NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

Embedded Systems
(EC16101)
Project Report
Academic Year 2023-2024

Submitted by:
Group III
Saurav Kumar (B210062EC)
Tejas Khillare (B210095EC)
Kaushik Gupta (B210050EC)
Harsh Srivastava (B210048EC)
Semester VI

Submitted to:
Dr. Neha K. Nawandar
(Course instructor)
Department of ECE, NIT Sikkim

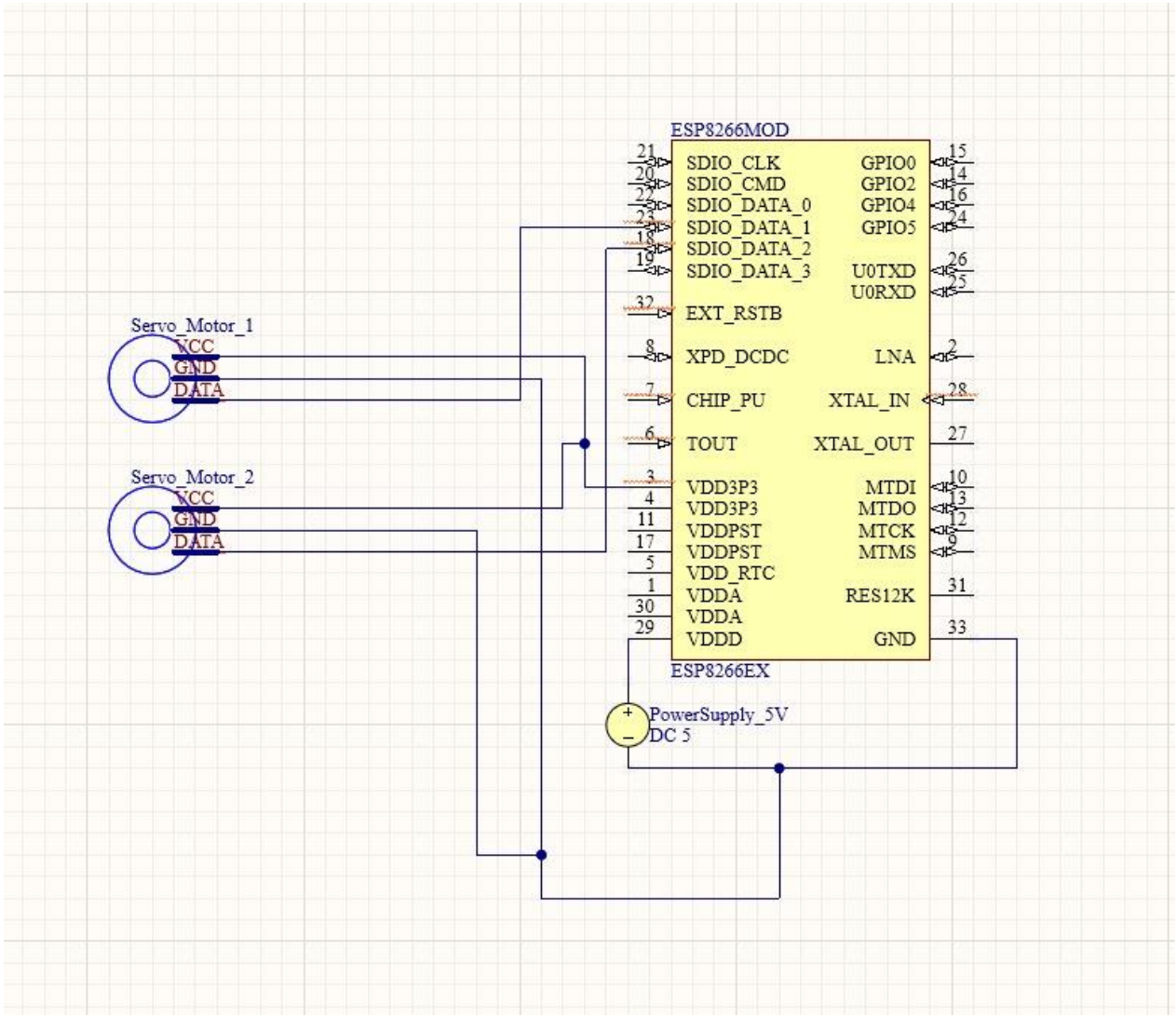
Problem Statement

Design and implement a home automation system using Node-MCU and servo motors integrated into a 3D printed mechanism. The system aims to provide wireless control via an Android application, facilitating the remote operation of switches within a household.

