Smart Home Lighting Control System Using ESP32 IoT Modules.

Group members.

- MEC.22.B1.14 –R.D.B.D MUNASINGHE
- MEC.22.B1.25 – N.P.R VITHANAGE

Date

2025-06-21

TABLE OF CONTENT

INTRODUCTION

Problem

In many households, lighting systems are still predominantly manual and disconnected from modern smart technologies. Users often have to physically switch lights on or off, which can be inconvenient, especially when away from home or in large houses with multiple rooms. Moreover, traditional lighting systems do not provide real-time feedback or automation, which can lead to unnecessary energy consumption and increased electricity costs. For example, lights may be left on unintentionally, wasting energy and contributing to higher utility bills. Additionally, manual control lacks customization options such as scheduling, remote access, or integration with other smart home devices. This gap highlights the need for a practical, user-friendly solution that allows homeowners to efficiently manage their lighting systems remotely and intelligently.

Motivation

The motivation behind this project stems from the growing demand for smart home technologies that enhance convenience, energy efficiency, and user comfort. Smart lighting systems are a key component of modern homes, offering benefits such as remote control, automation, and energy savings. By leveraging Internet of Things (IoT) technologies, it is possible to transform traditional lighting into an intelligent system that can be controlled via smartphones or computers from anywhere with internet access. This not only improves user experience but also contributes to sustainable living by reducing unnecessary energy consumption. Furthermore, developing a system based on widely available and cost-effective hardware like the ESP32 encourages accessibility and scalability, making smart home technology attainable for a broader audience.

Aim & Objectives

The primary aim of this project is to design and implement a practical IoT-based smart lighting control system using two ESP32 modules. This system will enable users to remotely control and monitor their home lighting through a Python-based web application. The project will focus on:

- Establishing reliable Wi-Fi communication between ESP32 devices and the web app.

- Utilizing MQTT and HTTP protocols for efficient, real-time data exchange.
- Structuring control and status data using JSON format for seamless integration.
- Programming the ESP32 modules in C++ to handle sensor inputs, control relays, and communicate with the web app.
- Developing a user-friendly Python web dashboard that allows users to turn lights on/off, schedule lighting, and view real-time status updates.

Through these objectives, the project aims to deliver a smart lighting solution that is practical, scalable, and enhances everyday living by combining hardware and software IoT components effectively.

Literature Review

Existing Work in IoT-Based Smart Home Lighting Control

The integration of IoT technologies into home automation has been extensively studied, with a focus on improving convenience, energy efficiency, and user comfort. Several research projects have demonstrated the feasibility of using ESP32 microcontrollers for smart home automation, particularly in controlling lighting and other household appliances.

Anisha S C et al. (2024) presented a home automation system using the ESP32 board as a micro web server, enabling remote control of lights, fans, and other devices via smartphones. Their system highlights the ESP32's versatility in interfacing with hardware and providing Wi-Fi connectivity for remote management, thereby enhancing comfort and security in smart homes. Similarly, a study by IOSR Journals (2024) developed an ESP32-based home automation system that incorporated sensors such as Light Dependent Resistors (LDR) and motion detectors (RCWL-0516) to automate lighting based on ambient light and occupancy. This approach not only improved user convenience but also contributed to energy conservation by ensuring lights are only on when needed.

Another notable work by SSRN (2024) integrated the Blynk IoT platform with ESP32 to control multi-channel relay modules for home appliances. This system provided dual control modes—manual switches and smartphone app—ensuring operation even without internet connectivity. The real-time feedback feature enhanced user awareness and control, addressing a common limitation in IoT systems dependent on continuous internet access. Furthermore, the study by IJRASET (2024) emphasized the role of ESP32 as a central hub in home automation, integrating multiple sensors and actuators to enable seamless communication and control of household devices.

