AI-Based Smart Bulb for Adaptive Home Automation

Mid-Term Progress Report

by Caleb Ram

Student ID: 6801936

Supervisor: Dr Ahmed Elzanaty

Started: Thursday 2nd October, 2025

Submission: Monday 24th November 2025

MEng in Electronic Engineering with Computer Systems

Faculty of Engineering and Physical Sciences

University of Surrey

Executive Summary

This report presents the mid-term progress of developing an AI-based smart bulb system for adaptive home automation. The project aims to create a complete smart lighting solution incorporating custom ESP32 hardware, embedded programming, and on-device machine learning that learns user lighting preferences and automatically adapts to daily routines without cloud connectivity.

The system's core innovation lies in lightweight TinyML that processes user behaviour patterns locally on the ESP32 microcontroller, learning preferred brightness levels, colour temperatures, and timing patterns for different times of day and activities. The AI model will detect routines such as morning wake-up times, evening wind-down periods, and room occupancy patterns, automatically adjusting lighting without manual intervention. Integration with Apple HealthKit enables the system to access sleep data with user consent, correlating lighting preferences with sleep quality and automatically triggering gradual dimming before detected bedtime and natural sunrise simulation for wake-up.

Completed work to date includes a fully functional Swift iOS application has been developed and deployed to physical devices. Features include user authentication with email verification, Bluetooth Low Energy device discovery and connection management, and real-time control of power, brightness, RGB colour, and four lighting effects. A Flask backend provides secure user management and device storage. The system implements dual-mode architecture supporting both physical hardware and software simulation for testing. The BLE communication protocol is fully specified for bidirectional communication between app and hardware, with characteristic UUIDs defined for control commands and status monitoring.

Remaining work includes the ESP32 firmware implementing BLE services and TinyML model for on-device learning, AI algorithm development for pattern recognition and behaviour prediction, physical hardware assembly including LED drivers and thermal management, housing design, HealthKit integration for sleep-aware automation, comprehensive testing of adaptive learning accuracy, and system integration. Multi-room coordination and advanced automation rules will be implemented if time permits.

The project demonstrates practical application of embedded machine learning, IoT communications, behavioural pattern recognition, and privacy-preserving AI with emphasis on local processing, user autonomy, and intelligent automation.

Contents

1 Introduction

1.1 Project Motivation

Modern home automation systems typically require explicit user input to control lighting, relying on manual adjustments, preset schedules, or basic motion detection. These approaches fail to adapt to the dynamic nature of human behaviour and circadian rhythms, resulting in inefficient energy usage and suboptimal lighting conditions that may negatively impact sleep quality and overall wellbeing [1]. Furthermore, existing commercial smart bulbs often require cloud connectivity and external servers, raising privacy concerns regarding user behaviour data and creating dependencies on internet connectivity for basic functionality [2].

The relationship between light exposure and human circadian rhythms is well documented in scientific literature. Inappropriate lighting during evening hours suppresses melatonin production, delaying sleep onset and reducing sleep quality [3]. Conversely, gradual light exposure mimicking natural sunrise can improve morning alertness and mood [4]. Despite this knowledge, most individuals lack automated systems that appropriately adjust lighting based on their personal schedules and physiological needs.

This project addresses these limitations by developing a smart bulb system that learns user behaviour patterns through on-device artificial intelligence, integrates with health data platforms to understand sleep patterns, and operates autonomously without requiring cloud connectivity. By processing all data locally using TinyML techniques, the system maintains user privacy whilst providing adaptive lighting that supports healthy circadian rhythms and energy efficiency [5].

The project also explores the technical challenges of building IoT devices from first principles, including thermal management in high-power LED systems, electrical safety isolation, wireless communication protocol implementation, and efficient embedded AI algorithm design. This hands-on approach provides deeper understanding of the complete system stack, from physical hardware design through embedded firmware to cross-platform application development.

1.2 Aims and Objectives

The primary aim of this project is to design, build, and evaluate a functional prototype of an AI-based smart bulb system that automatically adapts lighting conditions based on learned user behaviour patterns and health data integration, whilst maintaining local data processing for privacy preservation.

1.2.1 Primary Objectives

The core technical objectives that must be achieved for project success are:

Hardware Development: Design and construct a complete smart bulb system including LED lighting elements, power management circuitry, wireless communication module, and safe physical housing. This involves either systematic teardown and analysis of existing commercial products or full custom assembly from individual components, with particular attention to thermal management, electrical safety isolation between high and low voltage circuits, and compliance with relevant safety standards.

Embedded Systems Implementation: Develop firmware for ESP32 or ESP8266 microcontroller platform to control LED brightness and colour temperature, implement wireless communication protocols (Wi-Fi and Bluetooth), manage power states for energy efficiency, and execute lightweight machine learning algorithms for behaviour pattern recognition. The firmware must operate reliably with real-time responsiveness whilst managing multiple concurrent processes through appropriate threading and asynchronous programming techniques [6].

AI and Learning System: Implement on-device machine learning algorithms using TinyML approaches to identify recurring patterns in user behaviour, such as typical wake times, bedtimes, and room usage patterns. The system must learn these patterns without requiring pre-programming or manual configuration, adapting automatically as user routines change over time. All learning and inference must occur locally on the microcontroller without external server communication [7].

Health Data Integration: Develop iOS application using Swift and HealthKit framework to securely access user sleep data with appropriate permissions and consent mechanisms. Implement features for automatic sleep detection triggering wind-down mode with gradual dimming, and wake detection initiating natural morning light simulation. All health data processing must comply with privacy regulations and remain under user control [8].

Cross-Platform Interface Development: Create intuitive user interfaces for both iOS native application and web-based Progressive Web App, enabling device control, schedule management, and preference configuration across mobile and desktop platforms. Implement real-time communication between interfaces and embedded device using appropriate protocols such as WebSockets for responsive interaction [9].

1.2.2 Secondary Objectives

If time permits following completion of primary objectives, the following enhancements will be pursued:

Enhanced AI Capabilities: Extend machine learning system to recognise more complex behaviour patterns, predict future lighting needs based on contextual information such as day of week or season, and implement reinforcement learning approaches that improve based on user feedback signals [10].

Multi-Room Coordination: Implement communication protocols enabling multiple smart bulbs to coordinate behaviour across different rooms, creating adaptive lighting scenes that respond to user movement patterns throughout the home and maintain consistent lighting preferences across spaces.

Advanced Health Features: Integrate additional health metrics beyond sleep data, such as activity levels or heart rate variability if available through HealthKit, to further optimise lighting conditions. Implement alertness-based adjustments using circadian rhythm models to support optimal cognitive performance throughout the day [11].

1.3 Project Scope and Constraints

The project scope encompasses complete system development from hardware design through software implementation to testing and evaluation. The deliverable prototype will demonstrate all primary features operating reliably, with comprehensive documentation of design decisions, implementation challenges, and performance evaluation.

Several constraints shape the project approach:

Hardware Constraints: The system must utilise readily available components that can be sourced within reasonable budget and timeframe. Development will use prototyping boards and modules. Physical housing will be 3D printed using available university facilities, limiting material choices and potentially affecting thermal management solutions.

Safety Requirements: All electrical design must maintain appropriate isolation between mains voltage circuits and low-voltage control electronics. The system must not present electrical shock or fire hazards during normal operation or foreseeable failure modes. While full regulatory compliance testing (CE marking, etc.) is beyond project scope, design will follow relevant safety principles and standards [12].

Computational Limitations: Machine learning algorithms must operate within the processing and memory constraints of ESP32 microcontroller (dual-core 240 MHz, 520 KB RAM). This necessitates careful algorithm selection and optimisation, excluding complex deep learning models in favour of lightweight approaches suitable for embedded deployment [13].

Platform Limitations: iOS application development requires access to Apple development environment and devices for testing. HealthKit integration is iOS-specific; Android users will access core functionality through web interface but without native health data integration. This platform asymmetry reflects different vendor approaches to health data APIs and represents a realistic constraint in cross-platform IoT development.

Time Constraints: The project must be completed within the academic year timeframe, requiring careful prioritisation of primary objectives and realistic assessment of achievable scope. Secondary objectives are explicitly identified as time-permitting enhancements rather than core requirements.

2 Literature Survey and Technology Review

2.1 Smart Lighting Systems and Home Automation

The evolution of smart lighting systems represents convergence of LED technology, wireless communications, and embedded computing. Traditional lighting control relied on simple switches or timers, whilst modern smart bulbs integrate sensors, processors, and network connectivity to enable programmable and automated behaviour [14].

Commercial smart lighting platforms such as Philips Hue, LIFX, and Nanoleaf demonstrate market demand for intelligent lighting control. These systems typically employ hub-based or direct Wi-Fi architectures, supporting features including dimming, colour temperature adjustment, scheduling, and smartphone control [15]. However, most commercial solutions require cloud connectivity for advanced features and lack transparent on-device learning capabilities that adapt without explicit programming.

