`

GUJARAT TECHNOLOGIAL UNIVERSITY
(GTU)

AHEMADABAD- 382424

Vishwakarma Government Engineering College, Chandkheda-382424
(Affiliated with Gujarat Technological University, Ahmedabad)

A
Project
report
On

IoT Based Strawman's Eye

Prepared as a part of the requirement for the subject of I.O.T
B.E- Semester- **7**
(Electronics and Communication Branch)

Submitted by:

Sr. No.	Name	Enrollment No.
1	Purohit Anurag	220170111102
2	Kareliya Dhrumil	23017311014
3	Ram Axay	220170111104
4	Varu Anand	220170111142

Guided By:
K.N.Patel
Academic Yeat:
2025-26

`

Table of Contents

Table of Figures

`

1. Abstract

This project presents the design and development of a home security system using an ESP32-CAM module, PIR motion sensor, and servo motor. The system continuously moves a servo-mounted ESP32-CAM to monitor its surroundings. When motion is detected by the PIR sensor, the servo pauses, and the ESP32-CAM captures a photograph which is immediately sent to a configured email address as an alert. After a short delay, the servo resumes scanning. This IoT solution provides real-time motion-triggered surveillance suitable for smart home and low-cost security applications

2. Introduction

With the increasing demand for smart and automated home security, integrating real-time surveillance capabilities into IoT devices has become essential. This project leverages the ESP32-CAM module, a low-cost development board with built-in camera and wireless connectivity, paired with a PIR (Passive Infrared) motion sensor to detect human movement. The innovative use of a servo motor allows the camera to cover a wider area, enhancing monitoring effectiveness. Upon motion detection, the system captures images and sends them instantly to a predefined email address, providing immediate alerts and evidence. This approach not only increases the level of security but also demonstrates the versatility of ESP32-based IoT solutions for DIY makers and security enthusiasts

3. List of Components

- ESP32-CAM module (with OV2640 camera)
- PIR Motion Sensor **(**e.g., HC-SR501**)**
- Servo motor (SG90 or similar)
- FTDI USB-to-Serial adapter (for programming ESP32-CAM)
- Breadboard or PCB (for circuit connections)
- Jumper wires
- 5V DC power supply (or 2×18650 battery holder for portable use)
- Arduino Uno

