Dokumentation zur IT-PROJEKT

**KÜHLRAUM**

**Die Steuerung einer Tür (sowie. Lüfter) eines Kühlraums mit Temperatur- und Feuchtigkeit- Erfassung**

YVAN VALDER SIMO GUENO, 7212803

14. Dezember 2022

Prüfer: Pr. Dr. Kai Luppa

**Der Hauptprogramm**

#Alle definierten Bibliotheken müssen installiert werden

#---------------------------------------------------------------------------------

----------------------------------------------------------------------------------

#Importierung der Bibliothek paho.mqtt.client (mqtt), damit MQTT-clients #miteinander kommuniziert

import paho.mqtt.client as mqtt

#Importierung der Bibliothek zum verbinden und steuern von Sensoren und Aktoren

import RPi.GPIO as GPIO

#Importierung der Funktion , damit die While-Schleife gleichzeitig mit dem #loop\_forever funktioniert

import threading

#Importierung der Bibliothek für die Zeit und für den Method sleep()

import time

from time import sleep

#Importierung der Biliothek zur Steuerung des Feutchtigkeitssensors sowie der #Temperatur

import dht11

#bibliothek luma und luma.led\_matrix , um einen Text auf einem LED-Matrix-Display #anzuzeigen

import luma

from luma.led\_matrix.device import max7219

from luma.core.interface.serial import spi, noop

from luma.core.render import canvas

from luma.core.virtual import viewport

from luma.core.legacy import text, show\_message

from luma.core.legacy.font import proportional, CP437\_FONT, TINY\_FONT, SINCLAIR\_FONT, LCD\_FONT

#---------------------------------------------------------------------------------

led =[29,31,33,35,38] #Pins led

interrupt=22 #Button\_pin für Tkinter

b= 12 #Buzzer Pin

relay\_pin = 40 #relay Pin

motion\_pin = 16 #Pin des Bewegungssensors

servoPin = 37 #Servo Pin

DHT11\_pin= 7 #DHT11 Pin

#---------------------------------------------------------------------------------

GPIO.setmode(GPIO.BOARD) #Die GPIO Boardkonfiguration benutzen.

GPIO.setwarnings(False) #Deaktivieren der Anzeige von GPIO-Warnungen

GPIO.setup(relay\_pin, GPIO.OUT) #konfiguration des Relay\_pins als Ausgang

GPIO.setup(led, GPIO.OUT) #konfiguration der LEDs als Ausgang

GPIO.setup(b, GPIO.OUT) #Konfiguration des Buzzers als Ausgang

GPIO.setup(servoPin, GPIO.OUT) # Set servoPin to OUTPUT mode

GPIO.setup(motion\_pin, GPIO.IN) #Der Pin der Deklarierten Variable wird als #Input(Eingang) gesetzt.

GPIO.setup(interrupt, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP) #Hier wird den Pin als

#Eingang gesetzt.

#---------------------------------------------------------------------------------

#--------------Funktion zum automatischen Öffnen und Schließen der Tür -----------

def tür():

GPIO.setup(servoPin, GPIO.OUT)

# pin 37 for servo1 frequenz 50Hz

servo1 = GPIO.PWM(servoPin,50)

# 0° wird für den Servo der Referenz Position sein

servo1.start(0)

# Turn servo1 to 90°

servo1.ChangeDutyCycle(7.5)

time.sleep(0.5)

servo1.ChangeDutyCycle(0)

# Wait for 2 seconds

time.sleep(3)

# servo1 back to 0°

servo1.ChangeDutyCycle(2)

time.sleep(0.5)

servo1.ChangeDutyCycle(0)

servo1.stop()

#GPIO.cleanup()

#------------------Funktion nur zum Schließen der Tür-----------------------------

def tür\_zu():

GPIO.setup(servoPin, GPIO.OUT)

# pin 37 for servo1 frequenz 50Hz

servo1 = GPIO.PWM(servoPin,50)

# 0° wird für den servo der Referenz position sein

servo1.start(0)

# servo1 back to 0°

servo1.ChangeDutyCycle(2)

time.sleep(0.5)

servo1.ChangeDutyCycle(0)

servo1.stop()

#GPIO.cleanup()

#------------------Funktion nur zum Öffnen der Tür----------------------------

def tür\_auf():

GPIO.setup(servoPin, GPIO.OUT)

# pin 37 for servo1 frequenz 50Hz

servo1 = GPIO.PWM(servoPin,50)

# 0° wird für den servo der Referenz position sein

servo1.start(0)

# Turn servo1 to 90

servo1.ChangeDutyCycle(7.5)

time.sleep(0.5)

servo1.ChangeDutyCycle(0)

servo1.stop()

#GPIO.cleanup()

#-----------------------Ankündigung oder Laden der Türöffnung---------------------

def ledwater():

for i in led:

