GAUTAM BUDDHA UNIVERSITY

MINOR PROJECT REPORT ON

HOME AUTOMATION

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNDER THE GUIDANCE OF

Mr. Neeraj Kaushik

MCA(AI) SEM III [2023-2024]

Submitted By

Group Member	Roll Number
Rajkumar	225/PCA/011
Shabnam Kumari	225/PCA/027
Soni Saraswat	225/PCA/030
Akanksha Tyagi	225/PCA/026

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY GAUTAM BUDDHA UNIVERSITY, GREATER NOIDA, 201312, U.P., (INDIA)

Candidate's Declaration

We, hereby, certify that the work embodied in this minor project report entitled "**Home Automation**" in partial fulfillment of the requirements for the award of the Degree of MCA submitted to the School of Information and Communication Technology, Gautam Buddha University, Greater Noida is an authentic record of ourown work carried out under the supervision of Mr. Neeraj Kaushik , School of ICT. The matter presented in this report has not been submitted in any other University / Institute for the award of any other degree or diploma. Responsibility for any plagiarism related issue stands solely with us:

Group Member	Roll Number
Rajkumar	225/PCA/011
Shabnam Sharma	225/PCA/027
Soni Saraswat	225/PCA/030
Akanksha Tyagi	225/PCA/026

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief. However, responsibility for any plagiarism related issue solely stands with the students.

Signature of the Supervisor:

Name: Mr. Neeraj Kaushik

Date /Place: 23/Nov/2023, Greater Noida

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organization. We would like to extend my sincere thanks to all of them.

We are highly indebted to **Mr. Neeraj Kaushik** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to thank Dean of ICT Prof. **Sanjay kumar sharma** and HOD of ICT, **Dr. Neeta Singh** whose support helped us to complete our work. We would like to express our gratitude towards our friends & family members for their kind cooperation and encouragement which helped us in completion of this major project.

Table of Contents

Abstract

The main objective of this project is to develop a home automation system using an ESP 32 dev board with Wifi and Bluetooth being remotely controlled by any Android OS smart phone. As technology is advancing so houses are also getting smarter. Modern houses are gradually shifting from conventional switches to centralized control system, involving remote controlled switches. Presently, conventional wall switches located in different parts of the house makes it difficult for the user to go near them to operate. Even more it becomes more difficult for the elderly or physically handicapped people to do so. Remote controlled home automation system provides a most modern solution with smart phones. In order to achieve this, a esp32 module is interfaced at the receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF commands to the receiver where loads are connected. By touching the specified location on the GUI, the loads can be turned ON/OFF remotely through this technology.

List of figures

1. Introduction

Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives real easy. Have you ever wondered about home automation which would give the facility of controlling tube lights, fans and other electrical appliances at home using a remote control? Off-course, Yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it. We have come up with a new system called Arduino based home automation using Bluetooth. This system is super-cost effective and can give the user, the ability to control any electronic device without even spending for a remote control. This project helps the user to control all the electronic devices using his/her smart phone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save our time. To save people's time we are introducing Home Automation system using Bluetooth . With the help of this system you can control your home appliances from your mobile phone. You can turn on/off your home appliances within the range of Bluetooth.

2. Components required

1. NodeMCU ESP32
2. ESP 32 Cam
3. FTDI
4. 4 channel relay(5v)
5. LEDs
6. Power supply
7. Micro USB cable
8. Connecting wires
9. Bread board
10. Smart phone
11. Sinric pro Application
12. Arduino IDE
13. Programming language C/C++

3. DESCRIPTION

1.1 NodeMCU ESP32

Esp32 is a low-cost development board with WIFI & Bluetooth chips embedded in it. The main feature of ESP32 that makes it better than other development boards is that it is dual-core 32-bit with an ESP-WROOM-32 LX6 microprocessor. ESP32 plays an important role in developing IoT-based applications as it consists of two wireless technology such as WIFI and Bluetooth. ESP32 has an operating voltage of 2.2v to 6v with the onboard regulator to provide stable voltage and an available output current of more than 500 mA. ESP32 has a dual-core processor which runs independently of each other. 1 core at 240 MHz, 2 cores at 240 MHz adjustable clock frequency. Esp32 has a flash memory of 4 MB which makes ESP32 faster. it has a built-in hall effect sensor.
ESP32 is a low-cost, low-power system on a chip (SoC) series with Wi-Fi and dual-mode Bluetooth capabilities. It's built around a Tensilica Xtensa dual-core processor, running at 160MHz (overclockable to 240MHz). With 520KB of SRAM, 34 GPIO pins, and integrated Wi-Fi and Bluetooth, the ESP32 is designed for simple, cost-effective, and power-efficient IoT applications. While it doesn't have built-in USB ports, it can be programmed via USB-UART, and while it doesn't include onboard storage, it does support microSD and SPI flash. It is developed by Espressif Systems. The ESP32 can be programmed using the Arduino IDE, ESP-IDF (IoT Development Framework), MicroPython, JavaScript, Lua, and more. ESP32 operates at 2.7V - 3.6V and has a wide operating temperature range from -40°C to 125°C.

