Internet of Things

Higher National Diploma in Software Engineering

Assessment 4 Documentation

22.2F

COHDSE222F-011 - S.E. KOPERAHEWA

COHDSE222F-013 - K.G.D.S. HANSAKA

COHDSE222F-021 - W.M.T.D UDANGAWE

COHDSE222F-024 - S.S. WIMALASIRI

COHDSE222F-027 - S.N. OCKERSZ

School of Computing and Engineering

National Institute of Business Management

Colombo-7

Preamble

Abstract

The Smart Burglar Alarm System is a cutting-edge security solution that uses IoT technology to protect buildings from unauthorized access. By integrating Arduino PIR sensors with ESP32 boards, the system detects motion in different areas of the building, such as the garage, lobby, living room, and courtyard. The sensors wirelessly transmit data to a central receiver, enabling real-time monitoring through a web or mobile application.

With a focus on power efficiency, the system conserves energy by putting the devices into deep sleep mode when not in use. They wake up only when motion is detected, ensuring reliable operation while minimizing power consumption.

The central station, connected to Wi-Fi, acts as the control hub. It allows users to remotely monitor the status of each area and receive instant alerts in the form of visible and audible notifications whenever unauthorized access is detected.

To enhance security, the central station is equipped with an ESP-32 CAM OV2640 module. This camera captures photos when motion is detected and uploads them to a cloud database. Users can view these photos, which are accompanied by timestamps, providing valuable visual evidence of any suspicious activity.

The Smart Burglar Alarm System offers a scalable and flexible solution for building security. Its modular design allows for easy customization and expansion to meet specific needs. With its advanced features and user-friendly interface, the system provides an effective way to protect homes and businesses from intruders.

