Mechatronics System Integration (MCTA3203)

Week **6**: Smart Surveillance System with Motorized Camera Base (ver 1.0)

Smart Surveillance System Using ESP32-CAM

Overview

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. An example of this system is shown in **Fig. 1**.

This practical exercise demonstrates key mechatronic concepts including wireless data transmission, PWM control, and sensor integration.

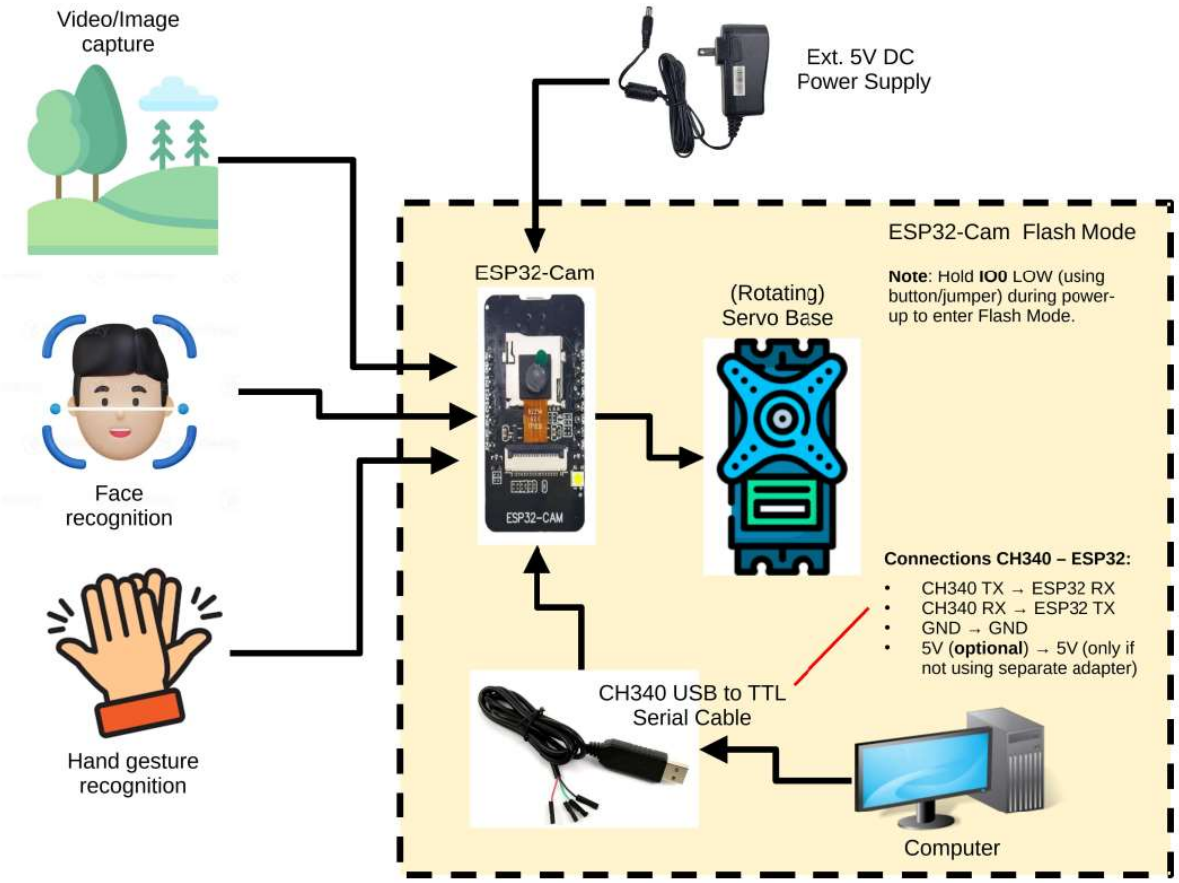

Fig. 1 System Overview

Objectives

By the end of this lab, students will be able to:

1. Interface and stream live video using the ESP32-CAM module.
2. Control a servo motor for horizontal camera panning.
3. Build and test a basic IoT-based surveillance prototype.

4. Understand the integration of microcontroller, sensors, actuators, and wireless communication in mechatronic systems.

Required Hardware

Component	Quantity	Notes
ESP32-CAM Module	1	Ensure it uses **OV2640** or **OV2460** camera
USB-to-Serial Adapter (FTDI) or CH340 USB to TTL Serial Cable	1	Required for programming ESP32-CAM
SG90 or MG90S Servo Motor	1	Used to pan the camera
Pushbutton	1	For manual motion control
5V Power Supply (2A)	1	Strongly recommended for stable operation
Jumper Wires	Several	For all connections
Breadboard	1	Optional for prototyping
Mounting Bracket or Servo Base	1	For physical camera panning
Capacitor (100µF or higher)	Optional	To stabilize power to servo

Circuit Diagram Summary

1a. ESP32-CAM to FTDI (for programming)

ESP32-CAM Pin	FTDI Pin
U0T	RX
U0R	TX
5V	5V
GND	GND
IO0	GND (Only during upload)

⚠ **IO0** must be connected to GND during code upload/flash.