Research literature documents various approaches to adaptive lighting systems. Caird and Roy examined user interactions with smart home technologies, finding that successful systems balance automation with user control, avoiding excessive complexity or loss of user agency [16]. This insight guides the project's design philosophy: automation should enhance rather than replace user control, with learned behaviours remaining transparent and modifiable.

2.2 LED Technology and Driver Circuits

Light-emitting diode technology provides the foundation for modern smart lighting due to high efficiency, long lifetime, compact form factor, and precise controllability [17]. Unlike incandescent bulbs with inherent colour temperature, LED systems can implement tunable white or full RGB colour capabilities through appropriate multi-channel designs.

LED driver circuits must provide constant current rather than constant voltage to ensure stable light output and prevent thermal runaway. Switching-mode power supplies offer higher efficiency compared to linear regulators, essential for managing heat in compact bulb housings [18]. Pulse-width modulation enables dimming control whilst maintaining colour consistency, though high-frequency switching is necessary to avoid visible flicker that can cause eye strain and headaches.

Thermal management represents a critical challenge in LED bulb design. Junction temperature directly affects LED efficiency, colour characteristics, and lifetime. Effective heat sinking requires consideration of thermal resistance from junction through package and housing to ambient environment [19]. Three-dimensional printed housings must account for material thermal conductivity limitations, potentially requiring metal inserts or increased surface area for adequate heat dissipation.

2.3 Embedded Systems and IoT Platforms

The ESP32 microcontroller has emerged as a popular platform for IoT development, integrating dual-core processor, Wi-Fi and Bluetooth connectivity, and rich peripheral set in a low-cost package. The Espressif IoT Development Framework (ESP-IDF) provides professional-grade development environment with real-time operating system support, enabling complex multi-threaded applications [20].

Wireless communication protocol selection significantly impacts system architecture. Wi-Fi provides high bandwidth and straightforward integration with existing network infrastructure but consumes substantial power. Bluetooth Low Energy offers lower power consumption suitable for battery-operated devices but with reduced range and throughput. MQTT protocol has become standard for IoT messaging, providing lightweight publish-subscribe architecture suitable for resource-constrained devices [21].

Real-time operating systems enable concurrent task execution essential for responsive IoT devices. FreeRTOS, integrated within ESP-IDF, provides task scheduling, inter-task communication, and synchronisation primitives. Proper task design prevents priority inversion and ensures time-critical operations such as communication protocol handling and dimming control meet timing requirements [22].

2.4 Machine Learning on Embedded Devices

TinyML represents emerging field applying machine learning to microcontroller-class devices with severe resource constraints. Traditional deep learning models requiring gigabytes of memory and billions of operations per inference are impractical for embedded deployment. TinyML emphasises model compression, quantisation, and algorithm selection appropriate for devices with kilobytes of RAM and megahertz-range processors [5].

Suitable algorithms for on-device learning in smart bulb context include time-series analysis for pattern detection, clustering algorithms for grouping similar behaviour periods, and simple neural networks for classification tasks. Decision trees and random forests offer interpretability advantages, allowing users to understand learned behaviours. Online learning approaches enable continuous adaptation without retraining complete models [23].

Pattern recognition in user behaviour data involves identifying recurring temporal patterns such as daily wake-up times, bedtimes, and periods of room occupancy. Hidden Markov Models and dynamic time warping have been successfully applied to activity recognition tasks with computational requirements suitable for embedded implementation [24]. For this project, hybrid approach combining rule-based heuristics with lightweight statistical learning is most appropriate given computational constraints.

2.5 Health Data Integration and Privacy

Apple's HealthKit framework provides standardised API for accessing health and fitness data stored on iOS devices, with strong privacy protections requiring explicit user permission for each data type accessed. Sleep analysis data includes sleep periods, sleep stages, and sleep quality metrics that can inform lighting automation decisions [8].

Privacy considerations are paramount when processing health data. The General Data Protection Regulation (GDPR) and similar privacy frameworks establish requirements for data minimisation, purpose limitation, and user consent. Storing and processing data locally on user-controlled devices rather than cloud servers provides strongest privacy protection whilst enabling personalised functionality [25].

Research on privacy-preserving machine learning demonstrates techniques such as federated learning and differential privacy that enable learning from user data without exposing individual information. For single-user device context, on-device learning inherently provides privacy, but system design must prevent inadvertent data leakage through network communications or logging [26].

2.6 Circadian Rhythm and Lighting Effects

Human circadian rhythms are primarily entrained by light exposure patterns, with short-wavelength blue light having strongest effect on circadian phase shifting and melatonin suppression. Evening exposure to blue-enriched light delays circadian phase, making sleep onset more difficult, whilst morning blue light exposure advances circadian phase and improves alertness [1].

Colour temperature of lighting significantly affects these biological responses. Warm white light (2700-3000K) contains less blue spectrum and is appropriate for evening use, whilst cool white light (5000-6500K) mimics daylight and supports morning alertness. Dynamic adjustment of colour temperature throughout the day, known as human-centric lighting, aims to support natural circadian rhythms [27].

Light intensity also plays important role, with higher illuminance levels promoting alertness and lower levels facilitating transition to sleep. Gradual dimming over 30-60 minutes before intended bedtime can signal approaching sleep period, whilst gradual brightening simulating dawn has been shown to facilitate gentler awakening compared to abrupt lighting changes or alarm sounds [4].

2.7 Summary of Key Findings

Literature review reveals that whilst smart lighting technology is mature and commercially available, existing solutions generally lack sophisticated on-device learning capabilities and rely heavily on cloud connectivity. Integration of health data for lighting automation is emerging area with limited existing implementations, particularly for systems maintaining local data processing.

Technical feasibility of the project is supported by availability of appropriate microcontroller platforms (ESP32), well-documented frameworks for embedded development and TinyML, and established APIs for health data access (HealthKit). Key challenges identified include thermal management in compact form factor, implementation of effective machine learning algorithms within embedded constraints, and creation of intuitive user interfaces that make automation transparent and controllable.

The project addresses identified gaps by combining on-device AI, health data integration, and privacy-preserving architecture in unified system built from first principles. This approach provides opportunity to explore full system stack whilst creating practical device that demonstrates principles of adaptive, user-respecting home automation.

3 Requirements Analysis and System Design

3.1 Functional Requirements

3.1.1 Core Functional Requirements

FR1 - Basic Lighting Control: The system shall provide adjustable LED brightness from 0-100% with smooth dimming transitions. Colour temperature adjustment shall be supported across warm white (2700K) to cool white (6500K) range with minimum 100K adjustment granularity.

FR2 - Wireless Connectivity: The device shall support both Wi-Fi (IEEE 802.11 b/g/n) and Bluetooth Low Energy connectivity for communication with control applications. Initial device configuration shall be possible via Bluetooth when Wi-Fi credentials are not yet configured.

FR3 - Schedule Management: Users shall be able to define time-based schedules specifying desired brightness and colour temperature at particular times. Schedules shall support different configurations for different days of the week to accommodate varying routines.

FR4 - Behaviour Learning: The system shall automatically detect recurring patterns in user manual adjustments and room usage, building internal model of typical user behaviour. Pattern detection shall operate continuously without requiring explicit training mode or user interaction.

FR5 - Adaptive Automation: Based on learned behaviour patterns, the system shall automatically adjust lighting at predicted times without manual intervention. Automation shall be gradual and non-disruptive, avoiding abrupt changes that might startle users.

FR6 - Health Data Integration: iOS application shall request user permission to access HealthKit sleep data. Upon receiving sleep period data, the system shall automatically initiate wind-down mode before typical bedtime and wake-up mode before typical wake time.

FR7 - Wind-Down Mode: When approaching sleep period, the system shall gradually reduce brightness and shift colour temperature toward warm white over configurable duration (default 30 minutes). Final light level shall be dim warm white suitable for sleep preparation.

FR8 - Wake-Up Mode: Before wake time, the system shall gradually increase brightness and shift colour temperature from warm to cool white over configurable duration (default 20 minutes), simulating natural sunrise to facilitate gentle awakening.

FR9 - Manual Override: All automatic behaviours shall be immediately overridable by manual user adjustment through any control interface. Manual overrides shall not disrupt long-term learning but shall take precedence over automated actions.

FR10 - Cross-Platform Control: The system shall be controllable via native iOS application and web-based Progressive Web App, with feature parity between platforms except for HealthKit integration which is iOS-specific.

3.1.2 Additional Functional Requirements

FR11 - Away Mode: The system shall support away mode that varies lighting based on learned patterns to simulate occupancy when users are absent, with appropriate random variations to appear natural.