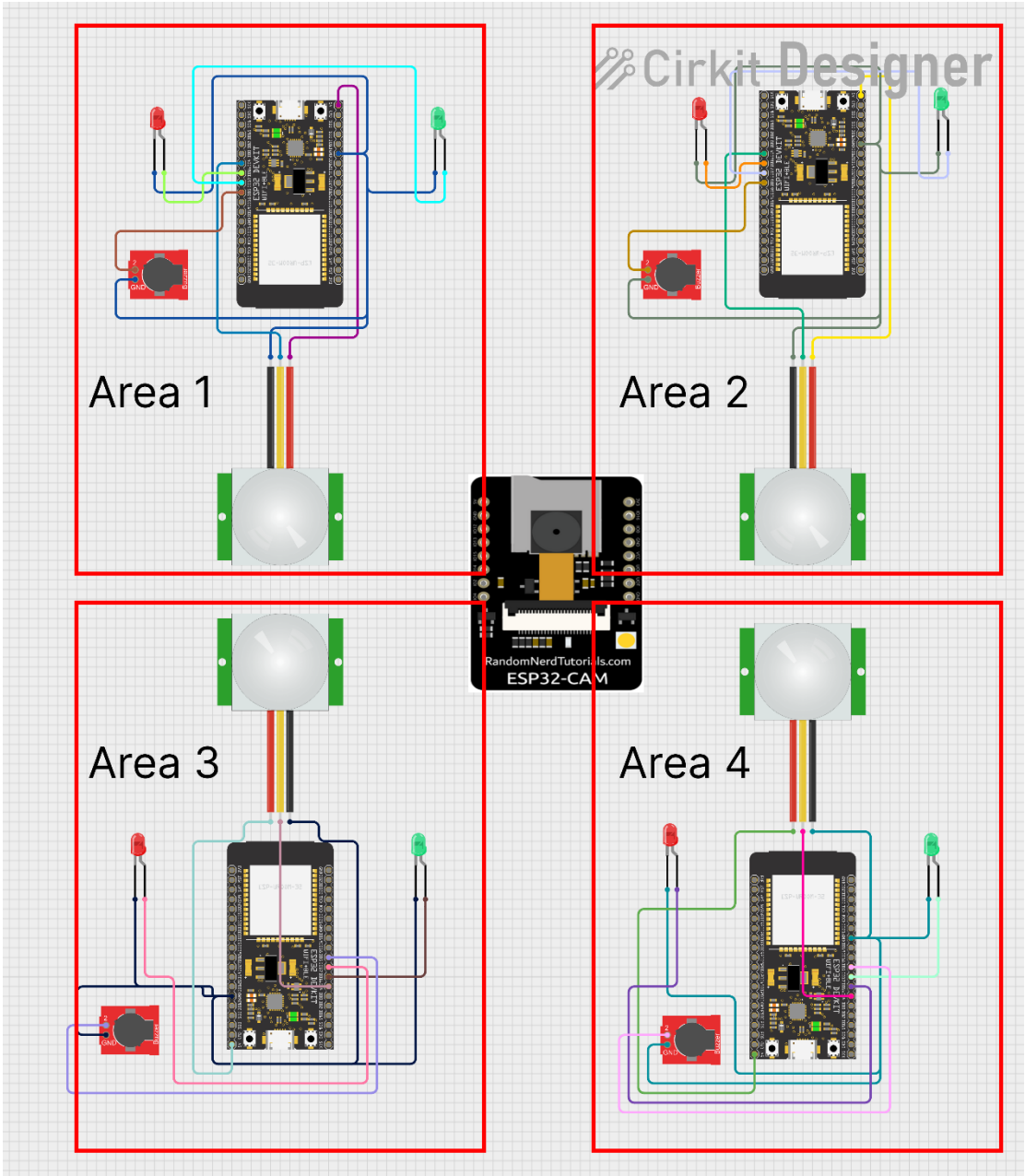
Table of Contents

Chapter 1: Introduction

In today's security-conscious world, the Smart Burglar Alarm System has emerged as a revolutionary solution that leverages IoT technology to safeguard buildings. By seamlessly integrating Arduino PIR sensors, NodeMCU boards, and ESP-32 CAM OV2640 modules, this system creates a comprehensive and advanced defense against unauthorized access. What makes this system truly unique is its ability to deliver real-time updates and notifications through a user-friendly web or mobile application, allowing users to effortlessly monitor their building's security status from anywhere, regardless of their physical location. The system employs intelligent power management techniques to optimize energy consumption, making it an excellent choice for applications where power efficiency is critical. By utilizing deep sleep mode and selectively waking up only when motion is detected, the devices strike a careful balance between reliability and power conservation, enabling prolonged operation without compromising security measures. The central station acts as the control hub, facilitating seamless communication with the sensors and serving as the gateway for real-time data transmission. Equipped with the ESP-32 CAM OV2640 module, the central station enhances security by capturing high-resolution photos when motion is detected. These photos are securely uploaded to a cloud-based database, complete with timestamps, providing valuable visual evidence in case of any suspicious activities. The Smart Burglar Alarm System goes beyond traditional security measures by empowering users with remote monitoring capabilities. Through a user-friendly interface, individuals can effortlessly access the application and receive instant notifications and alerts when unauthorized access is detected. The system's versatility is demonstrated by its ability to monitor multiple areas within a building, ensuring comprehensive coverage and offering peace of mind. In summary, the Smart Burglar Alarm System represents a significant advancement in building security, utilizing IoT advancements to deliver an intelligent, scalable, and user-centric solution that provides the utmost protection against unauthorized access. By seamlessly integrating cutting-edge hardware, implementing efficient power management techniques, and enabling remote monitoring capabilities, this system establishes itself as a transformative force in the field of building security.

