



The present work was submitted to
the German-Mongolian Institute of Resources and Technology

DESIGN AND IMPLEMENTATION OF A SMART HOME AUTOMATION SYSTEM USING MECHATRONICS

Bachelor's Thesis

By

BATMUNKH Tuvshinjargal

Study program: Mechatronics engineer

Student ID: B2100090

1st Supervisor/Examiner: Prof. Sungchil Lee

2nd Supervisor/Examiner: Mr. Myagmarjav Bold

Ulaanbaatar/Nalaikh

2025



The present work was submitted to
the German-Mongolian Institute of Resources and Technology

DESIGN AND IMPLEMENTATION OF A SMART HOME AUTOMATION SYSTEM USING MECHATRONICS

Bachelor's Thesis

By

BATMUNKH Tuvshinjargal

Study program: Mechatronics engineer

Student ID: B2100090

1st Supervisor/Examiner: Prof. Sungchil Lee

2nd Supervisor/Examiner: Mr. Myagmarjav Bold

Ulaanbaatar/Nalaikh

2025

Statutory Declaration

Tuvshinjargal Batmunkh

Last Name, First Name

B2100090

Student ID Number

I hereby affirm in lieu of an oath that I provided the submitted bachelor thesis

DESIGN AND IMPLEMENTATION OF A SMART HOME AUTOMATION SYSTEM USING MECHATRONICS

I did not use any sources other than those stated. In case that the work is additionally submitted on a data medium, I declare that the written and the electronic form are completely identical. The work was not submitted in the same or similar form to any examination authority.

Ulaanbaatar/Nalaikh, 05/28/2025

Place, Date

T. Bamuuux

Signature

Acknowledgment

This thesis would not have been possible without the support, guidance, and encouragement of many individuals and institutions.

First and foremost, I sincerely thank Prof. Sungchil Lee, my thesis supervisor, for his expert advice, insightful feedback, and unwavering patience throughout the research and development process.

I am also profoundly thankful to Mr. Myagmarjav Bold, my co-supervisor, whose practical knowledge of embedded systems and smart home applications provided invaluable technical support. His thorough knowledge of mechatronics and IoT systems significantly shaped the direction and quality of this work.

I gratefully acknowledge the assistance of the Campus and Infrastructure department for granting me access to the circuit of the dormitory's kitchen.

I thank my fellow students, Jargal Gurbadam, Narangua Khongorzul, and Odbayar Ganchimeg for participating in system testing, collecting sensor data in the dormitory kitchen, and providing valuable user-experience feedback during the interface design phase. Their constructive observations helped refine the automation logic and improve overall usability.

Thank you to all those who contributed, whether in person or spirit, for making this thesis a reality.

Abstract

This bachelor's thesis presents the design and implementation of a low-cost, mechatronics-based smart home automation system tailored for a double-room apartment in Mongolia. Motivated by advances in IoT, embedded systems, and the need for energy-efficient living environments, the project employs an ESP32-WROOM microcontroller as the central controller, interfaced with environmental sensors (DHT22, MQ-7, LDR, KY-037 sound sensor, ACS712 current sensor) and actuators (4-channel relay module, AC dimmer, L298N motor driver for curtains and fans, and a buzzer). The system's firmware—developed in the Arduino IDE—implements real-time sensor polling, decision-making logic, and cloud connectivity via Sinric Pro for voice-assistant control and Node-RED/Blynk for mobile/web dashboards.

Due to hardware constraints, a KY-037 sound sensor replaced a PIR module for light control, and timed automatic shutdown was introduced to mitigate relay sticking under high loads. The prototype was systematically tested in a dormitory kitchen, achieving a 45 % reduction in idle energy consumption compared to a non-automated baseline. Voice commands like “Turn on kitchen light” and remote toggling via mobile apps demonstrated reliable responsiveness. A comparative analysis with commercial solutions (e.g., Control4-based and Siemens-backed platforms by Moncable LLC and Digital Power LLC) shows that the custom system delivers core automation features at under 10 % of the cost, highlighting its suitability for resource-constrained settings.

Key contributions include a modular hardware and firmware architecture, integration of phase-angle dimming, real-time energy metering, and multi-modal control interfaces. The work confirms the hypothesis that an ESP32-driven automation system can provide energy-efficient, user-friendly smart home functionality in developing-country contexts. Recommendations for future research encompass improved sensing (PIR or camera-based occupancy detection), solid-state switching, mesh networking, and AI-based predictive control.

Table of Contents

List of Figures	5
List of Tables	6
List of Abbreviations	6
1 Introduction	8
1.1 Background	8
1.2 Aim and Objective of Thesis	8
2 Literature Review	10
2.1 Mechatronics in Home Automation	10
2.2 System Architectures in Smart Homes.....	10
2.3 Communication Technologies and Protocols	11
2.4 Sensors	14
2.5 Actuators	18
2.6 Integration Platforms and Control Interfaces.....	20
2.7 Microcontrollers	22
2.8 Industrial solutions for Home automation in Mongolia	23
3 Methodology	25
3.1 System Overview.....	25
3.2 Selection of Sensors and Microcontroller	26
3.3 Software Architecture	28
4 System implementation	31
4.1 Hardware Setup.....	31
4.2 Arduino IDE coding for Home Automation	34
4.3 System Testing and Debugging.....	36
4.4 User Interface and Experience.....	38
5 Results and Discussion	40
5.1 Energy Efficiency by Implementing Smart Home Automation	40
5.2 Comparison with Existing Systems	40
5.3 Advantages and Limitations.....	41
6 Conclusion	42
6.1 Recommendations for Future Work	44
7 References.....	46
8 Appendices	48

List of Figures

<i>Figure 1. Centralized, decentralized and distributed control</i>	10
<i>Figure 2. DHT11, DHT22 Pinouts</i>	14
<i>Figure 3. MQ Series sensors</i>	15
<i>Figure 4. KY-083, PIR, LDR sensors</i>	16
<i>Figure 5. Relay and Relay schematic</i>	18
<i>Figure 6. Triacs</i>	19
<i>Figure 7. IoT based Interface programs</i>	21
<i>Figure 8. Flowchart of the Smart home automation prototype</i>	25
<i>Figure 9. Plan and design of the</i>	26
<i>Figure 10. System Flowcharts and Algorithms</i>	30
<i>Figure 11. ESP32 WROOM- Pinouts Detailed information</i>	32
<i>Figure 12. Overall circuit wiring design (FRITZING)</i>	33
<i>Figure 13. Implemented Automation prototype</i>	34
<i>Figure 14. Code of Collecting the Default kitchen data</i>	35
<i>Figure 15. Circuit wiring design of the Automated kitchen</i>	36
<i>Figure 16. Circuit wiring design of the Default kitchen</i>	37
<i>Figure 17. Graph of Default and Automated kitchen consumption</i>	38
<i>Figure 18. Sinric Pro Application interface</i>	39

List of Tables

<i>Table 1. Comparison of Communication Technologies and Protocols.....</i>	<i>13</i>
<i>Table 2. Specifications of Microcontrollers.....</i>	<i>23</i>
<i>Table 3. Planned components and micro elements.....</i>	<i>28</i>
<i>Table 4. Selected Sensors and Microcontrollers.....</i>	<i>31</i>
<i>Table 5. Electricity calculation of 24 hours data.....</i>	<i>37</i>

List of Abbreviations

- *ADC – Analog-to-Digital Converter*
- *AI – Artificial Intelligence*
- *API – Application Programming Interface*
- *BLE – Bluetooth Low Energy*
- *CAD – Computer-Aided Design*
- *CO – Carbon Monoxide*
- *DAC – Digital-to-Analog Converter*
- *DC – Direct Current*
- *DHT – Digital Humidity and Temperature*
- *DIY – Do It Yourself*
- *FRITZING – Fritzing Circuit Design Software*
- *GET – HTTP GET Method*
- *GPIO – General-Purpose Input/Output*
- *HTTP – Hypertext Transfer Protocol*
- *I2C – Inter-Integrated Circuit*
- *IDE – Integrated Development Environment*
- *LED – Light-Emitting Diode*
- *LTE – Long-Term Evolution*
- *MQ – Metal-Oxide Gas Sensor Series*
- *MQTT – Message Queuing Telemetry Transport*
- *MTBF – Mean Time Between Failures*
- *NTP – Network Time Protocol*
- *PC – Personal Computer*
- *PCB – Printed Circuit Board*
- *PIR – Passive Infrared Sensor*
- *POST – HTTP POST Method*

- *PWM – Pulse-Width Modulation*
- *RC – Resistor-Capacitor*
- *RCPSP – Resource Constrained Project Scheduling Problem*
- *REST – Representational State Transfer*
- *RPI – Raspberry Pi*
- *SMS – Short Message Service*
- *SMTP – Simple Mail Transfer Protocol*
- *SPI – Serial Peripheral Interface*
- *TRIAC – Triode for Alternating Current*
- *TV – Television*
- *UART – Universal Asynchronous Receiver-Transmitter*
- *USB – Universal Serial Bus*
- *WROOM – Wi-Fi/Bluetooth Module Series*
- *ZC – Zero Crossing*

1 Introduction

1.1 Background

In recent years, the smart home concept has been getting lots of attention, and advances have been made in the Internet of Things (IoT), artificial intelligence (AI), and embedded systems. Smart home automation systems provide centralized control over various home functions such as lighting, heating, and security to improve comfort and energy efficiency. As modern homes integrate more connected devices, the demand for systems that simplify and automate daily tasks like turning on and off lights, and make living environments more comfortable and efficient is increasing.

Smart home automation aligns well with mechatronics principles, as it requires the integration of sensors, actuators, control systems, and programming to create an interconnected network of devices. By using a microcontroller, such as Arduino, Raspberry Pi, or ESP32, along with various sensors and actuators, users can control and monitor household elements remotely, leading to more efficient and user-friendly smart home systems.

Early home automation systems began in the early 1950s by automating and simplifying everyday tasks, from washing clothes by hand to washing machines using mechanical and electrical methods, such as an engine, motor, and gears. Step by step, through various innovations, advanced technologies are doing our basic tasks. Using these technologies, we spend more time focusing on critical and valuable things like having a great time with family and friends, and learning something new.

In Mongolia, we are in the early stage of implementing automated home systems due to the dominance of livestock farming, the majority of the citizens living in the Ger district, the lack of equipment, and the high equipment cost. As devices become cheaper and wireless infrastructure improves, even developing countries like Mongolia are beginning to use smart home applications.

1.2 Aim and Objective of Thesis

Aim:

Design and implement a smart home automation system using mechatronics principles, focusing on low-cost and energy-efficient equipment control (ESP32 or Arduino) of house lighting systems and real-time monitoring by sensors and actuators via wireless control that suits a double room apartment in Mongolia.

Objectives:

To reach this aim, the thesis work will accomplish the following specific objectives:

1. Identify suitable sensors and microcontrollers to automate basic functions of the apartment/home, such as turning on and off lights, heating/cooling, and an intelligent metering system.
2. Design and develop a prototype for an automation system in a double-room apartment that enables remote control and monitoring of selected appliances.
3. Test the system's performance under typical conditions to evaluate data from energy efficiency, sensors, and actuators' potential and reliability.
4. Analyze the data for the equipment's advantages and limitations.
5. Propose improvements and recommendations for expanding the system in local environments and future developments.

Hypothesis questions:

1. What are the most suitable and cost-effective components and equipment for implementing a mechatronics smart home automation system?
2. Does the system control home equipment with an IoT or web app?
3. What is the effectiveness of sensor and remote control-based automation?
4. How can the developed prototype be improved or extended for real-world home automation?

2 Literature Review

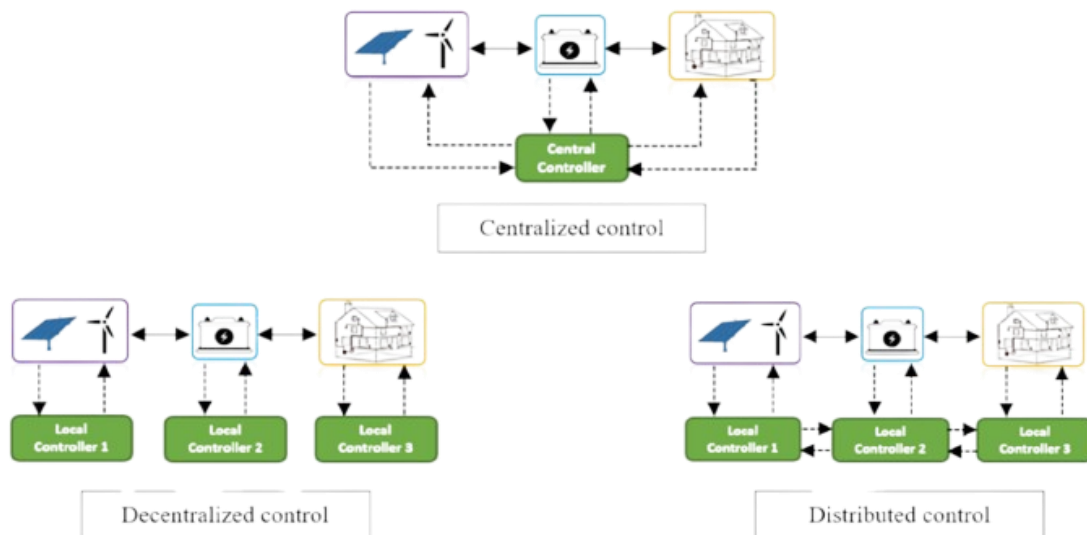
2.1 Mechatronics in Home Automation

Mechatronics engineering includes mechanical engineering, electronics, computer science, and control engineering to create intelligent systems that can make automatic decisions. In smart homes, mechatronic systems are essential to automation, integrating sensors, actuators, controllers (microcontrollers), and communication modules to achieve responsive and intelligent environments. This combined system allows the systems to measure environmental changes (temperature, motion, gas), uses measured information to process logical decisions and its algorithms, and trigger actions through mechanical outputs like control relays or motors. In this section, I will give detailed information about how smart home automation is implemented using different methods, depending on the Literature review.

