DAFTAR PUSTAKA

Adjie. (2020). “Menangani Esp32-Cam Module Kamera Arduino”. Http://Indomaker.Com/2020/03/23/Menangani-Esp32-Cam-Module-Kamera-Arduino/. Diakses Pada 1 April 2022 Pukul 23.29.

Basabilik, P. A. A. P. (2021). *RANCANG BANGUN SISTEM PEMANTAU KEDATANGAN TAMU BERBASIS INTERNET OF THINGS (IOT)*. *9*(2), 110–116.

Dewi, N.H.L. Dkk. (2019). Prototype Smart Home Dengan ESP8266 Esp8266 Berbasis Iot. *Jurnal Ilmiah Teknik, 1*(2), 101–107. https://doi.org/10.56127/juit.v1i2.169.

Fuady, Samratul. Dkk. (2020). “Deteksi Objek Menggunakan Metode Single Shot Multibox Detector Pada Alat Bantu Tongkat Tunanetra Berbasis Kamera”. 3(2), 39–43. Https://Doi.Org/10.33087/Jepca.V3i2.38

Hui, J. (2018). SSD object detection: Single Shot MultiBox Detector for real-time processing. https://jonathan-hui.medium.com/ssd-objectdetection-single shot-multibox-detector-for-realtime-processing-9bd8deac0e06

Ibrahim. (2020). “Analisis Akurasi Pengenalan Wajah Menggunakan Algoritma Viola Jones Dan Modified Self Organizing Map”. Tesis. Universitas Sumatera Utara.

Immersa. (2014). "Pengenalan Mikrokontroler". Https://Www.Immersa Lab.Com/Pengenalan-Mikrokontroler.Htm. Diakses Pada 6 November 2021 Pukul 23.15

Jurnawi, Imam. (2020). “Implementasi Pengenalan Wajah Secara Real Time Untuk Sistem Absensi Menggunakan Metode Pembelajaran Deep Learning Dengan Pustaka Open Cv (Computer Vision)”. Skripsi. Universitas Sumatera Utara.

Kurniasih, W. https://www.gramedia.com/literasi/pengertian-server/ (diakses pada 19 Maret 2023 pukul 13.37 WITA)

Madoi, Y. P. (2018). “Rancang Bangun Alat Pengaman Rumah Menggunakan

Sensor Pir (Passive Infra Red) Berbasis Sms Gateway*"*. 77.

Manengal, V. D., Lumenta, A. S. M., & Rumagit, A. M. (2014). Perancangan Sistem Monitoring Mengajar Berbasis Mikrokontroler Atmega 8535. *Journal Teknik Elektro Dan Komputer*, *3*(1), 2301–8402.

Mursyid, D. R. A. (2020). "Monitoring Kualitas Air pada Tambak Udang Vaname di Takalar Berbasis Android". Skripsi. Universitas Hasanuddin.

Muslimin, Z., Wicaksono, M. A., Fadlurachman, M. F., & Ramli, I. (2019). Rancang Bangun Sistem Keamanan dan Pemantau Tamu pada Pintu Rumah Pintar Berbasis Raspberry Pi dan Chat Bot Telegram. *Jurnal Penelitian Enjiniring*, *23*(2), 121–128. https://doi.org/10.25042/jpe.112019.05

Ningsih, D. S. (2010). No Title. In *Skripsi*. https://digilib.uns.ac.id/dokumen/download/15199/MzAxMDY=/Prototype-wastafel-otomatis-berbasis-mikrokontroler-AT89S51-abstrak.pdf

Permana, A. Yudi. (2019) "Perancangan Sistem Informasi Penjualan Perumahan Menggunakan Metode Sdlc Pada Pt. Mandiri Land Prosperous Berbasis Mobile". Jurnal Teknologi Pelita Bangsa, Vol 10, No. 2, Pp. 153- 167.

Prihatmoko, C. R. (2021). "Pengembangan Teknologi Smart Hybrid Reader Untuk Sistem Smart Campus Unhas". Universitas Hasanuddin.

Qrimly, K.(2017). "Apa itu Mikrokontroler?". https://www.logicgates.id/blogs/news/apa-itu-mikrokontroler

Riyanto, P.G. (2021). " Mengenal Telegram, Aplikasi Chat Yang Dilirik Sebagai Pengganti *Whatsapp*". Https://Tekno.Kompas.Com/Read/2021/01/13/19150027/Mengenal-Telegram-Aplikasi-Chat-Yang-Dilirik-Sebagai-Pengganti-Whatsapp?Page=All. Diakses Pada 27 November 2021 Pukul 21.45

Setiawan, D., Dkk. (2022). "Implementasi Esp32-Cam Dan Blynk Pada Wifi Door Lock". *4307*(1), 159–164.

Sistem, R., Informatika, P. T., Teknologi, I., & Malang, A. (2021). *Perbandingan*

Metode Deteksi Wajah Menggunakan OpenCV Haar Cascade , OpenCV Single Shot Multibox Detector (SSD) dan DLib CNN. *1*(10), 609–614.

Sukusvieri, A. (2020). *IMPLEMENTASI METODE SINGLE SHOT DETECTOR (SSD) UNTUK PENGENALAN WAJAH IMPLEMENTASI METODE SINGLE SHOT DETECTOR (SSD)*.

Suryana, O., & Kuningan, U. (2018). *Server dan Web Server*. *August*.

Suprianto, Dodit. dkk (2013). "Sistem Pengenalan Wajah Secara Real-Time Dengan Adaboost, Eigenface Pca & Mysql". Jurnal EECCIS Vol. 7, No. 2.

Sutama, V. A., Wibowo, S. A., & Rahmania, R. (2020). "Investigasi Pengaruh Step Training Pada Metode Single Shot Multibox Detector Untuk Marker Dalam Teknologi Augmented Reality". Xii(1), 1–11.

Uno, A., Xyz, P. T., K, R. S., & Sembada, G. (2020). Jurnal E-Komtek (Elektro-Komputer-Teknik . "Perancangan Sistem Keamanan Menggunakan Solenoid Door Lock Berbasis". 4(1), 62–74.

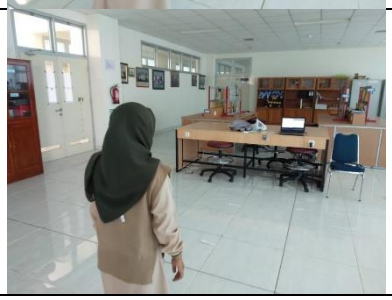
Virgusta, Y. D. (2020). "Rancang Bangun Alat Home Security Terintegrasi Bel Dan Alarm Menggunakan Teknologi Internet Of Things (Iot)". *Jurnal Ekonomi Volume 18, Nomor 1 Maret201*, *2*(1), 41–49

Zulkhaidi, T. C. A., & Maria, E. (2019). *Pengenalan Pola Bentuk Wajah dengan OpenCV*. *3*(2), 181–186.

