UAS INSTRUMENTASI BERBASIS JARINGAN

LAPORAN

Diajukan untuk memenuhi salah satu tugas Mata Kuliah

Instrumentasi Berbasis Jaringan

Dosen Pengampu: Drs. Maman Budiman, M. Eng., Ph.D. dan

Dr. Nina Siti Aminah, S.Si.,M.Si.

Oleh :

Muhammad Naufal Fadhilah	(10220059)
Muhammad Fazli Rais	(10220072)
Elizabeth Michele Simbolon	(10220004)

FAKULTAS ILMU PENGETAHUAN ALAM

INSTITUT TEKNOLOGI BANDUNG

FISIKA

TAHUN AJARAN 2023/2024

DAFTAR ISI

Bab I Pendahuluan

Pada era digital saat ini, teknologi keamanan semakin berkembang dengan pesat. Salah satu inovasi yang menarik perhatian adalah penggunaan *face recognition* atau pengenalan wajah sebagai metode autentikasi. *Face recognition* memungkinkan sistem untuk mengidentifikasi dan memverifikasi identitas seseorang berdasarkan ciri-ciri wajah yang unik. Pintu pintar atau smartdoor dengan sistem *face recognition* merupakan salah satu contoh implementasi praktis dari teknologi ini. Pintu pintar ini memiliki kemampuan untuk mengenali wajah pengguna yang telah terdaftar sehingga dapat memberikan akses yang terkontrol. Dengan menggunakanPython libraries seperti OpenCV dan *face recognition*, pengembang dapat memanfaatkan kekuatan deep learning untuk melakukan pengenalan wajah dengan akurasi yang tinggi.

Pada laporan ini, kami akan membahas tentang cara kerja *face recognition* menggunakan Python libraries seperti OpenCV dan *face recognition* untuk pintu pintar cerdas. Kami akan melakukan analisis yang meliputi tahap deteksi wajah, ekstraksi fitur wajah, dan pencocokan dengan wajah yang terdaftar. Selain itu, kami juga akan menjelaskan bagaimana teknologi deep learning, khususnya penggunaan library *face recognition* dengan fungsi face_encodings, dapat digunakan untuk meningkatkan akurasi pengenalan wajah. Dalam laporan ini, kami akan menguraikan langkah-langkah implementasi dan mengevaluasi kinerja sistem pintu pintar berbasis *face recognition*. Analisis dan kesimpulan yang dihasilkan akan memberikan wawasan tentang potensi dan keterbatasan teknologi ini, serta saran-saran untuk meningkatkan kinerja sistem.

Tujuan dari eksperimen ini, yaitu

1. Membuat suatu sistem kunci pintu otomatis melalui pengenalan wajah dan sensor jarak
2. Membuat sistem keamanan pintu dengan meloloskan dua wajah dan menolak wajah lain yang tidak terdaftar

Sertakan batasan-batasan pada percobaan yang dilakukan:

1. Stepper motor yang digunakan adalah 28BYJ-48
2. Sensor yang digunakan untuk menentukan kapan pintu terkunci adalah sensor ultrasonik

3. Hanya terdapat 2 orang sebagai wajah yang diloloskan
4. Realisasi alat menggunakan sebuah prototyping smart door dengan *face recognition*, dimana prototype yang dibuat menggunakan pintu dari aklirik yang dipasang dengan “selotan” atau kunci pintu sederhana sebagai penguncinya

Asumsi yang digunakan dalam eksperimen ini adalah:

1. Semua instrumen yang digunakan dapat bekerja dengan baik sebagaimana mestinya
2. Kondisi lingkungan ideal dan tidak menganggu kualitas kerja dari alat instrumen yang digunakan sehingga hasil dari eksperimen dapat digunakan sebagai data

Bab II Dasar Teori

II.1 Internet of Things (IoT)

IoT adalah konsep teknologi yang memungkinkan objek-objek fisik terhubung ke internet dan saling berkomunikasi melalui jaringan komputer, serta dapat dikendalikan dan dipantau dari jarak jauh. Objek-objek tersebut dapat berupa perangkat elektronik seperti sensor, aktuator, perangkat medis, kendaraan, rumah pintar, dan peralatan industri. IoT terdiri dari dua elemen utama yaitu perangkat keras (hardware) dan perangkat lunak (software). Perangkat keras dapat berupa sensor, aktuator, mikrokontroler, dan perangkat komunikasi seperti Wi-Fi atau Bluetooth. Sedangkan perangkat lunak dapat berupa aplikasi, sistem operasi, dan platform untuk mengelola data.

IoT memungkinkan objek-objek fisik untuk mengumpulkan data dan berbagi informasi secara otomatis dan terus-menerus dengan sistem lainnya melalui jaringan internet. Data yang dikumpulkan oleh objek-objek ini kemudian dapat dianalisis dan digunakan untuk meningkatkan efisiensi, kinerja, dan kenyamanan di berbagai bidang. Selain itu, IoT juga mengandalkan teknologi-teknologi lain seperti big data, cloud computing, dan kecerdasan buatan (AI) untuk memproses dan menganalisis data secara cepat dan akurat. Teknologi-teknologi tersebut memungkinkan aplikasi dan sistem IoT untuk mengambil keputusan secara otomatis berdasarkan data yang diperoleh dari objek-objek fisik yang terhubung.