`

4. Circuit Diagram

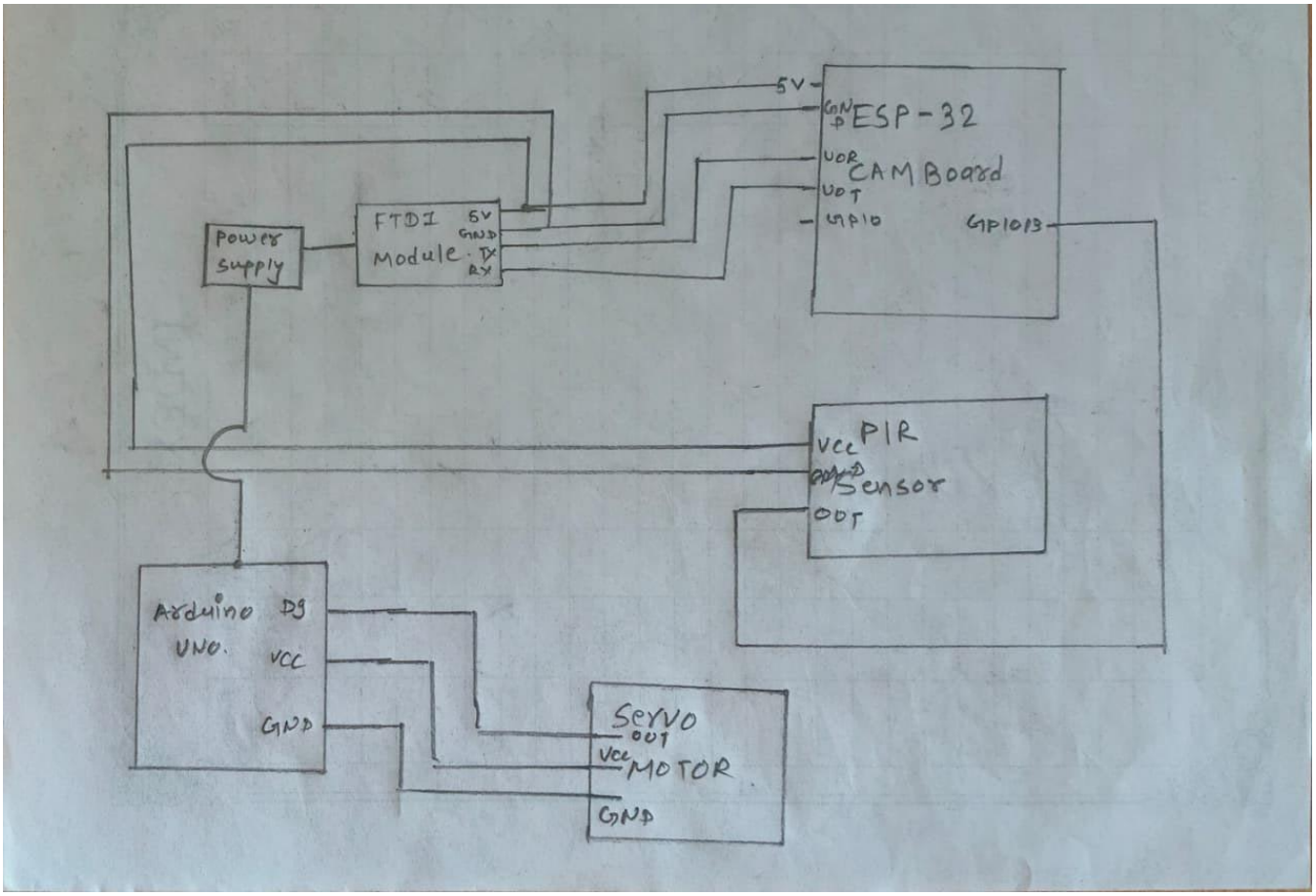

Figure 1 - Circuit Diagram

5. Working

This system is designed to enhance the coverage area of an ESP32-CAM security camera by using a servo motor controlled by an Arduino Uno. The main objective is to continuously rotate the ESP32-CAM (mounted on the servo), pause at specified angles for a defined period, and then continue the rotation, thereby scanning a wider zone for motion or surveillance.

Step-by-Step Working:

1. **System Power-Up**
   Both the Arduino Uno and ESP32-CAM are powered on. The servo motor is attached to the Arduino Uno's PWM output pin (typically pin 9).

2. **Servo Rotation Control**
   The Arduino Uno is programmed to rotate the servo in steps (for example, every 10 or 30 degrees) from 0° to 180° (or, with a special 360° servo, a full circle). At each

`

angle, the Arduino pauses for a pre-set duration (e.g., 5 seconds). During this pause, the ESP32-CAM stabilizes for clear imaging.

3. **Full Area Scanning**
   This process repeats in a loop. After reaching the maximum angle, the servo reverses direction and sweeps back. This slow and systematic rotation allows the camera to cover a wide field of view, increasing the chances of detecting motion within the surveillance area.

4. **Motion Detection (optional, with PIR on ESP32-CAM)**
   Separately, the ESP32-CAM runs its own code to monitor motion using a PIR sensor. When motion is detected, the ESP32-CAM can pause the Arduino Uno's servo via a digital signal (optional integration), capture an image, and send an email alert.

5. **Image Capture and Email Alert (on ESP32-CAM)**
   The ESP32-CAM, upon stabilization and/or motion detection, captures an image and sends an email with the photo attached to a pre-configured recipient.

6. **Resume Scanning**
   The Arduino Uno resumes the servo's rotation, enabling continuous, periodic surveillance.

`

6. Code

```
#include "esp_camera.h"
#include <ESP32Servo.h>
#include <WiFi.h>
#include <FS.h>
#include <LittleFS.h>
#include <ESP_Mail_Client.h>

//----------------------------
// MODIFY THESE SETTINGS
//----------------------------
const char* ssid = "OnePlus Nord CE4 ";
const char* password = "123321123";

#define emailSenderAccount   "220170111102@vgecg.ac.in"
#define emailSenderPassword  "jsfgossfiec"
#define smtpServer           "smtp.gmail.com"
#define smtpServerPort       465
#define emailRecipient       "anuragpu6353@gmail.com
#define emailSubject         "Motion detected - ESP32CAM Photo"

//----------------------------
// Pin Mapping
//----------------------------
#define PIR_PIN    13
#define SERVO_PIN  12
```

`

// Camera Pins (for AI-Thinker module; adapt if your module is different)

#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

#define FILE_PHOTO_PATH "/photo.jpg"

