



**REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT  
DEPARTMENT OF ELECTRICAL ENGINEERING  
UNIVERSITAS INDONESIA**

**CCTV ESP32 CAM Based with Joystick Control**

**GROUP 4**

<b>DANIEL NIKO MARDJAJA</b>	<b>2206026183</b>
<b>AQSHAL ILHAM SAMUDERA</b>	<b>2206829995</b>
<b>MUHAMMAD FAHISH HARITSAH BIMO</b>	<b>2206059616</b>
<b>FARHAN NUZUL NOUFENDRI</b>	<b>2206024442</b>

## PREFACE

Dalam era modern ini, teknologi Internet of Things (IoT) telah menjadi bagian penting dari kehidupan sehari-hari, menghadirkan solusi yang inovatif untuk berbagai kebutuhan manusia. Salah satu penerapannya yang paling relevan adalah dalam bidang keamanan, di mana sistem pengawasan berbasis IoT semakin banyak digunakan untuk menjaga properti, mengawasi lingkungan, dan meningkatkan rasa aman. Dengan kemampuan untuk terhubung dan dikendalikan dari jarak jauh, sistem ini memberikan fleksibilitas dan efisiensi yang belum pernah ada sebelumnya.

Peningkatan aksesibilitas perangkat keras seperti ESP32-CAM memungkinkan lebih banyak individu dan organisasi untuk merancang dan membangun sistem pengawasan yang canggih namun terjangkau. Modul ini tidak hanya berfungsi sebagai kamera, tetapi juga sebagai pusat kendali yang dapat terhubung dengan perangkat lain melalui jaringan internet. Ditambah dengan kontrol manual menggunakan joystick, sistem pengawasan ini menawarkan kemampuan untuk mengarahkan kamera sesuai kebutuhan secara real-time.

Melalui penggabungan teknologi ini, dihasilkan sebuah solusi pengawasan yang tidak hanya hemat biaya tetapi juga mudah untuk diimplementasikan. Diharapkan, pendekatan ini dapat memberikan inspirasi bagi siapa saja yang ingin memanfaatkan teknologi IoT untuk menciptakan sistem pengawasan yang efektif dan efisien.

Depok, December 10, 2024

Group 4

## TABLE OF CONTENTS

<b>CHAPTER 1.....</b>	<b>4</b>
<b>INTRODUCTION.....</b>	<b>4</b>
1.1 PROBLEM STATEMENT.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	6
1.5 TIMELINE AND MILESTONES.....	7
<b>CHAPTER 2.....</b>	<b>8</b>
<b>IMPLEMENTATION.....</b>	<b>8</b>
2.1 HARDWARE DESIGN AND SCHEMATIC.....	8
2.2 SOFTWARE DEVELOPMENT.....	10
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	28
<b>CHAPTER 3.....</b>	<b>30</b>
<b>TESTING AND EVALUATION.....</b>	<b>30</b>
3.1 TESTING.....	30
3.2 RESULT.....	34
3.3 EVALUATION.....	35
<b>CHAPTER 4.....</b>	<b>36</b>
<b>CONCLUSION.....</b>	<b>36</b>

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 PROBLEM STATEMENT**

Keamanan merupakan kebutuhan mendasar bagi setiap individu dan komunitas, terutama dalam melindungi properti, lingkungan, serta keselamatan pribadi. Namun, data kriminalitas menunjukkan bahwa masalah ini masih jauh dari kata teratas. Berdasarkan laporan Pusat Informasi Kriminal Nasional (Pusiknas) Polri, hingga pertengahan Juli 2024 tercatat 51.312 kasus pencurian dengan pemberatan dan pencurian biasa. Jenis kejahatan ini menjadi salah satu yang paling masif terjadi, bila mengesampingkan kecelakaan lalu lintas. Tingginya angka ini menunjukkan bahwa kebutuhan akan sistem keamanan yang efisien dan dapat diakses menjadi semakin mendesak.

Sayangnya, banyak sistem pengawasan keamanan yang tersedia saat ini memiliki kendala berupa biaya tinggi, kebutuhan infrastruktur yang kompleks, serta keterbatasan fleksibilitas. Sebagian besar sistem CCTV tradisional bersifat statis, memerlukan instalasi fisik yang rumit, dan hanya dapat diakses secara terbatas, sehingga kurang sesuai dengan kebutuhan pengawasan yang dinamis dan real-time. Kondisi ini menyulitkan banyak masyarakat, khususnya di wilayah dengan sumber daya terbatas, untuk memiliki sistem keamanan yang memadai.

Dengan perkembangan teknologi modern, khususnya dalam bidang Internet of Things (IoT), terdapat potensi besar untuk menghadirkan solusi yang lebih sederhana, hemat biaya, dan fleksibel. Teknologi ini memungkinkan integrasi perangkat yang mampu meningkatkan efektivitas pengawasan sekaligus memberikan kemudahan akses jarak jauh secara real-time. Hal ini membuka peluang untuk menciptakan sistem keamanan yang tidak hanya terjangkau, tetapi juga efisien dan mudah diimplementasikan oleh berbagai kalangan.

#### **1.2 PROPOSED SOLUTION**

Salah satu solusi yang bisa diterapkan untuk mengatasi kebutuhan sistem keamanan yang terjangkau dan efektif adalah implementasi sistem CCTV berbasis Internet of Things (IoT) yang memanfaatkan ESP32-CAM untuk streaming video dan pengambilan gambar. Modul ESP32-CAM digunakan sebagai perangkat utama untuk mengakses dan mentransmisikan data visual secara real-time, memungkinkan pengawasan yang efisien. Selain itu, sistem ini dilengkapi dengan ESP32 tambahan yang berfungsi untuk mengontrol posisi kamera menggunakan joystick, memberikan fleksibilitas dalam memantau area tertentu sesuai kebutuhan pengguna. Untuk mendukung kontrol dan pengiriman data secara real-time, sistem ini terintegrasi dengan Firebase yang berfungsi sebagai backend untuk penyimpanan data, serta Telegram yang memungkinkan pengguna menerima pemberitahuan dan mengendalikan sistem melalui aplikasi pesan secara langsung.

Dengan menggunakan ESP32-CAM, joystick, dan platform IoT seperti Firebase dan Telegram, solusi ini menawarkan sistem pengawasan yang terjangkau dan mudah diimplementasikan. Dibandingkan dengan sistem CCTV tradisional yang memerlukan perangkat keras mahal dan instalasi yang kompleks, solusi ini memberikan alternatif yang lebih hemat biaya dan fleksibel. Integrasi teknologi IoT memungkinkan pengawasan dapat dilakukan secara jarak jauh dan real-time, memberi pengguna kontrol penuh atas sistem keamanan mereka tanpa keterbatasan waktu atau lokasi. Dengan desain yang sederhana dan efisien, sistem ini diharapkan dapat membantu meningkatkan keamanan di berbagai lingkungan, khususnya bagi mereka yang membutuhkan solusi yang praktis dan dapat diakses dengan mudah.

