Arduino Solarmeter

Software manual

Use this manual to configure and install the software in your Arduino

3/28/2016 Revision 11.42

Table of Contents

Introduction

The purpose of the ***Arduino Solar Meter*** software is to collect data from different sensors and send the data via internet to different sites. These sites store the data and can show different graphs representing the actual or history data.

The Arduino Solar Meter software runs on the Arduino. This software has to be downloaded to the Arduino via a host system. This can be any computer that can run the Arduino IDE (Integrated development Environment) program. After downloading, the Arduino is a stand-alone system.

The program will always be uploading data to the PvOutput.org site. This site is specially designed to monitor solar panel performance.

It is also possible to upload the Gas meter readings to mindergas.nl. This is a dutch site that can be used to store and compare gas usage.

Finally, all data can be uploaded to Exosite.com. This generic site can show actual and history data of all sensors in a number of formats.

Installation

Arduino IDE	http://arduino.cc/en/Main/Software
Arduino Solar Meter	http://solarmeter.codeplex.com/

Arduino Installation (for Windows)

Download the latest Arduino installation file and execute it. This will install the development environment in the "Program files(x86)" folder.

Solarmeter installation

Unzip the file *Solarmeter_V11_42.zip* file to a folder where you want to keep all Arduino projects. In this example, the projects are stored in D:\Arduino\Projects.

You now have a folder structure like this:

```
D:\Arduino\Projects\Solarmeter
      AnalogSensor.cpp
      AnalogSensor.h
      BaseSensor.cpp
      BaseSensor.h
      Exosite.ino
      Exosite_org.h
      FerrarisSensor.cpp
      FerrarisSensor.h
      FlashMini.h
      Logging.ino
      Mail.ino
      Mindergas.ino
      P1Power.cpp
      P1Power.h
      P1GasSensor.cpp
      P1GasSensor.h
      PVoutput.ino
      S0Sensor.cpp
      S0Sensor.h
      Solarmeter.ino
      Temperature.cpp
      Temperature.h
      Time.ino
      userdefs_org.h
      Watchdog.ino
      Webstuff.ino
```

The main program is called ***Solarmeter.ino***. All other files are classes and functions used by the main program. Make sure you have all the capitals right. Arduino is case sensitive!

The files ***userdefs_org.h*** and ***exosite_org.h*** are templates for the system configuration.

They contain a sample of all configuration options. The real configuration must be stored in the files ***userdefs.h*** and ***exosite.h***.

If you do not have these files from a previous configuration, you must copy them from the template files.