LAMPIRAN

Lampiran 1

Proses pengujian jarak deteksi sensor

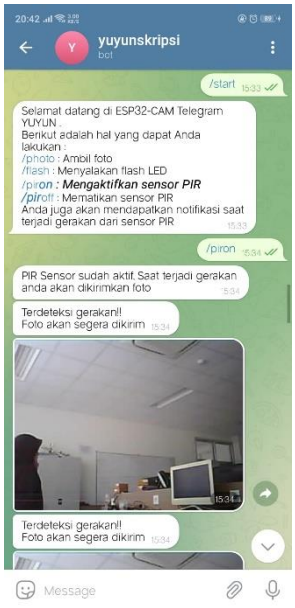
Jarak (meter)	Gambar
1	
2	
3	

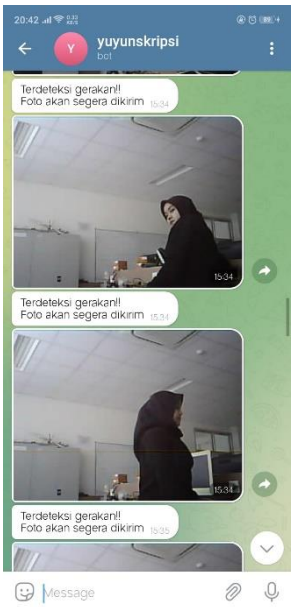
4		
5		
6		
7		
8		
9		

10	

Lampiran 2

Gambar hasil notifikasi ke telegram

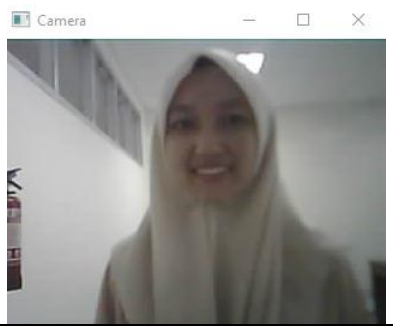


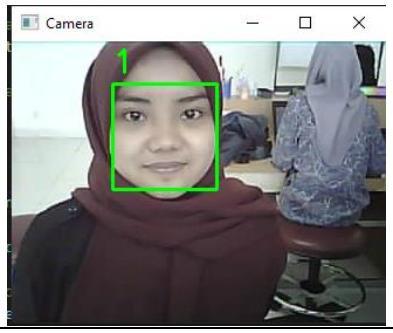


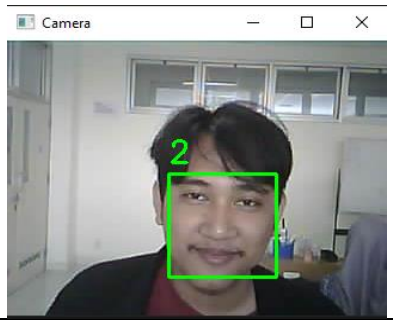
Lampiran 3

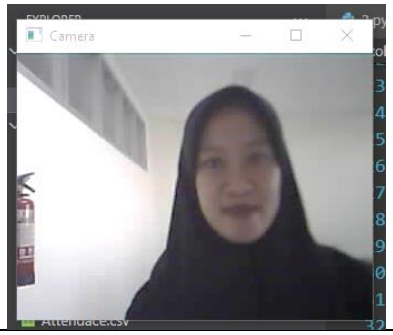
Jarak sistem mengenali wajah dengan nilai toleransi 0,4