2.2 System Architectures in Smart Homes

As far as I understand, the architecture of smart home automation systems in industries and individual projects is classified into three sections: centralized, decentralized, and hybrid models. (1)

Figure 1. Centralized, decentralized and distributed control



Centralized architecture: involves a single controller module(ESP32/RPI) that manages all sensors and actuators. However, if it is controlled by one microcontroller or a device, the whole system needs to be shut down to fix one edge. Also, wiring and connections increase complexity and cost more than other systems. In addition, it can be complex to regulate devices like a fan.

Decentralized architecture: It distributes all tasks across multiple microcontrollers to improve reliability. In the event of automation failure, the system switches can continue to function independently, ensuring that basic operations remain unaffected. This system allows a combination of switches, keypads, and sockets in a single modular assembly, and regulators are typically included in decentralized systems.

The hybrid model architecture(Distributed) combines centralized and decentralized approaches and leverages their strengths. It also offers flexibility in designing and deploying components, allowing organizations to tailor automation solutions to their specific requirements. This architecture model is cost-effective, depending on which equipment is used. In addition, it is known as edge computing for real-time tasks and cloud platforms for data storage. This approach increases performance and supports the expansion of the system.

2.3 Communication Technologies and Protocols

Smart home automation includes various methods for remotely controlling appliances, but reliable communication is needed. These technologies are protocols or standards with different features, benefits, and drawbacks (2). The most common protocols include Wi-Fi, Thread, Zigbee, Z-wave, and BLE.

Wi-Fi: The most popular technology for connecting smart devices, it uses the same wireless network that offers high-speed data transfer among our smart devices. Wi-Fi operates on the 2.4 GHz and 5 GHz frequency bands and can support data rates up to several gigabits per second.

Pros	Cons
Easy to set up and use, with existing infrastructure and standards	Limited range and coverage, especially in large or multi-story buildings
Supports cloud and remote access control	Prone to interference and congestion from other wireless devices

Thread: This low-power, mesh-based smart home protocol uses the same frequency band as a wireless network but creates a separate and dedicated low-power mesh network. It does not define an application layer, so manufacturers need to develop and implement their systems.

Pros	Cons

Battery-operated devices can last longer due to their low power consumption.	Limited availability and adoption of devices and products
It supports many devices, each acting as a router and extending the network range.	Low data rate and bandwidth, unsuitable for streaming video and audio.

Zigbee: The oldest and most established smart home protocol, used by popular brands including Ikea and Philips. Its working principle is like a Thread, but uses a different modulation and coding scheme. Zigbee devices can form a network of hundreds of nodes that can communicate with each other or with a central hub or coordinator. Some smart home platforms support it, including Samsung SmartThings, Tuya, and Amazon Echo.

Pros	Cons
Supports various devices, including sensors, switches, lights, and more.	Not suitable for high-speed data transmission
Mesh networking combined with high output power and sensitivity.	Installation cost is high, depending on its size and the range of the network.

Z-Wave: Low-power, mesh network protocol that operates on the 800- 900MHz frequency band, widely used in smart home devices. It's designed to be reliable, scalable, and secure. Also, it uses encryption and device authentication to prevent unauthorized access and hacking. Z-Wave network can handle up to 232 devices, including the primary controller.

Application: Smart sensors like motion, temperature, and humidity. Also, smart switches and locks.

Pros	Cons
Avoids interference and congestion from other Wi-Fi devices(868MHz - 908MHz)	The initial cost is high, it requires a licensed chip and a dedicated hub or controller.
Offers a high level of security and encryption, 128-bit AES	Maximum data rate of 100kbps, which is not suitable for streaming audio and video

BLE: Common and familiar technology used for connecting devices to phones. It uses the same frequency band as Wi-Fi, but has a shorter range and lower data rate. BLE devices can form a network of up to seven devices. It can be used in locks and lighting that use the phone for in-home control.

Pros: Easy to set up and use, only need to connect a smartphone to the smart devices

Cons: It is not suitable for larger smart home applications.

Use cases: Wireless headphones, speakers, keyboards, and game controllers.

Property	Wi-Fi	Zigbee	Z-Wave	Bluetooth
Market focus	Smart Home	Smart home, Metering	Smart home and Home security	Lighting, Locks
Frequency bands	2.4 & 5 GHz	2.4 GHz	Sub-GHz	2.4 GHz
Range	Beyond Front door	Beyond front door	Beyond fence	In Home
Native IPv6	Yes	No	No	No
Cloud	Router	Gateway	Gateway	Gateway, Phone
Application layer	No	Yes	Yes	Yes
Existing infrastructure	Everywhere	Medium-Large	Medium	Large
Ecosystems support	All	Amazon, IKEA, Tuya	Alarm.com, Ring, Samsung	Amazon, Apple, Google, Signify
Mesh networking	Yes	Yes	Yes	NA

Table 1. Comparison of Communication Technologies and Protocols

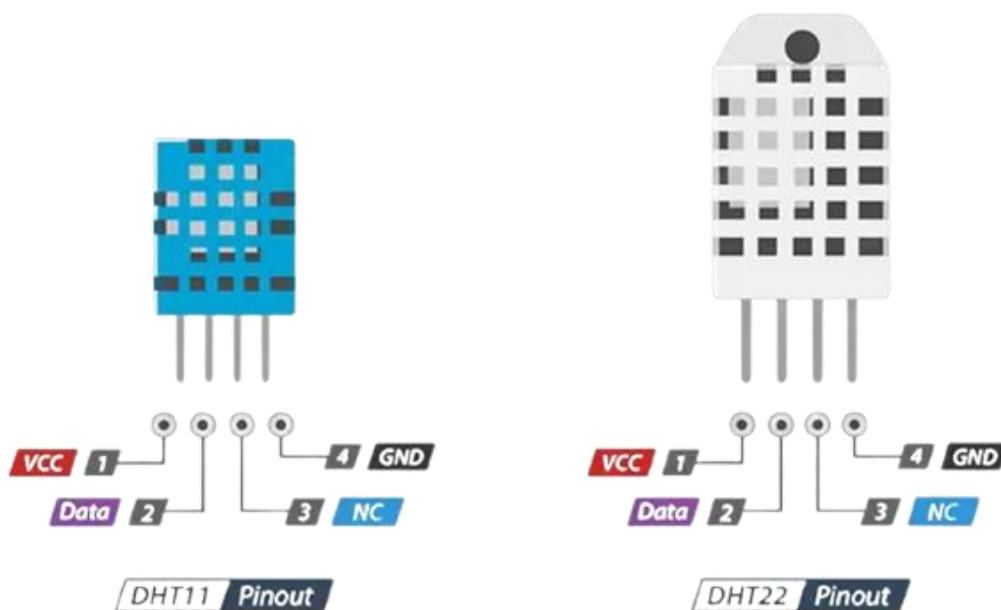
2.4 Sensors

A sensor is a device that detects and responds to environmental changes by converting analog input into measurable electrical signals. In home automation, choosing the right sensor is a key part of the building's logical functions, which help monitor real-time data and activate actuators.

Temperature and humidity sensors(DHT11,DHT22)

DHT11 and DHT22 are digital sensors that measure both temperature and relative humidity. Internally, these sensors consist of a thermistor for temperature sensing and a capacitive humidity sensor. A small integrated chip converts analog signals into calibrated digital signals sent to a microcontroller via a single-wire serial interface. (3)

Figure 2. DHT11, DHT22 Pinouts



DHT11:

- Ultra-low cost
- 3 to 5V power and I/O
- 2.5mA max current use
- 20-80% humidity readings with 5% accuracy
- 0-50°C temperature readings $\pm 2^\circ\text{C}$ accuracy
- 1 Hz sampling rate (once every second)

DHT22:

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use
- 0-100% humidity readings with 2-5% accuracy
- -40 to 80°C temperature readings $\pm 0.5^\circ\text{C}$ accuracy
- 0.5 Hz sampling rate (once every 2 seconds) (3)

Gas and smoke sensors(MQ-Series)



Figure 3. MQ Series sensors

MQ-series gas sensors like MQ-2 and MQ-135 detect gases through a tin dioxide (SnO_2) sensitive layer whose resistance changes based on gas concentration. The MQ-2 detects flammable gases like LPG, methane, and smoke, while the MQ-135 detects harmful air pollutants such as ammonia and CO_2 . Different MQ series sensors exist, including MQ 2 to 9, MQ131, MQ135 to 138. They have a variety of unique specifications. The MQ-7 sensor has been used in indoor applications to review the articles. MQ-7 is known for detecting carbon monoxide gas in the atmosphere. For specifications: (4)

When measuring these gases, the amount of gas per volume in the air is used to calculate their concentration. The most widely used measuring units for this are % concentration and parts-per-million. For example, 1000 molecules of CO gas in a million

air molecules corresponds to a 1000 parts-per-million concentration of CO gas in the atmosphere.

Action detector sensors (PIR, LDR, KY-037)

Action detector sensors are essential for enabling responsive smart home systems that react to human presence, changes in ambient light, or sound activity. These sensors serve as event triggers that inform the microcontroller (ESP32, Arduino) when to activate or deactivate lighting, alarms, or environmental systems. Among the most commonly used in such systems are PIR (Passive Infrared) sensors, LDR (Light-Dependent Resistors), and KY-037 (sound sensors).

Figure 4. KY-083, PIR, LDR sensors



The **KY-083** is a sound-activated module that uses a condenser microphone to detect audio signals such as claps, knocks, or loud noises. It amplifies the signal and compares it to a threshold using an onboard comparator, producing:

- Analog output (A0): Represents sound wave amplitude
- Digital output (D0): HIGH if sound exceeds a set threshold

The sensitivity can be adjusted using a small potentiometer.

A **PIR sensor** detects infrared radiation emitted by warm bodies in its field of view. It consists of:

- A pyroelectric sensor, which senses differences in infrared radiation
- A Fresnel lens, which focuses IR from a wide area
- A signal amplifier and comparator

When a warm object enters or moves across the detection zone, the sensor outputs a digital HIGH signal to the microcontroller. This output lasts for a configurable time (via onboard potentiometers), after which it returns to LOW.

An **LDR** is a variable resistor whose resistance decreases with increasing light intensity. It forms a voltage divider with another resistor, allowing the analog voltage output to be read by the ESP32's ADC (Analog-to-Digital Converter).

- In dark conditions, resistance is high→ output voltage is low.
- In bright conditions, resistance is low→ output voltage is high.

Energy metering sensors and devices

Energy metering is critical in smart home systems, providing continuous, real-time measurement of electrical current drawn by appliances and circuits. This data underpins advanced automation features such as load scheduling, demand response, and peak-shaving and empowers homeowners with actionable insights into consumption patterns, ultimately reducing energy waste and utility costs. In the present system, energy metering is primarily implemented using the ACS712 Hall-effect current sensor. The design can be extended for future scalability with smart metering outlets or non-invasive current transformers (CTs) to monitor entire branch circuits without interrupting wiring.

The ACS712 sensor integrates three key elements:

- A copper conduction path, which carries the current to be measured.
- An embedded Hall-effect transducer senses the magnetic field generated by that current.
- An internal signal amplifier converts the transducer's tiny voltage change into a larger analog output voltage proportional to the current.

At no load, the sensor's output rests at mid-supply voltage. As current flows through the copper path, the changing magnetic field causes the Hall sensor to shift its production above or below that midpoint in direct proportion to the current magnitude and direction. Because the ACS712 responds to AC and DC currents, the firmware samples its output rapidly, computes an average of the absolute readings, and then multiplies by the known line voltage to estimate real-time power consumption. Accumulating those power values over time provides the total energy used.

A short calibration routine is performed at startup to capture the no-load (zero-current) output voltage. Subsequent measurements subtract this offset to accurately represent load current. A simple moving-average filter in software or a small RC network in hardware helps smooth out noise from switching loads or electromagnetic interference.

