**8. óra: Szellőztető rendszer online**

1. 5 perces beszámoló tetszőleges tartalommal és üzenettel valamint tetszőlegesen választott retorikai eszközökkel. Nekünk kell eldönteni, hogy mi lehetett az üzenet. Figyelni kell a beszéd felépítésére (bevezető, fő rész és konklúzió) és arra, hogy a tartalom minden eleme támogassa az üzenetet, illetve, hogy használjunk retorikai eszközöket.
2. 3 perces véleményalkotás  
   - bevezető, fő rész, konklúzió  
   - 2 pozitív és egy konstruktív javaslat  
   - konklúzióban összefoglalni mi lehetett az üzenet, volt-e retorikai eszköz használva és adni még egy pozitív megjegyzést.
3. Szellőztető mérése online:
   1. *Bevezető*:  
      Miután elkészítettük a saját szellőztető rendszerünket és meg voltál elégedve a működésével, azon kaptad magad, hogy lehet nem ártana tudni azt is, hogy a nap/hét/év melyik szakában lép a rendszer működésben. Gyorsan rájöttél, hogy egyfolytában nem lehetsz ott mellette és azt is tuttad, hogy nem csak akkor szeretnéd megnézni a rendszert, amikor otthon vagy.

Eldöntötted, hogy a következő projekted az lesz, hogy felteszed a mérési eredményeket a felhőbe és interneten kersztül bárhonnan tudod majd ellenőrízni az ingatlanod paramétereit.

* 1. *Előkészületek*:  
     Készítsünk ki próbapaneleket, jumper kábeleket, 10 kOhm ellenállásokat a thermisztornak, DHT11 thermisztorokat, DC motorokat és az LD293 vezérlőt. Emellé szükség lesz mérőműszerre is.   
     - Rakjuk össze az áramkört.   
     - Teszteljük le pythonból a motort és a DHT11 műküdését, hogy jó e az áramkör.  
       
     *Részletekre bontás:*Elkészíteni az áramkört.
     + Importálni az adafruit DHT11 objektumát, és a Motor objektumot, illetve a temperature\_functions modulból a megfelelő függvényeket.
     + Inicializálni a motort és a DHT11-t.
     + Definiálni egy függvényt, ami kiolvassa a hőmérsékletet.
     + Definiálunk egy függvényt ami kiolvassa a páratartalmat.
     + Definiálunk egy függvényt, ami a megadott feltételek alapján vezérli a ventillátort.
     + Kövessük a cloud4rpi regisztrációs utasításait.  
       - Állítsuk be a honlap adatainak frissítési idejét.
  2. *Elektronika:*  
     - Készítsük el az áramkört. Érdemes kihasználni a közös földelést és tápot, hiszen sok dróttal dolgozunk majd.
  3. *Kód:*  
     - Egeszítsük ki a lenti kódvázlatot.  
     - Vegyük át a motor mozgatásához szükséges metódusokat, pl. Forward, backward, stop.  
     - Vegyük át dht11 objektum metódusait, pl. temperature és humidity.   
     - Vegyünk példát a try except szerkezetre és magyarázzuk el miért fontos:  
       
     a = 3

b = 0

c = a/b  
  
try:

c = a/b

except:

print('Hiba tortent c ertekeben, ezert konstans erteket rendelunk hozza')

c = 2

print(f'c = {c}')

* 1. *Tesztelés:*  
     - teszteljük a programot.
  2. *Fejlesztés:*

- Módosítsuk, hogy milyen gyakran frissüljön az információ a felhőben.

- Ábrázoljuk grafikonon az idő függvényében, hogy mikor volt bekapcsolva a motor. Ha bekapcsolt állapotban van, akkor ábrázoljunk 1-et, ha kikapcsolt állapotban van akkor 0-t.

Kód minta:

Temperature\_functions\_studentVersion.py:

# register on cloud4rpi

# kiralysandor123@gmail.com

# sudo pip install cloud4rpi

# https://github.com/cloud4rpi/cloud4rpi-raspberrypi-python

# https://cloud4rpi.io/raspberry-pi-projects/humidity-in-the-cloud

# https://cloud4rpi.io/s/2CWE6pX2N

# -\*- coding: utf-8 -\*-

from time import sleep

import sys, time

import cloud4rpi

import rpi

import adafruit\_dht

from gpiozero import Motor

from temperature\_functions import get\_hum, get\_temp, ventillation

# Put your device token here. To get the token,

# sign up at https://cloud4rpi.io and create a device.

DEVICE\_TOKEN = 'B7uUHBqu6SCPayDjgbj1hNUHc'

# Change these values depending on your requirements.

DATA\_SENDING\_INTERVAL = 3  # secs

DIAG\_SENDING\_INTERVAL = 3  # secs

POLL\_INTERVAL = 0.5  # 500 ms

dev = adafruit\_dht.DHT11(3)

motor = Motor(23,24,25)

last\_update = time.time()-10

def get\_motor\_status(m):

    # visszaadott ertek , hogy a motor forog e vagy sem, azaz a motor is\_active attributuma (ami vagy True vagy False)

def main():

    # Put variable declarations here

    # Available types: 'bool', 'numeric', 'string', 'location'

    variables = {

        'DHT11 Temp': {

            'type': 'numeric' ,

            'bind': lambda : get\_temp(dev),

        },

        'DHT11 Humidity': {

            'type': 'numeric' ,

            'bind': lambda : get\_hum(dev),

        },

        'Motor On': {

            'type': 'bool',

            'value': False,

            'bind': lambda : get\_motor\_status(motor)

        },

        'CPU Temp': {

            'type': 'numeric',

            'bind': rpi.cpu\_temp

        },

    }

    diagnostics = {

        'CPU Temp': rpi.cpu\_temp,

        'IP Address': rpi.ip\_address,

        'Host': rpi.host\_name,

        'Operating System': rpi.os\_name,

        'Client Version:': cloud4rpi.\_\_version\_\_,

    }

    device = cloud4rpi.connect(DEVICE\_TOKEN)

    # Use the following 'device' declaration

    # to enable the MQTT traffic encryption (TLS).

    #

    # tls = {

    #     'ca\_certs': '/etc/ssl/certs/ca-certificates.crt'

    # }

    # device = cloud4rpi.connect(DEVICE\_TOKEN, tls\_config=tls)

    try:

        device.declare(variables)

        device.declare\_diag(diagnostics)

        device.publish\_config()

        # Adds a 1 second delay to ensure device variables are created

        sleep(1)

        data\_timer = 0

        diag\_timer = 0

        while True:

            if data\_timer <= 0:

                tt = # merd a homersekletet a get\_temp fuggveny hasznalataval

                hum = # merd a paratartalmat a get\_hum fuggveny hasznalataval

                # hivd meg a ventillation fuggvenyt, ami eldonti, hogy be kell e kapcsolni a motort vagy sem

                device.publish\_data()

                data\_timer = DATA\_SENDING\_INTERVAL

            if diag\_timer <= 0:

                device.publish\_diag()

                diag\_timer = DIAG\_SENDING\_INTERVAL

            sleep(POLL\_INTERVAL)

            diag\_timer -= POLL\_INTERVAL

            data\_timer -= POLL\_INTERVAL

    except KeyboardInterrupt:

        cloud4rpi.log.info('Keyboard interrupt received. Stopping...')

    except Exception as e:

        error = cloud4rpi.get\_error\_message(e)

        cloud4rpi.log.exception("ERROR! %s %s", error, sys.exc\_info()[0])

    finally:

        sys.exit(0)

if \_\_name\_\_ == '\_\_main\_\_':

    main()