ALA-TOO INTERNATIONAL UNIVERSITY

IT & Business College

Department:
Computer Science(SCA-23C)

Project Title:
Led - Controlled System

Project report:
Group: Team 1(Info-Space)

Team Members:

- **Aidarbek Kachiev (ID238715026)**
- **Beknazar Usenkanov(ID238715025)**
- **Shakhizada Kurmanbekova(ID238715039)**
- **Adelya Dyusheeva(ID238715040)**
- **Amina Nurdinova(ID238715047)**
- **Amantur Sabyrbekov(ID238715033)**

Instructor: Mirlan Nurbekov
Date: 17.12.2025

Table of Contents

Table of figures

Abstract

We designed a Bluetooth-based light control system for household lamps through a mobile application and a physical button. Its purpose was creating a flexible, easy-to-use, multi-user system to enable more people to access home lighting, lessening the burden of more complicated wired infrastructure, which leads to the improvement of convenience for the home. The system works by scanning the range of Bluetooth devices that allow you to choose which lamps to control (via buttons or via the application). An administrator may assign and manage user permissions, monitoring which lamps users may access. It provides reliable connectivity, effective device selection, and secure permission management to enable many users to manage individual lamps within a household environment.

1.Introduction & Background

Introduction to Home Automation Systems. Home Automation refers to technologies for using computers in the home that can manage the home operation in order to do life with online connected devices at your house. It improves comfort, convenience, energy efficiency and other features by allowing user to control the remote devices through mobile apps or another interface.

Wireless Communication Technologies in Smart Homes. Wi-Fi, Zigbee, and Bluetooth serve well as standard wireless communication protocols for smart home systems; Bluetooth presents a direct device-to-device communication approach within a personal area network with low cost and wide access.

Low Cost Solution to a Smart Lighting Solution with Bluetooth. This App is based on a real practical low-cost Application for Smart Lighting Design in the Residential area. Unlike conventional smart home solutions that need a central hub or sophisticated networks, Bluetooth-based systems deliver a wireless interaction between a user's smartphone and home gear.

Related Studies and Current Bluetooth-Based Systems. Studies reveal that Bluetooth Communication Technology enables access to home appliances and lights using smartphone apps, including low-cost hardware like Arduino microcontrollers, in the home via smartphone applications through Bluetooth communication. Similar systems include Arduino-based home automation prototypes with Bluetooth, which are Arduino and Android app-based prototypes for home automation that the user can use to turn the lighting and appliances on and off remotely. For instance, even a standard Bluetooth wireless home automation service that is also low budget requires an Android phone and Arduino board as its main tool for controlling various household appliances. Other designs utilize a Bluetooth smartphone connected Arduino UNO microcontroller to control bulbs and motors using Bluetooth.

Motivation and Novelty of the proposed system. But the proposed project is more basic and multi-user in scope with the management of user consent. Most academic implementations focus on single-user control or extensive IoT-based systems connected through sensors. The system eases the workload for shared household automation, allowing a few people to control lighting by Bluetooth scanning and screen selection and by a user allowing for multiple users of the system to access and monitor their devices and the admin who gives them control. This system eliminates additional networking equipment and complicated integration in cohabitation with other residential facilities.

2.Problem statement & objectives

2.1 What problem our projects solves

Currently, remote control systems often suffer from difficulties such as a limited wireless protocol range, unstable connection states and long pauses and response times, and use of intricate network infrastructure. These challenges make the methods for lighting control less convenient, therefore less reliable and result in ineffective lighting control systems and automation is hindered in smart home and Internet of Things lighting control cases. Additionally, the different protocols and devices might not be well integrated as a whole so this means installing and integrating such solutions into the house is extremely difficult. Hence, our design and implementation of a Bluetooth-powered LED enabled direct lighting control system via mobile app has proved to be a simple solution and offers stability and responsiveness with no need for an active internet connection. Furthermore, it minimizes reliance on external networks and provides the benefits of energy conservation and accessibility into mobile devices through Bluetooth Low Energy technology. Users switch on and off their lights with our remote lighting control app which also allows a real-time monitoring of the state of the lights for state updates. This method increases reliability, convenience and accessibility, whether delivered directly or by remote management and using a mobile application.

