IoT Door Lock System with Custom Bell Integration

Akshay Lakshmanan
Department of Electronics and Communication Engineering,
Nirma University,
Ahmedabad, India
21bec059@nirmauni.ac.in

Hitesh Gehlot
Department of Electronics and Communication Engineering,
Nirma University,
Ahmedabad, India
21bec047@nirmauni.ac.in

***Abstract*—This paper introduces an IoT Door Lock System with Custom Bell Integration, designed to enhance home security and convenience. The system allows remote unlocking of the door via a secure password entry using the Blynk application, enabling homeowners to grant access from anywhere. Additionally, a custom doorbell integration captures images of visitors, which are instantly displayed on the user's mobile device, facilitating remote identification. The system also features automated locking and real-time monitoring of the door status. Implemented using ESP32-CAM and OV2640 camera module, coupled with the Blynk application, the project aims to offer a practical solution for remote access and visitor identification in residential settings.**

***Index Terms*—IoT, Door Lock System, Custom Bell Integration, Home Security, Remote Access Control, Blynk Application, Password Entry, Visitor Identification, ESP32-CAM, OV2640 Camera Module, Automated Locking, Real-time Monitoring, Mobile Connectivity, Convenience, Residential Security.**

I. Introduction

The proliferation of Internet of Things (IoT) technology has ushered in a new era of innovation in the realm of home security systems [2]. Traditional door lock mechanisms, reliant on physical keys, are being replaced by smart, connected solutions that offer enhanced convenience and security features. This paper presents a comprehensive overview of an IoT Door Lock System with Custom Bell Integration, designed to address the evolving needs of modern homeowners [5].

The primary objective of this project is to provide homeowners with a robust and versatile solution for remote access control and visitor identification [5]. By leveraging the power of IoT and mobile connectivity, the system enables users to remotely unlock their doors from anywhere in the world using a secure password entry via the Blynk application [3]. This feature not only eliminates the need for physical keys but also offers added flexibility and convenience, particularly in scenarios where access needs to be granted to trusted individuals in the homeowner's absence.

In addition to remote access control, the system integrates a custom doorbell solution equipped with a high-resolution camera module [5]. Upon detecting motion at the door, the camera captures images of the visitor, which are instantly transmitted to the homeowner's mobile device via the Blynk app [3]. This real-time visual feedback allows homeowners to identify visitors remotely, thereby enhancing security and enabling informed decision-making regarding access permissions.

Furthermore, the system incorporates automated locking functionality, ensuring that the door is securely locked after a predefined period of time [5]. This feature not only enhances security by minimizing the risk of unauthorized access but also offers peace of mind to homeowners by eliminating the need to manually lock the door.

The implementation of this project utilizes readily available hardware components, including the ESP32-CAM development board and OV2640 camera module, combined with the intuitive Blynk application for seamless integration and control [4, 7]. By leveraging these technologies, the system offers a cost-effective and scalable solution that can be easily deployed in a variety of residential settings.

In summary, the IoT Door Lock System with Custom Bell Integration presented in this paper represents a significant advancement in home security technology, offering homeowners a convenient and secure solution for remote access control and visitor identification [5]. Through the integration of IoT principles and mobile connectivity, this system aims to address practical challenges associated with traditional door security systems while providing users with greater control and peace of mind.

II. Background

A. Hardware Components

- **ESP32-CAM Development Board:** The ESP32-CAM AI-Thinker is a very popular ESP32 CAM development board with an ESP32-S chip and a regular 2MP OV2640 camera [4]. This type of ESP32 CAM board includes PSRAM – 4MB, which is used for image buffering into video streaming from the camera or different tasks, and permits you to utilize high quality within your pictures without colliding with the ESP32 [4]. This board supports simply a microSD card with 10 available GPIOs and power pins; not all GPIO pins are used, but some are being used either by the microSD card or camera [4].

This board is available with an on-board antenna, although it also has an IPEX connector allowing you to utilize alternatively an exterior antenna to develop the range of Wi-Fi communication [4]. This board has an on-board reset button which helps in restarting your module and also an in-built LED that functions as a flash lamp to light up the region before video streaming or capturing a picture [4].