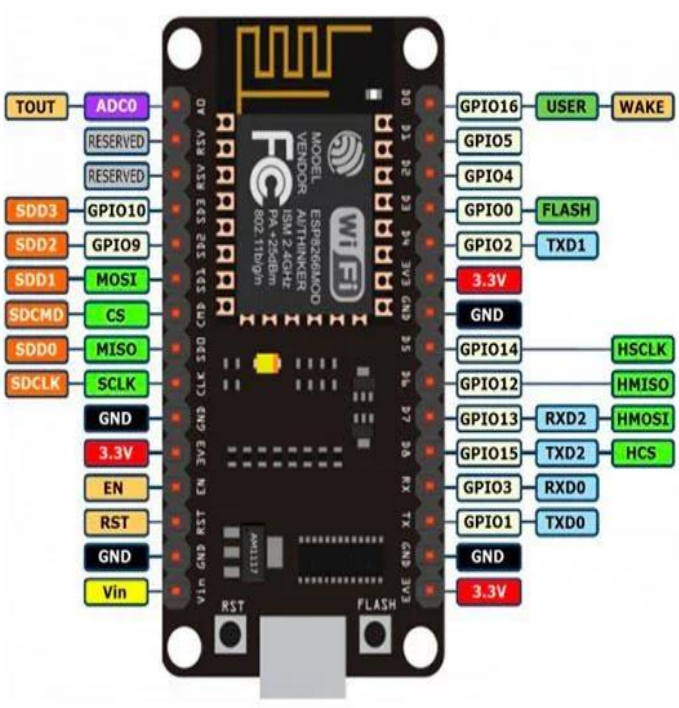

Fig 1: NodeMCU ESP32

The ESP32 includes a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 is already integrated with antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, power management module, and advanced adaptive biasing for low-power operation. It supports a variety of Bluetooth features such as classic Bluetooth, Bluetooth low energy, and Bluetooth 4.2 specifications. ESP32 includes both RAM and flash memory, the size of which depends on the specific variant. Typically, an ESP32 variant has 512KB of SRAM and 4MB of flash memory. The ESP32 chip has 34 GPIO pins which can support functions such as ADC (Analog to Digital Conversion), DAC (Digital to Analog Conversion), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver/Transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), and more.
The ESP32 is widely used in IoT devices, wearable electronics, smart home applications, and many other scenarios due to its features, low cost, and ease of use. It provides a high level of integration, and its extensive protocol support makes it a very versatile chip for building connected devices.
Both Nodemcu ESP8266 and ESP 32 belong to the ESP family NodeMCU ESP8266 and ESP32 are both popular choices for IoT projects, but they serve different needs and have different capabilities.

Types of esp32 development board:-

- ESP32 DEV KIT DOIT
- ESP32-DevKitC
- ESP32-PICO-KIT
- ESP-EYE
- ESP32 Thing
- SX1278 ESP32 (LORA)
- ESP32-CAM
- ESP 32s

Specifications of ESP32

ESP32 has a lot more features than ESP8266 and it is difficult to include all the specifications in this Getting Started with ESP32 guide. So, I made a list of some of the important specifications of ESP32 here. But for complete set of specifications, I strongly suggest you to refer to the Datasheet.

- Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz.
- 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.
- Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.
- Support for both Classic Bluetooth v4.2 and BLE specifications.
- 34 Programmable GPIOs.
- Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity include 4 x SPI, 2 x I^2C, 2 x I^2S, 3 x UART.
- Ethernet MAC for physical LAN Communication (requires external PHY).
- 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.
- Motor PWM and up to 16-channels of LED PWM.

- Secure Boot and Flash Encryption.
- Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and RNG.

Different Ways to Program

A good hardware like ESP32 will be more user friendly if it can be programmed (writing code) in more than one way. And not surprisingly, the ESP32 supports multiple programming environments.

Some of the commonly used programming environments are:

- Arduino IDE
- PlatformIO IDE (VS Code)
- LUA
- MicroPython
- Espressif IDF (IoT Development Framework)
- JavaScript

As Arduino IDE is already a familiar environment, we will use the same to program ESP32 in our upcoming projects. But you can definitely try out others as well.