Dalam hal keamanan, IoT juga membutuhkan teknologi keamanan yang kuat untuk melindungi data dan jaringan dari serangan dan ancaman siber. Teknologi keamanan seperti enkripsi data, sertifikat digital, dan firewall sangat penting untuk menjaga keamanan dan privasi data di dalam jaringan IoT.

II.2 Message Queuing Telemetry Transport (MQTT)

MQTT (Message Queuing Telemetry Transport) adalah protokol komunikasi yang dirancang untuk pengiriman pesan dalam skenario IoT. MQTT awalnya dikembangkan oleh IBM dan kemudian menjadi protokol standar yang dikelola oleh organisasi OASIS (Organization for the Advancement of Structured Information Standards).

MQTT menggunakan model publish-subscribe, yang memungkinkan pengiriman pesan asinkron dari satu perangkat ke perangkat lain. Dalam model ini, ada tiga entitas utama yaitu

publisher, subscriber, dan broker. Publisher adalah perangkat yang mengirimkan pesan, subscriber adalah perangkat yang menerima pesan, dan broker adalah perantara yang mengirimkan pesan dari publisher ke subscriber.

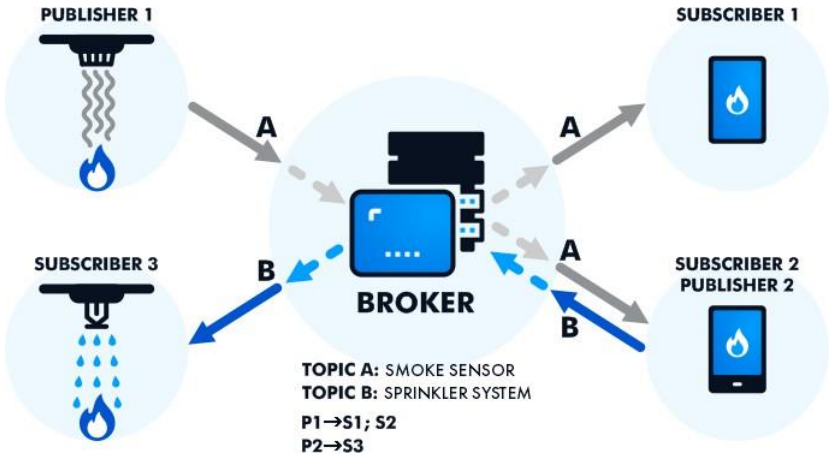
MQTT memiliki topik sebagai cara untuk mengatur pesan yang dikirimkan antara publisher dan subscriber. Topik MQTT adalah string yang digunakan sebagai alamat pesan, yang dapat dibagi menjadi beberapa level dengan pemisah / (garis miring). Sebagai contoh, topik "suhu/ruangan1" akan membawa pesan terkait suhu di ruangan1.

MQTT menggunakan tiga kualitas layanan pesan yang berbeda yaitu:

- QoS 0: Pesan dikirim satu kali dan tanpa konfirmasi pengiriman. Jika pesan tidak berhasil dikirim, maka pesan tersebut akan hilang.
- QoS 1: Pesan dikirim dengan setidaknya satu konfirmasi pengiriman. Jika pesan tidak diterima oleh broker, maka pesan akan dikirim kembali hingga diterima.
- QoS 2: Pesan dikirim dengan konfirmasi pengiriman yang dijamin tepat satu kali. Pesan akan dikirim kembali jika pesan tidak diterima oleh broker.

MQTT dapat diimplementasikan pada berbagai bahasa pemrograman dan platform. MQTT juga mendukung beberapa fitur keamanan seperti autentikasi, enkripsi, dan autorisasi.

Dalam aplikasi IoT, MQTT sering digunakan untuk menghubungkan perangkat sensor dan aktuator ke platform IoT seperti Raspberry Pi, AWS IoT, atau Google Cloud IoT. Dengan menggunakan MQTT, perangkat-perangkat ini dapat saling berkomunikasi dan berinteraksi secara efektif dan efisien.