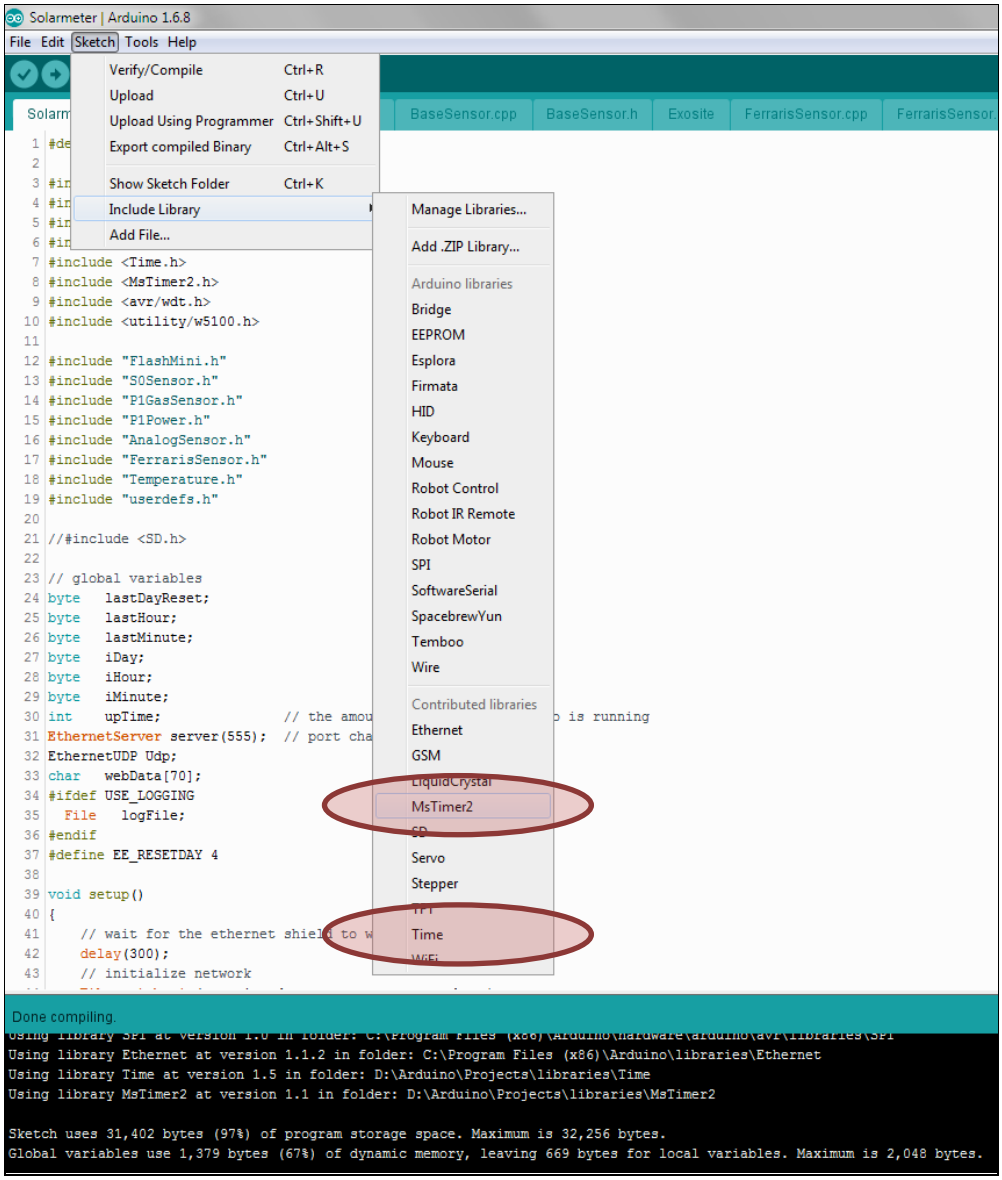
The rest of the files contain the source code of the project and do not need to be altered.

Library installation

To install libraries, Arduino now uses a library manager. Solarmeter requires 2 libraries:

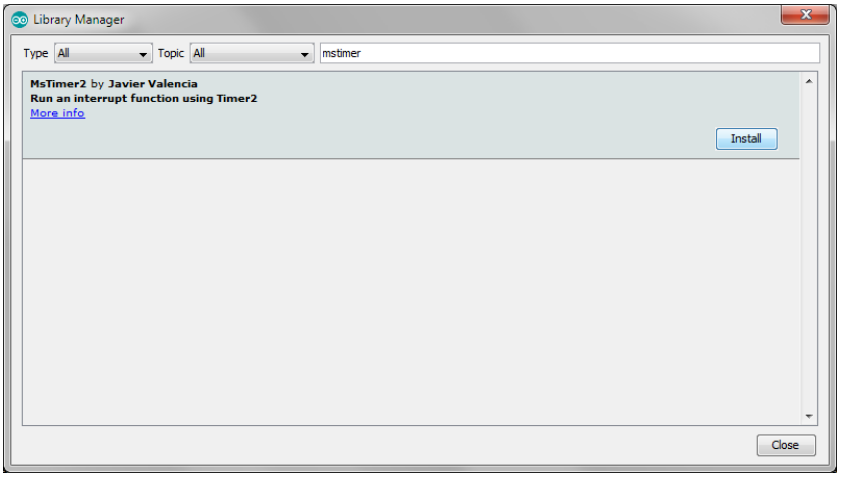
- Mstimer2
- Time

You can check if these libraries are already installed by starting the Arduino program and select Sketch > Include Library and check if the required libraries are in the list.



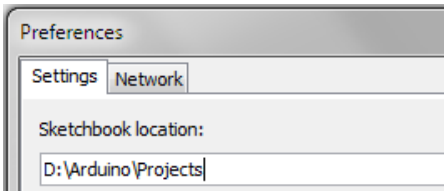
If the libraries are not present then you can add them via the menu Sketch > Include libraries > Manage Libraries.