1.2 ESP32-CAM Development Board

Introduction

ESP32-CAM is a low-cost ESP32-based development board with onboard camera, small in size. It is an ideal solution for IoT application, prototypes constructions and DIY projects. The board integrates WiFi, traditional Bluetooth and low power BLE , with 2 high performance 32-bit LX6 CPUs. It adopts 7-stage pipeline architecture, on-chip sensor, Hall sensor, temperature sensor and so on, and its main frequency adjustment ranges from 80MHz to 240MHz. Fully compliant with WiFi 802.11b/g/n/e/i and Bluetooth 4.2 standards, it can be used as a master mode to build an independent network controller, or as a slave to other host MCUs to add networking capabilities to existing devices ESP32-CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.

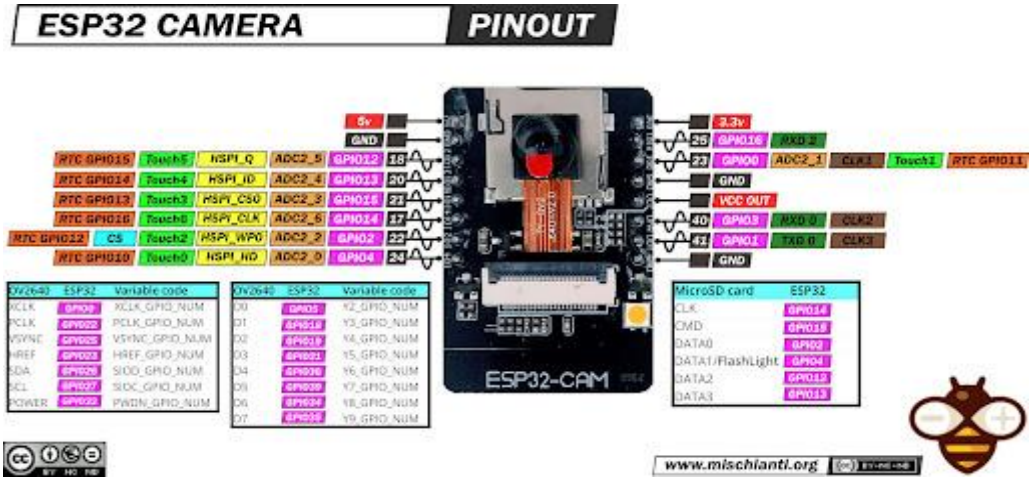

Fig 2: ESP32 Cam

Notes: 1.Please be sure that the power supply for the module should be at least 5V 2A, otherwise maybe there would be water ripple appearing on the image.
2.ESP32 GPIO32 pin is used to control the power of the camera, so when the camera is in working, pull GPIO32 pin low.
3.Since IO pin is connected to camera XCLK, it should be left floating in using, and do not connect it to high/low level.
4.The product has been equipped with default firmware before leaving the factory, and we do not provide additional ones for you to download. So, please be cautious when you choose to burn other firmwares.

Features

- Up to 160MHz clock speed, Summary computing power up to 600 DMIPS
- Built-in 520 KB
- SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC

- Support OV2640 and OV7670 cameras, Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

Specification

- SPI Flash: default 32Mbit
- RAM: built-in 520 KB+external 4MPSRAM
- Dimension: 27*40.5*4.5（±0.2）mm/1.06*1.59*0.18”
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards
- Wi-Fi: 802.11b/g/n/e/i
- Support Interface: UART, SPI, I2C, PWM
- Support TF card: maximum support 4G
- IO port: 9
- Serial Port Baud-rate: Default 115200 bps
- Image Output Format: JPEG(OV2640 support only), BMP, GRAYSCALE
- Spectrum Range: 2412 ~2484MHz

Circuit Diagram

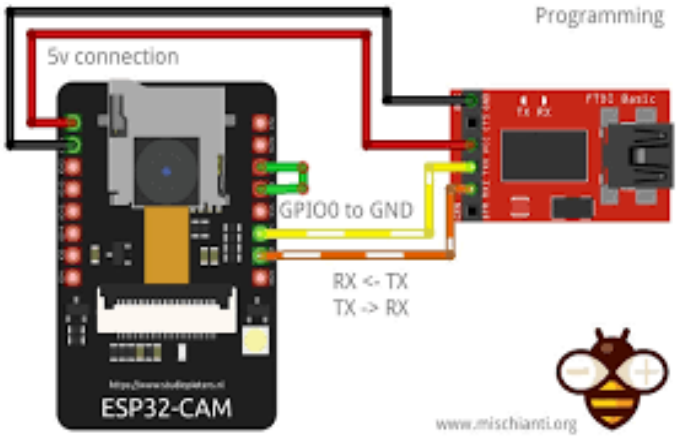

Fig 3: Circuit Diagram

1.3 FTDI

The FTDI (Future Technology Devices International) module refers to a family of products developed by FTDI, a company that specializes in the design and supply of silicon and software solutions for interfacing USB (Universal Serial Bus) to various electronic systems. FTDI's modules are commonly used to add USB connectivity to electronic devices that lack built-in USB capabilities.

