Work in progress

Smarty reader

Project	Smarty reader
Description	A self-made In Home Display for the smart meter
Status	Planning
Contact	fantawams
Participants:	fantawams

Smart meter description

In the year 2016, all electrical or gas network operator (DSO) in Luxembourg started to install so called smart meters (Smarty). The smarty is a meter which can be read out remotely and it gives the client, so you, the possibility to read out you consumption or production. What you do with that possibility is up to you.

Since then, there is a lot of talk about that meter. Some curse it, some glamorize it and some are just interested of what they can do with it. Instead of believing some rumours and since you are reading this, you are able to take a more realistic look at it.

If you would like to know more about the smarty, simply ask your DSO or take a look the Luxmetering web-site.

Project description

What this project is about, is simply the P1 port. This port is the possibility for the client, so you, to read out his smarty over the P1 port. You can also create your own statistics out of that data or simply use it to double check your bill. The possibilities to use this data is totally up to your own imagination.

The basic solution will work 100% offline, **no Internet**, but for some optional features, a internet connection is required.

FAQ

Is this legal in Luxembourg?
YES, as we only read out the P1 port.

Are you hacking smarty?
NO, we are hackers not criminals.

Where do i get my P1 key?
Ask your DSO, they are obligated to provide it.

What data can i read out over the P1 port?

Here you go. Most of these values are all on the meter display.

Info	**Obis code**	**present in the project?**
Version for P1 output	1-3:0.2.8.255	YES, if needed
current date – time	0 - 0:1.0.0.255	YES
Logical device name	0 - 0:42.0.0.255	YES
Total imported energy register (P+)	1 - 0:1.8.0.255	YES
Total exported energy register (P-)	1 - 0:2.8.0.255	YES
Total imported energy register (Q+)	1 - 0:3.8.0.255	YES
Total exported energy register (Q-)	1 - 0:4.8.0.255	YES
Instantaneous imported active power (P+)	1 - 0:1.7.0.255	YES
Instantaneous exported active power (P-)	1 - 0:2.7.0.255	YES
Instantaneous imported reactive power (Q+)	1 - 0:3.7.0.255	YES, unit missing for now
Instantaneous exported reactive power (Q-)	1 - 0:4.7.0.255	YES, unit missing for now
Active threshold (SMAX)	0 - 0:17.0.0.255	YES, units missing for now and strange value
Breaker control state	0 - 0:96.3.10.255	YES, value is binary so 0 or 1
Number of power failures	0 - 0:96.7.21.255	YES
Number of voltage sags L1	0 - 0:32.32.0.255	YES
Number of voltage sags L2	0 - 0:52.32.0.255	YES
Number of voltage sags L3	0 - 0:72.32.0.255	YES
Number of voltage swells L1	0 - 0:32.36.0.255	YES
Number of voltage swells L2	0 - 0:52.36.0.255	YES
Number of voltage swells L3	0 - 0:72.36.0.255	YES
Long message E-meter	0 - 0:96.13.0.255	YES
Long message channel x	0 - 0:96.13.x.255	NOK, need to take a look at it.
Instantaneous current L1	1 - 0:31.7.0.255	YES, values are in A resulution
Instantaneous current L2	1 - 0:51.7.0.255	YES, values are in A resulution
Instantaneous current L3	1 - 0:71.7.0.255	YES, values are in A resulution
Instantaneous active power (P+) L1	1 - 0:21.7.0.255	
Instantaneous active power (P+) L2	1 - 0:41.7.0.255	
Instantaneous active power (P+) L3	1 - 0:61.7.0.255	
Instantaneous active power (P-) L1	1 - 0:22.7.0.255	
Instantaneous active power (P-) L2	1 - 0:42.7.0.255	
Instantaneous active power (P-) L3	1 - 0:62.7.0.255	
Instantaneous reactive power (Q+) L1	1 - 0:23.7.0.255	
Instantaneous reactive power (Q+) L2	1 - 0:43.7.0.255	
Instantaneous reactive power (Q+) L3	1 - 0:63.7.0.255	
Instantaneous reactive power (Q-) L1	1 - 0:24.7.0.255	
Instantaneous reactive power (Q-) L2	1 - 0:44.7.0.255	
Instantaneous reactive power (Q-) L3	1 - 0:64.7.0.255	
Device type channel x	0 - x:24.1.0.255	
Equipment Identifier channel x	0 - x:96.1.0.255	
Last Index capture - time channel x	0 - x:24.2.1.255	
Last Index gas channel x	0 - x:24.2.1.255	
valve position gas channel x	0 - x:24.4.0.255	

Info	Obis code	present in the project?
Last Index water channel x	0 - x:24.2.1.255	
Last Index heat channel x	0 - x:24.2.1.255	

For further information, please refer to the P1 specifications in the source section.

What feature works offline?
Here you go.

- reading out smarty
- storing data in a local data base
- creating the web-interface
- creating statistics

What feature is optional?
Here you go:

- encrypted emails
- xmpp bot
- message reports
- creating statistics
- integreating smartyreader into a smart home system

What can i do with this project?
Whatever you want to do with it.

