**Kuidas on võimalik vaadata kodus arvutiga või telefonist oma toa õhuniiskust ja temperatuuri.**

1. Sissejuhatus.
2. Töö alustamine
3. Pythoni veebirakendus
4. Õhuniiskuse ja temperatuuri jälgimine Pytoni abil
5. Veebirakendus õhuniiskuse ja temperatuuri jälgimiseks
6. Graafilised näidikud.

# Sissejuhatus

1. Algse idee kohaselt, mis tuli aastal 2006, kasutati Raspberry PI's Atmeli ATmega644 mikrokontrollerit. Raspberry PI Foundationi eestkostja Eben Upton pani kokku grupi õpetajaid, akadeemikuid ja arvutientusiaste, et välja mõelda arvuti, mida lapsed kasutada saaksid. Praegu on saada erineva välimuse ning suurusega „vaarikaid“ erinevas versioonis. Riistvara kohta saame teada:

<https://www.raspberrypi.com/products/>

Energiasäästlik Raspberry PI töötab vaikselt ja kasutab palju vähem energiat kui teised arvutid.

Tänaseks on sellest õpilastele mõeldud ideest kasvanud võimas ja säästlik väikearvuti, millega saab väga huvitavaid projekte teha. Nendega saate tutvuda ka <https://projects.raspberrypi.org/en>

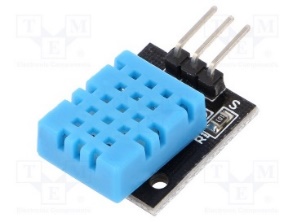
Uue Raspberry PI 4 kiirus ja jõudlus on varasemate mudelitega võrreldes märksa suurem. Sellega saab redigeerida dokumente, sirvida veebi samal ajal kasutada arvutustabeleid või koostada esitlusi.

# Mida vaja on?

Selle materjali põhjal saab teha väikese veebilehe, millelt me näeme toas olevat temperatuuri ja niiskustaset.

Kõigepealt peame poes käima ja endale muretsema Raspberry PI, toiteploki, 8GB Micro SD kaardi (võib ka suurem olla) ja anduri DHT11 ja mõned juhtmed. Allpool on toodud juhendi koostamise ajal toimivad viited.

<https://www.oomipood.ee/product/oky3068_1_temperatuuri_ja_niiskuse_dht11_moodul>

See andur on mõeldud niiskustaseme ja temperatuuri mõõtmiseks.

<https://www.oomipood.ee/product/rpi_starter_8gb_raspberry_pi_4_starter_kit_1_5ghz_4gb>

 Komplekt sisaldab vajalikud vidinad komplektina.

<https://www.oomipood.ee/product/oky0060_maketeerimislaua_juhtmed_40tk_ema_ema_20cm_2_54mm>

Juhtmed on vajalikud anduri ja Rasperry Pi omavaheliseks ühendamiseks.

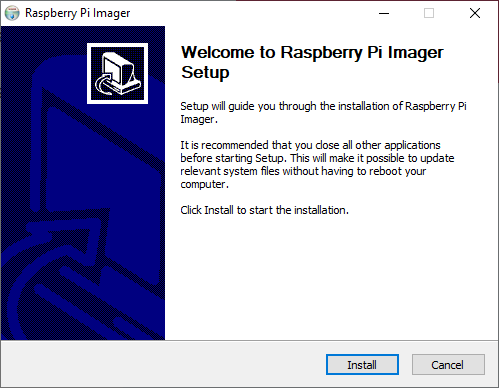
Lisaks on vaja veel USB klaviatuuri, USB hiirt ning monitori.