Chapter 2: Materials and Methods

Diagram



Materials

Equipment	Units
ESP32	4
ESP OV2640	1
Buzzer	4
LED(Red)	4
LED(Green)	4
PIR Sensor	4

Code

Sender Code for Area 1

```
#include <esp_now.h>
#include <WiFi.h>
#include <esp_sleep.h>
#include <esp_wifi.h>

// REPLACE WITH YOUR RECEIVER MAC Address
uint8_t broadcastAddress[] = [redacted]

constexpr char WIFI_SSID[] = [redacted]

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  char deviceName[20];
  bool movementDetected;
} struct_message;

// Create a struct_message called myData
struct_message myData;

esp_now_peer_info_t peerInfo;
bool packetSent = false;  // Flag to track if the packet has been sent

//Initiate led's and buzzer
const int redLedPin = GPIO_NUM_2;
const int greenLedPin = GPIO_NUM_15;
const int buzzerPin = GPIO_NUM_5;

int32_t getWiFiChannel(const char *ssid) {
  if (int32_t n = WiFi.scanNetworks()) {
    for (uint8_t i = 0; i < n; i++) {
      if (!strcmp(ssid, WiFi.SSID(i).c_str())) {
        return WiFi.channel(i);
      }
    }
  }
  return 0;
}
```

```
void setup() {
  // Init Serial Monitor
  Serial.begin(115200);
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);

  int32_t channel = getWiFiChannel(WIFI_SSID);
  esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of transmitted packet
  esp_now_register_send_cb([](const uint8_t *mac_addr, esp_now_send_status_t status) {
    Serial.print("\r\nLast Packet Send Status:\t");
    if (status == ESP_NOW_SEND_SUCCESS) {
      Serial.println("Delivery Success");
      packetSent = true;  // Set flag to true when packet is successfully sent
    } else {
      Serial.println("Delivery Fail");
    }
  });

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
```

```
  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("\nFailed to add peer");
    return;
  }

  if (esp_sleep_enable_ext0_wakeup(GPIO_NUM_4, 1) == ESP_OK) {
    Serial.println("Movement Detected");

    digitalWrite(redLedPin, 1);   // Turn on the red LED
    digitalWrite(greenLedPin, 0);  // Turn off the green LED
    digitalWrite(buzzerPin, 1);   // Turn on the buzzer
    delay(2000);                    // Wait for 2 seconds
    digitalWrite(redLedPin, 0);    // Turn off the red LED
    digitalWrite(buzzerPin, 0);    // Turn off the buzzer
    digitalWrite(greenLedPin, 1);

    // Set values to send
    strcpy(myData.deviceName, "Area 1");
    myData.movementDetected = true;
  } else {
    digitalWrite(redLedPin, 0);     // Turn off the red LED
    digitalWrite(greenLedPin, 1);  // Turn on the green LED
  }

  // Send message via ESP-NOW until packet is sent successfully
  while (!packetSent) {
    esp_now_send(broadcastAddress, (uint8_t *)&myData, sizeof(myData));
    delay(100);  // Wait for the transmission to complete
  }
  Serial.println("\nGoing to Sleep");
  // Go to deep sleep mode
  esp_deep_sleep_start();
}

void loop() {
  // Empty loop as it won't be executed during deep sleep
}
```