Jarak 60 cm

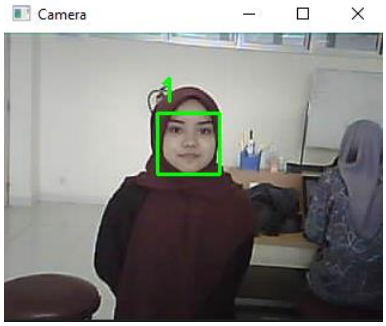


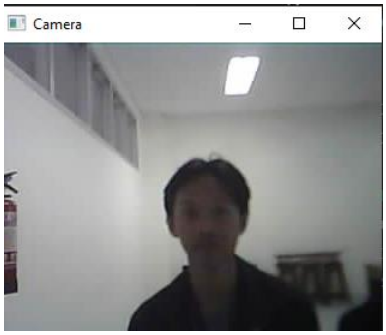


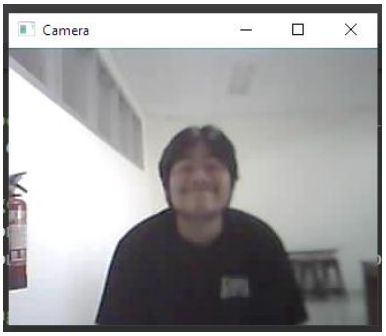


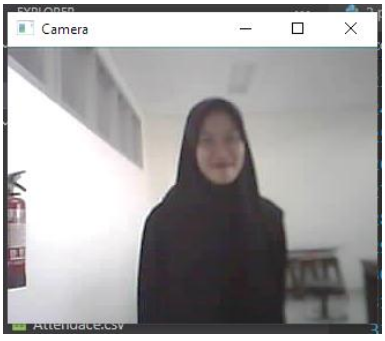


Jarak 120 cm

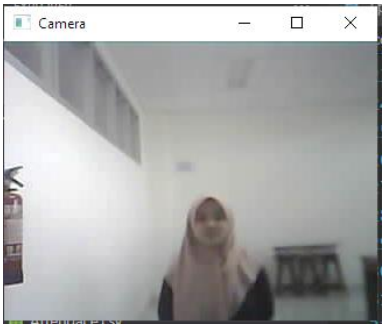


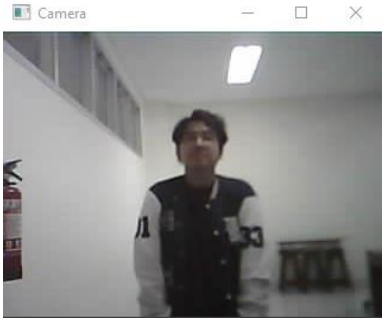


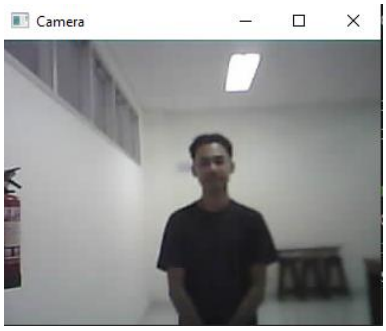


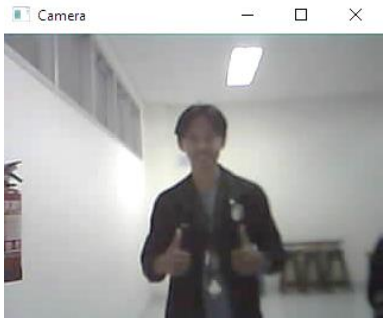


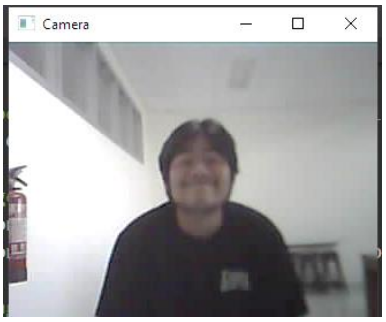
Jarak 180 cm

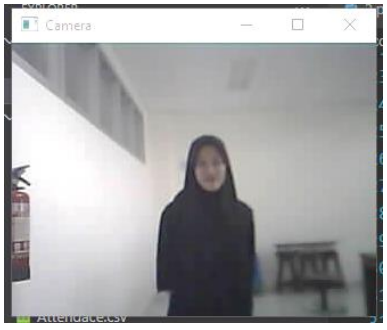


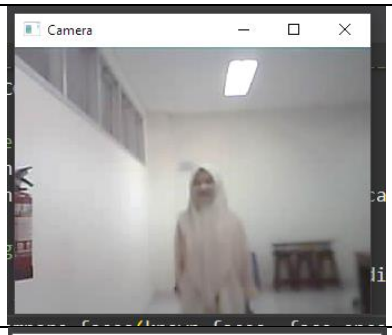
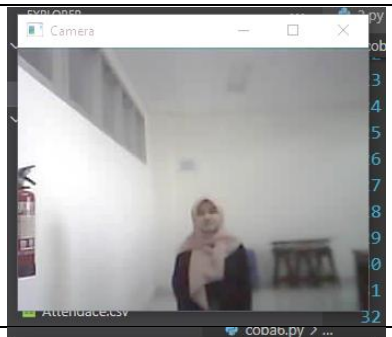
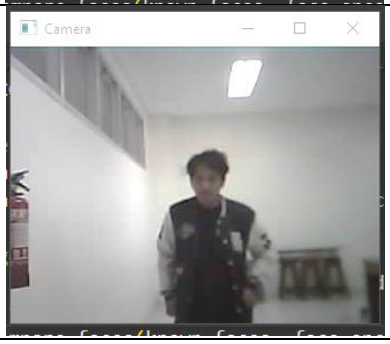
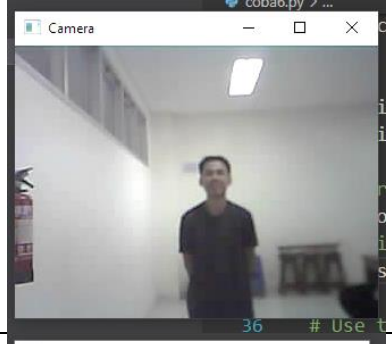
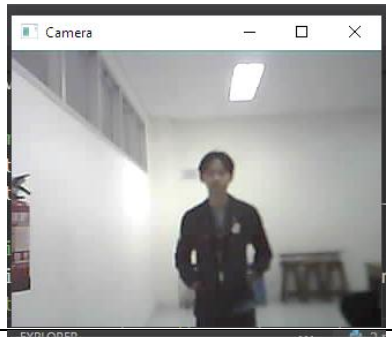
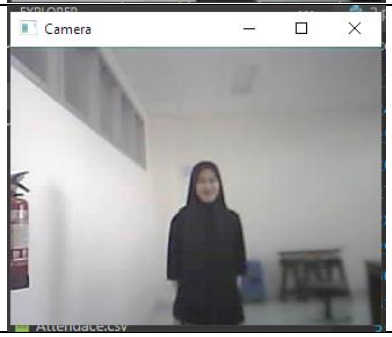








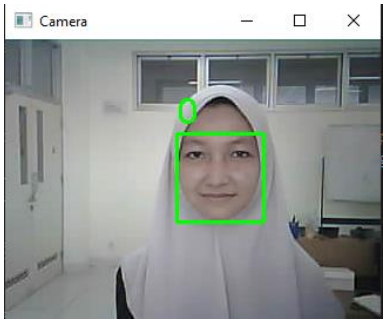


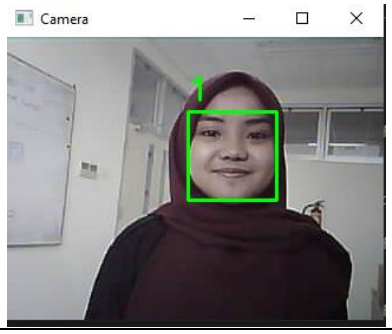
Jarak 240 cm	
	
	
	
	

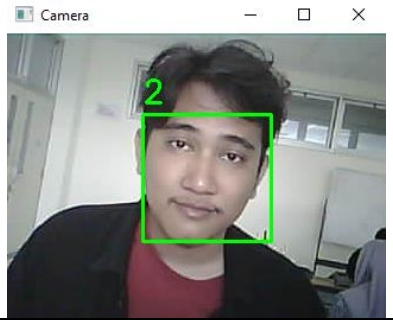
Lampiran 4

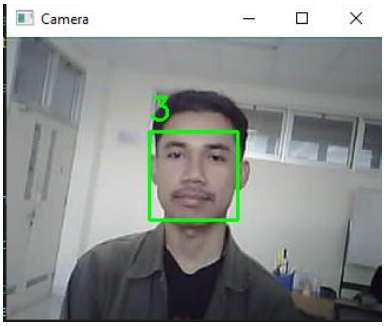
Jarak sistem mengenali wajah dengan nilai toleransi 0,5