In the search box, type the name of the library, click on the library in the list and click ‘install’:

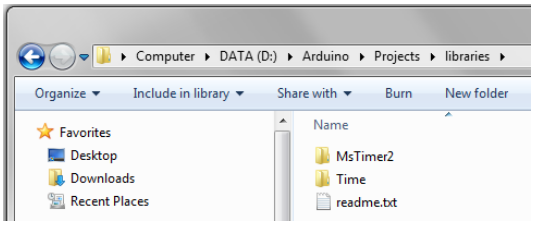


Repeat this for the other library.

Note: Arduino uses two locations to store libraries. Normally, the system libraries are stored in the installation folder (c:\Program files(x86)\Arduino\libraries). User installed libraries will be copied to the Sketchbook location. This location is set in the preferences menu:



In this case the libraries are stored here:



If you have installed the libraries manually, they must be in one of the two locations.

PvOutput

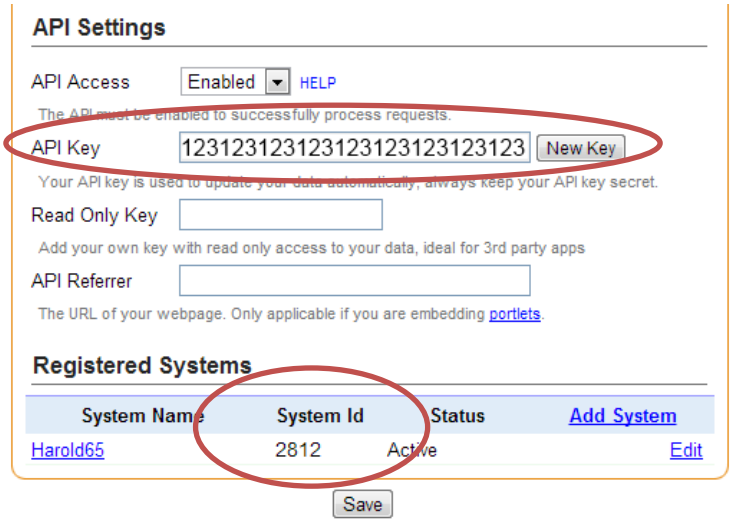
The main goal of the PvOutput site is to get an overview of the electrical energy usage of an object, normally a house. Energy is consumed (by equipment) and energy is generated (by solar panels).

PvOutput can store these energy flows per 5 or 10 minutes. It uses nice graphics to show live or history data per day, week, month or year. Also the data can be compared to all other systems on the site.

PvOutput can also calculate energy cost and savings, based on different tariffs that can be set on the site.

Before you can send data to PvOutput you have to create an account and add one or more systems.

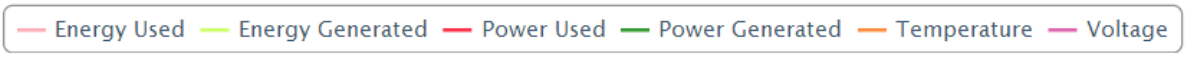
Once you have done this, go to the settings page and scroll to the bottom. You will see something like this:



The API key is needed once and is used to 'login' to your account. Do not give this number away or others can use your account. Note that the number does not fit in the textbox. Use 'Select all' if you want to copy the key. Also take care not to accidentally press the 'new key' button. A running system will not be able to upload data anymore.

The System ID (or SID) is created for every system you have added. This is also used to show the graphs and is public to everyone. To view the system above, for instance, you go to: http://pvoutput.org/intraday.jsp?sid=2812

Every PvOutput SID has a number of variables to log to. These show up on the live page in different colors.



Parameter:	Field	Graph color	Solarmeter:
v1	Energy Generation	Green area	
v2	Power Generation	Green line	2
v3	Energy Consumption	Red area	
v4	Power Consumption	Red line	4 (or 24, see below)
v5	Temperature	Orange	5
v6	Voltage	Purple	6
v7	Extended value 1	Custom	
v8	Extended value 2	Custom	8
v9	Extended value 2	Custom	
v10	Extended value 2	Custom	10

v11	Extended value 2	Custom	
v12	Extended value 2	Custom	12

V1 and v2 belong together and show Todays generated energy and the actual power generation.