# Töö alustamine

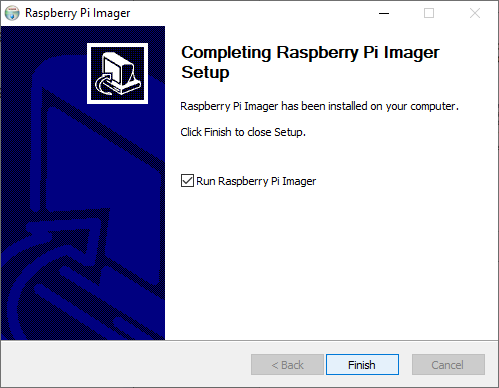
SD-kaardile, mis on tinglikult meie arvuti kõvaketas, tuleb paigaldada operatsioonisüsteem. Operatsioonisüsteem käivitatakse arvutis alglaadimisprogrammi poolt ning see juhib arvutisüsteemi tööd ja teenindab rakendusprogramme. Seda on vaja, et saaksime oma uues arvutis huvitavaid projekte teha. Selleks on vahendid <https://www.raspberrypi.com/software/>

Alustame Raspberry PI OS installeerimisega kasutades Raspberry PI Imagerit. Selleks on meil vaja tavalist arvutit, millel on olemas microSD kaardipesa.

Kui teie arvutis on MS Windows, siis valige „Download for Windows“. Teie arvutisse laetakse imager\_1.7.3.exe. Käivitage! Sarnaselt saab toimida ka Linuxi ja machos arvutite puhul.



Vajutage „Finish“



Sisestage oma arvuti kaardipesasse microSD kaart ning valige **kõige ülemine** valik, siis saate kohe kõik uuendused kaasa.

Pilt, millel on kujutatud tekst

Kirjeldus on genereeritud automaatselt





Küsimused:

Kellele on mõeldud Raspberry PI arvutid?

Miks on vaja operatsioonisüsteemi?

Milline on Raspberry PI „kõvaketas“?

Mille abil on võimalik SD-kaardile Opsüsteemi laadida?

Peale programmi töö lõpetamist võtke microSD kaart arvuti kaardipesast välja ja sisestage Raspberry Pi kaardipesasse. Seejärel võib Raspberry Pi sisse lülitada. Raspberry Pi arvutil ei ole eraldi sisselülitamisnuppu. Kui te ühendate toitejuhtme, siis Raspberry Pi lülitub kohe sisse.

Kui kõik läks hästi, siis peaks teie ekraanile ilmuma RB Pi avaleht. RB Pi-s kasutatav operatsioonisüsteem on Linux.

Kuigi eeldatavasti said kaardile paigalduse käigus vajalikud uuendused paigaldatud, siis enne Raspberry PI programmeerimise alustamist on mõistlik kontrollida, et meil on tarkvara värskendatud. Selleks on meil vaja internetiühendust kas siis WiFi või kaabli kaudu. Kaabelühenduse jaoks ei ole tõenäoliselt mingeid seadistusi vaja teha, WiFi võib vajada mõningast seadistust võrgunime ja parooli sisestamisega.

Keskkonna uuendamiseks on meil vajalik kasutada terminali. Käivitame oma käsud, kasutades sudo’t. Et meil oleks administraatori õigused siis on vajalik käskude ees kasutada sõna ’sudo’

Värskendame nii pakettide loendit kui ka installitud pakette:

sudo apt-get update -y

sudo apt-get upgrade -y

Kasutame Pythoni programmeerimiskeelt oma niiskusanduriga suhtlemiseks. Ilmselt on see juba RB Pi-s paigaldatud, kuid kui seda ei ole, siis tuleb toimida allpoololeva kohaselt. Kas Python on juba olemas, saab kontollida, kui terminali käsurealt sisestate ’python’ või ’python3’. Käsureale peab ilmuma tekst Pythoni versiooniga ja ’>>>’ märgid. Kui need ilmusid, siis tagasi saab Ctrl-Z ja Enteri vajutamisega. Kui Pythonit ei ole, siis tuleb toimida nii:

Installime nii python3 kui ka python3-pip käivitades alloleva käsu.

sudo apt-get install python3-dev python3-pip

Kasutades pipi jätkame ja installime Adafruiti DHT teegi Raspberry PI-sse.

