#

meteo-station obv BME280 Digitale Barometer Druk en Vochtigheid Sensor Module

#

bron: https://www.tinytronics.nl/shop/nl/sensoren/temperatuur-lucht-vochtigheid/bme280-digitale-barometer-druk-en-vochtigheid-sensor-module

Een zeer compacte barometer die werkt via I2C of SPI. De BME280 is een 3-in-1 module

die temperatuur, druk en vochtigheid kan meten.

#

De module kan alleen gevoed worden

met 3.3VDC. De I2C/SPI werkt dus ook met 3.3V en

---

je hebt dus een level converter nodig bij gebruik van bijv. een 5V Arduino Uno.

#

Het standaard I2C adres van deze module is 0x76. Dit moet mogelijk in de

---

voorbeeldcode/library veranderd worden van 0x77 naar 0x76. Indien je de SDO pin

---

verbind met Vcc, dan wordt het I2C adres 0x77.

#

(Arduino) project met de ESP2866 en BME280 (nuttig voor aansluitingen) is te vinden op

---

https://core-electronics.com.au/projects/thingspeak-temperature-pressure-logger

#

MicroPython library voor de BME280 en ESP2866 gevonden op GitHub:

https://github.com/robert-hh/BME280

#

BvH, 25-05-2019

#

LIBRARIES:

#

We start by importing the Pin class from the machine library, as this will enable us to use

the GPIO pins. We need to use a wait-time in the loop and import the sleep

function from the

time library.

```
from machine import Pin, I2C
import time
```

using mqtt for exchanging data

```
from umqttsimple import MQTTClient
#
```

this script assumes the default connection of the I2C bus

On pycom devuces that is P9 = SDA, P10 = scl

```
#
import bme280_float
```

Functies:

```
def do_blink(n=3):
# tripple blink
for x in range(n):
led.on()
time.sleep(0.5)
led.off()

def update_measurements():
```

```
# how to deal with a 'dict'?
# Example from https://www.tutorialspoint.com/python/python_dictionary.htm
# dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
# print "dict['Name']: ", dict['Name']
return(bme.values)
```

INITIALISATIE:

```
#
```

Next we create an object called led which will store the GPIO pin that we wish to use, and

whether it is an input or an output. In this case it is an output as we wish to light up the LED.

```
#
```

see pinout on https://escapequotes.net/esp8266-wemos-d1-mini-pins-and-diagram/

pin 16 = D0 (naar LED)

```
led = Pin(16, Pin.OUT)
```

show succesfull

do_blink()

##

Capacitieve vochtigheidsensor calibratie

##

values on right are inverse * 1000 values on left

dry air = 759 (0%) = 1.31752305665349143610013175231

water = 382 (100%) = 2.61780104712041884816753926702

The Difference = 1.30027799046692741206740751471

1 % = 0.0130027799046692741206740751471

initialiseer ADC op pin ADC0 (gpio2)

```
adc = machine.ADC(2)
```

show succesfull

```
do_blink()
```

Initialise the i2c interface. We use SCL-to-D1, SDA-to-D2.

pin 5 (= D1) SCL naar BME280-SCL.

pin 4 (= D2) SDA naar BME280-SDA.

```
i2c = I2C(sda=Pin(4), scl=Pin(5))
bme = bme280_float.BME280(i2c=i2c)
```

show succesfull

```
do_blink()
```

setup MQTT connection

```
def sub_cb(topic, msg):
print((topic, msg))
if topic == b'notification' and msg == b'received':
print('ESP8266-wijngaar-Achthoeven received a mqtt-message!')

def connect_and_subscribe():
```

```
global client_id, mqtt_server, topic_sub
client = MQTTClient(client_id, mqtt_server)
client.set_callback(sub_cb)
client.connect()
client.subscribe(topic_sub)
print('Connected to %s mqtt-broker, subscribed to %s topic' % (mqtt_server, topic_sub))
return client

def restart_and_reconnect():
print('Failed to connect to mqtt-broker. Reconnecting...')
time.sleep(10)
machine.reset()

try:
client = connect_and_subscribe()
except OSError as e:
print('Failed connecting to mqtt-broker. Error=' + e)
restart_and_reconnect()
```

show succesfull

```
do_blink()
```

All in an endless loop:

```
while True:
# retrieve BME280-measurements:
humPresTemp = update_measurements()
# show succesfull
do_blink(1)
# retrieve moisture measurement()
raw = adc.read()
print("moisture = " + raw)
SoilMoistVal = (((1 / adc.read())* 1000) / 0.0130027799046692741206740751471) - 101
if SoilMoistVal > 100:
SoilMoistVal = 100
if SoilMoistVal < 0:
SoilMoistVal = 0
# show succesfull
```

do_blink(1)
# show BME280-measurements

payload = values

```
print(humPresTemp)
# better version:
#values = read_compensated_data(result = None)

# once a minute, send a message with the data to the mqtt broker
try:
    client.check_msg()
    if (time.time() - last_message) > message_interval:
        msg = b'measurement #%d' % counter
```

msg = b'measurement #%d' + payload % counter

```
        client.publish(topic_pub, msg)
        last_message = time.time()
        counter += 1
except OSError as e:
    restart_and_reconnect()

# wait and measure approx. every 15 secs
time.sleep(measure_interval-0.5)
```