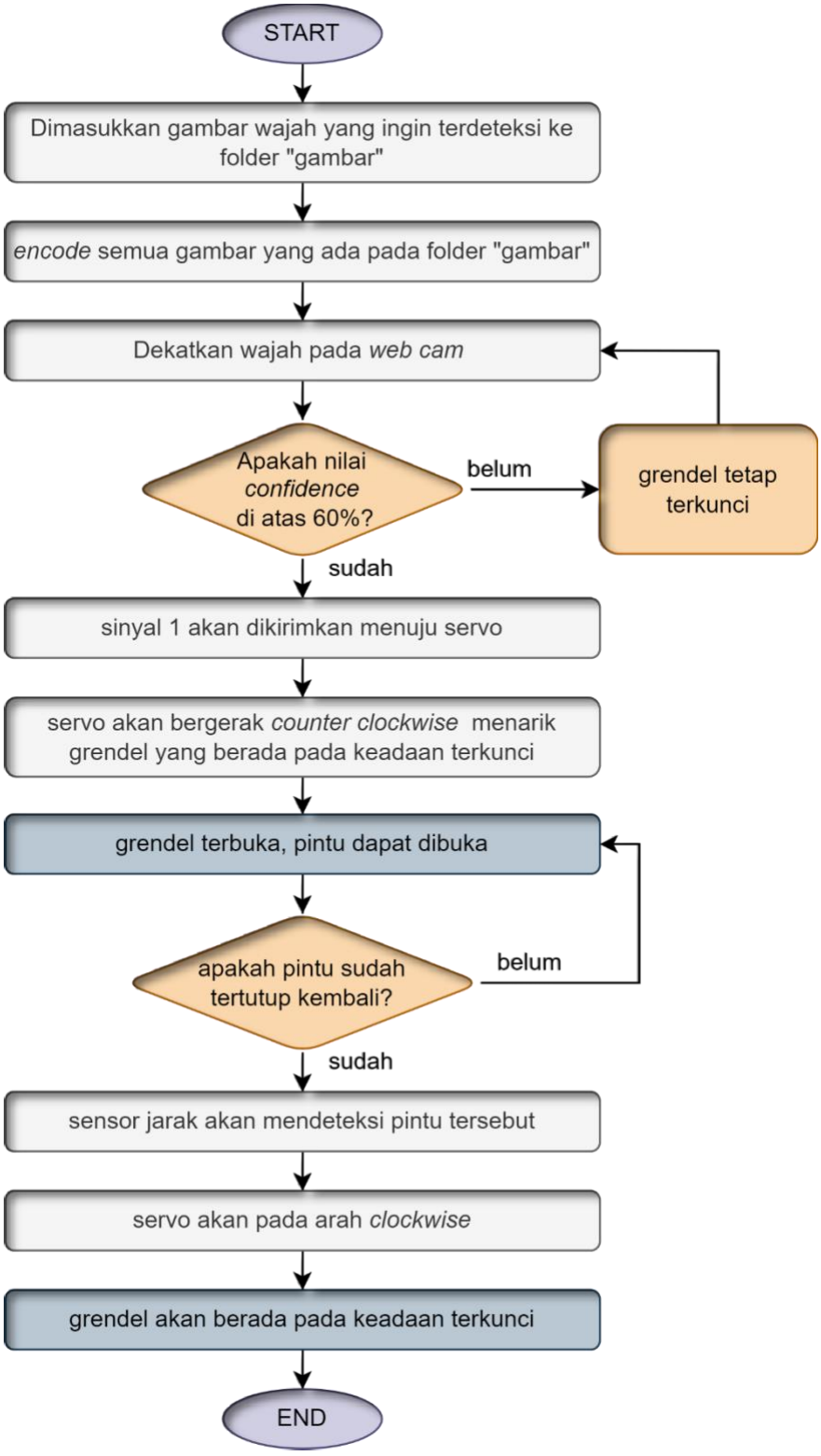


Gambar II.2.1 MQTT Workflow

Bab III Metode Eksperimen

III.1 Metode Eksperimen

Ekperimen yang kami lakukan dirangkum dalam diagram alir atau flowchart berikut.



Gambar III.1 *Flowchart* cara kerja alat

III.1.1 Membuat Program Python

III.1.1.1 Encoding Wajah

Pertama, download library yang dibutuhkan untuk melakukan proses encoding. Sehingga library tersebut dapat dipanggil dalam awal kode proses encoding

```
import cv2
import face_recognition
import os
import pickle
```

Setelah itu, masukkan gambar wajah orang yang ingin dideteksi dengan memasukkannya ke dalam folder “gambar” dengan ID atau filename yang berbeda beda tiap gambar . Lalu, kita ubah semua gambar pada folder “gambar” menjadi list dan kita ubah file gambar tersebut menjadi array dengan fungsi cv2.imread() dan memisahkan ID dan format gambar

```
pathList = os.listdir(folderPath)
pathList = [file for file in pathList if not file.startswith('.')]
imgList = []
ID = []
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    ID.append(os.path.splitext(path)[0])
```

Setelah itu, kita Buat suatu fungsi untuk melakukan encoding wajah dengan menggunakan fungsi ‘face_encodings’. Gambar yang telah dimasukkan ke dalam variabel array akan dimasukkan ke dalam fungsi yang telah dibuat sebelumnya Setalah hasil encoding didapatkan, buat file yang berisikan data encoding wajah untuk nanti dijadikan sebagai perbandingan dengan wajah dari kamera secara live.

```
def cariEncoder(imgList):
    encodeList = []
    for img in imgList:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    return encodeList
```

III.1.1.2 Pengenalan Wajah dan Transfer Data

Download library yang dibutuhkan untuk melakukan proses pengenalan wajah agar library tersebut dapat digunakan dan dipanggil diawal kode.

```
import os
import pickle
import face_recognition
import paho.mqtt.client as mqtt
import time
import cv2
```

Set kamera secara live menggunakan fungsi ‘cv2.VideoCapture(<posisi kamera>)’.Lalu, buat fungsi ‘on_message’ untuk menerima data dari Arduino dan buka file teks berisi suatu matriks hasil dari encoding wajah.

```
def on_message(client, userdata, msg):
    global check
    topic = msg.topic
    message = str(msg.payload.decode("utf-8"))
    if message == '1':
        check = True
    print("Received message: ", message, "on topic", topic)
    print("pintu tertutup!!")
```