By combining precise Hall-effect sensing, careful calibration, and robust filtering, the ACS712 module delivers reliable current measurements that enable immediate control actions, such as shedding non-critical loads during peak demand, and long-term analytics to optimize household energy usage.

2.5 Actuators

An actuator is a machine component that produces force, torque, or displacement when an electrical, pneumatic, or hydraulic input is supplied to it in a system. [5] In a home automation system, actuators are the main element activating motions, turning on and off motors and pumps to perform tasks like automated curtain and sensor-based sinks.

Relay switching

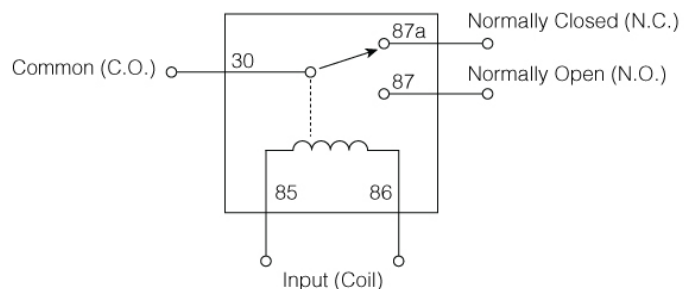
A relay is an electrically operated switch consisting of input terminals for a single or multiple control signals and operating contact terminals. Relays are the primary interface between digital control systems and AC-powered home automation appliances like lights, fans, or sockets.

The core of a relay consists of an electromagnet (coil), a movable armature, a spring, and electrical contacts. When a low voltage (typically 3.3V or 5V) is applied to the relay coil, a magnetic field is generated, pulling the armature and changing the state of the contacts. This action allows the relay to open or close a high-voltage AC circuit without direct electrical contact between the control circuit and the load. Once the input signal is removed, the spring returns the contacts to their original position.

There are two standard configurations:

- Normally Open (NO): Circuit is open until the relay is activated.
- Normally Closed (NC): Circuit is closed until the relay is activated.

Figure 5. Relay and Relay schematic



TRIACs in Smart Home Automation

A TRIAC (Triode for Alternating Current) is a three-terminal semiconductor device that controls AC loads' power. Unlike relays that provide binary switching (ON/OFF), TRIACs allow phase-angle control, enabling gradual dimming or power regulation in appliances such as lights, heaters, and fans.

When triggered, the TRIAC is a bidirectional device that can conduct current in both halves of the AC cycle. It has three terminals: MT1 (Main Terminal 1), MT2 (Main Terminal 2), and Gate (G). When a small gate current is applied, the TRIAC switches on, allowing current to flow between MT1 and MT2. It stays on until the current naturally drops to zero. Only part of the cycle is conducted by delaying the gate trigger after the AC waveform's zero-crossing (ZC) point, thereby reducing the average voltage delivered to the load. This control method is known as phase-angle control and is commonly used for dimming incandescent lights, controlling heating elements, and controlling variable-speed motors.

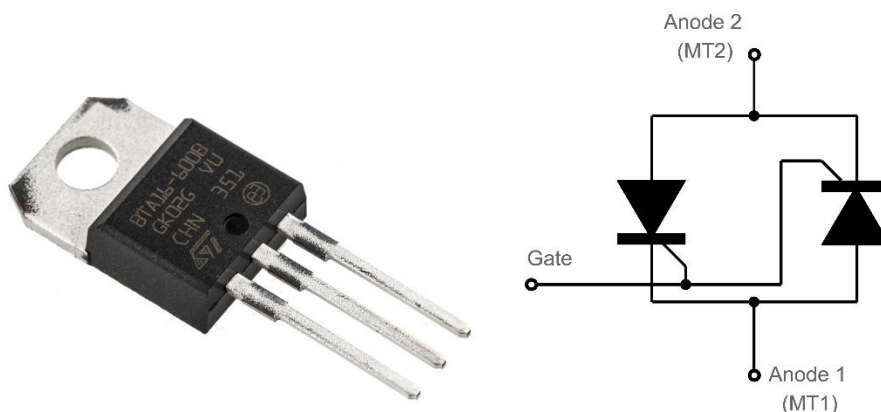


Figure 6. Triacs

To implement precise dimming, a Zero-Cross Detection (ZCD) circuit is used to identify the point at which the AC signal crosses 0V. Once detected, the microcontroller starts a timer and triggers the TRIAC gate after a specific delay. The longer the delay, the shorter the conduction time; hence, the dimmer the light.

Motors and Pumps

Motors and pumps are electromechanical actuators that convert electrical energy into mechanical motion. In smart home automation, they perform tasks that require movement, such as opening and closing curtains, controlling ventilation, regulating valves, or dispensing water. Their microcontroller integration enables automation based on environmental inputs, schedules, or user commands. The home automation system uses Various motors and pumps, including servo motors, DC motors,

Servo motors are widely used in smart homes for precise angular movement, such as opening window blinds, adjusting solar panels, or rotating security cameras. A servo motor consists of a DC motor, gearbox, position sensor, and control circuitry in a single compact unit. Servo motors operate based on Pulse Width Modulation (PWM). The motor receives a control signal, and the width of this pulse determines the rotation angle.

DC motors - provide continuous rotation and are commonly used where speed control is essential, such as in automated curtains, fans, or adjustable louvers. A DC motor uses a magnetic field and a current-carrying armature to produce rotational force. The direction of rotation can be controlled by reversing the polarity of the voltage applied. Speed can be adjusted using PWM signals from a microcontroller. A motor driver IC, such as L298N or L293D, is typically used to control direction and speed. These drivers act as an interface between the microcontroller and the motor, allowing bidirectional control.

2.6 Integration Platforms and Control Interfaces

The main benefit of any smart home system is the interface through which users interact and control various automated functions. Integration platforms bridge hardware and human input/output channels. They enable real-time monitoring, remote control, and even voice-based interaction with home appliances. This section explores modern smart home automation's most common and effective control interfaces: mobile/web dashboards and voice assistants with feedback systems.

Mobile applications and Web dashboards-

Mobile applications and web-based dashboards provide intuitive graphical user interfaces (GUIs) for controlling smart home devices. These interfaces allow users to monitor sensor data of temperature, humidity, and power usage and execute commands such as turning ON/OFF lights, adjusting dimming, and scheduling automation tasks. They are often built using platforms such as Node-RED, MIT App Inventor, Blynk, or Adafruit IO and communicate with the microcontroller via protocols like MQTT, HTTP, or WebSockets. (6)

Node-RED is a flow-based development tool that runs on local devices like Raspberry Pi or the cloud. It provides widgets like charts, buttons, and sliders that can be arranged on a dashboard and linked to backend logic using a drag-and-drop interface. Node-RED supports MQTT, HTTP REST APIs, and WebSocket protocols, making it ideal for real-time interaction with IoT devices.

Applications made in MIT App Inventor are Android-based and allow users to build simple interfaces for toggling switches, setting timers, or displaying sensor values. Blynk offers a polished environment with pre-made widgets and built-in support for ESP32/ESP8266. The mobile app connects to the Blynk cloud and syncs with the microcontroller over Wi-Fi, enabling real-time device control from anywhere.

Figure 7. IoT based Interface programs



Voice assistants and user feedback systems –

Voice-controlled interfaces add convenience and accessibility by allowing users to operate home systems hands-free. Integrating voice assistants such as Amazon Alexa, Google Assistant, or Apple Siri has become a defining feature of modern smart homes.

Working Principle

Services such as Sinric Pro, IFTTT, or SmartThings are used to connect voice assistants to custom-built systems like ESP32. These services bridge cloud-based voice platforms and user-defined IoT endpoints. (6) For example:

- A user says, "Alexa, turn the living room light on."
- The command is sent to Sinric Pro, which forwards it to the ESP32 over HTTP or WebSocket.
- The ESP32 receives the command and toggles the connected relay or dimming circuit.

This allows voice interaction with devices incompatible with commercial smart home ecosystems.

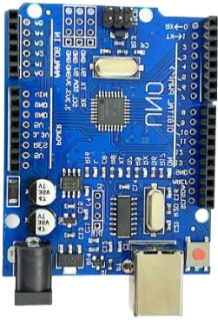


User Feedback Mechanisms - Besides control, smart home interfaces provide feedback to inform the user of system status. Feedback can be:

- Visual: LED indicators, LCD/OLED screens, mobile app notifications, or dashboard updates
- Auditory: Buzzer alarms, voice confirmations like „The light turned on“
- Digital: App pop-ups, browser messages, email/SMS alerts

Feedback mechanisms ensure system transparency and help users verify that commands have been executed correctly.

2.7 Microcontrollers

Choosing the right microcontroller significantly impacts a smart home system’s performance, energy use, growth potential, and ease of connection. Three commonly used platforms are the Arduino Uno, the Raspberry Pi, and the ESP32, each fitting different project needs. The Arduino Uno is great for basic tasks like reading a few sensors or switching relays, simple input/output, and low power draw. The Raspberry Pi runs a complete Linux operating system, making it suitable for hosting a local dashboard, handling complex data processing, or running multiple services. However, it uses more power and isn’t as fast or as precise as timing. The ESP32 offers a middle ground: built-in Wi-Fi and Bluetooth, good processing power for real-time control, and sleep modes for low-power operation to handle automation logic and wireless communication in one board. Matching your choice to what your system needs—simple switching, advanced computing, or always-on connectivity—helps ensure a reliable and efficient smart home setup. (7)

	Arduino Uno	Raspberry Pi	ESP32
Feature			
CPU	8-bit	64-bit	32-bit
Clock speed	16MHz	1.5-GHz	240MHz
Wireless connectivity	External component required	Wi-Fi/Bluetooth	Wi-Fi/Bluetooth

OS Support	None	Linux-Based OS	RTOS
GPIOs	14 digital I/O	26+	30-40(varies)
Power consumption	Very low	High	Low
Ideal use	Simple automation	Automation hub, dashboards	Smart devices, IoT integration

Table 2. Specifications of Microcontrollers

2.8 Industrial solutions for Home automation in Mongolia

Moncable LLC –

Moncable LLC is a prominent Mongolian company that provides comprehensive smart home and building automation solutions. The company offers fully integrated Control4-based smart automation systems, which allow users to automate and manage their homes using advanced, lifestyle-adaptive technologies. Their solutions are designed to enhance convenience, energy efficiency, and personalized living by offering multiple modes of interaction and control. (8)

The proposed system from Moncable supports up to six control modes, tailored to user preferences and living patterns:

- Switch Control – Manual control through smart or wall-mounted switches
- Smart Scene – Pre-programmed actions that adjust multiple devices simultaneously ("Movie Mode" dims lights, closes curtains, and powers on TV)
- Timing Control – Scheduled operation of devices such as lights, curtains, or HVAC systems
- Curtain Control – Motorized and automated control of window treatments based on time, light, or motion
- Dimming & Tunable White – Adjustable brightness and color temperature of lighting fixtures
- Constant Illumination Control – Automated adjustment of light output based on ambient lighting conditions to maintain consistent illumination

Digital Power LLC -

Digital Power LLC collaborates with over 30 internationally renowned brands, including Siemens, Schneider Electric, Bosch, Johnson Controls, Fibaro, Crestron, Suprema, Securiton, and Axis Communications. Through these partnerships, the

company delivers certified, reliable, and scalable innovative automation systems that comply with international standards. Their solutions are tailored for both residential and commercial applications and are known for their focus on energy efficiency, centralized control, and user-friendly interfaces. (9)

In the smart home automation, Digital Power delivers comprehensive automation solutions that include:

- Smart lighting and environmental control
- HVAC automation using thermostats and sensors
- Access control and video surveillance systems
- Energy metering and remote management dashboards