GPIO.setup(i, GPIO.OUT)

GPIO.output(i , GPIO.LOW)

#led on

sleep(0.5)

if i==38:

break;

#led off

GPIO.output(led , GPIO.HIGH)

#---------------Funktion zum Abrufen der temperature und Feuchtigkeit-------------

def temp():

# Temperatur und Feuchtigkeit mit DHT11 abrufen

instance = dht11.DHT11(DHT11\_pin)

result = instance.read()

#Ruf Datein bis gültige Werte

while not result.is\_valid():

result = instance.read()

#Anzeigen der Result in der Konsole

print("Temperature: %-3.1f C" % result.temperature)

print("Humidity: %-3.1f %%" % result.humidity)

return result

#-----Funktion zum anzeigen der Temperatur und Feuchtigkeit auf dem Matrix -------

def matrix(result):

# Matrix Gerät festlegen und erstellen.

serial = spi(port=0, device=1, gpio=noop())

device = max7219(serial, cascaded= 1, block\_orientation=90,rotate= 0)

# Matrix Initialisierung in der Konsole anzeigen

print("[-] Matrix initialized")

#conversion der Abgerufenen Werten im String, damit diese darstellbar seien

matrix0=str(result.temperature)

matrix2=str(result.humidity)

#Die Temperatur und feuchtigkeit wird hier zu der GUI geschickt

client.publish("fhdo/itp/gp1/11",matrix0)

client.publish("fhdo/itp/gp1/13",matrix2)

global matrix1

matrix1= "Temperature "+matrix0 +"C"

# Ausgegebenen Text in der Konsole Anzeigen

print("--Temperature: %s Grad --" % matrix0)

#Anzeigen der Temperatur

show\_message(device, matrix1 , fill="white", font=proportional(CP437\_FONT), scroll\_delay=0.1)

#----------------------Funktion bei der Öffnung der Tür---------------------------

def on\_open():

global open\_1

open\_1='Door Open.....'

#veröffebtlichung der Stand der Tür auf der grafischen Benutzeroberfläche

client.publish("fhdo/itp/gp1/12",'op')

print(open\_1)

#Buzzer Ton

#Gebe Geraeusch aus

GPIO.output(b, GPIO.HIGH)

#warte eine halbe Sekunde

time.sleep(0.5)

#Stoppe Geraeuschausgabe

GPIO.output(b, GPIO.LOW)

#Led anzeigen

ledwater()

# Oeffne Relais

GPIO.output(relay\_pin, GPIO.LOW)

# warte eine halbe Sekunde

time.sleep(0.5)

# schliesse Relais

GPIO.output(relay\_pin, GPIO.HIGH)

# Wird der print Befehl ausgeführt

time.sleep(0.1)

# 0,1 Sekunde Warten, dann Öffne der Tür

tür()

#---------------------------------------------------------------------------------

#--------------------------Ton bei dem Schließen der Tür mit der GUI-------------

def buzzer\_0():

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.25)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

time.sleep(0.25)

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.5)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

#-------------------------Ton beim Einschalten des Lüfters------------------------

def buzzer\_1():

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.25)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

time.sleep(0.25)

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.25)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

#----------------------------------Ton beim Ausschalten des Lüfters---------------

def buzzer\_2():

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.25)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

time.sleep(0.25)

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.25)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

time.sleep(0.25)

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.5)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

#----------------------------------Funktion zur Steurerung des Lüfters------------

def Turn\_on():

buzzer\_1()

serial = spi(port=0, device=1, gpio=noop())

device = max7219(serial, cascaded= 1, block\_orientation=90, rotate= 0)

show\_message(device, 'On' , fill="white", font=proportional(CP437\_FONT), scroll\_delay=0.7)

def Turn\_off():

buzzer\_2()

serial = spi(port=0, device=1, gpio=noop())

device = max7219(serial, cascaded= 1, block\_orientation=90,

rotate= 0)

show\_message( device, 'Off' , fill="white", font=proportional(CP437\_FONT), scroll\_delay=0.8)

#---------------------Funktion beim Öffnen der Tür mit dem GUI -------------------

def on\_open\_msg():

global open\_1

open\_1='Door Open.....'

print(open\_1)

#Buzzer Ton

GPIO.output(b, GPIO.HIGH)

#Gebe Geraeusch aus

time.sleep(0.5)

#warte eine halbe Sekunde

GPIO.output(b, GPIO.LOW)

#Stoppe Geraeuschausgabe

ledwater()

# Oeffne Relais

GPIO.output(relay\_pin, GPIO.LOW)

# warte eine halbe Sekunde

time.sleep(0.5)

# schliesse Relais

GPIO.output(relay\_pin, GPIO.HIGH)

# Wird der print Befehl ausgeführt

time.sleep(0.1)