V3 and v4 also belong together for the consumed energy and power.

V5 and v6 do not have a total but only display actual values on the live pages.

V7 to v12 are only available if you are a donor of PvOutput.

To make the configuration not too complicated, the solarmeter sensors must be configured for one of the variables in the last column. The corresponding energy values will automatically be uploaded to the correct variable.

An exception to this is the P1 power sensor and the Ferraris sensor. Because these sensors measure the net energy that flows in or out the house, they cannot be used as 'consumption' sensor directly.
The total energy consumption is the value measured by the P1 or Ferraris sensor PLUS the power generated by the solar panels.
These sensors have to get variable number '24'. This triggers the software to do the calculation automatically.

Example:
If you connect an S0 meter to '2', the actual power measured will be logged to 'v2' and shown as a green line and the corresponding energy value for that day will be logged to variable 'v1' and show as a green area.

- If you connect an Analog sensor to '6', only the actual value will be logged to v6
- If you put multiple sensors on the same variable, their values will be added.
- If you do not want values to be added, you have to put them on a separate graph by using a different SID.
- V1, v2, v3 or v4 has to be present on every SID you upload to.

The exact configuration of your sensor is explained in the 'configuration' chapter.

Configuration

In the ***userdefs.h*** file there are a number of options you have to select before you can build the software.

Note: The software will not check if the options set in this file are correct. There simply is no room in the Arduino to check all correct combinations. If the program does not work or is showing strange data, most likely there is a mistake in this file. So please check this carefully.

File logging settings	Meaning
#define USE_LOGGING	Logging to SD card is enabled Note: due to a bug in the Arduino precompiler, you also have to change a line in Sorarmeter.ino: remove the slashes before //#include <SD.h>.
//#define USE_LOGGING	Logging to SD card is disabled
⚠ **If you do not plan to use the SD card, please remove it from the Ethernet shield. Users reported bad behavior when the card is present but not used.**	

PvOutput settings	Meaning
#define PVOUTPUT_API_KEY "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"	Replace the xxx with your own API key
#define UPDATEINTERVAL 5	The update interval must match what you have set in the PvOutput settings: PvOutput->Settings->System->Live settings->Status interval

Sensor settings	Meaning
#define NUMSENSORS 3	Set this value to the total number of sensors you want to use. If you have 1 S0 meter and 1 Gas sensor, set NUMSENSORS to 2
S0Sensor S1(p,n,sid,v,f);	For every S0 sensor you have, add this line and replace the parameters as follows: **S1** = a unique name for the sensor **p** = the digital pin where the sensor is connected **n** = the number of pulses per kWh that this sensor generates **sid** = the pvoutput SID where you want the data to be logged **v** = the variable number of the SID. See PvOutput section for an explanation **f** = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.
⚠ **Note for Arduino MEGA users: Do not use pin 4 for a S0 sensor. This pin will not work correctly.**	
AnalogSensor S2(a,n,sid,v,f);	For every Analog sensor you have (gas, water), add this line and replace the parameters as follows:

