LAPORAN PRAKTIKUM
SISTEM DETEKSI WAJAH MENGGUNAKAN ESP32 DAN OPENCV(PYTHON)

Disusun Oleh:

Nama	: L Hafidl Alkhair
NIM	: 2023903430060
Kelas	: TRKJ 2.C
Jurusan	: Teknologi Informasi dan Komputer
Program Studi	: Teknologi Rekayasa Komputer Jaringan
Dosen Pembimbing	: Attahariq,SST.,M.T

JURUSAN TEKNOLOGI INFORMASI KOMPUTER

PRODI TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEUMAWE

TAHUN AJARAN 2024-2025

BAB I
PENDAHULUAN

A. Latar Belakang

Dalam era teknologi yang terus berkembang, sistem keamanan berbasis pengenalan wajah semakin banyak diterapkan untuk meningkatkan efisiensi dan keamanan. Pengenalan wajah menawarkan solusi tanpa kontak fisik untuk proses autentikasi dan verifikasi. Proyek ini menggunakan ESP32 sebagai pengendali dan OpenCV (Python) untuk mendeteksi wajah. Komunikasi antara Python dan ESP32 dilakukan melalui komunikasi serial, yang memungkinkan pengendalian perangkat keras seperti LED dan buzzer untuk memberi tanda visual dan suara. Penggunaan ESP32 mempermudah integrasi dengan berbagai sensor dan aktuator, sementara OpenCV memberikan kemampuan deteksi wajah yang cepat dan akurat.

B. Tujuan

Adapun tujuan dari praktikum ini adalah:

1. Mempelajari cara menggunakan OpenCV untuk mendeteksi wajah.
2. Mengimplementasikan komunikasi serial antara Python dan ESP32.
3. Mengontrol perangkat keras (LED dan buzzer) menggunakan sinyal dari sistem deteksi wajah.
4. Mengintegrasikan sistem deteksi wajah untuk membedakan wajah yang dikenali dan tidak dikenali.

BAB II

ALAT DAN BAHAN

A. Alat
   1. Laptop/PC dengan sistem operasi Windows/Linux.
   2. ESP32
   3. Kamera/webcam.
   4. Kabel USB untuk koneksi ESP32.
   5. LED (Merah dan Hijau).
   6. Buzzer.
   7. Breadboard.
   8. Kabel jumper.

B. Bahan
   1. Software Python (dengan pustaka opencv, face_recognition, numpy, dan pyserial).
   2. Arduino IDE untuk memprogram ESP32.
   3. File gambar untuk data wajah yang dikenal (disimpan dalam folder images).

BAB III
LANGKAH KERJA

A. Langkah Kerja

a. Install Python dan pustaka yang diperlukan dengan perintah berikut:
"pip install opencv-python face_recognition numpy pyserial"

b. Install Arduino IDE dan siapkan driver ESP32

c. Buat folder bernama images dan simpan gambar wajah yang akan dikenali. Nama file gambar akan digunakan sebagai nama pemilik wajah.

d. Buat file Python dan salin kode berikut:

```
import cv2
import face_recognition
import os
import numpy as np
import serial
import time
import threading


# Inisialisasi komunikasi serial dengan ESP32
try:
    arduino = serial.Serial('COM4', 9600, timeout=1) # Ganti 'COM4' dengan
port ESP32 Anda
    time.sleep(2)  # Tunggu koneksi serial stabil
    print("Koneksi serial dengan ESP32 berhasil.")
except serial.SerialException as e:
    print(f"Error: {e}")
    arduino = None


# Load known faces dari folder "images"
known_face_encodings = []
known_face_names = []
```

```
image_folder = "images"
for filename in os.listdir(image_folder):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        image_path = os.path.join(image_folder, filename)
        image = face_recognition.load_image_file(image_path)
        encoding = face_recognition.face_encodings(image)[0]  # Ambil encoding pertama
        known_face_encodings.append(encoding)
        known_face_names.append(os.path.splitext(filename)[0])  # Nama file tanpa ekstensi

print(f"Loaded {len(known_face_encodings)} known faces from '{image_folder}'.")

# Inisialisasi webcam
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Kamera tidak dapat dibuka.")
    exit()

# Fungsi untuk mengirim data ke ESP32
def send_to_arduino(command):
    if arduino:
        arduino.write(command.encode())

last_command = None

while True:
    success, frame = cap.read()
    if not success:
        print("Gagal menangkap gambar dari kamera.")
        continue
```

```
    # Convert frame ke RGB untuk face recognition
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Deteksi wajah di frame saat ini
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame,
face_locations)

    face_matched = False

    for face_encoding, face_location in zip(face_encodings, face_locations):
        # Cek apakah wajah cocok dengan yang ada di folder images
        matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
        face_distances =
face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)

        # Dapatkan koordinat wajah
        top, right, bottom, left = face_location

        # Gambar kotak hitam dasar yang mengikuti wajah
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 0), 2)

        if matches[best_match_index]:
            name = known_face_names[best_match_index]
            face_matched = True

            # Ubah warna kotak menjadi hijau untuk wajah yang terverifikasi
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
```

```
            cv2.rectangle(frame, (left, top - 35), (right, top), (0, 255, 0), -1)  # Background hijau untuk teks
            cv2.putText(frame, f"Authorized: {name}", (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 255), 2)

            if last_command != "green\n":
                threading.Thread(target=send_to_arduino, args=("green\n",)).start()
                last_command = "green\n"
        else:
            # Ubah warna kotak menjadi merah untuk wajah yang tidak dikenali
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
            cv2.rectangle(frame, (left, top - 35), (right, top), (0, 0, 255), -1)  # Background merah untuk teks
            cv2.putText(frame, "Unauthorized", (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 255), 2)

            if last_command != "red\n":
                threading.Thread(target=send_to_arduino, args=("red\n",)).start()
                last_command = "red\n"

    # Jika tidak ada wajah yang cocok dan belum ada perintah "red\n", kirim perintah merah
    if not face_matched and last_command != "red\n":
        threading.Thread(target=send_to_arduino, args=("red\n",)).start()
        last_command = "red\n"

    # Tampilkan hasil di layar
    cv2.imshow("Face Recognition", frame)

    if cv2.waitKey(1) & 0xFF == 27:  # Tekan ESC untuk keluar
```

```
        break

cap.release()
cv2.destroyAllWindows()
if arduino:
    arduino.close()
```

