Nesnelerin İnterneti ve Uygulamaları

Onur GÖNÜLLÜ - 1722130013

1-)Mqtt protokolü aracılığı ile Sisteme Tanımlı Yüzler (Yüz Tanıma) İle Açılan Kapı Projesini çalıştıran kodu yazınız.

Arduino Kodu

```
#include <MQTT.h>
#include <MQTTClient.h>
#include <ESP8266WiFi.h>
#include <MQTTClient.h>
#define led1 D2
char ssid[] = "Redmi";
char pass[] = "onur1905";
const char* broker = "onurgonullu.cloud.shiftr.io";
char mqttUserName[] = "onurgonullu";
char mqttPass[] = "12345";
WiFiClient net;
MQTTClient client;
unsigned long lastMillis = 0;
int alinan_mesaj_topic1;
int alinan_aktivasyon;
void connect() {
Serial.print("\nconnecting to wifi.");
while (WiFi.status() != WL_CONNECTED) {
Serial.print(".");
delay(5000);
}
char clientID[] ="OnurGonullu";
Serial.print("\nconnecting to broker...");
while (!client.connect(clientID,mqttUserName,mqttPass)) {
Serial.print(".");
delay(5000);
}
Serial.println("\nconnected!");
client.subscribe("kapi_control");
}
void messageReceived(String &topic, String &payload) {
if (topic=="kapi_control") {
alinan_mesaj_topic1 = (payload.toInt());
girdi_oku();
}
}
```

```
void setup() {
Serial.begin(115200);
pinMode(led1, OUTPUT);
WiFi.begin(ssid, pass);
client.begin(broker, net); //broker , wifi
client.onMessage(messageReceived);
connect();
}
void loop() {
client.loop();
delay(10);  // <- fixes some issues with WiFi stability
if (!client.connected())connect();
if (millis() - lastMillis > 3000) {
girdi_oku();
lastMillis = millis();
}
}
void girdi_oku() {
if (alinan_mesaj_topic1 == 1) {
digitalWrite(led1, HIGH);
Serial.println("kapi acildi");
}
else {
digitalWrite(led1, LOW);
Serial.println("kapi kapatildi");
}
}
```

Python(Main) Kodları

```
import cv2
from simple_facerec import SimpleFacerec
from mqtt_adapter import mqtt_yaz
from excel_adapter import excel_yaz
# Encode faces from a folder
sfr = SimpleFacerec()
sfr.load_encoding_images("image_databasefile/")


cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    # Detect Faces
```

```
    face_locations, face_names = sfr.detect_known_faces(frame)
    for face_loc, name in zip(face_locations, face_names):
        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
        cv2.putText(frame, name,(x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)
        mqtt_yaz(name) #mqtt burada cagiriyoruz
        excel_yaz(name)
    cv2.imshow("Yuz Tanima", frame)
    key = cv2.waitKey(1) #bir tuşa basılırsa
    if key == 27: # ve bu tuş esc ise
        break
cap.release()
cv2.destroyAllWindows()
```

Python Mqtt Kodları

```
import paho.mqtt.client as mqttclient
import time
def on_connect(client,usedata,flags,rc):
    if rc==0:
        print("client is connected")
        global connected
        connected=True
    else:
        print("connection failed")
#mqtt broker codes:
connected=False
mqtt_port = 1883
mqtt_broker = "onurgonullu.cloud.shiftr.io"
mqtt_username = "onurgonullu"
mqtt_password = "12345"
client = mqttclient.Client("MQTT")
client.username_pw_set(mqtt_username,password=mqtt_password)
client.on_connect=on_connect
client.connect(mqtt_broker,port=mqtt_port)
client.loop_start()
while connected!=True:
    time.sleep(0.2)
#mqqt kodları burada bitti.
def mqtt_yaz(name): #name değiskeni mainden gönderilir
 if (name == "unknown"):
     client.publish("kapi_control", 0)
```

```
    client.publish("isim_control", name)
    client.loop_stop()
 else:
    client.publish("kapi_control", 1)
    client.publish("isim_control", name)
    time.sleep(2)
    client.publish("kapi_control", 0)
    client.publish("isim_control", "unknown")
    #client.publish("kapi_control", "ac")
    client.loop_stop()
```

Python SimpleFacerec Klasörü

```
import face_recognition
import cv2
import os
import glob
import numpy as np
class SimpleFacerec:
   def __init__(self):
      self.known_face_encodings = []
      self.known_face_names = []
      # Resize frame for a faster speed
      self.frame_resizing = 0.25
   def load_encoding_images(self, images_path):
      """
      Load encoding images from path
      :param images_path:
      :return:
      """
      # Load Images
      images_path = glob.glob(os.path.join(images_path, "*.*"))
      print("{} encoding images found.".format(len(images_path)))
      # Store image encoding and names
      for img_path in images_path:
         img = cv2.imread(img_path)
         rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
         # Get the filename only from the initial file path.
         basename = os.path.basename(img_path)
         (filename, ext) = os.path.splitext(basename)
         # Get encoding
         img_encoding = face_recognition.face_encodings(rgb_img)[0]
```

```
        # Store file name and file encoding
        self.known_face_encodings.append(img_encoding)
        self.known_face_names.append(filename)
    print("Encoding images loaded")
def detect_known_faces(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
    # Find all the faces and face encodings in the current frame of video
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
        name = "unknown"

        # # If a match was found in known_face_encodings, just use the first one.
        # if True in matches:
        #     first_match_index = matches.index(True)
        #     name = known_face_names[first_match_index]
        # Or instead, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = self.known_face_names[best_match_index]
        face_names.append(name)
    # Convert to numpy array to adjust coordinates with frame resizing quickly
    face_locations = np.array(face_locations)
    face_locations = face_locations / self.frame_resizing
    return face_locations.astype(int), face_names
```