	S2 = a unique name for the sensor **a** = the analog pin where the sensor is connected **n** = the number of pulses per m^3 that this sensor generates **sid** = the pvoutput SID where you want the data to be logged **v** = the variable number of the SID. See PvOutput for an explanation. **f** = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.
FerrarisSensor S3(a1,a2,n,sid,f);	If you want to monitor your ferrarismeter, add this line and replace the parameters as follows: **S3** = a unique name for the sensor **a1** = the analog pin where the left sensor is connected **a2** = the analog pin where the right sensor is connected **n** = the number of pulses per kWh that this sensor generates. This number is printed on the meter as "C=275". Fill in the number behind the "C=" **sid** = the pvoutput SID where you want the data to be logged The FerrarisSensor always logs to v3 and v4 **f** = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.
Temperature T1("station",sid,f)	By using this sensor, the actual temperature will be retrieved from XML.BUIENRADAR.NL. The station number is a string with the number of a nearby weatherstation. Go to http://gratisweerdata.buienradar.nl/#Station to find this number or go directly to xml.buienradar.nl and lookup a nearby station and its number. **sid** = the pvoutput SID where you want the temperature to be logged. **f** = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput. The Temperature sensor always logs to v5. Set the scaling factor (f) to 10 if you want to log in correct units.
P1Power P1(&Serial,sid,v,f);	This sensor will be reading values from the smart meter P1 port. The meter will send the actual counter values for production and consumption, actual consumed and generated power and if your gasmeter is also 'smart', the P1 port will also contain the actual gas counter value. **&Serial** defines the serial port that is connected to the P1 port. If you have an Arduino Uno or compatible, there is only one serial port "&Serial". If you have an Arduino MEGA, you can select one of 4 serial ports named "&Serial", "&serial1", "&Serial2" and "&Serial3". **sid** = the pvoutput SID where you want the temperature to be logged. **v** = the variable number of the SID. See PvOutput for an explanation. **f** = the scaling factor. The value calculated by the sensor will be

	divided by the factor before uploading to PvOutput.
P1GasSensor P1Gas(&P1,sid,v,f);	This sensor can only be used if you also have the P1Power sensor defined. **&P1** = The name of the P1Power sensor **sid** = the pvoutput SID where you want the gas usage to be logged. **v** = the variable number of the SID. See PvOutput for an explanation. **f** = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.
BaseSensor* sensors[] = {&S1,&S2,&S3};	This line defines the scanning order of the sensors in the software. Each defined sensor must be in this list or it will not be used by the software. Also the sequence must be so that different SID's appear in ascending order. The number of items in the list must match NUMSENSORS. Also do not change the formatting of this line.

Temperature sensor settings	Meaning
//#define USE_GRAADDAGEN	If temperature sensor T1 is defined, it will log the actual temperature of the selected weatherstation.
#define USE_GRAADDAGEN	If T1 and G1 (Analog Sensor) are defined, todays gas usage and average temperature are used to calculate a factor. This factor is calculated as: F = (24/hour) * (Gas / (18-Tavg) The factor gives you an indication of the isolation factor of your house and the way you use the heating.. See also http://www.mindergas.nl/degree_days/explanation for an explanation (dutch only)

Mail settings	Meaning
#define USE_MAIL	A mail will be sent once a day
//#define USE_MAIL	Mail function is disabled
#define MAIL_TIME 21	Defines the time of the mail in hours. In this example the mail will be sent at 21:00
#define MAIL_TO "someone@somemail.com"	The destination address
#define MAIL_FROM "arduino@meterkast.nl"	Any valid mail address will do here
#define MAIL_SERVER "smtp.mymailprovider.nl"	Use the address of your own mail provider. Note that google cannot be used as mail provider

since this would require authentication.

Network settings	Meaning
static byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };	The MAC address of the Arduino. Any address will do as long as it is unique in your network.
static byte ip[] = { 192, 168, 1, 99 };	The IP address of the Arduino. This must match the gateway and subnet settings of your router
static byte dnsserver[] = { 8, 8, 8, 8 };	Preferred value is the IP address of your router. If this does not work, fill in the address of your local DNS server. Finally you can use Google's DNS server at 8,8,8,8.
static byte gateway[] = { 192, 168, 1, 1 };	This is the local address of your router. If you navigate to this address and get the router configuration page, the address is correct.
static byte subnet[] = { 255, 255, 255, 0 };	Normally you do not need to change this value.

Miscellaneous settings	Meaning
#define USE_WD	If this line is present then the watchdog function is enabled. The watchdog will make sure that the Arduino is reset when the software is not responding anymore. Note: Due to a bug in the Arduino MEGA bootloader, the watchdog will not work correctly. Newer MEGA's already have an updated bootloader but there is no way to check the current bootloader version. If you enable the watchdog in a MEGA but it still hangs, problably the bootloader must be updated. Arduino UNO does not have this problem.
#define TIME_OFFSET 0	This number can be used to shift the internal clock in the Arduino by a number of seconds. This is useful if you do not want the Arduino to upload data to PvOutput exactly on the hour, because the rest of the world is also doing that and PvOutput sometimes cannot handle this. By shifting the Arduino time, the upload moment is also shifting, allowing you to upload just before or after the bulk.