Setelah itu, Set parameter yang dibutuhkan untuk menggunakan mqtt sebagai media untuk mentransfer data. Buat sebuah program untuk menyatakan bahwa apakah kamera kita menangkap wajah seseorang atau tidak dengan menggunakan fungsi face locations(). Lakukan matching wajah antara wajah hasil encoding dengan wajah secara live menggunakan fungsi ‘face_distance’. Fungsi face distance ini akan menghitung standar deviasi antara array wajah yang ada pada gambar pada folder “gambar” dan array wajah yang ada pada kamera

```
while True:
    data = 0
    success, cam = camera.read()
    camS = cv2.resize(cam, (0,0), None, 0.25, 0.25)
    camS = cv2.cvtColor(cam, cv2.COLOR_BGR2RGB)
    faceCurFrame = face_recognition.face_locations(camS)
    encodeCurFrame = face_recognition.face_encodings(camS, faceCurFrame)
```

```
if len(faceCurFrame) > 0:
    if check:
        for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
            faceDis = face_recognition.face_distance(encodeList, encodeFace)
            checkFace = face_recognition.compare_faces(encodeList, encodeFace)
```

Untuk menciptakan level *confidence* yang sesuai agar tidak menimbulkan kesalahan dalam melakukan matching wajah, maka perlu diberikan nilai *threshold* atau suatu batas tertentu untuk menentukan apakah wajah seseorang secara live dengan hasil encoding itu merupakan wajah dari orang yang sama. Jika sama, maka program python akan mengirimkan data untuk memberikan perintah kepada Arduino untuk membuka kunci pintu.

```
            matchIndices = faceDis <= thershold
            matches = any(matchIndices)
            print("checkFace", checkFace)
            print("faceDis", faceDis)
            print("matchIndices", matchIndices)
else:
    matches = False
if matches and check:
    data = 1
    print(data)
    client.publish("dajjal123", data)  # client.publish("topik publish", pesan)
    time.sleep(1)
    print("pintu terbuka!!")
else:
    data = 0
```

Setelah itu, pada kamera yang digunakan, akan kita tampilkan parameter *confidence* dan thresold agar proses pendeteksian wajah dapat lebih mudah dimengerti.

III.1.2 Membuat Program Arduino

III.1.2.1 Kalibrasi Stepper Motor

Pertama, hubungkan servo motor dengan pin pada ESP. Setup pin servo motor sebagai OUTPUT. Buat suatu program untuk memberikan perintah melalui serial monitor dengan sudut stepper motor sebagai input. Kemudian, kalibrasi stepper motor dengan memasukkan

input sudut positif (clockwise) dan negative (counter_clockwise) hingga mendapatkan sudut yang paling optimal.

III.1.2.2 Sistem Pintu Otomatis

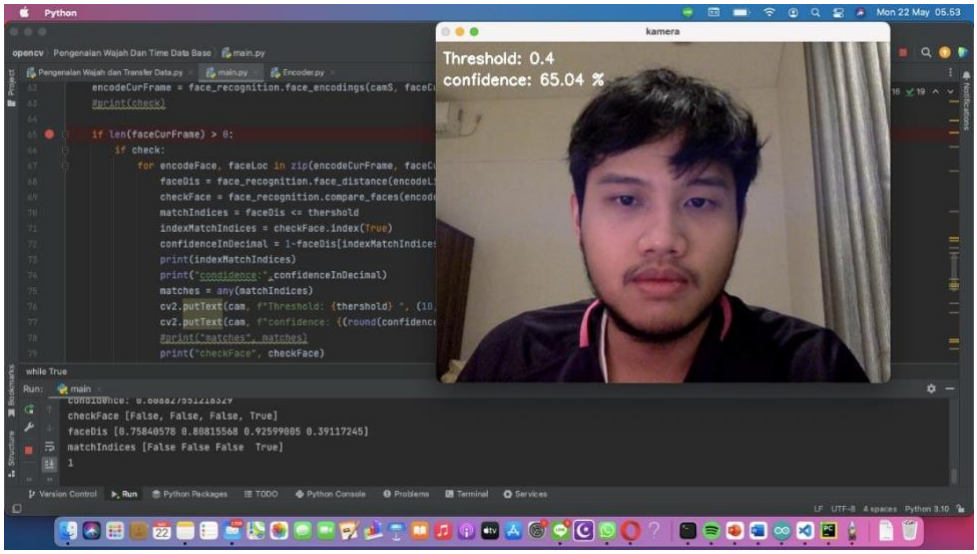
Pertama, membuat client mqtt untuk menerima data yang dikirimkan dari python. Tentukan pin untuk setiap komponen elektronik yang digunakan (stepper motor, sensor ultrasonic, LED) dan setup-nya. Buat suatu kondisi apabila sensor ultrasonic mencapai kurang dari 3 cm, maka stepper motor akan bergerak dan mengunci pintu. Selain itu, kondisi tersebut juga berfungsi untuk mentrasfer data ke python bahwa program pengenalan wajah akan diaktfikan kembali setelah pintu terkunci.