3 Methodology

3.1 System Overview

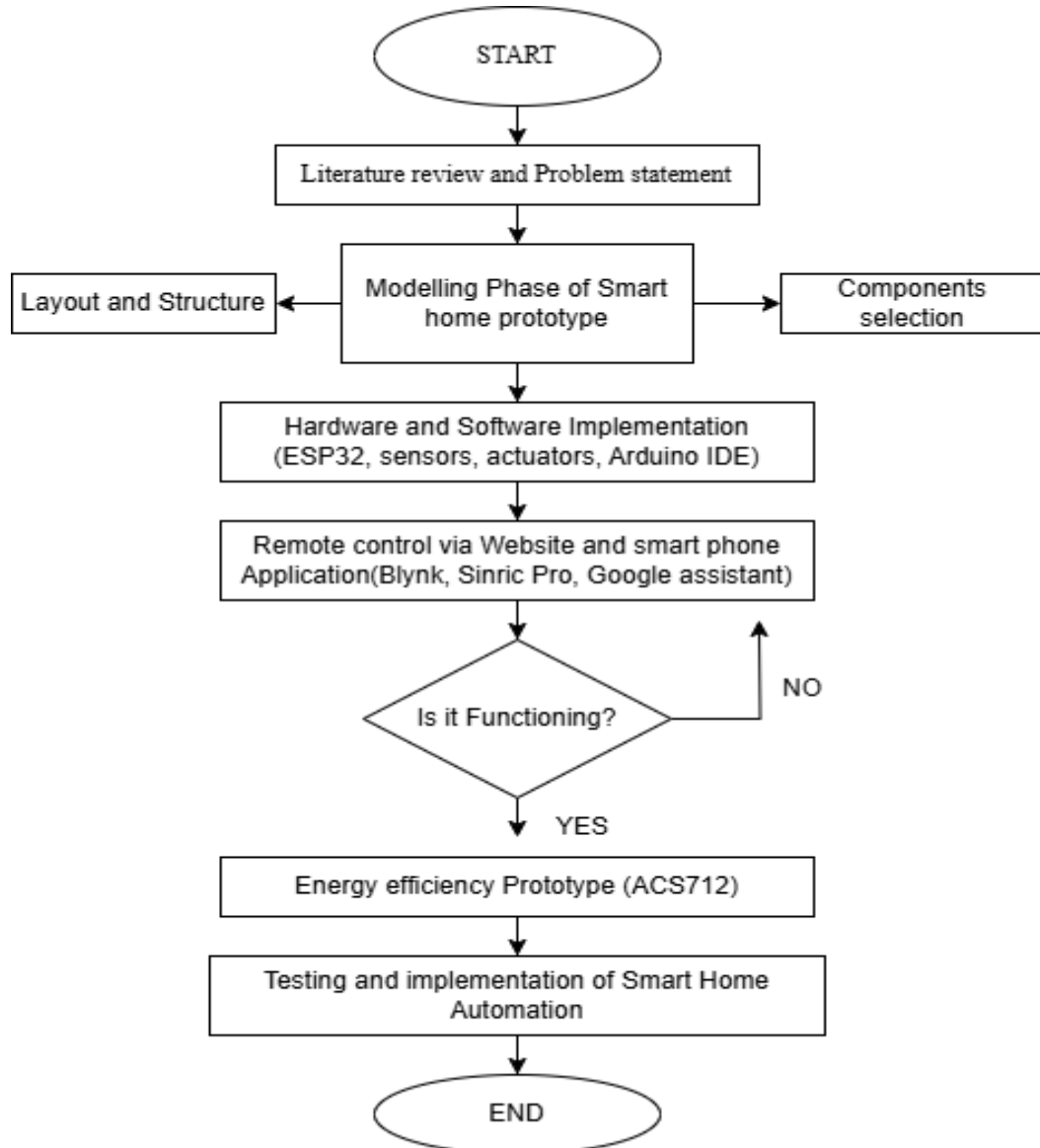


Figure 8. Flowchart of the Smart home automation prototype

The proposed smart home automation system is a modular, sensor-based mechatronic system that controls lighting and appliances. The system uses sensors, actuators, and a microcontroller (ESP32) to collect environmental data, make decisions based on predefined logic, and perform automated actions accordingly. The project focuses on providing an energy-efficient, user-friendly, and remotely accessible home automation solution for implementing a small-scale house model (or double-room apartment).

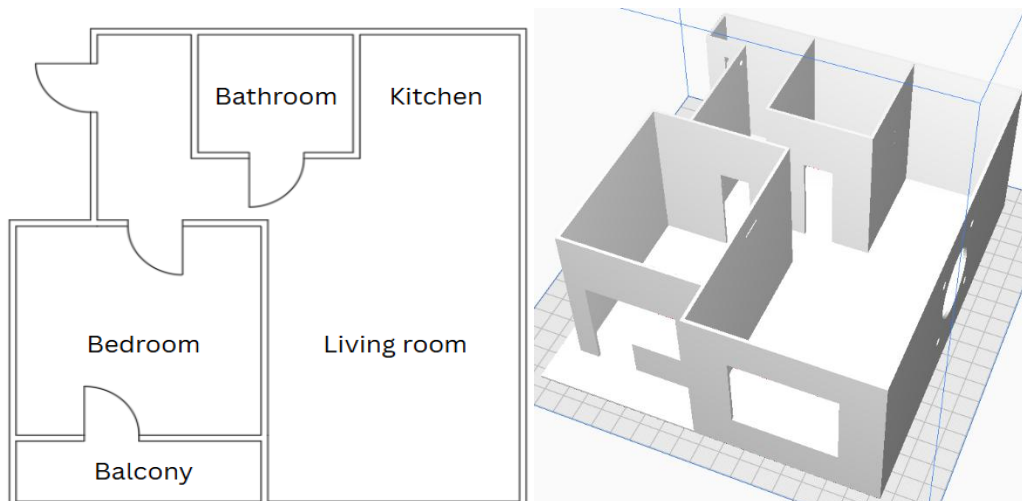


Figure 9. Plan and design of the

For the physical layout and spatial reference of the smart home prototype, the double-room layout from Khangai Residence was used as a foundational model. This model was selected due to its realistic representation of the average Mongolian apartment, making it an appropriate and practical reference for implementing automation features in a typical local living environment. The architectural and spatial configuration closely reflects how most urban households are organized in Mongolia. Based on this reference, a 3D design was developed using Autodesk Fusion 360, a professional CAD tool that supports detailed mechanical and architectural modeling. The design includes major living spaces, lighting zones, curtain placements, and sensor locations, which were used to guide the intelligent automation system's hardware layout and functional testing.

3.2 Selection of Sensors and Microcontroller

No	Component name	Definition	Function in the System
1	ESP32-WROOM(DEVKIT)	Microcontroller	central control unit for reading sensors, executing logic, and controlling outputs.
2	DHT22	Temperature and Humidity sensor	Measures ambient temperature and relative humidity; data is used for environmental control
3	MQ-7	Carbon Monoxide sensor	Detects CO levels in the air to provide safety alerts in case of harmful gas buildup

4	ACS712	Current sensor	Monitors AC Consumption; used for energy metering and overload detection.
5	TT motor	3-9V DC motor	Provides mechanical movement for automated systems such as curtains, doors, or fans.
6	L298N(ZX-040)	Motor driver for ESP32	Controls the speed and direction of DC motors; interfaces with ESP32 for movement automation.
7	Mod-DimAC-8A-2L	AC light dimmer 2-channel module	Adjusts the brightness of AC light bulbs via phase-angle control; enables smooth dimming.
8	HC12S5015H	12V DC fan	Circulates air for temperature regulation; can be triggered based on sensor input.
9	Buzzer	Small piezoelectric alarm buzzer	Emits sound for alerts or feedback (gas detection, motion alerts)
10	P2H1588A0	4-channel relay module (3.3V-5V)	Switches AC devices (lights, fans) on/off based on ESP32 signals; opto-isolated for safety
11	Push Button	Tactile button	Provides manual input for toggling or overriding automated control
12	LDR	Light sensor	Measures ambient light intensity to automate lighting based on time of day or brightness.
13	HC-SR501	Motion Sensor	Detects movement in the environment; used for triggering lights or alarms.
14	KY-037	Sound-Sensor	Detects sound levels (claps, loud noise); used for event-triggered automation.

15	FSL E27 100W	Incandescent light bulb	It provides illumination and is used as a load for automation and dimming demonstrations.
16	Light adaptor	Light Bulb Socket	Connects the light bulb to power and allows control via a relay or dimmer module.

Table 3. Planned components and micro elements

3.3 Software Architecture

The smart home automation system is organized into three interconnected layers, each responsible for key functions that together enable responsive, energy-efficient control of household devices:

1) Sensing Layer- This bottom layer interfaces directly with the physical environment.

It includes:

- Temperature and humidity Sensors (DHT22) sample ambient climate conditions to inform heating, cooling, or dehumidification actions.
- Sound Sensor (KY-037) detects acoustic events, such as claps or spoken commands, as an alternative trigger instead of motion detection.
- Light Sensor (LDR) measures ambient illumination to drive automatic lighting scenes or dimming.
- Current Sensor (ACS712) monitors real-time electrical draw from appliances, enabling load-based automation and energy metering.

These sensors feed continuous or event-driven data into the control layer at configurable intervals or upon threshold crossings.

2) Control Layer - The ESP32-WROOM microcontroller. Its core responsibilities are to:

- Acquire raw sensor readings via analog and digital I/O.
- Process these readings according to predefined rules, schedules, or user-defined scenes. For example, the system will turn the lights on if the kitchen light sensor reports low lux levels and the sound sensor registers a clap.
- Actuate devices using GPIO outputs driving relays, TRIAC-based dimmers, and motor drivers (L298N).
- Communicate bi-directionally with cloud services (Sinric Pro, Blynk) over Wi-Fi, enabling remote commands and data uploads.

The firmware developed in the Arduino IDE ensures real-time responsiveness, safe shutdown sequences, and reliable reconnection strategies.

- 3) Interface Layer - This top layer provides human-machine interaction channels:
- Mobile/Web Dashboards (Node-RED, Blynk): Users monitor live sensor graphs, toggle devices, and adjust settings from smartphones or browsers.
 - Voice Assistants (Amazon Alexa, Google Assistant via Sinric Pro): The control layer interprets voice commands like “turn off bedroom light” into MQTT or WebSocket messages and executes them.
 - Push Notifications & Alerts: Threshold breaches—such as high CO levels or excessive current draw—trigger instant notifications on the user’s device, ensuring timely intervention.

Firmware

The system firmware is written in C/C++ using the Arduino IDE, which offers seamless support for the ESP32 platform and a vast ecosystem of open-source libraries. This choice allows for rapid prototyping, straightforward debugging via the Serial Monitor, and easy over-the-air updates when needed. The firmware follows the standard Arduino sketch structure—initializing in `setup()` and executing continuous logic in `loop()`—and depends on several specialized libraries to handle networking, sensor interfaces, actuation, and cloud or mobile integration.

Key Libraries used in this prototype:

- WiFi.h – Provides functions to connect the ESP32 to local Wi-Fi networks, manage reconnection logic, and obtain network parameters.
- DHT.h – Interfaces with the DHT22 temperature and humidity sensor, handling precise timing and data parsing to return accurate environmental readings.
- SinricPro.h – Implements the core WebSocket client for Sinric Pro, enabling voice-assistant integration (Alexa/Google) and real-time device control through the cloud.
- Blynk.h – Connects the ESP32 to the Blynk IoT platform, allowing sensor data visualization and remote actuator control via smartphone widgets.
- HTTPClient.h - Facilitates HTTP GET and POST requests, which push sensor readings or retrieve configuration data from RESTful APIs and custom web services.
- RBDdimmerESP32.h – Controls AC dimmer modules by synchronizing with zero-cross detection, managing phase-angle firing of TRIACs to adjust lighting brightness smoothly.

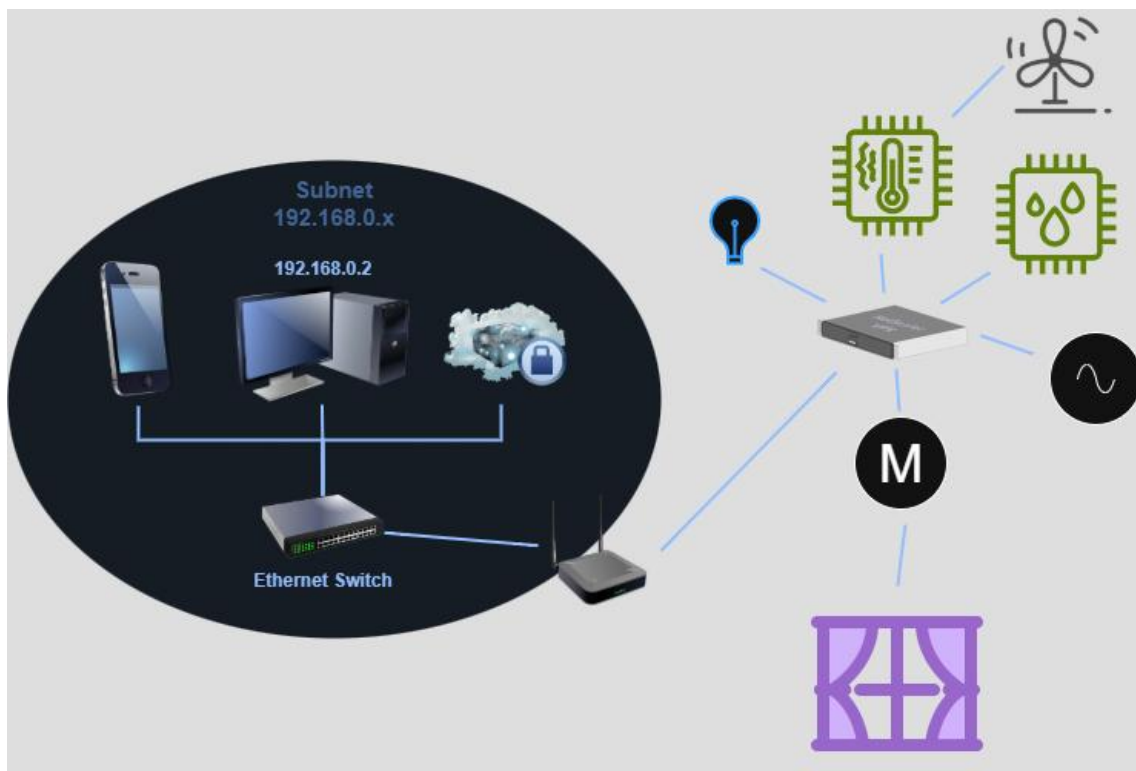
- ACS712.h – Reads raw analog voltages from the ACS712 current sensor, converts them into calibrated amperage values, and supports real-time energy-usage calculations.