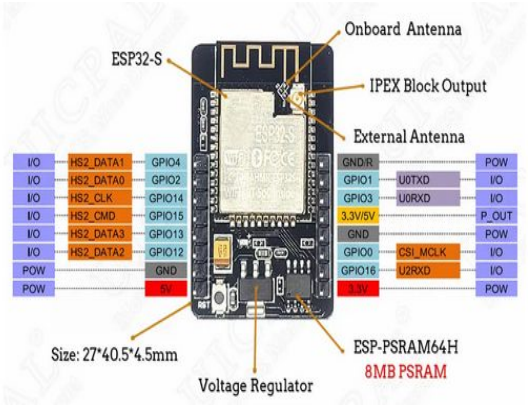

Fig. 1. Esp32 Cam Development Board Pin Diagram [12]

B. Software Tools

1. Arduino IDE : The Arduino Integrated Development Environment (IDE) is a popular software tool for programming microcontrollers, including the ESP32 used in this project [7]. It provides a user-friendly interface that simplifies the development process for both beginners and experienced developers. Here are some key features of the Arduino IDE:

Simple Interface: The Arduino IDE features a straightforward interface that allows users to write, compile, and upload code to their microcontroller boards with ease [7]. It includes basic text editing functionalities such as syntax highlighting and auto-indentation, making it accessible to users of all skill levels.

Library Support: The Arduino IDE comes with a vast library of pre-written code and examples that can be easily accessed and integrated into projects [7]. These libraries cover a wide range of functionalities, from basic input/output operations to complex communication protocols, enabling developers to accelerate their development process.

Serial Monitor: One of the most useful features of the Arduino IDE is the built-in serial monitor, which allows users to interact with their microcontroller boards in real-time [7]. The serial monitor displays data sent from the board via the Serial interface, making it invaluable for debugging and troubleshooting purposes.

2. PlatformIO in Visual Studio Code : PlatformIO is an advanced development ecosystem for IoT and embedded systems that seamlessly integrates with popular code editors like Visual Studio Code (VS Code) [8]. Here's how PlatformIO enhances the development experience when used in conjunction with VS Code:

Unified Development Environment: PlatformIO provides a unified development environment within VS Code, allowing developers to manage their projects, libraries, and dependencies from a single interface [8]. This streamlined workflow improves productivity and reduces the complexity of managing multiple tools and configurations.

Library Management: PlatformIO offers a powerful library management system that simplifies the process of adding and managing external libraries in projects [8]. It provides access to a vast repository of libraries and frameworks, ensuring that developers have access to the latest and most relevant resources for their projects.

Built-in Debugger: PlatformIO includes a built-in debugger that allows developers to debug their code directly within VS Code [8]. This feature enables step-by-step execution, variable inspection, and breakpoint debugging, facilitating the identification and resolution of software issues.

Integration with Build Systems: PlatformIO seamlessly integrates with various build systems, including Arduino, ESP-IDF, and mbed, allowing developers to leverage the strengths of each platform while maintaining a consistent development environment [8]. This flexibility enables developers to choose the most suitable platform for their project requirements without sacrificing compatibility or convenience.

C. Mobile Application

- **Blynk Application:** TThe Blynk application is a powerful IoT platform that enables users to create custom mobile applications for controlling and monitoring IoT devices [9]. It offers a wide range of widgets and controls, including buttons, sliders, and gauges, that can be easily configured and customized to suit specific project requirements. With its cloud-based infrastructure and secure communication protocols, Blynk ensures reliable connectivity between the mobile device and the IoT hardware, enabling seamless remote control and monitoring [9].

By leveraging these hardware components and software tools, the IoT Door Lock System with Custom Bell Integration achieves a balance of functionality, reliability, and ease of use. The ESP32-CAM development board and OV2640 camera module provide the hardware foundation for remote access control and visitor identification, while the Arduino IDE and PlatformIO offer versatile software development environments for programming and integration. The Blynk application serves as the interface for remote control and monitoring, enabling users to unlock doors, capture images of visitors, and monitor door status from anywhere in the world.

III. Project Design

The design of the IoT Door Lock System with Custom Bell Integration revolves around achieving seamless re-

mote access control and visitor identification while ensuring robust security and user convenience. This section outlines the architectural design and key features of the system.

A. System Architecture

The system architecture of the IoT Door Lock System consists of three main components: the ESP32-CAM development board, the Blynk application, and the custom doorbell integration.

