_RED    27 // GPIO27
#define PIN_GREEN  26 // GPIO26
#define PIN_BLUE   25 // GPIO25

BluetoothSerial SerialBT;
Preferences prefs;
```

```
WiFiClient  client;

typedef struct {
  char serial_num[20];
  uint8_t type_of_module;
  uint8_t ssid;
  uint8_t pwd;
} info_t;

String ssid_pass;
char *SSID1 = "POCO";
char *PASS = "9004652173";

String ssid;
String pass;

bool flag = false;

void error_indication(){
  analogWrite(PIN_RED, 255);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 0);
}

void bluetooth_mode_indication(){
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 255);
}

void wifi_mode_indication(){
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 255);
  analogWrite(PIN_BLUE, 0);
}

void initialize_ota_server(){
  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
    }
  }
  Serial.println("mDNS responder started");
  /*return index page which is stored in serverIndex */
  server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
  });
```

```
  server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
  });
  /*handling uploading firmware file */
  server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain",
      (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
  }, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
      Serial.printf("Update: %s\n", upload.filename.c_str());
      //start with max available size
      if (!Update.begin(UPDATE_SIZE_UNKNOWN)) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
      /* flashing firmware to ESP*/
      if (Update.write(upload.buf, upload.currentSize)
            != upload.currentSize) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_END) {
      //true to set the size to the current progress
      if (Update.end(true)) {
        Serial.printf(
            "Update Success: %u\nRebooting...\n", upload.totalSize);
      } else {
        Update.printError(Serial);
      }
    }
  });
  server.begin();
}

void setup()
{
  pinMode(PIN_RED,   OUTPUT);
  pinMode(PIN_GREEN, OUTPUT);
  pinMode(PIN_BLUE,  OUTPUT);

  prefs.begin("module_info", false);
  uint32_t serial = prefs.getUInt("serial_num", 0);
  uint8_t type = prefs.getUChar("type", 2);
  String ssid = prefs.getString("ssid", "");
  String pwd = prefs.getString("pwd", "");

  if (serial == 0 || tyep == 2){
```

```
    Serial.println("Important information are missing
----from the module. Please contact support.")
    error_indication();
  }else{
    Serial.begin(115200);
    Serial.println("Welcome to ESP32 Home Automation System");

    if (ssid == "" || pwd == ""){
      Serial.println("WiFi network not setup. Please setup
------using bluetooth");
      ThingSpeak.begin(client);
      if(!SerialBT.begin("ESP32"))
      {
        Serial.println("An error occurred initializing Bluetooth");
        error_indication();
      }else{
        flag = true;
        bluetooth_mode_indication();
      }
    }else{
      WiFi.begin(SSID1, PASS);

      Serial.print("Connecting");
      while (WiFi.status() != WL_CONNECTED)
      {
        delay(500);
        Serial.print(".");
      }
      Serial.println();
      Serial.print("Connected, IP address: ");
      Serial.println(WiFi.localIP());
      wifi_mode_indication();
      initialize_ota_server();
    }
  }
}

void loop()
{
  while(flag == true)
  {
    while(SerialBT.available())
    {
      ssid_pass = SerialBT.readString();
      Serial.println(ssid_pass);

      for (int i = 0; i < ssid_pass.length(); i++)
      {
        if (ssid_pass.substring(i, i+1) == ",")
        {
```

```
            ssid = ssid_pass.substring(0, i);
            pass = ssid_pass.substring(i+1);
            prefs.putString("ssid", ssid);
            prefs.putString("pwd", pwd);
            Serial.print("SSID =");Serial.println(ssid);
            Serial.print("Password =");Serial.println(pass);
            delay(2000);
            flag = false;

            int n1 = ssid.length();
            char char_array1[n1 + 1];
            strcpy(char_array1, ssid.c_str());

            int n2 = pass.length();
            char char_array2[n2 + 1];
            strcpy(char_array2, pass.c_str());

            WiFi.begin(char_array1, char_array2);