1b. ESP32-CAM to CH340 USB to TTL Serial Cable (for programming)

CH340 Cable Wire	ESP32-CAM Pin
Red (VCC)	5V
Black (GND)	GND
White (TXD)	U0R (GPIO3)
Green (RXD)	U0T (GPIO1)

2. Servo Motor to ESP32-CAM

Servo Wire	Connect To
Orange (Signal)	GPIO 2
Red (VCC)	5V
Brown (GND)	GND

Software Setup (Pre-Lab Task)

1. Install Arduino IDE
2. Install ESP32 Board Package via Board Manager
3. Select Board:
    - Board: `ESP32 Wrover Module`
    - Partition Scheme: `Huge App (3MB No OTA)`
    - Upload Speed: `115200`
4. Connect FTDI (or CH340 USB to TTL Serial Cable) to ESP32-CAM
5. Hold **IO0** to **GND**, then power the board (flash mode)

Code/Script Overview

- Initializes the ESP32-CAM
- Sets up video streaming over local Wi-Fi
- Controls a servo motor connected to GPIO 2
- Creates a basic web interface to stream the video
- Enables continuous panning via servo (can be modified for sensor-triggered motion)
  See full source code in *Appendix A*.

Testing and Demonstration

1. Upload the code using FTDI or CH340 USB to TTL Serial Cable
2. Open Serial Monitor at 115200 baud
3. Find the IP address printed by the ESP32-CAM
4. Enter the IP in a browser on the same Wi-Fi network
5. Observe the live video feed
6. Watch the servo panning back and forth
7. Press the button to simulate motion-based control (requires code extension)

Lab Exercises

Part A: Basic Camera Streaming

- Upload code and verify live video stream
- Identify and understand pin configuration

Part B: Servo Motion

- Observe and explain how servo PWM control is implemented
- Modify min/max positions for different rotation angles

Part C: Manual Motion Trigger

- Add a button
- Modify code to pan only when button is pressed

Assessment Criteria

Task	Marks
ESP32-CAM setup and streaming	20%
Servo integration and control	20%
Optional button integration	10%

Task	Marks
Working prototype demo	30%
Lab report (circuit + code + reflection)	20%

Safety Precautions

- Do not power the servo from USB alone — use a 5V 2A source
- Avoid short circuits on the breadboard
- Disconnect **IO0** from GND after uploading/flashing code

Task

1. Integrate a pushbutton for manual control of servo panning. [Mandatory]
   - Modify the existing code (which pans continuously) so that the servo only moves when the button is pressed.
   - Connect one leg of the pushbutton to a GPIO pin (e.g., GPIO 13) and the other to GND.
   - Use an internal pull-up resistor in software or an external resistor if needed.

   Pushbutton Connections

Pushbutton Pin	Connect To
One leg	GPIO (e.g., 13)
Other leg	GND

   Use pull-up or pull-down resistor if necessary.

2. Use ESP32-CAM face detection capabilities. [Intermediate]
   - Modify code to enable built-in face detection in ESP32-CAM.
   - Optionally, trigger servo panning or image capture when a face is detected.

3. Add vertical panning with a second servo motor. [Optional / Bonus]
   - Connect an additional servo to another GPIO (e.g., GPIO 14).
   - Implement control logic for up/down movement using manual input or face tracking.

References

[1] https://randomnerdtutorials.com/program-upload-code-esp32-cam/
How to Program / Upload Code to ESP32-CAM AI-Thinker (Arduino IDE)

[2] https://my.cytron.io/p-ch340-usb-to-ttl-serial-cable
CH340 USB to TTL Serial Cable

[3] https://arduino-er.blogspot.com/2020/09/program-esp32-cam-using-ftdi-adapter.html
Program ESP32-CAM using FTDI adapter

[4] https://www.youtube.com/watch?v=D3MPBPGT3cw
Program ESP32-CAM using FTDI adapter

[5] https://core-electronics.com.au/guides/esp32-cam-set-up/
Use a ESP32-CAM Module to Stream HD Video Over Local Network