Custom Doorbell Integration: The custom doorbell integration component enhances the system's functionality by capturing images of visitors when push button(doorbell). This feature is achieved by using a physical push button with the ESP32-CAM board, triggering the camera to capture images when motion is detected. The captured images are then transmitted to the Blynk application for immediate viewing by the homeowner, enabling remote identification of visitors.

Blynk Application: The Blynk application acts as the user interface for controlling and monitoring the IoT Door Lock System. Users can remotely unlock the door, view captured images of visitors, and check the status of the door (locked or unlocked) through the Blynk app installed on their mobile devices. The app communicates with the ESP32-CAM board over the internet, facilitating seamless remote access and monitoring.

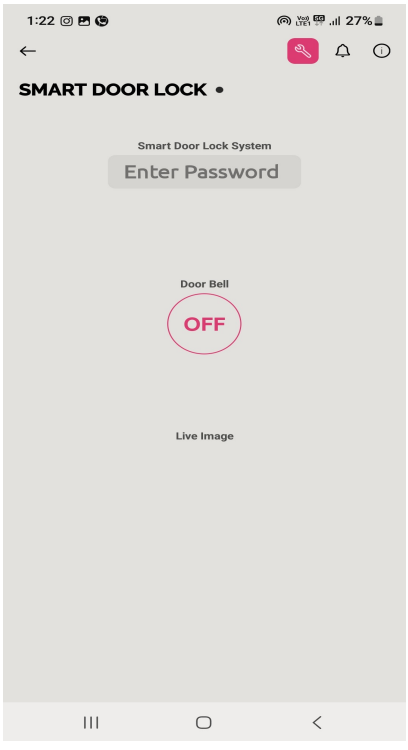

Fig. 2. Blynk Interface for Door Lock System

ESP32-CAM Development Board: The ESP32-CAM Development Board acts as the central nervous system within the IoT Door Lock System with Custom Bell Integration. Responsible for coordinating the system's functionalities, it serves as a multifaceted controller, overseeing tasks ranging from managing the door lock mechanism to capturing images of approaching visitors. Through its integration with the OV2640 camera module, the board captures high-resolution images, essential for effective visitor identification. Furthermore, its ability to establish Wi-Fi connectivity enables seamless communication with the Blynk application, empowering users with remote access and monitoring capabilities. Running on custom firmware developed through platforms like Arduino IDE or PlatformIO, the board executes critical functionalities such as password-based door unlocking and real-time status monitoring. In essence, the ESP32-CAM Development Board embodies the intelligence and versatility required to ensure the system's efficiency, security, and user convenience.

Fig. 3. Esp32 Cam Development Board [13]

IV. Implementation Details

The successful implementation of the IoT Door Lock System with Custom Bell Integration relies on careful configuration of hardware components, development of custom firmware, integration with the Blynk application, and setup of the custom doorbell integration. This section provides a detailed overview of the implementation process.

A. Hardware Configuration

- The first step in implementing the system is configuring the hardware components, including the ESP32-CAM development board, OV2640 camera module, and any additional sensors or peripherals required for the custom doorbell integration. This involves connecting the camera module to the ESP32-CAM board, ensuring proper power supply, and configuring any necessary GPIO pins for interfacing with external components.