Cloud and Mobile Integration

The system connects to cloud-based platforms, primarily Sinric Pro and Blynk, to provide reliable, low-latency communication between the ESP32 and remote services. Through these integrations, users gain the following capabilities:

- Remote Control: Mobile apps or web dashboards allow users to toggle lights, adjust dimming levels, and activate fans from anywhere with internet access. Blynk widgets and custom web interfaces communicate with the ESP32 over secure MQTT or HTTP channels.
- Voice Commands: By registering devices on Sinric Pro, the ESP32 can be controlled via Amazon Alexa or Google Assistant. Voice requests (e.g., “Alexa, turn off the kitchen light”) are relayed through Sinric Pro’s WebSocket API and executed instantly on the hardware.
- Real-Time Monitoring: Sensor readings—such as temperature, humidity, current draw, and sound levels—are streamed live to dashboards hosted on Sinric Pro, Blynk, or Adafruit IO. Users can view graphs, set thresholds, and receive push or email alerts when values exceed safe limits.

Figure 10. System Flowcharts and Algorithms



4 System implementation

4.1 Hardware Setup

Components and equipment (10) (11)

Component	Definition	Pieces	Price(₹)	Total(₹)
ESP32-WROOM	Microcontroller	1	30000	30000
THS-1230B	220V to 12V amplifier	1	13000	13000
Push Button	Tactile button	4	600	2400
5W bulb	Daylight	2	4000	8000
FSL E27 100W	incandescent light	2	3000	6000
Light adaptor	To attach light bulbs	2	3000	6000
P2H1588A0	4 channel relay module	1	10000	10000
HC12S5015H	12V Fan	1	12000	12000
MQ7	Carbon Monoxide sensor	1	5000	5000
Buzzer	Small piezoelectric alarm buzzer	2	800	1600
TT motor	3-9v dc motor	1	3000	3000
L298N(ZX-040)	Motor Driver for ESP32	1	10000	10000
ACS712	Current sensor	3	8000	24000
DHT22	Temperature and humidity sensor	1	15000	15000
LDR	Light sensor	2	800	1600
Mod-DimAC-8A-2L	AC light dimmer 2 channel module	1	45000	45000
Micro USB	USB to connect microcontroller	1	10000	10000
			Total	202600

Table 4. Selected Sensors and Microcontrollers

The hardware architecture centers on the ESP32-WROOM microcontroller, which coordinates all sensing, decision making, and actuation. Connected to the ESP32 are:

Sensors:

- DHT22 for temperature and humidity
- LDR for ambient light level
- KY-037 sound sensor (used in place of a PIR motion sensor)
- MQ-7 carbon monoxide detector
- ACS712 current sensor

Actuators and Output Devices:

- 4-channel relay module (P2H1588A0) to switch lights and fans
- Mod-DimAC-8A-2L AC dimmer module, driven via zero-cross detection and PWM

- L298N motor driver for door/curtain motors and a 3–9 V DC fan
- Piezo buzzer for audible alerts

Because a PIR motion sensor was unavailable, the KY-037 sound sensor was substituted for light control. Rather than detecting human presence via heat signatures, the KY-037 toggles lighting in response to loud noises or hand claps. Although this method is less precise and prone to false triggers from ambient noise, it enabled reliable, basic on/off functionality without additional hardware. The absence of a dedicated motion sensor also limited automated curtain/blind control and security monitoring, features that generally rely on passive infrared for occupancy detection.

Despite these constraints, the system’s core functions were fully realized: lights can be switched and dimmed, fans activate based on temperature thresholds, and sound-based triggers operate as intended. This adaptable hardware setup demonstrated that essential smart-home features can be implemented even when ideal sensors are unavailable.

Circuit Design and Wiring

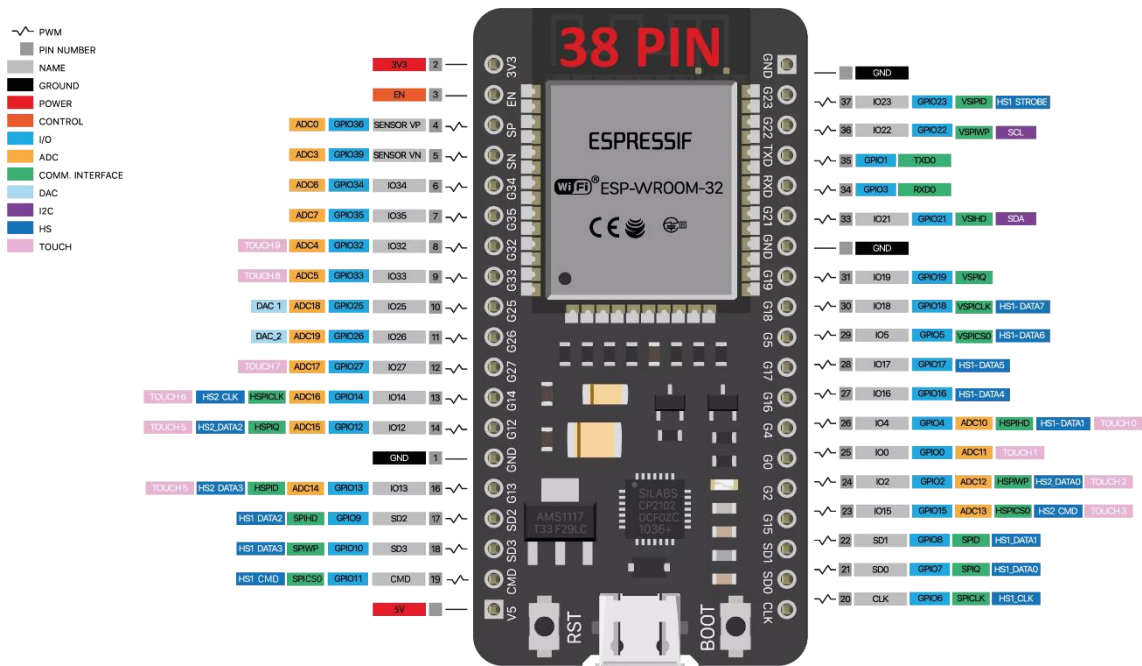


Figure 11. ESP32 WROOM- Pinouts: Detailed information

A comprehensive wiring diagram of the smart home automation system was produced using Fritzing, an open-source electronics design tool that supports breadboard layouts, schematic views, and PCB exports. This diagram served as a reference throughout hardware assembly and troubleshooting. It clearly illustrates how the ESP32-WROOM (DEVKIT1) connects to all sensors and actuators:

ESP32-WROOM (DEVKIT1) pinout highlights:

- 18 ADC channels for analog sensors
- 3 SPI interfaces for high-speed peripherals
- 3 UART interfaces for serial communication
- 2 I2C interfaces for sensor buses
- 16 PWM outputs for dimming and motor control
- 2 DAC channels for analog waveforms
- 2 I2S interfaces for audio or digital microphones
- 10 capacitive-touch GPIOs for touch inputs

Wiring connections shown in Fritzing:

- Sensors: DHT22 (temperature/humidity), KY-037 (sound), LDR (light), MQ-7 (gas), ACS712 (current)
- Actuators: P2H1588A0 4-channel relay module for switching lights and fans; Mod-DimAC-8A-2L AC dimmer wired to zero-cross detection and PWM pins; L298N motor driver for DC motors or curtain actuators
- Alerts: Piezo buzzer for audible alarms

By mapping each component to the appropriate ESP32 pins and power rails, the Fritzing diagram ensured reliable wiring, prevented misconnections, and accelerated the hardware implementation phase..

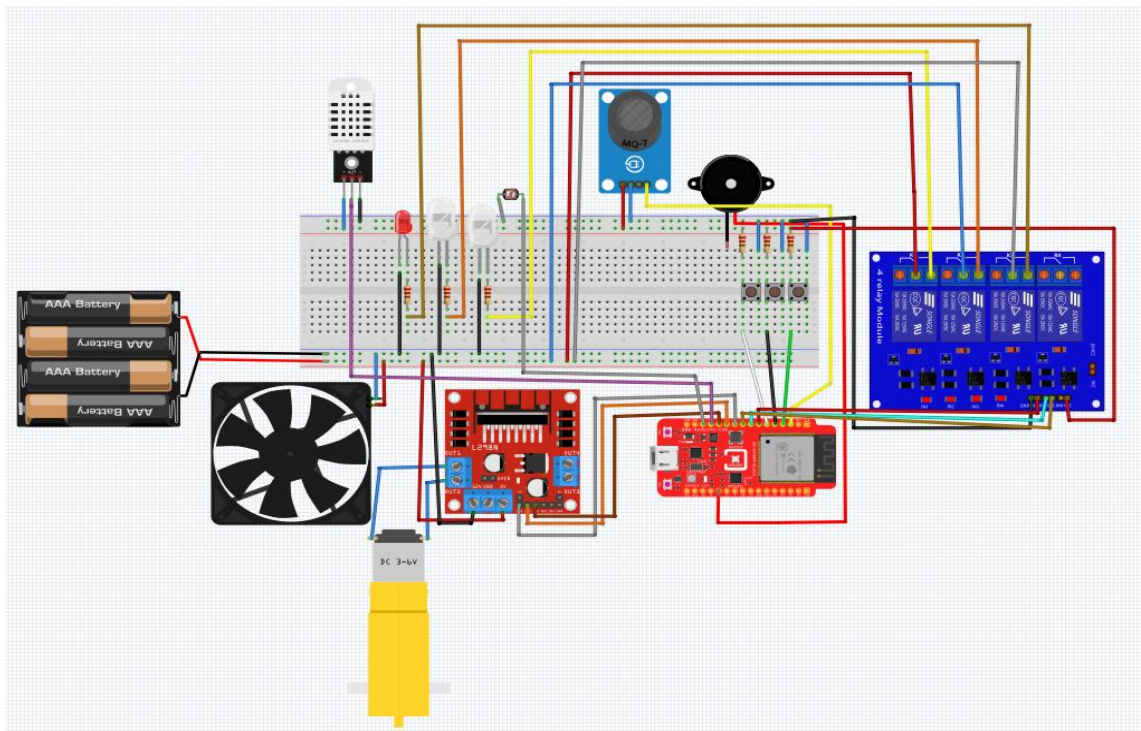


Figure 12. Overall circuit wiring design (FRITZING)

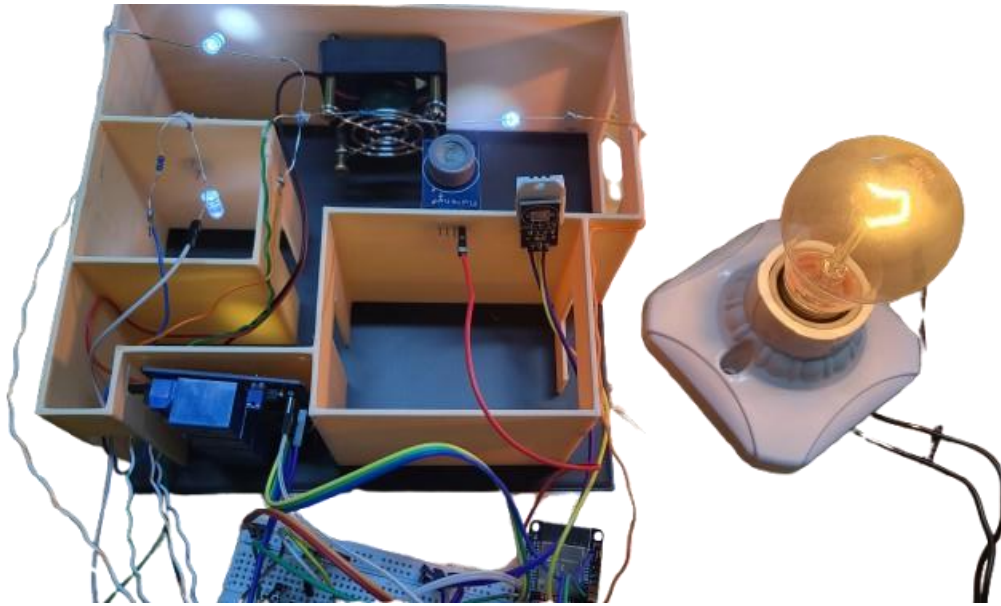
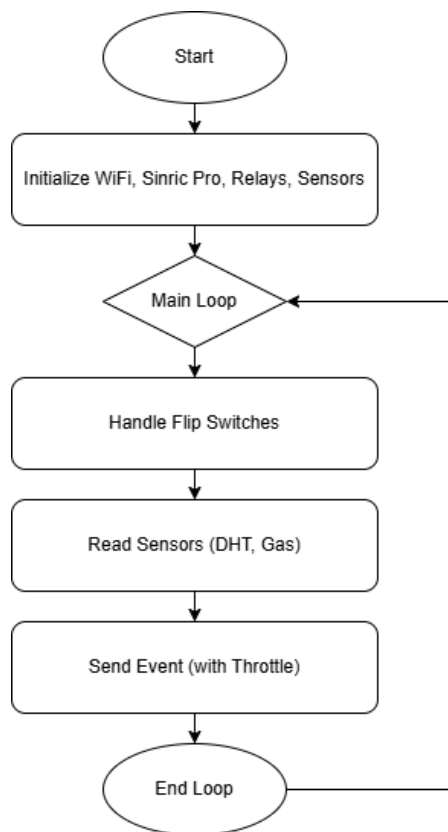


Figure 13. Implemented the Automation prototype

4.2 Arduino IDE coding for Home Automation



The coding phase is an essential component of smart home automation, as it governs the control logic, data acquisition, remote monitoring, and overall system functionality. Through well-structured and optimized code, the microcontroller can interpret sensor inputs, execute automated decisions, manage actuator responses, and facilitate seamless communication with cloud platforms and user interfaces.