Sender Code for Area 2

```
#include <esp_now.h>
#include <WiFi.h>
#include <esp_sleep.h>
#include <esp_wifi.h>

// REPLACE WITH YOUR RECEIVER MAC Address
uint8_t broadcastAddress[] = {

constexpr char WIFI_SSID[] =

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  char deviceName[20];
  bool movementDetected;
} struct_message;

// Create a struct_message called myData
struct_message myData;

esp_now_peer_info_t peerInfo;
bool packetSent = false;  // Flag to track if the packet has been sent

//Initiate led's and buzzer
const int redLedPin = GPIO_NUM_2;
const int greenLedPin = GPIO_NUM_15;
const int buzzerPin = GPIO_NUM_5;

int32_t getWiFiChannel(const char *ssid) {
  if (int32_t n = WiFi.scanNetworks()) {
    for (uint8_t i = 0; i < n; i++) {
      if (!strcmp(ssid, WiFi.SSID(i).c_str())) {
        return WiFi.channel(i);
      }
    }
  }
  return 0;
}
```

```
void setup() {
  // Init Serial Monitor
  Serial.begin(115200);
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);

  int32_t channel = getWiFiChannel(WIFI_SSID);
  esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of transmitted packet
  esp_now_register_send_cb([](const uint8_t *mac_addr, esp_now_send_status_t status) {
    Serial.print("\r\nLast Packet Send Status:\t");
    if (status == ESP_NOW_SEND_SUCCESS) {
      Serial.println("Delivery Success");
      packetSent = true;  // Set flag to true when packet is successfully sent
    } else {
      Serial.println("Delivery Fail");
    }
  });

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
```

```
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("\nFailed to add peer");
    return;
  }

  if (esp_sleep_enable_ext0_wakeup(GPIO_NUM_4, 1) == ESP_OK) {
    Serial.println("Movement Detected");

    digitalWrite(redLedPin, 1);   // Turn on the red LED
    digitalWrite(greenLedPin, 0);  // Turn off the green LED
    digitalWrite(buzzerPin, 1);   // Turn on the buzzer
    delay(2000);                     // Wait for 2 seconds
    digitalWrite(redLedPin, 0);    // Turn off the red LED
    digitalWrite(buzzerPin, 0);    // Turn off the buzzer
    digitalWrite(greenLedPin, 1);

    // Set values to send
    strcpy(myData.deviceName, "Area 2");
    myData.movementDetected = true;
  } else {
    digitalWrite(redLedPin, 0);     // Turn off the red LED
    digitalWrite(greenLedPin, 1);  // Turn on the green LED
  }

  // Send message via ESP-NOW until packet is sent successfully
  while (!packetSent) {
    esp_now_send(broadcastAddress, (uint8_t *)&myData, sizeof(myData));
    delay(100);  // Wait for the transmission to complete
  }
  Serial.println("\nGoing to Sleep");
  // Go to deep sleep mode
  esp_deep_sleep_start();
}

void loop() {
  // Empty loop as it won't be executed during deep sleep
}
```

Receiver Code

```
#include "WiFi.h"
#include "esp_camera.h"
#include "Arduino.h"
#include "soc/soc.h"           // Disable brownout problems
#include "soc/rtc_cntl_reg.h"  // Disable brownout problems
#include "driver/rtc_io.h"
#include <SPIFFS.h>
#include <FS.h>
#include <Firebase_ESP_Client.h>
//Provide the token generation process info.
#include <addons/TokenHelper.h>
#include <Time.h>

//ESP NOW libraries
#include <addons/TokenHelper.h>
#include <esp_now.h>

//Replace with your network credentials
const char* ssid = [redacted];
const char* password = [redacted];

// Insert Firebase project API Key
#define API_KEY [redacted]

// Insert Authorized Email and Corresponding Password
#define USER_EMAI[redacted]
#define USER_PASSWORD "[redacted]

// Insert Firebase storage bucket ID e.g bucket-name.appspot.com
#define STORAGE_BUCKET_ID [redacted]

// Photo File Name to save in SPIFFS
#define FILE_PHOTO "/data/photo.jpg"

const char* FILE_PATH = "/data/photo.jpg";
const char* FILE_MIME_TYPE = "image/jpeg";

```

```
// OV2640 camera module pins (CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27
#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22

bool received = false;

//Define Firebase Data objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig configF;

//Time Data Define
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 0;
const int daylightOffset_sec = 3600 * 5.5;
char timeString[20];
char mergedString[30];


//Loop Counter
unsigned long previousMillis = 0;
unsigned long interval = 60000;  // 1 minute interval
int loopCounter = 0;