Hardware and Software Components

1.) ESP8266 Wi-Fi Module
2.) 2 Servo Motor (SG-90)
3.) 3D-Printed Case
4.) C++ / Arduino IDE
5.) Flutter
6.) Solid Works

Circuit Diagram / Design Schematic



Theory

The rapid advancement of technology has led to the emergence of smart homes, where various appliances and devices can be controlled remotely for increased convenience and efficiency. In line with this trend, the aim of this project is to design and implement a home automation system using Node-MCU and servo motors. The system enables wireless control via an Android application, allowing users to remotely operate switches within their household.

ESP8266

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. The ESP8266 module is an extremely cost-effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime.

Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, and is designed to occupy minimal PCB area. It contains a self-calibrated RF allowing it to work under all operating conditions and requires no external RF parts. There is an almost limitless fountain of information available for the ESP8266, all of which has been provided by amazing community support.

Use in our project

The ESP8266 has been used as a brain for our project which serves as a main control unit and gateway of wireless communication. This connectivity is essential for enabling remote control and monitoring of the home automation system from anywhere within the network's coverage area. The ESP8266 ensures robust and efficient communication between our hardware system and the Android software. It handles the transmission and reception of data packets, allowing for seamless interaction between the two endpoints. It receives commands from the Android app, interprets them, and translates them into actions that the servo motors can execute, thereby enabling remote control of switches.

Servo motor (SG-90)

Servo motors are electromechanical devices utilized for precise control of angular position. In the context of the home automation project, servo motors are employed to physically toggle switches on the control board, thereby enabling the remote operation of household appliances and devices.

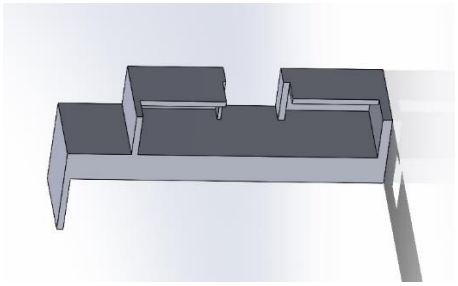
At the core of the servo motor is a DC motor, which converts electrical energy into mechanical rotation. The motor's speed and torque characteristics are crucial for its performance in various applications. A gearbox or gear train is utilized to reduce the motor's rotational speed while increasing its torque output. This gearing mechanism enables servo motors to achieve high torque at low speeds, essential for precise positioning tasks.

Servo motors incorporate a position feedback device, typically a potentiometer or an optical encoder, to provide real-time feedback on the motor's angular position. This feedback loop enables accurate control of the motor's output shaft position. This closed-loop control system ensures precise and stable operation under varying load conditions.

Use in our project.

Servo motors receive control signals from the ESP8266 microcontroller, which dictates the desired angular position of the motor's output shaft. Based on the received control signals, the servo motor's control circuitry adjusts the motor's position by driving the motor to the specified angle. By rotating the servo motor's shaft to specific angles, switches can be toggled on or off, facilitating the automation of lighting fixtures, fans, and other electrical appliances within the household.
By attaching levers or arms to the servo motors' shafts, they can physically interact with the switches on the control board. When activated, the servo motors rotate to specific angles, causing the switches to move accordingly and initiate the desired action (turning appliances on or off). The use of servo motors for switch control enhances the overall user experience of the home automation system. It provides a tangible and interactive interface for users, allowing them to remotely control their appliances with precision and ease.