2.2 Objectives of the Project

1. Develop a Bluetooth-controlled LED and optionally robotic arm system.
   Build the hardware/software architecture enabling LED modules and/or a mechanical actuator via Bluetooth Low Energy with low power consumption, stable connectivity and modern mobile compatibility. Design a mobile app for device control and state monitoring.
2. Implement an easy to use mobile app for command and control (on/off, brightness adjustment, mode switching) with reading of state in real time, resolving the problem of management complexities and fragmentation of user control with the mobile applications of OEMs.
3. Test the communication range, response time, and operational reliability.
4. Carry out realistic user experimental testing on Bluetooth range, command latency and system stability in real-world settings, taking into account known difficulties of using long-distance networked illumination with remote lighting systems for example delay, connection failure and protocol incompatibility.
5. Evaluate end users' experience and robustness of the system under normal operational conditions for daily utilization cases.
6. Evaluate the potential to enhance daily user interactions by simplifying both manual and remote control and the constraints of a system applicable to home automation and IoT ecosystems.
7. Minimize reliance on cumbersome network infrastructure.
8. Control the lighting system to operate without continuous internet connection, creating accessibility, responsiveness, and reliability.

3.System requirements & constraints

3.1 Functional Requirements (What the system must do)

1. **Wireless Control via Bluetooth Low Energy (BLE)**
   • The system must establish a stable BLE connection with a mobile device.
   • It must receive commands such as ON/OFF, brightness level, mode switching, or actuator movement.

2. **Real-Time Command Execution**
   • The device must respond to user commands with minimal delay.
   • The system must update the mobile app with the current state (LED state, brightness, actuator position).

3. **Local Operation Without Internet**
   • The system must function entirely via local Bluetooth communication and must not require Wi-Fi or cloud access.

4. **Manual Override Control**
   • The hardware must include a manual switch/button for local LED operation in case the mobile device is unavailable.

5. **State Feedback to Mobile App**
   • The controller must transmit real-time status information (power state, sensor/position feedback if a robotic arm is used).

6. **Low Power Operation**
   • The system must minimize energy consumption when idle and during communication using BLE power-saving modes.

3.2 Non-Functional Constraints

BLE:
The system should ensure that indoor Bluetooth Low Energy (BLE) communication can be kept within 5 to 15 meters of a given location. Note that obstacles and interference can limit such range.

Latency:
So that the user can receive a smooth input, make sure that it takes between 100 to 300 ms to act on the current commands. Users are able to sense feedback is instantaneous.

Power consumption:
Keep power consumption in mind. The power consumption per unit must be within the chosen power

supply or battery that the LED driver and BLE modules will consume. That is to say, the BLE module should run in low-energy mode such that less than 10 to 20 mA is delivered when the modules are in transmission, and less than 1 mA when idle.

Constraints on cost:
The final hardware costs should be held within a budget. That is, we need to select pieces, such as inexpensive microcontroller, BLE module, LED driver etc., which can contribute to a good compromise between performance and cost.

Reliability:
Reliability is the last requirement that should be considered: system should also be designed to run for the indefinite duration by overcoming intermittent disconnection in a way which does not affect the user's experience.

3.3 Mechanical Constraints (Size & Weight)

Compact Hardware Layout

• The PCB assembly, LED/actuator assembly needs to fit inside a small enclosure that's suitable for home/light industrial use.
• Suggested enclosure size: 50–120 mm in length according to components.

Lightweight Construction

• The device must be light to facilitate installation (e.g., <150 g in the LED-only design, a little above if part of a robotic arm is included).

Mounting Requirements

• There shall be mounting points as needed to mount the system where it is to be placed (on a desk or within a lamp or on a wall).

Safety Requirements

Electrical Safety
• Depending on the LED driver specifications, maximum operating voltage must not exceed 12–24 V DC.
• As LEDs and actuators run through the operating range, the current must be safe to prevent overheating.

Overcurrent & Thermal Protection

• The components must have fuses, current-limiting resistors, microcontroller shutdown logic to prevent damage.

Protection Moving Parts (if robotic arm or actuator is included)

• The moving elements must be manoeuvred at a safe torque and speed to prevent injury.
• Mechanical stops or limit switches should be used to prevent overextension.