```

```
// Must match the sender structure
typedef struct struct_message {
  char deviceName[20];
  bool movementDetected;
} struct_message;

// Create a struct_message called myData
struct_message myData;


// Check if photo capture was successful
bool checkPhoto(fs::FS& fs) {
  File f_pic = fs.open(FILE_PHOTO);
  unsigned int pic_sz = f_pic.size();
  return (pic_sz > 100);
}

// Capture Photo and Save it to SPIFFS
void capturePhotoSaveSpiffs(void) {
  camera_fb_t* fb = NULL;  // pointer
  bool ok = 0;             // Boolean indicating if the picture has been taken correctly
  do {
    // Take a photo with the camera
    Serial.println("Taking a photo...");

    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return;
    }
    // Photo file name
    Serial.printf("Picture file name: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);
    // Insert the data in the photo file
    if (!file) {
      Serial.println("Failed to open file in writing mode");
    } else {
```

```
    } else {
      file.write(fb->buf, fb->len);  // payload (image), payload length
      Serial.print("The picture has been saved in ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Size: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    // Close the file
    file.close();
    esp_camera_fb_return(fb);

    // check if file has been correctly saved in SPIFFS
    ok = checkPhoto(SPIFFS);
    if (ok) {
      Serial.println("Photo Captured");
    }
  } while (!ok);
}

void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
}

void initSPIFFS() {
  if (!SPIFFS.begin(true)) {
    Serial.println("An Error has occurred while mounting SPIFFS");
    ESP.restart();
  } else {
    delay(500);
    Serial.println("SPIFFS mounted successfully");
  }
}
```