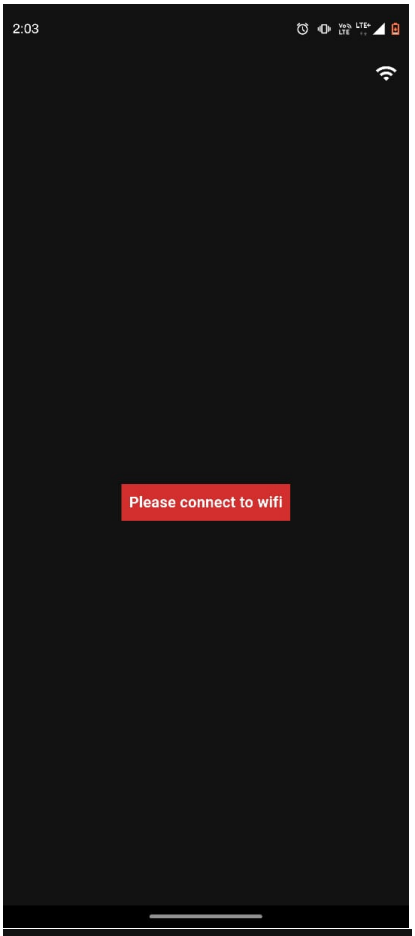
3D-Printed Case

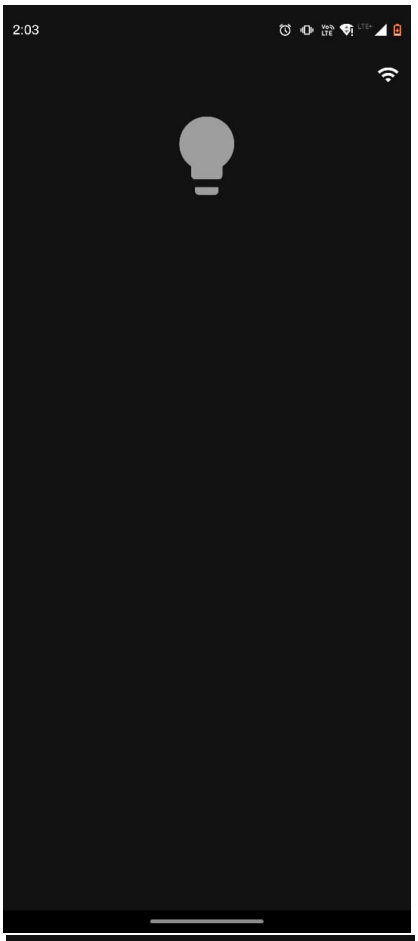


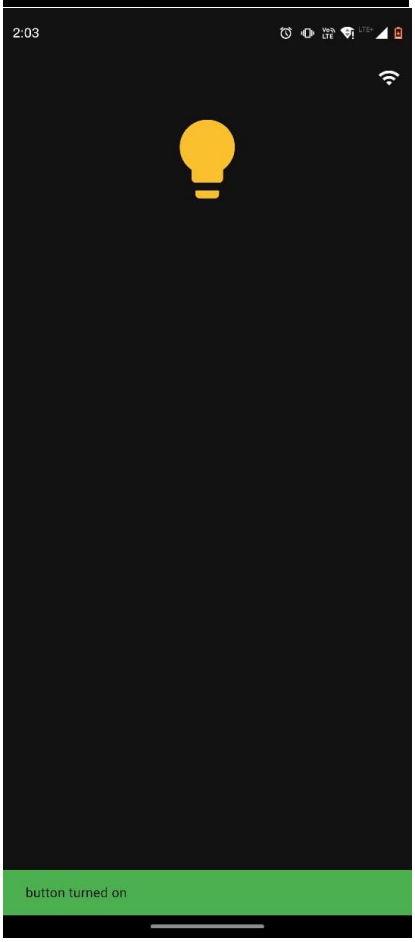
The 3D printed case for the hardware components of the home automation system was conceptualized to provide housing that is both functional and aesthetically pleasing. This is just a prototype model, but we can make it more optimized and good looking. The design process began with a thorough assessment of the dimensions and requirements of the hardware components, ensuring compatibility and optimal fit within the case.