# 0,1 Sekunde Warten

# Beginn einer Schleife

tür\_auf()

#--------------------------------Funktion Zur Nutzung des Bewegungsensors---------

#Die Empfindlichkeit des Bewegungssensors kann mit einem Potentiometer 3a #im #Abbildung ++ eingestellt werden.

def motion():

#GPIO.setup(interrupt, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

client.publish("fhdo/itp/gp1/12",’wait’)

while GPIO.input(interrupt)==GPIO.HIGH :

if(GPIO.input(motion\_pin) == 1): # Wenn der Sensor Input = 1 ist

print ("Bewegung Erkannt!") # Wird der print Befehl ausgeführt

on\_open() #Automasches Öffnen und Schließen der Tür

result = temp()

matrix(result) #Darstellung der Temperatur und Feuchtigkeit

time.sleep(0.15) # 0.15 Sekunde Warten

client.publish("fhdo/itp/gp1/12",’wait’)

#if(GPIO.input(motion\_pin) == 0): # Wenn der Sensor Input = 0 ist

#print ("Keine Bewegung ...") # Wird der print Befehl ausgeführt

global a

a="Die manuelle Steuerung ist aus!"

print(a)

#Die Steuerung wird hier nur durch dem GUI gemacht

client.publish("fhdo/itp/gp1/12","ANALOG")

GPIO.cleanup(servoPin)

# der servopin muss nach jeder Drehung gelöscht werden ,damit er immer die #gleiche Drehung macht.

#-----------------------------------------------------------------------------

#--Funktion , wenn der Broker mit der vorgegebenen Topic eine Nachricht bekommt---

def on\_message (client, userdata, message):

msg = str (message.payload.decode ("utf-8"))

print(message.topic,'-->' ,msg)

if message.topic=="fhdo/itp/gp1/12":

if msg=="open":

on\_open\_msg()

result=temp()

matrix(result)

elif msg=='closed':

buzzer\_0()

tür\_zu()

elif msg=='turn':

Turn\_on()

elif msg=='off':

Turn\_off()

elif msg=="press":

print('Pressed...')

thread = threading.Thread(target=motion)

thread.start()

#---------------------Funktion beim Anschließen oder Abonnement der Client ------

def on\_connect (client, userdata, flags, rc):

client.subscribe ('fhdo/itp/gp1/#')

#----------------------------Anwendung der MQTT- Protokoll--------------------------------------------------------------------------------

BROKER\_ADDRESS="broker.emqx.io"

#Objekt der Classe Client wird erstellt

client = mqtt.Client ()

#subskription der Client

client.on\_connect = on\_connect

client.on\_message =on\_message

#Anschließen mit einem Broker

client. connect (BROKER\_ADDRESS)

print ("Connected to MQTT Broker:"+" broker.emqx.io")

#Aufruf der Funktion Motion zum automatischen Öffnen und Schließen der Tür

motion()

client. loop\_forever()

#End

#Import the required Libraries

#import time

import paho.mqtt.client as mqtt

#importierung der Tkinter Bibliothek ,die die Erstellung von grafischen #Benutzeroberflächen ermöglicht.

from tkinter import \*

from tkinter import messagebox

#import PIL Bibliotheken ,die Unterstützung für das Öffnen, Bearbeiten und #Speichern vieler #verschiedener Bilddateiformate bietet.

from PIL import Image,ImageTk

#Declaration von Variablen für die Feuchtigkeit und der Temperature

msg\_temperatur=None

msg\_humidity=None

#Auflistung verschieden Topics

topics = ['fhdo/itp/gp1/12','fhdo/itp/gp1/11','fhdo/itp/gp1/13']

#--------------------------------------------------------------------------#-----------------------------

#----------Funktion , wenn der Broker mit der vorgegebenen Topic eine #Nachricht bekommt-------

def on\_message(client , userdata, message):

msg=str(message.payload.decode())

print(message.topic, '--->', msg)

if message.topic==topics[0]:

if msg=="ANALOG":

Nachricht="Die manuelle Steuerung ist aus!..."

Fact.set(Nachricht)

elif msg=="op":

Fact.set("Welcome....")

elif msg=="wait":

Fact.set("Waiting on Movement....")

if message.topic==topics[1]:

msg\_temperatur="Temperature: "+msg+" °C"

Fact\_temp.set(msg\_temperatur)

if message.topic==topics[2]:

msg\_humidity='Humidity: '+msg+" %"

Fact\_humid.set(msg\_humidity)

#---------------------Funktion beim Anschließen oder Abonnement der Client

def on\_connect (client, userdata, flags, rc):

client.subscribe ('fhdo/itp/gp1/#')

#------------------------Anwendung der MQTT- Protokoll-----------------

client = mqtt.Client()

BROKER\_ADDRESS="broker.emqx.io"

client.on\_connect = on\_connect

client.on\_message =on\_message

client. connect (BROKER\_ADDRESS)

client. loop\_start()

#----Ausführende Funktion , wenn man auf OPEN klickt ----------------

def btn\_on\_click():

client.publish(topics[0], 'open')

Nachricht ="Der Tür ist geöffnet!..."