Fig 4: FTDI

One of the most well-known products from FTDI is the FT232R USB-to-UART (Universal Asynchronous Receiver-Transmitter) IC (Integrated Circuit). This IC, often referred to simply as the FTDI chip, provides a bridge between USB and serial communication, allowing microcontrollers or other embedded systems to communicate with a host computer via USB.

FTDI modules are popular in the electronics and embedded systems communities for their ease of use and reliable performance. They often come in the form of USB-to-serial adapters, development boards, or modules that can be integrated into custom designs. These modules are widely used in scenarios where a direct USB connection is needed for programming, debugging, or interfacing with microcontrollers and other embedded devices.

1.4 Relay

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relayscontrol power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.

Fig 5: Single Channel relay

Fig 6: 4 Channel relay

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relayscontrol power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".
The Arduino Relay module allows a wide range of microcontroller such as Arduino, AVR ,PIC, ARM with digital outputs to control larger loads and devices like AC or DC Motors, electromagnets, solenoids, and incandescent light bulbs. This module is designed to be integrated with 2 relays that it is capable of control 2 relays.The relay shield use one QIANJI JQC-3F high-quality relay with rated load 7A/240VAC,10A/125VAC,10A/28VDC.The relay output state is individually indicated by a light-emitting diode.

Applications of relay:-

Relays are used wherever it is necessary to control a high power or high voltage circuit with a low power circuit, especially when galvanic isolation is desirable. The first application of relays was in long telegraph lines, where the weak signal received at an intermediate station could control a contact, regenerating the signal for further transmission. High-voltage or high-current devices can be controlled with small, low voltage wiring and pilots switches. Operators can be isolated from the high voltage circuit. Low power devices such as microprocessors can drive relays to control electrical loads beyond their direct drive capability. In an automobile, a starter relay allows the high current of the cranking motor to be controlled with small wiring and contacts in the ignition key.
Electromechanical switching systems including Strowger and Crossbar telephone exchanges made extensive use of relays in ancillary control circuits. The Relay Automatic Telephone Company also manufactured telephone exchanges based solely on relay switching techniques designed by Gotthilf Ansgarius Betulander. The first public relay based telephone exchange in the UK was installed in Fleetwood on 15 July 1922 and remained in service until 1959.
The use of relays for the logical control of complex switching systems like telephone exchanges was studied by Claude Shannon, who formalized the application of Boolean algebra to relay circuit design in A Symbolic Analysis of Relay and Switching Circuits. Relays can perform the basic operations of Boolean combinatorial logic. For example, the boolean AND function is realised by connecting normally open relay contacts in series, the OR function by connecting normally open contacts in parallel. Inversion of a logical input can be done with a normally closed contact. Relays were used for control of automated systems for machine tools and production lines. The Ladder programming language is often used for designing relay logic networks.
Early electro-mechanical computers such as the ARRA, Harvard Mark II, Zuse Z2, and Zuse Z3 used relays for logic and working registers. However, electronic devices proved faster and easier to use. Because relays are much more resistant than semiconductors to nuclear radiation, they are widely used in safety-critical logic, such as the control panels of radioactive waste-handling machinery.
Electromechanical protective relays are used to detect overload and other faults on electrical lines by opening and closing circuit breakers.

1.5 LED

LED stands for **light emitting diode**. LED lighting products produce light up to 90% more efficiently than incandescent light bulbs. How do they work? An electrical current passes through a microchip, which illuminates the tiny light sources we call LEDs and the result is visible light. To prevent performance issues, the heat LEDs produce is absorbed into a heat sink.

Lifetime of LED Lighting Products

The **useful life** of LED lighting products is defined differently than that of other light sources, such as incandescent or compact fluorescent lighting (CFL). LEDs typically do not "burn out" or fail. Instead, they experience 'lumen depreciation', wherein the brightness of the LED dims slowly over time. Unlike incandescent bulbs, LED "lifetime" is established on a prediction of when the light output decreases by 30 percent.

How are LEDs Used in Lighting

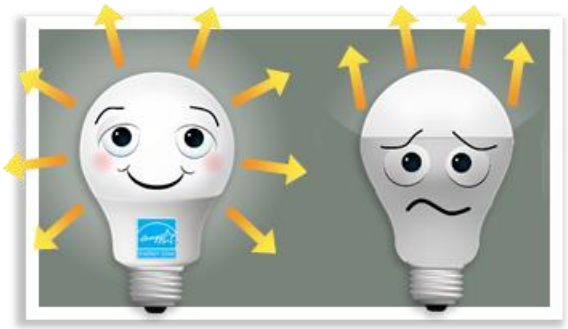

Fig 7: LED

LEDs are incorporated into bulbs and fixtures for general lighting applications. Small in size, LEDs provide unique design opportunities. Some LED bulb solutions may physically resemble familiar light bulbs and better match the appearance of traditional light bulbs. Some LED light fixtures may have LEDs built in as a permanent light source. There are also hybrid approaches where a non-traditional "bulb" or replaceable light source format is used and specially designed for a unique fixture. LEDs offer a tremendous opportunity for innovation in lighting form factors and fit a wider breadth of applications than traditional lighting technologies.