FR12 - Energy Monitoring: The system shall track and report energy consumption over time, allowing users to understand usage patterns and potential savings from adaptive automation.

FR13 - Local Operation: All core functionality including learning, automation, and basic control shall operate without internet connectivity, relying only on local network connection between device and control applications.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

NFR1 - Response Time: User commands from control applications shall result in visible lighting changes within 200 milliseconds to maintain perception of direct control and immediate responsiveness.

NFR2 - Dimming Smoothness: Brightness and colour temperature transitions shall execute smoothly without visible steps or flicker. PWM frequency shall exceed 500 Hz to prevent perceptible flicker under all conditions including camera recordings.

NFR3 - Learning Speed: Behaviour patterns occurring on consecutive days for one week shall be detected and incorporated into automation model. Less frequent patterns shall be detected within appropriate timeframe based on occurrence frequency.

NFR4 - Power Efficiency: The system shall achieve minimum 80% power conversion efficiency from mains to LED output. Standby power consumption when light is off shall not exceed 0.5W.

3.2.2 Reliability and Safety Requirements

NFR5 - Electrical Safety: The design shall maintain minimum 4mm creepage distance between mains voltage circuits and low-voltage control circuits. All exposed conductive parts shall be grounded or double-insulated.

NFR6 - Thermal Protection: LED junction temperature shall not exceed manufacturer's maximum rating during continuous operation at maximum output. The system shall implement thermal throttling reducing output if temperature limits are approached.

NFR7 - Fault Tolerance: The system shall recover automatically from communication failures, power interruptions, and software crashes without requiring manual intervention. Persistent storage shall preserve learned behaviours and user settings across power cycles.

NFR8 - Data Privacy: All user behaviour data and health information shall be stored locally on user-controlled devices. No user data shall be transmitted to external servers without explicit user consent for specific purposes.

3.2.3 Usability Requirements

NFR9 - Setup Simplicity: Initial device configuration shall be completable by non-technical users within 5 minutes following clear instructions. No network infrastructure changes (router configuration, port forwarding) shall be required.

NFR10 - Interface Intuitiveness: Control applications shall follow platform-specific design guidelines (Human Interface Guidelines for iOS, Material Design for web). Common lighting adjustments shall be accomplishable within two taps or clicks.

NFR11 - Automation Transparency: Users shall be able to view what patterns the system has learned and why specific automated actions are occurring. Learned behaviours shall be explicable in natural language rather than opaque model outputs.

3.3 System Architecture

The system architecture follows layered approach separating hardware, firmware, and application concerns whilst enabling communication across layers through well-defined interfaces.

3.3.1 Hardware Architecture

The hardware architecture centres on ESP32 microcontroller managing LED drivers, wireless communications, and interfacing with power supply. Power input accepts 110-240V AC mains voltage, with isolated power supply providing regulated low-voltage DC (12V or 24V) for LED strings and 3.3V for control electronics.

LED driver circuit consists of constant-current buck converter topology, with PWM dimming control from ESP32 GPIO pins. Separate channels enable independent control of warm white and cool white LED strings, allowing colour temperature adjustment through relative intensity control. For future RGB implementations, additional channels would be incorporated with appropriate driver circuits.

Thermal management employs aluminium heat sink thermally coupled to LED circuit board, with 3D-printed housing designed to maximise convection airflow whilst maintaining safe enclosure of electrical components. Temperature sensor monitors heat sink temperature, enabling firmware-based thermal protection.

Wireless communication utilises ESP32 integrated Wi-Fi and Bluetooth transceivers with appropriate antenna design for acceptable signal strength within typical room dimensions. External antenna connection point is provided for troubleshooting range issues during development.

3.3.2 Firmware Architecture

Firmware architecture implements multi-threaded design using FreeRTOS, with separate tasks for communication handling, learning algorithm execution, dimming control, and sensor monitoring. Task priorities ensure time-critical operations (dimming control, communication responses) preempt background tasks (pattern learning, logging).

Communication task manages both Wi-Fi and Bluetooth connections, parsing incoming commands and publishing device state updates. MQTT protocol is used for Wi-Fi communication, with custom BLE service for Bluetooth. State changes from any source propagate through internal event system to ensure consistency.

Learning task periodically analyses stored behaviour data, updating internal models of typical patterns. Execution occurs during low-priority intervals to avoid interfering with real-time control. Persistent storage in flash memory preserves learned models across power cycles.

Dimming control task generates PWM signals controlling LED driver, with smooth transitions implementing gradual changes over specified durations. Anti-flicker algorithms ensure stable output even during rapid updates from learning or manual control.

3.3.3 Application Architecture

iOS application implements native SwiftUI interface following Apple Human Interface Guidelines, with HealthKit integration for sleep data access. Application communicates with device via both Bluetooth (for initial setup) and Wi-Fi (for ongoing control). Local network discovery uses mDNS to automatically locate devices without requiring manual IP address entry.

Web Progressive Web App implements responsive interface using React framework, compatible with modern browsers on desktop and mobile platforms. WebSocket connection provides real-time bidirectional communication with device, updating interface immediately when device state changes due to automation or control from other interfaces.

Shared backend service implemented in Node.js provides WebSocket server, MQTT broker, and coordination between multiple devices and clients. This component runs locally on user's network rather than cloud server, maintaining privacy whilst enabling multi-device coordination if implemented.

3.4 Use Case Scenarios

3.4.1 Use Case 1: Initial Setup and Configuration

User installs smart bulb in standard E27 socket and powers on. Device enters setup mode, advertising Bluetooth service. User opens iOS or web application, which automatically discovers nearby unconfigured device. User selects device and enters home Wi-Fi credentials through application interface. Device connects to Wi-Fi network and confirms successful connection. Application guides user through optional HealthKit permission grant, explaining how sleep data will be used. Setup completes with device ready for use, having taken approximately 3 minutes.

3.4.2 Use Case 2: Daily Adaptive Behaviour

User maintains regular weekday routine, waking at 7:00 AM and going to sleep around 11:00 PM. Over first week, system observes manual lighting adjustments: user turns on light at full brightness with cool white at 7:00 AM, reduces to 60% warm white around 9:00 PM, and turns off around 11:15 PM. After one week, system detects recurring pattern and begins automation. At 6:40 AM, light gradually increases from off to full cool white over 20 minutes, completing at 7:00 AM. At 10:45 PM, light automatically dims from current level to 30% warm white over 30 minutes. User can override at any time by manual adjustment, but typically allows automation to proceed as it matches desired routine.

3.4.3 Use Case 3: Weekend Routine Variation

User typically sleeps later on weekends, waking around 9:00 AM instead of 7:00 AM. System observes different Saturday and Sunday patterns compared to weekday patterns. After three weekends, system maintains separate learned schedules for weekdays versus weekends, automatically adjusting wake-up lighting to 9:00 AM on Saturday and Sunday. User appreciates not being woken early by automated lighting on days off work.

3.4.4 Use Case 4: Health Data Enhanced Automation

User grants HealthKit permission. System monitors sleep data and detects that actual sleep onset varies between 10:45 PM and 11:30 PM throughout week, with wake times varying between 6:45 AM and 7:15 AM. Rather than using fixed schedule, system adjusts wind-down timing based on typical sleep onset time for that day of week, starting gradual dimming 30 minutes before expected sleep. Wake-up lighting completes at typical wake time, providing gentle natural light before alarm sounds. This health-data-informed automation proves more effective than fixed schedule at supporting consistent sleep patterns.

4 Experimental Methods and Development Work to Date

4.1 Development Environment Setup

iOS development environment utilises Xcode 26.1.1 on macOS, with iOS 26.1 SDK enabling latest SwiftUI features and HealthKit capabilities. Physical iPhone and iPad devices have been configured for development and testing, with proper iOS Bluetooth permissions (NSBluetoothAlwaysUsageDescription) configured in Info.plist for BLE communication. Developer certificates and provisioning profiles have been configured for on-device testing across multiple physical devices.

Version control repository has been established using Git, with separate branches for embedded firmware specification, iOS application, and backend server. Documentation directory maintains design decisions, BLE protocol specifications, and development notes. Regular commits ensure work is preserved and enable rollback if experimental approaches prove unsuccessful.

Flask-based backend server development environment established with Python virtual environment, SQLite database for user and device management, and SMTP integration for email verification. Local development server configured to accept connections from physical iOS devices on local network.

4.2 Hardware Development Progress

4.2.1 Component Selection and Procurement

ESP32-WROOM-32D module selected as final production component, offering integrated antenna, FCC/CE certification, and established supply chains. A dedicated BLE service has been configured using a universally unique identifier, UUID. One service UUID defines the overall control service, and five characteristic UUIDs handle power, brightness, RGB colour control, mode selection, and status reporting.

