Building your own Energy consumption metrics

At the time of writing this entry, energy prices have been steadily increasing month over month in the Western world. Energy suppliers offer in their packages the option to track almost real time the energy consumption at a household, and the best, they don't charge for this. The next lines are intended for DIY lovers who prefer to use an in-house made solution (based on a Raspberry pi Zero) and does not require to broadcast your data to third parties.

Why measuring Energy consumption (close to) real-time?

Some research suggest that tracking consumption could lead to consumption reduction. This would happen by following the cause-effect model: increased feedback →increase awareness and knowledge →changes in energy use behavior →decrease in consumption [1]. This same study finds that smart meters can have a positive effect, increase the sense of control and empower the individual to take stronger actions, but also highlights issues that can arise:

- A household might not be willing to change consumption beyond some accepted baseline
- The device needs to be attractive
- Gender and age can have effects on the result

Other studies find that real-time feedback plus ease to consume plus high frequency can indeed lead to changes in behavior. The key is on keeping the monitoring front end simple and easy to use [2].

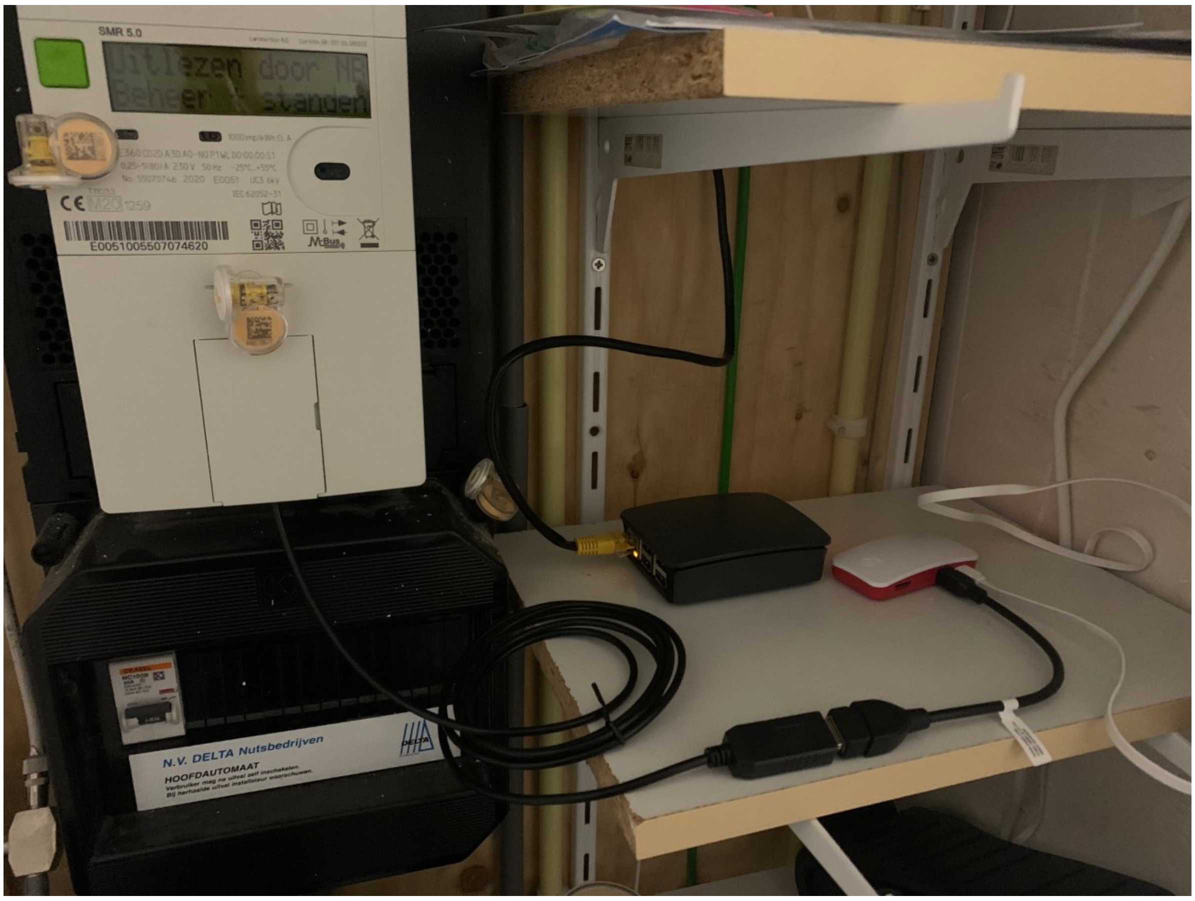
Dutch Smart Meters

More and more household meters in the Netherlands are smart meters. This allows energy companies to get access to the household consumption data without the need to manually capture it. These meters also come with the option to extract their data via a P1 cable.