Kasutame seda Pythoni teeki oma DHT11 niiskus-/temperatuurianduriga suhtlemiseks.

Boonusena toetab see teek (moodul) ka DHT22 ja AM2302 niiskuse/temperatuuri andureid, mis teeb sellest suurepärase vahendi erinevate andurite kasutamise õppimiseks.

DHT-teegi installimiseks oma Raspberry PI-sse käivitage järgmine käsk:

pip3 install adafruit-circuitpython-dht

sudo apt-get install libgpiod2

Kui oleme oma RB PI-sse installinud Adafruit DHT teegi, saame nüüd Pytoni abil õhuniiskuse ja temperatuuri andurilt lugeda andmeid.

Küsimused:

Kuidas uuendada tarkvara?

Miks on meil vaja sudo’t?

Millise käsuga installida python3 ja pyton-pip?

Mille abil installitakse teeke?

# Raspberry Pi seadistamine

Kui me tahame teha Pythonis veebirakendust, on mõistlik kasutada Flaski-nimelist raamistikku, see hõlmab endas ka lihtsa veebiserveri. Ka Flask võib juba olla RB Pi—sse paigaldatud, seda saab kontrollida käsuga

pip list

Kui Flaski ei ole, siis toimime alljärgnevalt:

Kasutame pipi, et installida Flask meie Raspberry PI-sse, käivitades alloleva käsu:

sudo pip install flask

Kui ülaltoodud kood on edukalt käivitatud, tähendab see, et Flask on installitud. Nüüd käivitame testprogrammi ja kontrollime flaski veebiserverit.

# Testime Flaski paigaldust

Looge fail nimega " flask\_test.py " ja kopeerige allolev kood ning kleepige see faili ja seejärel salvestage. Programmide kirjutamiseks soovitame kasutada RB Pi-s olevat programmi ’Geany’, aga võite omal valikul ka teisi programme kasutada nagu Thonny või konsoolis käivituv ’nano’:

nano flask\_test.py

Käesoleva juhendi järgi võite kõik failid paigutada näiteks oma kodukataloogi.

**# Loeme sisse flask mooduli:**

from flask import Flask

**# Loome Flask’i rakenduse**

app = Flask(\_\_name\_\_)

**# Kui olete oma veebilehitseias Raspberry PI IP-aadressi sisestanud, käivitub allolev kood:**

@app.route("/")

def main():

return "Tere maailm!"

**#kui koodi käivitatakse terminalist**

if \_\_name\_\_ == "\_\_main\_\_":

**# Server ’kuulab’ porti 80 ja teatab kõigist vigadest.**

app.run(host='0.0.0.0', port=80, debug=True)

## Saame sellise koodi (Pythonis on olulised rea alguse taanded:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

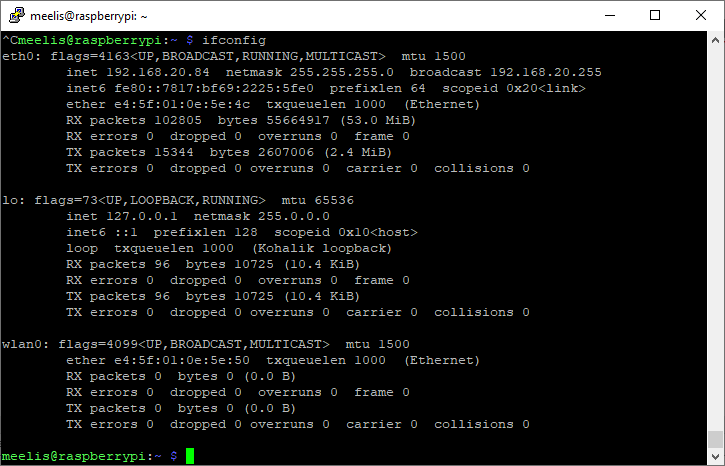
def main():

return "Tere, maailm!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=80, debug=True)