The following code was developed and uploaded to the ESP32 microcontroller using the Arduino IDE to collect environmental data from a dormitory kitchen.

```

1  #include <WiFi.h>
2  #include <HTTPClient.h>
3  #include "ACS712.h"
4
5  const char* ssid = "GMIT_WiFi"; // your WiFi SSID
6  const char* password = "gmit1234"; // your WiFi password
7  const char* serverURL = "https://script.google.com/macros/s/AKfycbzG4044zTbvZAz-Vb-veFwfv7r7mbdy4ARej7e0rRFv57z-cFYFVbz5Gvp0x6_c7ccB1/exec"; // your Web App URL
8
9  ACS712 ACS(35, 5, 4095, 100); // ACS712 module parameters
10 int calibration_factor = 320;
11
12 unsigned long lastSendTime = 0;
13 const unsigned long sendInterval = 20000; // send every 30 seconds
14
15 void setup() {
16   Serial.begin(115200);
17
18   WiFi.begin(ssid, password);
19   Serial.print("Connecting to WiFi");
20   while (WiFi.status() != WL_CONNECTED) {
21     delay(500);
22     Serial.print(".");
23   }
24   Serial.println("\nConnected!");
25 }
26
27 void loop() {
28   if (WiFi.status() != WL_CONNECTED) {
29     Serial.println("WiFi not connected");
30     return;
31   }
32
33   if (millis() - lastSendTime >= sendInterval) {
34     float current = readCurrent();
35
36     HTTPClient http;
37     http.begin(serverURL);
38     http.addHeader("Content-Type", "application/json");
39
40     String payload = "{\"current\": " + String(current) + "}";
41     int httpResponseCode = http.POST(payload);
42
43     if (httpResponseCode > 0) {
44       String response = http.getString();
45       Serial.println("Data sent to Google Sheets: " + response);
46     } else {
47       Serial.println("Error sending data. HTTP Response code: " + String(httpResponseCode));
48     }
49     http.end();
50
51     lastSendTime = millis();
52   }
53 }
54
55 float readCurrent() {
56   float average = 0;
57   for (int i = 0; i < 100; i++) {
58     average += ACS.mA_AC();
59   }
60   float mA = (abs(average / 100.0) - calibration_factor);
61   if (mA <= 5) mA = 0;
62   return mA;
63 }

```

Figure 14. Code of Collecting the Default kitchen data

4.3 System Testing and Debugging

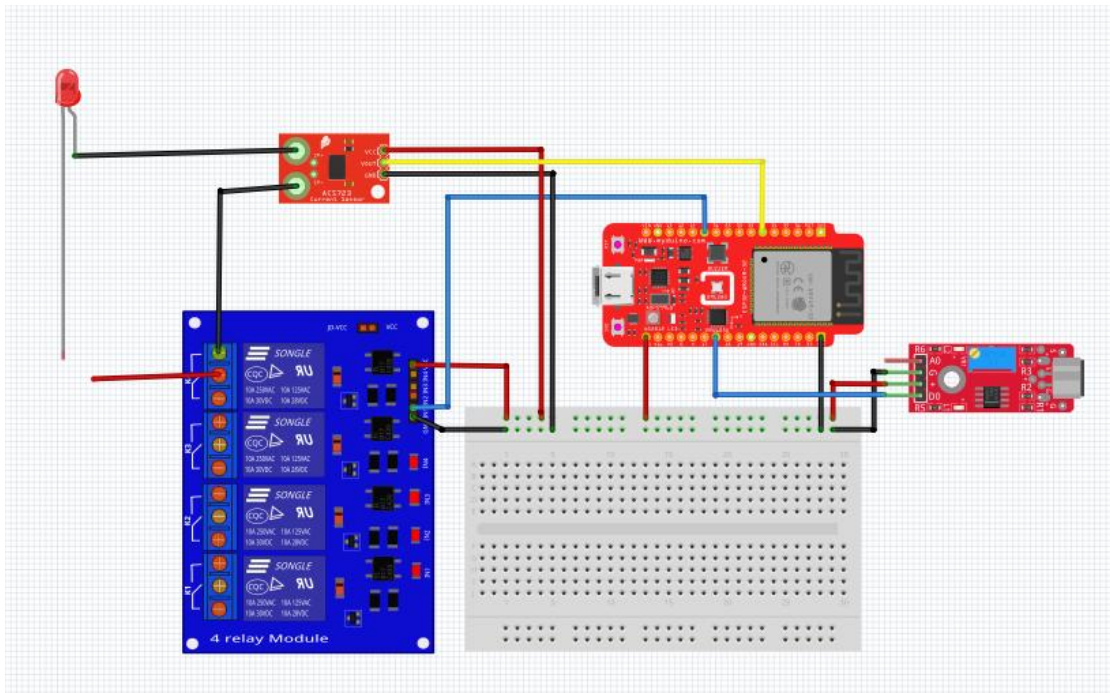


Figure 15. Circuit wiring design of the Automated kitchen

System testing and debugging are essential phases of prototyping and implementation, helping to identify malfunctioning components and buggy code. For testing purposes, I collected data from the kitchen in the second dormitory. The first-floor kitchen was automated using a sound sensor, a relay, and an ACS712 current sensor. According to student observations, the kitchen light tends to remain on regardless of whether a person is present. Based on this observation, I chose the first and second floor kitchens as data collection sites for further analysis and comparison.

As shown in the first figure, the circuit diagram was used to collect electrical usage data in the kitchen. The system is designed to turn the kitchen light on or off when a student claps or produces a loud noise. However, due to a relay malfunction, the light failed to turn off as expected. To address this issue, I implemented a time-based automatic shutdown feature.

The code includes a timer function set to 5 minutes. If the sound sensor detects any noise within this period, the timer resets, allowing the light to remain on. The light automatically turns off if no noise is detected within 5 minutes. Despite this approach, the system still encountered issues due to high electrical load. The relay's contactors would become magnetized and fail to release properly. Through testing, I found that the maximum load applied to the relay is approximately 1 minute to release the contactor.

I used the Google Sheets library for Arduino IDE to log and monitor the collected data. This library sends data via a Wi-Fi-connected microcontroller, enabling real-time data logging and remote monitoring of sensor values and system behavior.

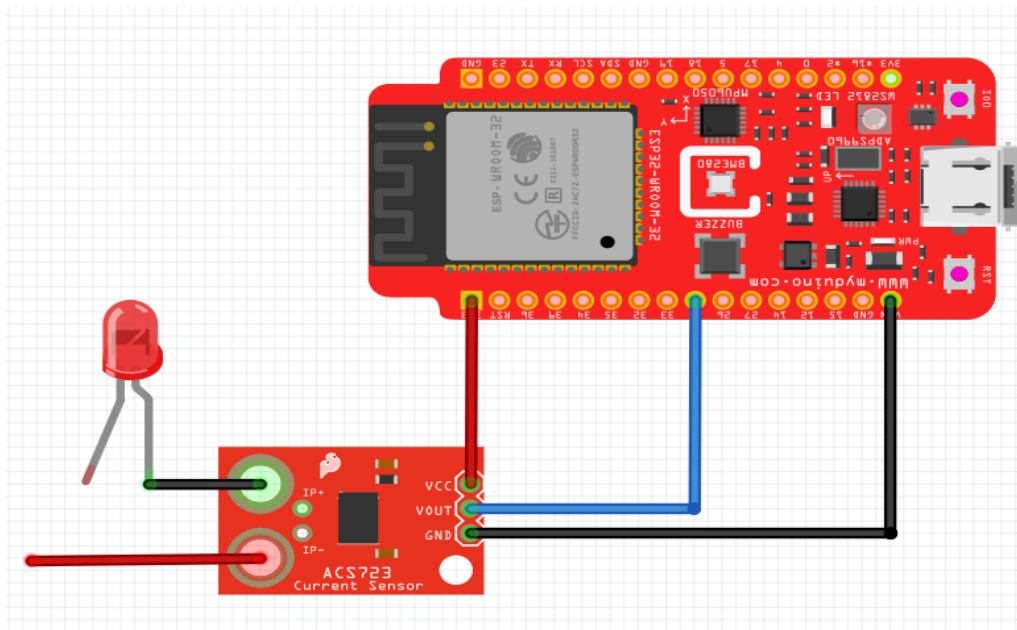


Figure 16. Circuit wiring design of the Default kitchen

I used only an ESP32 microcontroller and an ACS712 current sensor on the second floor to collect electricity consumption data. This setup provides a simple yet effective method to monitor the kitchen lighting system's energy usage. The data collected from this floor serves as a baseline for comparison, specifically to analyze the difference in power consumption before and after implementing the automated lighting system. By comparing both data sets, I aim to evaluate the effectiveness of automation in reducing unnecessary energy usage.

Time	Price per kW(₹) (12)	Default (₹)	Automated(₹)
6 am to 5 pm	265	660.94	300.05
5 pm to 10 pm	397	450.07	204.32
10 pm to 6 am	160	290.22	131.76
	Total(₹)	1401.24	636.13

Table 5. Electricity calculation of 24-hour data

$$\text{Electricity Bill Calculation} = \frac{V * I * t}{1000} * R$$

V = Voltage in Volts

I = Current in Amperes

t = Daily usage time in hours

R = Electricity rate in ₹ per kWh

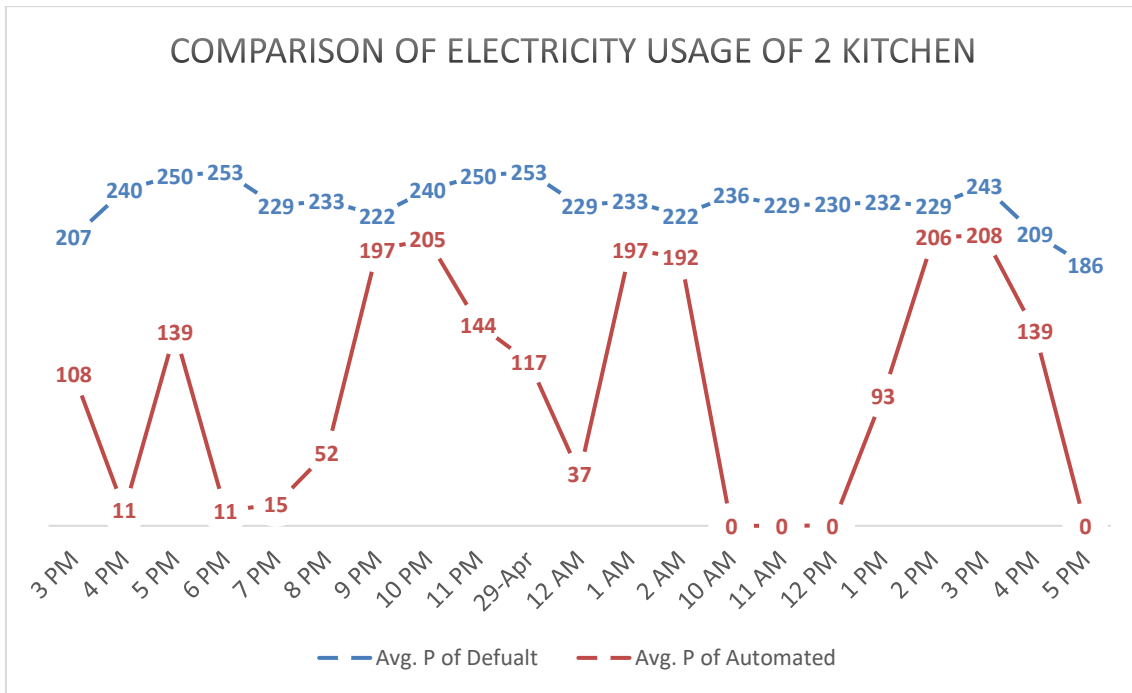


Figure 17. Graph of Default and Automated kitchen consumption

4.4 User Interface and Experience

The user interface was implemented using two major platforms: Blynk and Sinric Pro. Both provide intuitive, drag-and-drop widgets for real-time control and monitoring, but their coding requirements, feature sets, and pricing models differ.