Mindergas.nl

www.mindergas.nl is a dutch site where users can store their daily gas-usage. Normally this is done manually but the site makers have created the possibility to upload data automatically.

The Arduino solarmeter software can do this for you. Follow the next steps to get things running:

1. Send an email to mailto:info@mindergas.nl and request a **http token for the Arduino software**.
2. You will receive a 20 digit token.
3. Configure the userdefs.h file:

MinderGas settings	Meaning
//#define USE_MINDERGAS	Logging to mindergas is disabled
#define USE_MINDERGAS	Logging to mindergas is enabled
#define MG_KEY "xxxxxxxxxxxxxxxxxxxx"	Replace the xxx with the token you received via mail
#define MG_SENSOR G1	Analog Sensor G1 is used to provide the gas meter value (correct meter value has to be setup via status page)
#define MG_SENSOR P1Gas	Sensor P1Gas is used to provide the gas meter value (correct meter value is already present, no further action needed)

The meter value will be uploaded to mindergas at every midnight. Check the site or the Arduino status page to check if all went well.

Exosite.com

Exosite is also a site that can log data for you. Unlike PvOutput it is not restricted to energy usage or consumption. If you would like to log your weightloss, it is also possible at Exosite ;-)

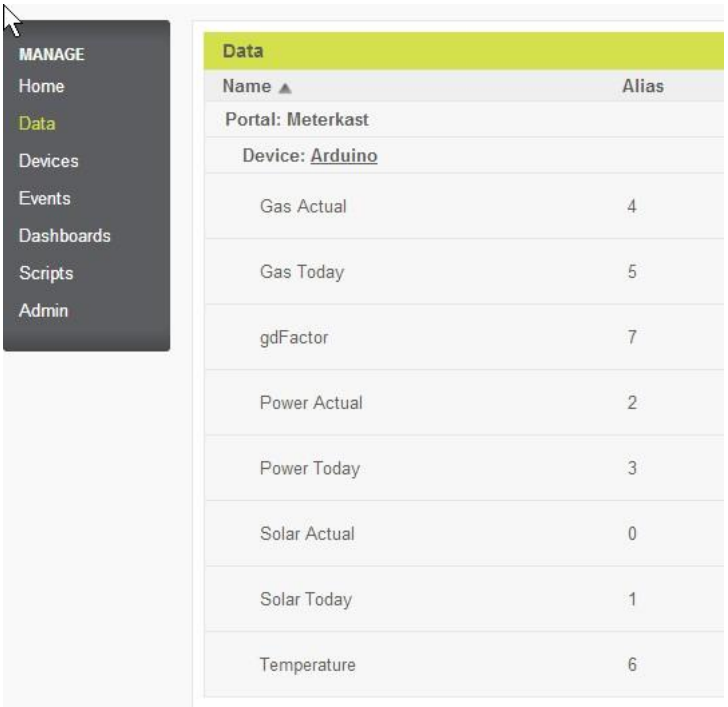
Setup

You have to follow the next steps to let the Arduino upload data to exosite:

1. Create an account.
   Follow the instructions on the site
2. Create your device.
   Follow the instructions on the site
3. The device will have a CIK. This is the key needed for uploading. Copy this number to the userdefs.h file:

Exosite settings	Meaning
#define EXOSITE_KEY "xxxxxxxxxxxxxxxxxxxx"	Replace the xxx with the CID key of your device
//#define EXOSITE_KEY "xxxxxxxxxxxxxxxxxxxx"	Exosite uploading is disabled
#define EXOSITEUPDATEINTERVAL 5	The values are sent to the server every 5 minutes

4. Create the data streams. There can be a maximum of 10 streams in the free edition.
   The name, unit and other data can be freely defined.
   The important parameter of the data streams is the “ALIAS” value. The Arduino software will use this alias to reference a data stream:



To map the Sensor data to the Exosite data streams, the exosite.h file has to be changed. The program defines 10 variables v[0] to v[9].
These variables are filled with sensor data and sent to the Exosite datastream that has the same alias as the variable index:

For example:

v[0]=S1.Actual	Will be sent to “Solar actual”
v[1]=S1.Today	Will be sent to “Solar today”
v[6]=T1.Actual	Will be sent to “Temperature”
V[9]=S1.Midnight + S1.Today	The actual total counter of S1 will be sent

Once filled correctly the Arduino will upload new data every minute.