It couldn't be without the help of Theo van der Sluijs in his post Read P1 port smart meter with Raspberry and Python and one post in the Domoticx site that I could get to the details on what libraries to install to get Python to read the high frequency telegrams that come from the smart meter. Theo van der Sluijs also gives a hint on how to setup a sqlite database in the pi to store the meter reads.

A way to read smart meter data

I will skip the details on how to setup the Raspberry Pi. In my installation I will be using a Raspberry Pi Zero 2W with Raspberry Pi OS. I bought the P1 cable via online reseller, and because the Pi Zero only receives mini USB I also needed to procure an adaptor USB-A to mini USB. The photo also displays a Pi in a black case but this is being used for another purpose, so please don't be mislead by this.



The final logic that I built is inspired on the posts mentioned above, however, it has many simplifications. The first portion of the code deals with the libraries that must be already installed in Python and loads them. Then the building of the dictionary that contains the codes of the lines that I want to capture from the telegram coming out. The telegram comes with 28 lines therefore this step is important so I only try to keep the relevant lines of the telegram.

```
#!/usr/bin/env python
import serial
import re
import sys
from datetime import datetime
import sqlite3

#dictionary of codes
obiscodes = {
    "0-0:1.0.0": "Timestamp",
    "1-0:1.8.1": "Rate 1 (day) - total consumption kWh",
    "1-0:1.8.2": "Rate 2 (night) - total consumption kWh",
    "1-0:2.8.1": "Rate 1 (day) - total production kWh",
    "1-0:2.8.2": "Rate 2 (night) - total production kWh",
    "0-1:24.2.1": "Gas Total Consumption m3",
    }
```

The next step in the code deals with the configuration of the serial parameters. Once the serial port is open, a count is initialize with 1 and a while loop iterates over each one of the 28 lines of one snapshot of the telegram. For each line it compares whether that line is one of the lines that I want to capture. If relevant, then its value is captured in a local variable. Once the iteration process is through the serial process is closed.

```
# ESMR 5.0
ser.baudrate = 115200
ser.bytesize=serial.EIGHTBITS
ser.parity=serial.PARITY_NONE
ser.stopbits=serial.STOPBITS_ONE
ser.xonxoff = 0
ser.rtscts = 0
ser.timeout = 12
ser.port = "/dev/ttyUSB0"

#Open serial port
try:
    ser.open()
except:
    sys.exit ("Could not open serial port"  % ser.name)

#read N number of lines
cont = 1
while cont < 28:
    p1_line = ''
    line_raw = ser.readline()
    line_str = str(line_raw)
    p1_line = line_str.strip()
    if re.search("0-0:1.0.0",p1_line):
        timestamp_str = p1_line[12:24]
        timestamp = datetime.strptime(timestamp_str,'%y%m%d%H%M%S')
    if re.search("1-0:1.8.1",p1_line):
        consumption_night = float(p1_line[12:22])
    if re.search("1-0:1.8.2",p1_line):
        consumption_day = float(p1_line[12:22])
    if re.search("1-0:2.8.1",p1_line):
        production_night = float(p1_line[12:22])
    if re.search("1-0:2.8.2",p1_line):
        production_day = float(p1_line[12:22])
    if re.search("0-1:24.2.1",p1_line):
        gas_consumption = float(p1_line[28:37])
        gas_timestamp_str = p1_line[13:25]
        gas_timestamp = datetime.strptime(gas_timestamp_str,'%y%m%d%H%M%S')

    cont = cont + 1
```

```
#Close port and show status
try:
    ser.close()
except:
    sys.exit ("Oops %s. Cannot close the serial port." % ser.name )
```