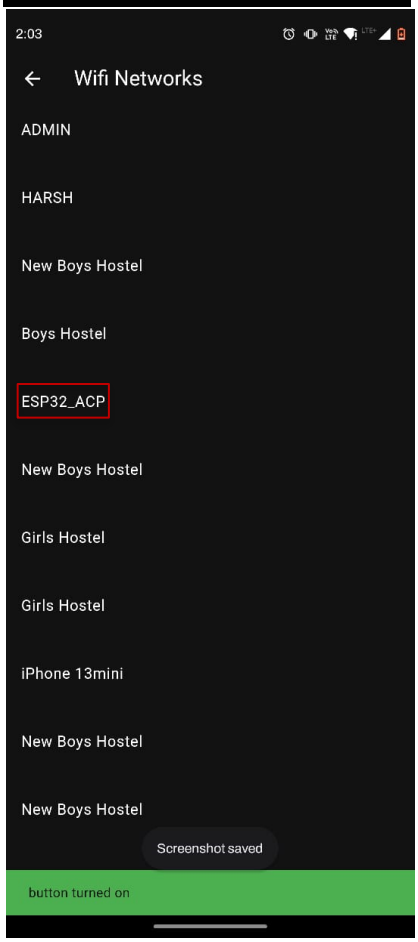
The design of the 3D printed case was meticulously crafted using SolidWorks, a powerful computer-aided design (CAD) software. SolidWorks provided the tools and capabilities necessary to translate the conceptual design into a detailed and precise digital model. The choice of material for 3D printing was carefully considered to ensure durability, strength, and suitability for the intended application. We have used PLA material to build the model. Upon completion of the printing process, the 3D printed case was carefully fitted with the hardware components of the home automation system.

Android Application









The Android application for the user side has is developed using Flutter, an open-source UI software development kit created by Google. Flutter has been chosen for its cross-platform compatibility, allowing for the development of both Android and iOS applications from a single codebase. Flutter's rich set of customizable widgets and Material Design principles were leveraged to create a visually appealing and user-friendly interface.

The Flutter application communicates with the ESP8266 microcontroller via Wi-Fi to control the home automation system. Through a series of API endpoints, the application sends commands to the ESP8266, instructing it to toggle switches and activate or deactivate household appliances. Real-time feedback from the ESP8266 is also displayed in the application, allowing users to monitor the status of their devices remotely. Users can toggle switches on/off to control household appliances remotely.

Codes

ESP8266 Code

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ArduinoJson.h>
#include <string>
#include <Servo.h>
#include <ArduinoJson.h>

#ifndef APSSID
#define APSSID "ESP32_ACP"
#define APPSK "ece@123ece"
#endif

/* Declare the pins for the leds and their status var */
Servo servoOn, servoOff;
int servoOnPin = D1, servoOffPin = D2;
bool buttonState = 0;

/* Set these to your desired credentials. */
const char *ssid = APSSID;
const char *password = APPSK;

ESP8266WebServer server(80);

/* Just a little test message.  Go to http://192.168.4.1 in a web browser
   connected to this access point to see it.*/
void handleRoot() {
  server.send(200, "text/html", "<h1>You are connected</h1>");
}
```

```
void turnOn() {
  // set servo 1 to position 90
  if (buttonState == 0) {
    servoOn.write(0);
    servoOff.write(0);
    buttonState = 1;
  } else {
    servoOn.write(90);
    servoOff.write(90);
    buttonState = 0;
  }
  // Build the JSON response
  StaticJsonDocument<200> jsonDoc;  // Adjust the size according to your needs
  jsonDoc["status"] = "success";
  jsonDoc["message"] = "Button state toggled";
  jsonDoc["buttonStatus"] = buttonState;
  // Convert the JSON document to a string
  String jsonResponse;
  serializeJson(jsonDoc, jsonResponse);
  // Send the JSON response
  server.send(200, "application/json", jsonResponse);
}

/*Setup function to initialize the initial state */
void setup() {
  servoOn.attach(servoOnPin, 544, 2400);
  servoOff.attach(servoOffPin, 544, 2400);
  delay(1000);
  Serial.begin(115200);
  Serial.println();
  Serial.print("Configuring access point...");
  /* You can remove the password parameter if you want the AP to be open. */
  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  server.on("/", handleRoot);
  server.on("/button", turnOn);
  // server.on("/off", turnOff);

  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}
```

Application Code

main.dart

```dart
import 'package:automation_app/homescreen.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Home Automation',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.dark(),
        useMaterial3: true,
      ),
      home: const HomePage(),
    );
  }
}
```

wifiscreen.dart

```dart
// ignore_for_file: prefer_const_constructors
import 'dart:async';
import 'dart:developer';

import 'package:flutter/material.dart';
import 'package:location/location.dart';
import 'package:wifi_iot/wifi_iot.dart';
import 'package:wifi_scan/wifi_scan.dart';

///  Main WifiScreen Class
class WifiScreen extends StatefulWidget {
  const WifiScreen({super.key});

  @override
  State<StatefulWidget> createState() => WifiScreenState();
}

/// WifiScreen State Class
class WifiScreenState extends State<WifiScreen> {
  String _screenState = 'loading';