Bab IV Hasil dan Pembahasan

IV.1 Hasil Eksperimen

IV.1.1 Hasil dan Pembahasan

Hasil eksperimen menunjukkan bahwa teknik pengenalan wajah menggunakan Python libraries seperti OpenCV dan *face recognition* mampu mengenali wajah dengan cukup baik. Berikut dokumentasi dari proses pengenalan wajah.



Gambar IV.1 Pengambilan gambar wajah pada web cam

Pada gambar wajah yang telah dikenal, metode ini berhasil memberikan hasil yang akurat dengan jarak antara encoding wajah yang diambil dan encoding wajah yang telah dikenal relatif rendah (misalnya kurang dari 0.5). Namun, pada gambar wajah yang tidak dikenal, metode ini berhasil mengidentifikasi bahwa wajah tersebut tidak cocok dengan encoding wajah yang telah dikenal dengan hasil jarak yang cukup tinggi (misalnya lebih dari 0.5). Kinerja metode ini juga bergantung pada kualitas gambar wajah yang diambil. Gambar wajah yang buram atau dengan cahaya yang tidak memadai dapat mengurangi akurasi pengenalan wajah. Berikut merupakan urutan dari sistem *face recognition* atau pengenalan wajah yang kami lakukan

1. Persiapan Dataset:
   - Membuat dataset yang berisi gambar-gambar wajah individu yang akan dikenali.
   - Gambar-gambar ini dapat diperoleh dengan menggunakan kamera atau webcam.
2. Ekstraksi Fitur (Face Encoding):
   - Menggunakan OpenCV untuk mendeteksi wajah dalam gambar yang telah diambil.

- Setelah wajah terdeteksi, menggunakan fungsi face_encodings dari library *face recognition* untuk menghasilkan vektor fitur (face encoding) dari wajah tersebut.
- Fungsi face_encodings akan menerapkan model deep learning yang telah dilatih sebelumnya untuk mengekstraksi fitur-fitur unik dari wajah.

3. Pembuatan Database Wajah yang Dikenal:
    - Menyimpan vektor fitur (face encoding) yang dihasilkan dari langkah sebelumnya ke dalam database sebagai data referensi.
    - Database ini akan berisi vektor fitur dari wajah-wajah yang dikenal dan akan digunakan untuk membandingkan dengan wajah yang akan diuji nanti.
4. Deteksi dan Pengenalan Wajah:
    - Menggunakan kamera atau webcam untuk mengambil gambar wajah individu saat runtime.
    - Menggunakan OpenCV untuk mendeteksi wajah dalam gambar yang diambil.
    - Setelah wajah terdeteksi, menggunakan fungsi face_encodings untuk menghasilkan vektor fitur dari wajah tersebut.
5. Perbandingan dan Verifikasi:
    - Membandingkan vektor fitur yang dihasilkan dari wajah yang baru saja terdeteksi dengan vektor fitur yang ada dalam database wajah yang dikenal.
    - Menggunakan metode perbandingan seperti Euclidean distance atau cosine similarity untuk mengukur sejauh mana kedua vektor fitur mendekati atau cocok satu sama lain.
    - Jika kedua vektor fitur memiliki kesamaan yang mencukupi, maka sistem akan mengenali wajah tersebut sebagai wajah yang dikenal.
6. Tindakan Pintu Pintar/smartdoor:
    - Jika wajah yang diambil dikenali sebagai wajah yang dikenal, sistem pintu pintar dapat memberikan akses yang sesuai, seperti membuka pintu secara otomatis.
    - Jika wajah yang diambil tidak dikenali atau tidak cocok dengan wajah yang dikenal, sistem pintu pintar akan menolak akses dan mungkin memberikan peringatan atau melaporkan kejadian yang mencurigakan.

Pada langkah ekstraksi fitur menggunakan fungsi face_encodings, library *face recognition* akan menggunakan model deep learning yang telah dilatih sebelumnya, seperti model ResNet atau model pendeteksi wajah yang telah dilatih pada jutaan gambar wajah. Model tersebut akan

menghasilkan representasi numerik dari fitur-fitur wajah yang sangat diskriminatif. Kemudian, vektor fitur ini digunakan untuk membandingkan wajah-wajah yang dikenal dengan wajah yang baru saja terdeteksi.

IV.1.2 Analisis

Setelah mengimplementasikan *face recognition* menggunakan OpenCV dan *face recognition* pada smart door, kami melakukan analisis terhadap sistem yang telah dibangun. Berikut adalah beberapa hasil analisis yang kami temukan. Pertama, berangkat dari kelebihan. Ditemukan beberapa kelebihan dari implementasi ini diantaranya ialah kemudahan dari implementasi itu sendiri, menyangkut Penggunaan Python libraries seperti OpenCV dan *face recognition* mempermudah proses implementasi *face recognition* pada smart door. Kedua, akurasi pengenalan yang tinggi, dengan menggunakan model *deep learning* dan vektor fitur wajah yang unik, sistem mampu mengenali wajah dengan tingkat akurasi yang tinggi. Terakhir, kecepatan respons, sistem mampu mendeteksi dan mengenali wajah dalam waktu yang relatif cepat, sehingga memungkinkan akses yang segera. Adapun terdapat juga kelemahan-kelemahan yang kami temukan selama prosesnya, diantaranya ialah keterbatasan dataset, kinerja sistem sangat bergantung pada kualitas dataset yang digunakan untuk melatih model deep learning. Dataset yang terbatas dapat mempengaruhi akurasi pengenalan. Selain itu, ketergantungan pada pencahayaan dan posisi wajah, pencahayaan yang buruk atau posisi wajah yang tidak ideal dapat mempengaruhi kemampuan sistem dalam mendeteksi dan mengenali wajah.