### 1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. ESP32-CAM dapat melakukan streaming video secara langsung ke web server melalui URL
2. Kamera dapat digerakkan secara horizontal dan vertikal menggunakan servo yang terhubung dengan joystick ESP32.
3. Pengguna dapat mengirim perintah ke bot Telegram untuk melakukan tindakan-tindakan keamanan pada kamera, seperti mengambil foto dan menyalaikan atau mematikan lampu flash kamera.

4. ESP32-CAM harus dapat membaca dan memperbarui posisi servo menggunakan data X dan Y yang diambil dari Firebase secara real-time.
5. ESP32 joystick harus dapat membaca dan mengirimkan data X, Y, dan status tombol dengan akurat dan tanpa gangguan.
6. ESP32-CAM harus selalu membaca dan memperbarui posisi kamera berdasarkan nilai joystick yang ada di Firebase dalam waktu real-time.

## 1.4 ROLES AND RESPONSIBILITIES

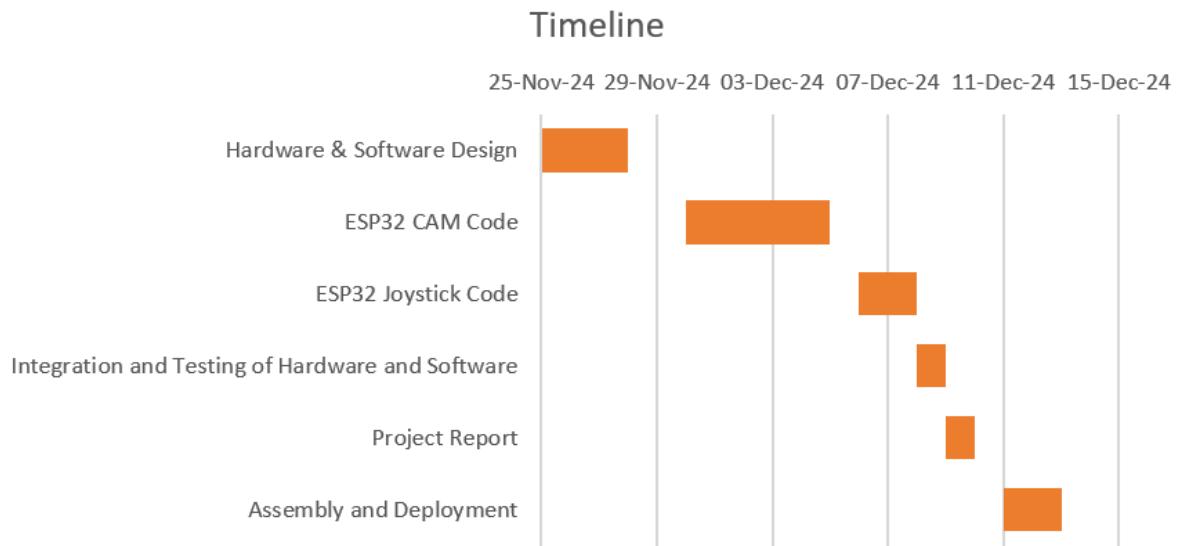
The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Role 1	Programming Kode ESP32 CAM, Pembuatan ReadME	DANIEL NIKO MARDJAJA
Role 2	Programming kode integrasi Telegram pada ESP32 CAM, Penulisan Laporan	AQSHAL ILHAM SAMUDERA
Role 3	Programming kode koneksi Firebase pada ESP32 CAM, Penulisan Laporan, Pembuatan ReadME	MUHAMMAD FAHISH HARITSAH
Role 4	Programming kode ESP32 Joystick	FARHAN NUZUL NOUFENDRI

Table 1. Roles and Responsibilities

## 1.5 TIMELINE AND MILESTONES

---



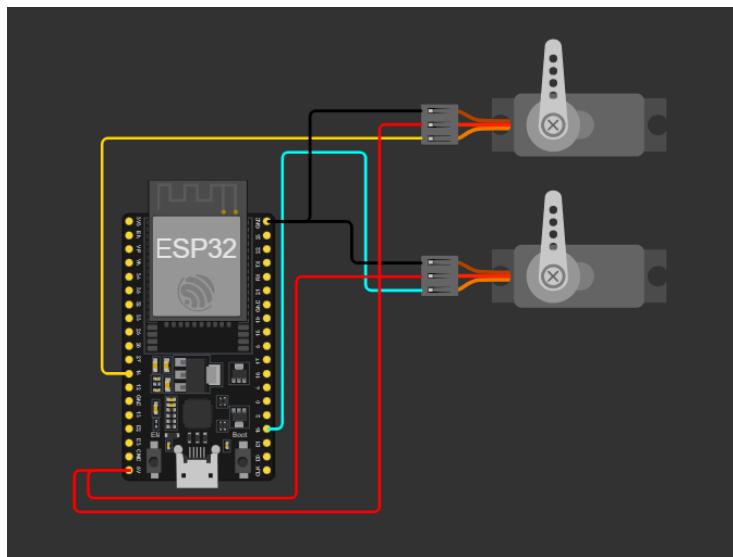
## CHAPTER 2

### IMPLEMENTATION

#### 2.1 HARDWARE DESIGN AND SCHEMATIC

Perancangan dan desain hardware dibuat sebagaimana rupa sehingga dapat menjalankan rangkaian agar bekerja sesuai rencana dan memenuhi keseluruhan kriteria dari tujuan pembuatan. Untuk memenuhi ini, dirancang sebuah alat yang menggunakan dua buah ESP32 berbeda dengan perannya masing-masing, yaitu ESP32 untuk fungsi CAM dan ESP32 untuk fungsi Joystick.

