17.2.2016 Kari Naakka v1.0

**Tietoa Raspberry serverista**

**1. Yleistä:**

Toteutettu Pythonilla ver 2.7.x

Root käyttäjätunnus **pi**

Salasana **kn13**

Python ohjelma sijaitsee hakemistossa **/home/pi/test2\_ICT/**

Alkuperäisen ohjelman nimi **main18.py**

Ajossa olevan ohjelman tappaminen **CTRL C**

Ohjelman kopiointi **sudo cp main18.py oma1.py**

Ohjelman käynnistäminen **sudo python oma1.py**

Ohjelman editointi Nano editorilla **sudo nano oma1.py**

Nano editorin muutosten tallentaminen **CTRL X ja save mod buff > yes Y**

**File name to Write oma1.py** **<ENTER>**

Ohjelman käynnistäminen **sudo python oma1.py**

**2. Koodirivit**

Seuraavassa on alkuperäisen koodin lyhyet kommentit

urlb stringin lähetys serverille:

response =urllib2.urlopen( urlb ).read( )

import time,

KN 19.2.2016 v. 3.0

Aktivoidaan tarvittavat kirjastot:

Time, urllib2, time, sys, flask

paho mqtt client

Mittaustaulukko mit[] jonne mittaustiedot tallennetaan

Lähetys tamkin tietokantaan

Tulostetetaan aika terminaaliin 1 sekunnin välein

Muodostetaan muuttja minu josta voidaa tutkia parilliset ja parittomat minuutit, minu 0/1

Lähetetään kanavat 1-10 a-luokka kun aika on:  
minuutti = pariton  
sekunnit = 40

Muodostetaan kunkin mittauarvon tiedosta yksi ali stringi.

Esim. st1 on merkkijono, jossa tuulennopeuden numeerinen arvo on muutettu merkkijonoksi

Tulostetaan merkkijonot päätteelle

#

# ICT kurssin Raspi koodi 6.1.2016 KN

#

import paho.mqtt.client as mqtt

import time, datetime, threading

import urllib2, sys

from flask import Flask, render\_template , request

#

#

# 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3

# 0,1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0,1,2,3,4,5,6,7,8,9,0,1,2 3 4 5 mittaustiedot taulukko

mit = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

#

#

# Lahetys ajastus looppi lahetys 2 minut valein

#

def lahetys\_dbeen():

ltime = time.localtime(time.time())

print 'TIME =:',ltime.tm\_hour,':',ltime.tm\_min,'.',ltime.tm\_sec,"\r",

sys.stdout.flush()

sek=int(ltime.tm\_sec)

sek\_t=sek;

minu=int(ltime.tm\_min)

minu=minu % 2 # parilliset ja parittomat minuutit

sek\_t=sek\_t % 30

############################## kanavat 1-10 alku ##########################

if ((minu == 1) & (sek == 40 )): # paril minuuttu ja sekuntti = 40

#if sek == 1: # nopeasykli testukseen

print "\n"

print ' Mittaukset 1-12 Lahetys'

global mit

st1="%04.2f" % float(mit[1]/100.0) #tuulen nopeus 1

st2="%01d" % float(mit[2]/100.0) #tuulen suunta 1

st3="%04.1f" % float(mit[3]/100.0) #lampotila 1

st4="%04.1f" % float(mit[4]/100.0) #sademaara 1

st5="%04.1f" % float(mit[5]/100.0) #kosteus sisa

st6="%04.1f" % float(mit[6]/100.0) #kosteus ulko

st7="%04.2f" % float(mit[7]/100.0) #tuulen nopeus 2

st8="%01d" % float(mit[8]/100.0) #tuulen suunta 2

st9="%04.1f" % float(mit[9]/100.0) #lampotila 2

st10="%04.1f" % float(mit[10]/100) #sademaara 2

st11="%04.1f" % float(mit[11]/100) #ilmanpaine 1

st12="%04.1f" % float(mit[12]/100) #ilmanpaine 2

print 'T-nop 1=',st1,

print ' T-suun 1=',st2,

print ' Lamp 1=',st3,

print ' Sade 1=',st4,

print ' Kost S=',st5,

print ' Kost U=',st6

print 'T-nop 2=',st7,

print ' T-suun 2=',st8,

Lähetetään kanavat 21-32

Timerin käynnistys

MQTT osuus

on connect funktio

Aloitetaan tulleen MQTT sanoman

tulkitseminen

Rakennetaan kanavanumero

Rakennetaan mittausarvo

luetusta merkkijionosta

print ' Lamp 2=',st9,

print ' Sade 2=',st10,

print ' ilmp 1=',st11,

print ' ilmp 2=',st12

print

######################### kanavat 1-12 loppu #####################

######################### kanavat 21-32 alku #####################

######################### kanavat 21-32 loppu ####################

threading.Timer(1, lahetys\_dbeen).start()

lahetys\_dbeen()

#######################################################################

#### MQTT lukeminen

#######################################################################

def on\_connect(mosq, obj, rc):

print("rc: " + str(rc))

###############################

### MQTT sanoma tullut

##############################

def on\_message(mqttc, obj, msg):

global message\_in

print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload)),

message\_in = msg.payload

ch=0

m1=0

m=0

ch = ch + 10 \* (ord (message\_in[0])-48) + (ord(message\_in[1])-48)

print " CH :",

print ch,

if ((ch<34) & (ch>0) & (len(message\_in)>8)):

#

# Rakennetaan mittausarvo luvuksi MQTT sanomasta

#

print " Mit= ",

m1=int (message\_in[3])

m=m1 \* 100000

m1=int (message\_in[4])

m=m+ 10000 \* m1

m1=int (message\_in[5])

m=m+ 1000 \* m1

m1=int (message\_in[6])

m=m+ 100 \* m1

m1=int (message\_in[7])

m=m+ 10 \* m1

m1=int (message\_in[8])

m=m+ m1

# Tulosta mittausarvo lukuna

Tulostetaan mittausarvo

Talletetaan mittausarvo taulukkoon

mit[ ]

publish = ei käytetä tässä

MQTT clientin alustus

funktioiden callbackien asettaminen

määritellään mqtt serverin IP ja portti 1883

Aloitetaan MQTT sanomien kuuntelu

MQTT loopin käynnistys

http serverin asettaminen

ei käytetä tässä

HTPP porttinumeron asetus = 80

print m

mit[ch]=m

def on\_publish(mosq, obj, mid):

global message

#print("mid: " + str(mid))

mqttc = mqtt.Client()

mqttc.on\_publish = on\_publish

def on\_subscribe(mosq, obj, mid, granted\_qos):

print("Subscribed: " + str(mid) + " " + str(granted\_qos))

def on\_log(mosq, obj, level, string):

print(string)

# Assign event callbacks

mqttc.on\_message = on\_message

mqttc.on\_connect = on\_connect

mqttc.on\_subscribe = on\_subscribe

# Connect

mqttc.connect("localhost", 1883,60)

# Start subscribe, with QoS level 0

mqttc.subscribe("outTopic", 0)

# Publish a message

mqttc.publish("outTopic", "my message")

# Continue the network loop, exit when an error occurs

mqttc.loop\_start()

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

@app.route("/")

@app.route("/", methods=['POST'])

def mymenu():

return 1

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

app.run(host='0.0.0.0', port=80, debug=False) #Run flask app !