Exosite will store all the data, but older data will be compressed to a lower resolution.

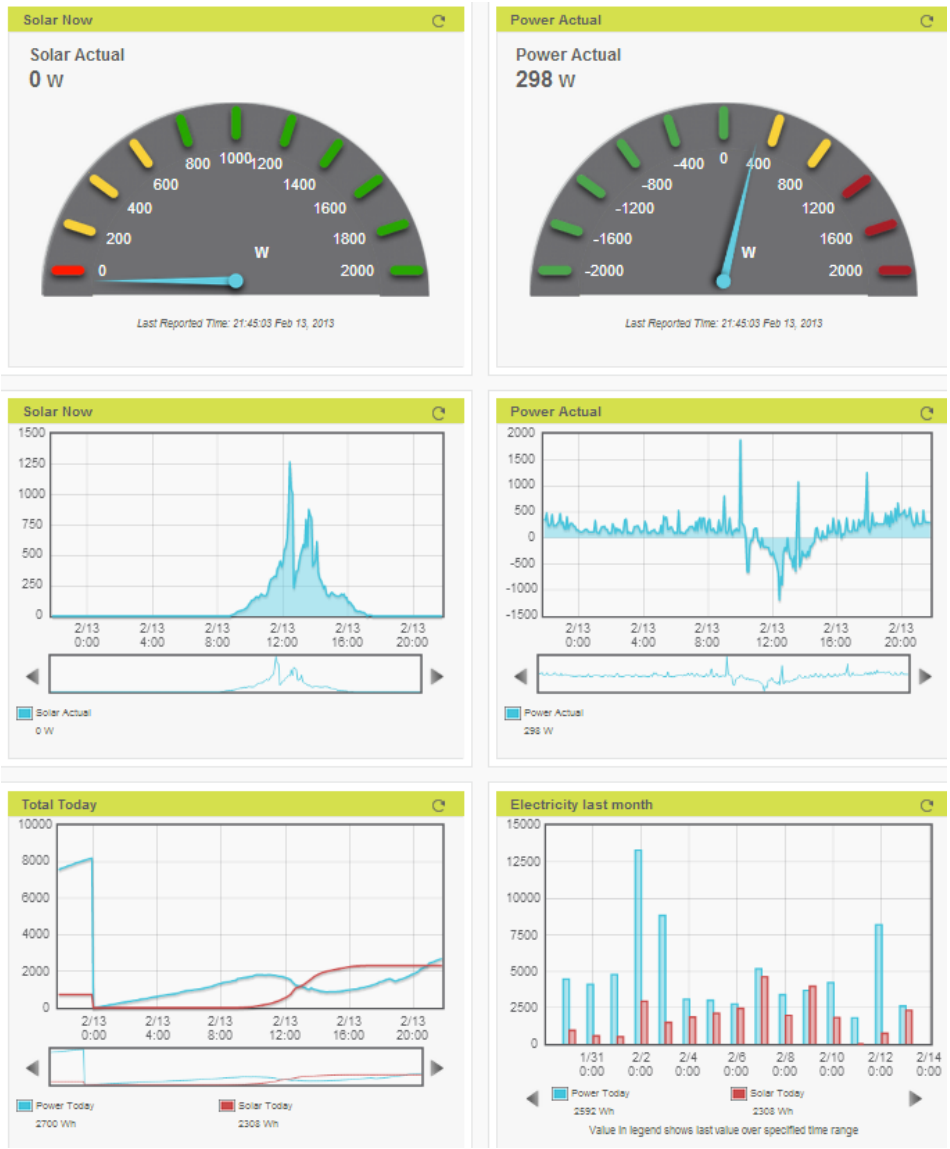
Dashboard

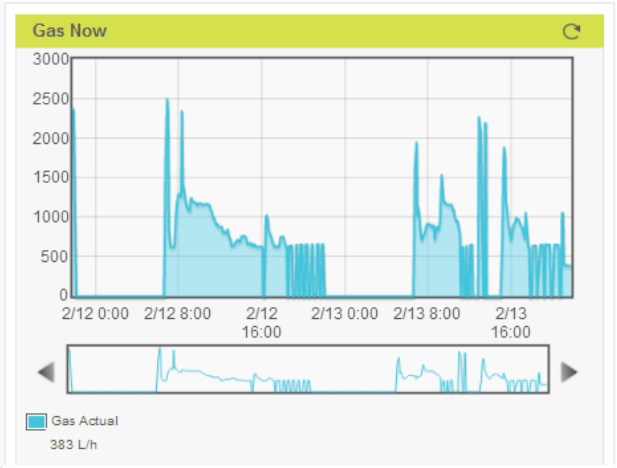
Now that the link between Arduino sensors and exosite datastreams is made, you can concentrate on the dashboards

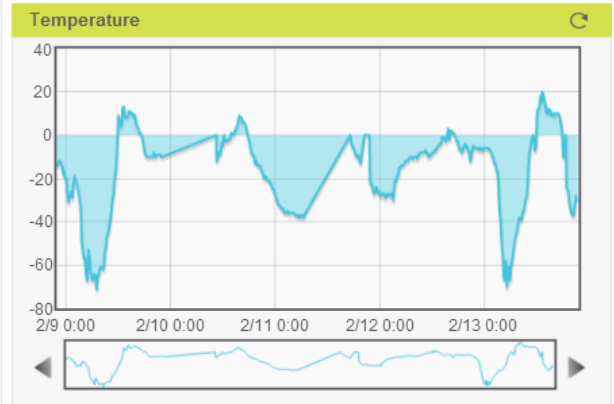