Käsuga „ifconfig“ saate terminalis teada oma arvuti ip aadressi:

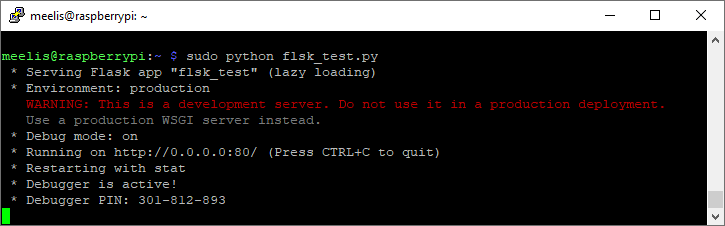


Siin on ip aadress 192.168.20.84

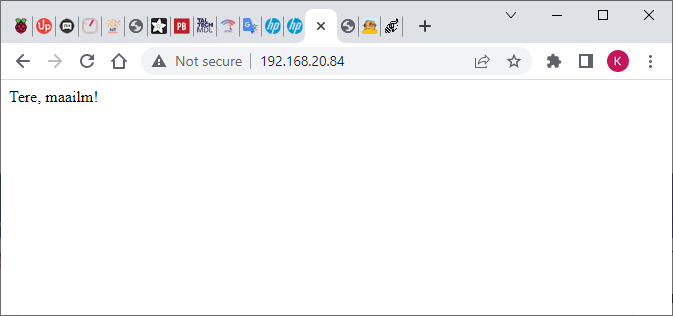
Edasi käivitage Raspberry PI terminaliaknas fail nimega " flask\_ test.py " alloleva käsuga:

sudo python flask\_test.py

Pärast selle käsu täitmist peaksite nägema terminalis kirjutatud ridu:



Kui me nüüd veebilehitseja reale kirjutame oma ip adressi siis näeme:



Küsimused:

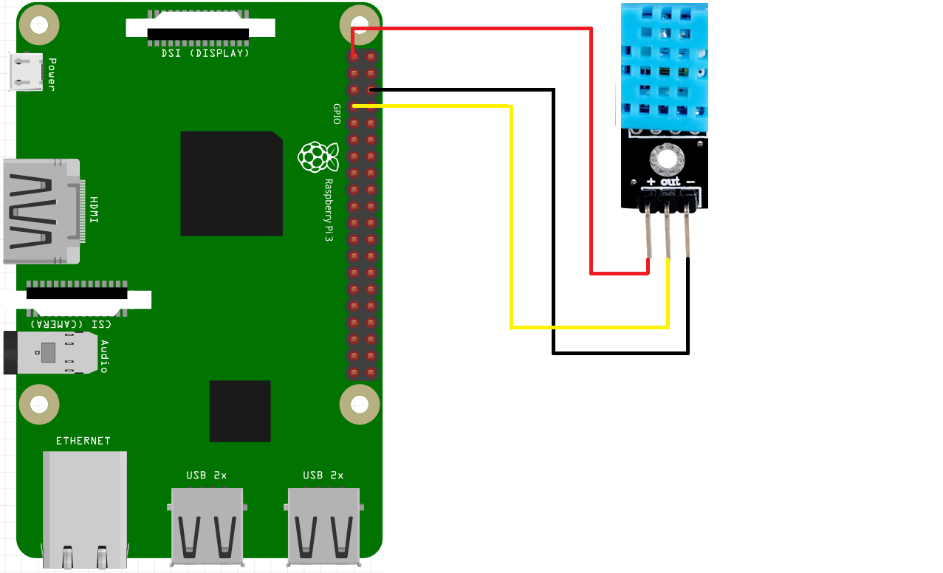
Milline on pythoni programmi laiend?

Kuidas saada teada oma arvuti IP aadress?

Millise käsuga käivitada pythoni programm

# Anduri ühendamine

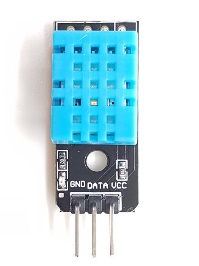
Järgmisena proovime oma Raspberry-Pi lugeda õhu temperatuuri ja niiskust. Skeemi kokku seadmiseks lülitame oma arvuti välja.