Advancing beyond basic automation, recent research has incorporated machine learning (ML) to create adaptive lighting control systems. A comprehensive study by IAR Consortium (2024) proposed an indoor lighting control system using ESP32, IoT, and ML algorithms. The system utilized LDR and PIR sensors to monitor ambient light and occupancy, combined with weather data from the OpenWeather API, to dynamically adjust lighting levels. The ML models, including Decision Tree Regression and Multilayer Perceptron Neural Networks, significantly improved energy efficiency by reducing consumption by 34.8% compared to non-adaptive systems, while maintaining occupant comfort through precise light intensity control. The system was scalable and supported remote monitoring via the Blynk platform, demonstrating the potential of integrating IoT with advanced data processing for smart lighting

In addition to control and automation, energy sustainability has been a focus in smart home research. Hadizadeh Masali et al. (2024) explored the integration of solar energy systems with smart home lighting control using ESP32. Their eco-smart system combined real-time monitoring and control of lighting and temperature with solar power, reducing reliance on grid electricity and promoting environmental sustainability. The study highlighted the role of communication protocols and microcontrollers in creating an efficient, remotely controllable home environment.

Gaps and Limitations in Existing Solutions

Despite these advancements, several limitations persist in current smart lighting systems:

- Internet Dependency: Many IoT lighting solutions rely heavily on continuous internet connectivity for remote control and monitoring. Systems like the one using Blynk provide fallback manual controls, but internet outages still pose challenges for seamless operation.
- Limited Adaptability: While some systems incorporate sensors for ambient light and motion detection, few fully leverage adaptive algorithms or machine learning to optimize lighting based on environmental and user behavior patterns comprehensively.
- Scalability and Integration: Existing systems often focus on single-room or small-scale implementations. There is a need for scalable architectures that can support multiple devices and integrate with broader smart home ecosystems without significant complexity.
- User Interface and Control: Many projects provide basic smartphone apps or web interfaces but lack intuitive, feature-rich dashboards that combine control, scheduling, and real-time feedback in a single platform.
- Security and Privacy: As IoT devices become more interconnected, ensuring secure communication and data privacy remains a critical concern that is not deeply addressed in many existing works.

How This Project Addresses the Gaps

The proposed project aims to develop a practical, scalable IoT smart lighting control system using two ESP32 modules that overcomes these limitations by:

- Robust Communication: Utilizing both HTTP and MQTT protocols over Wi-Fi to ensure reliable, low-latency communication between ESP32 modules and the Python web app, reducing dependency on external cloud services and mitigating internet outage issues.
- JSON Data Structuring: Employing JSON for structured data exchange enables flexible and extensible communication, facilitating easy integration of additional sensors or features in the future.
- Comprehensive Control Interface: Developing a Python-based web dashboard that provides users with real-time status updates, manual control, and scheduling capabilities in an intuitive interface, improving usability over basic smartphone apps.

- Modular and Scalable Design: Using two ESP32 modules allows distributed control and monitoring, demonstrating a scalable approach that can be expanded to multiple rooms or devices.
- Security Considerations: Implementing basic security measures in communication protocols and data handling to protect user data and prevent unauthorized access.