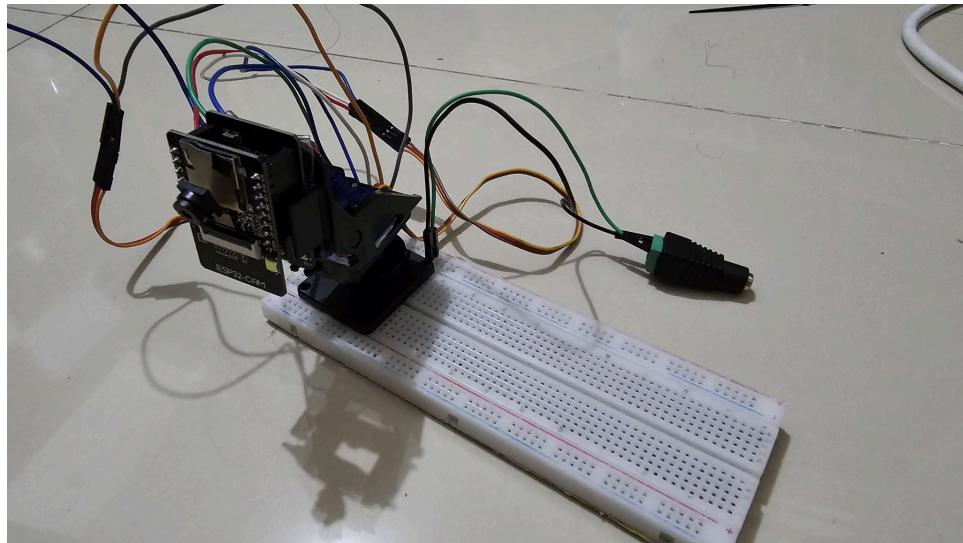
##### ESP32-CAM



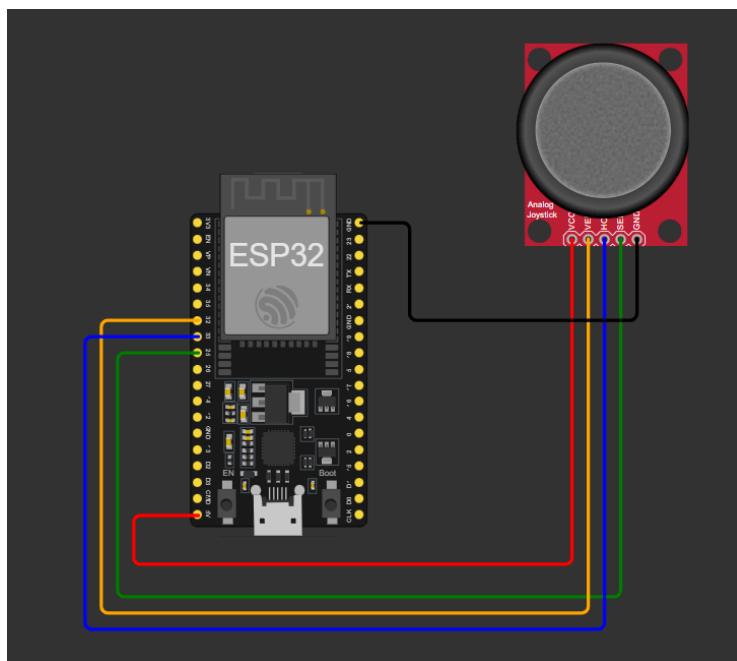
Perancangan hardware untuk ESP32 CAM dengan dua servo dilakukan sedemikian rupa sehingga rangkaian dapat menjalankan fungsi kendali motor servo secara akurat. ESP32 CAM berfungsi sebagai pengendali utama yang mengatur sinyal PWM untuk kedua servo. Kedua servo motor dihubungkan masing-masing ke pin GPIO pada ESP32 CAM, misalnya GPIO25 dan GPIO26. Pin daya servo (VCC) dihubungkan ke sumber daya eksternal 5V untuk memastikan suplai arus mencukupi, sedangkan ground servo dihubungkan ke ground dari sumber daya eksternal dan ESP32 CAM untuk menyamakan referensi tegangan. Untuk mendukung konektivitas perangkat, breadboard digunakan sebagai media untuk penyambungan kabel jumper yang menghubungkan komponen dengan rapi. Dalam sistem ini, ESP32 CAM disuplai daya melalui konektor USB atau regulator daya eksternal. Desain

ini memastikan setiap komponen dapat berfungsi tanpa gangguan, dengan sinyal PWM mengatur sudut pergerakan kedua servo secara terkoordinasi.

Maka, hasil fisik rangkaianya akan seperti berikut:



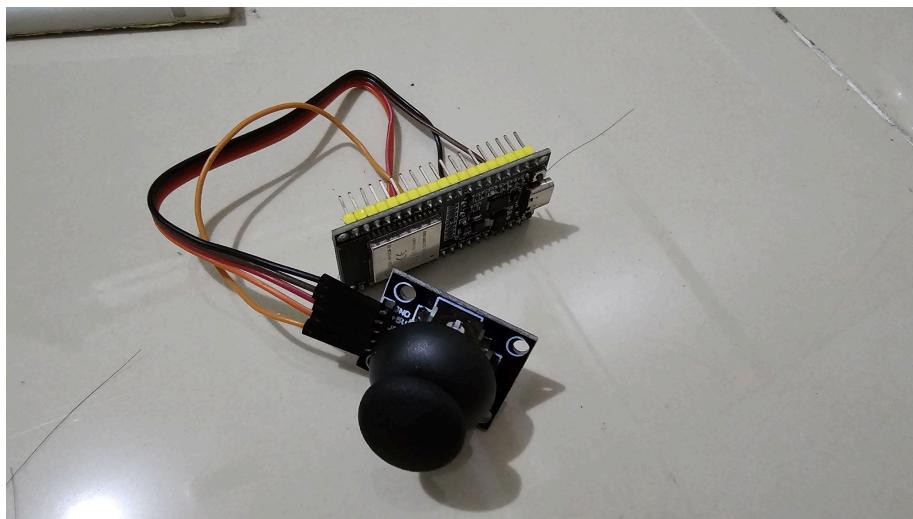
ESP32-Joystick



Desain hardware untuk ESP32 dengan joystick modul dirancang untuk membaca input dari joystick secara analog dan digital. Modul joystick memiliki tiga output utama, yaitu VRx, VRy, dan SW, yang dihubungkan ke ESP32. Output VRx dan VRy, yang berupa sinyal

analog, dihubungkan ke pin ADC ESP32, misalnya GPIO32 dan GPIO33. Sementara itu, pin SW, yang merupakan tombol tekan pada joystick, dihubungkan ke pin digital, misalnya GPIO27, untuk membaca kondisi aktif atau tidaknya tombol. Joystick disuplai daya melalui pin 3.3V pada ESP32, dan ground joystick dihubungkan langsung ke ground ESP32. Dengan desain ini, joystick dapat digunakan untuk mengontrol arah atau posisi berdasarkan gerakan sumbu X dan Y, sementara tombol digunakan untuk fungsi tambahan sesuai kebutuhan. Breadboard digunakan untuk merapikan rangkaian dan mempermudah penyambungan kabel antar komponen. Desain ini memastikan pembacaan sinyal dari joystick berjalan stabil dan terintegrasi dengan baik ke dalam sistem kendali ESP32.

Maka, hasil fisik rangkaianya akan seperti berikut:



## 2.2 SOFTWARE DEVELOPMENT

Pada Proyek ini kita akan memakai dua perangkat ESP32 yang terhubung melalui Wi-Fi: ESP32-CAM untuk kamera dan streaming video, serta ESP32 Joystick untuk mengontrol posisi kamera menggunakan joystick. Kemudian juga pada IoT kita sistemnya akan menggunakan Firebase untuk berbagi data secara real-time dan Telegram untuk kendali jarak jauhnya.

Disini ESP32-CAM bertanggung jawab untuk menangkap gambar, streaming video, dan mengontrol posisi kamera. Kamera OV2640 diatur untuk streaming langsung melalui server web yang di-host di ESP32. Kemudian nantinya posisi kamera itu akan di control

menggunakan servo motor, yang diatur berdasarkan data joystick (X, Y) yang disimpan di Firebase. Seorang usernya juga nanti bisa mengirim perintah melalui Telegram bot untuk mengambil foto atau menghidupkan/mematikan flash, dan foto dikirim kembali melalui Telegram.