4.3 Firmware Development Progress

4.3.1 Wireless Communication Implementation

Bluetooth Low Energy communication protocol fully specified using CoreBluetooth framework specifications. BLE GATT server architecture defined with custom service for device control. Characteristics defined for brightness (0-255 range), RGB colour (three-byte values), power state (boolean), lighting mode (0=Solid, 1=Fade, 2=Rainbow, 3=Pulse), and status monitoring, with read/write permissions and notification support for bidirectional communication. BLE will be used for initial device discovery, pairing, and real-time control from iOS application.

4.4 Software Application Development Progress

4.4.1 iOS Application Development

SwiftUI-based iOS application implements comprehensive device management and control interface. The application features secure user authentication with email verification using six-digit codes sent via SMTP, password reset functionality with timed code expiration, and session management with persistent login state. User credentials stored securely on Flask backend server with Secure Hash Algorithm, SHA-256 password hashing.

Device management interface provides BLE scanning for nearby smart bulbs with real-time device discovery, custom device naming and room assignment, persistent storage of user's devices linked to account, and separation between simulated and physical hardware devices. Dual-mode architecture implemented supporting both physical ESP32 hardware and software simulator, enabling complete development and testing without requiring physical bulbs. Simulator mode toggle in Settings view allows seamless switching between modes, with appropriate filtering of devices based on current mode.

4.5 Control Interface and Communication

The control interface provides comprehensive real-time device management through an intuitive SwiftUI-based interface. The power state control utilises a toggle switch for immediate on/off functionality, while brightness adjustment is managed through a slider control operating across the full 0-255 range with real-time percentage display calculated as `(brightness/255) × 100`. RGB colour selection is implemented via SwiftUI's native `ColourPicker` component, which converts selected colours to 8-bit RGB values (0-255) for transmission to the device. Four quick-access preset colours (White, Red, Green, Blue) are provided through the `QuickColourButton` component for common lighting scenarios. The lighting effect system supports four distinct modes: Solid (mode 0), Fade (mode 1), Rainbow (mode 2), and Pulse (mode 3), each selectable through the `EffectButton` visual interface. All control actions trigger immediate state updates with visual feedback through the `BulbVisualView` component, which dynamically renders the bulb's current state using radial gradients and opacity adjustments based on RGB values and brightness levels.

Device state synchronisation is maintained through the `BulbState` structure, which tracks power status, brightness level, RGB colour values, and active effect mode. The interface continuously reflects current bulb status, with the `BulbVisualView` component providing real-time visual representation of the device state through dynamic colour blending and animated transitions.

4.6 Bluetooth Low Energy Communication

BLE communication is implemented using Apple's CoreBluetooth framework within the `BLEManager` class, which serves as both `CBCentralManagerDelegate` and `CBPeripheralDelegate`. The implementation requires proper iOS permission configuration through `NSBluetoothAlwaysUsageDescription` in the application's Info.plist file, providing users with clear explanation of Bluetooth usage during permission requests.

Device discovery is initiated through `centralManager.scanForPeripherals(withServices:)` targeting the specific service UUID, which must match the ESP32 device configuration. Discovered devices are added to the `discoveredBulbs` array with metadata including device name, RSSI signal strength, and unique peripheral identifier. Connection management is handled through `centralManager.connect(_:options:)` with automatic peripheral delegate assignment and service discovery initiation upon successful connection.

The communication protocol defines five BLE characteristics for bidirectional data exchange: Power Control for single-byte write for power state (0x00/0x01), Brightness Control for single-byte write for brightness (0-255), Colour Control for three-byte write for RGB values, Mode Control for single-byte write for effect mode (0-3) and Status Notification for six-byte notification containing complete device state.

Control commands are transmitted using `writeValue(_:for:type:)` with `.withResponse` type to ensure reliable delivery. Status updates are received through characteristic notifications enabled via `setNotifyValue(true, for:)`, allowing the device to push state changes to the application without polling. Connection state monitoring is implemented through delegate callbacks for connection success, disconnection events, and connection failures, with appropriate UI feedback provided through the `bluetoothState` published property.

4.7 Simulator Mode Implementation

To enable development and testing without physical ESP32 hardware, a comprehensive simulator mode has been implemented. The `simulatorMode` flag, stored in `UserDefaults`, determines whether the application operates with simulated or real Bluetooth devices. When simulator mode is enabled, the `BLEManager` generates three persistent simulated bulbs with consistent UUIDs stored across application launches, ensuring database associations remain valid during development.

Simulated bulbs are marked with the `isSimulated` property throughout the application stack, from the `SmartBulb` model through the database schema (via `is_simulated` column in the bulbs table). The backend `/get_bulbs` endpoint filters bulbs based on the `simulator_mode` parameter, ensuring users only see bulbs appropriate for their current mode. Visual indicators distinguish simulated bulbs through orange "play" icons and status labels.

All BLE operations in simulator mode use simulated delays (e.g., 1.0 second connection delay) to replicate real hardware behaviour, while control commands immediately update the local `bulbState` without actual Bluetooth transmission. This architecture allows complete UI/UX testing and backend integration validation on the iOS Simulator, which does not support CoreBluetooth functionality.

Mode switching is handled through the `SettingsView` toggle, which updates `UserDefaults`, posts a `SimulatorModeChanged` notification to all active `BLEManager` instances, and triggers a refresh of the bulb list in `HomeView` to display only mode-appropriate devices.

4.8 Network Communication Architecture

Network communication is implemented using `URLSession` for HTTP requests to a Flask backend server. The `APIConfig` structure uses Swift compiler directives to configure appropriate endpoints: the iOS Simulator uses localhost (`http://127.0.0.1:5000`) for development, while physical devices use the local network IP address (`http://192.168.1.45:5000`) to communicate with the Flask server running on the development machine.

All API endpoints follow a consistent JSON request/response pattern with proper HTTP status codes (200 for success, 404 for not found, 409 for conflicts, 500 for server errors). The application implements comprehensive error handling for network failures, JSON parsing errors, and HTTP error responses, providing appropriate user feedback through published state properties and UI messages.

User authentication is maintained through session management with the current user's email stored in `UserDefaults` under the `currentUserEmail` key. This session token is included in all bulb management API calls (`/add_bulb`, `/get_bulbs`, `/update_bulb`, `/delete_bulb`) to associate devices with the authenticated user account.

4.9 Deployment and Testing

The application has been successfully configured for deployment to physical iOS devices through proper Apple Developer certificate configuration and provisioning profile setup. iOS permission prompts for Bluetooth access function correctly on physical devices, displaying the configured usage description. The `BLEManager` implements proper Bluetooth state monitoring, displaying status messages ranging from ”Ready” (powered on), ”Bluetooth Off”, ”Unauthorized”, ”Not Supported”, to ”Simulator Mode” based on current system state and application configuration.

Network communication has been validated across the local network, with the Flask backend server accepting requests from physical iOS devices and returning appropriate JSON responses. Database operations for user registration, authentication, and bulb management function correctly with proper error handling for edge cases such as duplicate registrations and invalid credentials.

The simulator mode implementation has been extensively tested on iOS Simulator, validating that all UI components, navigation flows, and state management logic function correctly without physical Bluetooth hardware. Mode switching between simulator and real hardware modes has been verified, confirming that bulb lists update appropriately and connection logic follows the correct code paths based on the `simulatorMode` flag.

4.9.1 Backend Server Development

Flask-based RESTful API server implemented with comprehensive endpoint coverage for user and device management. Authentication endpoints provide user registration with email verification, secure login with credential validation, password reset with verification codes, and email availability checking for registration validation. Device management endpoints support adding new bulbs with simulation mode flag, retrieving user's devices filtered by simulator/hardware mode, updating device names and room assignments, and deleting devices from user accounts.

SQLite database schematic implemented with users table storing email and hashed passwords, and bulbs table storing device IDs, names, room assignments, and simulation mode flags with foreign key relationships to users. Database initialisation includes automatic table creation and migration for adding simulation mode column to existing databases.

Email verification system implemented using Gmail SMTP with app-specific password, sending formatted verification codes, and 5-minute code expiration handled by client-side timer. Server runs on local network accepting connections from both localhost (iOS Simulator) and local network IP (physical devices), with Cross Origin Resource Sharing, CORS, configured for development.

4.10 Preliminary Testing and Results

4.10.1 System Integration Testing

End-to-end testing verified iOS application functionality across development and deployment workflow. User authentication tested with successful account creation, email verification, login/logout cycles, and password reset flows. Device management tested with adding simulated bulbs, switching between simulator and hardware modes, editing device details, and removing devices from accounts. Application successfully deployed to physical iPad and iPhone with proper Bluetooth permissions, developer certificates, and network connectivity.

BLE communication architecture validated through simulator mode testing with instant connection to simulated devices, bidirectional state updates, control command processing, and proper disconnection handling. Real-time control interface tested with immediate UI updates, smooth transitions between states, and consistent behaviour across multiple app launches.