Can i upload my data to a cloud?
If that is what you really want to do, than yes but it is definitely not part of this project.

Can i connect this solution to a smart home system?
Of course you can. Your data will be available in you network in a database using the MQTT protocol and some part of this project, is written in python.

Can you shown me how?
Yes, if i get to that point.

How will this work?

- The smarty streams every 10 seconds data out of the P1 port
- This data is by default encrypted
- We use a small PCB to decrypt it and send it to our network. That will be our broker
- Then we use a MQTT server in our network to store the data in a database using prometheus.
- We now use a RPI to create our webinterface with grafana. This RPI can be the same RPI with the MQTT server.

What you need to understand about this project!

- Internet is only needed for the setup, afterwards you only net a network
- You need your own P1 key for your smart meter. Ask your DSO to provide it. For example Creos.
- This project will be as open-source as possible. You don't need to pay for the software, PCB layout or any kind of digital information. If you like this project, you can always do a small

Donations. the more donation we get, the more projects we can do.
- Later on you can buy PCBs from us and also whole kits. In that case you will only pay for the material nothing more. You will not pay for anykind of digital information only for the hardware.
- You are responsible to secure your data in your network. If you need help, feel free to ask us.
- You can choose if you want to upload your data in a cloud or not. The basic functions of this solution, will work offline.
- For some feature internet is required, but these features are optional. You do not have to implement them if you want to run it completely offline.

Project stages

- get it working in the simplest way
- implement security on RPI
- setup prometheus
- setup grafana
- implement security on WEMOS
- add email encryption
- add encrypted XMPP bot messages
- be creative with your data
- tune the WEMOS code to use LAN and not WLAN
- tune the WEMOS code to add SD card storage
- tune the WEMOS code to use an RTC

Setup

Broker

Link to download the broker files Our broker is a WEMOS D1 mini Pro and the code is written in C++. To do this, use the arduino software.

To be able to flash the WEMOS, we still need a few plugins, these are easily added.

- In the arduino software under File/preference add http://arduino.esp8266.com/stable/package_esp8266com_index.json to Additional_Boards_Manager_URLs. Now under tools/Board/Board-manager find the **wemos d1 mini pro** and install the board. Next select under board the wemos d1 mini pro board.
- install a this crypto library
- install this mqtt library
- In the arduino software under tools check the following:
  - Flash size is 16M
  - CPU Frequency is 160MHz
  - Upload speed is 921600

Please change the following values in the code to match your network and smarty.

```
// Network parameters
// wemo static IP
IPAddress wemos_ip (000,000,000,000);
```

```
// DNS IP
IPAddress dns_ip (000,000,000,000);
// Gateway IP
IPAddress gateway_ip (000,000,000,000);
// Subnetmask
IPAddress subnet_mask(000,000,000,000);

// APN parameters
// APN name
const char ssid[] = "Your APN name";
// APN password
const char password[] = "Your super secret APN password";
// IP of your mqtt server
const char *mqtt_server = "000.000.000.000";

// your client ID
const char *clientId = "name for WEMOS on network";

//P1 Key for your smarty SAG10307xxxxxxxx
uint8_t key_SM_LAM_1[] = {0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx,
                          0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx, 0xxx};
```

It is extremely important to use the correct format for the P1 key in the code.

If your P1 Key, provided by your network operator, is for example: 000102030405060708090A0B0C0D0E0F, than the format in the code would be.

```
//P1 Key for your smarty SAG10307xxxxxxxx
uint8_t key_SM_LAM_1[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                          0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F};
```

After editing the parts to meet your requirements, the only step needed is to flash the WEMOS. Flash the WEMOS without connected to the PCB or it will not work.

RPI

Link to download the rpi part The RPI part is a bit tricky as you need to understand, that the RPI is just used to create the web-interface. The MQTT server can be installed on a RPI but does not need to be on it. It can be any machine in your network. How ether for this project, we will assume that the MQTT server is on the same machine as the RPI, which creates the web-interface. How to setup the MQTT server will also be explained here.

Needed

- Raspberry Pi
- 5Vdc Power supply for the Raspberry Pi
- a micro SD card for the Raspberry Pi
- raspbian lite

- a network
- Internet, for setup only. Afterwards it depends on what you want to do
- Broker PCB
- 1 RJ12 cable

If you prefer, you can simply buy a whole Raspberry Pi kit

pre-setup

You can ignore the pre-setup if you wish, make just shure you meet the requirements.