            Serial.print("Connecting");
            while (WiFi.status() != WL_CONNECTED)
            {
              delay(500);
              Serial.print(".");
            }
            Serial.println();
            Serial.print("Connected, IP address:");
            Serial.println(WiFi.localIP());
            wifi_mode_indication();
            initialize_ota_server();
            break;
          }
        }
      }
    }

    server.handleClient();
    delay(1);
}
```

Now that the user has configured Wi-Fi SSID and password for all the sensor and controller modules, now it is time to upload the firmware with the logic for automation to sensor and controller modules using the OTA web updater.

For this a configuration tool running on a web browser is used. It is important to note that a computer is necessary when setting up for the first time, updating the firmware, when a change is made to the configuration of the overall system to re-calibrate the system for operation without any issues and errors.

The user has to plan where the sensor modules and the controller modules will be placed and create a map show as below to recognize the positioning of the sensor and controll modules.

Figure 8: Example of identification of positions of the sensors

The recognized coordinates has to be given to the configuration tool as a list containing the serial number, module type and the coordinates. Module type is a integer. If the module type is 0, that means the module is a sensor module. If the module type is 1, that means the module is a controller module.

Listing 3: Example configuration input for the configuration tool

```
(Serial Number, Module Type, Coordinates)

58963214702589632147, 1, (x1, y1, z1)
74125896321478523698, 1, (x2, y2, z2)
36985214725896321478, 0, (x3, y3, z3)
12369854785236974125, 0, (x4, y4, z4)
98745632147859632147, 0, (x5, y5, z5)
65412398745632147859, 1, (x6, y6, z6)
85236974125896321478, 0, (x7, y7, z7)
14785236987456321478, 0, (x8, y8, z8)
63214785236974125896, 1, (x9, y9, z9)
36987412589632147852, 0, (x10, y10, z10)