Fact.set(Nachricht)

#----Ausführende Funktion , wenn man auf CLOSED klickt ----------------

def btn\_off\_click():

client.publish(topics[0], 'off')

Nachricht ="Der Lufter ist aus!..."

Fact.set(Nachricht)

#----Ausführende Funktion , wenn man auf Turn\_On klickt ----------------

def btn\_turn\_click():

client.publish(topics[0], 'turn')

Nachricht ="Der Lufter ist ein!..."

Fact.set(Nachricht)

#----Ausführende Funktion , wenn man auf Turn\_Off klickt ----------------

def btn\_closed\_click():

client.publish(topics[0], 'closed')

Nachricht ="Der Tür ist zu!..."

Fact.set(Nachricht)

#----Ausführende Funktion , wenn man auf MANUELLE klickt ----------------

def btn\_press\_click():

messagebox.askquestion("Manuelle Steurerung",

"Wollen Sie wirklich die manuelle Steuerung aktivieren?")

if 'yes':

client.publish(topics[0], 'press')

Nachricht="Manuelle Steuerung aktiviert!..."

Fact.set(Nachricht)

print(Nachricht)

else:

Nachricht="Die Steuerung ist noch Digital!..."

Fact.set(Nachricht)

print(Nachricht)

#Create an instance of tkinter frame

win = Tk()

win.title("IT-Projekt Gruppe1")

#win.iconbitmap('E:\a.ico')

# Ändert der size des Fenster

win.geometry("1280x620")

# Create text widget and specify size.

titel= Label(win, text="KÜHLRAUM",fg="deep sky blue", bg="royal blue", font="Helvetica 16 bold italic").grid(row=0, column=2, columnspan=2)

Tür=Label(win, text='Tür-Button', fg='black', bg='white', font='verdana').grid(row=3, column=0, columnspan=2)

lufter=Label(win, text='Lufter-Button', fg='black', bg='white', font='verdana').grid(row=3, column=6, columnspan=2)

#glogale Declaration alle Textvariable zum Anzeigen in Labels

global Fact

global Fact\_temp

global Fact\_humid

Fact=StringVar()

Fact\_temp=StringVar()

Fact\_humid=StringVar()

Fact.set("--Stand--")

Fact\_temp.set('--Temperature--')

Fact\_humid.set('--Humidity--')

#Erstellung eine Objekt my\_img zur classe ImageTk

my\_img = ImageTk.PhotoImage(Image.open("fh.png"))

#Erstellung eines Labels zur Darstellung des Bildes

Bild= Label(image=my\_img).grid(row=2, column=2,rowspan=8)

#Erstellung von Labeln zum Anzeigen der Temperatur, der Feuchtigkeit und #des Standes des JOYPIs

temperatur\_info= Label(win, textvariable=Fact\_temp, fg="black", bg="white", font="VERDANA").grid(row=6,column=0,columnspan=2)

feutchtigkeit\_info= Label(win, textvariable=Fact\_humid, fg="black", bg="white", font="VERDANA").grid(row=6,column=6,columnspan=2)

stand= Label(win, textvariable=Fact, fg="black", bg="white", font="VERDANA").grid(row=10,column=2,columnspan=3)

#Erstellung allen gebrauchten Buttons

bouton0=Button(win, text="OPEN", relief=RAISED,fg="SpringGreen4",bg="SpringGreen2", font="VERDANA",padx=30, command= btn\_on\_click).grid(row=4,column=0)

bouton1=Button(win, text="Turn\_ON", relief=RAISED, fg="blue4",bg="turquoise1", font="VERDANA",padx=30, command=btn\_turn\_click ).grid(row=4, column=6)

bouton2=Button(win, text="CLOSED", relief=RAISED, fg="white",bg="grey28", font="VERDANA",padx=30, command=btn\_closed\_click ).grid(row=4,column=1)

bouton3=Button(win, text="Turn\_OFF", relief=RAISED, fg="white",bg="grey28", font="VERDANA",padx=30, command=btn\_off\_click).grid(row=4, column=7)

bouton4=Button(win, text="Schließen", relief=SUNKEN, cursor="spider",fg="white",bg="red2", font="IMPACT",command=win.quit).grid(row=10,column=7)

bouton5=Button(win, text="MANUELLE", relief=GROOVE, cursor="spider",fg="DarkOrange1",bg="WHITE", font="IMPACT",command=btn\_press\_click).grid(row=10,column=0)

win.mainloop()

#weist Python an, die Ereignisschleife von Tkinter auszuführen