Enclosure Safety

- All components must be enclosed to avoid contact with wires, solder joints, or hot surfaces.
- Ventilation is required if heat dissipation is expected.

4.System design overview

System Architecture

The system is designed as a sequential data-flow chain connecting each of the following elements:

Mobile Application. Bluetooth Communication Module (HC-05). Microcontroller (Arduino). IR Remote Control with IR Receiver (VS1838B). Actuators (LEDs).

Operation:

Users control the LEDs wirelessly, either from the mobile app or IR remote control. The mobile app takes user input and converts it into Bluetooth command packets sent to and received by the HC-05 Bluetooth module. At the same time, commands that come out of the IR remote can be received by the VS1838B IR receiver. Both Bluetooth commands and IR commands are transmitted to the Arduino microcontroller, which reads all input signals and outputs the LEDs output – that is, make you turn them on or off, as well as adjust the light and brightness.

Data Flow:

Mobile app → Bluetooth module → Microcontroller → LEDs

IR remote control → IR receiver → Microcontroller → LEDs

Power Supply:

5V from a power source drives the Arduino board and external links and runs the whole system.

Additional Notes:

There are no other communication devices in use beyond Bluetooth HC-05 and IR receiver. All components work at 5V from Arduino. (The associated System Architecture diagram provides a picture of these connections and data flows.)

Figure 1: System Architecture Diagram

4.1 Hardware design

4.1.1 Hardware Architecture

Main Hardware Blocks

Your system consists of the following key hardware components:

1. **Microcontroller (Main Controller)**
    - Arduino Nano / Arduino Uno / ESP32
    - Responsible for processing Bluetooth commands and generating control signals.
2. **Communication Module (Bluetooth)**
    - **HC-05 / HC-06** (classic Bluetooth) *или*
    - **ESP32** (встроенный BLE модуль)
    - Handles wireless communication with the mobile application.
3. **Power Supply**
    - 5V USB or regulated power source for the microcontroller
    - 3.3V for ESP32 (built-in regulator)
    - **Separate power line** for motors/servo actuators (typically 5–6V)
    - Power distribution:
      • Microcontroller: 5V or 3.3V
      • LEDs: 3.3V or 5V depending on type
      • Servo motors: 5–6V
      • Motors: may require higher current (use drivers)
4. **Sensors (optional)**
    - Button (manual override)
    - Light sensor / limit switch (if servo/motor is involved)
5. **Actuators**

- **LED** (simple digital/PWM output)
- **Servo motor** (PWM control)
- **DC motor** (requires motor driver, e.g., L298N or L293D)

4.1.2 Signal Flow: Inputs → Processing → Outputs

The inputs of the system are the smartphone application and the IR remote control. The smartphone application sends control commands via Bluetooth using the HC-05 module, while the IR remote control transmits infrared signals that are received by the VS1838B IR receiver.

All incoming commands are forwarded to the microcontroller (Arduino), which performs the main processing and decision-making. The microcontroller interprets the received Bluetooth and IR commands and generates the appropriate control signals.

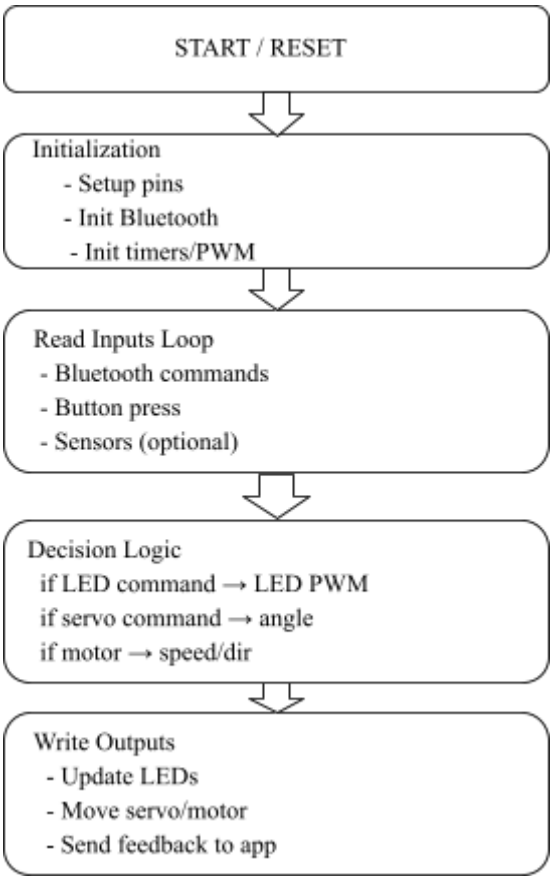
The outputs of the system are the LEDs. Based on the processed commands, the LEDs are switched on or off or their brightness is adjusted using digital and PWM signals. The entire system is powered by a power bank, and all components operate on 5V supplied by the Arduino.

