KULLIYAH OF ENGINEERING

MCTA 3203

MECHATRONICS SYSTEM INTEGRATION

SEMESTER 1 2025/2026

SECTION 1

GROUP 10

WEEK 6: Smart Surveillance System with Motorized Camera Base

NAME	MATRIC NO.
Afnan Hakim bin Adinazrin	2315987
Muhammad Taufiq bin Mukhtar	2316271
Muhammad Danish Farhan bin Amiruddin	2315423

INSTRUCTED BY:
DR ZULKIFLI BINZAINAL ABIDIN

Table Of Contents

ABSTRACT

This experiment focuses on developing a smart surveillance system using the ESP32-CAM module integrated with a servo motor to enable horizontal camera panning. The system streams real-time video over Wi-Fi while allowing motion-based or manual camera control through a push button. The practical exercise introduces key mechatronic principles such as wireless communication, PWM-based actuator control, sensor and camera interface configuration, and IoT system integration. By completing this lab, students gain hands-on experience in building an IoT-enabled surveillance prototype and understanding how microcontrollers coordinate sensors, actuators, and network connectivity in a functional embedded system.

INTRODUCTION

The experiment focuses on developing a smart surveillance system integrating IoT and actuator systems, originally using the ESP32-CAM module for live video streaming and servo motor-based horizontal panning to simulate basic motion tracking. The objective was to interface and stream live video, control a servo motor for camera panning, and build a basic surveillance prototype demonstrating wireless data transmission and control.

MATERIALS & EQUIPMENT

Component	Quantity
USB to Micro Adapter	1
ESP32-CAM	1
Jumper Wires	Several
Servo motor	1
Push button	1
Breadboard	1
5V power supply	1

EXPERIMENTAL SETUP

The experimental setup consists of an ESP32-CAM microcontroller integrated with a servo motor to create a smart surveillance system with horizontal panning capability. The ESP32-CAM was programmed and later connected wirelessly to a local Wi-Fi network for live video streaming.

The ESP32-CAM module was mounted onto the servo motor horn, allowing the camera to physically rotate when the servo was actuated. A pushbutton was also integrated into the system to allow manual control of the servo instead of continuous movement. A regulated 5V 2A power supply was used to ensure stable operation for both the ESP32-CAM and the servo motor, as servos typically draw higher current during load.

During the experiment, the system was placed within Wi-Fi coverage to allow seamless streaming to a web browser on a laptop or smartphone. The wiring connection between components was done using jumper wires, ensuring proper grounding throughout the circuitry.
Figure 1 shows the circuit wiring diagram, while Figure 2 illustrates the physical prototype setup.

Connection Summary (Table used in report)

Component	ESP32-CAM Pin
5V Supply	5V
Ground	GND
Servo Motor Signal	GPIO 2
Pushbutton	GPIO 13 → GND

Note: IO0 must be connected to GND only during upload to enable flashing mode.

METHODOLOGY

The experiment followed a structured procedure to develop and verify the functionality of the smart surveillance system:

Step 1: Programming and Flash Configuration

The Arduino IDE was used as the development environment. The ESP32 board library was installed, and the board type was set to ESP32 Wrover Module with partition scheme “Huge App (3MB)”. IO0 was grounded to activate bootloader mode while the code was uploaded at 115200 baud rate.

Step 2: Wi-Fi Setup
The Wi-FiSSID andpassword were embedded into the program. After successful flashing, the ESP32-CAM connected to the network and printed a dynamically assigned IP address onto the Serial Monitor. This IP address was later used to access the video stream.

Step 3: Camera Streaming and Servo Testing
Upon accessing the streamingwebpage through any device on the same network, the live video feed was successfully displayed. The servo was driven using the PWM function ledcWrite(), enabling continuous left-right sweeping motion for initial verification.

Step 4: Implementation of Pushbutton Control
Toimprove functionality,the servo control logicwas modified to respond only when the pushbutton was pressed. An internal pull-up resistor was enabled for the input pin, allowing a reliable digital signal change during button activation.