<https://github.com/rnieva/Playing-with-Sensors---Raspberry-Pi>

Paneme kokku oma elektriskeemi nagu pildil näha on. Alustame anduri ühendamisega.

See diagramm peaks aitama aru saada, kuhu Raspberry PI iga kontakt peab minema.



Pin 1 on VCC (toiteallikas)

Pin 2 on DATA (andmesignaal)

Pin 3 on GND (maandus)

Anduri ühendame vastavalt ülalolevale joonisele. Anduri jala 1 ühendame RB Pi kontaktiga 1, keskmise jala RB PI sama rea neljanda kontaktiga ja jala 3 ühendame ülemise rea kolmanda kontaktiga.

# Programmeerimine

Teeme endale Pythoni programmi, millega saame lugeda temperatuuri ja õhuniiskust.

Selle faili loomise alustamiseks võite oma Raspberry PI-sse sisestada järgmise käsu:

nano katse.py

Programmi kood:

import time

import board

import adafruit\_dht

dhtDevice = adafruit\_dht.DHT11(board.D4)

while True:

try:

temperature\_c = dhtDevice.temperature

humidity = dhtDevice.humidity

print(

"Temp: {:.1f} C Humidity: {}% ".format(

temperature\_c, humidity

)

)

except RuntimeError as error:

print(error.args[0])

time.sleep(2.0)

continue

except Exception as error:

dhtDevice.exit()

raise error

time.sleep(2.0)

Selgitame, mida programm peaks tegema:

Vajalike lisamoodulite paigaldamine

import time – operatsioonid ajaga

import board – operatsioonid kontaktidega

import adafruit\_dht – operatsioonid DHT anduritega

Lisaks saab lugeda ka <https://pydoc.pages.taltech.ee/introduction/module.html>

Nüüd seadistame vastava anduri tüübi, mis on kasutusel ja millise kontaktiga ühendatud:

dhtDevice = adafruit\_dht.DHT11(board.D4)

Sellel real määratleme anduri seaded, mida soovime DHT teegi jaoks kasutada. Adafruit DHT teegil on kolm valikut, mille saab valida sõltuvalt kasutatavast andurist.

Meie valisime selles juhendis DHT11 anduri, seega määratlesime oma andurina "adafruit\_dht.DHT11".

*Kui kasutate kas DHT22 või AM2302 andureid, võite kasutada vastavalt kas ’adafruit\_dht.DHT22’ või ’adafruit\_dht.AM2302’.*

Sulgudes määrame (GPIO) kontakti numbri, millega anduri andmesignali väljund on Raspberry PI-ga ühendatud (Käesoleval juhul kontakt 4).

*Kui kasutasite teist GPIO kontakti, muutke seda numbrit vajadusel.*

Järgmisena käivitame pideva (lõputu) tsükli, kasutades lauset "while TRUE: ". See tsükkel töötab seni, kuni kasutaja Pythoni programmi seiskab.

Iga tsükli alguses loeme andurilt niiskuse ja temperatuuri näidud

while True:

try:

temperature\_c = dhtDevice.temperature

humidity = dhtDevice.humidity

Trükime andmed ekraanile vastavas formaadis:

print(

"Temp: {:.1f} C Humidity: {}% ".format(

temperature\_c, humidity

)

)