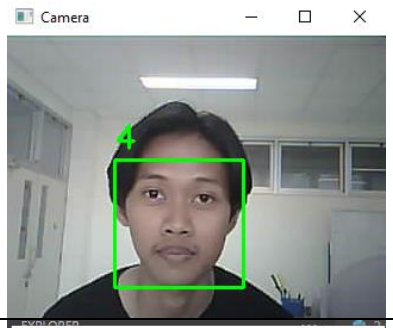
Jarak 60 cm

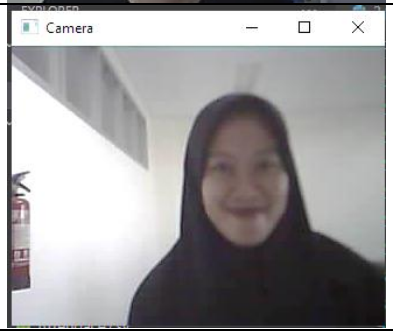




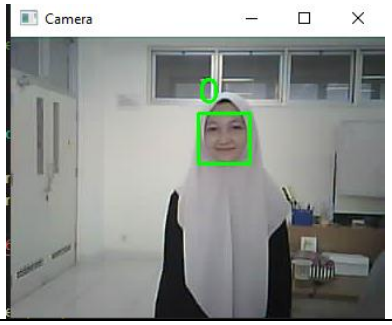


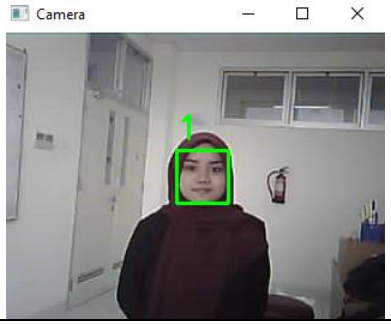


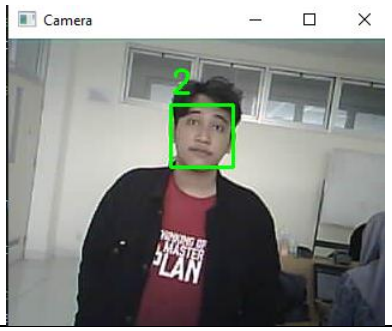


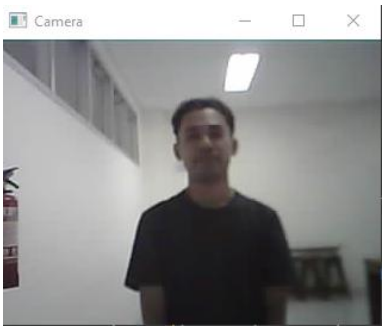


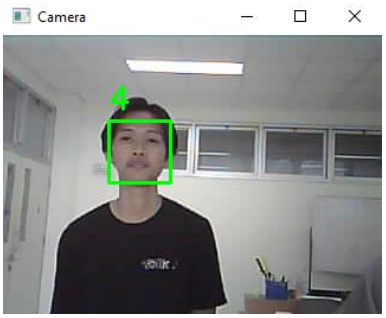
Jarak 120 cm

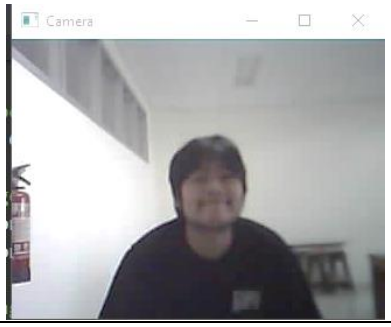


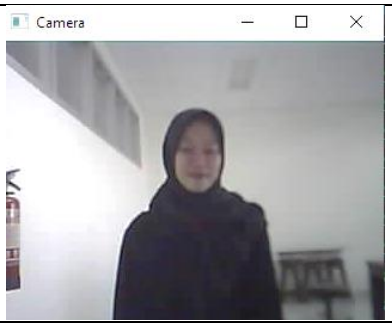




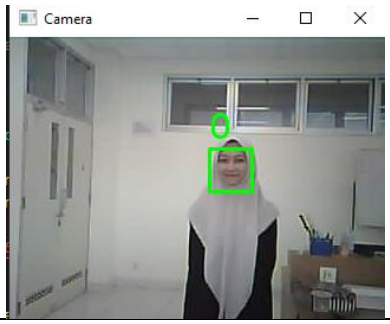


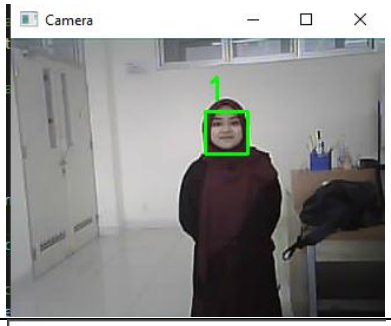


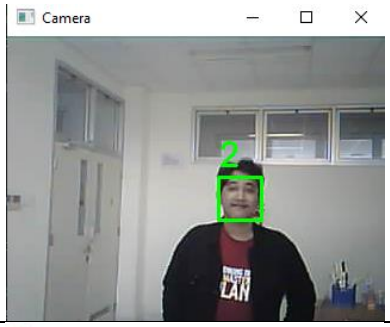