Backend server integration tested with successful API communication from physical devices, database persistence across server restarts, concurrent user sessions, and proper error handling for invalid requests. Network configuration validated for both simulator (localhost) and physical device (local network IP) scenarios.

4.11 Challenges Encountered and Solutions

4.11.1 iOS Bluetooth Permission Configuration

Initial deployment to physical iPad revealed Bluetooth state stuck on ”Unknown” despite Bluetooth being enabled on device. Investigation identified missing NSBluetoothAlwaysUsageDescription key in Info.plist, preventing iOS from requesting Bluetooth permission from user.

Solution involved adding proper Info.plist entry with user-facing description: ”This app needs Bluetooth to connect to your smart bulbs”. After adding permission key, completely deleting app from device, and rebuilding, iOS properly displayed permission prompt on launch. After granting permission, Bluetooth state correctly showed ”Ready” and BLE scanning functioned properly.

4.11.2 Network Configuration for Physical Devices

Initial network communication failed when deploying to physical devices despite working in iOS Simulator. Simulator successfully connected to Flask backend at localhost (127.0.0.1:5000), but physical devices received ”Network error” when attempting same connection.

Solution implemented APIConfig structure with compiler directives distinguishing simulator and physical device environments. Simulator builds use localhost address, while physical device builds use Mac’s local network IP address (192.168.1.45:5000). Both Mac and iOS devices must be on same Wi-Fi network. This architecture enables seamless testing across simulator and multiple physical devices without code changes.

4.11.3 Simulator Mode Architecture

Challenge emerged in testing workflow requiring physical hardware for every test, significantly slowing development iteration. Need identified for testing complete application functionality without ESP32 hardware during development phase.

Solution implemented dual-mode architecture with simulator mode flag stored in UserDefaults. When enabled, BLEManager generates simulated devices with consistent UUIDs during scanning, instantly ”connects” to simulated bulbs, and processes control commands by updating local state without hardware communication. Backend database stores is_simulated flag with each device, enabling proper filtering when loading user’s devices. Mode switching in Settings view posts notification triggering refresh of device lists and BLE manager state. This architecture enabled complete development and testing of iOS application features before hardware availability.

4.11.4 Database Mode Filtering

Initial implementation stored all devices without distinguishing simulated from physical bulbs, causing confusion when switching between modes as both device types appeared in device list regardless of current mode setting.

Solution modified database schema to include is_simulated column, updated backend API to accept simulation flag during device addition, and implemented filtered queries returning only devices matching current mode (simulated devices when simulator mode ON, physical devices when OFF). This separation prevents attempting to connect to wrong device type and provides clear user experience when switching between modes.

5 Planned Deliverables and Work Plan

5.1 Expected Deliverables

5.1.1 Primary Deliverables

Functional Prototype: Complete working smart bulb system demonstrating all primary requirements including wireless control, adaptive automation, health data integration, and cross-platform interfaces. Prototype will include properly designed housing, safety-compliant electrical design, and reliable operation over extended testing period.

iOS Application: Native iOS application built in Swift with SwiftUI interface, implementing device control, schedule management, HealthKit integration, and visualisation of learned behaviour patterns. Application will be tested on physical iOS devices and documented with user guide.

Web Application: Progressive Web App providing cross-platform access to device control and configuration. Application will function on modern desktop and mobile browsers, implementing responsive design and real-time communication with embedded device.

Embedded Firmware: Complete ESP32 firmware implementing LED control, wireless communications, pattern learning, automation logic, and safety monitoring. Firmware will be documented with code comments, architecture diagrams, and build instructions.

Hardware Documentation: Comprehensive documentation of hardware design including circuit schematics, component specifications, PCB layout (if implemented), 3D housing models, assembly instructions, and safety analysis. Documentation sufficient for reproduction by technically competent individuals.

Final Report: Comprehensive dissertation documenting project motivation, literature review, requirements analysis, system design, implementation details, testing methodology, results evaluation, and conclusions. Report will follow academic standards with appropriate citations and technical depth.

Demonstration Video: Video demonstrating system operation including setup process, manual control, adaptive automation behaviour, health data integration, and multi-platform control. Video will be suitable for technical and non-technical audiences.

5.1.2 Supporting Deliverables

User Documentation: Quick-start guide and user manual explaining installation, configuration, and operation of the system. Documentation will be written for non-technical users with clear instructions and troubleshooting guidance.

Developer Documentation: Technical documentation for developers wishing to extend or modify the system, including API specifications, protocol descriptions, and extension points for additional features.

Test Results: Comprehensive test results including functional testing, performance measurements, reliability testing, safety verification, and user acceptance testing if time permits.

Source Code Repository: Complete source code for firmware, iOS application, and web application hosted in version control repository with appropriate licensing and contribution guidelines.

5.2 Remaining Work

5.2.1 Hardware Completion Tasks

Housing Design and Fabrication: Finalise 3D modelling of the bulb housing in Autodesk Fusion 360, incorporating thermal management features, secure mounting points for electrical components, and safety enclosures. Print the housing using university 3D printing facilities with high-temperature materials such as PETG or ABS. Perform thermal performance testing under continuous operation and assess mechanical durability including drop and vibration resistance.

PCB Layout: Design a printed circuit board integrating the ESP32 module, LED driver circuit, power supply, and peripheral components. Ensure proper isolation between mains and low-voltage circuits, minimise electromagnetic interference, and optimise thermal performance. Manufacture a small batch of boards for testing and refinement if time and budget allow.

Final Assembly and Testing: Assemble the complete prototype in the printed housing with secure cable management and strain relief. Conduct comprehensive electrical safety, thermal, and reliability testing under varied environmental conditions.

5.2.2 Firmware Completion Tasks

Advanced Learning Algorithms: Refine pattern detection algorithms using real-world test data, improving robustness and accuracy. Extend detection to irregular patterns such as gradual routine shifts and seasonal adjustments.

Optimisation: Profile firmware to identify bottlenecks and implement improvements for reduced power consumption, faster response times, and better memory usage. Implement deep sleep modes for periods of inactivity to minimise standby power.

Error Handling: Enhance error handling with comprehensive logging, graceful degradation for unavailable features, and clear diagnostics accessible through the web interface.

5.2.3 Application Completion Tasks

iOS Application Refinement: Complete remaining UI screens for automation configuration, pattern visualisation, and detailed device settings. Implement proper error handling, loading states, and user feedback. Conduct user interface testing with potential users and gather feedback for refinement.

Web Application Feature Parity: Ensure web application offers equivalent functionality to the iOS application, except for platform-specific HealthKit integration. Refine user interface for both desktop and mobile experiences.

Backend Server Hardening: Improve server reliability, security, and performance. Implement proper authentication for multiple users, enhance logging for troubleshooting, and optimise WebSocket handling for minimal latency.

5.2.4 Testing and Evaluation Tasks

Comprehensive System Testing: Perform structured testing of all functional requirements, documenting test procedures, results, and identified issues. Include edge cases, failure scenarios, and long-term reliability testing.

Performance Evaluation: Measure and analyse metrics including response times, power consumption, learning accuracy, and user satisfaction. Compare results against initial requirements and industry benchmarks where available.

Security Analysis: Review system security considering potential vulnerabilities in wireless communications, firmware updates, and data storage. Implement and document appropriate security measures and limitations.

5.2.5 Documentation Tasks

Technical Documentation: Complete comprehensive technical documentation covering all system aspects sufficient for reproduction, modification, and troubleshooting. Include well-commented source code, architecture diagrams, and design rationale explanations.

User Documentation: Write clear, concise user documentation with appropriate screenshots, diagrams, and step-by-step instructions. Test documentation with non-technical users ensuring comprehensibility.

Booklet: Potentially make a user manual.

Final Report: Write comprehensive documentation with project motivation through implementation to evaluation.

5.3 Work Plan and Timeline

5.3.1 November 2025

- Complete iOS application core functionality for iPhone and iPad, tested on physical devices and simulator.
- Implement simulator mode for ESP32 bulbs to enable seamless testing before physical hardware is available.
- Verify Bluetooth connectivity and discovery in ESP32 simulation mode.
- Refine initial learning algorithms based on simulated usage data.
- Milestone: Bulb design and component selection completed
- Milestone: Initial prototype software via simulator mode

5.3.2 December 2025

- Focus on core ESP32 bulb features: on/off switching, dimming, and basic colour control.
- Begin PCB layout and preparation for manufacturing (subject to budget approval).
- Refine 3D housing concept, preparing for first test prints.
- Continue development during university break period.
- Begin web application development if time permits.

5.3.3 January 2026

- Milestone: Submit mid-term report (January 2nd)
- Exams period: focus on preparation and revision.
- Finalise 3D housing design and begin test prints.
- Manufacture and populate PCB, integrate with ESP32 module.
- Assemble complete bulb prototype with housing and electronics.
- Start designing and adding more core hardware functions (power, AI dimming, colour control) for stability and reliability.