Bab V Kesimpulan dan Saran

V.1 Kesimpulan

Berdasarkan eksperimen, implementasi pintu pintar dengan menggunakan face recognition melalui Python libraries seperti OpenCV dan face recognition menawarkan solusi yang efektif dalam mengendalikan akses dan meningkatkan keamanan. Dengan tingkat akurasi pengenalan yang tinggi dan kemudahan implementasi, sistem ini dapat menjadi alternatif yang menarik untuk pintu pintar konvensional.

Meskipun demikian, diperlukan perhatian ekstra terhadap kualitas dataset, pencahayaan, dan posisi wajah agar sistem dapat berfungsi secara optimal. Selain itu, penting juga untuk mempertimbangkan aspek privasi dan keamanan data wajah pengguna dalam pengembangan dan implementasi sistem ini. Dengan melakukan peningkatan yang sesuai, penggunaan face recognition dalam smart door dapat menjadi solusi yang lebih aman, efisien, dan nyaman untuk pengendalian akses pintu.

V.2 Saran

Berdasarkan analisis yang telah dilakukan, terdapat beberapa saran dan rekomendasi yang dapat kami berikan untuk meningkatkan kinerja sistem pintu pintar berbasis face recognition:

- Perluasan Dataset: Mengumpulkan dataset yang lebih luas dan representatif akan membantu meningkatkan akurasi pengenalan wajah. Dataset yang mencakup variasi pencahayaan, sudut pandang, dan ekspresi wajah akan memperkaya pelatihan model deep learning.
- Pemrosesan Gambar Lebih Lanjut: Menerapkan teknik pemrosesan gambar seperti deteksi dan koreksi pencahayaan, peningkatan kualitas gambar, atau pengenalan pose wajah dapat membantu mengatasi keterbatasan pencahayaan dan posisi wajah yang tidak ideal.
- Integrasi Teknologi Tambahan: Menggabungkan face recognition dengan teknologi lain seperti deteksi gerakan atau sensor kehadiran dapat memperkuat sistem pintu pintar untuk keamanan yang lebih baik.
- Pengujian Lanjutan: Melakukan pengujian yang lebih mendalam dengan berbagai skenario penggunaan dan variasi kondisi lingkungan akan membantu mengidentifikasi kelemahan potensial dan mengoptimalkan kinerja sistem.

DAFTAR PUSTAKA

Bandyopadhyay, D., & Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. Wireless Personal Communications, 58(1), 49-69.

Ashton, K. (2009). That 'Internet of Things' thing. RFiD Journal, 22(7), 97-114.

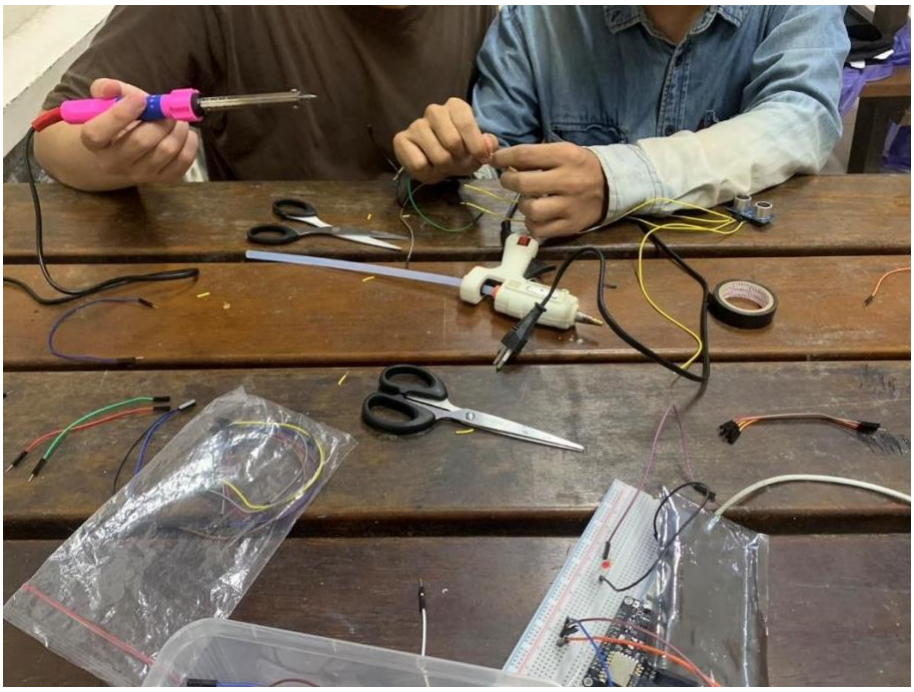
Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey of topics and trends. Information Systems Frontiers, 17(2), 261-274.