// initialize accessPoints and subscription
```

```dart
  List<WiFiAccessPoint> accessPoints = [];
  StreamSubscription<List<WiFiAccessPoint>>? subscription;

  void _startListeningToScannedResults() async {
    // check platform support and necessary requirements
    final can =
        await WiFiScan.instance.canGetScannedResults(askPermissions: true);
    switch (can) {
      case CanGetScannedResults.yes:
        // listen to onScannedResultsAvailable stream
        subscription =
            WiFiScan.instance.onScannedResultsAvailable.listen((results) {
          // update accessPoints
          log('Can get the result');
          setState(() {
            _screenState = 'Enabled';
            accessPoints = results;
          });
        });
        // ...
        break;

      case (CanGetScannedResults.noLocationServiceDisabled):
        await Location().requestService();
        setState(() {
          _screenState = 'loading';
        });

      default:
        log('Error while getting the result');
    }
  }

// make sure to cancel subscription after you are done
  @override
  dispose() {
    super.dispose();
    subscription?.cancel();
  }

  @override
  void initState() {
    _startListeningToScannedResults();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    checkWifiState();
    return Scaffold(
        appBar: AppBar(
          title: const Text('Wifi Networks'),
        ),
```

```
      body: _screenState == 'loading'
          ? Center(
              child: CircularProgressIndicator(),
            )
          : _screenState == 'DISABLED'
              ? Center(
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                      Icon(
                        Icons.wifi_off,
                        size: 100,
                        color: Colors.red,
                      ),
                      Text(
                        'WiFi is disabled',
                        style: TextStyle(
                          color: Colors.red,
                        ),
                      ),
                      OutlinedButton(
                        onPressed: () {
                          setState(() {});
                          WiFiForIoTPlugin.setEnabled(true,
                              shouldOpenSettings: true);
                        },
                        child: Text(
                          'Enable WiFi',
                          style: TextStyle(
                              color: Colors.white,
                              fontWeight: FontWeight.bold,
                              fontSize: 16),
                        ),
                      ),
                    ],
                  ),
                )
              : StreamBuilder(
                  stream: WiFiScan.instance.onScannedResultsAvailable,
                  builder: (context, snapshot) {
                    return ListView.builder(
                      itemCount: snapshot.data?.length ?? 0,
                      itemBuilder: (context, index) {
                        return InkWell(
                            child: ListTile(
                                title: Text(
                                  snapshot.data![index].ssid,
                                  // style: TextStyle(color: Colors.white),
                                ),
                                subtitle: Column(
                                  children: [
                                    // Text('Bssid : ${snapshot.data![index].level}')
                                  ],
```

```
                                  )),
                              onTap: () async {
                                WiFiForIoTPlugin.disconnect();

                                await WiFiForIoTPlugin.connect(
                                  snapshot.data![index].ssid,
                                  withInternet: false,
                                  // security: NetworkSecurity.WPA,
                                  password: 'ece@123ece',
                                );
                                WiFiForIoTPlugin.forceWifiUsage(true);
                              });
                        },
                      );
                    }));
  }

  void checkWifiState() async {
    if (await WiFiForIoTPlugin.isEnabled()) {
      setState(() {
        _screenState = 'ENABLED';
      });
    } else {
      setState(() {
        _screenState = 'DISABLED';
      });
    }
  }
}
```

homescreen.dart

```
// ignore_for_file: use_build_context_synchronously

import 'dart:convert';
import 'dart:developer';

import 'package:automation_app/wifiscreen.dart';
import 'package:connectivity_plus/connectivity_plus.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class HomePage extends StatefulWidget {
  const HomePage({Key? key});

  @override
  State<StatefulWidget> createState() => HomePageState();
}

class HomePageState extends State<HomePage> {
  late final Connectivity connectionObj;
```

```dart
late final Stream<ConnectivityResult> subscription;
// List<bool> buttonState = [false, false, false, false];
bool buttonState = false;
final numController = TextEditingController();

// get http => null;