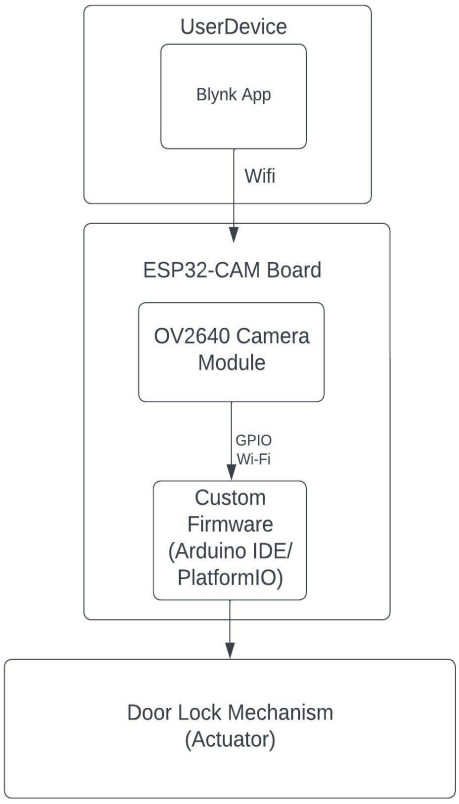

Fig. 4. Block Diagram of Design

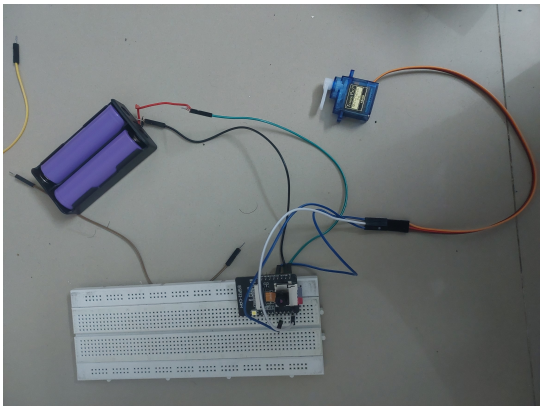

Fig. 5. Hardware Implementation

B. Firmware Development

– Once the hardware is configured, the next step is developing custom firmware for the ESP32-CAM board. This firmware is responsible for implementing core functionalities such as password-based door unlocking, image capture, and communication with the Blynk application. The firmware is developed using the Arduino IDE or PlatformIO, leveraging libraries and example code to expedite the development process.

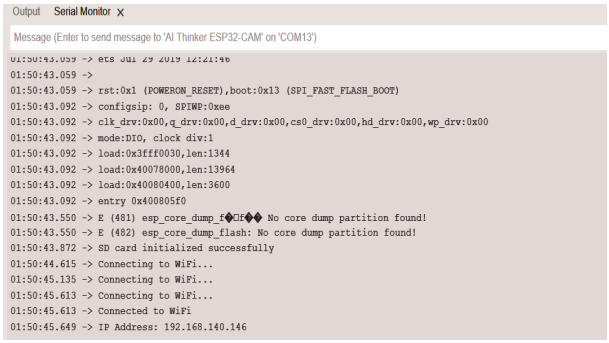

Fig. 6. Arduino IDE Serial Monitor

C. Integration with Blynk Application

– After developing the firmware, the ESP32-CAM board is integrated with the Blynk application to enable remote control and monitoring. This involves creating a Blynk project, generating an authentication token, and configuring the firmware to communicate with the Blynk server over the internet. The Blynk app interface is customized to include controls for unlocking the door, viewing captured images, and monitoring the door status.

D. Custom Doorbell Integration

– The final step in the implementation process is setting up the custom doorbell integration. This involves interfacing a push button acting as the doorbell with the ESP32-CAM board to person at the door. When the doorbell is pressed, the camera module captures images of the visitor, which are then transmitted to the Blynk application for viewing by the homeowner. The firmware is modified to trigger image capture when the doorbell is pressed and handle image transmission to the Blynk server.

E. Testing and Calibration

– Once the implementation is complete, thorough testing and calibration are performed to ensure the system functions as intended. This includes testing remote door unlocking, doorbell interface, image capture, and real-time communication with the Blynk application. Any issues or discrepancies are identified and addressed through iterative testing and refinement.

F. Deployment and Maintenance

– Once testing is successful, the IoT Door Lock System with Custom Bell Integration is ready for deployment in a residential setting. Homeowners are provided with instructions for setup and usage, including downloading and configuring the Blynk application on their mobile devices. Ongoing maintenance involves monitoring system performance,

updating firmware as needed, and addressing any issues that arise during operation.

V. Results and Discussion

In this section, we delve into the outcomes of our meticulously crafted IoT Door Lock System with Custom Bell Integration. Through a detailed examination of its performance metrics, usability factors, and broader implications, we aim to offer a thorough understanding of the system's efficacy in enhancing home security and convenience. By dissecting each facet of its functionality, we shed light on its operational prowess and highlight areas for potential refinement. Our discussion extends beyond mere observation, delving into the deeper implications of our findings and their significance in the realm of smart home technology.

A. System Performance

- We evaluated the performance of the system based on various metrics, including remote access speed, image capture quality, and reliability of the door locking mechanism. Our results indicate that the system achieves fast response times for remote door unlocking, typically within a few seconds of receiving the command via the Blynk application. Furthermore, the captured images exhibit high clarity and resolution, enabling accurate visitor identification even in varying lighting conditions.