Juhul, kui andmete lugemine ei peaks õnnestuma (vigu juhtub üsna sageli, DHT-andureid on raske lugeda), jätkake. Veahalduse kohta saab rohkem lugeda: <https://www.w3schools.com/python/gloss_python_error_handling.asp>

except RuntimeError as error:

print(error.args[0])

time.sleep(2.0)

continue

except Exception as error:

dhtDevice.exit()

raise error

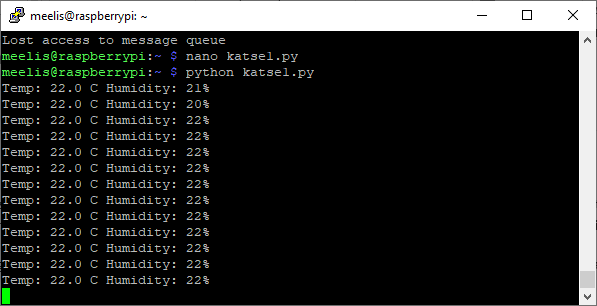
Teeme väikese pausi 2s.:

time.sleep(2.0)

Salvestame ja käivitame käsurealt:

python katse.py

Näeme terminalis järgmist pilti:



Küsimused:

Miks me kasutame oma programmis „while True“?

Kui me tahame et programm ootaks järgmise käsu toitmisel 5sekundid, siis millise käsu me peame kirjutama?

# Veebirakenduse loomine

Nüüd, kui oleme tutvunud DHT andurilt andmete lugemisega ning Flaski kasutamisega, teeme rakenduse, mis kuvab neid näitusid veebis. Kuna meil on nüüd tegemist suurema projektiga, siis teeme selle jaoks eraldi kataloogi. Selleks kasutame käsku mkdir ja paneme nimeks näiteks „katse“.

mkdir katse

Siseneme sinna käsuga:

cd katse

Kui juba kataloogide tegemiseks läks siis teeme veel kaks kataloog, mida me hiljem kasutame:

mkdir static templates

**Alustame pythoni koodi tegemisega:**

nano katse.py

Kirjutame koodi:

import adafruit\_dht

import board

import datetime

import time

from flask import Flask, render\_template

dhtDevice = adafruit\_dht.DHT11(board.D4)

app = Flask(\_\_name\_\_)

@app.route("/")

def katse():

now = datetime.datetime.now()

timeString = now.strftime("%d -%m-%Y %H:%M")

humidity = dhtDevice.humidity

temperature = dhtDevice.temperature

templateData = {

'time' : timeString,

'temperature' : temperature,

'humidity' : humidity

}

return render\_template('index.html', \*\*templateData)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=8080)

Seletame selle lahti:

Alustame jälle teekide (moodulite) laadimisega. Seekord lisame ka teegi kellaaja ja kuupäeva kuvamiseks.

import adafruit\_dht

import board

import datetime

import time

Loeme sisse flask mooduli:

from flask import Flask, render\_template

Nüüd seadistame DHT seadme milline andur on kasutusel ja millisele kontaktiga ühendatud:

dhtDevice = adafruit\_dht.DHT11(board.D4)

Loome Flask’i rakenduse

app = Flask(\_\_name\_\_)

Kui olete oma veebilehitseias Raspberry PI IP-aadressi sisestanud, käivitub allolev kood:

@app.route("/")

def katse():

Loeme kuupäeva ja kellaaja

now = datetime.datetime.now()

Näitame siis vastavas järjekorras : päev-kuu-aasta ja siis tund:minut

timeString = now.strftime("%d -%m-%Y %H:%M")

Loeme niiskuse:

humidity = dhtDevice.humidity

Loeme temperatuuri:

temperature = dhtDevice.temperature

Näitame siis seda vastavalt:

templateData = {

'time' : timeString,

'temperature' : temperature,

'humidity' : humidity

}

Pöördume veebis kuvamiseks html faili poole index.html

return render\_template('index.html', \*\*templateData)

koodi käivitamine terminalist:

if \_\_name\_\_ == "\_\_main\_\_":

Server kuulab porti 8080

app.run(host='0.0.0.0', port=8080)

Seekord kasutame porti 8080, nii ei pea me administraatori õigustes (sudo) programmi käivitama.