Kemudian selanjutnya ada rangkaian ESP32 Joystick yang bisa membaca input dari joystick (X, Y) dan tombol. Data ini dikirimkan ke Firebase, yang kemudian dibaca oleh ESP32-CAM untuk mengubah posisi kamera. Jika tombol ditekan, perintah tambahan (seperti pengambilan foto) dikirim ke Firebase, yang akan diproses oleh ESP32-CAM.

Kemudian untuk integrasi dari kedua program itu nantinya komunikasi kedua perangkat akan dibantu dengan pemakaian Firebase. Alurnya itu nanti ESP32 Joystick mengirimkan data joystick ke Firebase, lalu ESP32-CAM membacanya untuk menggerakkan kamera sesuai input terbaru. Pengguna juga dapat memberikan perintah melalui Telegram bot untuk kontrol kamera, yang langsung diproses oleh ESP32-CAM.

Kemudian dengan bantuan Firebase ini membuat kita berhasil membuat sistem realtime sehingga posisi kamera dapat diubah dengan cepat. Dan kemudian user juga bisa mengontrol kamera secara jarak jauh melalui Telegram bot, yang bisa membuat user merasakan control yang efektif dan efisien

Kode Program:

ESP Pertama:

```
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <WiFi.h>
#include <WebServer.h>
#include <WiFiClientSecure.h>
#include <FirebaseESP32.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <ESP32Servo.h>
#include <addons/TokenHelper.h>
```

```
#include <addons/RTDBHelper.h>
#include <WiFiManager.h>

#define FIREBASE_HOST "" // URL Firebase Database
#define FIREBASE_AUTH "" // Secret Key Firebase

// Initialize Telegram BOT
String BOTtoken = "";
String CHAT_ID = "";

bool sendPhoto = false;

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);

#define FLASH_LED_PIN 4
#define SERVO_1 14
#define SERVO_2 15
bool flashState = LOW;

//Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

TaskHandle_t TaskCamHandle = NULL;
TaskHandle_t TaskFirebaseHandle = NULL;

WebServer server(80);

Servo servol;
Servo servo2;

int servolPos = 90;
int servo2Pos = 90;

// Firebase dan objek WiFi
FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;

//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

void configInitCamera(){
    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;

    //init with high specs to pre-allocate larger buffers
    if(psramFound()) {

```

```

    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10; //0-63 lower number means higher
quality
    config.fb_count = 2;
} else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12; //0-63 lower number means higher
quality
    config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    delay(1000);
    ESP.restart();
}

// Drop down frame size for higher initial frame rate
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_CIF); //  

UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
}

void handleNewMessages(int numNewMessages) {
    Serial.print("Handle New Messages: ");
    Serial.println(numNewMessages);

    for (int i = 0; i < numNewMessages; i++) {
        String chat_id = String(bot.messages[i].chat_id);
        if (chat_id != CHAT_ID){
            bot.sendMessage(chat_id, "Unauthorized user", "");
            continue;
        }

        // Print the received message
        String text = bot.messages[i].text;
        Serial.println(text);

        String from_name = bot.messages[i].from_name;
        if (text == "/start") {

```

```

        String welcome = "Welcome , " + from_name + "\n";
        welcome += "Use the following commands to interact with the
ESP32-CAM \n";
        welcome += "/photo : takes a new photo\n";
        welcome += "/flash : toggles flash LED \n";
        bot.sendMessage(CHAT_ID, welcome, "");
    }

    if (text == "/flash") {
        flashState = !flashState;
        digitalWrite(FLASH_LED_PIN, flashState);
        Serial.println("Change flash LED state");
    }

    if (text == "/photo") {
        sendPhoto = true;
        Serial.println("New photo request");
    }
}

string sendPhotoTelegram() {
    const char* myDomain = "api.telegram.org";
    String getAll = "";
    String getBody = "";

    camera_fb_t * fb = NULL;
    fb = esp_camera_fb_get();
    if(!fb) {
        Serial.println("Camera capture failed");
        delay(1000);
        ESP.restart();
        return "Camera capture failed";
    }

    Serial.println("Connect to " + String(myDomain));
}

if (clientTCP.connect(myDomain, 443)) {
    Serial.println("Connection successful");

    String head = "--electronicclinic\r\nContent-Disposition:
form-data; name=\"chat_id\"; \r\n\r\n" + CHAT_ID +
"\r\n--electronicclinic\r\nContent-Disposition: form-data;

```

```

name=\"photo\";
filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
String tail = "\r\n--electronicclinic--\r\n";

uint16_t imageLen = fb->len;
uint16_t extraLen = head.length() + tail.length();
uint16_t totalLen = imageLen + extraLen;

clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");
clientTCP.println("Host: " + String(myDomain));
clientTCP.println("Content-Length: " + String(totalLen));
clientTCP.println("Content-Type: multipart/form-data;
boundary=electronicclinic");
clientTCP.println();
clientTCP.print(head);

uint8_t *fbBuf = fb->buf;
size_t fbLen = fb->len;
for (size_t n=0;n<fbLen;n=n+1024) {
    if (n+1024<fbLen) {
        clientTCP.write(fbBuf, 1024);
        fbBuf += 1024;
    }
    else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        clientTCP.write(fbBuf, remainder);
    }
}

clientTCP.print(tail);

esp_camera_fb_return(fb);

int waitTime = 10000; // timeout 10 seconds
long startTimer = millis();
boolean state = false;

while ((startTimer + waitTime) > millis()){
    Serial.print(".");
    delay(100);
    while (clientTCP.available()) {
        char c = clientTCP.read();

```

```

        if (state==true) getBody += String(c);
        if (c == '\n') {
            if (getAll.length()==0) state=true;
            getAll = "";
        }
        else if (c != '\r')
            getAll += String(c);
        startTimer = millis();
    }
    if (getBody.length()>0) break;
}
clientTCP.stop();
Serial.println(getBody);

}

else {
    getBody="Connected to api.telegram.org failed.";
    Serial.println("Connected to api.telegram.org failed.");
}

return getBody;

}

const char HEADER[] =
"HTTP/1.1 200 OK\r\n"
"Access-Control-Allow-Origin: *\r\n"
"Content-Type: multipart/x-mixed-replace;
boundary=123456789000000000000987654321\r\n";
const char BOUNDARY[] = "\r\n--123456789000000000000987654321\r\n";
const char CTNTTYPE[] = "Content-Type: image/jpeg\r\nContent-Length:
";

void handleNotFound() {
    String message = "Server is running!\n\n";
    message += "URI: ";
    message += server.uri();
    message += "\nMethod: ";
    message += (server.method() == HTTP_GET) ? "GET" : "POST";
    message += "\nArguments: ";
    message += server.args();
    message += "\n";
    server.send(200, "text/plain", message);
}