e. Buka Arduino IDE dan salin kode berikut:

```
const int redPin = 2;    // Pin untuk LED merah
const int greenPin = 4;  // Pin untuk LED hijau
const int buzzerPin = 15; // Pin untuk buzzer

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(buzzerPin, LOW);

  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
```

```
  if (command == "green") {
    // Wajah sesuai: Nyalakan LED hijau, matikan LED merah dan buzzer
    digitalWrite(greenPin, HIGH);
    digitalWrite(redPin, LOW);
    digitalWrite(buzzerPin, LOW);
  } else if (command == "red") {
    // Wajah tidak sesuai: Nyalakan LED merah dan buzzer, matikan LED hijau
    digitalWrite(greenPin, LOW);
    digitalWrite(redPin, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay(500);  // Buzzer berbunyi selama 500ms
    digitalWrite(buzzerPin, LOW);
  }
 }
}
```

f. Unggah kode ke ESP32 menggunakan kabel USB.

g. Hubungkan pin ESP32 ke perangkat keras sesuai dengan konfigurasi berikut:

- LED Merah: Pin 2
- LED Hijau: Pin 4
- Buzzer: Pin 15

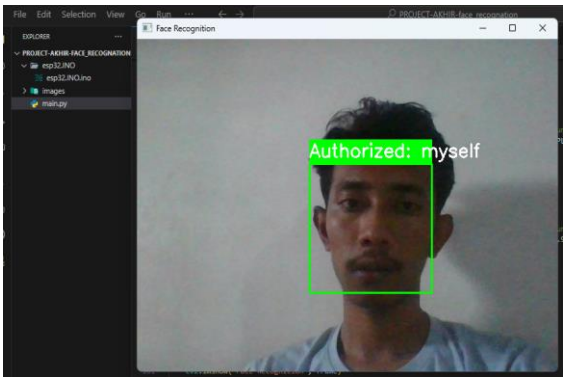
Pastikan semua koneksi sesuai skema.

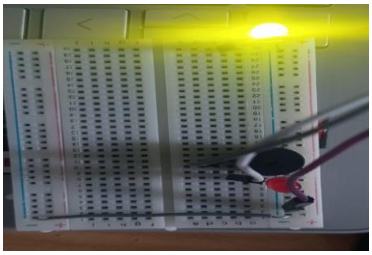
h. Jalankan file maka Sistem akan mendeteksi wajah menggunakan webcam dan mengirimkan sinyal ke ESP32 untuk mengontrol LED dan buzzer.