5.3.4 February 2026

- Begin integration of AI and automation features (after core functionality is stable).
- Conduct extended reliability testing and performance evaluation.
- Deploy prototype for real-world usage data collection.
- Analyse results from system and reliability testing; implement improvements.
- Conduct security analysis and apply necessary measures.
- Continue final report writing, focusing on implementation and testing chapters.
- Refine automation algorithms based on actual usage patterns.
- Prepare demonstration materials, including prototype showcase and video.

5.3.5 March 2026

- If time permits add some refinement to processes using threading
- Finalise report writing and editing.
- Conduct final system testing and validation.

5.3.6 April 2026

Prepare and rehearse presentation materials.

Project demonstration and presentation.

Submit final report.

5.4 Risk Assessment

5.4.1 Technical Risks

Risk 1 - Thermal Management Inadequate: Heat sink design may prove insufficient for reliable continuous operation at maximum output.

Mitigation: Conservative thermal design with margin above minimum requirements. Thermal throttling implemented in firmware reducing output if temperature limits approached. Larger heat sink specified if initial testing reveals issues.

Risk 2 - Learning Algorithm Ineffective: Pattern detection may fail to identify useful patterns or produce excessive false positives disrupting rather than enhancing user experience.

Mitigation: Extensive testing with simulated and real usage data before deploying automation. Conservative confidence thresholds preventing premature automation. Manual override always available allowing users to disable automation if unsatisfactory.

Risk 3 - Wi-Fi Reliability Issues: Wireless connectivity may prove unreliable in real-world environments with interference, weak signals, or problematic router configurations.

Mitigation: Robust connection handling with automatic reconnection and clear status indication. Bluetooth fallback enabling local control if Wi-Fi unavailable. Comprehensive testing across various network conditions.

5.4.2 Project Management Risks

Risk 4 - Component Availability: Electronic components may become unavailable due to supply chain issues common in current market conditions.

Mitigation: Order critical components early in project timeline. Identify alternative compatible components as backup options. Maintain flexible design allowing component substitution without major redesign.

Risk 5 - Schedule Overrun: Project scope may prove larger than anticipated, causing delays and potentially incomplete deliverables.

Mitigation: Realistic timeline with clearly defined primary and secondary objectives.

Primary objectives prioritised ensuring core functionality completed even if time constraints prevent secondary enhancements. Regular progress review with supervisor enabling early identification of schedule issues.

Risk 6 - PCB Manufacturing Delays: If custom PCB manufactured, delays or manufacturing defects could impact timeline.

Mitigation: Early PCB submission maximising available time for manufacturing. Continued development using breadboard prototypes if PCB delayed. Alternative vendors identified as backup if primary manufacturer experiences issues.

5.4.3 Safety Risks

Risk 7 - Electrical Safety Failure: Design or manufacturing defect could create electrical shock or fire hazard.

Mitigation: Conservative safety design following established principles even without formal certification. Thorough safety testing including isolation verification, ground continuity, and thermal stress testing. Prototype operation supervised, not deployed in unsupervised residential use until safety thoroughly validated.

Risk 8 - Thermal Safety: Inadequate thermal management could cause excessive temperatures creating fire risk.

Mitigation: Thermal protection in firmware automatically reduces output if temperature limits approached. Thermal fuses as last-resort protection against runaway temperature. Housing materials selected for appropriate temperature tolerance. Testing under worst-case conditions validates thermal safety margins.

6 Estimated Total Cost

6.1 Hardware Components

- ESP32-WROOM-32D Module (×2 for development): £9.00
- Dual-channel COB LED Module (2700K/6000K): £8.00
- MOSFET PWM Control Module: £9.00
- LED Thermal Conductive Tape: £6.00
- Aluminium Heat Sink: £10.00
- E27 Lamp Base: £5.00
- Temperature Sensor (Thermistor): £1.00

Hardware Subtotal: £46.00

6.2 Materials and Fabrication

- 3D Printing Filament (PETG, 0.5kg): £12.00
- Thermal Interface Material: £3.00
- Assembly Supplies: £8.00

Materials Subtotal: £23.00

6.3 Software and Services

- Apple Developer Account (already owned): £0.00
- Cloud Services (not required due to local architecture): £0.00
- Development Software (all open-source or freely available): £0.00

Software Subtotal: £0.00

6.4 Documentation and Presentation

- Report Printing and Binding: £15.00
- Demonstration Materials: £5.00

Documentation Subtotal: £20.00

6.5 Contingency

- Component Replacements (damaged parts, alternatives): £15.00
- Additional Testing Materials: £10.00

Contingency Subtotal: £25.00

6.6 Total Project Budget

Estimated Total Cost: £116.00

This budget represents realistic estimate for completing primary project objectives. Most significant costs are electronic components and PCB manufacturing if pursued. 3D printing costs include potential filament needed. Software costs eliminated through open-source tools and existing Apple Developer account. I have also included printing my final report and therefore the printing costs. Contingency allocation provides buffer for unexpected requirements or component replacements.

Back to TOC

7 Plan of Final Report Structure

7.1 Proposed Chapter Structure

The final dissertation will follow standard Electronic Engineering project report structure, expanding upon this mid-term report with complete implementation details, comprehensive testing results, and thorough evaluation.

Chapter 1: Introduction

- Project motivation and context
- Problem statement and significance
- Aims and objectives
- Project scope and constraints
- Report structure overview
- Contributions and achievements

Chapter 2: Background and Literature Review

- Smart home automation systems
- LED technology and driver circuits
- Embedded systems and IoT platforms
- Machine learning on embedded devices
- Health data integration and privacy
- Circadian rhythm and lighting effects
- Summary of existing approaches and identified gaps

Chapter 3: Requirements Analysis and System Design

- Functional and non-functional requirements

- System architecture overview
- Hardware architecture and component selection
- Firmware architecture and software design
- Application architecture and interface design
- Communication protocols and data flows
- Security and privacy considerations
- Use case scenarios

Chapter 4: Implementation

- Hardware implementation details
- Circuit design and PCB layout
- Housing design and thermal management
- Firmware implementation
- LED control and PWM generation
- Wireless communication implementation
- Learning algorithm implementation
- iOS application development
- Web application development
- Backend services implementation
- Implementation challenges and solutions

Chapter 5: Testing and Evaluation

- Testing methodology and procedures
- Functional testing results
- Performance evaluation

- Power consumption analysis
- Thermal performance testing
- Learning algorithm evaluation
- Usability testing
- Security analysis
- Reliability and long-term testing
- Comparison with requirements

Chapter 6: Results and Discussion

- System performance summary
- Learning algorithm effectiveness
- User experience evaluation
- Energy efficiency analysis
- Comparison with existing solutions
- Limitations and constraints
- Success criteria assessment
- Discussion of findings

Chapter 7: Conclusions and Future Work

- Summary of achievements
- Objectives fulfilment assessment
- Contributions to field
- Lessons learned
- Future enhancements and extensions
- Alternative approaches
- Broader implications
- Final remarks

7.2 Anticipated Contributions

This project makes several contributions to the intersection of embedded systems, IoT, and home automation:

Privacy-Preserving Smart Home Design: Demonstrates feasibility of sophisticated home automation without cloud dependency, processing all user data locally whilst maintaining full functionality. This approach addresses growing privacy concerns in consumer IoT devices.

On-Device Learning Implementation: Practical implementation of lightweight machine learning on resource-constrained embedded platform, showing that adaptive behaviour does not require powerful servers or extensive computational resources. Provides reference implementation for TinyML applications in home automation context.

Health Data Integration: Novel integration of health platform data (HealthKit) with physical home automation device, creating feedback loop between physiological data and environmental control. Demonstrates potential for health-aware smart home systems.

Complete System Development: End-to-end implementation from hardware design through embedded firmware to cross-platform applications provides comprehensive case study in IoT system development, valuable for education and future projects.

Open Design Approach: Comprehensive documentation and open-source code release enables reproduction, modification, and extension by others, contributing to open hardware and software communities.

8 Conclusions

8.1 Progress Summary

Significant progress has been made during the first term across key project areas. Hardware component selection and procurement have been completed successfully. Initial prototyping of the ESP32-based bulb has validated core circuit designs and highlighted areas for thermal management that will require attention in later stages.

Embedded firmware development has established core functionality, including LED control and wireless Bluetooth communication. A simulator mode for ESP32 bulbs has been implemented, enabling testing of the iOS application before physical hardware is available. Initial pattern learning algorithms have been prototyped using simulated usage data.

Software development has produced a functional iOS application running on both iPhone and iPad devices, tested on physical devices and simulators. Communication infrastructure has been verified in simulator mode, demonstrating that the app can detect and interact with ESP32 devices over Bluetooth.