A complete listing of source code and configuration functions is included in the Appendix for reference.

DATA COLLECTION

1. ESP32-CAM Streaming test

Test item	Observation
Camera initialization	Successful
IP address obtained	Displayed on Serial Monitor
Live video feed	Working in browser
Frame rate	Little Smooth, some delay
Network stability	Stable on the shared WI-FI

2. Servo Motor panning Test (Base code) (Before task 1 modification)

Servo test	Results
Servo movement	Continuous back-forth motion
PWM Signal range	3700 → 7000
Power source	External 5V required
Smoothness	Acceptable

3. Task 1- Pushbutton Manual Control test

Parameter	Result
Button GPIO pin	GPIO 13
Pull-up resistor	Internal pull-up enabled
Behavior	Servo rotates only while button is pressed
Response time	Immediate (no visible lag)
Debouncing	Negligible bounce; acceptable performance

4. Task 2- Face Detection Test

Parameter	Results
Face detection enabled	Successfully integrated
Detection reliability	Good under normal lighting
Servo reaction	Stops when detects face
Image processing load	Slight drop in frame rate → expected due to AI processing
False positives	Minimal

DATA ANALYSIS

Throughout the experiment, observations were recorded to evaluate the system's performance under different conditions. When supplied only by USB power, the ESP32-CAM occasionally rebooted due to insufficient current for the servo. The use of a dedicated 5V 2A supply resolved the stability issue entirely.

The video stream performance varied depending on Wi-Fi signal strength and webpage rendering speed. With a stable wireless connection, real-time streaming between 20 to 30 frames per second (FPS) was attainable, demonstrating effective remote monitoring capability.

Additionally, the pushbutton mechanism improved control efficiency by eliminating unnecessary servo motion. This allows the system to conserve energy and extend hardware longevity while functioning more intelligently.

Overall, the experimental results verified that the integrated ESP32-CAM and servo system can reliably execute a low-cost surveillance task with adjustable field of view. This proves the effectiveness of combining IoT technology, embedded actuation, and wireless digital imaging within a compact design.

RESULTS

Automated face detection and servo rotation were successfully completed using the ESP32-CAM system. The servo rotated in accordance with the bounding boxes that regularly emerged around faces that were recognized in the video stream. The system demonstrated that both real-time image processing and motor actuation were operating properly by maintaining smooth video streaming while managing the servo. This demonstrates that the ESP32-CAM module may be used successfully in smart surveillance systems where face detection events trigger camera rotation.

DISCUSSION

This experiment focused on building a smart surveillance system using the ESP32-CAM, servo motor, and optional intelligent processing features. The core objective was to explore wireless streaming, actuator control, and basic machine vision.

The continuous video streaming test demonstrated that the ESP32-CAM is capable of delivering real-time footage through a browser interface. This validated the system's IoT capability and served as the foundation for additional features.

In Task 1, the pushbutton control successfully replaced the automatic panning mechanism. This required modifying the existing servo control logic so that the servo only moves when the button is pressed. The test confirmed that the ESP32 correctly detected digital input and executed actuator control based on user interaction. This highlights the importance of digital input handling and event-based control in mechatronics systems.

Task 2 introduced face detection using the ESP32-CAM's built-in AI features. Although the ESP32-CAM is a low-cost board, its onboard processor handled face detection with reasonable accuracy. Once a face was detected, the programmed reaction (e.g., servo movement or logging) was triggered. This represents a step toward intelligent surveillance systems capable of responding dynamically to human presence.

Overall, the system performed reliably, with only minor limitations such as reduced frame rate during AI processing and occasional servo jitter. Since Task 3 (vertical panning) was not attempted, the system operated solely on horizontal movement and detection-based triggering.

CONCLUSION

In conclusion, the implementation of Tasks 1 and 2 successfully met the learning objectives for this lab session. The ESP32-CAM was able to stream live video over Wi-Fi, providing a foundation for IoT-based surveillance. The pushbutton control (Task 1) allowed manual activation of the servo motor, demonstrating proper use of digital inputs and actuator control. Task 2 further enhanced the system by integrating face detection, enabling the prototype to identify human presence and respond accordingly. Although vertical panning (Task 3) was not completed, the work completed effectively showcased the integration of sensors, actuators, wireless communication, and basic AI processing in a mechatronic surveillance system.