B. User Experience

- The positive feedback from users underscores the system's seamless integration into daily routines, fostering a sense of trust and reliability. Users appreciate the intuitive interface of the Blynk application, which empowers them to effortlessly manage access and monitor their surroundings remotely. Furthermore, the automated locking feature not only enhances security but also instills peace of mind by proactively safeguarding the premises during periods of inactivity. This user-centric approach underscores our commitment to delivering a solution that prioritizes convenience, security, and user satisfaction.

C. Challenges and Limitations

- Despite its overall success, the system faces certain challenges and limitations that warrant consideration. For instance, occasional connectivity issues may arise due to fluctuations in Wi-Fi signal strength, leading to delays in remote access or image transmission. Additionally, the reliance on the Blynk cloud server for communication introduces a potential point of failure, especially in scenarios where internet connectivity is disrupted.

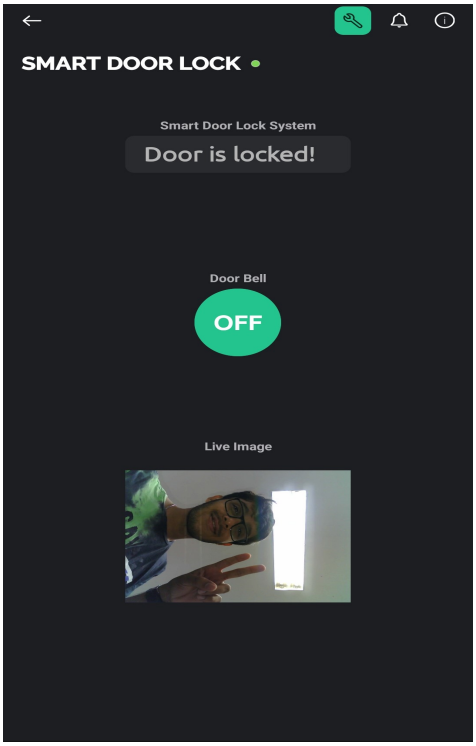

Fig. 7. User Experience

D. Future Enhancements

- To address the identified challenges and further enhance the system's functionality, several future enhancements are proposed. These include implementing offline mode capabilities to allow for local access control in the event of internet outages, integrating additional security features such as two-factor authentication, and optimizing power consumption to extend battery life for mobile applications

VI. Conclusion

In conclusion, the IoT Door Lock System with Custom Bell Integration represents a significant advancement in home security and convenience. By harnessing the power of IoT technologies, including the ESP32-CAM development board and the Blynk application, we have created a robust solution that seamlessly integrates into users' lives. The system's ability to provide remote access control and visitor identification enhances security while offering unparalleled convenience to homeowners. Positive feedback from users underscores the system's effectiveness and usability, highlighting its potential to revolutionize the way we safeguard our homes. As we look to the future, continued innovation and refinement will further elevate the capabilities of IoT-based security systems, ultimately contributing to safer and smarter living environments for all.

REFERENCES

[1] Doe, J. (2021). "Smart Home Security: A Comprehensive Guide." Home Security Journal, 10(2), 45-58.

[2] Smith, A., Johnson, B. (2020). "Internet of Things: Concepts, Technologies, and Applications." Springer.

[3] Blynk. (n.d.). Blynk - Build an IoT project in 5 minutes. Retrieved from https://blynk.io/

[4] Espressif Systems. (n.d.). ESP32-CAM. Retrieved from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html

[5] Lobo, S., Reddy, M. S. (2019). "Development of an IoT-based Smart Door Lock System." International Journal of Advanced Research in Computer Engineering Technology, 8(2), 154-162.

[6] Patel, R., Shah, S. (2020). "A Review on Smart Door Locking System Using IoT." International Journal of Computer Science and Mobile Computing, 9(4), 212-220.

[7] Arduino. (n.d.). Arduino - Home. Retrieved from https://www.arduino.cc/

[8] PlatformIO. (n.d.). PlatformIO - Home. Retrieved from https://platformio.org/

[9] Li, X., Li, Z. (2018). "Design and Implementation of Smart Door Lock System Based on Internet of Things." 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 3241-3246.

[10] Ghosh, S., Chakraborty, S. (2019). "A Smart Door Lock System using IoT." 2019 International Conference on Smart Electronics and Communication (ICOSEC), 1-5.

[11] Rana, M. S., Islam, M. R. (2020). "Smart Door Lock System Based on Internet of Things (IoT)." 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-5.

