SEM 1 25/26

SECTION 1

COURSE CODE: MCTA 3202

LECTURER NAME:
ZULKIFLI BIN ZAINAL ABIDIN

WEEK 5: SMART SURVEILLANCE SYSTEM WITH MOTORIZED CAMERA BASE (VER 1.0)

PREPARED BY:
GROUP 1

NAME	MATRIC NUMBER
AMIR MUIZZUDDIN BIN NORAIDIL RAZMAN	2129579
MUHAMMAD HAIKAL HANIF BIN ABDUL RAZAK	2213297
AHMAD ILHAM BIN ABDUL AZIZ	2112109

INTRODUCTION

This lab introduces the integration of IoT and actuator systems through the ESP32-CAM module, enabling students to develop a simple smart surveillance system. The system is capable of live video streaming over Wi-Fi and uses a servo motor to enable horizontal panning, simulating basic motion tracking. The system can also be extended with manual input (e.g., button) or a passive infrared (PIR) sensor for motion detection. The objectives for this experiment is to able to know interface and stream live video using ESP32-CAM module, control a servo motor for horizontal camera panning, build and test a basic IoT-based surveillance prototype and finally understand the integration of microcontroller,sensors,actuators and wireless communications in mechatronic systems.

MATERIALS

1. ESP32-CAM module.
2. Servo Motor
3. Push Button
4. 5V Power Supply
5. Jumper Wires

EXPERIMENT SETUP

This experiment involves the integration of the ESP32-CAM module, a servo motor for panning, and a manual push button to simulate motion control. A stable Wi-Fi network is required for live video streaming.

3.2.1 ESP32-CAM Programming Setup

To upload code to the ESP32-CAM, the following wiring is used:

ESP32-CAM Pin	FTDI / CH340 Pin
U0T	RX
U0R	TX
5V	5V
GND	GND

IO0 → GND	Only during flashing

Important:

The IO0 pin must be connected to GND during code upload to enter flash mode.

The Arduino IDE is configured with:

- Board: **ESP32 Driver Module**
- Partition Scheme: **Huge App (3MB No OTA)**
- Upload Speed: **115200 baud**

3.2.2 Servo Motor Wiring

The servo motor provides horizontal panning controlled via PWM.
Connections are as follows:

Servo Wire	Connect To
Orange (Signal)	**GPIO 2** (PWM output)
Red (VCC)	**5V external**
Brown (GND)	**GND** (shared with ESP32-CAM)

Note:
A 5V 2A external supply is recommended to prevent ESP32-CAM restarts.

3.2.3 Push Button Integration (Task Requirement)

The push button is added to allow manual activation of servo movement.

Button Pin	Connection

One leg	GPIO 13
Other leg	GND

Software uses **internal pull-up resistor**, meaning:

- Button **pressed = LOW**
- Button **released = HIGH**

3.2.4 Wi-Fi Streaming Setup

After programming, the ESP32-CAM:

1. Connects to a Wi-Fi network.
2. Prints its IP address on Serial Monitor.
3. Hosts two pages:
    - `/` → Main webpage with embedded video stream.
    - `/stream` → MJPEG video stream.

Users view the live feed by entering the IP address in any browser on the same network.

3.2.5 Mechanical Setup

- The ESP32-CAM is mounted on a servo bracket.
- The servo rotates 0–180° horizontally.
- All components are fixed on a stable base to prevent vibration.
- Cables are arranged to avoid obstructing the camera's view or servo movement.

METHODOLOGY

4.1 Hardware Setup

1. Connect ESP32-CAM to the FTDI programmer:
    - FTDI TX → ESP32-CAM RX
    - FTDI RX → ESP32-CAM TX
    - FTDI 5V → ESP32-CAM 5V
    - FTDI GND → ESP32-CAM GND
    - IO0 → GND (only for uploading the program)
2. Connect the SG90 servo motor:
    - Orange (signal) → GPIO2
    - Red (VCC) → External 5 V power
    - Brown (GND) → Common GND (same as ESP32)
3. Connect the pushbutton:
    - One side → GPIO13
    - Other side → GND
    - INPUT_PULLUP is used in software
4. After uploading, remove IO0 → GND and reset the ESP32-CAM.