4.1.3 Voltage Levels & Power Notes

- **3.3V**
  • ESP32 logic level
  • Some sensors
- **5V**
  • Arduino logic
  • LEDs
  • Servo motors
- **6V (or external)**
  • DC motors, high-current actuators
- **Important:** Servo motors **must not** be powered directly from Arduino 5V pin → use external 5V supply.

4.2 Software Design

Figure 2: Software Flowchart (Main Logic)



4.2.1 Explanation of Software States

1. Setup / Initialization

- Configure GPIO pins
- Initialize serial/Bluetooth module
- Initialize PWM channels for LED/servo
- Set default actuator states (LED off, servo at initial angle)

2. Reading Inputs

- Read **Bluetooth data packets** (command type + values)
- Read **button input** for manual override
- Read **sensors**, if installed (limit switch, light sensor)

3. Decision Logic

- Parse received command
- Determine if the action is:
  - LED ON/OFF
  - LED brightness level
  - Servo target angle
  - Motor speed/direction
- Check safety limits (angle limits, max brightness, etc.)

4. Writing Outputs

- Set LED output using PWM or digital write
- Move servo/motor to target position/speed
- Send updated state back to the mobile app over Bluetooth
- Loop back to reading inputs

5.Prototype versioning concept

5.1 Prototype 0000 - Basic LED Toggle (pilot test)

Purpose

The initial prototype was developed to verify that the Arduino board can control only one LED. The task was to check basic digital output functionality before introducing additional components or communication modules.

Working Features

- The LED successfully turns ON and OFF.
- The digital output pin (pin 2) operates correctly.
- Basic timing logic (delay) is functional.

```
int ledPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
```

Figure 3: Controls one LED to test basic Arduino output.

What Did Not Work

- Only one LED is supported.
- No interaction methods (no IR remote, no Bluetooth).
- No command logic or state tracking.
- The prototype cannot scale to more LEDs.

5.2 Prototype Version 0001 - Multi-LED Sequential Test

Purpose

This prototype increases the system from one LED to three LEDs.
The goal was to check whether the Arduino could reliably control multiple outputs and maintain correct timing.

```
int leds[3] = {2, 3, 4};

void setup() {
  for(int i = 0; i < 3; i++) {
    pinMode(leds[i], OUTPUT);
  }
}
```

Figure 4: Controls multiple LEDs sequentially using loops.

Fixes from Version 0000

- The limitation of a single LED was removed.
- Array-based LED management was introduced.
- Output control became scalable.

```
void loop() {
  for(int i = 0; i < 3; i++) {
    digitalWrite(leds[i], HIGH);
    delay(300);
    digitalWrite(leds[i], LOW);
  }
}
```

Figure 5: It cycles the first three LEDs on and off in sequence

New Features

- Three LEDs operate independently.
- Sequential activation pattern verifies correct array indexing.
- More complex timing and looping logic.

Still Not Working

- No wireless interfaces.
- No external control (only internal logic).
- No command processing or LED state memory.

5.3 Prototype Version 0002 - IR Remote Control Prototype

Purpose

This release includes NEC IR-remote capability.
That meant wirelessly controlling it and ensuring it will respond to pressed buttons.

Fixes from Version 2

- System no longer runs automatically — LEDs are controlled by user input.
- Added reaction to IR commands instead of timed loops.
- Removed limitations of fixed sequences.