The final step of the codes deals with the incorporation of the local variables into a local database (stored in the Pi). In short, each variable is appended in a table along with the timestamp.

```
#Connect to db and insert reading
#steps to create db
conn = sqlite3.connect(db)
cur = conn.cursor()

#insert read values in table
cur.execute("insert into reads values(?,?,?,?,?,?,?)",
            (timestamp, consumption_day, consumption_night,
             production_day, production_night, gas_consumption,
             gas_timestamp))

#save and close db connection
conn.commit()
conn.close()
```

This routines is run every 5 minutes and is configured as a **crontab** job inserting the below line with crontab -e.

```
*/5 * * * * /home/pi/Documents/projects/meter_reading/read_serial_steps.py
```

Finally, once the sqlite database gets updated another crontab job dumps the main database table into a csv file. The reason for this is to have a fast and smooth of Pandas connecting to its data source for further data analysis. Within the same job, another instruction does a secure shell copy to an AWS instance

```
#!/bin/bash
sqlite3 -header -csv ~/Documents/projects/meter_reading/db/ac_meter_read.db
"select * from reads;" > ~/Documents/projects/meter_reading/db/data.csv
```

In summary, the above steps are executed to achieve the following results:

- Connect a Raspberry PI to the smart meter and take a snapshot of consumption data
- Process the snapshot and pick the relevant data points: electricity meter reading, gas meter reading, electricity production meter reading and a timestamp
- Append the readings to a local sqlite database
- Dump the single table from this database to a csv file
- Transfer this file to a cloud instance where further data processing will take place -- some comments to setup this instance below
- All the steps are automated via crontab jobs

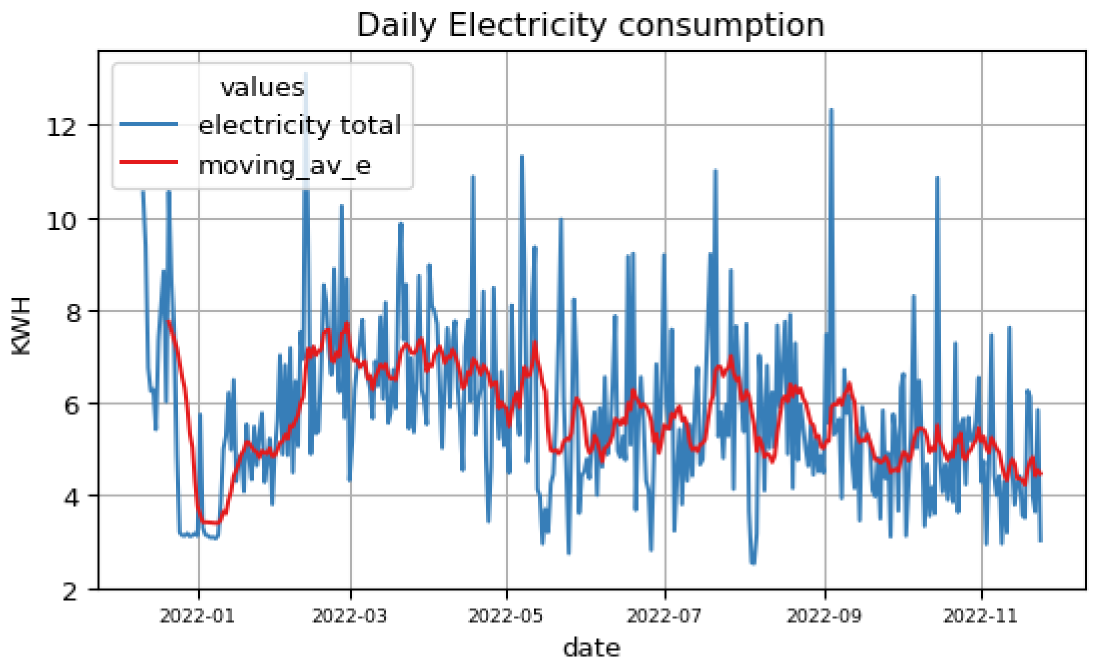
Setting up a AWS EC2 instance

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla vitae consectetur tellus. Integer tempor erat sed placerat blandit. Aliquam vitae dolor nulla. Nullam pulvinar felis ac diam pulvinar, nec sodales sem suscipit. Aliquam sed malesuada purus, mollis condimentum justo. Fusce condimentum arcu sem, et gravida enim dictum sed. Vivamus vulputate efficitur commodo. Donec porta eu lectus ac porta. Suspendisse at pellentesque leo. Sed eu euismod sem. Nulla scelerisque egestas arcu et elementum.