@override
void initState() {
  connectionObj = Connectivity();
  subscription = Connectivity().onConnectivityChanged;
  super.initState();
}

void showSnackMessage(BuildContext context, String message, Color color) {
  ScaffoldMessenger.of(context).clearSnackBars();
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Container(
        child: Text(message),
      ),
      backgroundColor: color,
    ),
  );
}
Future<void> toggleButton() async {
  try {
    final response = await http.get(Uri.parse("http://192.168.4.1/button"));

    if (response.statusCode == 200) {
      final jsonResponse = json.decode(response.body);
      print(jsonResponse);

      showSnackMessage(
        context,
        "button turned ${jsonResponse['buttonStatus'] ? "on" : "off"}",
        jsonResponse['buttonStatus'] ? Colors.green : Colors.grey,
      );
      log(jsonResponse['buttonStatus'].toString());
      setState(() {
        buttonState = jsonResponse["buttonStatus"];
      });
    } else {
      showSnackMessage(
        context,
        "Unexpected status code: ${response.statusCode}",
        Colors.grey,
      );
    }
  } catch (e) {
    showSnackMessage(context, "Error toggling button: $e", Colors.red);
  }
}
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      actions: [
        IconButton(
          icon: Icon(Icons.wifi),
          onPressed: () {
            Navigator.of(context).push(MaterialPageRoute(builder: (context) {
              return WifiScreen();
            }));
          },
        )
      ],
    ),
    body: SafeArea(
      child: Center(
        child: StreamBuilder<ConnectivityResult>(
          stream: subscription,
          builder: (context, snapshot) {
            if (snapshot.data != ConnectivityResult.wifi) {
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                mainAxisSize: MainAxisSize.max,
                children: [
                  Container(
                    decoration: BoxDecoration(color: Colors.red[700]),
                    padding: const EdgeInsets.all(8),
                    child: const Text(
                      "Please connect to wifi",
                      style: TextStyle(
                        fontWeight: FontWeight.w600,
                        fontSize: 16,
                      ),
                    ),
                  ),
                ],
              );
            } else {
              return Column(
                children: [
                  IconButton(
                    onPressed: () {
                      toggleButton();
                    },
                    icon: Icon(
                      Icons.lightbulb,
                      size: 100,
                      color: buttonState ? Colors.yellow[700] : Colors.grey,
                    ),
                  ),
                ],
```

```
                );
              }
            },
          ),
        ),
      ),
    );
  }
}

class FilledButton extends StatelessWidget {
  final VoidCallback onPressed;
  final Widget child;