```

```
}

void handle_jpg_stream() {
    char buf[32];
    int s;

    WiFiClient client = server.client();
    client.write(HEADER, strlen(HEADER));
    client.write(BOUNDARY, strlen(BOUNDARY));

    while (client.connected()) {
        if (!client.connected()) break;

        camera_fb_t *fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            break;
        }

        s = fb->len;
        client.write(CTNTTYPE, strlen(CTNTTYPE));
        sprintf(buf, "%d\r\n\r\n", s);
        client.write(buf, strlen(buf));
        client.write((char *)fb->buf, s);
        client.write(BOUNDARY, strlen(BOUNDARY));
        esp_camera_fb_return(fb);
    }
}

const char JHEADER[] =
    "HTTP/1.1 200 OK\r\n"
    "Content-Disposition: inline; filename=capture.jpg\r\n"
    "Content-Type: image/jpeg\r\n\r\n";

void handle_jpg() {
    WiFiClient client = server.client();
    camera_fb_t *fb = esp_camera_fb_get();

    if (!fb) {
        Serial.println("Camera capture failed");
        return;
    }
```

```

    client.write(JHEADER, strlen(JHEADER));
    client.write((char *)fb->buf, fb->len);
    esp_camera_fb_return(fb);
}

void updateServoPositions(int x, int y) {
    const int THRESHOLD_HIGH = 3072;
    const int THRESHOLD_LOW = 1024;
    const int STEP = 10;

    // Servo 2 (Horizontal Movement)
    if (x > THRESHOLD_HIGH && y > THRESHOLD_HIGH) {
        servo2Pos = constrain(servo2Pos + STEP, 0, 180);
    } else if (x < THRESHOLD_LOW && y < THRESHOLD_LOW) {
        servo2Pos = constrain(servo2Pos - STEP, 0, 180);
    }

    // Servo 1 (Vertical Movement)
    if (x < THRESHOLD_LOW && y > THRESHOLD_HIGH) {
        serv01Pos = constrain(serv01Pos + STEP, 0, 180);
    } else if (x > THRESHOLD_HIGH && y < THRESHOLD_LOW) {
        serv01Pos = constrain(serv01Pos - STEP, 0, 180);
    }

    serv01.write(serv01Pos);
    servo2.write(servo2Pos);
}

void TaskCam(void *pvParameters) {
    while(1) {
        if (sendPhoto) {
            Serial.println("Preparing photo");
            sendPhotoTelegram();
            sendPhoto = false;

        }
        if (millis() > lastTimeBotRan + botRequestDelay) {
            int numNewMessages = bot.getUpdates(bot.last_message_received +
1);
            while (numNewMessages) {
                Serial.println("got response");

```

```
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
}
lastTimeBotRan = millis();
}
vTaskDelay(10 / portTICK_PERIOD_MS);
}

void TaskFirebase(void *pvParameters) {
    int reconnectAttempts = 0;
    const int MAX_RECONNECT_ATTEMPTS = 5;

    while(1) {
        if (!Firebase.ready()) {
            Serial.println("Firebase not ready. Attempting to
reconnect...");
            Firebase.reconnectWiFi(true);

            reconnectAttempts++;
            if (reconnectAttempts >= MAX_RECONNECT_ATTEMPTS) {
                Serial.println("Failed to reconnect to Firebase.
Restarting...");
                ESP.restart();
            }

            delay(2000);
            continue;
        }

        reconnectAttempts = 0;

        int xValue = 0, yValue = 0, button = 0;

        if (!Firebase.getInt(firebaseData, "/joystick/X")) {
            Serial.println("Firebase X read error: " +
firebaseData.errorReason());
        } else {
            xValue = firebaseData.intData();
        }

        if (!Firebase.getInt(firebaseData, "/joystick/Y")) {
```

```
    Serial.println("Firebase Y read error: " +
firebaseData.errorReason());
} else {
    yValue = firebaseData.intData();
}

if (!Firebase.getInt(firebaseData, "/joystick/Button")) {
    Serial.println("Firebase Button read error: " +
firebaseData.errorReason());
} else {
    button = firebaseData.intData();
    if (button == 1) {
        delay(200);
        sendPhoto = true;
    }
}

updateServoPositions(xValue, yValue);
vTaskDelay(10 / portTICK_PERIOD_MS);
}

}

void setup() {
    WiFiManager wm;
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
    Serial.begin(115200);
    Serial.setDebugOutput(false);

    if (!psramFound()) {
        Serial.println("PSRAM not found. Ensure it is enabled in your board
configuration.");
        while (true);
    }

    pinMode(FLASH_LED_PIN, OUTPUT);
    digitalWrite(FLASH_LED_PIN, flashState);

    configInitCamera();

    if (!wm.autoConnect("ESP-CAM")) {
        Serial.println("Gagal menghubungkan ke WiFi, restart
ESP...");
```

```
ESP.restart();
}

clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);

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

Serial.println();
Serial.print("ESP32-CAM IP Address: ");
Serial.println(WiFi.localIP());

config.host = FIREBASE_HOST;
config.signer.tokens.legacy_token = FIREBASE_AUTH;

servo1.attach(SERVO_1, 1000, 2000);
servo2.attach(SERVO_2, 1000, 2000);

servo1.write(servo1Pos);
servo2.write(servo2Pos);

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
Serial.println("Terhubung ke Firebase!");

server.on("/mjpeg/1", HTTP_GET, handle_jpg_stream);
server.on("/jpg", HTTP_GET, handle_jpg);
server.onNotFound(handleNotFound);

server.begin();
Serial.println("HTTP server started");

xTaskCreatePinnedToCore(
    TaskCam,
    "CamTask",
    8192,
    NULL,
    1,
    &TaskCamHandle,
    0
```

```

) ;

xTaskCreatePinnedToCore(
    TaskFirebase,
    "FirebaseTask",
    8192,
    NULL,
    1,
    &TaskFirebaseHandle,
    1
);

void loop() {
    server.handleClient();
}

```

ESP Kedua :

```

#include <Arduino.h>
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <WiFiManager.h>

// Ganti dengan informasi Firebase Anda
#define FIREBASE_HOST ""
#define FIREBASE_AUTH ""

// Pin definitions
#define VRX_PIN      33
#define VRY_PIN      32
#define BUTTON_PIN   25

// Firebase dan objek WiFi
FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;

// Global variables

```

```
int xValue = 0;
int yValue = 0;
int buttonState = 1;

// Function prototypes
void TaskReadJoystick(void *pvParameters);
void TaskSendToFirebase(void *pvParameters);

void setup() {
    WiFiManager wm;
    Serial.begin(115200);

    pinMode(BUTTON_PIN, INPUT_PULLUP);

    if (!wm.autoConnect("ESP-Joystick")) {
        Serial.println("Gagal menghubungkan ke WiFi, restart
ESP...");
        ESP.restart();
    }

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

    config.host = FIREBASE_HOST;
    config.signer.tokens.legacy_token = FIREBASE_AUTH;

    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);
    Serial.println("Terhubung ke Firebase!");

    xTaskCreatePinnedToCore(
        TaskReadJoystick,
        "Read Joystick",
        1024,
        NULL,
        1,
        NULL,
        0
    );
}
```

```
xTaskCreatePinnedToCore (
    TaskSendToFirebase,
    "SendFirebase",
    8192,
    NULL,
    1,
    NULL,
    1)
;

}

void loop() {
    vTaskDelay(portMAX_DELAY);
}

void TaskReadJoystick(void *pvParameters) {
    (void) pvParameters;
    while (1) {
        xValue = analogRead(VRX_PIN);
        yValue = analogRead(VRY_PIN);
        buttonState = digitalRead(BUTTON_PIN);

        Serial.print("X = ");
        Serial.print(xValue);
        Serial.print(", Y = ");
        Serial.println(yValue);

        Serial.print("Button = ");
        if (buttonState == LOW) {
            Serial.println("PRESSED");
        } else {
            Serial.println("NOT PRESSED");
        }

        vTaskDelay(50 / portTICK_PERIOD_MS);
    }
}

void TaskSendToFirebase(void *pvParameters) {
    while (1) {
        if (Firebase.ready()) {
```