Wi-Fi router 1: (x_, y_, z_)
```

After giving these inputs to the configuration tool, the configuration tool will generate the firmware for each of the modules and send them via OTA to the modules. After that the system will be able to perform the Smart Home Automation tasks.

7.5 Mobile Application

A smart phone application for the Smart Home Automation System will be deployed to the android platform. Android platform is chosen as the platform to release the mobile application initially because the easy accessibility, low fee of Play Store, does not have any restrictions on the development device or the platform. A user can log in to the application by connecting to the Wi-Fi router at least one sensor or controller module is connected to. There are multiple account roles and privileges for the users of the Smart Home Automation System. Admin privilege will grant every permission accessible to a user including manual override, checking status, controll all the LED bulbs, fans and other electronic appliances, push new firmware updates and many more.

Privileges that grant access to LED bulbs, fans and electrical appliances in a certain room or rooms can be created within the application. Users with those permissions can only control those LED bulbs, fans and electrical appliances within that room or rooms. Admin has the privilege to specify whether a LED bulb connected to a controller module is a dimmable or a non-dimmable LED bulb. According to those settings set by the Admin, the controller will enable or disable the dimmer circuit within a controller module.

In addition to the primary features, the Smart Home Automation System's application will also incorporate advanced security protocols to safeguard user data and prevent unauthorized access. Each user account will be secured through multi-factor authentication, ensuring that only verified individuals can access the system. Furthermore, the application will feature a comprehensive activity log that records all interactions with the system, allowing users to monitor and review past actions. This log will include timestamps, user identities, and specific changes made to the system settings or appliance statuses. The system will also support automated routines, where users can set schedules for different appliances, like having the lights dim at a specific time in the evening or the thermostat adjusting the temperature when they leave for work. These enhancements aim to offer a seamless and secure user experience, making the Smart Home Automation System not only versatile and user-friendly but also robust in terms of security and integration capabilities.

8 Cost Estimation

The below cost estimations provides a basic overview of the expenses involved in building the sensor and controller modules. Bulk purchasing and optimizations in the design could potentially reduce these costs further. Also note that the these cost estimations are for building a prototype sensor module and a controller module. The cost of a production ready sensor module and and controller module may differ from these estimates.

8.1 Sensor Module

The following table list the components needed for creating a prototype sensor module and their average price in USD.

No	Component Name	Average Price($)
01	HLK-LD2450 Radar Sensor	15.00
02	TSL2561 Luminosity Sensor	5.00
03	ESP32 DevKit V1	8.00
04	RGB LED	1.00
05	Miscellaneous Components (resistors, capacitors, connectors, PCB, etc.)	10.00
06	220V to 5V converter	5.00

Table 1: Average component prices for sensor module

The total cost for a prototype sensor module is $44.00 which approximately equivalent to LKR 13,000.00

8.2 Controller Module

The following table list the components needed for creating a prototype controller module and their average price in USD.

No	Component Name	Average Price($)
01	Relay Module	3.00
02	ACS712 Current Sensor	5.00
03	ESP32 DevKit V1	8.00
04	RGB LED	1.00
05	SSR Array	8.00
06	Miscellaneous Components (resistors, capacitors, connectors, PCB, etc.)	10.00
07	220V to 5V converter	5.00

Table 2: Average component prices for controller module

The total cost for a prototype controller module is $40.00 which approximately equivalent to LKR 12,000.00

9 Future Directions & Improvements

The integration of smart home automation systems using the ESP32 for houses with conventional wiring and switch systems holds significant promise for the future. However, there are numerous directions and improvements that can be pursued to enhance

the effectiveness, user experience, and capabilities of such systems. Here are several potential future directions and improvements with more detailed descriptions:

- **Simplified Setup Process**

  One of the primary barriers to widespread adoption of smart home technology is the complexity of the setup process. Future developments could focus on creating more user-friendly installation guides and automated configuration tools. Enhancements could include:

  - ***Mobile App Integration:*** Develop a dedicated mobile app that guides users through the installation process step-by-step with interactive tutorials. This app could include visual aids, troubleshooting tips, and real-time support chat.