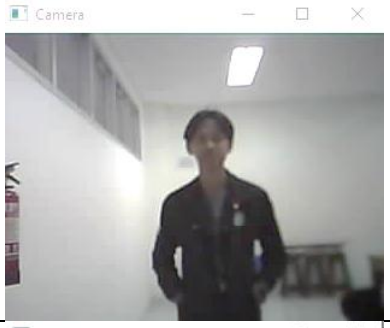


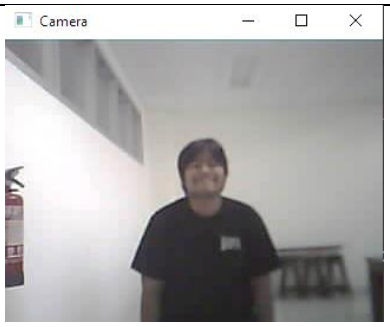
Jarak 180 cm

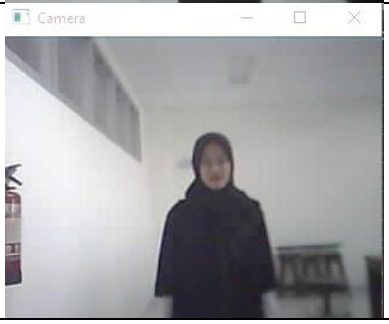




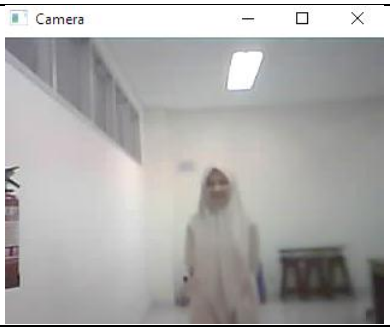


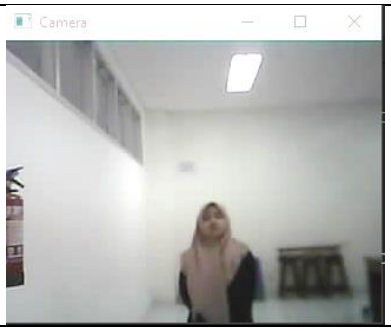


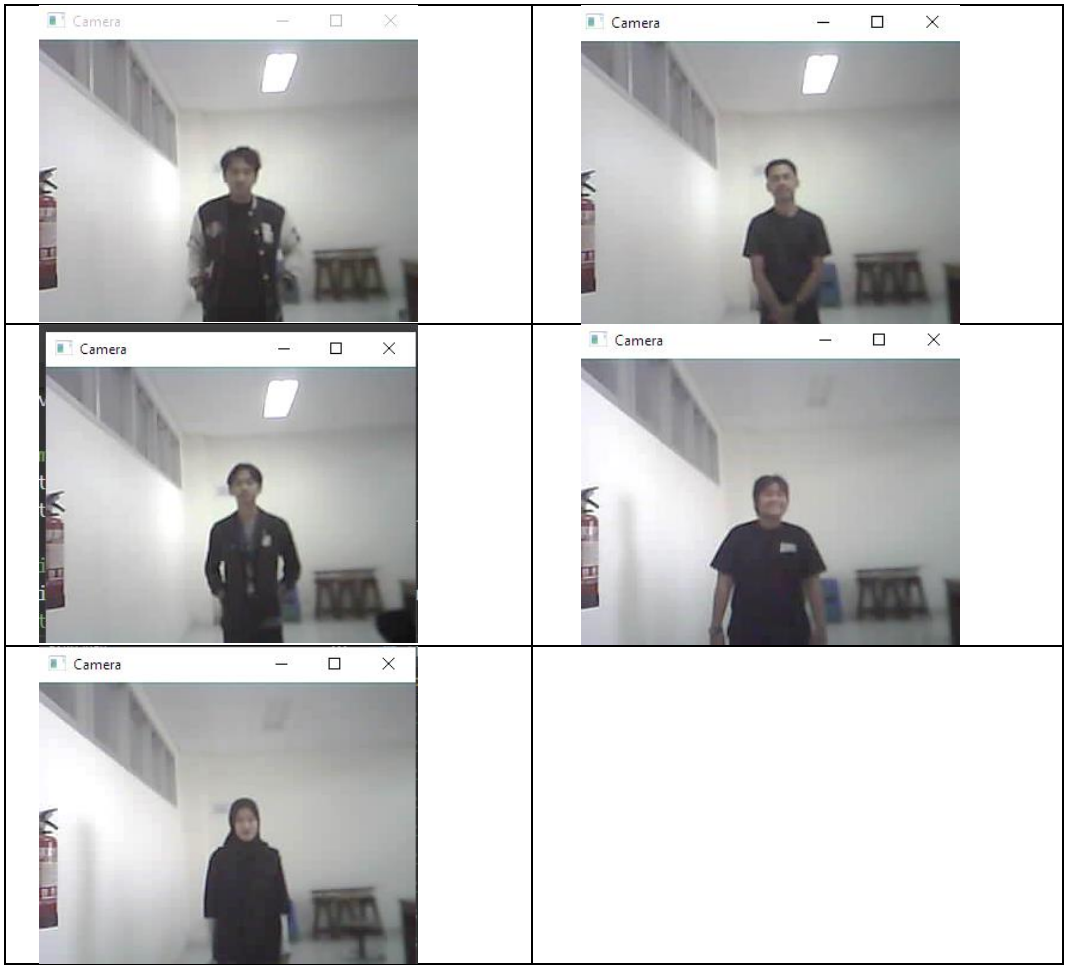




Jarak 240 cm



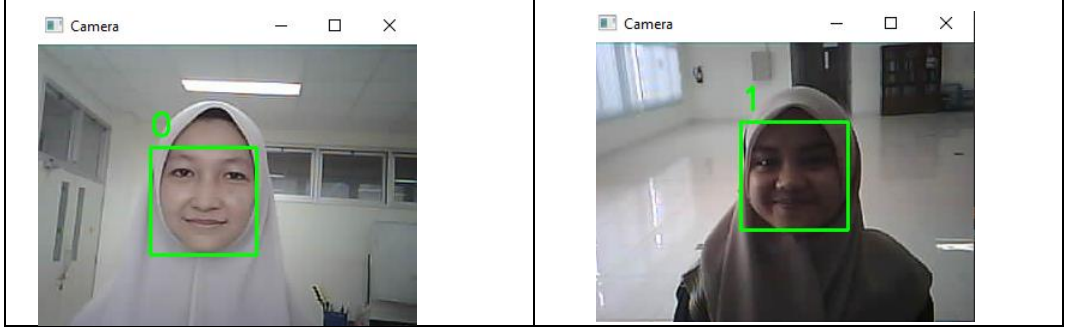


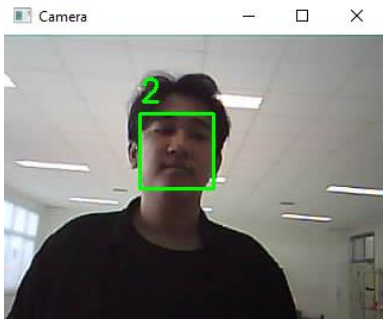


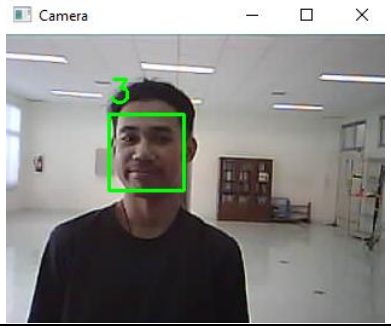
Lampiran 5

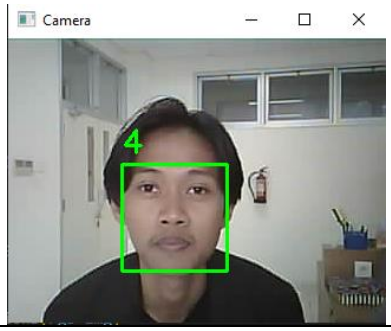
Jarak sistem mengenali wajah dengan nilai toleransi 0,6