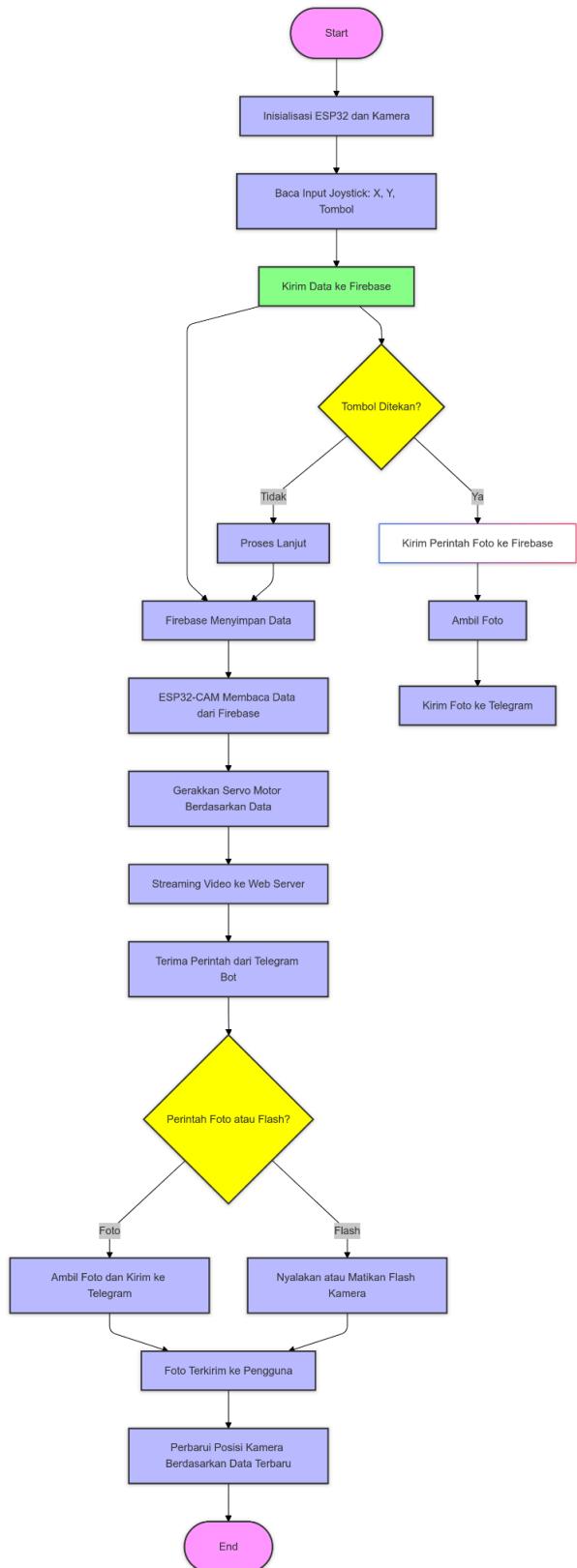
```
    if (Firebase.setInt(firebaseData, "/joystick/X", xValue)) {
        Serial.println("X value dikirim ke Firebase.");
    } else {
        Serial.print("Gagal mengirim X: ");
        Serial.println(firebaseData.errorReason());
    }

    if (Firebase.setInt(firebaseData, "/joystick/Y", yValue)) {
        Serial.println("Y value dikirim ke Firebase.");
    } else {
        Serial.print("Gagal mengirim Y: ");
        Serial.println(firebaseData.errorReason());
    }

    int buttonStatus = (buttonState == LOW) ? 1 : 0;
    if (Firebase.setInt(firebaseData, "/joystick/Button",
buttonStatus)) {
        Serial.println("Button state dikirim ke Firebase.");
    } else {
        Serial.print("Gagal mengirim Button: ");
        Serial.println(firebaseData.errorReason());
    }
} else {
    Serial.println("Firebase belum siap!");
}

vTaskDelay(500/ portTICK_PERIOD_MS);
}
}
```

## Flowchart Alur Program



## **2.3 HARDWARE AND SOFTWARE INTEGRATION**

Proses pengintegrasian antara hardware dan software pada sistem ini dimulai dengan konfigurasi dua modul utama, yaitu ESP32-CAM sebagai modul kamera dan ESP32 Joystick sebagai modul pengendali. Pada ESP32-CAM, kamera dikonfigurasikan melalui library bawaan ESP32 dengan menetapkan parameter seperti resolusi gambar, pin GPIO, dan format data. Modul ini juga dilengkapi dengan servo motor untuk pengendalian arah kamera, yang diatur menggunakan library ESP32Servo. Selain itu, ESP32-CAM memanfaatkan koneksi Wi-Fi untuk berkomunikasi dengan layanan Firebase dan Telegram, yang digunakan untuk menyimpan data dan mengirimkan foto dari kamera secara real-time.

Di sisi lain, ESP32 Joystick berfungsi sebagai modul input untuk mengontrol arah gerakan kamera. Joystick ini mengirimkan data posisi (X dan Y) serta status tombol (pressed/released) yang dibaca menggunakan fungsi analogRead dan digitalRead. Data ini kemudian dikirim ke Firebase menggunakan library FirebaseESP32, yang bertindak sebagai platform perantara untuk menyimpan data joystick. Melalui Firebase, ESP32-CAM dapat membaca data yang dikirimkan oleh joystick untuk mengatur posisi servo motor, memungkinkan pengendalian arah kamera secara jarak jauh.

Integrasi software dilakukan dengan pembuatan tugas menggunakan FreeRTOS, di mana masing-masing fungsi utama seperti pengambilan foto, pembaruan posisi servo, dan sinkronisasi data Firebase dijalankan pada task yang terpisah. Komunikasi antara ESP32-CAM dan Telegram menggunakan protokol HTTP Secure (HTTPS) memungkinkan pengguna untuk menerima gambar secara langsung dari kamera melalui bot Telegram. Sementara itu, konfigurasi jaringan Wi-Fi dilakukan dengan bantuan WiFiManager, yang memungkinkan kedua modul terhubung ke jaringan dengan mudah tanpa memerlukan konfigurasi manual. Dengan menerapkan kombinasi hardware dan software ini, pembuatan sistem yang terintegrasi untuk pengambilan gambar dan kontrol jarak jauh secara real-time melalui joystick dan layanan cloud dapat dicapai.