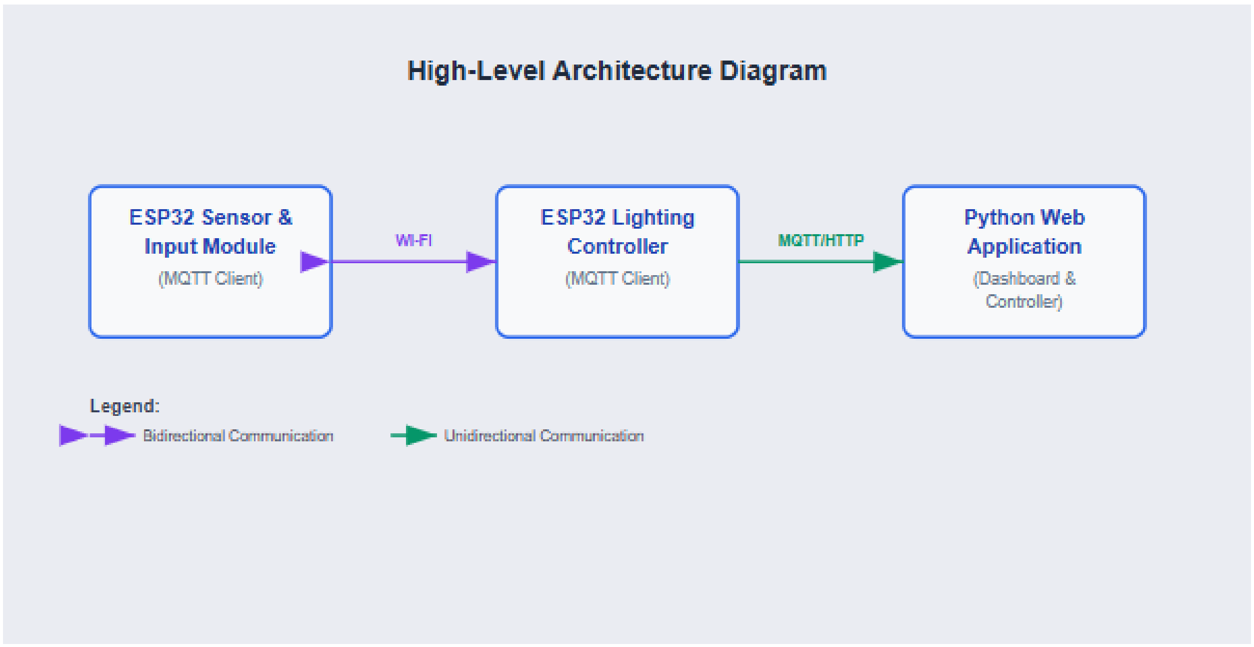
By combining these elements, the project not only replicates the functionalities of existing systems but also enhances adaptability, user experience, and reliability. It builds upon the strengths of prior research, such as the sensor-based automation and machine learning insights, while focusing on practical implementation and user-centric design.

Methodology & System Design

1. System Architecture

The proposed IoT smart lighting control system consists of two ESP32 modules communicating over Wi-Fi using MQTT and HTTP protocols. One ESP32 acts as the Controller Unit connected to the lighting hardware (relays or LED strips), while the other serves as a Sensor & Input Unit or a secondary controller for distributed control. Both modules connect to a local Wi-Fi network and communicate with a Python-based web application hosted on a server or PC, which provides a user-friendly dashboard for remote monitoring and control.

High-Level Architecture Diagram:


High-Level Architecture Diagram
ESP32 Sensor & Input Module
(MQTT Client)
WI-FI
ESP32 Lighting Controller
(MQTT Client)
MQTT/HTTP
Python Web Application
(Dashboard & Controller)
Legend:
Bidirectional Communication
Unidirectional Communication

- The ESP32 Sensor & Input Module gathers inputs (e.g., motion, light sensors, buttons).
- The ESP32 Lighting Controller manages the physical lights via relays or LED drivers.
- The Python Web App communicates via MQTT broker or HTTP REST API to send commands and receive status updates.

2. Hardware Components

Component	**Purpose & Justification**
ESP32 DevKitC (x2)	Central microcontrollers with Wi-Fi, Bluetooth, GPIOs, ADCs, and PWM support; ideal for IoT applications. ESP32 supports C++ programming and multiple communication protocols
Relay Module (e.g., 4 or 8-channel)	To switch high-voltage house lights on/off safely via ESP32 GPIOs, enabling control of standard AC lighting circuits
PIR Motion Sensor	Detects human presence to automate lighting based on occupancy, improving energy efficiency
LDR (Light Dependent Resistor)	Measures ambient light intensity to enable adaptive lighting control (turn lights off when natural light is sufficient)
LED Indicators	Provide visual feedback on system status and manual controls
Power Supply	5V/3.3V regulated supply for ESP32 and sensors; ensures stable operation
Breadboard & Jumper Wires	For prototyping and connecting components without soldering