Appendix A

```
/*
   Display video stream from an ESP32-CAM on a web page,
   with horizontal panning motion controlled by a servo motor.

   For more details:
   https://electroniqueamateur.blogspot.com/2020/02/mouvement-panoramique-
avec-esp32-cam-et.html
*/

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

// WiFi network credentials
const char* ssid = "**********";
const char* password = "***********";

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-
replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-
Length: %u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

int stream_port_number; // Stream server port number

const int servo_pin = 2; // Servo motor connected to GPIO 2

// PWM values for servo's min and max positions
const int servo_min_position = 3700;
const int servo_max_position = 7000;
const int servo_speed = 10;

int servo_position = servo_min_position;
bool servo_direction = true; // true = forward, false = backward

// ****************************************************
// stream_handler: handles video streaming

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];

   static int64_t last_frame = 0;
   if (!last_frame) {
      last_frame = esp_timer_get_time();
   }

   res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
   if (res != ESP_OK) {
      return res;
   }
```

```
   httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

   while (true) {
      fb = esp_camera_fb_get();
      if (!fb) {
         Serial.println("Camera capture failed");
         res = ESP_FAIL;
      } else {
         if (fb->width > 400) {
            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, _STREAM_PART,
_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, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
      }
      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;
      }

      // --------- Servo motor movement handling -----------
      ledcWrite(servo_pin, servo_position);

      if (servo_direction) {
         servo_position += servo_speed;
         if (servo_position > servo_max_position) {
            servo_direction = false;
         }
      } else {
         servo_position -= servo_speed;
         if (servo_position < servo_min_position) {
            servo_direction = true;
```

```
        }
      }
   }
   last_frame = 0;

   return res;
}

// ***************************************************
// web_handler: builds the web page

static esp_err_t web_handler(httpd_req_t *req) {
   httpd_resp_set_type(req, "text/html");
   httpd_resp_set_hdr(req, "Content-Encoding", "identity");
   sensor_t * s = esp_camera_sensor_get();

   char webPage[175] = "";
   strcat(webPage, "<!doctype html> <html> <head> <title id='title'>ESP32-
CAM</title> </head> <body> <img id='stream' src='http://");

   // Stream server address (example: 192.168.0.145:81)
   char address[20] = "";
   sprintf(address, "%d.%d.%d.%d:%d", WiFi.localIP()[0], WiFi.localIP()[1],
WiFi.localIP()[2], WiFi.localIP()[3], stream_port_number);
   strcat(webPage, address);
   strcat(webPage, "/stream'> </body> </html>");
   int pageLength = strlen(webPage);

   return httpd_resp_send(req, (const char *)webPage, pageLength);
}

// ***************************************************
// startCameraServer: starts the web and stream servers

void startCameraServer() {
   httpd_config_t config = HTTPD_DEFAULT_CONFIG();

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

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

   Serial.printf("Starting web server on port: '%d'\n",
config.server_port);
   if (httpd_start(&camera_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(camera_httpd, &index_uri);
   }

   config.server_port += 1;
   config.ctrl_port += 1;
   Serial.printf("Starting stream server on port: '%d'\n",
config.server_port);
```

```
   if (httpd_start(&stream_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(stream_httpd, &stream_uri);
   }

   stream_port_number = config.server_port;
}

// ******************************************************
// setup: camera initialization and WiFi connection

void setup() {
   Serial.begin(115200);
   Serial.println();
   Serial.println("====");

   // Pin configuration for AI Thinker ESP32-CAM
   camera_config_t config;
   config.ledc_channel = LEDC_CHANNEL_0;
   config.ledc_timer = LEDC_TIMER_0;
   config.pin_d0 = 5;
   config.pin_d1 = 18;
   config.pin_d2 = 19;
   config.pin_d3 = 21;
   config.pin_d4 = 36;
   config.pin_d5 = 39;
   config.pin_d6 = 34;
   config.pin_d7 = 35;
   config.pin_xclk = 0;
   config.pin_pclk = 22;
   config.pin_vsync = 25;
   config.pin_href = 23;
   config.pin_sscb_sda = 26;
   config.pin_sscb_scl = 27;
   config.pin_pwdn = 32;
   config.pin_reset = -1;

   config.xclk_freq_hz = 20000000;
   config.pixel_format = PIXFORMAT_JPEG;
   config.frame_size = FRAMESIZE_VGA;
   config.jpeg_quality = 10;
   config.fb_count = 2;

   // Camera initialization
   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();

   // Servo motor initialization
   ledcSetup(2, 50, 16); // channel, frequency, resolution
   ledcAttachPin(servo_pin, 2); // pin, channel

   Serial.println("");
   Serial.print("Connecting to WiFi network: ");
   Serial.println(ssid);
   delay(100);

   WiFi.begin(ssid, password);
```

```
   while (WiFi.status() != WL_CONNECTED) {
      delay(250);
   }

   Serial.println("WiFi connected");
   Serial.println("");

   delay(100);

   startCameraServer();

   Serial.print("Camera ready. Access it at: http://");
   Serial.print(WiFi.localIP());
   Serial.println("");
}

// ***************************************************
// loop does nothing!

void loop() {
   delay(10000);
}
```