New Features

- Integrated IR receiver (pin 11) into the system.
- Each button on the remote toggles a specific LED.
- Basic wireless control successfully validated.

5.3.4Still Not Working

- No Bluetooth communication.
- Only three LEDs are supported.

- No unified command protocol.
- No state tracking system for larger LED arrays.

```ino
#include <IRremote.hpp>

#define IR_PIN 11
#define CMD_1 0x0C
#define CMD_2 0x18
#define CMD_3 0x5E

int leds[3] = {2, 3, 4};

void setup() {
  IrReceiver.begin(IR_PIN);
  for(int i = 0; i < 3; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  if (IrReceiver.decode()) {
    uint8_t cmd = IrReceiver.decodedIRData.command;

    if (cmd == CMD_1) digitalWrite(leds[0],
      !digitalRead(leds[0]));
    if (cmd == CMD_2) digitalWrite(leds[1],
      !digitalRead(leds[1]));
    if (cmd == CMD_3) digitalWrite(leds[2],
      !digitalRead(leds[2]));

    IrReceiver.resume();
  }
}
```

Figure 6: Controls LEDs using an IR remote

5.4 Prototype Version 0003 - Final System (IR + Bluetooth + 9 LEDs)

Purpose

The final prototype combines all system components including the following: nine LEDs, IR remote control, Bluetooth command processing, and internal LED state tracking. This version represents the complete and fully functional system.

Fixes from Version 3

- Added Bluetooth HC-05 communication.
- Expanded LED support from 3 to 9.
- Introduced a full command parser.
- Implemented LED state memory.
- Added multi-LED toggle functions for IR buttons.
- Replaced simple toggling with a structured control system.

New Features

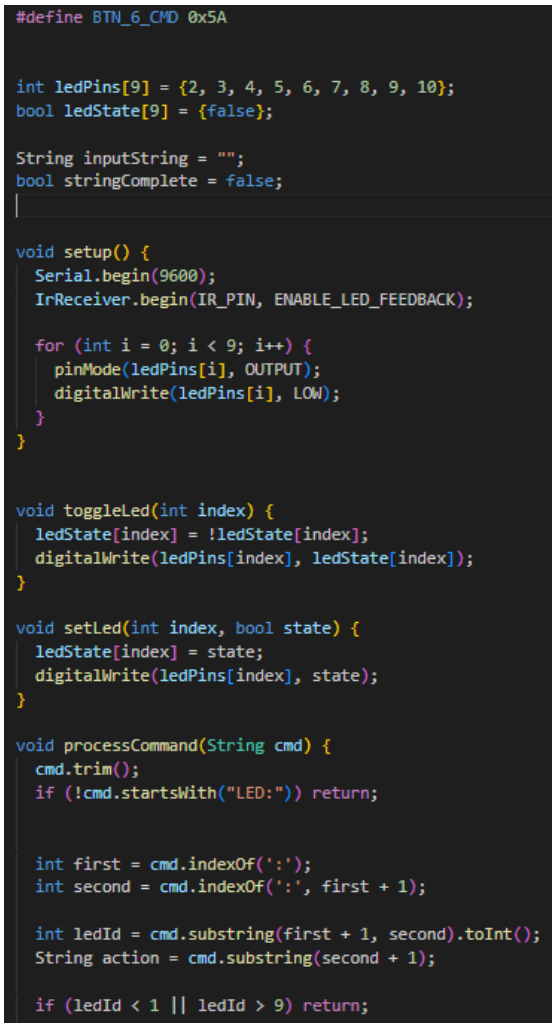
- Dual wireless interfaces (IR + Bluetooth).
- Nine independent LEDs with individual states.
  Acknowledgement messages (ACK:id:action).
- Full integration of IR signals with non-repeat filtering.
- Clean and scalable LED array architecture.

Working Correctly

- All nine LEDs toggle according to IR remote input.
- Bluetooth accepts commands in structured format (LED:<id>:<action>).
- State tracking prevents desynchronization.
- System is stable under repeated commands.