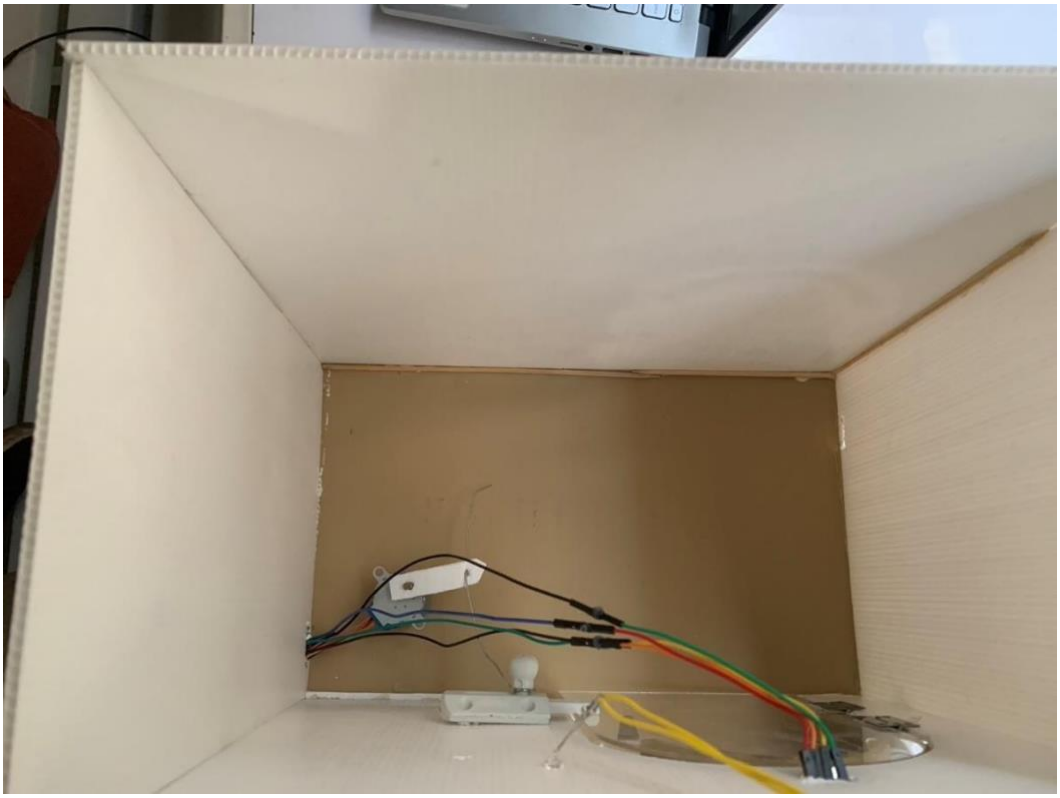
Atzori, L., Iera, A., & Morabito, G. (2010). From "smart objects" to "social objects": The next evolutionary step of the internet of things. IEEE Communications Magazine, 48(2), 150-156.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

LAMPIRAN

1. Dokumentasi





2. Kode python dan Arduino

2.1. Kode Python untuk encoding wajah

```
import cv2
import face_recognition
import os
import pickle

folderPath = 'gambar'


# membuat semua file gambar di dalam folder menjadi sebuah list
pathList = os.listdir(folderPath)
pathList = [file for file in pathList if not file.startswith('.')]
imgList = []
ID = []

# mengubah file gambar menjadi gambar asli dengan fungsi cv2.imread() dan memisahkan ID dan format gambar
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    ID.append(os.path.splitext(path)[0])

# fungsi untuk mengencode wajah menjadi sebuah informasi beruba matriks
def cariEncoder(imgList):
    encodeList = []
    for img in imgList:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    return encodeList

# Menentukan pattern wajah menggunakan fungsi cariEncoder
print("mulai encoding.... ")
encodeList = cariEncoder(imgList)
encodeListID = [encodeList, ID]
print("encoding selesai.")

# Menyimpan data pattern ke dalam file teks bernama encodeFile.p dalam bentuk biner
```

```
file = open("encodeFile.p", "wb")
pickle.dump(encodeListID, file)
file.close()
print("file telah tersimpan")
```