Teeme faili index.html (kataloogi templates, mis on juba enne tehtud):

nano templates/index.html

<!DOCTYPE html>

<head>

<title>Raspberry ilmajaam</title>

</head>

<body>

<h1>Kuupäev ja kellaeg on: {{ time }}</h1>

<h2>Temperatuur on: {{ temperature }}</h2>

<h2>Niiskus on: {{ humidity }}</h2>

</body>

</html>

Kuvame kuupäev ja kellaja mis on järgmistel ridadel. Täpsemad juhised saate <https://www.w3.org/MarkUp/>

Kõike mis on suletud topelt loogelistesse sulgudsse käsitletakse muutujana, mis edastatakse meie pythoni skriptile.

**Käivitame oma projekti käsuga:**

python katse.py

Vaatamiseks sisestame oma veebilehitsejasse oma ip ja sinna järgi kooloni ning pordinumbri. Näiteks nii:

<http://192.168.0.91:8080>

Pilt, millel on kujutatud tekst

Kirjeldus on genereeritud automaatselt

Nüüd teeme selle ka värvilisemaks kasutades stiilifaili (css):

nano static/style.css

teeme väga lihtsa kujunduse, millega tausta teeme siniseks ja teksti kollaseks:

body {

background: blue;

color: yellow;

}

Ja selleks et seda teaks ka meie veebi leht lisame faili index.html lause:

<link rel="stylesheet" href="../static/style.css/">

Siis on meie fail:

<!DOCTYPE html>

<head>

<title>Raspberry ilmajaam</title>

<link rel="stylesheet" href="../static/style.css/">

</head>

<body>

<h1>Kuupäev ja kellaeg on: {{ time }}</h1>

<h2>Temperatuur on: {{ temperature }}</h2>

<h2>Niiskus on: {{ humidity }}</h2>

</body>

</html>

Käivitame uuesti oma projekti käsuga:

python katse.py

Vaatame uuesti oma veebilehte ja näeme:

Pilt, millel on kujutatud tekst

Kirjeldus on genereeritud automaatselt

Küsimused:

Kuidas edastada html failist andmed pythoni skriptile?

Millise laiendiga on kujundus failid

Millal on vaja kasutada html laiendiga tekst faili?

# Graafilised näidikud

Jätkame oma veebilehe nägusamaks muutmisega, selleks on palju võimalusi, siinkohal kasutame selleks javacripti.

Teeme uue projektikausta ja paneme sellele nimeks SensorsWeb ning lisame sinna sisse kohe kaustad ’static’ ja ’templates’

SensorsWeb kausta teeme uue faili ja paneme talle nimeks ’appDHTWeb.py’:

import adafruit\_dht

import board

import datetime

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

dhtDevice = adafruit\_dht.DHT11(board.D4)

def getDHTdata():

now = datetime.datetime.now()

timeString = now.strftime("%Y-%m-%d %H:%M:%S")

temperature = dhtDevice.temperature

humidity = dhtDevice.humidity

if humidity is not None and temperature is not None:

hum = humidity

temp = temperature

return timeString, temp, hum

@app.route("/")

def index():

time, temp, hum = getDHTdata()

templateData = {

'time' : time,

'temp' : temp,

'hum' : hum

}

return render\_template('index.html', \*\*templateData)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=8080, debug=False)

Varasemast koodist erineb see peamiselt sellevõrra, et andmete saamine andurilt on tõstetud eraldi funktsiooni ’getDHTdata()’.

Kausta templates lisame uue faili index.html:

<!DOCTYPE html>

<head>

<title>Raspberry Pi ja andurid</title>

<link rel="stylesheet" href='../static/style.css'/>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<style>

body {

text-align: center;

}

#g1,

#g2 {

width: 200px;

height: 160px;

display: inline-block;

margin: 1em;

}

</style>

</head>

<body>

<h1>Raspberry Pi andurite andmed</h1>

<div id="g1"></div>

<div id="g2"></div>

<hr>

<h3> Viimane lugemiaeg: {{ time }} ==> <a href="/"class="button">Värskenda</a></h3>