Figure 7: Controls nine LEDs via IR and Bluetooth with state tracking.

```
  int index = ledId - 1;

  if (action == "TOGGLE") {
    toggleLed(index);
  } else if (action == "ON") {
    setLed(index, true);
  } else if (action == "OFF") {
    setLed(index, false);
  }

  Serial.print("ACK:");
  Serial.print(ledId);
  Serial.print(":");
  Serial.println(action);
}


void loop() {


  if (IrReceiver.decode()) {

    if (!(IrReceiver.decodedIRData.flags & IRDATA_FLAGS_IS_REPEAT)) {
      uint8_t cmd = IrReceiver.decodedIRData.command;

      switch (cmd) {
        case BTN_1_CMD: toggleLed(0); toggleLed(1); break;
        case BTN_2_CMD: toggleLed(2); toggleLed(3); break;
        case BTN_3_CMD: toggleLed(4); toggleLed(5); break;
        case BTN_4_CMD: toggleLed(6); break;
        case BTN_5_CMD: toggleLed(7); break;
        case BTN_6_CMD: toggleLed(8); break;
      }
    }

    IrReceiver.resume();
  }


  while (Serial.available()) {
    char c = (char)Serial.read();
    if (c == '\n') {
      stringComplete = true;
      break;
    }
    inputString += c;
  }

  if (stringComplete) {
    processCommand(inputString);
    inputString = "";
    stringComplete = false;
  }
}
```



```
#define BTN_6_CMD 0x5A


int ledPins[9] = {2, 3, 4, 5, 6, 7, 8, 9, 10};
bool ledState[9] = {false};

String inputString = "";
bool stringComplete = false;


void setup() {
  Serial.begin(9600);
  IrReceiver.begin(IR_PIN, ENABLE_LED_FEEDBACK);

  for (int i = 0; i < 9; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW);
  }
}


void toggleLed(int index) {
  ledState[index] = !ledState[index];
  digitalWrite(ledPins[index], ledState[index]);
}

void setLed(int index, bool state) {
  ledState[index] = state;
  digitalWrite(ledPins[index], state);
}

void processCommand(String cmd) {
  cmd.trim();
  if (!cmd.startsWith("LED:")) return;


  int first = cmd.indexOf(':');
  int second = cmd.indexOf(':', first + 1);

  int ledId = cmd.substring(first + 1, second).toInt();
  String action = cmd.substring(second + 1);

  if (ledId < 1 || ledId > 9) return;
```

Figure 8: Handles LED control by processing IR remote signals

6. Implementation

Language & Tools:

- Arduino C/C++
- PlatformIO IDE
- Hardware: Arduino Uno / Nano, 9 LEDs, IR receiver, HC-05 Bluetooth module

System Architecture

The final system integrates multiple components:

1. **LED Array Control** – 9 LEDs with individual state tracking.
2. **IR Remote Input** – Uses NEC protocol to toggle LEDs.
3. **Bluetooth Communication** – Accepts structured commands to control LEDs.
4. **Command Parser** – Processes IR and Bluetooth commands and maintains LED states.

6.1 Key Functions

Figure 9: LED Control

```
25  void toggleLed(int index) {
26    ledState[index] = !ledState[index];
27    digitalWrite(ledPins[index], ledState[index]);
28  }
29
30  void setLed(int index, bool state) {
31    ledState[index] = state;
32    digitalWrite(ledPins[index], state);
33  }
```

```
void processCommand(String cmd) {
  cmd.trim();
  if (!cmd.startsWith("LED:")) return;

  int first = cmd.indexOf(':');
  int second = cmd.indexOf(':', first + 1);

  int ledId = cmd.substring(first + 1, second).toInt();
  String action = cmd.substring(second + 1);

  if (ledId < 1 || ledId > 9) return;

  int index = ledId - 1;

  if (action == "TOGGLE") toggleLed(index);
  else if (action == "ON") setLed(index, true);
  else if (action == "OFF") setLed(index, false);

  Serial.print("ACK:");
  Serial.print(ledId);
  Serial.print(":");
  Serial.println(action);
}
```

Figure 10: Bluetooth Command Handling

6.2 Notes

- LED state array prevents desynchronization between commands.
- IR input is filtered to ignore repeated signals.
- Bluetooth commands follow the structured format "LED:<id>:<action>".