2.2. Kode Python untuk Pengenalan Wajah dan Transfer Data

```
import os
import pickle
import face_recognition
import paho.mqtt.client as mqtt
import time
import cv2

# set kamera
camera = cv2.VideoCapture(0)
camera.set(3, 640)
camera.set(4, 480)

# paramater untuk menampilkan teks pada kamera
font = cv2.FONT_HERSHEY_SIMPLEX
font_scale = 0.7
font_color = (255, 255, 255)  # White color
thickness = 2

def on_message(client, userdata, msg):
    global check
    topic = msg.topic
    message = str(msg.payload.decode("utf-8"))
    if message == '1':
        check = True
    print("Received message: ", message, "on topic", topic)
    print("pintu tertutup!!")

# mqtt
client = mqtt.Client()
client.on_message = on_message

mqttBroker = "broker.mqtt-dashboard.com"
port = 1883
```

```
client.connect(mqttBroker, port)
client.subscribe("naufal123")


# variabel
check = True
matches = []
thershold = 0.4


print("loading file encode ...")
file = open("EncodeFile.p", 'rb')
encodeListID = pickle.load(file)
file.close()
encodeList, ID = encodeListID
print("file encode loaded")


while True:
    data = 0
    success, cam = camera.read()

    camS = cv2.resize(cam, (0,0), None, 0.25, 0.25)
    camS = cv2.cvtColor(cam, cv2.COLOR_BGR2RGB)


    faceCurFrame = face_recognition.face_locations(camS)
    encodeCurFrame = face_recognition.face_encodings(camS, faceCurFrame)
    #print(check)

    if len(faceCurFrame) > 0:
        if check:
            for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
                faceDis = face_recognition.face_distance(encodeList, encodeFace)
                checkFace = face_recognition.compare_faces(encodeList,
encodeFace)
                matchIndices = faceDis <= thershold
                indexMatchIndices = checkFace.index(True)
                confidence InDecimal = 1-faceDis[indexMatchIndices]
                print(indexMatchIndices)
                print("condidence:",confidence InDecimal)
                matches = any(matchIndices)
```

```
            cv2.putText(cam, f"Threshold: {thershold} ", (10, 30), font, font_scale,
 font_color, thickness)
            cv2.putText(cam, f"confidence : {(round(confidence InDecimal*100,
 2))} %", (10, 60), font, font_scale, font_color, thickness)
            #print("matches", matches)
            print("checkFace", checkFace)
            print("faceDis", faceDis)
            print("matchIndices", matchIndices)


    else:
        matches = False


    if matches and check:
        data  =  1
        print(data)
        time.sleep(2)
        client.publish("dajjal123", data) # client.publish("topik publish", pesan)
        print("pintu terbuka!!")

    else:
        data = 0


    cv2.imshow('kamera', cam)
    cv2.waitKey(1)

    # memproses pesan mqtt
    client.loop()
```

2.3. Kode Arduino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <NewPing.h>


// Update these with values suitable for your network.

const char* ssid = "DODOT";
const char* password = "210310PH";
```

```
const char* mqtt_server = "broker.mqtt-dashboard.com";

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
int Trigger_pin = D1;
int Echo_pin = D2;
int IN_1 = D5;
int IN_2 = D6;
int IN_3 = D7;
int IN_4 = D8;
int delaytime = 2;
int j = 0;
int LED = D4;
int check = 0;

// Definisikan jarak maksimal yang ingin diukur (dalam
cm) #define MAX_DISTANCE 200

// Inisialisasi objek ultrasonik
NewPing sonar(Trigger_pin, Echo_pin, MAX_DISTANCE);


void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

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

  randomSeed(micros());

  Serial.println("");
```

```
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int
length) {
  Serial.print("Message arrived [");
  Serial.print(topic); Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first
character


  if ((char)payload[0] == '1' & j < 180 & check == 0) {
    for (int i = 0; i<180; i++) {
      clockwise(); j++;
      Serial.println(j);
    }
    digitalWrite(LED, LOW);
    check = 1;


  }
}


void reconnect() {
    // Loop until we're reconnected while
(!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-"; clientId +=
String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {

    Serial.println("connected");
    // Once connected, publish an announcement...
    client.publish("naufal123", msg);
```

```
      // ... and resubscribe
      client.subscribe("dajjal123");
      } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying delay(5000);
      }
    }
}

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);  // Initialize the
BUILTIN_LED pin as an output
  pinMode(LED, OUTPUT);
  pinMode(IN_1,OUTPUT);
  pinMode(IN_2,OUTPUT);
  pinMode(IN_3,OUTPUT);
  pinMode(IN_4,OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) { reconnect();
  }
  client.loop();

  unsigned long now = millis(); delay(50);
  if (now - lastMsg > 2000) {
    lastMsg = now;
  // Mengukur jarak dengan sensor ultrasonik
     unsigned int distance = sonar.ping_cm();
     Serial.print("distance: ");

  Serial.println( distance);
  if (check == 1) {

    if (distance <= 3) {
      delay(3000);
      String msg = "1";
```

```
        check = 0;
        for (int i = 0; i<180; i++) {
          counter_clockwise();
          j--;
        }

        digitalWrite(LED, HIGH);
        Serial.print("Publish message: ");
        erial.println(msg);
        client.publish("naufal123", msg.c_str());
      }

    }

  }
}

void clockwise() {
  step1();
  delay(delaytime);
  step2();
  delay(delaytime);
  step3();
  delay(delaytime);
  step4(); delay(delaytime);
}

void counter_clockwise(){
  step4();
  delay(delaytime);
  step3();
  delay(delaytime);
  step2();
  delay(delaytime);
  step1();
  delay(delaytime);

}

void step1() {
  digitalWrite(IN_1, HIGH);
  digitalWrite(IN_2, HIGH);
  digitalWrite(IN_3, LOW);
  digitalWrite(IN_4, LOW);
```

```
}

void step2() {
  digitalWrite(IN_1, HIGH);
  digitalWrite(IN_2, LOW);
  digitalWrite(IN_3, LOW);
  digitalWrite(IN_4, HIGH);

}

void step3() {
  digitalWrite(IN_1, LOW);
  digitalWrite(IN_2, LOW);
  digitalWrite(IN_3, HIGH);
  digitalWrite(IN_4, HIGH);
}

void step4() {
  digitalWrite(IN_1, LOW);
  digitalWrite(IN_2, HIGH);
  digitalWrite(IN_3, HIGH);
  digitalWrite(IN_4, LOW);
}
```