BAB IV
HASIL DAN PEMBAHASAN

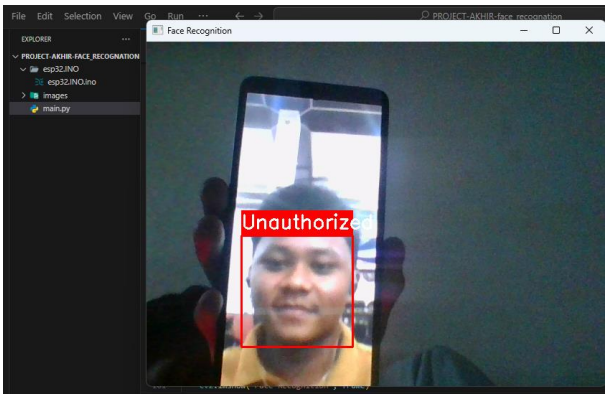
A. HASIL

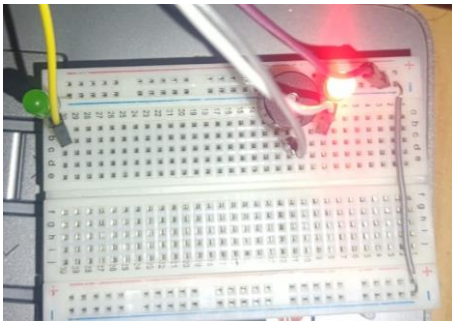
1. Sistem berhasil mendeteksi wajah yang dikenali dan menyalakan LED hijau sebagai tanda autentikasi berhasil.





2. Jika wajah tidak dikenali, sistem menyalakan LED merah dan mengaktifkan buzzer selama 500 ms.





3. Komunikasi serial antara Python dan ESP32 berjalan dengan lancar.

Kondisi	LED Merah	LED Hijau	Buzzer
Wajah dikenali	OFF	ON	OFF
Wajah tidak dikenali	ON	OFF	ON (500 ms)

B. PEMBAHASAN

1. Efektivitas Sistem:
    - Integrasi antara OpenCV untuk deteksi wajah dan ESP32 untuk kontrol perangkat keras terbukti efektif dalam memberikan solusi autentikasi berbasis wajah secara real-time.
    - Proses komunikasi serial yang stabil memungkinkan instruksi dari Python untuk dikirim ke ESP32 tanpa penundaan signifikan.
2. Kualitas Deteksi:
    - Penggunaan pustaka face_recognition: Pustaka ini mempermudah proses deteksi wajah dengan akurasi yang baik, namun bergantung pada kualitas gambar referensi.
    - Untuk meningkatkan akurasi, gambar wajah referensi sebaiknya diambil dengan pencahayaan yang konsisten dan sudut yang jelas.
3. Penggunaan ESP32:
    - ESP32 berperan sebagai pengendali untuk memberikan output visual dan audio. Keunggulan ESP32 adalah kemampuannya untuk menangani tugas multitasking sederhana dan komunikasi serial dengan perangkat eksternal.
    - LED dan buzzer digunakan sebagai indikator visual dan audio untuk memberikan umpan balik yang jelas kepada pengguna.
4. Kendala dan Solusi:
    - Pencahayaan: Deteksi wajah kurang akurat dalam kondisi pencahayaan rendah. Solusi: tambahkan sumber cahaya eksternal atau gunakan algoritma peningkatan gambar.
    - Kecepatan Proses: Sistem dapat mengalami penundaan jika terlalu banyak wajah yang harus dibandingkan. Solusi: optimalkan jumlah gambar referensi atau gunakan perangkat dengan GPU yang lebih baik.

BAB V
KESIMPULAN

A. KESIMPULAN

Berdasarkan hasil dan pembahasan dari praktikum ini, dapat disimpulkan bahwa:

1. Sistem deteksi wajah menggunakan OpenCV dan ESP32 bekerja efektif dalam membedakan wajah yang dikenali dan tidak dikenali. Wajah yang sesuai menghasilkan indikator positif berupa LED hijau, sedangkan wajah yang tidak dikenali memicu LED merah dan buzzer sebagai peringatan.
2. Komunikasi serial antara Python dan ESP32 berjalan dengan baik dan stabil, memungkinkan pengiriman perintah kontrol perangkat keras secara real-time. Ini membuktikan bahwa ESP32 dapat diandalkan untuk proyek pengendalian sederhana dengan instruksi dari komputer.
3. Akurasi deteksi wajah dipengaruhi oleh kualitas gambar dan kondisi lingkungan. Pencahayaan yang baik dan gambar referensi yang jelas akan meningkatkan performa sistem. Kendala seperti wajah tertutup sebagian atau jarak yang terlalu jauh dapat diatasi dengan pengaturan lebih lanjut pada algoritma deteksi.
4. Integrasi perangkat keras sederhana (LED dan buzzer) dengan deteksi wajah memberikan solusi yang mudah diterapkan untuk sistem keamanan berbasis pengenalan wajah. Proyek ini dapat dikembangkan lebih lanjut dengan menambahkan fitur seperti penyimpanan log akses atau penggunaan sensor tambahan.

Dengan hasil yang diperoleh, sistem ini dapat menjadi prototipe awal untuk aplikasi keamanan di berbagai bidang seperti akses pintu otomatis, sistem absensi, atau perangkat keamanan rumah tangga.