Preliminary testing has confirmed that the core system functions correctly. Areas identified for further refinement include thermal performance of the housing and optimisation of the learning algorithms. The project remains on schedule, with a clear plan for completing remaining objectives within the available timeframe.

8.2 Assessment of Progress Against Timeline

Project progress is aligned with the timeline established at the start of the project. November milestones, including component selection, iOS application core development, and initial simulator testing, have been achieved.

Software development has progressed efficiently due to prior experience and the use of established frameworks, allowing early verification of key functionality. Hardware development has proceeded with expected challenges, particularly relating to thermal management, which have been taken into account when selecting components for future build.

The mid-term report submission in early January represents a key milestone, providing an opportunity to review progress and refine priorities for the remaining work. Overall, the project is progressing as planned and is well-positioned to complete core functionality and subsequent AI feature integration on schedule.

8.3 Key Learnings

Several important lessons have emerged from work completed to date:

Thermal Management Awareness: Even at early prototyping stages, considering heat generation from the ESP32 and LED circuitry is important. Initial tests on simulated hardware have highlighted the need to monitor temperature and plan housing ventilation for future prototypes.

Safety Considerations: Awareness of electrical safety principles has been reinforced, particularly regarding low-voltage circuit design and proper component selection. Using pre-certified power supply modules simplifies safe hardware development for student projects.

Embedded Resource Constraints: Programming the ESP32 microcontroller requires attention to memory and processing limitations. Writing efficient firmware and planning algorithm implementation with limited resources are critical skills that have been applied in initial development.

Communication Protocol Selection: Implementing Bluetooth Low Energy (BLE) in the simulator and iOS application early demonstrated the importance of selecting appropriate protocols for device discovery and control. Understanding BLE GATT services and characteristics has guided the design of the communication interface.

Value of Prototyping: Developing functional prototypes in both hardware simulator and iOS application form has enabled early testing and validation of core functionality. Early experimentation has helped identify potential issues with device connectivity and LED control before physical hardware is available.

8.4 Next Steps

Immediate priorities for January focus on finalising the mid-term report and completing core hardware development. The 3D housing design will be finalised and initial test prints conducted to validate thermal performance and mechanical fit. PCB manufacturing and component population will proceed once budget approval is confirmed, followed by assembly of the complete ESP32 bulb prototype. Core functionality including on/off control, dimming, and colour adjustment will be tested for stability and reliability.

Firmware development will continue with refinement of learning algorithms based on simulated and initial real-world usage data. Attention will be given to error handling, recovery mechanisms, and diagnostic capabilities to ensure reliable operation. Integration of AI and advanced automation features will commence once core bulb functionality is stable.

Application development will focus on refining the iOS interface and testing simulator mode functionality. Optional web application development will continue if time permits, ensuring parity with the iOS app where possible. Backend server reliability and security will be reviewed and improved as required.

February priorities include extended system testing, real-world deployment of the prototype, and analysis of collected usage data to inform final refinements. Automation algorithms will be adjusted based on observed behaviour patterns, and security analysis will be completed. Final report writing will continue, with implementation and testing chapters completed during this period.

March will focus on final report completion, system validation, preparation of demonstration materials, and presentation. Remaining work is well-defined, and progress to date indicates that the project objectives are achievable within the available timeframe, providing a fully functional adaptive smart bulb system with privacy-preserving design principles and end-to-end hardware-to-software integration.

References

[1] C. Cajochen, S. Frey, D. Anders, J. Späti, M. Bues, A. Pross, R. Mager, A. Wirz-Justice, and O. Stefani, “Evening exposure to a light-emitting diodes (LED)-backlit computer screen affects circadian physiology and cognitive performance,” *Journal of Applied Physiology*, vol. 110, no. 5, pp. 1432–1438, May 2011. ↑↑

[2] E. Zeng, S. Mare, and F. Roesner, “End user security and privacy concerns with smart homes,” in *Proceedings of the Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, Santa Clara, CA, USA, Jul. 2017, pp. 65–80. ↑

[3] A.-M. Chang, D. Aeschbach, J. F. Duffy, and C. A. Czeisler, “Evening use of light-emitting eReaders negatively affects sleep, circadian timing, and next-morning alertness,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 4, pp. 1232–1237, Jan. 2015. ↑

[4] M. C. Giménez, M. Geerdinink, P. Verstegen, Y. Mesman, Y. de Vries, Y. A. de Kort, W. D. van Marken Lichtenbelt, and A. L. Schlangen, “Evaluating the circadian performance of a polychromatic white LED light source,” *Lighting Research & Technology*, vol. 49, no. 5, pp. 586–600, Aug. 2017. ↑↑

[5] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, 1st ed. Sebastopol, CA, USA: O’Reilly Media, 2019. ↑↑

[6] Espressif Systems, *ESP-IDF Programming Guide*, 2023. Available: https://docs.espressif.com/projects/esp-idf/. [Accessed: Nov. 15, 2025]. ↑

[7] C. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J.-S. Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, “Benchmarking TinyML systems: Challenges and direction,” *arXiv preprint arXiv:2003.04821*, 2021. ↑

[8] Apple Inc., *HealthKit Framework Documentation*, 2023. Available: https://developer.apple.com/documentation/healthkit. [Accessed: Nov. 15, 2025]. ↑↑

[9] I. Grigorik, *High Performance Browser Networking*, 1st ed. Sebastopol, CA, USA: O’Reilly Media, 2013. ↑

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. ↑

[11] A. C. Skeldon and D.-J. Dijk, “School start times and daylight saving time confuse California lawmakers,” *Current Biology*, vol. 27, no. 18, pp. R1139–R1140, Sep. 2017. ↑

[12] International Electrotechnical Commission, “IEC 60598-1:2020 Luminaires - Part 1: General requirements and tests,” *IEC Standards*, 2020. ↑

[13] P. P. Ray, “A review on TinyML: State-of-the-art and prospects,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, Apr. 2022. ↑

[14] X. Guo, S. H. Lin, M. R. Hamblin, and J. Xu, “Photobiomodulation in Alzheimer’s disease,” in *Photobiomodulation, Photomedicine, and Laser Surgery*, vol. 38, no. 10, pp. 589–597, Oct. 2010. ↑

[15] C. Gomez, J. Oller, and J. Paradells, “Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology,” *Sensors*, vol. 12, no. 9, pp. 11734–11753, Sep. 2012. ↑

[16] S. Caird and R. Roy, “User-centred improvements to energy efficiency products and renewable energy systems: Research on household adoption and use,” *International Journal of Innovation and Sustainable Development*, vol. 3, nos. 1–2, pp. 77–91, 2008. ↑

[17] E. F. Schubert, *Light-Emitting Diodes*, 2nd ed. Cambridge, UK: Cambridge University Press, 2006. ↑

[18] R. W. Erickson and D. Maksimović, *Fundamentals of Power Electronics*, 2nd ed. New York, NY, USA: Springer, 2007. ↑

[19] A. Christensen and S. Graham, “Thermal effects in packaging high power light emitting diode arrays,” *Applied Thermal Engineering*, vol. 29, nos. 2–3, pp. 364–371, Feb. 2009. ↑

[20] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, “The audacity of fiber-wireless (FiWi) networks: Revisited for clouds and cloudlets,” *China Communications*, vol. 12, no. 8, pp. 33–45, Aug. 2015. ↑

Back to TOC

[21] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S - A publish/subscribe protocol for Wireless Sensor Networks,” in *Proc. 3rd International Conference on Communication Systems Software and Middleware (COMSWARE 2008)*, Bangalore, India, Jan. 2008, pp. 791–798. ↑

[22] R. Barry, *Using the FreeRTOS Real Time Kernel: A Practical Guide*, 1st ed. Seattle, WA, USA: Amazon Digital Services, 2009. ↑

[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, Mar. 2014. ↑

[24] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989. ↑

[25] P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017. ↑

[26] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282. ↑

[27] J. L. Souman, A. M. Tinga, S. C. J. Te Pas, R. van Ee, and B. N. S. Vlaskamp, “Acute alerting effects of light: A systematic literature review,” *Behavioural Brain Research*, vol. 337, pp. 228–239, Jan. 2018. ↑

Appendices

Appendix A: Health and Safety Assessment

Project Title: AI-Based Smart Bulb for Adaptive Home Automation

Student Name: Caleb Ram

Student ID: 6801936

Supervisor: Dr Ahmed Elzanaty

Hazard Identification and Risk Assessment

Hazard 1: Electrical Shock from Mains Voltage

Description: Working with 230V AC mains voltage during power supply integration and testing presents risk of electric shock causing injury or death.