LEDs and Heat

LEDs use heat sinks to absorb the heat produced by the LED and dissipate it into the surrounding environment. This keeps LEDs from overheating and burning out. **Thermal management** is generally the single most important factor in the successful performance of an LED over its lifetime. The higher the temperature at which the LEDs are operated, the more quickly the light will degrade, and the shorter the useful life will be.

LED products use a variety of unique heat sink designs and configurations to manage heat. Today, advancements in materials have allowed manufacturers to design LED bulbs that match the shapes and sizes of traditional incandescent bulbs. Regardless of the heat sink design, all LED products that have earned the ENERGY STAR have been tested to ensure that they properly manage the heat so that the light output is properly maintained through the end of its rated life.

Block diagram

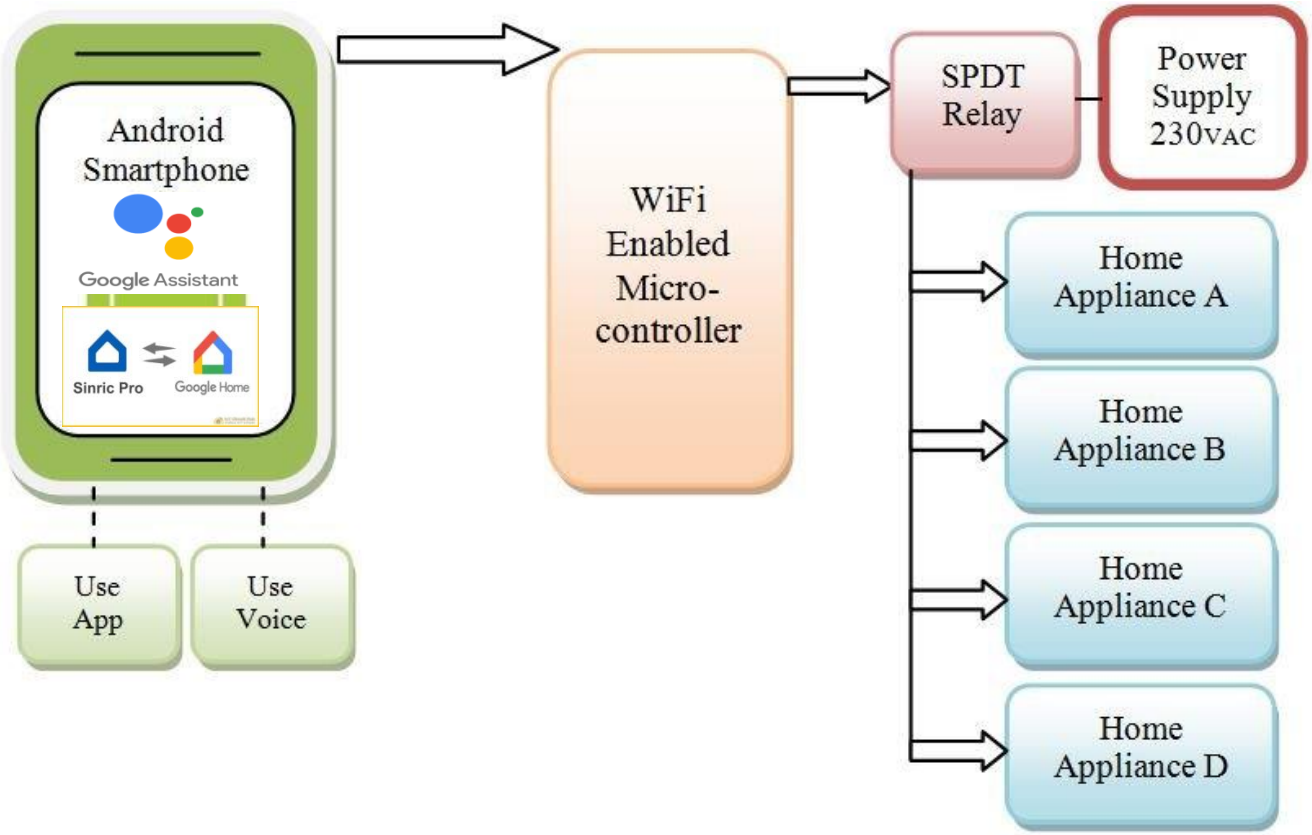

Fig 8: Home Automation Block Diagram

Circuit Diagram

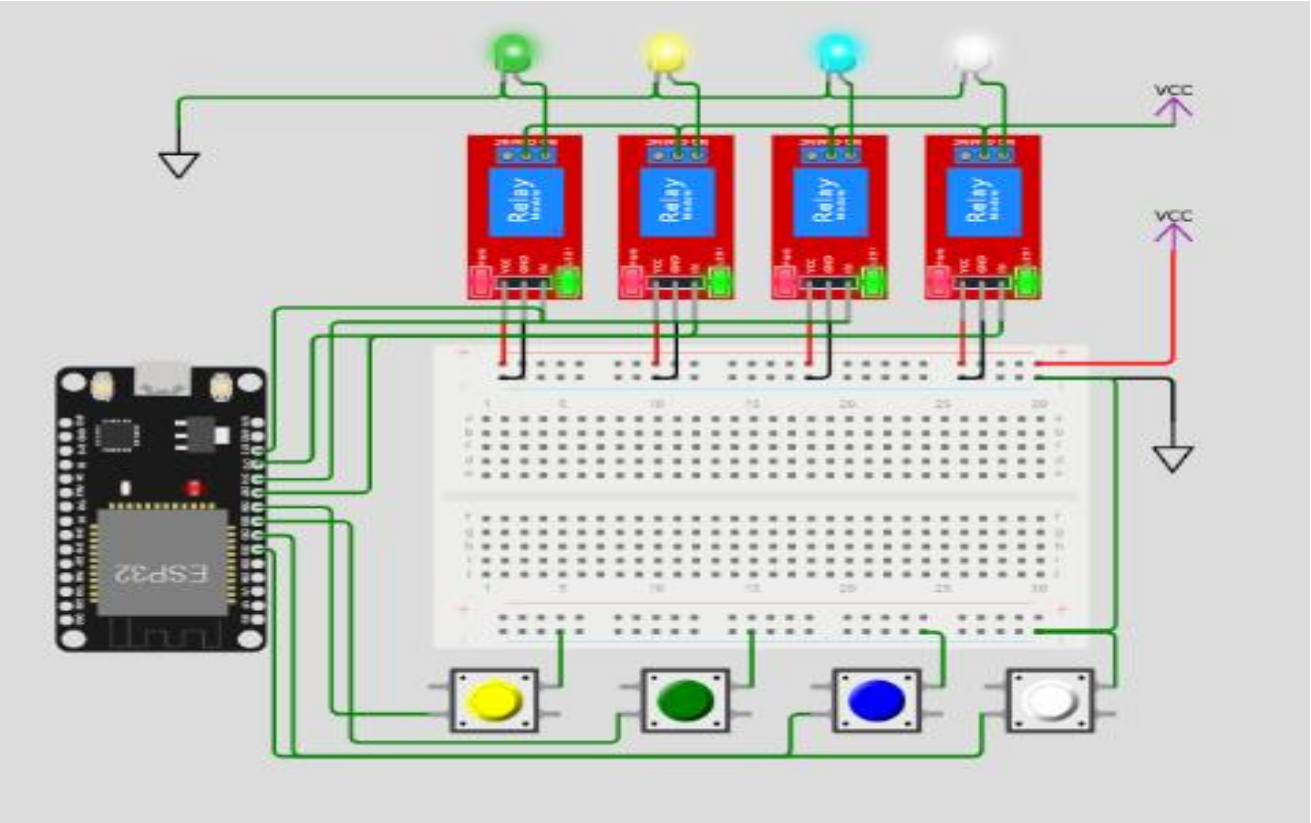

Fig 9: Circuit Diagram

Source code(ESP32)

```
// ESP32 Google Assistant + Alexa + Manual HomeAutomation with Sinric Pro
#include <Arduino.h>
#include <WiFi.h>
#include "SinricPro.h"
#include "SinricProSwitch.h"
#include <map>

#ifdef ENABLE_DEBUG
      #define DEBUG_ESP_PORT Serial
      #define NODEBUG_WEBSOCKETS
      #define NDEBUG
#endif
#define WIFI_SSID         "Redmi Note 8"      //Enter WiFi Name
#define WIFI_PASS         "Raj@1234"      //Enter WiFi Password
#define APP_KEY           "a9881071-8fc7-4759-9b0c-48b85b734393"                  //Enter APP-KEY
#define APP_SECRET        "85a2a1ca-9441-4fb4-84c6-7ba4c13b1f0c-893f4343-147d-483f-92ca-
8f8136769870"    //Enter APP-SECRET

//Enter the device IDs here
#define device_ID_1   "65867305cf6199223f5d2b3b"  //SWITCH 1 ID
#define device_ID_2   "6586734754ee86d4f7ee766c"  //SWITCH 2 ID
#define device_ID_3   "658672d8cf6199223f5d2afc"  //SWITCH 3 ID
#define device_ID_4   "XXXXXXXXXXXXXXXXXXX"  //SWITCH 4 ID

#define RelayPin1 23  //D23
#define RelayPin2 22  //D22
#define RelayPin3 21  //D21
#define RelayPin4 19  //D19

#define SwitchPin1 13  //D13
#define SwitchPin2 12  //D12
#define SwitchPin3 14  //D14