Pada rangkaian asli, akan dilakukan penghubungan antara ESP32 dengan komputer yang memiliki Arduino IDE untuk proses pemrograman dan debugging. Komputer digunakan untuk mengunggah kode ke ESP32-CAM dan ESP32 Joystick melalui koneksi USB. Arduino IDE juga berperan dalam memantau data serial untuk memastikan bahwa komunikasi antara perangkat berjalan dengan baik, seperti pengiriman data joystick ke Firebase atau

pengambilan gambar oleh ESP32-CAM. Setelah kode berhasil diunggah, ESP32-CAM dan ESP32 Joystick akan bekerja secara mandiri sesuai dengan program yang telah dibuat, memungkinkan integrasi antara perangkat keras dan perangkat lunak untuk pengendalian kamera jarak jauh yang efisien.

## CHAPTER 3

### TESTING AND EVALUATION

#### 3.1 TESTING

Proses pengujian dilakukan secara langsung menggunakan Arduino IDE dengan mengupload modul ESP CAM dan ESP Joystick. Tujuan percobaan ini untuk memastikan ESP CAM dapat terhubung ke Firebase, dapat menangkap gambar atau streaming melalui kamera, dan menerapkan perintah-perintah yang diinput oleh user. Begitu juga dengan ESP Joystick dapat terhubung ke web portal agar terhubung ke WiFi.

Output ESP-CAM

```
x : 3107
y : 2992*wm:AutoConnect
*wm:Connecting to SAVED AP: Room - 2304
*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:AutoConnect: SUCCESS
*wm:STA IP Address: 192.168.55.110

ESP32-CAM IP Address: 192.168.55.110
Terhubung ke Firebase!
HTTP server started

x : 3107
y : 2992
button : 0
x : 3107
y : 2992
button : 0
x : 3107
y : 2992
button : 0
x : 3107
```

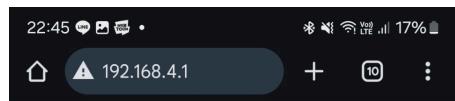
Ini adalah output dari ESP-CAM setelah koneksi ke wifi. ESP-CAM ini menggunakan wifi manager untuk mengkoneksikan wifi secara dinamis dan dapat terlihat diatas karena wifi telah terkoneksi sebelumnya maka terdapat auto connect yang akan mengoneksikan secara otomatis wifi dari ESP32. setelah terhubung ke wifi ESP-CAM akan membuat web server

yang digunakan untuk stream video hasil kamera dan akan menghubungkan ke firebase untuk mengambil data dari firebase.

## Output ESP-Joystick

```
Button = NOT PRESSED
*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:AutoConnect: FAILED for 3154 ms
*wm:StartAP with SSID: ESP-Joystick
*wm:AP IP address: 192.168.4.1
*wm:Starting Web Portal
```

Gambar di atas adalah output ketika ESP-Joystick membuka web portal untuk menghubungkan ESP32-joystick ke wifi. IP untuk web portal secara default adalah 192.168.4.1. Hp user harus terhubung ke AP dari ESP32-joystick untuk membuka Wifi Manager.



## WiFiManager

ESP-Joystick

Configure WiFi

Info

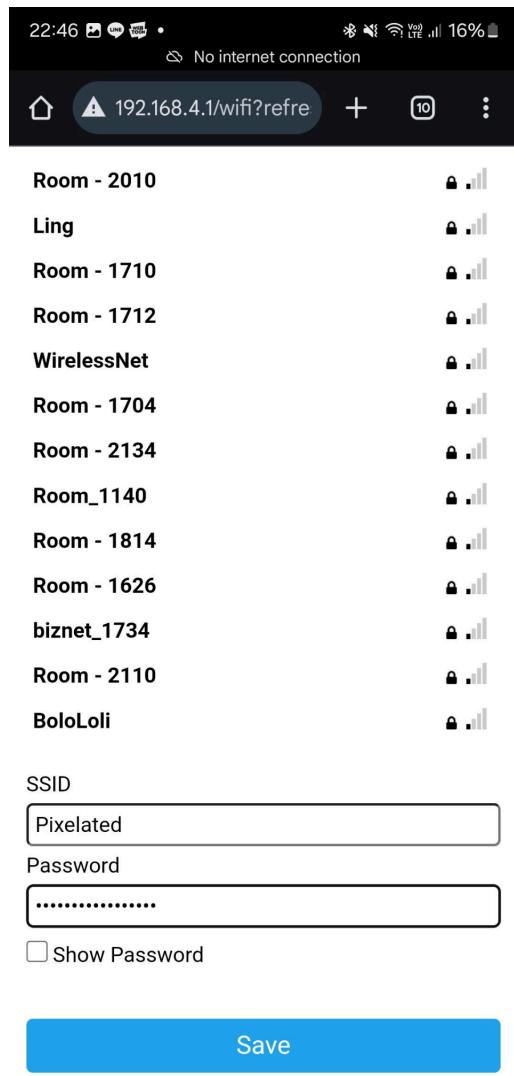
Exit

Update

Not connected to Pixelated  
AP not found

Gambar di atas adalah tampilan dari wifi manager saat pertama kali dibuka. Untuk hp berjenis samsung, harus membuka web portal secara manual yaitu dengan memasukkan

192.168.4.1 ke browser. Sedangkan pada merek hp lain akan langsung membuka web portal seperti saat melakukan koneksi ke hotspot ui.



Gambar tersebut adalah tampilan ketika menunjukkan seluruh wifi yang ada dan akan memasukan SSID serta password agar ESP32 dapat langsung terkoneksi dengan wifi yang diinginkan.

```
*wm:AP IP address: 192.168.4.1
*wm:Starting Web Portal
*wm:33 networks found
*wm:33 networks found
*wm:33 networks found
*wm:33 networks found
*wm:32 networks found
*wm:Connecting to NEW AP: Pixelated
*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:Connect to new AP [SUCCESS]
*wm:Got IP Address:
*wm:192.168.226.36
*wm:config portal exiting

Terhubung ke WiFi!
Terhubung ke Firebase!
X = 3104, Y = 3006
Button = NOT PRESSED
X = 3063, Y = 2987
```