4.2 Software Setup

1. Install ESP32 board package in Arduino IDE via:
   `https://dl.espressif.com/dl/package_esp32_index.json`
2. Select board:
   Tools → Board → AI Thinker ESP32-CAM

3. Set partition scheme:
   Huge APP (3MB No OTA)

4. Upload the modified CameraWebServer code, which includes:

    - Servo control via LEDC PWM
    - Pushbutton activation logic
    - HTTP video streaming
    - Optional face detection routines

4.3 Workflow

1. Power the ESP32-CAM and connect to the same Wi-Fi network.
2. Open the Serial Monitor to read the assigned IP address.
3. Enter the IP in a browser to access the live camera feed.
4. Press the pushbutton to trigger servo sweeping.
5. Observe servo movement on the live stream.
6. (Optional) Enable face detection and verify that faces are highlighted with bounding boxes.

4.3 Full Coding

```
#include "esp_camera.h"

#include <WiFi.h>

#include "esp_http_server.h"

#include <ESP32Servo.h>


const char* ssid = "zzz";

const char* password = "12345627";
```

```
#define SERVO_PIN 13

#define BUTTON_PIN 12

Servo servo;

int angle = 90;

bool forward = true;

bool moveServo = false;

bool lastButtonState = HIGH;

bool buttonState = HIGH;

bool calmDetected = false;


// 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


// Variables for motion detection
uint8_t prevBrightness = 0;
int stableFrameCount = 0;
const int STABLE_THRESHOLD = 3;
const int BRIGHTNESS_CHANGE_THRESHOLD = 6;
httpd_handle_t stream_httpd = NULL;
// HTML page
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
  <title>ESP32-CAM Stream</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { font-family: Arial; text-align: center; margin: 0; }
    .container { padding: 20px; }
    img { width: 100%; max-width: 640px; }
    .status { margin: 20px; font-size: 18px; }
    button { padding: 15px 30px; font-size: 18px; margin: 10px; cursor: pointer; }
  </style>
</head>
<body>
  <div class="container">
    <h1>ESP32-CAM Control</h1>
```

```
    <img src="/stream" />
    <div class="status">
      <p>Servo Status: <span id="servo">OFF</span></p>
      <p>Calm Detected: <span id="calm">NO</span></p>
    </div>
    <button onclick="toggleServo()">Toggle Servo</button>
  </div>
  <script>
    function toggleServo() {
      fetch('/toggle').then(response => response.text()).then(data => {
        document.getElementById('servo').innerText = data;
      });
    }
    setInterval(() => {
      fetch('/status').then(response => response.json()).then(data => {
        document.getElementById('servo').innerText = data.servo;
        document.getElementById('calm').innerText = data.calm;
      });
    }, 1000);
  </script>
</body>
</html>
)rawliteral";

// Stream handler
static esp_err_t stream_handler(httpd_req_t *req) {
```

```
camera_fb_t * fb = NULL;
esp_err_t res = ESP_OK;
size_t _jpg_buf_len = 0;
uint8_t * _jpg_buf = NULL;
char * part_buf[64];

res = httpd_resp_set_type(req, "multipart/x-mixed-replace; boundary=frame");
if(res != ESP_OK) return res;

while(true) {
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if(fb->format != PIXFORMAT_JPEG) {
      bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
      esp_camera_fb_return(fb);
      fb = NULL;
      if(!jpeg_converted) {
        Serial.println("JPEG compression failed");
        res = ESP_FAIL;
      }
    } else {
      _jpg_buf_len = fb->len;
      _jpg_buf = fb->buf;
```

```
      }
    }
    if(res == ESP_OK) {
      size_t hlen = snprintf((char *)part_buf, 64, "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n",
_jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK) {
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK) {
      res = httpd_resp_send_chunk(req, "\r\n--frame\r\n", 14);
    }
    if(fb) {
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf) {
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK) break;
  }
  return res;
}
```

```
// Index page handler
static esp_err_t index_handler(httpd_req_t *req) {
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}

// Toggle servo handler
static esp_err_t toggle_handler(httpd_req_t *req) {
  moveServo = !moveServo;
  Serial.print("Web toggle - Servo: ");