#define SwitchPin4 27  //D27

#define wifiLed   2   //D2

#define BAUD_RATE   9600

#define DEBOUNCE_TIME 250

typedef struct {      // struct for the std::map below
  int relayPIN;
  int flipSwitchPIN;
} deviceConfig_t;


std::map<String, deviceConfig_t> devices = {
```

```
    //{deviceId, {relayPIN,  flipSwitchPIN}}
    {device_ID_1, {  RelayPin1, SwitchPin1 }},
    {device_ID_2, {  RelayPin2, SwitchPin2 }},
    {device_ID_3, {  RelayPin3, SwitchPin3 }},
    {device_ID_4, {  RelayPin4, SwitchPin4 }}
};

typedef struct {      // struct for the std::map below
  String deviceId;
  bool lastFlipSwitchState;
  unsigned long lastFlipSwitchChange;
} flipSwitchConfig_t;

std::map<int, flipSwitchConfig_t> flipSwitches;

void setupRelays() {
  for (auto &device : devices) {
    int relayPIN = device.second.relayPIN;
    pinMode(relayPIN, OUTPUT);
    digitalWrite(relayPIN, HIGH);
  }
}

void setupFlipSwitches() {
  for (auto &device : devices)  {
    flipSwitchConfig_t flipSwitchConfig;

    flipSwitchConfig.deviceId = device.first;
    flipSwitchConfig.lastFlipSwitchChange = 0;
    flipSwitchConfig.lastFlipSwitchState = false;

    int flipSwitchPIN = device.second.flipSwitchPIN;

    flipSwitches[flipSwitchPIN] = flipSwitchConfig;
    pinMode(flipSwitchPIN, INPUT_PULLUP);
  }
}

bool onPowerState(String deviceId, bool &state)
{
  Serial.printf("%s: %s\r\n", deviceId.c_str(), state ? "on" : "off");
  int relayPIN = devices[deviceId].relayPIN;
  digitalWrite(relayPIN, !state);
  return true;
}

void handleFlipSwitches() {
  unsigned long actualMillis = millis();
  for (auto &flipSwitch : flipSwitches) {
    unsigned long lastFlipSwitchChange = flipSwitch.second.lastFlipSwitchChange;
```

```
    if (actualMillis - lastFlipSwitchChange > DEBOUNCE_TIME) {

      int flipSwitchPIN = flipSwitch.first;
      bool lastFlipSwitchState = flipSwitch.second.lastFlipSwitchState;
      bool flipSwitchState = digitalRead(flipSwitchPIN);
      if (flipSwitchState != lastFlipSwitchState) {
#ifdef TACTILE_BUTTON
        if (flipSwitchState) {
#endif
          flipSwitch.second.lastFlipSwitchChange = actualMillis;
          String deviceId = flipSwitch.second.deviceId;
          int relayPIN = devices[deviceId].relayPIN;