<script src="../static/raphael-2.1.4.min.js"></script>

<script src="../static/justgage.js"></script>

<script>

var g1, g2;

document.addEventListener("DOMContentLoaded", function(event) {

g1 = new JustGage({

id: "g1",

value: {{temp}},

valueFontColor: "yellow",

min: -10,

max: 50,

title: "Temperatuur",

label: "Celcius"

});

g2 = new JustGage({

id: "g2",

value: {{hum}},

valueFontColor: "yellow",

min: 0,

max: 100,

title: "Niiskustase",

label: "%"

});

});

</script>

</body>

</html>

Faili teises pooles näeme javascripti ja viiteid javaskripti moodulitele. Vajalikud javascripti moodulid saame alla laadida <https://toorshia.github.io/justgage/> Allalaetavas faili on palju faile, sh. ka näidiseid, kuid meid huvitavad kaks faili: ’justgage.js’ ja ’raphael-2.1.4.min.js’. Paigutame need oma kausta ’static’. Kausta ’static’ lisame ka stiilifaili ’style.css’:

body{

background: blue;

color: yellow;

padding:1%

}

.button {

font: bold 15px Arial;

text-decoration: none;

background-color: #EEEEEE;

color: #333333;

padding: 2px 6px 2px 6px;

border-top: 1px solid #CCCCCC;

border-right: 1px solid #333333;

border-bottom: 1px solid #333333;

border-left: 1px solid #CCCCCC;

}

img{

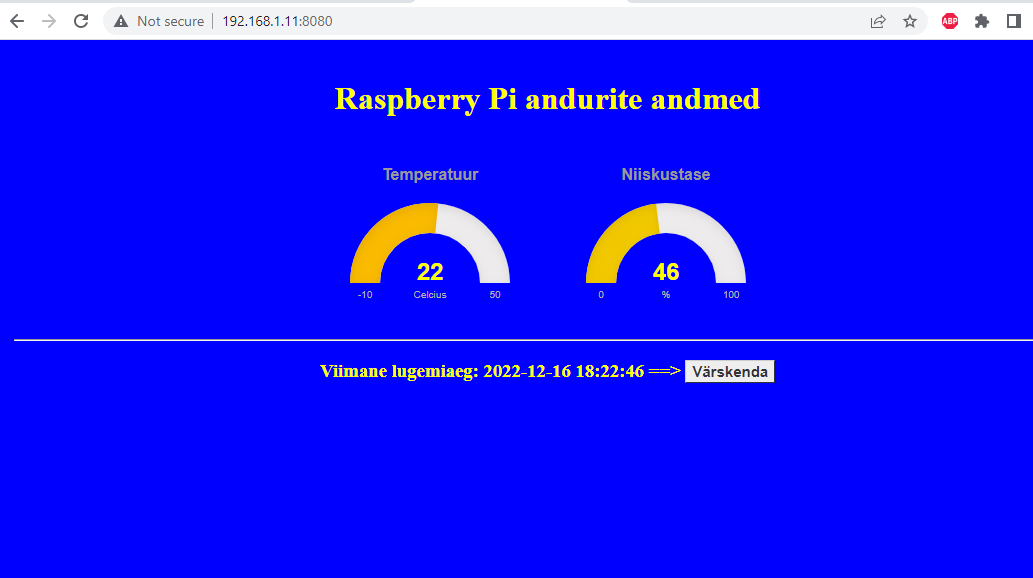
display: display: inline-block

}

Käivitame koodi käsuga:

python appDHTWeb.py

Kui kõik toimib, siis võiks avaneda oma ip-aadressi sisestamisel umbes selline pilt:



Kuidas saate veel oma lehe välimust muuta saate lugeda <https://courses.cs.ut.ee/MTAT.03.132/2013_spring/uploads/Main/CSS%20praktikumi%20juhend>

<https://www.w3.org/TR/CSS21/selector.html>

Küsimused:

Leidke vastus mida teeb rida border-top: 1px solid #CCCCCC;?