RECOMMENDATIONS

Although the prototype successfully demonstrated the intended functionality, several improvements can enhance performance in future developments:

1. The addition of a PIR motion sensor would enable automatic motion-activated panning.
2. A second servo motor may be installed to allow vertical tilt for full 2-axis surveillance control.
3. The built-in face detection feature of the ESP32-CAM can be enabled to provide intelligent tracking.
4. Upgrading to an external Wi-Fi antenna may improve streaming quality and range.
5. A large capacitor or dedicated servo driver circuit should be included to improve power stability.

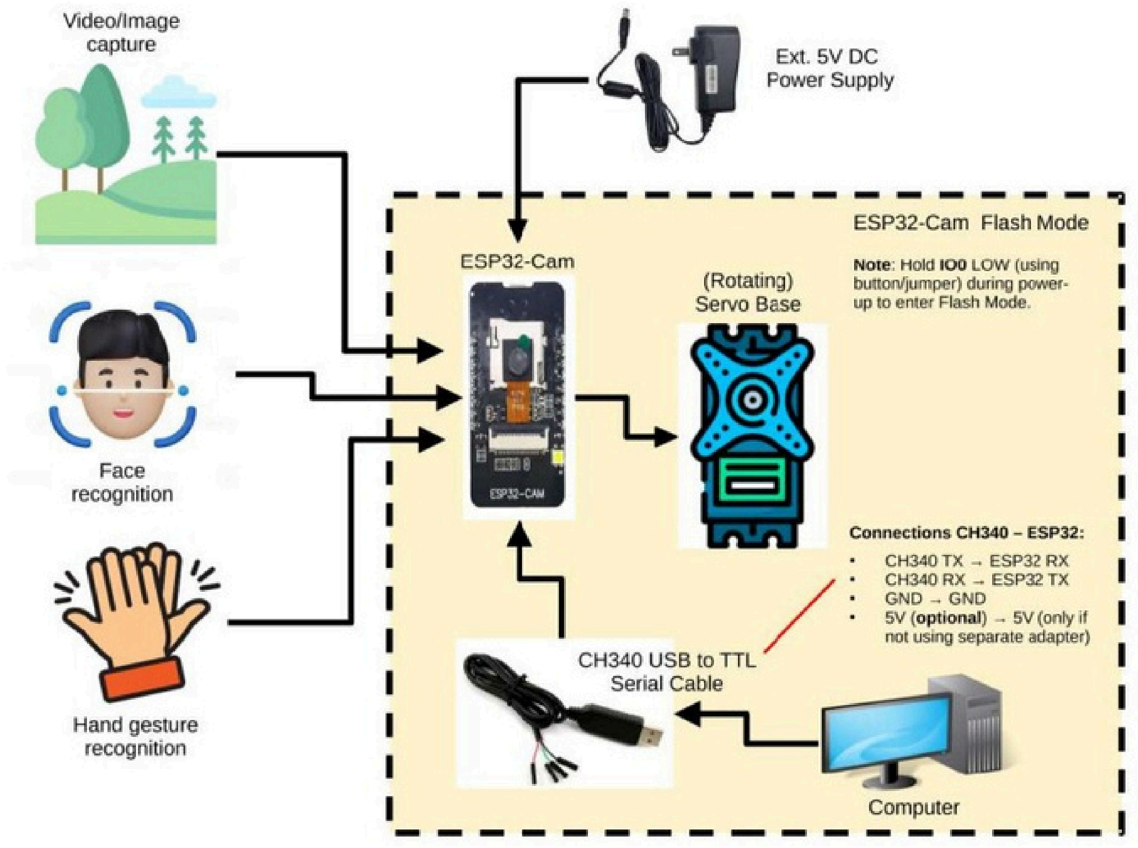
These recommendations will help advance this prototype into a more reliable and professional surveillance system suitable for real-world applications.