6.3 Full Code:

The complete working code is included in GitHub:
https://github.com/tw1zzyy/info-team-arduino.git

7.Testing & Resulting

7.1Test Procedure

LED Control Testing

- Confirmed each of the 9 LEDs via Bluetooth and IR.
- Assessed the sequential and simultaneous activation of multiple LEDs.
- Verified LED states are maintained by the ledState array during repeated commands.

IR Remote Testing

- Correct recognition of NEC IR remote commands was verified.
- Verified repeated signals are filtered (holding a button does not trigger multiple toggles).
- Tested button combinations, e.g., BTN_1_CMD toggles two LEDs at once.

Bluetooth Testing

- Sent commands in "LED:<id>:<action>" format via the HC-05 module.
- All actions tested: ON, OFF, TOGGLE.
- Checked all acknowledgement messages as correct: "ACK:<id>:<action>".

Edge Cases

- For system stability, checked invalid commands (for example, LED:10:ON).
- Tested rapid sequential commands to verify system response.

Results / Outcomes

- All 9 LEDs are properly controlled with both IR and Bluetooth.
- LED states are synchronized and stored in the ledState array to avoid desynchronization.
- IR and Bluetooth commands are executed independently and simultaneously without conflicts.
- The system is resistant to repeated or invalid commands.

- IR repeat filtering works to block unintended LED toggles.

Conclusion

- The prototype addresses all the target requirements:
    - 9 LED control,
    - dual wireless interfaces (IR and Bluetooth),
    - command handling with acknowledgements,
    - and robust error/repeat handling.

8.Discussion (problems, limitations)

What worked really well
The IR remote control worked reliably and allowed us to control the system easily from a distance. Once configured, the LED control responded correctly to user input.

What problems you faced (noise, power, delays)
We faced several issues during development, including weak LEDs and connection problems with the HC-05 and HC-06 Bluetooth modules. There were also minor delays caused by unstable connections and power limitations.

How the prototype evolved from initial version to final version
At the beginning, we only tested turning LEDs on and off using the HC-05 module via a mobile phone. Later, we added an IR receiver to control the system using a remote control. After that, we extended the jumper cables so that the LEDs could be placed and distributed around different parts of the house.

What trade-offs you made
We removed the timer because we could not find a suitable battery for it. Additionally, we eliminated several extra features to make the project more stable, reduce errors, and ensure reliable operation. We also decided to extend the wires, even though this increased complexity, in order to improve flexibility in LED placement.

9.Conclusion & future work

9.1 Problem we solved:

Remote home lighting control was a problem that got solved, with the use of wireless and IR technologies, rather than manual switches. Useful smart home automation based on Bluetooth and IR receivers has also been shown in academic studies to control appliances remotely using an Arduino and wireless modules.

9.2 What your final system can do:

The final system connects the user with an IR remote control to turn LEDs on and off, while placing LEDs in various locations using extended wires. This simple home lighting control system fits into academic papers on smart home lighting systems with Bluetooth and mobile control.

9.3 Future work ideas:

There are a few ways the system could be improved in the future:

Add more features. Like brightness control, automation modes, scheduling — like enhanced smart lighting and IoT automation systems covered in the research.
Better enclosure. A suitable enclosure would shield the electronics and give it better durability and aesthetics.
Cloud connection and sensors. Connecting cloud services and other sensors (e.g., motion, light, environmental sensors) would facilitate more intelligent automated control and remote monitoring, consistent with the current trends on the smart home.

10.References (harvard style)

Works Cited

Bolimera, R., et al. “Wireless home automation using Arduino and Bluetooth.” *International Journal of Data Science and IoT Management System*, vol. 4(4), pp. 343–346, https://doi.org/10.64751/ijdim.2025.v4.n4.pp343-346.

Iyen, C., et al. “Implementation of Bluetooth Enabled Home Automation System.” *European Journal of Theoretical and Applied Sciences*, vol. 2(2), pp. 310–318, https://doi.org/10.59324/ejtas.2024.2(2).27.

Khan, F.M., et al. “Intelligent Bluetooth‑Driven Automation: Multi‑Sensor Integration for Real‑Time Smart Home Control.” *Spectrum of Engineering Sciences*, vol. 3(7), pp. 173–180, https://sesjournal.org/index.php/1/article/view/580.