#define FILE_PHOTO "photo.jpg"

Servo myServo;

SMTPSession smtp;

//---- FUNCTION DECLARATIONS

void sendEmailWithPhoto();

void capturePhotoSaveLittleFS(void);

```
void smtpCallback(SMTP_Status status);

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  Serial.begin(115200);

  // Servo and PIR setup
  myServo.attach(SERVO_PIN);
  pinMode(PIR_PIN, INPUT);

  // WiFi setup
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);

  // Initialize SPIFFS/LittleFS
  if(!LittleFS.begin()) {
    Serial.println("LittleFS Mount Failed");
    return;
  }

  // Camera configuration
  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.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_XGA; // Adjust as needed
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Initialize camera
if (esp_camera_init(&config) != ESP_OK) {
  Serial.println("Camera init failed");
```

```
    return;
  }

  Serial.println("ESP32CAM Ready, waiting for motion...");
}

void loop() {
  if (digitalRead(PIR_PIN) == HIGH) {
    // Optional sweep: move servo 0-180-0, photo at the most central position
    myServo.write(0);
    delay(400);
    myServo.write(90);
    delay(400);
    capturePhotoSaveLittleFS();
    sendEmailWithPhoto();
    myServo.write(180);
    delay(400);
    myServo.write(0); // Return
    delay(5000); // Prevent continuous trigger
  }
  delay(100);
}

void capturePhotoSaveLittleFS(void) {
  camera_fb_t * fb = NULL;

  // Discard first 2-3 frames
  for (int i=0; i<3; i++) {
```

```
  fb = esp_camera_fb_get();
  esp_camera_fb_return(fb);
 }

 fb = esp_camera_fb_get();
 if(!fb) {
  Serial.println("Camera capture failed");
  ESP.restart();
  return;
 }
 File file = LittleFS.open(FILE_PHOTO_PATH, FILE_WRITE);
 if(!file) {
  Serial.println("Failed to open file for writing");
  esp_camera_fb_return(fb);
  return;
 }
 file.write(fb->buf, fb->len);
 file.close();
 esp_camera_fb_return(fb);
 Serial.println("Photo captured and saved");
}

void sendEmailWithPhoto() {
 smtp.debug(1);
 smtp.callback(smtpCallback);

 ESP_Mail_Session session;
 session.server.host_name = smtpServer;
```

`

```
session.server.port = smtpServerPort;
session.login.email = emailSenderAccount;
session.login.password = emailSenderPassword;
session.login.user_domain = "";

SMTP_Message message;
message.sender.name = "ESP32CAM";
message.sender.email = emailSenderAccount;
message.subject = emailSubject;
message.addRecipient("User", emailRecipient);

String htmlMsg = "<h2>Motion detected! Photo attached.</h2>";
message.html.content = htmlMsg.c_str();

SMTP_Attachment att;
att.descr.filename = FILE_PHOTO;
att.descr.mime = "image/jpg";
att.file.path = FILE_PHOTO_PATH;
att.file.storage_type = esp_mail_file_storage_type_flash;
att.descr.transfer_encoding = Content_Transfer_Encoding::enc_base64;

message.addAttachment(att);

if (!smtp.connect(&session)) return;

if (!MailClient.sendMail(&smtp, &message, true)) {
  Serial.println("Email Send Failed: " + smtp.errorReason());
}
```

```
  message.clear();
}

// Print status of SMTP process
void smtpCallback(SMTP_Status status) {
  Serial.println(status.info());
}
```

7. Conclusion

In conclusion, this project successfully designed and implemented an intelligent IoT-based surveillance system using the ESP32-CAM module integrated with a PIR motion sensor and continuous rotation servo motor. When motion is detected, the servo pauses its continuous rotation to position the camera, capture an image, and send it via email to the user, enabling prompt remote monitoring. The solution leverages easy-to-use open-source tools and affordable hardware components, making it cost-effective and accessible.

The system demonstrates reliable real-time motion detection and photo capture, while the continuous servo rotation allows for broad area coverage. Incorporating NTP time synchronization ensures secure email SMTP communication. Although straightforward, the project lays a strong foundation for future enhancements such as more sophisticated motion filtering, cloud storage, video streaming, and integration with home automation platforms.

Overall, this IoT project showcases how combining ESP32-CAM capabilities, efficient sensor integration, and smart communication protocols can greatly enhance home security and remote surveillance solutions in a scalable, wireless, and economical manner.