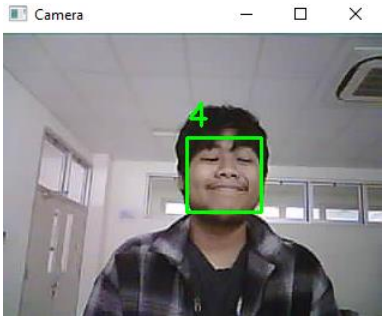
Jarak 60 cm

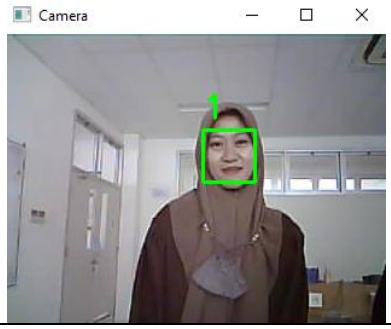



Camera
2

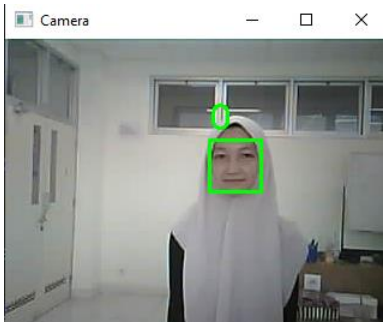

Camera
3

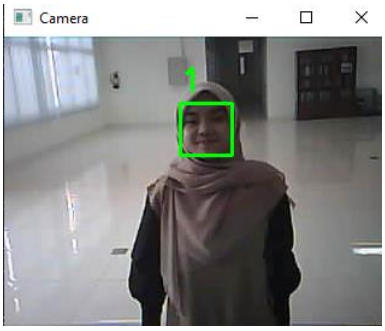

Camera
4

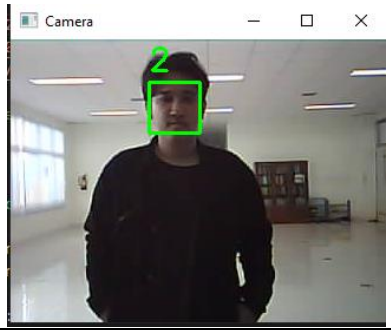

Camera
4

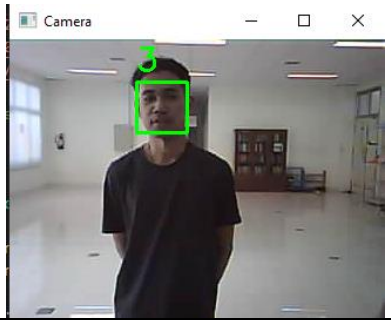

Camera
1

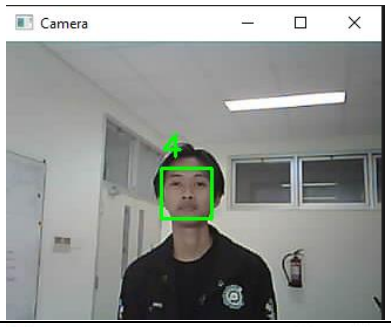
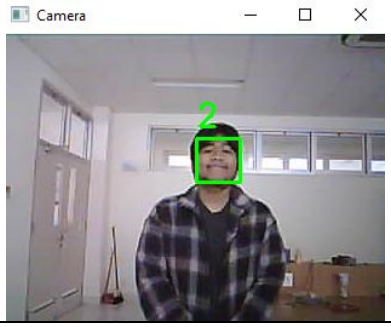
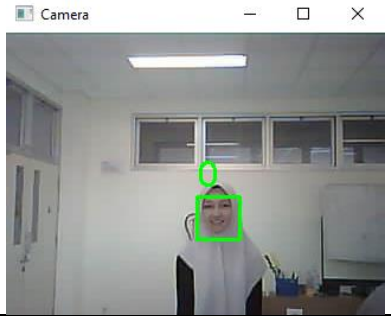
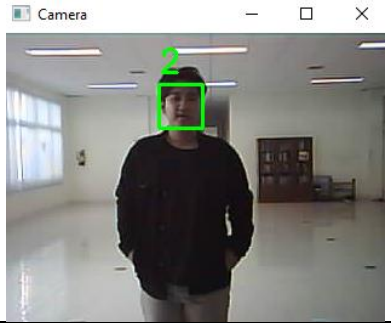
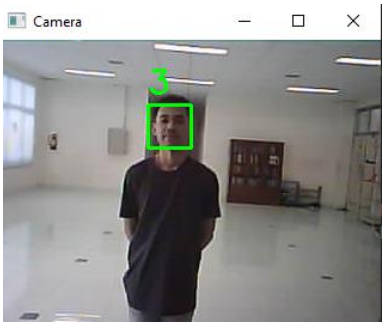
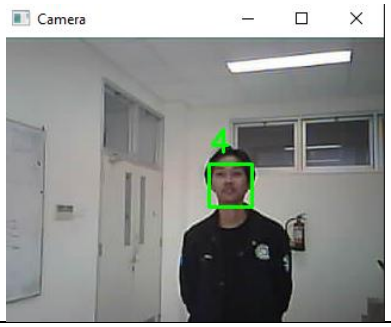
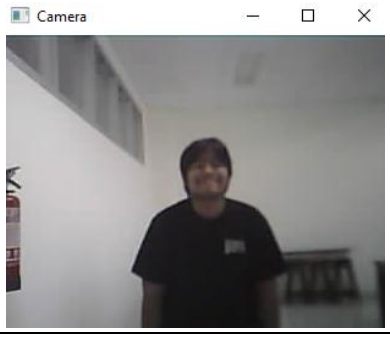
Jarak 120 cm

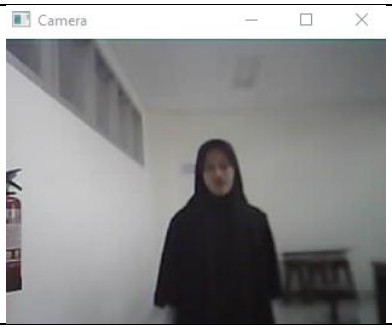

Camera
0


Camera
1

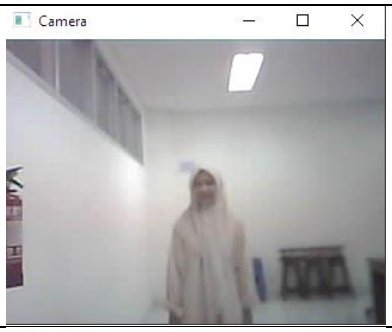

Camera
2

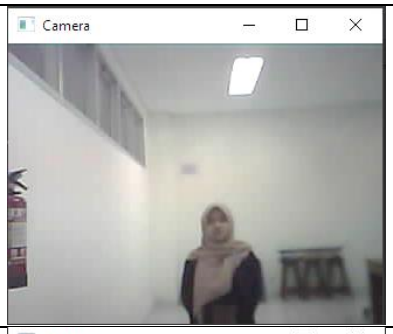

Camera
3

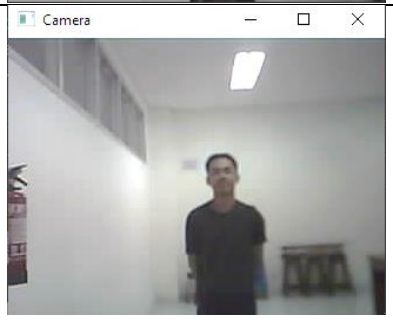
<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td></td></tr>
<tr><td>Jarak 180 cm</td><td></td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

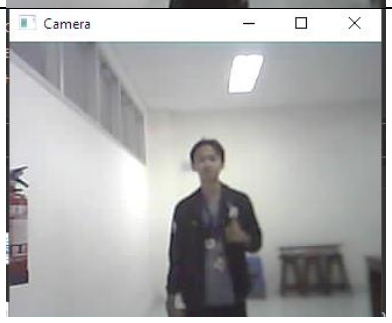


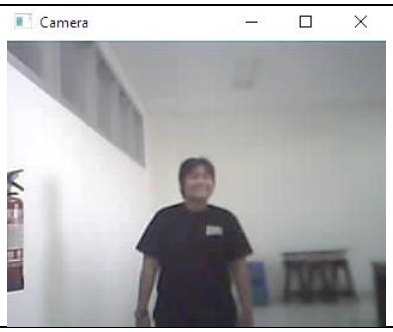
Jarak 240 cm

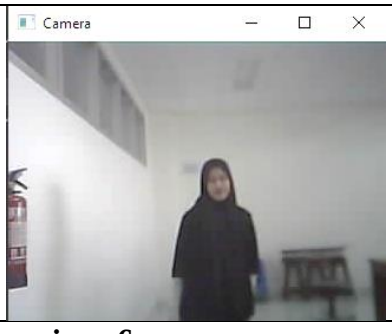












Lampiran 6

1. **Kode program untuk modul ESP32-CAM**

```
#include "esp_camera.h"
```

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <Wire.h>

#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

const char* ssid = "oon-nee";
const char* password = "lagal1g0";
String chatId = "939746173";
String BOTtoken = "5634269850:AAG9VAWRkVA2NTD7069UwavT-
gOUZNNTti4";
bool sendPhoto = false;