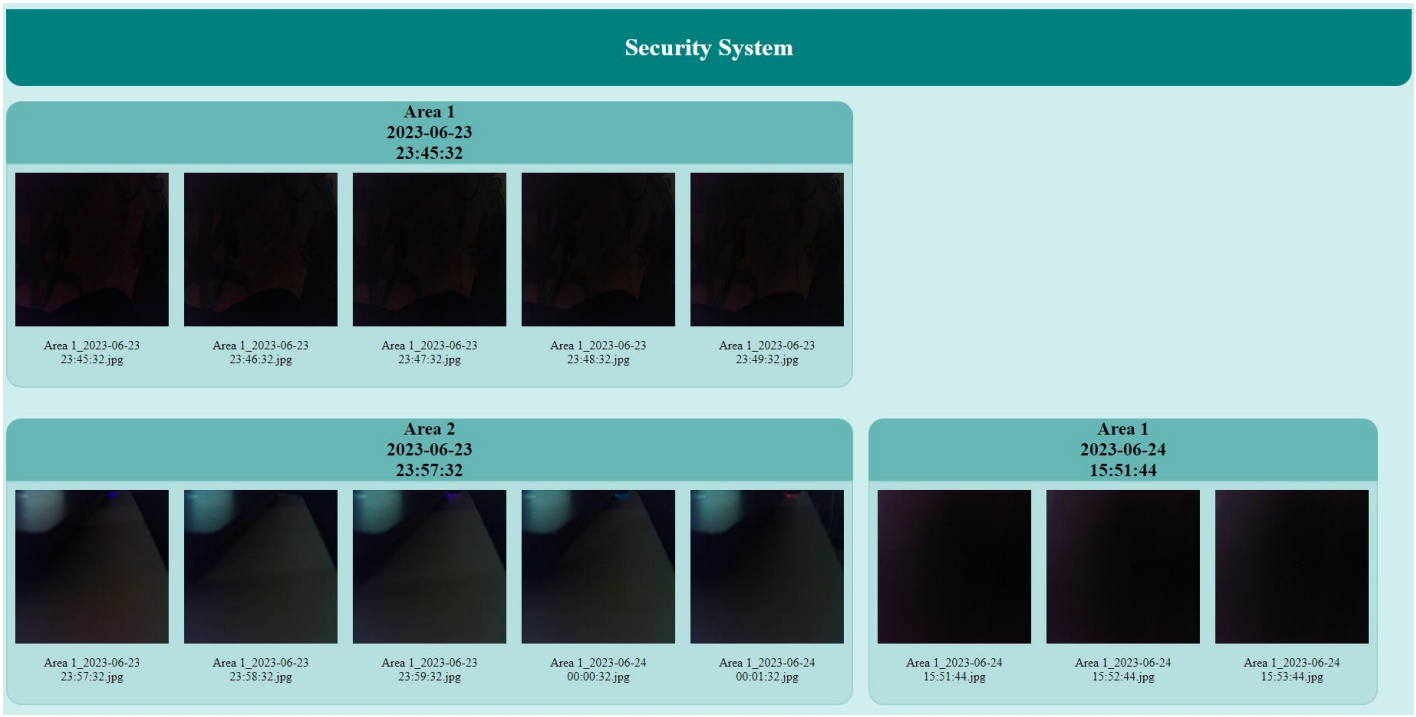
```
151 void initCamera() {
152   // OV2640 camera module
153   camera_config_t config;
154   config.ledc_channel = LEDC_CHANNEL_0;
155   config.ledc_timer = LEDC_TIMER_0;
156   config.pin_d0 = Y2_GPIO_NUM;
157   config.pin_d1 = Y3_GPIO_NUM;
158   config.pin_d2 = Y4_GPIO_NUM;
159   config.pin_d3 = Y5_GPIO_NUM;
160   config.pin_d4 = Y6_GPIO_NUM;
161   config.pin_d5 = Y7_GPIO_NUM;
162   config.pin_d6 = Y8_GPIO_NUM;
163   config.pin_d7 = Y9_GPIO_NUM;
164   config.pin_xclk = XCLK_GPIO_NUM;
165   config.pin_pclk = PCLK_GPIO_NUM;
166   config.pin_vsync = VSYNC_GPIO_NUM;
167   config.pin_href = HREF_GPIO_NUM;
168   config.pin_sscb_sda = SIOD_GPIO_NUM;
169   config.pin_sscb_scl = SIOC_GPIO_NUM;
170   config.pin_pwdn = PWDN_GPIO_NUM;
171   config.pin_reset = RESET_GPIO_NUM;
172   config.xclk_freq_hz = 20000000;
173   config.pixel_format = PIXFORMAT_JPEG;
174
175   if (psramFound()) {
176     config.frame_size = FRAMESIZE_UXGA;
177     config.jpeg_quality = 10;
178     config.fb_count = 2;
179   } else {
180     config.frame_size = FRAMESIZE_SVGA;
181     config.jpeg_quality = 12;
182     config.fb_count = 1;
183   }
184   // Camera init
185   esp_err_t err = esp_camera_init(&config);
186   if (err != ESP_OK) {
187     Serial.printf("Camera init failed with error 0x%x", err);
```

```
    ESP.restart();
  }
}

void OnDataRecv(const uint8_t* mac, const uint8_t* incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Device Name: ");
  Serial.println(myData.deviceName);
  Serial.print("Movement Detected: ");
  Serial.println(myData.movementDetected ? "Yes" : "No");


  loopCounter = 0;
  received = true;
}

void getTime() {
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;
  }
  strftime(timeString, sizeof(timeString), "%Y-%m-%d %H:%M:%S", &timeinfo);
  Serial.println(timeString);
}

void setup() {

  // Serial port for debugging purposes
  Serial.begin(115200);
  initWiFi();
  initSPIFFS();
  // Turn-off the 'brownout detector'
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  initCamera();
```

```
    // Assign the api key
    configF.api_key = API_KEY;
    //Assign the user sign in credentials
    auth.user.email = USER_EMAIL;
    auth.user.password = USER_PASSWORD;
    //Assign the callback function for the long running token generation task
    configF.token_status_callback = tokenStatusCallback;  //see addons/TokenHelper.h

    Firebase.begin(&configF, &auth);
    Firebase.reconnectWiFi(true);

    //Configure Time
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

    // Init ESP-NOW
    if (esp_now_init() != ESP_OK) {
      Serial.println("Error initializing ESP-NOW");
      return;
    }

    // Once ESPNow is successfully Init, we will register for recv CB to
    // get recv packet info
    esp_now_register_recv_cb(OnDataRecv);
  }

  void loop() {
    while (received) {


      unsigned long currentMillis = millis();

      if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;
        Serial.print("Movement detected Picture ");
        Serial.println(loopCounter + 1);
        if (loopCounter < 5) {

          capturePhotoSaveSpiffs();
          delay(100);

          if (Firebase.ready()) {

            getTime();

            strcpy(mergedString, "/data/");
            strcat(mergedString, myData.deviceName);
            strcat(mergedString, "_");
            strcat(mergedString, timeString);
            strcat(mergedString, ".jpg");

            //data/photo.jpg

            Serial.print("Uploading picture ...");

            //MIME type should be valid to avoid the download problem.
            //The file systems for flash and SD/SDMMC can be changed in FirebaseFS.h.
            if (Firebase.Storage.upload(&fbdo, STORAGE_BUCKET_ID, FILE_PHOTO, mem_storage_type_flash, mergedString, "image/jpeg")) {

              Serial.printf("\nDownload URL: %s\n", fbdo.downloadURL().c_str());


            } else {
              Serial.println(fbdo.errorReason());
            }
          }
          loopCounter++;
        }
      }
      if (loopCounter >= 5) {
        received = false;
      }
    }
  }
```