REFERENCES

[1] Random Nerd Tutorials, *How to Program / Upload Code to ESP32-CAM AI-Thinker (Arduino IDE).*
[2] Cytron Technologies, *CH340 USB to TTL Serial Cable.*
[3] Arduino-er Blog, *Program ESP32-CAM using FTDI Adapter.*
[4] YouTube Tutorial,*Program ESP32-CAM using FTDI Adapter.*
[5] Core Electronics, *UseanESP32-CAM Module to Stream HD Video Over Local Network.*

APPENDICES

System Overview:



Code used:

```
#include "esp_camera.h"
#include <WiFi.h>
#include <ESP32Servo.h>
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"

//NEW           CODE______________
volatile int detectedFaceX = 0;
volatile int detectedFaceY = 0;
Servo myservo1; Servo myservo2;
int pos = 0; int count = 0;
static const int servoPin1=13;
static const int servoPin2=14;
bool facedetect1 = false; const
int buttonpin = 15; //END NEW
CODE______________            //
===========================
//  Enter  your  WiFi  credentials  //
===========================    const
char*  ssid  =  "Afnan";  const  char*
password = "i2345678";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  myservo1.attach(servoPin1);
  myservo2.attach(servoPin2);
  myservo1.write(pos);
  myservo2.write(pos);
  pinMode(buttonpin,INPUT);
  Serial.setDebugOutput(true);
  Serial.println();

  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.frame_size  =  FRAMESIZE_UXGA;  //config.pixel_format
  =  PIXFORMAT_JPEG;  //  for  streaming  config.pixel_format  =
  PIXFORMAT_RGB565; // for face











detection/recognition (Use this one for face recognition)
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG
quality
  //                      for larger pre-allocated frame buffer.
  if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#ifdefined(CAMERA_MODEL_ESP_EYE)
  pinMode(13,   INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  //camera init
  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();
  //initial sensors are flipped vertically and colors are a bit
saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  //drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#ifdefined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

//Setup LED FLash if LED pin is defined in camera_pins.h
#ifdefined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

voidloop() {
  //Forbutton task
  intSTARTBUT = digitalRead(buttonpin);
  if(STARTBUT == HIGH)
```

```
 {
    myservo1.write(pos);
    pos += 30;
    if(pos>=180)
    {
      pos=0;
    }
  }
  //For servo detection
  if(facedetect1 == true && count == 0)
  {
    if(detectedFaceX > 180)
    {
      myservo1.write(180);
      myservo2.write(180);
    }
    else
    {
      myservo1.write(detectedFaceX);
      myservo2.write(detectedFaceY);
    }
  }
  delay(1000);
}
```

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to Sir Zulkifli bin Zainal Abidin , our course instructor for *Mechatronics System Integration (MCTA3203)*, for his invaluable guidance, encouragement, and continuous support throughout the completion of this Digital Logic System lab project.
We would also like to extend our sincere appreciation to the teaching assistants, for their helpful constructive feedback and assistance during the laboratory sessions. Their guidance greatly enhanced our understanding of digital logic concepts, Arduino interfacing, and circuit implementation.
Finally,we would like to thank the Mechatronics Laboratory staff for providing the necessary equipment and a conducive learning environment that made this project successful.

Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us.

Signature: [illegible]
Name: Afnan Hakim bin Adinazrin
Matric Number: 2315987
Contribution: Abstract,Introduction,Materials&Equipment,Results

Read	/
Understand	/
Agree	/

Signature: [illegible]
Name: Muhammad Taufiq bin Mukhtar
Matric Number: 2316271
Contribution: Equipment setup,Methodology,Data analysis
Recommendation

Read	/
Understand	/
Agree	/

Signature: [illegible]
Name: Muhammad Danish Farhan bin Amiruddin
Matric Number: 2315423
Contribution: Data collection,Discussion,Conclusion
Appendices,References

Read	/
Understand	/
Agree	/