          bool newRelayState = !digitalRead(relayPIN);
          digitalWrite(relayPIN, newRelayState);

          SinricProSwitch &mySwitch = SinricPro[deviceId];
          mySwitch.sendPowerStateEvent(!newRelayState);
#ifdef TACTILE_BUTTON
        }
#endif
        flipSwitch.second.lastFlipSwitchState = flipSwitchState;
      }
    }
  }
}

void setupWiFi()
{
  Serial.printf("\r\n[Wifi]: Connecting");
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.printf(".");
    delay(250);
  }
  digitalWrite(wifiLed, HIGH);
  Serial.printf("connected!\r\n[WiFi]: IP-Address is %s\r\n", WiFi.localIP().toString().c_str());
}

void setupSinricPro()
{
  for (auto &device : devices)
  {
    const char *deviceId = device.first.c_str();
    SinricProSwitch &mySwitch = SinricPro[deviceId];
    mySwitch.onPowerState(onPowerState);
  }
```

```
  SinricPro.begin(APP_KEY, APP_SECRET);
  SinricPro.restoreDeviceStates(true);
}

void setup()
{
  Serial.begin(BAUD_RATE);

  pinMode(wifiLed, OUTPUT);
  digitalWrite(wifiLed, LOW);

  setupRelays();
  setupFlipSwitches();
  setupWiFi();
  setupSinricPro();
}

void loop()
{
  SinricPro.handle();
  handleFlipSwitches();
```

Source code(Esp32 Camera)

```
#include "esp_camera.h"
#include <WiFi.h>


 #define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

// Enter your WiFi credentials
const char* ssid = "Redmi Note 8";
const char* password = "Raj@1234";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
```

```
  if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
```

```
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}
void loop() {
  delay(10000);
}
```

Applications

- **Remote Access:** Control and monitor your home devices from anywhere with an internet connection, using a smartphone or computer.
- **Voice Control:** Integration with voice-activated assistants like Amazon Alexa, Google Assistant, allows hands-free control of smart devices.
- **Energy Management:** Automated control of lights, and appliances can lead to lower utility bills.
- **Smart Cameras:** Monitor your home with security cameras accessible remotely.
- **Time Savings:** Save time by remotely managing your home while away.
- **Save Energy:** Helps to save energy.