Simwaba, M.J.C., and Shabiyemba, M. “IoT Based Smart Home Automation System: Design and Development.” *Scientific Journal of Engineering and Technology*, vol. 2(2), pp. 45–52, https://doi.org/10.69739/sjet.v2i2.438.

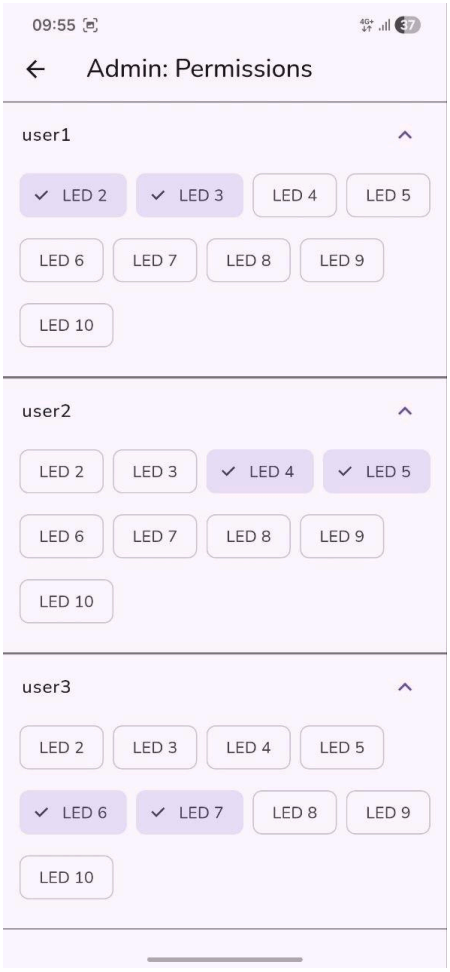
Thombare, N., et al. “IoT Based Smart Home Automation System using Bluetooth.” *International Journal of Scientific Research in Science and Technology*, https://ijsrst.com/index.php/home/article/view/IJSRST2513116.

Torres‑Hernandez, C.M., et al. “Smart Homes: A Meta‑Study on Sense of Security and Home Automation.” *Technologies*, vol. 13(8):320, https://doi.org/10.3390/technologies13080320.

Torres‑Hernandez, C.M., et al. “Smart Homes: A Meta‑Study on Sense of Security and Home Automation.” *Technologies*, vol. 13(8):320, https://doi.org/10.3390/technologies13080320.

11.Appendices (code, circuit diagrams, extra data):

Figure 11: Light - change application admin page



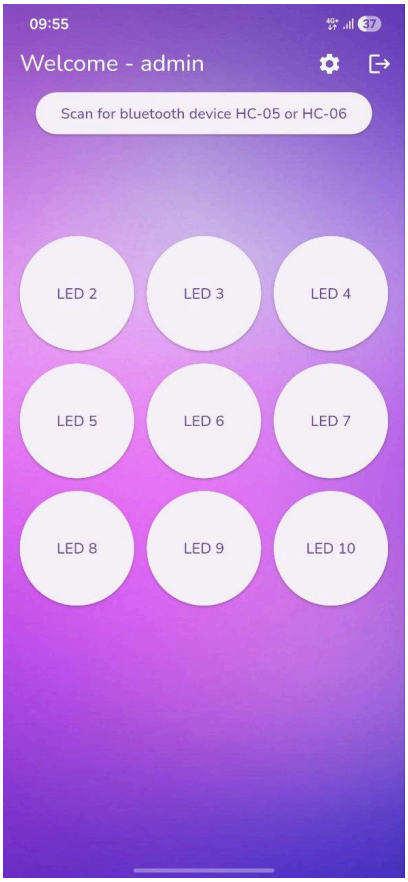


Figure 12: Light - change application choose led to control

Figure 13: Ready house with leds

Github **https://github.com/tw1zzyy/info-team-arduino.git**

Youtube https://youtube.com/shorts/wDUuHgMM18k?si=ie8mGVlyuT-oafRc(Video of working)

Youtube https://youtu.be/8mywCUjXW_o?si=iNIxO-JFlAc5QhHg (Overview)