WiFiClientSecure clientTCP;

UniversalTelegramBot bot(BOTtoken, clientTCP);
#define LED 14
#define PIR 2
bool LEDState = false;
bool flag = 0;
long prevMillis=0;
int interval = 5000;

int botRequestDelay = 1000;   // mean time between scan messages
long lastTimeBotRan;     // last time messages' scan has been done

void startCameraServer();
void handleNewMessages(int numNewMessages);
String sendPhotoTelegram();

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(PIR, INPUT);
  digitalWrite(PIR,LOW);

  Serial.begin(115200);
  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_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;

   if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(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
   s->set_framesize(s, FRAMESIZE_QVGA);

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

  WiFi.mode(WIFI_STA);
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);
   clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add
root certificate for api.telegram.org
   while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
   }
   Serial.println("");
   Serial.println("WiFi connected");

   startCameraServer();

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

void loop() {
   while (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid, password);
    Serial.print(".");
    delay(2000);
   }
   if(LEDState) {
    digitalWrite(LED, HIGH);
   }
   else{
    digitalWrite(LED,LOW);
```

```
  }
  if (sendPhoto){
    Serial.println("Preparing photo");
    sendPhotoTelegram();
    sendPhoto = false;
  }

  if (flag){
    delay(1000);
    if(digitalRead(PIR) == 1){
      Serial.print("Motion Detected, Value = ");
      Serial.println(digitalRead(PIR));
      String motion = "Terdeteksi gerakan!!\n";
      motion += "Foto akan segera dikirim\n";
      bot.sendMessage(chatId, motion, "");
      sendPhotoTelegram();
      activateRelay();
    }
  }

  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();
  }
 }
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 = "--RandomNerdTutorials\r\nContent-Disposition: form-
data; name=\"chat_id\"; \r\n\r\n" + chatId + "\r\n--
RandomNerdTutorials\r\nContent-Disposition: form-data;
name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
    String tail = "\r\n--RandomNerdTutorials--\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=RandomNerdTutorials");
    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 (c == '\n'){
            if (getAll.length()==0) state=true;
            getAll = "";
          }
          else if (c != '\r'){
            getAll += String(c);
          }
          if (state==true){
            getBody += 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;
}

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

  for (int i = 0; i < numNewMessages; i++){
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != chatId){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

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

```
    String fromName = bot.messages[i].from_name;

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

    if (text == "/piron"){
      flag = 1;
      bot.sendMessage(chatId, "PIR Sensor sudah aktif, Saat terjadi gerakan
anda akan dikirimkan foto", "");
    }

    if (text == "/piroff"){
      flag = 0;
      bot.sendMessage(chatId, "PIR sensor sudah mati, Anda tidak akan
menerima pemberitahuan lagi saat terjadi gerakan", "");
    }

//    if (text == "/readings"){
//      String readings = getReadings();
//      bot.sendMessage(chatId, readings, "");
//    }

    if (text == "/start"){
      String welcome = "Selamat datang di ESP32-CAM Telegram YUYUN
.\n";
      welcome += "Berikut adalah hal yang dapat Anda lakukan :\n";
      welcome += "/photo : Ambil foto\n";
      welcome += "/pir_on : Mengaktifkan sensor PIR\n";
      welcome += "/pir_off : Mematikan sensor PIR\n";
      welcome += "Anda juga akan mendapatkan notifikasi saat terjadi
gerakan dari sensor PIR\n";
      bot.sendMessage(chatId, welcome, "Markdown");
    }
  }
}
```

2. Kode program untuk modul ESP8266

```
#include <ESP8266WiFi.h>
#include <ESPAsyncWebServer.h>

const char* ssid = "Gentle";
const char* password = "0nepiec3";

short relayPin = 4;
```

```
short buttonPin = 5;
bool relayState = LOW;
unsigned long relayStartTime = 0;

AsyncWebServer server(80);

void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, relayState);

  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

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

  server.on("/open_relay", HTTP_GET, [](AsyncWebServerRequest*
request) {
    relayState = HIGH;
    relayStartTime = millis();
    Serial.println("Relay ON");
    request->send(200, "text/plain", "Relay diaktifkan");
  });

  server.on("/close_relay", HTTP_GET, [](AsyncWebServerRequest*
request) {
    relayState = LOW;
    Serial.println("Relay OFF");
    request->send(200, "text/plain", "Relay dimatikan");
  });

  server.begin();
}

void loop() {

  bool buttonState = digitalRead(buttonPin);
```

```
 if (buttonState == LOW) {
  relayState = HIGH;
  relayStartTime = millis();
  Serial.println("Relay ON");
 }

 // Mengecek waktu relay aktif dan mematikan relay setelah 5 detik
 if (relayState == HIGH && millis() - relayStartTime >= 5000) {
  relayState = LOW;
  Serial.println("Relay OFF");
 }

 digitalWrite(relayPin, relayState);
 server.handleClient();

}
```

3. **Kode program untuk SSD**

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

base_model = MobileNetV2(weights = "imagenet", include_top = False,
input_shape = (224, 224, 3))

head_model = base_model.output
head_model = GlobalAveragePooling2D()(head_model)
head_model = Dense(128,activation='relu')(head_model)
head_model = Dense(2, activation='softmax')(head_model)
```