- flash your micro SD with raspbian lite. How you do this is up to you
- boot the RPI without network connection. Connect Display and mouse to it.
- login with default user pi and default password raspberry
- ENTER

```
sudo raspi-config
```

- change default password
- change timezone to your timezone
- change keyboard layout, if necessary
- enable ssh, this will be very much needed as we setup the RPI later on over SSH. If you do not wish to enable SSH, you will always need a Display an keyborad to setup or change something on the RPI
- resize memory from 64 to 16, this will improve performance of the RÜI
- resize filesystem
- create folder backupfiles with

```
sudo mkdir backupfiles
```

this folder will hold backups of configuration files. If something goes wrong, we will always have the backup files to fix it

- make copy of /etc/dhcpcd.conf

```
sudo cp /etc/dhcpcd.conf /home/backupfiles/dchcpcd.conf.org
```

- edit /etc/dhcpcd.conf , add static IP

```
sudo nano /etc/dhcpcd.conf
```

```
   # Custom static IP address for eth0.
   interface eth0
   static ip_address=000.000.000.000
   static routers=192.168.1.1
   static domain_name_servers=192.168.1.1 8.8.8.8

   # Custom static IP address for wlan0.
   interface wlan0
   static ip_address=000.000.000.000
```

```
    static routers=192.168.1.1
    static domain_name_servers=192.168.1.1 8.8.8.8
```

change these values to match your network

- make a copy of new dhcpcd.conf and add in the file name the date so you know when you changed it

```
sudo cp /etc/dhcpcd.conf /home/backupfiles/chcpcd.conf_yyyymmdd
```

RPI configuration

- connect your RPI to a network with internet access and boot it up and login over ssh

```
ssh pi@IP_of_the_RPI
```

- check the IP address if the static IP is correct use

```
ifconfig
```

- create a scripts folder, his folder will hold your own scripts

```
sudo mkdir /home/scripts
```

- create a RPI update script in the scripts folder, to simplfy the update process

```
cd /home/scripts
```

```
sudo nano rpiupdate.sh
```

add the following in the new file

```
 # This script will update your RPI to the newest version in one go
#!/bin/bash
sudo apt-get clean
sudo apt-get update
sudo apt-get autoremove
sudo apt-get -y upgrade
sudo apt-get dist-upgrade
sudo rpi-update
sudo reboot
```

- check internet connection with a ping

```
ping -c 4 www.google.com
```

if the ping succeeds, we can start, if not you need to debug it.

- update your RPI with the new script

```
sudo bash rpiupdate.sh
```

this can take a while and the RPI will reboot so you will lose the ssh connection and have to re-login after it.

- Now install python 3 and python3-pip on the RPI

```
sudo apt-get -y install python3
```

```
sudo apt-get -y install python3-pip
```

- Next will be the MQTT client

```
sudo pip3 install paho-mqtt
```

- As already mentioned, the decrypted data will be stored on a MQTT server, later on we will define this server with the IP address. So you can choose yours if you already have one running in your network. If you don't have one already running, do the following

```
sudo apt-get install mosquitto
```

- Check if the service is running

```
service mosquitto status
```

- As we need to copy a few files from our PC to our RPI, we are going to implement public key authentification instead of password authentication. This will simplify the ssh login process and of course increase already the security of our RPI. First create a ssh-key

```
ssh-keygen -t rsa -b 4096 -C "contact info" -f "purpose for the key"
```

This will create a 4096 Bit RSA ssh key.

- Connect to your RPI with SSH.
- Create the new folder ".ssh".

```
mkdir /home/pi/.ssh
```

- Create in this folder create now the file "authorized_keys"

```
cd /home/pi/.ssh
```

```
nano authorized_keys
```

- Copy your public key into the autorized_key file. For this you will open the authorized_key file and simply copy the content of your public ssh key into it. Be sure that the .ssh folder and the authorized_keys file have the right permissions. Make also sure that the privat key on your PC have the correct permissions.

  ```
  sudo chmod 700 /home/pi/.ssh
  ```

```
sudo chmod 600 /home/pi/.ssh/authorized_keys
```

On your PC.

```
sudo chmod 600 /home/youruser/.ssh/yourprivatsshkey
```

```
sudo chmod 600 /home/youruser/.ssh/yourpublicsshkey
```

- Next will be to configure the ssh config.

```
sudo nano /etc/ssh/sshd_config
```

Activate the publickey authentication and deactivate the password authentication.

- Now we will reboot the RPi.

```
sudo reboot
```

If you cannot access over ssh your RPI, simply plugout the SD-card and debug the situation on your labtop.

- Copy now the smartyreader.zip into the /home/pi directory

```
scp /home/youruser/whereeveryourfileis/smartyreader.zip
pi@RPIIPaddress:/home/pi/
```

- Unzip the zip and delete the zip

```
unzip smartyreader.zip
```

```
rm -r smartyreader.zip
```

source

A lot of information about smarty and the P1 port can be found in official documents released by DSOs or simply asking the DSOs and Luxmetering.

- P1 port specifications of the DSO Electris
- DSMR specification for version 5.0.2
- weigu.lu

This project is not an original idea of mine. The original idea came from weigu.lu, He build a PCB with smd parts and since i simply cannot solder smd, i redesigned the PCB for normal parts. Also i implemented a few other feature which are not included in his project, example email encryption and XMPP bot. So feel free to take a look on his website, he some nice ideas. Just like him, we distribute this project under the GNU General Public License.

disclaimer

will be reworked

GNU General Public License

No warranties