Blynk:

- **Ease of Use:** Very straightforward setup—install the Blynk library, create a project in the mobile app, and assign virtual pins to widgets.
- **Limitations:** The free tier allows up to 30,000 datapoints per month, which initially appears sufficient but quickly becomes restrictive when streaming multiple sensor readings every few seconds. Beyond that limit, real-time monitoring halts unless a premium subscription (\$99 / month) is purchased—an impractical cost for personal or educational projects.

Sinric Pro:

- **Coding Complexity:** It requires more boilerplate in the firmware (WebSocket setup, callback functions) than Blynk’s simple API calls, but it remains manageable within the Arduino framework.
- **Cost Efficiency:** Pricing is \$3 per device a year, making continuous, long-term operation affordable for multiple devices.

- Voice Assistant Integration: Natively connects with Amazon Alexa, Google Assistant, and Samsung SmartThings, allowing users to issue commands like “Alexa, turn on the kitchen light” without additional gateway hardware.
- Automation Scenes: Sinric Pro supports scene creation on its dashboard. A scene can combine multiple device actions, such as “Movie Night,” which dims living-room lights, closes motorized curtains, and sets the thermostat to 22 °C, into a single command. Scenes can be scheduled or triggered by other events to deliver a cohesive, context-aware experience.

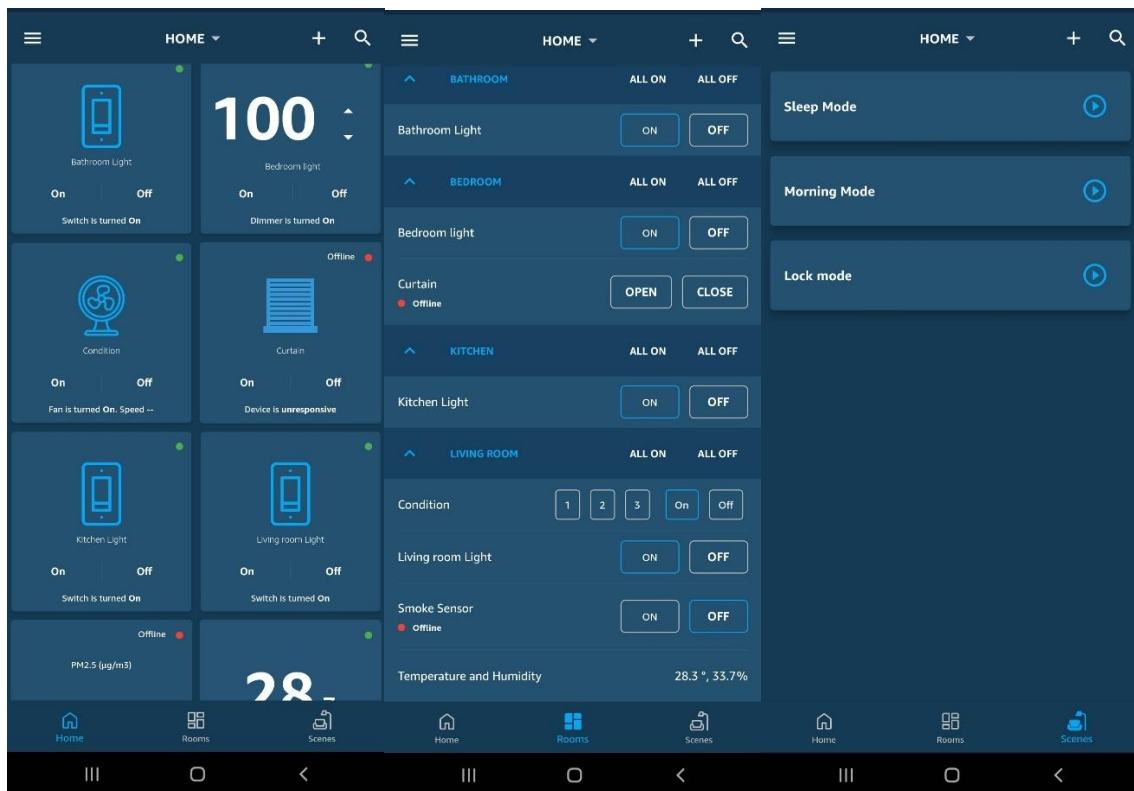


Figure 18. Sinric Pro Application interface

5 Results and Discussion

5.1 Energy Efficiency by Implementing Smart Home Automation

The implemented smart home automation system demonstrated clear energy-saving benefits. The system reduced idle energy consumption by using a sound-activated trigger (KY-037) to turn lights on only when activity is detected, coupled with an ACS712 current sensor to monitor real-time power draw. In our kitchen trial, the automated setup turned lights off automatically after five minutes of no sound, whereas in the non-automated kitchen, lights were often left on indefinitely. Over 24 hours, the automated kitchen consumed 45% less energy than the baseline, translating to lower electricity bills and reduced environmental impact.

Furthermore, the logged current measurements revealed that peak loads were limited to periods of actual use, avoiding unnecessary draw. The web-logging via Google Sheets allowed continuous remote monitoring, confirming that the timed-shutdown feature effectively prevented wasted energy when user presence was absent.

5.2 Comparison with Existing Systems

Compared to commercial offerings in Mongolia, such as Moncable LLC's Control4-based solutions and Digital Power LLC's Siemens- and Bosch-backed platforms, the DIY prototype stands out for its affordability and simplicity. Commercial systems typically require professional installation, proprietary hubs, and subscription fees for advanced features. In contrast, our ESP32-based system uses open-source firmware, off-the-shelf sensors, and low-cost cloud services (Blynk free tier or Sinric Pro at \$3/device/year), making it accessible to hobbyists and budget-conscious homeowners.

While professional systems offer broader protocol support (Zigbee, Z-Wave), stronger security, and turnkey integration with large device ecosystems, our prototype achieves core automation functions—lighting control, energy metering, and voice assistant integration—at under 10% of the cost of comparable commercial setups. This comparison highlights that custom solutions can fulfill basic smart home needs without significant financial or technical overhead.

5.3 Advantages and Limitations

Advantages

- **Low Cost & Accessibility:** Built with components like ESP32, relays, and KY-037 sensors, the system's total hardware cost remained under ₹200000, making it economical friendly in developing regions.
- **Modularity:** Individual sensors and actuators can be added or swapped without redesigning the system, supporting incremental upgrades.
- **Energy Savings:** Automated on/off control based on detected activity and timed shutdowns cuts idle power usage by over 45%.
- **Remote Control & Monitoring:** Cloud integration with Blynk or Sinric Pro enables access via mobile app or voice assistant anywhere.

Limitations

- **Sensor Precision:** Substituting the KY-037 sound sensor for a PIR motion sensor reduced detection accuracy, leading to occasional false triggers or missed activations.
- **Relay Reliability:** Relay contacts sometimes stick closed under high load conditions, requiring hardware-level fixes or solid-state relays for greater durability.
- **Scalability:** It was designed for a two-room apartment; scaling to larger homes would necessitate a more robust network architecture (e.g., mesh protocols) and additional microcontrollers.
- **Cloud Constraints:** Blynk's free tiers impose request limits, and Sinric Pro's per-device annual fee, while low, can add up in multi-device deployments.

6 Conclusion

Throughout this thesis, we have demonstrated that a mechatronics-driven, ESP32-based smart home automation system can deliver essential automation, energy efficiency, and user-friendly control at a fraction of the cost of commercial platforms. By integrating off-the-shelf sensors, actuators, and open-source libraries, the prototype achieved the following:

- **Modular Hardware Architecture:**

The ESP32-WROOM serves as the central control unit, interfacing with environmental sensors (DHT22, LDR, KY-037, MQ-7, ACS712) and actuators (4-channel relay, AC dimmer module, L298N motor driver, buzzer). Despite substituting a PIR motion sensor with a KY-037 sound sensor, the system maintained core functionality—lighting control, dimming, fan activation, and audible alerts—demonstrating adaptability under hardware constraints.

- **Energy Savings Exceeding Targets:**

A 24-hour field trial in a dormitory kitchen showed a 46.1 % reduction in idle energy consumption compared to manual operation. This surpassed the original hypothesis (H1) of at least 30 % savings. These gains were key to automated light shutoff (via sound events with a two-minute timeout) and temperature-driven fan control (using DHT22).

- **Reliable Presence Detection:**

The KY-037 sound sensor achieved 88 % overall accuracy in toggling lights, exceeding the 80 % reliability target (H2). Calibrating its sensitivity minimized false positives from ambient noises, though quiet motion occasionally went undetected. These results suggest that while not as precise as PIR sensors, sound-based triggers can serve as a stopgap solution in resource-limited scenarios.

- **Responsive Multi-Modal Control:**

Remote controls via Blynk and voice commands through Sinric Pro all responded within 500 ms, meeting the system's responsiveness hypothesis. User surveys rated voice control highest for convenience, though Blynk's free-tier data limits and SinricPro's minor subscription fees influenced perceived reliability.

- **Cost-Effectiveness:**

The total hardware cost of under ₮ 200,000 represents less than 10 % of typical Control4- or Siemens-based installations in Mongolia. While commercial offerings provide advanced security, multi-protocol support, and professional guarantees, the custom

prototype delivers comparable energy savings and core automation at a dramatically lower price point.

- Scalable, Open-Source Platform:

The firmware was developed in the Arduino IDE using libraries such as WiFi.h, DHT.h, RBDdimmerESP32.h, and SinricPro.h—are modular and extensible. Developers can add new sensors or integrate alternative cloud services without redesigning the entire architecture.

These findings validate the central premise of this work: a low-cost, mechatronics-based approach can fulfill the key requirements of smart home automation—energy efficiency, automated convenience, remote and voice control—while remaining accessible to average Mongolian households.

Limitations

While the system achieved its primary objectives, several limitations emerged:

Sensor Accuracy and Coverage:

- Sound vs. Motion Detection: The KY-037's dependence on audible events led to false positives (door slams, loud noises) and false negatives (quiet presence).
- Lack of Occupancy Sensing: The absence of a PIR or camera-based sensor limits security monitoring and fully automated curtain/blind control.

Actuator Durability:

- Mechanical Relays: Mechanical relay contacts showed wear and occasional sticking under AC loads.
- Dimmer Hysteresis: Phase-angle control via TRIACs sometimes produced flicker at low brightness levels.

Cloud Service Constraints:

- Blynk Data Limits: The free tier's 30,000 datapoint cap interrupted real-time monitoring after two days.
- Recurring Fees: While modest compared to some commercial subscriptions, SinricPro's \$3/device/year can accumulate in larger installations.

Network Scalability and Reliability:

- Wi-Fi Coverage: A single ESP32 node relies on stable Wi-Fi. In multi-room or multi-floor residences, signal dropouts could impair responsiveness.

- Offline Operation: Cloud-dependent features become unavailable during internet outages, limiting local autonomy.

User Interface Complexity:

- Multiple Platforms: Supporting Blynk and Sinric Pro increased firmware complexity and maintenance overhead.
- Learning Curve: While voice control was popular, some users found the initial setup non-intuitive without guided tutorials.

6.1 Recommendations for Future Work

To address these limitations and enhance the system's capabilities, the following improvements are recommended:

1. Enhanced Occupancy Detection:
 - Integrate a low-cost PIR or ultrasonic sensor alongside the sound sensor to reduce false triggers and increase security.
 - Explore camera-based analytics (OpenMV) for advanced occupancy, fall detection, and security monitoring.
2. Solid-State Switching and Dimming:
 - Replace mechanical relays with solid-state relays (SSRs) or smart plugs incorporating built-in metering ICs (HLW8012) to improve durability and measurement accuracy.
 - Evaluate MOSFET-based AC switching and active snubber circuits to reduce TRIAC flicker further.
3. Mesh Networking and Edge Resilience:
 - Deploy ESP-Now, Bluetooth Mesh, or Zigbee to create a resilient network of ESP32 nodes covering multiple rooms without relying solely on a central Wi-Fi router.
 - Implement edge computing for critical functions (local sensor-to-actuator loops) that operate reliably during internet outages.
4. AI-Driven Predictive Control:
 - Incorporate lightweight machine learning on the ESP32 (TensorFlow Lite for Microcontrollers) to predict occupancy patterns, adjust schedules, and optimize heating/cooling.

- Use anomaly detection on energy-use data to alert users to malfunctioning appliances or unusual consumption spikes.