  - ***Plug-and-Play Modules:*** Design modules that can be easily connected to existing wiring without requiring significant electrical knowledge. These modules would come with clear, color-coded connectors and be pre-configured to minimize setup time, reducing the need for professional installation services.

- **Deep Learning for Enhanced Automation**

  Incorporating deep learning algorithms can significantly improve the automation and adaptability of smart home systems. Potential applications include:

  - ***Behavioral Learning:*** Use machine learning to analyze and predict user habits, allowing the system to anticipate needs and adjust settings automatically. For instance, the system could learn the preferred lighting settings and adjust them according to the time of day and user activity.
  - ***Energy Management:*** Optimize energy consumption by learning patterns of device usage and adjusting operations to minimize waste. The system could automatically turn off lights and appliances when not in use and suggest energy-saving tips based on usage data.

- **Speech Command Integration**

  Enhancing user interaction through voice control can make smart home systems more accessible and convenient. This could involve:

  - ***Natural Language Processing (NLP):*** Implement advanced NLP techniques to understand and respond to a wider range of voice commands. The system could handle complex commands and provide personalized responses based on the user's past interactions.
  - ***Multi-Language Support:*** Expand voice command functionality to support multiple languages and dialects, making the system more inclusive. This feature could be particularly beneficial in multilingual households, ensuring that all members can easily interact with the system.

- **Additional Sensor Integration**

  Expanding the range of sensors can enhance the system's ability to monitor and respond to environmental conditions. New sensors could include:

  - ***Air Quality Sensors:*** Monitor pollutants and provide alerts or automatic responses to improve indoor air quality. The system could trigger air purifiers or ventilation systems when pollutant levels exceed safe thresholds.

  - ***Temperature and Humidity Sensors:*** Fine-tune HVAC systems to maintain optimal indoor climate conditions. These sensors could ensure that each room is kept at a comfortable temperature and humidity level, improving overall comfort and potentially reducing energy costs.
  - ***Motion and Proximity Sensors:*** Improve security and energy efficiency by detecting movement and adjusting lighting and appliances accordingly. For example, lights could automatically turn on when someone enters a room and turn off when the room is vacant.

- **Enhanced Security Measures**

  As smart home systems become more interconnected, security becomes increasingly important. Future improvements could focus on:

  - ***Data Encryption:*** Strengthen data encryption methods to protect user information and prevent unauthorized access. This could involve implementing end-to-end encryption for all communications between devices and the central hub.
  - ***Two-Factor Authentication:*** Implement two-factor authentication for access to the smart home system, ensuring that only authorized users can control devices. This could involve a combination of passwords and biometric authentication, such as fingerprint or facial recognition.

- **Integration with Renewable Energy Sources**

  Smart home systems can be designed to work seamlessly with renewable energy sources, promoting sustainability. This could involve:

  - ***Solar Panel Integration:*** Develop interfaces that allow the smart home system to manage and optimize the use of solar energy. The system could track energy production and consumption, automatically adjusting device usage to maximize the use of solar power.
  - ***Energy Storage Management:*** Integrate with home battery systems to manage energy storage and usage efficiently. The system could ensure that stored energy is used during peak times, reducing reliance on the grid and lowering energy costs.

- **Modular and Scalable Design**

  Creating a modular and scalable system can allow homeowners to expand their smart home setup as needed. Future directions could include:

  - ***Interchangeable Modules:*** Design modules that can be easily added or replaced without affecting the overall system. This modular approach would allow users to start with a basic setup and gradually add more features and capabilities.
  - ***Scalable Network Architecture:*** Ensure the system can handle an increasing number of devices and sensors without performance degradation. This could involve implementing robust network protocols and ensuring that the central hub can manage high data traffic efficiently.

- **Improved User Interfaces**

  Enhancing the user interface can make it easier for homeowners to interact with and control their smart home systems. Potential improvements include:

- ***Intuitive Dashboard:*** Develop a centralized dashboard that provides a comprehensive overview of the home's status and allows for easy control of devices. This dashboard could be accessible via mobile devices, tablets, and desktops, ensuring that users can monitor and control their system from anywhere.
- ***Voice and Gesture Control:*** Explore innovative control methods such as gesture recognition and advanced voice control for a more seamless user experience. Gesture control could allow users to manage their home with simple hand movements, while voice control could provide hands-free convenience.