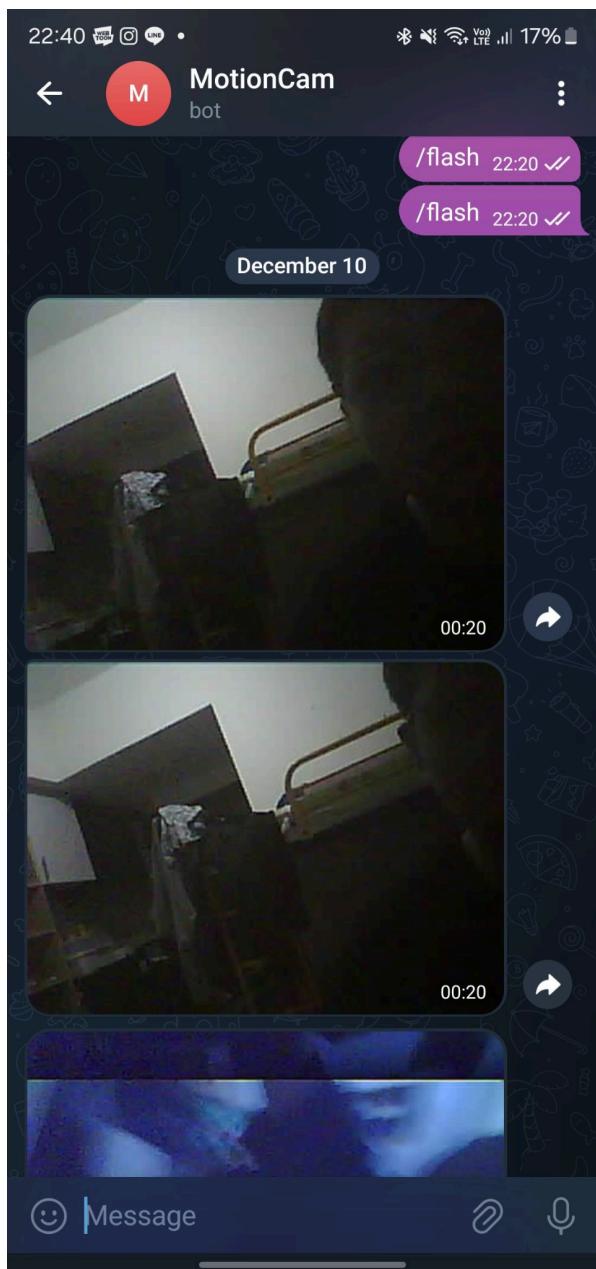
Setelah melakukan konfigurasi wifi pada web portal, maka akan terlihat bahwa nama AP akan terlihat dan akan terlihat pula IP address dari ESP-Joystick. Kemudian ESP32-Joystick akan membaca input sensor yaitu joystick dan akan mengirimkan ke firebase.

Berikut adalah output yang dihasilkan ketika mengirim ke firebase dan saat membaca sensor:

```
Terhubung ke WiFi!
Terhubung ke Firebase!
X = 3104, Y = 3006
Button = NOT PRESSED
X = 3063, Y = 2987
Button = NOT PRESSED
X = 3105, Y = 3020
Button = NOT PRESSED
X = 3115, Y = 3023
Button = NOT PRESSED
X = 3093, Y = 2987
Button = NOT PRESSED
X = 3108, Y = 2998
Button = NOT PRESSED
X = 3085, Y = 2992
Button = NOT PRESSED
X = 3061, Y = 2986
Button = NOT PRESSED
X = 3069, Y = 2983
Button = NOT PRESSED
X = 3071, Y = 2993
Button = NOT PRESSED
X = 3071, Y = 2989
Button = NOT PRESSED
X = 3060, Y = 2992
Button = NOT PRESSED
X value dikirim ke Firebase.
X = 3063, Y = 2993
Button = NOT PRESSED
```

### 3.2 RESULT

Berikut adalah hasil yang sudah dijalankan sebelumnya yaitu untuk melakukan foto dan akan mengirimkannya ke telegram. Hal ini dapat terjadi ketika user mengetikkan /photo di telegram ataupun ketika button dari joystick ditekan.



Berikut adalah hasil dari stream ESP-CAM ke web server yang sudah berhasil mendapatkan output yang sesuai.



### 3.3 EVALUATION

Evaluasi yang mungkin dilakukan dalam proyek ini mencakup beberapa aspek penting, seperti performa sistem dalam mengirim dan menerima data secara real-time antara ESP32 Joystick, Firebase, dan ESP32-CAM. Stabilitas koneksi WiFi juga perlu diuji untuk memastikan perangkat dapat berfungsi dengan baik di lingkungan dengan gangguan sinyal. Selain itu, responsivitas kontrol servo melalui joystick dan akurasi pergerakannya dalam mengarahkan kamera harus dievaluasi, termasuk keterlambatan (latency) yang terjadi.

Konsumsi daya sistem juga perlu diperhatikan, terutama jika perangkat digunakan untuk operasi jangka panjang. Pengujian pada fitur pengambilan foto dan pengiriman gambar melalui Telegram juga harus dilakukan untuk memastikan kualitas gambar dan waktu pengiriman sesuai kebutuhan. Terakhir, usability atau kemudahan pengguna dalam mengoperasikan sistem, seperti antarmuka Telegram dan respon Firebase, dapat menjadi fokus evaluasi untuk meningkatkan pengalaman pengguna.

## **CHAPTER 4**

### **CONCLUSION**

Proyek ini berhasil menunjukkan bagaimana integrasi perangkat ESP32 Joystick, Firebase, dan ESP32-CAM dapat digunakan untuk menciptakan sistem kontrol kamera jarak jauh berbasis IoT. Dengan memanfaatkan Firebase sebagai platform komunikasi data real-time, proyek ini memungkinkan pengguna untuk mengendalikan pergerakan servo pada ESP32-CAM menggunakan joystick secara akurat dan responsif. Fitur tambahan seperti pengambilan gambar melalui kamera dan pengiriman hasilnya ke Telegram menambah nilai praktis dari sistem ini, terutama untuk kebutuhan pemantauan jarak jauh. Keseluruhan implementasi menunjukkan bahwa kombinasi perangkat keras dan perangkat lunak yang digunakan dapat diandalkan dalam menyediakan solusi IoT yang efisien dan fleksibel.

Namun, terdapat beberapa hal yang perlu diperhatikan untuk pengembangan lebih lanjut. Stabilitas koneksi WiFi dan responsivitas sistem dalam berbagai kondisi lingkungan masih perlu diuji lebih mendalam untuk memastikan kinerja optimal. Selain itu, pengurangan latensi komunikasi data serta peningkatan antarmuka pengguna di Telegram dapat menjadi fokus perbaikan. Dengan berbagai potensi yang ada, proyek ini memiliki aplikasi luas, mulai dari sistem pemantauan keamanan hingga pengawasan di area yang sulit dijangkau, menjadikannya solusi inovatif yang dapat disesuaikan dengan berbagai kebutuhan.

## REFERENCES

- [1] Techiesms, "ESP32 CAM Send Photo to Telegram with PIR Sensor," YouTube, Sep. 14, 2020. [Online]. Available: <https://www.youtube.com/watch?v=UA3cqgpFHX>. [Accessed: Dec. 10, 2024].
- [2] Robu.in, "ESP32 CAM Send Photo to Telegram," YouTube, Aug. 6, 2020. [Online]. Available: <https://www.youtube.com/watch?v=D0mCmRcsd7w>. [Accessed: Dec. 10, 2024].
- [3] Play With Ideas, "ESP32 Joystick," YouTube, Jan. 24, 2022. [Online]. Available: <https://www.youtube.com/watch?v=XGYmqotY6mo>. [Accessed: Dec. 10, 2024].

## **APPENDICES**

### **Appendix A: Project Schematic**

Put your final project latest schematic here

### **Appendix B: Documentation**

Put the documentation (photos) during the making of the project