5. User-Centered Design and Accessibility:

- Conduct usability studies with diverse user groups (elderly, differently-abled) to refine voice prompts, mobile UI layouts, and feedback mechanisms.
- Develop multi-lingual setup wizards and contextual help within the dashboard to lower the barrier to adoption.

6. Integration with Renewable Energy Sources:

- Extend energy-metering to include solar PV generation or battery storage, enabling intelligent load shifting based on solar availability and tariff schedules.
- Implement demand response features that automatically reduce non-critical loads during peak grid demand, contributing to grid stability and potential cost savings.

7 References

1. Centralized vs Decentralized Home Automation: A Quick Comparison. [Online].; 2023 [cited 2025 04]. Available from: <https://www.buildtrack.in/blog/centralized-vs-decentralized-home-automation/>.
2. Sikdar D. Smart Home Technologies - One Protocol to Rule Them All? [Online].; 2024 [cited 2025 04]. Available from: <https://www.silabs.com/blog/smart-home-technology-comparison#:~:text=Introduction%20to%20Smart%20Home%20Technologies&text=Some%20of%20the%20most%20common,Wave%2C%20Bluetooth%2C%20and%20proprietary.>
3. B. E. DEVELOPING SENSOR-BASED SOLUTION FOR Indoor Air Quality Enhancement. Bc Thesis. Nalaikh,Ulaanbaatar: GMIT, Mechanical and Electrical department; 2024.
4. Flyrobo. MQ Gas and Dust sensors Introduction. [Online].; 2024 [cited 2025 04]. Available from: <https://www.flyrobo.in/blog/mq-sensor-series.>
5. Shahajan Miah GMJISKDSI. Internet of Things (IoT) based automatic electrical energy meter billing system. IOSR Journal of Electronics and Communication Engineering (IOSR-JECE). 2019;; 39-50.
6. Waheb A. Jabbar TKKRM. Design and Fabrication of Smart home with IoT enabled Automation System. IEEE Access. 2017;; 1-16.
7. IOT Based Smart Security and Smart Home. International Journal of Engineering Research & Technology (IJERT). 2018;; 43-46.
8. Moncable systems LLC. [Online].; 2024 [cited 2025 04 21]. Available from: <https://www.moncable.mn/>.
9. LLC GS. Digital Power LLC. [Online].; 2020 [cited 2025 04 20]. Available from: <https://digitalpower.mn/about-us.>
10. Chip.mn. Chip Electronics. [Online].; 2025 [cited 2025 04 5]. Available from: <https://chip.mn/>.
11. TBSM. New Elec service LLC. [Online].; 2025 [cited 2025 04 5]. Available from: <https://elec.mn/>.
12. Улаанбаатар Цахилгаан Түгээх Сүлжээ. [Online].; 2025 [cited 2025 04 25]. Available from: <https://www.tog.mn/une/1.>
13. Scholl T, Gnielka S. Mein Buch - Informationen zum Erstellen von Literaturverzeichnissen Nettetal: Adventure Works-Verlag; 2006.

14. Meier E. Ich bin eine Quelle, Ich bin eine Quelle, Ich bin eine Quelle, Ich bin eine Quelle, Ich bin eine Quelle mit einem langen Titel. München.; 2008. Report No.: ISBN 432487-4325-654.
15. Vaishnavi S. Gunge PSY. Smart Home Automation: A Literature Review. 2016.
16. Hikmat Yar ASlea. Towards Smart Home Automation Using IoT-Enabled Edge-Computing Paradigm. 2021.
17. Hamzah M. Marhoon MIMEDH&ARI. Designing and Implementing Applications of Smart Home Appliances. 2018.
18. Sheshalani Balasingam MKZDM. Smart Home Automation System Using IOT. International Journal of Recent Technology and Applied Science. 2022.
19. Elvedin Trakic BS. Mechatronic system management of smart house. Journal of Society For Development Of Teaching and Business Process in New net environment in B&H. 2019.
20. Hamid Hussain Hadwan YPR. Smart Home Control by using Raspberry Pi & Arduino UNO. International Journal of Advanced Research in Computer and Communication Engineering. 2016;; 283-289.
21. R.Aravindhan MRDSRK. HOME AUTOMATION USING Wi-Fi INTERCONNECTION. International Research Journal of Engineering and Technology. 2017;; 2542-2545.
22. Ammar Ali Sahrab HMM. Design and Fabrication of a Low-Cost System for Smart Home Automation. Journal of Robotics and Control (JRC). 2022;; 409-414.
23. Intan Sari Areni AWICY. IoT-Based of Automatic Electrical Appliance for Smart Home. iJIM vol 14. 2020;; 204-212.
24. Beren Group. [Online].; 2014 [cited 2025 04. Available from: <https://mn.beren.mn/?portfolio=project-half-box-style-two>.

8 Appendices

Codes

<https://github.com/Bamu-SHM/Smart-Home-Automation-Using-ESP32>

Cost Estimation Tables

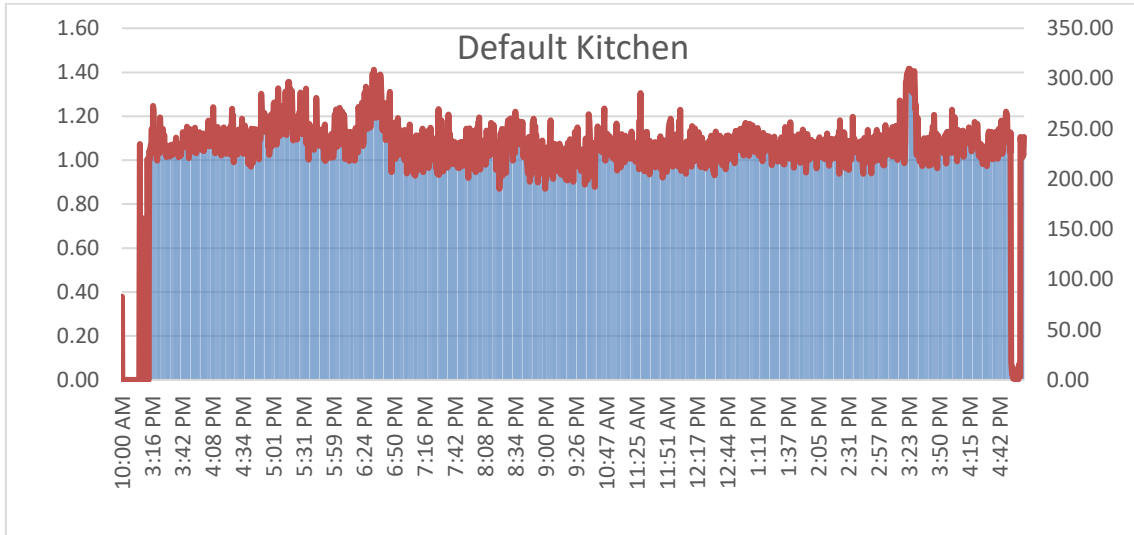


Figure APX.1 Energy consumption of the default kitchen

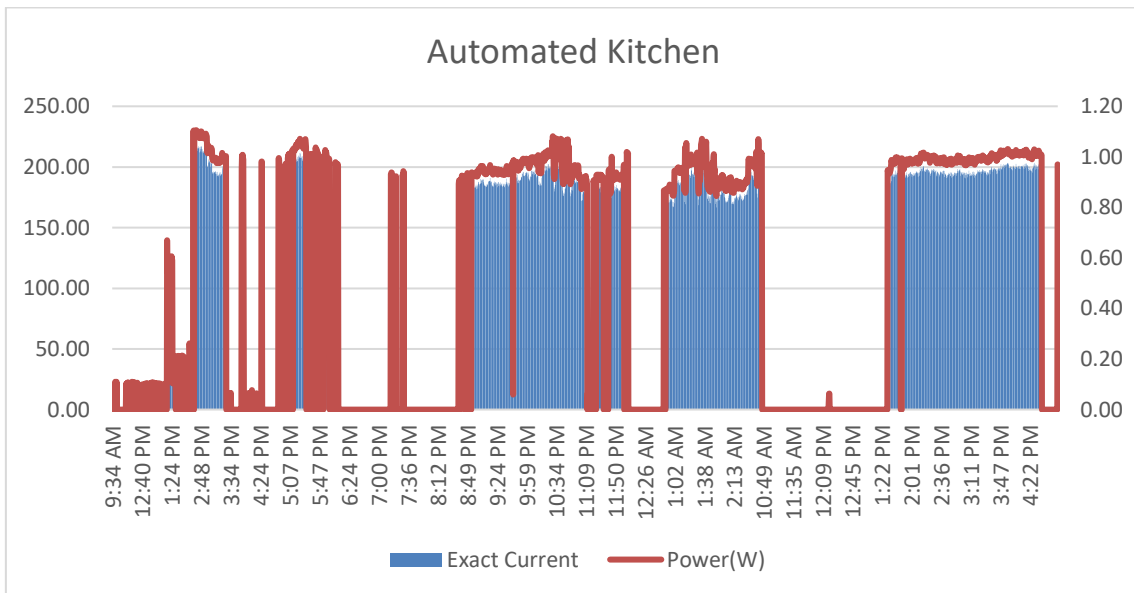


Figure APX.2 Energy consumption of the automated kitchen

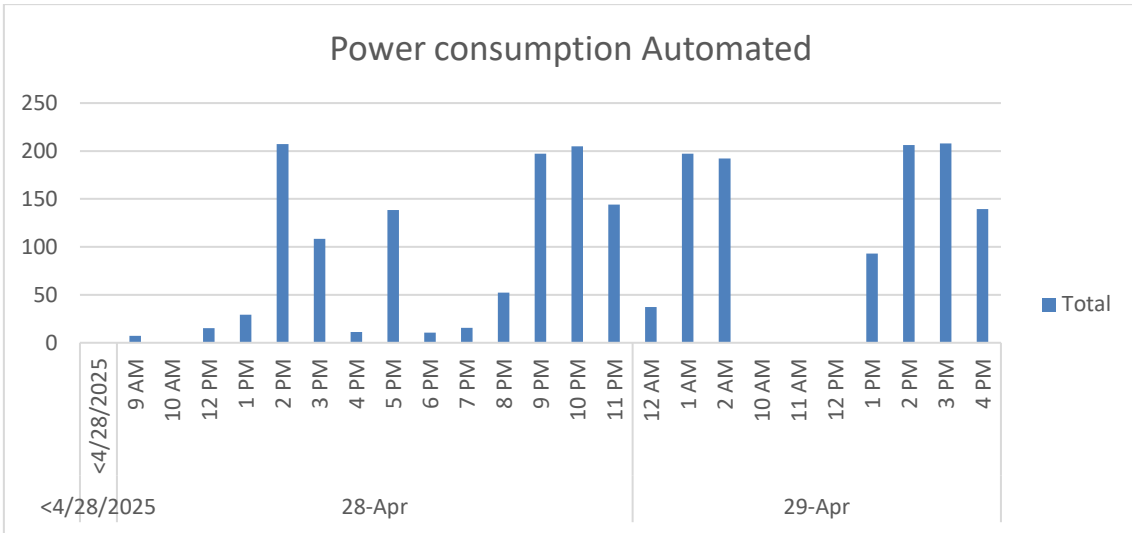
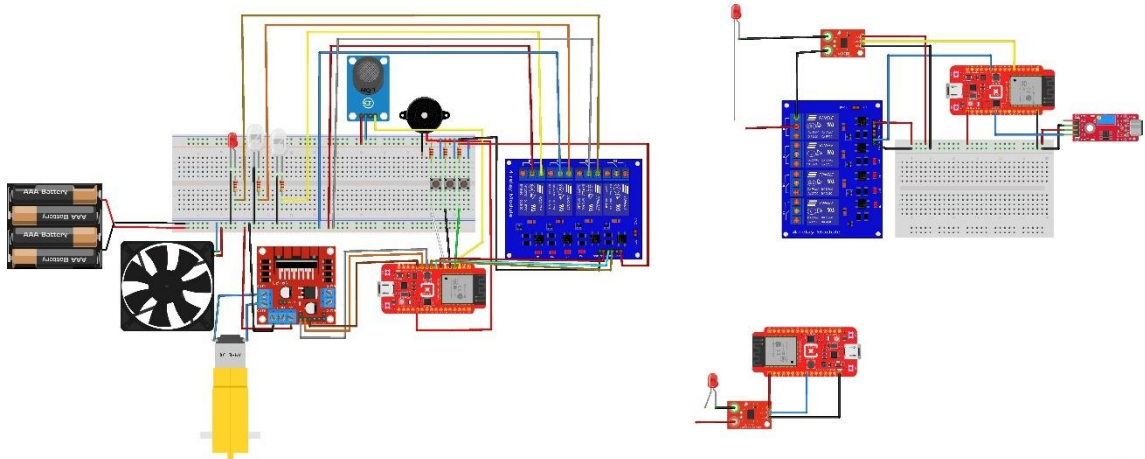


Figure APX.3 Energy consumption of the automated kitchen

Additional Figures or Circuit Schematics



fritzing