- **Interoperability with Other Smart Systems**

  Ensuring that the smart home system can communicate and operate with other smart devices and platforms can enhance functionality. This could involve:

  - ***Standard Protocols:*** Adopting and supporting common communication protocols to ensure compatibility with a wide range of third-party devices. This would allow users to integrate various smart devices into a single cohesive system.
  - ***API Development:*** Create robust APIs that allow for easy integration with other smart home systems and services. This could enable advanced automation scenarios, such as integrating the smart home system with smart city infrastructure or other IoT platforms.

By pursuing these future directions and improvements, the ESP32-based smart home automation system can become more user-friendly, efficient, secure, and adaptable, ultimately leading to greater adoption and satisfaction among homeowners.

10 Conclusion

The paper on "Smart Home Automation Integration Using ESP32 for Houses with Conventional Wiring and Switch Systems" explores the design, implementation, and potential future directions of integrating smart home automation using the ESP32 microcontroller. This system is tailored to work with conventional household wiring and switch systems, making it a non-invasive and cost-effective solution for upgrading traditional homes.

The study begins by emphasizing the advantages of the ESP32, a robust microcontroller with built-in Wi-Fi and Bluetooth capabilities, which is both affordable and versatile. This choice addresses the economic constraints often associated with smart home systems, making the technology accessible to a broader range of users. The proposed system leverages the ESP32's capabilities to create a network of sensor and controller modules that communicate over a Wi-Fi-based mesh network, enhancing reliability and coverage even in areas with limited Wi-Fi signal.

Key strengths of the system include its non-invasive installation, user-friendly design, and reduction in the need for specialized technical expertise. The design allows for easy integration with existing home infrastructure, significantly reducing installation time and costs. The inclusion of manual overrides and smartphone applications for control ensures that users can interact with the system seamlessly, providing flexibility and peace of mind.

However, the system's initial scope is somewhat limited to basic automation tasks such as controlling lights and appliances, which may not appeal to users seeking more advanced features. Additionally, its reliance on Wi-Fi connectivity poses challenges in areas with unstable or limited internet access, and compatibility with older devices and infrastructure can be problematic.

The paper outlines several future directions to enhance the system's functionality and user experience. These include developing interchangeable modules for scalability, improving user interfaces with intuitive dashboards and advanced control methods like voice and gesture recognition, and ensuring interoperability with other smart systems through standard communication protocols and robust APIs. These improvements aim to make the system more adaptable, secure, and efficient, ultimately leading to greater adoption and satisfaction among homeowners.

A SWOT analysis highlights the project's potential in capturing market demand for affordable and easy-to-install smart home solutions while also identifying threats from established competitors and the need for ongoing innovation to stay relevant.

In conclusion, the integration of ESP32 for smart home automation in houses with conventional wiring offers a promising solution that balances cost, ease of installation, and user-friendliness. While there are challenges and limitations to address, the system's modular design and potential for future enhancements position it well for widespread adoption. By focusing on improving functionality, user interfaces, and interoperability, the ESP32-based smart home automation system can evolve to meet the growing demands of modern households, paving the way for smarter and more connected living environments.

References

[1] S. Weerawardhana, T. G. Weerakoon, S. Wimalasena, and N. Moganaraj, "Factors influencing the adoption of smart building and service preferences in sri lanka," *Baltic Journal of Real Estate Economics and Construction Management*, vol. 12, no. 1, pp. 18–35, 2024.

[2] C. Ionescu, A. Vasile, N. Codreanu, and R. Negroiu, "Comparative studies on dimming capabilities of retrofit led lamps," in *Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies VIII*, vol. 10010. SPIE, 2016, pp. 758–767.

[3] "Esp32 wi-fi & bluetooth soc espressif systems," accessed: 2024-05-30. [Online]. Available: https://www.espressif.com/en/products/socs/esp32

[4] R. Piyare and M. Tazil, "Bluetooth based home automation system using cell phone," Jun. 2011. [Online]. Available: https://doi.org/10.1109/isce.2011.5973811

[5] D. Satria, S. Yana, R. Munadi, and S. Syahreza, "Sistem peringatan dini banjir secara real-time berbasis web menggunakan arduino dan ethernet," *Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi)*, vol. 1, no. 1, p. 1, Oct. 2017. [Online]. Available: https://doi.org/10.35870/jtik.v1i1.27