  Serial.println(moveServo ? "ON" : "OFF");
  httpd_resp_set_type(req, "text/plain");
  return httpd_resp_sendstr(req, moveServo ? "ON" : "OFF");
}

// Status handler
static esp_err_t status_handler(httpd_req_t *req) {
  char json[100];
  snprintf(json, sizeof(json), "{\"servo\":\"%s\",\"calm\":\"%s\"}",
           moveServo ? "ON" : "OFF",
           calmDetected ? "YES" : "NO");
  httpd_resp_set_type(req, "application/json");
  return httpd_resp_sendstr(req, json);
}

void startCameraServer() {
```

```
httpd_config_t config = HTTPD_DEFAULT_CONFIG();
config.server_port = 80;

httpd_uri_t index_uri = {
  .uri = "/",
  .method = HTTP_GET,
  .handler = index_handler,
  .user_ctx = NULL
};

httpd_uri_t stream_uri = {
  .uri = "/stream",
  .method = HTTP_GET,
  .handler = stream_handler,
  .user_ctx = NULL
};

httpd_uri_t toggle_uri = {
  .uri = "/toggle",
  .method = HTTP_GET,
  .handler = toggle_handler,
  .user_ctx = NULL
};

httpd_uri_t status_uri = {
  .uri = "/status",
```

```
    .method = HTTP_GET,
    .handler = status_handler,
    .user_ctx = NULL
  };

  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
    httpd_register_uri_handler(stream_httpd, &stream_uri);
    httpd_register_uri_handler(stream_httpd, &toggle_uri);
    httpd_register_uri_handler(stream_httpd, &status_uri);
    Serial.println("Camera server started");
  }
}

void setup() {
  Serial.begin(115200);
  Serial.println("ESP32-CAM Calm Detection + Button Control");

  pinMode(BUTTON_PIN, INPUT_PULLUP);
  servo.attach(SERVO_PIN);
  servo.write(angle);

  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;  // Changed to JPEG for streaming
config.frame_size = FRAMESIZE_VGA;     // Better quality
config.jpeg_quality = 10;
config.fb_count = 2;

if (esp_camera_init(&config) != ESP_OK) {
  Serial.println("Camera init failed!");
  while (true);
}
```

```
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");
  Serial.print("Camera Stream URL: http://");
  Serial.println(WiFi.localIP());

  startCameraServer();
}

void loop() {
  // Button control
  buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == LOW && lastButtonState == HIGH) {
    delay(50);
    buttonState = digitalRead(BUTTON_PIN);
    if (buttonState == LOW) {
      moveServo = !moveServo;
      Serial.print("Button pressed - Servo: ");
      Serial.println(moveServo ? "ON" : "OFF");
    }
  }
  lastButtonState = buttonState;
```

```
// Camera calm detection (periodic check, not every loop)
static unsigned long lastCheck = 0;
if (millis() - lastCheck > 200) {  // Check every 200ms
  lastCheck = millis();

  camera_fb_t* fb = esp_camera_fb_get();
  if (fb) {
    uint32_t total = 0;
    for (int i = 0; i < fb->len; i += 10) {  // Sample every 10th pixel
      total += fb->buf[i];
    }
    uint8_t avgBrightness = total / (fb->len / 10);
    esp_camera_fb_return(fb);

    int brightnessChange = abs(avgBrightness - prevBrightness);

    if (brightnessChange < BRIGHTNESS_CHANGE_THRESHOLD) {
      stableFrameCount++;
      if (stableFrameCount >= STABLE_THRESHOLD && !calmDetected) {
        calmDetected = true;
        Serial.println("CALM detected - Servo STOPPED");
      }
    } else {
      stableFrameCount = 0;
      if (calmDetected) {
        calmDetected = false;
```

```
        Serial.println("Movement detected");
      }
    }
    prevBrightness = avgBrightness;
  }
 }
 // Servo control
 if (moveServo && !calmDetected) {
  if (forward) angle++;
  else angle--;
  if (angle >= 180) forward = false;
  if (angle <= 0) forward = true;
  servo.write(angle);
  delay(20);
 } else {
  delay(20);
 }
}
```

CONCLUSIONS

In this experiment, we successfully learned how to set up and use the ESP-32 CAM module for live streaming video and how to control a servo motor for basic camera panning. By combining microcontroller, Wi-Fi connection and servo control, we were able to build a simple IoT-based surveillance system. This lab also showed how sensors, actuators and wireless communication can work together in mechatronics applications. Overall, the activity helped us understand the basic idea behind smart surveillance systems and how they can be expanded with tools like buttons or PUR sensors for motion detection.