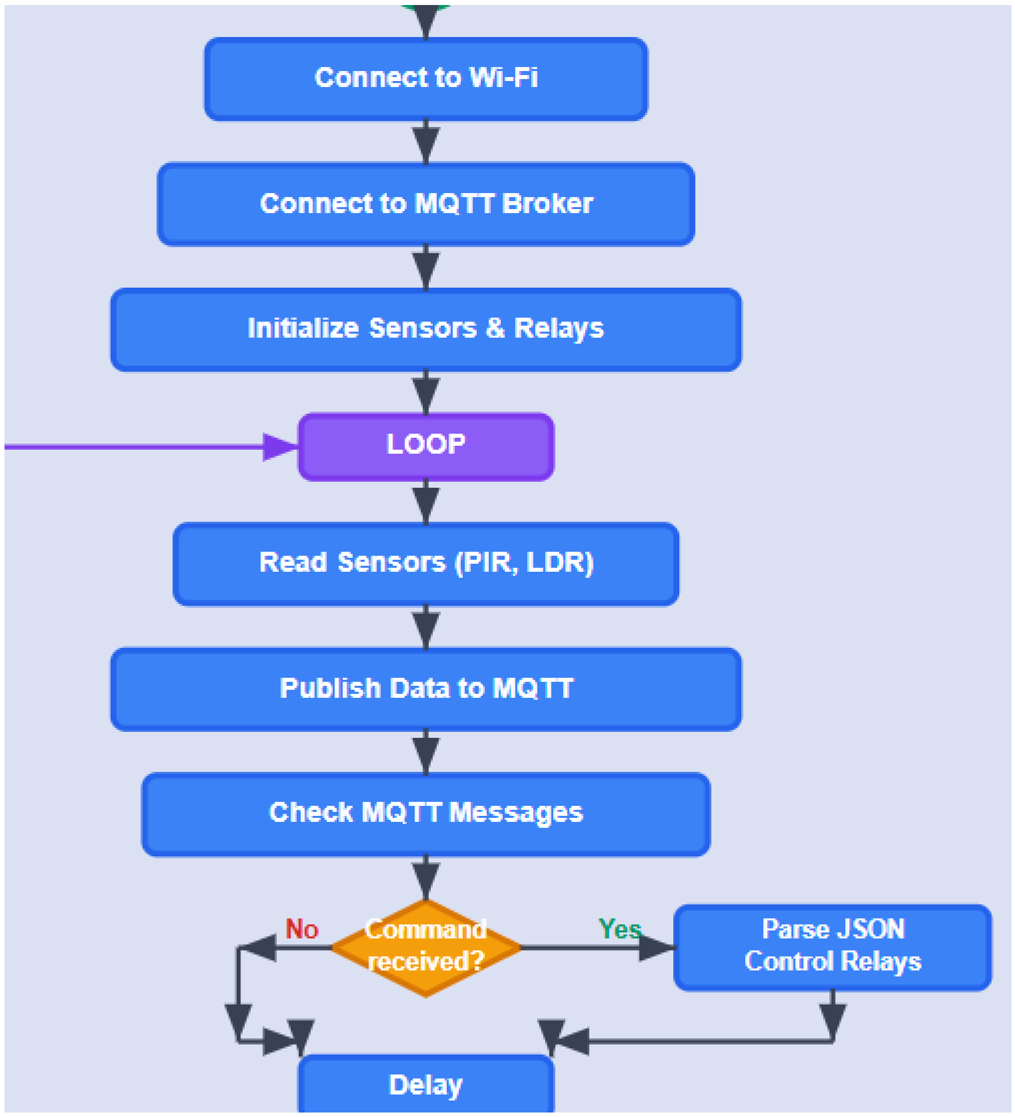
3. Software Design

Functionality:

- Connect ESP32 to Wi-Fi network.
- Initialize sensors (PIR, LDR) and relay outputs.
- Read sensor data periodically.
- Publish sensor data/status via MQTT in JSON format.
- Subscribe to MQTT topics for control commands.
- Parse incoming JSON commands to toggle relays or adjust lighting.

- Optionally, provide HTTP REST endpoints for direct control.

Flowchart (simplified):



Communication Protocols: MQTT & HTTP

- MQTT is chosen for its lightweight, efficient publish/subscribe model, ideal for real-time IoT device communication with low bandwidth and latency. It enables decoupled communication between devices and the web app.

- HTTP is optionally used for RESTful API endpoints to allow direct commands or status queries from the web app or external clients.