```
model_train = Model(inputs = base_model.input, outputs = head_model)

for layer in base_model.layers:
        layer.trainable = False

model_train.summary()

directory_dataset = "D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/dataset/datasetdevi/"
category_dataset = ["Devi", "Unknown"]

data = []
labels = []

for category in category_dataset:
        path = os.path.join(directory_dataset, category)
        for img in os.listdir(path):
                img_path = os.path.join(path, img)
                image = load_img(img_path, target_size=(224, 224))
                image = img_to_array(image)
                image = preprocess_input(image)
                data.append(image)
                labels.append(category)

lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)
# print(labels)


learning_rate_size = 1e-4
epoch_size = 10
batch_size_number = 8

(trainX, testX, trainY, testY) = train_test_split(data,

                labels,

                test_size=0.20,

                stratify=labels,
```

```
                    random_state=42)

aug = ImageDataGenerator(rotation_range=20,
                    zoom_range=0.15,
                    width_shift_range=0.2,
                    height_shift_range=0.2,
                    shear_range=0.15,
                    horizontal_flip=True,
                    fill_mode="nearest")

checkpoint = ModelCheckpoint("D:/lib/fuzzy/MobileNet-SSD_Face-
Detection-and-Recognition/dataset/result/Devi/face_recog30.h5",
                      monitor="val_loss",
                      mode="min",
                      save_best_only = True,
                      verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                    min_delta = 0,
                    patience = 3,
                    verbose = 1,
                    restore_best_weights = True)

callbacks = [earlystop, checkpoint]

print("[INFO] compiling model...")
opt = Adam(lr=learning_rate_size, decay=learning_rate_size / epoch_size)
model_train.compile(loss="binary_crossentropy",
                optimizer=opt,
                metrics=["accuracy"])

print("[INFO] training head...")
H = model_train.fit(aug.flow(trainX, trainY,
batch_size=batch_size_number),
                steps_per_epoch=len(trainX) // batch_size_number,
                validation_data=(testX, testY),
                validation_steps=len(testX) // batch_size_number,
                epochs=epoch_size,
                callbacks = callbacks)

predIdxs = model_train.predict(testX, batch_size=batch_size_number)

predIdxs = np.argmax(predIdxs, axis=1)
```

```
print(classification_report(testY.argmax(axis=1), predIdxs,
        target_names=lb.classes_))

plt.style.use("ggplot")
plt.figure()

plt.plot(H.history['accuracy'] , label = 'train acc')
plt.plot(H.history['loss'] , label = 'train loss')


plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig("D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/dataset/result/Devi/Devi.png")

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import imagenet_utils
from tensorflow.keras.models import load_model

import numpy as np
import imutils
import time
import cv2
import os
from os import listdir
from os.path import isfile, join

prototxt_path = "D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/face-detector-model/ssd_face.prototxt"
weights_path = "D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/face-detector-model/ssd_face.caffemodel"

faceNet = cv2.dnn.readNet(prototxt_path, weights_path)

# load the face mask detector model from disk
model_recog = load_model('D:/lib/fuzzy/MobileNet-SSD_Face-Detection-
and-Recognition/dataset/result/Devi/face_recog30.h5')

path = "D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/dataset/test/Devi/"
# path = "/content/drive/MyDrive/FaceRecog/new_dataset/test/unknown/"
```

```
many_file_in_dict = len(os.listdir(path))
image_file = os.listdir(path)
# print(image_file)

for count in range(many_file_in_dict):
        path_image = os.path.join(path + image_file[count])
        input_image = cv2.imread(path_image)

        # cv2_imshow(input_image)

        (h, w) = input_image.shape[:2]
        blob = cv2.dnn.blobFromImage(input_image, 1.0, (224, 224),
(104.0, 177.0, 123.0))

        faceNet.setInput(blob)
        detections = faceNet.forward()

        # faces = []
        pred = {"0", "1"}

        for i in range(0, detections.shape[2]):
                confidence = detections[0, 0, i, 2]
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                if confidence > 0.5:
                        (startX, startY) = (max(0, startX), max(0, startY))
                        (endX, endY) = (min(w - 1, endX), min(h - 1,
endY))

                        face_detect = cv2.cvtColor(input_image,
cv2.COLOR_BGR2GRAY)
                        face_detect = cv2.resize(face_detect, (224, 224))
                        face_detect = img_to_array(face_detect)
                        face_detect = preprocess_input(face_detect)
                        face_detect = np.expand_dims(face_detect, axis=0)

                        preds = model_recog.predict(face_detect)
                        label_percent =
str("{:.2f}%".format((max(max(preds))) * 100))

                        preds = np.argmax(preds)

                        # facial = pred[str(preds)]
```

```
                    if preds == 0:
                        label = ""
                        color = (0, 0, 255)
                    if preds == 1:
                        label = "Devi"
                        color = (0, 255, 0)

                    label_full = label + " " + label_percent
                    label_size = cv2.getTextSize(label_full,
cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
                    cv2.rectangle(input_image, (startX, startY), (endX,
endY), color, 2)
                    cv2.rectangle(input_image, (startX, startY -20 ),
((startX + label_size[0][0]) + 10, startY - 2),
                                  (255, 255, 255),
cv2.FILLED)
                    cv2.putText(input_image, label_full, (startX + 4 ,
startY - 6), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 0))

        image_resize = cv2.resize(input_image, (480, 480))
        # print(preds)
        cv2.imwrite("D:/lib/fuzzy/MobileNet-SSD_Face-Detection-and-
Recognition/dataset/result/Devi/test" + str(count + 1) + "-28.jpg",
                    image_resize)
        time.sleep(1)
        print('\n')
```

4. **Kode program untuk *facerecognition***

```
import cv2
from urllib.request import urlopen
import numpy as np
import face_recognition
import requests

ESP32_CAM_IP = '192.168.43.159'
url = f'http://192.168.43.159/capture'
relay_url = f'http://192.168.43.152/open_relay'  # URL untuk
mengaktifkan relay di ESP32-CAM

# Load the known faces and their names
known_faces = []
known_names = []
for i in range(5):
   known_image = face_recognition.load_image_file(f'foto{i}.jpg')
```

```
    known_encoding = face_recognition.face_encodings(known_image)[0]
    known_faces.append(known_encoding)
    known_names.append(f'{i}')

def activate_relay():
    try:
        response = requests.get(relay_url)
        if response.status_code == 200:
            print('Relay diaktifkan')
        else:
            print('Gagal mengaktifkan relay')
    except requests.exceptions.RequestException as e:
        print('Gagal menghubungi ESP32-CAM:', e)

while True:
    img_resp = urlopen(url)
    imgnp = np.asarray(bytearray(img_resp.read()), dtype="uint8")
    img = cv2.imdecode(imgnp, -1)

    # Convert the image from BGR (OpenCV format) to RGB
(face_recognition format)
    rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Find all the faces in the image
    face_locations = face_recognition.face_locations(rgb_img)
    face_encodings = face_recognition.face_encodings(rgb_img,
face_locations)

    # Loop over each face in the image
    for face_location, face_encoding in zip(face_locations, face_encodings):
      # See if the face is a match for the known faces
        matches = face_recognition.compare_faces(known_faces,
face_encoding, tolerance=0.4)

        # Use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_faces,
face_encoding)
        best_match_index = np.argmin(face_distances)
        # If there is a match, display the name
        if matches[best_match_index]:
            name = known_names[best_match_index]
            top, right, bottom, left = face_location
            cv2.rectangle(img, (left, top), (right, bottom), (0, 255, 0), 2)
            cv2.putText(img, name, (left, top - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
```

```
            activate_relay()
        cv2.imshow("Camera", img)
        if cv2.waitKey(1) == 113:
            break

cv2.destroyAllWindows()
```