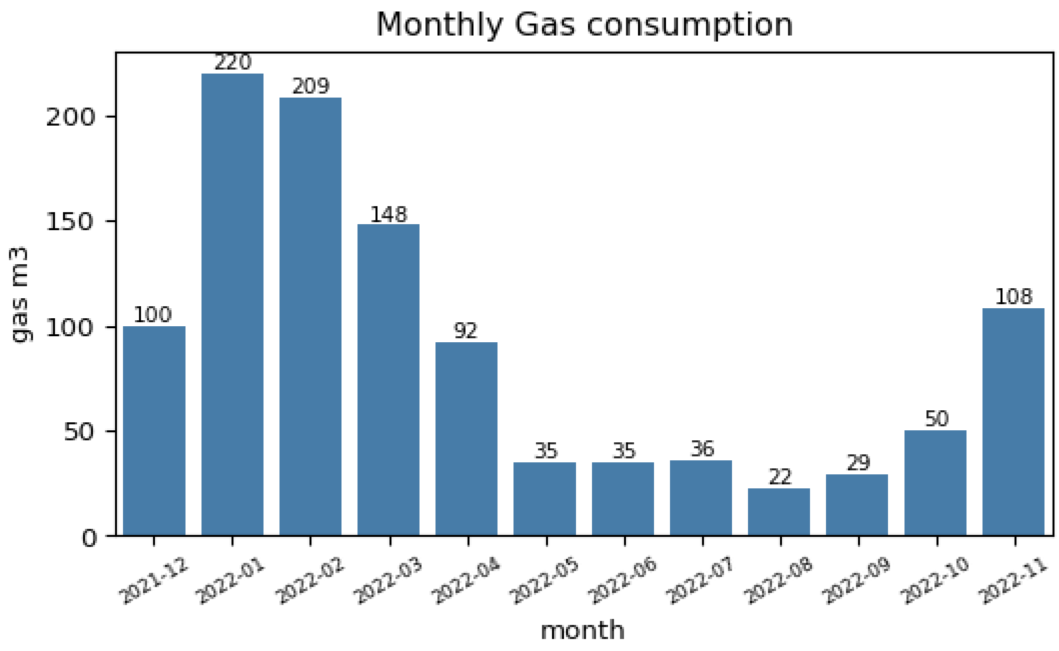
Sed rhoncus lorem id elit maximus, eget porta est fermentum. Etiam egestas, massa vitae aliquet scelerisque, libero felis suscipit turpis, ut molestie mauris metus pharetra mi. Vivamus scelerisque, mi non ultricies faucibus, augue tortor vulputate dui, sed ullamcorper turpis tellus eget felis. Aliquam condimentum elit dui, sed vestibulum mi lobortis et. Maecenas iaculis neque dui, sit amet ornare nulla facilisis et. Cras et dolor commodo, cursus metus ac, accumsan nulla. Donec ac molestie nisi.

Setting up a Shiny app with Python code

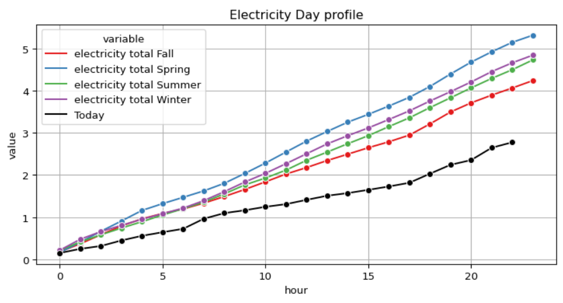
Integer eget metus condimentum augue laoreet fermentum. Cras consequat metus ante, fringilla lobortis ex fermentum ut. Curabitur in porta quam, eget gravida orci. Aenean euismod, diam ut convallis ultricies, purus urna consectetur tortor, ut tempus magna nunc a dolor. Sed dapibus ligula turpis, et finibus purus imperdiet a. Donec tempus quam nec dui bibendum, sit amet pulvinar libero maximus. Suspendisse placerat ullamcorper tincidunt. Sed sit amet odio ac purus rutrum sodales ut a sem. Suspendisse quis dui nulla. Vestibulum non vestibulum augue. Nulla ligula est, cursus in odio vel, imperdiet laoreet nibh. Etiam a nisl placerat, rutrum neque ac, semper lacus. Quisque posuere est quis nisl venenatis imperdiet.