MQTT Topics:

Topic	**Description**
home/lights/sensor	ESP32 publishes sensor data (JSON)
home/lights/control	Web app publishes control commands
home/lights/status	ESP32 publishes light status updates

Data Structure (JSON)

Sensor Data

- pir: Motion detected (1) or not (0)
- ldr: Analog light intensity reading (0-4095)
- timestamp: ISO 8601 time of reading

Control Command

- light_id: Identifier for the light or relay channel
- state: "on" or "off"
- brightness: Percentage (0-100), if PWM dimming supported

Python Web Application

Features:

- User authentication (optional)
- Dashboard displaying real-time sensor data and light status

- Controls to turn lights on/off and adjust brightness
- Scheduling feature to automate lighting based on time
- MQTT client to publish commands and subscribe to sensor/status topics
- REST API endpoints to interact with ESP32 HTTP interface (if implemented)

Framework & Tools

- Flask or Django for backend web server
- Paho-MQTT Python library for MQTT communication
- JavaScript (e.g., React or simple AJAX) for dynamic UI updates
- Bootstrap or similar CSS framework for responsive design

Design

- The web app subscribes to MQTT topics to receive sensor updates and light status.
- User actions on the dashboard send MQTT messages to control lights.
- The app maintains a synchronized view of the system state.

Network Design

- Both ESP32 modules connect to the home Wi-Fi router.
- An MQTT broker runs locally on a PC or cloud server accessible to all devices.
- The Python web app runs on the same server or a dedicated machine connected to the same network.
- Communication flow is secured via Wi-Fi WPA2; MQTT can be optionally secured with username/password or TLS.

Data Flow

Step-by-step Data Journey:

- Sensor Reading: ESP32 Sensor Module reads PIR and LDR sensors.

- Data Publishing: Sensor data is formatted as JSON and published to MQTT topic home/lights/sensor.
- Data Reception: Python web app subscribes to home/lights/sensor, receives data, and updates the dashboard UI.
- User Command: User interacts with the web app to toggle lights or adjust brightness.
- Command Publishing: Web app publishes JSON command to MQTT topic home/lights/control.
- Command Reception: ESP32 Lighting Controller subscribes to home/lights/control, parses command, and actuates relays or LEDs.
- Status Update: ESP32 publishes updated light status to home/lights/status.
- Status Display: Web app receives status update and refreshes UI accordingly.

Implementation Plan & Timeline

Task Allocation

Task	**Description**	**Assigned To**
Project Setup & Research	Study ESP32 capabilities, Wi-Fi, MQTT, HTTP, JSON, and Python web app frameworks.	Member 1
Hardware Procurement & Setup	Acquire ESP32 boards, sensors (PIR, LDR), relay modules, and power supplies.	Member 2
ESP32 Firmware Development	Program ESP32 modules in C++: Wi-Fi connection, sensor reading, relay control, MQTT & HTTP comms	Member 1
Sensor Module Implementation	Interface PIR and LDR sensors with ESP32, publish sensor data via MQTT in JSON format.	Member 2
Lighting Controller Implementation	Control relays to switch lights on/off based on received MQTT commands.	Member 1
Python Web App Development	Build dashboard for monitoring sensor data and controlling lights, implement MQTT client.	Member 2
Documentation & Report Writing	Prepare project proposal, methodology, results, and final report.	Both Members

Presentation Preparation	Create slides and rehearse project presentation.	Both Members

Detailed Schedule

Week	Tasks	Deliverables/Goals
Week 1	Finalize project scope and design proposal - Research ESP32, MQTT, HTTP, JSON basics	Project Proposal document completed
Week 2	Procure hardware components - Set up development environment (Arduino IDE/PlatformIO) - Basic ESP32 Wi-Fi connection test	Hardware ready ESP32 connected to Wi-Fi
Week 3	Develop sensor module firmware: PIR and LDR sensor interfacing - Publish sensor data via MQTT	Sensor data published successfully
Week 4	Develop lighting controller firmware: relay control via MQTT commands - Implement JSON parsing for commands	Relay control working via MQTT
Week 5	Start Python web app development: set up Flask/Django framework - Implement MQTT client to receive sensor data	Basic web dashboard receiving sensor data
Week 6	- Add control features to web app: buttons to toggle lights, send MQTT commands - Implement JSON message formatting	Web app controls lights remotely
Week 7	- Integrate sensor and lighting modules with web app - Conduct end-to-end system testing and debugging	Fully functional integrated system
Week 8	- Optimize code and UI - Prepare documentation and final report sections - Prepare presentation slides	Final report draft Presentation ready
Week 9	- Final testing and validation - Rehearse presentation	Project submission Presentation delivered