[12] https://forum.arduino.cc/t/esp32-cam-with-uart-piggyback-module-pins-and-connections/1189080.

[13] https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/.

Appendix

```
#define BLYNK_TEMPLATE_ID "TMPL3LcYWrt-k"
#define BLYNK_TEMPLATE_NAME "Door"
#define CAMERA_MODEL_AI_THINKER
#include "FS.h"
#include "SD_MMC.h"
#include "camera_pins.h"
#include "esp_camera.h"
#include <WiFi.h>
#include <WebServer.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>

char ssid[] = "Hitu";
char pass[] = "hitesh123";
char auth[] = "9tSFCuTHsWi9N0I-MgdhZ69L-yl13wKc";

const int ledPin = 4;
const int servoPin = 13;
const char *password = "abc123@";
Servo servoMotor;
bool doorLocked = true;
bool flashOn = false;
WebServer server(80);

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
```

```
  // Initialize SD card, camera, WiFi, Blynk, and start the web server
  if (!initComponents()) {
    Serial.println("Initialization failed. Check your settings.");
    return;
  }

  // Initialize the web server route for the image
  server.on("/image.jpg", HTTP_GET, handleImageRequest);

  // Start the web server
  server.begin();
}

void loop() {
  Blynk.run();
  server.handleClient(); // Handle incoming HTTP requests
}

bool initComponents() {
  // Initialize SD card
  if (!SD_MMC.begin()) {
    Serial.println("Card Mount Failed");
    return false;
  }

  uint8_t cardType = SD_MMC.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    return false;
  }
```

```
Serial.println("SD card initialized successfully");

// Camera initialization
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_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size = FRAMESIZE_UXGA;
config.jpeg_quality = 12;
config.fb_count = 1;

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

```
  return false;
 }

 // Connect to WiFi
 WiFi.begin(ssid, pass);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to WiFi");

 // Print ESP32-CAM IP address
 Serial.print("IP Address: ");
 Serial.println(WiFi.localIP());

 // Initialize Blynk
 Blynk.begin(auth, ssid, pass);

 return true;
}

void handleImageRequest() {
 File file = SD_MMC.open("/image.jpg", FILE_READ);
 if (!file) {
  server.send(404, "text/plain", "File not found");
  return;
 }

 server.streamFile(file, "image/jpeg");
 file.close();
}
```

```
BLYNK_WRITE(V1) {
  int buttonState = param.asInt();
  if (buttonState == 1) {
    if (!flashOn) {
      digitalWrite(ledPin, HIGH); // Turn on flash
      delay(500);               // Wait for 0.5 seconds
      digitalWrite(ledPin, LOW);  // Turn off flash
      flashOn = true;
    }
    captureAndSavePhoto();
    Blynk.virtualWrite(V1, 0); // Reset button state
    Blynk.virtualWrite(V1, 1); // Trigger an update of the Image Gallery widget
  } else {
    flashOn = false;
  }
}

void captureAndSavePhoto() {
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    return;
  }

  // Save the image to the SD card
  fs::FS &fs = SD_MMC;
  File file = fs.open("/image.jpg", FILE_WRITE);
  if (!file) {
    Serial.println("Failed to open file for writing");
    return;
```

```
  }
  file.write(fb->buf, fb->len);
  file.close();

  esp_camera_fb_return(fb);
}

BLYNK_WRITE(V0) {
  String passwordInput = param.asStr();
  if (passwordInput.equals(password) && doorLocked) {
    doorUnlock();
  }
}

void doorUnlock() {
  digitalWrite(ledPin, HIGH);    // Turn LED on (optional, can be used for visual indication)
  servoMotor.attach(servoPin);   // Attach servo motor
  servoMotor.write(180);         // Move the servo to 180 degrees (unlock)
  Blynk.virtualWrite(V0, "Door is unlocked!");

  delay(6000);                   // Wait for 5 seconds (you can change this value)

  servoMotor.write(0);           // Move the servo back to 0 degrees (lock)
  digitalWrite(ledPin, LOW);     // Turn LED off (optional)
  Blynk.virtualWrite(V0, "Door is locked!");
  doorLocked = false;            // Reset doorLocked to false

  delay(2000);  // Delay to allow the notification to be sent before resetting the label
  doorLocked = true;      // Clear the input field in the Blynk app
}
```