[6] G. Song, Z. Wei, W. Zhang, and A. Song, "Design of a networked monitoring system for home automation," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 3, pp. 933–937, 2007.

[7] M. Collotta and G. Pau, "A novel energy management approach for smart homes using bluetooth low energy," *IEEE Journal on selected areas in communications*, vol. 33, no. 12, pp. 2988–2996, 2015.

[8] S. Chattoraj, "Smart home automation based on different sensors and arduino as the master controller," *International Journal of Scientific and Research Publications*, vol. 5, no. 10, pp. 1–4, 2015.

[9] A. ElShafee and K. A. Hamed, "Design and implementation of a wifi based home automation system," *International Journal of Computer and Information Engineering*, vol. 6, no. 8, pp. 1074–1080, 2012.

[10] J. Chandramohan, R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. Gopinath, S. Ranjithkumar *et al.*, "Intelligent smart home automation and security system using arduino and wi-fi," *International Journal of Engineering And Computer Science (IJECS)*, vol. 6, no. 3, pp. 20 694–20 698, 2017.

[11] A. Shuhaiber and I. Mashal, "Understanding users' acceptance of smart homes," *Technology in society*, vol. 58, p. 101110, 2019.

[12] A. Hasibuan, M. Mustadi, I. E. Y. Syamsuddin, and I. M. A. Rosidi, "Design and implementation of modular home automation based on wireless network, rest api, and websocket," in *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE, 2015, pp. 362–367.

[13] D. Pavithra and R. Balakrishnan, "Iot based monitoring and control system for home automation," in *2015 global conference on communication technologies (GCCT)*. IEEE, 2015, pp. 169–173.

[14] R. K. Kodali and S. Soratkal, "Mqtt based home automation system using esp8266," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. IEEE, 2016, pp. 1–5.

[15] D. A. Aziz, "Webserver based smart monitoring system using esp8266 node mcu module," *International Journal of Scientific & Engineering Research*, vol. 9, no. 6, pp. 801–808, 2018.

[16] B. Gadgay, V. Pujari *et al.*, "Iot based smart environmental monitoring using arduino," *Internasional Journal for Research in Applied Science & Engineering Technology*, vol. 5, no. VI, 2017.

[17] R. K. Kodali, V. Jain, S. Bose, and L. Boppana, "Iot based smart security and home automation system," in *2016 international conference on computing, communication and automation (ICCCA)*. IEEE, 2016, pp. 1286–1289.

[18] A. P. Murdan and L. Gunness, "An internet of things based system for home automation using web services and cloud computing," *Journal of Electrical Engineering, Electronics, Control and Computer Science*, vol. 3, no. 1, pp. 29–36, 2017.

[19] H. Jiang, C. Cai, X. Ma, Y. Yang, and J. Liu, "Smart home based on wifi sensing: A survey," *IEEE Access*, vol. 6, pp. 13 317–13 325, 2018.

[20] B. Iyer, N. P. Pathak, and D. Ghosh, "Rf sensor for smart home application," *International Journal of System Assurance Engineering and Management*, vol. 9, pp. 52–57, 2018.

[21] "Mesh networking," accessed: 2024-05-30. [Online]. Available: https://en.wikipedia.org/wiki/Mesh$_n etworking$

[22] "Esp-wifi-mesh - esp32 — esp-idf programming guide latest documentation." [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/esp-wifi-mesh.html?highlight=mesh

[23] Gmag, "Github - gmag11/painlessmesh: Esp8266 based mesh. this is a mirror copy of https://gitlab.com/painlessmesh/painlessmesh please add comments, issues and pull requests on gitlab so that all information is centralized." [Online]. Available: https://github.com/gmag11/painlessMesh

[24] "How to configure an esp mesh network using arduino ide – communicate among and between esp32, esp8266, and nodemcu." [Online]. Available: https://circuitdigest.com/microcontroller-projects/how-to-configure-an-esp-mesh-network-using-arduino-ide-to-communicate-between-esp32-esp8266-and-nodemcu

[25] "Esp32 - rgb led — esp32 tutorial." [Online]. Available: https://esp32io.com/tutorials/esp32-rgb-led

[26] "Ac phase angle control for light dimmers and motor speed control using 555 timer and pwm signal," accessed: 2024-05-30. [Online]. Available: https://circuitdigest.com/microcontroller-projects/ac-phase-angle-control-for-light-dimmers-and-motor-speed-control-using-555-timer-and-pwm-control

[27] E. Gate, "Change esp32 wi-fi credentials from mobile app." Jan. 2021. [Online]. Available: https://www.youtube.com/watch?v=5my1MScDvGE

[28] "arduino-esp32/libraries/preferences/examples/prefs2struct/prefs2struct.ino at master · espressif/arduino-esp32." [Online]. Available: https://github.com/espressif/arduino-esp32/blob/master/libraries/Preferences/examples/Prefs2Struct/Prefs2Struct.ino

[29] S. Santos and S. Santos, "Esp32 save data permanently using preferences library — random nerd tutorials," Mar. 2021. [Online]. Available: https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/

[30] ——, "Esp32 over-the-air (ota) programming — random nerd tutorials," Jun. 2020. [Online]. Available: https://randomnerdtutorials.com/esp32-over-the-air-ota-programming/

[31] "Over the air updates (ota) - esp32 — esp-idf programming guide v5.2.1 documentation." [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/ota.html