Limitations

- **Dependency on Power and Internet Connectivity:** Smart devices and automation systems rely heavily on electricity and a stable internet connection. Power outages or network issues could disrupt the functionality of these systems.
- **Security and Privacy Risks:** Increased connectivity raises concerns about potential security breaches. Hackers could exploit vulnerabilities in smart devices, compromising sensitive data or gaining unauthorized access to the home.
- **Maintenance and Updates:** Regular updates and maintenance are crucial for the security and optimal performance of home automation systems. Failure to keep devices and software up-to-date may result in vulnerabilities.

Conclusion

The system as the name indicates, 'Home automation' makes the system more flexible and provides attractive user interface compared to other home automation systems. In this system we integrate mobile devices into home automation systems. A novel architecture for a home automation system is proposed using the relatively new communication technologies. The system consists of mainly three components is a NodeMCU ESP32 microcontroller and relay circuits. WIFI is used as the communication channel between android phone and the NodeMCU ESP32 microcontroller. We hide the complexity of the notions involved in the home automation system by including them into a simple, but comprehensive set of related concepts. This simplification is needed to fit as much of the functionality on the limited space offered by a mobile device's display. This paper proposes a low cost, secure, ubiquitously accessible, auto-configurable, remotely controlled solution. The approach discussed in the paper is novel and has achieved the target to control home appliances remotely using the WiFi technology to connects system parts, satisfying user needs and requirements. WiFi technology capable solution has proved to be controlled remotely, provide home security and is costeffective as compared to the previously existing systems. Hence we can conclude that the required goals and objectives of home automation system have been achieved. The system design and architecture were discussed, and prototype presents the basic level of home appliance control and remote monitoring has been implemented. Finally, the proposed system is better from the scalability and flexibility point of view than the commercially available home automation systems.

References

1. Shelke, P., Kulkarni, S., Yelpale, S., Pawar, O., Singh, R., & Deshpande, K. (2018). A NodeMCU based home automation system. *Int. Res. J. Eng. Technol*, *5*(6), 127-129.

2. Asadullah, M., & Raza, A. (2016, November). An overview of home automation systems. In *2016 2nd international conference on robotics and artificial intelligence (ICRAI)* (pp. 27-31). IEEE.

3. Susany, R., & Rotar, R. (2020). Remote control android-based applications for a home automation implemented with Arduino Mega 2560 and ESP 32.

4. Babiuch, M., & Postulka, J. (2020). Smart home monitoring system using esp32 microcontrollers. *Internet of Things*, 82-101.

5. Palaniappan, S., Hariharan, N., Kesh, N. T., & Vidhyalakshimi, S. (2015). Home automation systems-A study. *International Journal of Computer Applications*, *116*(11).

6. Teymourzadeh, R., Ahmed, S. A., Chan, K. W., & Hoong, M. V. (2013, December). Smart GSM based home automation system. In *2013 IEEE Conference on Systems, Process & Control (ICSPC)* (pp. 306-309). IEEE.

7. Stolojescu-Crisan, C., Crisan, C., & Butunoi, B. P. (2021). An IoT-based smart home automation system. *Sensors*, *21*(11), 3784.

8. David, N., Chima, A., Ugochukwu, A., & Obinna, E. (2015). Design of a home automation system using arduino. *International Journal of Scientific & Engineering Research*, *6*(6), 795-801.

9. Kaur, I. (2010). Microcontroller based home automation system with security. *International journal of advanced computer science and applications*, *1*(6).

10. Van Der Werff, M., Gui, X., & Xu, W. L. (2005). A mobile-based home automation system.

11. Felix, C., & Raglend, I. J. (2011, July). Home automation using GSM. In *2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies* (pp. 15-19). IEEE.

12. Inoue, M., Uemura, K., Minagawa, Y., Esaki, M., & Honda, Y. (1985). A home automation system. *IEEE Transactions on Consumer Electronics*, (3), 516-527.

13. Vishwakarma, S. K., Upadhyaya, P., Kumari, B., & Mishra, A. K. (2019, April). Smart energy efficient home automation system using IoT. In *2019 4th international conference on internet of things: Smart innovation and usages (IoT-SIU)* (pp. 1-4). Ieee.