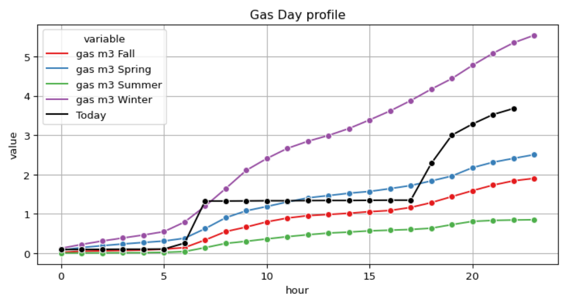


Morbi vitae eros nec magna ultricies feugiat. Mauris sit amet fermentum tortor. Integer nec dui laoreet, fermentum lacus vitae, commodo quam. Duis nec justo sit amet ante iaculis varius eget in nisl. Etiam sollicitudin, leo nec elementum pellentesque, risus dui efficitur mi, vitae tempor est ex at nulla. Mauris cursus et diam et placerat. Phasellus venenatis in dolor nec malesuada. Fusce at porttitor ipsum.

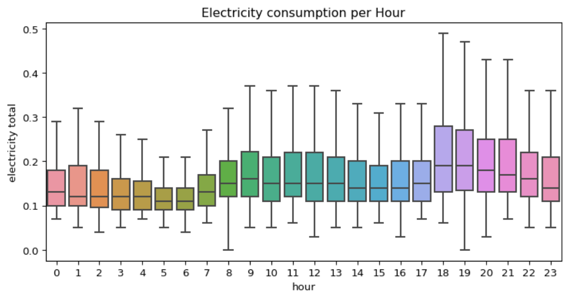


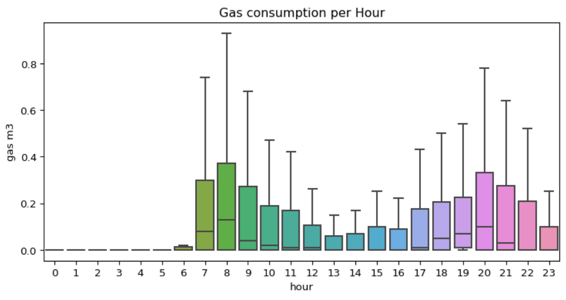
Integer eget metus condimentum augue laoreet fermentum. Cras consequat metus ante, fringilla lobortis ex fermentum ut. Curabitur in porta quam, eget gravida orci. Aenean euismod, diam ut convallis ultricies, purus urna consectetur tortor, ut tempus magna nunc a dolor. Sed dapibus ligula turpis, et finibus purus imperdiet a. Donec tempus quam nec dui bibendum, sit amet pulvinar libero maximus. Suspendisse placerat ullamcorper tincidunt. Sed sit amet odio ac purus rutrum sodales ut a sem. Suspendisse quis dui nulla. Vestibulum non vestibulum augue. Nulla ligula est, cursus in odio vel, imperdiet laoreet nibh. Etiam a nisl placerat, rutrum neque ac, semper lacus. Quisque posuere est quis nisl venenatis imperdiet.





Integer eget metus condimentum augue laoreet fermentum. Cras consequat metus ante, fringilla lobortis ex fermentum ut. Curabitur in porta quam, eget gravida orci. Aenean euismod, diam ut convallis ultricies, purus urna consectetur tortor, ut tempus magna nunc a dolor. Sed dapibus ligula turpis, et finibus purus imperdiet a. Donec tempus quam nec dui bibendum, sit amet pulvinar libero maximus. Suspendisse placerat ullamcorper tincidunt. Sed sit amet odio ac purus rutrum sodales ut a sem. Suspendisse quis dui nulla. Vestibulum non vestibulum augue. Nulla ligula est, cursus in odio vel, imperdiet laoreet nibh. Etiam a nisl placerat, rutrum neque ac, semper lacus. Quisque posuere est quis nisl venenatis imperdiet.





1. Hargreaves, T., Nye, M., & Burgess, J. (2010). Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors. Energy policy, 38(10), 6111-6119. ↩

2. Kobus, C. B., Mugge, R., & Schoormans, J. P. (2015). Long-term influence of the design of energy management systems on lowering household energy consumption. International Journal of Sustainable Engineering, 8(3), 173-185. ↩