Chapter 3: Results

We have successfully developed a system that fulfills the specified requirements. When motion is detected in any of the four designated areas, an ESP32 module, which is in deep sleep mode, wakes up and activates the Red LED and buzzer for a duration of 2 seconds. Subsequently, a data packet is sent using the ESP-NOW protocol. The receiving end, which consists of the ESP OV2640 camera module, interprets the packet to determine the motion detection status and the corresponding area. If motion is detected, the module captures pictures at regular intervals of 1 minute for a duration of 5 minutes. All captured images are then uploaded to Firebase storage. Users can conveniently access these images through a web page, which provides a visual representation of the relevant area, along with the timestamp indicating when motion was detected, and a collection of images taken during the incident.



Chapter 4: Discussion

The developed system successfully addresses the requirements of motion detection and image capture in multiple designated areas. By utilizing the ESP32 module in deep sleep mode, power consumption is minimized, allowing for efficient operation over extended periods. Upon detecting motion, the system promptly activates the Red LED and buzzer, providing a visible and audible alert. The use of the ESP-NOW protocol enables seamless communication between the ESP32 module and the ESP OV2640 camera module, ensuring reliable transmission of data packets.

To enhance the system's functionality, future improvements could be considered. One aspect to focus on is refining the motion detection algorithm to increase accuracy and reduce false positives. Advanced techniques, such as background subtraction or machine learning algorithms, could be explored to achieve more precise motion detection results. Additionally, offering users the ability to configure and adjust the motion detection areas would provide greater flexibility and customization options.

Real-time alerts could be incorporated into the system to promptly notify users of detected motion. Implementing push notifications or integrating email/SMS alerts would ensure that users receive immediate notifications, enabling them to take appropriate actions in a timely manner.

Regarding image capture, a more intelligent approach can be adopted by capturing a burst of images for a short duration upon motion detection, instead of capturing images at fixed time intervals. This approach would provide a more comprehensive representation of the incident, allowing for more accurate analysis and assessment.

References

1. RandomNerdsTutorial (2023) *ESP32: ESP-NOW Encrypted Messages*. [Online] Available from < https://randomnerdtutorials.com/esp32-esp-now-encrypted-messages/> [20th June 2023]
2. RandomNerdsTutorial (2023) *ESP32-CAM Save Picture in Firebase Storage*. [Online] Available from < https://randomnerdtutorials.com/esp32-cam-save-picture-firebase-storage/.> [20th June 2023]
3. RandomNerdsTutorial (2023) *ESP32 Deep Sleep with Arduino IDE and Wake Up Sources*. [Online] Available from <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/> [20th June 2023]
4. RandomNerdsTutorial (2023) *ESP32-CAM PIR Motion Detector with Photo Capture*. [Online] Available from < https://randomnerdtutorials.com/esp32-cam-pir-motion-detector-photo-capture/> [20th June 2023]

Acknowledgements

We have put a lot of effort into this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to all of them. We are highly indebted to Mr. Asanka for his guidance and supervision. We would like to thank him for teaching us and providing the necessary information and resources for this project. Our thanks and appreciations also go to our colleagues in developing the project. Thank you to all the people who have willingly helped us out with their abilities.