Risk Level: High

Control Measures:

- All mains voltage work conducted with power disconnected except during necessary testing
- Isolation transformer used during mains-connected testing
- One-hand rule observed when probing live circuits
- Insulated tools used exclusively
- Residual Current Device (RCD) protection on test bench
- Appropriate creepage and clearance distances maintained in circuit design
- Double insulation or earthing of exposed conductive parts
- Supervisor or competent person present during initial mains testing

Residual Risk: Low

Hazard 2: Thermal Burns from Heat Sink and LED

Description: LED and heat sink reach elevated temperatures during operation, potentially causing burns on contact.

Risk Level: Medium

Control Measures:

- Warning labels affixed to prototype indicating hot surfaces
- Appropriate cooling-off period before handling after operation
- Thermal testing conducted with temperature monitoring equipment
- Housing design prevents accidental contact with hot components
- Thermal cutoff protection implemented in firmware

Residual Risk: Low

Hazard 3: Fire Risk from Thermal Runaway or Component Failure

Description: Inadequate thermal management or component failure could cause excessive heating leading to fire.

Risk Level: Medium

Control Measures:

- Conservative thermal design with adequate safety margins
- Firmware thermal protection reducing output if temperature limits approached
- Non-flammable housing materials (PETG or ABS with appropriate fire rating)
- Testing conducted on non-flammable surface with fire extinguisher available
- Thermal fuses as redundant overheat protection
- Prototype operation supervised, not left unattended during extended testing

Residual Risk: Low

Hazard 4: Soldering Fumes and Chemical Exposure

Description: Soldering produces fumes containing rosin and flux compounds that may cause respiratory irritation.

Risk Level: Low

Control Measures:

- Fume extraction used during all soldering operations
- Work conducted in well-ventilated area
- Lead-free solder used exclusively
- Frequent breaks during extended soldering sessions

Residual Risk: Very Low

Hazard 5: Eye Damage from LED Light Exposure

Description: Direct viewing of high-brightness LED may cause temporary or permanent eye damage.

Risk Level: Low

Control Measures:

- LED testing conducted with diffuser or indirect viewing
- Avoid direct gaze at illuminated LED
- Warning labels indicating eye safety precautions
- Brightness limited during testing to levels safe for direct viewing

Residual Risk: Very Low

Hazard 6: Sharp Tools and Components

Description: Use of cutting tools, component leads, and heat sink fins may cause cuts or punctures.

Risk Level: Low

Control Measures:

- Appropriate personal protective equipment (safety glasses, gloves when appropriate)
- Proper tool handling and storage procedures
- Component leads trimmed carefully and disposed of properly
- First aid kit readily available in workspace

Residual Risk: Very Low

Declaration

I confirm that I have read and understood this health and safety assessment. I will follow all specified control measures and report any incidents or near-misses to my supervisor immediately. I understand that deviation from approved procedures may result in disciplinary action and project suspension.

Student Signature: ______________________ Date: ____________

Supervisor Signature: ______________________ Date: ____________

Appendix B: Ethics Considerations

Project Title: AI-Based Smart Bulb for Adaptive Home Automation

Ethical Review Assessment

This project involves collection and processing of personal health data through Apple HealthKit integration. The following ethical considerations have been identified and addressed:

Data Privacy and Protection

All health data processing occurs locally on user-owned devices. No health data is transmitted to external servers, cloud services, or third parties. HealthKit permissions requested explicitly with clear explanation of how data will be used. Users maintain complete control over data access and can revoke permissions at any time.

Informed Consent

Application clearly explains what data is collected, how it is used, and what benefits automation provides. Technical language avoided in favour of clear, understandable descriptions. Users must explicitly grant permission before any health data access occurs. Consent can be withdrawn at any time without affecting basic lighting control functionality.

Transparency and Explainability

System makes learned behaviour patterns visible to users through application interface. Users can understand why automation actions occur based on displayed patterns. All automated behaviours can be overridden or disabled by user preference. System does not make autonomous decisions that cannot be explained or controlled by users.

Testing and Evaluation

Any user testing will be conducted with appropriate informed consent documentation. Participants will be provided with information sheets explaining study purpose, procedures, data handling, and right to withdraw. No vulnerable populations will be involved in testing. Testing will be limited to adults capable of informed consent.

Data Security

While data remains local, appropriate security measures implemented including encrypted storage of sensitive configuration data, authentication for remote access if implemented, and secure communication protocols (TLS/SSL) for network communications.

Risk-Benefit Analysis

Primary risk involves inappropriate lighting automation disrupting user comfort or sleep. This risk is mitigated through manual override capabilities, gradual automation engagement, and user control over all system behaviours. Benefits include improved sleep quality, energy efficiency, and convenience, with users making informed decision about acceptable trade-offs.

Assessment Conclusion

No ethics review is required for this project, as it does not involve human participants, human data, human tissue, or any identifiable personal information. All data processing occurs locally on user-controlled devices, and no external transmission of data takes place.

The project design follows privacy-by-design principles and complies with relevant data protection regulations including GDPR principles of data minimisation, purpose limitation, and user consent.

While the system may be shared with a small number of users for informal usability testing, this does not involve collection of personal data or sensitive information, and therefore no formal ethics review is required.

Appendix C: Project Meetings Log

Meeting 1 - Project Selection and Initial Planning

Date: 7 October 2025

Attendees: Caleb Ram, Dr Ahmed Elzanaty

Discussion Points:

- Project proposal presented and discussed
- Scope refinement focusing on achievable objectives within timeframe
- Discussion of technical challenges particularly thermal management

Actions:

- Student: Complete literature review, order initial components, set up development environment
- Supervisor: Provide feedback on project scope and recommended reading materials

Meeting 2 - Progress Review and Technical Discussion

Date: 17 November 2025

Attendees: Caleb Ram, Dr Ahmed Elzanaty

Discussion Points:

- Review of literature findings and technology selection rationale
- Initial hardware prototyping results including thermal concerns
- Discussion of firmware architecture and task priority design
- HealthKit integration approach and privacy considerations
- Literature review scope and key areas to investigate
- Safety requirements and electrical design considerations

- Mid-term report structure and expectations

Actions:

- Student: Continue mid-term report writing
- Supervisor: Write email to Laurence asking about budget and sending components needed.

Meeting 3 - Mid-Term Progress Review

Date: 1 December 2025

Attendees: Caleb Ram, Dr Ahmed Elzanaty

Discussion Points:

- Demonstration of current prototype functionality
- Review of learning algorithm preliminary results
- Discussion of PCB layout considerations versus continued breadboard approach
- Timeline review and risk assessment
- Mid-term report draft sections submitted for review

Actions:

- Student: Act on mid-term report feedback given over Christmas break, and continue begin building ESP32 bulb and then designing PCB if budget permits
- Supervisor: Provide feedback on first draft for Mid-Term Report

Planned Future Meetings

Regular fortnightly meetings scheduled throughout project duration. Next meeting scheduled for 15 December 2025 to review hardware finalisation progress and discuss testing methodology for evaluation phase.

Appendix D: Initial Email Correspondence

The following email correspondence documents initial project proposal and confirmation with project supervisor Dr Ahmed Elzanaty.

Email correspondence from project proposal and setup phase

Specific email content has been incorporated into other sections of the report with a screenshot showing proof of project initialisation with start date.

Appendix E: Component Specifications

Update this when components have all arrived as they arrive

ESP32-WROOM-32D Module

- Processor: Dual-core Xtensa LX6, 240 MHz
- Memory: 520 KB SRAM, 4 MB Flash
- Wireless: Wi-Fi 802.11 b/g/n, Bluetooth 4.2 / BLE
- GPIO: 34 programmable pins
- Peripherals: SPI, I2C, I2S, UART, ADC, DAC, PWM
- Operating Voltage: 3.0 - 3.6 V
- Power Consumption: 80 mA (active), 5 A (deep sleep)

COB LED Module

- Configuration: Dual-channel warm/cool white
- Colour Temperature: 2700K (warm), 6000K (cool)
- CRI: ¿95
- Forward Voltage: 36V per channel
- Forward Current: 300 mA per channel
- Luminous Flux: 800 lm (warm), 900 lm (cool)

- Thermal Resistance: 8 ℃/W junction to board

LM3409 LED Driver

- Topology: Buck (step-down) constant current
- Input Voltage: 6 - 42 V
- Output Current: Up to 1 A (adjustable via resistor)
- Efficiency: ¿90% typical
- Switching Frequency: 750 kHz
- Dimming: PWM and analogue inputs
- Protection: Over-temperature, under-voltage lockout

IRM-20-12 Power Supply

- Type: Enclosed switching AC-DC converter
- Input: 85 - 264 VAC, 47 - 63 Hz
- Output: 12 VDC, 1.67 A (20W)
- Isolation: 3 kVAC input to output
- Efficiency: 84% typical
- Protection: Short circuit, overload, over-voltage
- Safety: UL/cUL, TUV, CE certified