	- Submit all deliverables	

Notes on Collaboration

- Member 1 focuses primarily on ESP32 firmware development, including Wi-Fi, MQTT communication, sensor interfacing, and relay control.
- Member 2 handles hardware setup, sensor integration, and Python web application development including MQTT client and UI.
- Both members collaborate closely during integration, testing, documentation, and presentation preparation.
- Weekly meetings will be scheduled to review progress, discuss challenges, and adjust plans as necessary.

Risk Management & Contingency

- Hardware delays: Early procurement to avoid schedule slips.
- Communication issues: Start with simple MQTT examples from templates like mbursa/esp32-iot-boilerplate to reduce debugging time.
- Software bugs: Allocate sufficient time for integration and testing.
- Learning curve: Use online tutorials and resources to accelerate learning of ESP32 and Python frameworks.

Budget/Resources

For this project, the primary hardware resources include two ESP32 development boards, sensors, relay modules, and basic prototyping accessories. The ESP32 boards are the core microcontrollers providing Wi-Fi connectivity and processing power, essential for implementing the IoT system.

- ESP32 Development Boards (x2): Each board costs approximately LKR 1,500 to LKR 1,900 (about $4–$6 USD) depending on the model and supplier. Two boards are required for the sensor/input and lighting control modules, totaling roughly LKR 3,000 to LKR 3,800.
- PIR Motion Sensor: Estimated cost around LKR 300–500. Used to detect occupancy for automated lighting control.
- LDR (Light Dependent Resistor): Low-cost component (~LKR 100–200) to measure ambient light levels.
- Relay Module (4 or 8 channel): Approximately LKR 1,000–1,500, used to switch household lights safely via the ESP32 GPIOs.
- Power Supplies, Jumper Wires, Breadboards: Estimated total of LKR 1,000 for stable power and prototyping connections.

Total Estimated Budget: LKR 5,400 to LKR 6,500 (approximately $15–$20 USD), which is cost-effective for a practical IoT smart lighting system.

The project leverages open-source software tools such as Arduino IDE for ESP32 programming and Python frameworks (Flask/Django) for the web app, minimizing software costs.

References

- Vaduva Ionut Lucian, "Control a Smart Light with ESP32 and Wi-Fi – IoT Home Automation," Hackster.io, April 3, 2025.
  Available: https://www.hackster.io/vaduvaionutlucian/control-a-smart-light-with-esp32-and-wi-fi-iot-home-automa-94e424
- IoT Circuit Hub, "Latest ESP32 Projects 2025 on IoT based Home Automation," December 7, 2022. Available: https://iotcircuithub.com/esp32-projects/
- Techatronic, "Light Bulb Control using IoT | ESP32 IoT Control project," April 5, 2024.
  Available: https://techatronic.com/light-bulb-control-using-iot-esp32/
- Random Nerd Tutorials, "Getting Started with the ESP32 Development Board," August 5, 2024.
  Available: https://randomnerdtutorials.com/getting-started-with-esp32/
- Arduino Titan, "ESP32 Light Sensor with LED Control | Smart Light Automation Tutorial," YouTube Video, 2024.
  Available: https://www.youtube.com/watch?v=YhuIzQ6_liw
- Techatronic, "Light Control Project Using ESP32 and Android App," August 6, 2024.
  Available: https://techatronic.com/light-control-project-using-esp32/
- ElectroniClinic, "ESP32 Home Automation Project, IoT Smart Phone Home Automation using Wifi & Blynk," YouTube Video, 2024.
  Available: https://www.youtube.com/watch?v=DxPIvYe1Ltg
- Instructables, "Smart Home Automation Using Blynk & ESP32 IoT Projects," 2024.
  Available: https://www.instructables.com/Smart-Home-Automation-Using-Blynk-ESP32-IoT-Projec/