  const FilledButton({
    Key? key,
    required this.onPressed,
    required this.child,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      child: child,
    );
  }
}
```

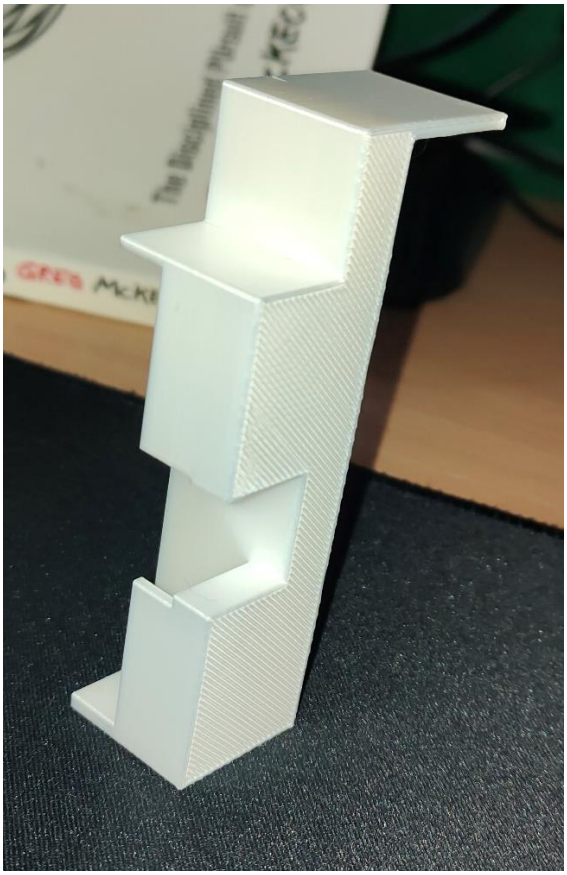
Discussion & Conclusion

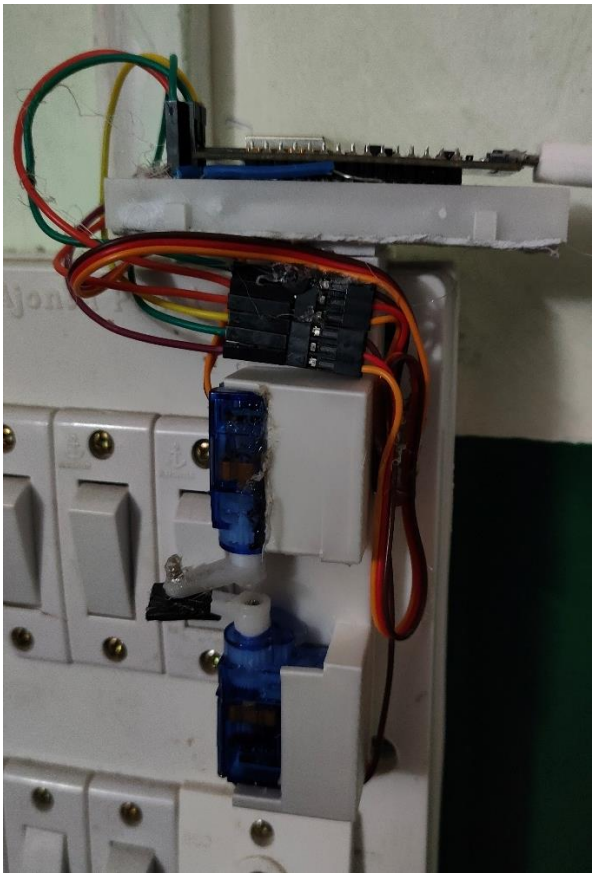
The implementation of the home automation system using Node-MCU, servo motors, Flutter for Android application, and 3D printed casing represents a significant advancement in modern home automation technology. This project successfully addresses the need for convenient and efficient control of household appliances through wireless communication and intuitive user interfaces. The integration of Node-MCU provides a robust foundation for wireless connectivity, allowing users to remotely control switches within their household via the Android application.

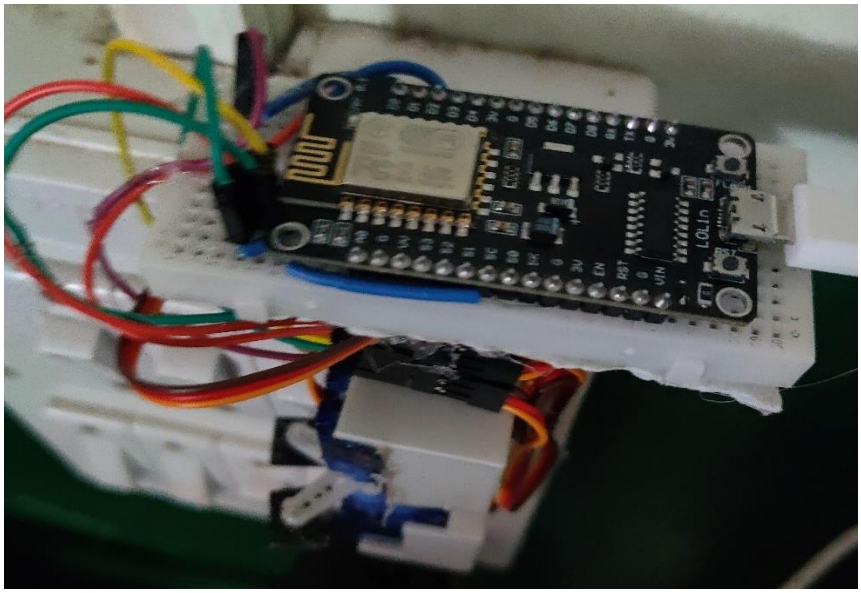
The home automation system developed in this project serves as a foundation for further scalability and expansion. Future developments could include additional functionalities such as voice control, scheduling, and integration with other smart home devices. Furthermore, advancements in hardware and software technologies may lead to enhanced performance, reliability, and energy efficiency in future iterations of the system. This project exemplifies the synergy between hardware and software components to create a sophisticated yet user-friendly home automation solution.

As technology continues to evolve and new possibilities emerge, the potential for further innovation and enhancement in home automation remains limitless. By leveraging the advancements in hardware and software technologies, future iterations of the system have the potential to redefine the standards of convenience, comfort, and sustainability in smart home environments.

Snapshots







Working Video Video Link_1 Video_Link_2