Experiment with the different types. You can also use different views of the same data. It is up to your imagination.

More documentation is also available.

Sample dashboards:



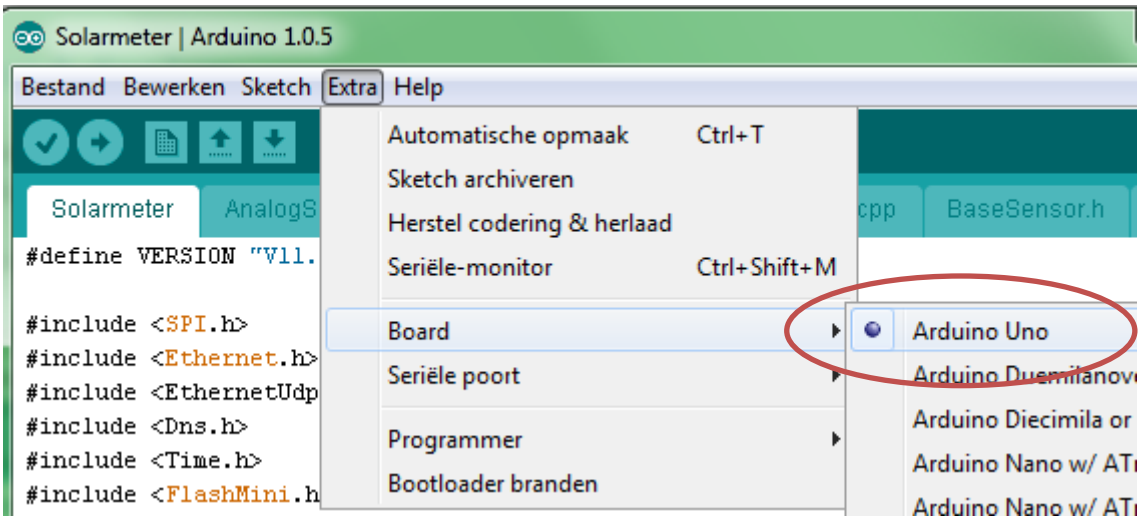

Gas Now
3000
2500
2000
1500
1000
500
0
2/12 0:00
2/12 8:00
2/12 16:00
2/13 0:00
2/13 8:00
2/13 16:00
Gas Actual
383 L/h


Temperature
40
20
0
-20
-40
-60
-80
2/9 0:00
2/10 0:00
2/11 0:00
2/12 0:00
2/13 0:00

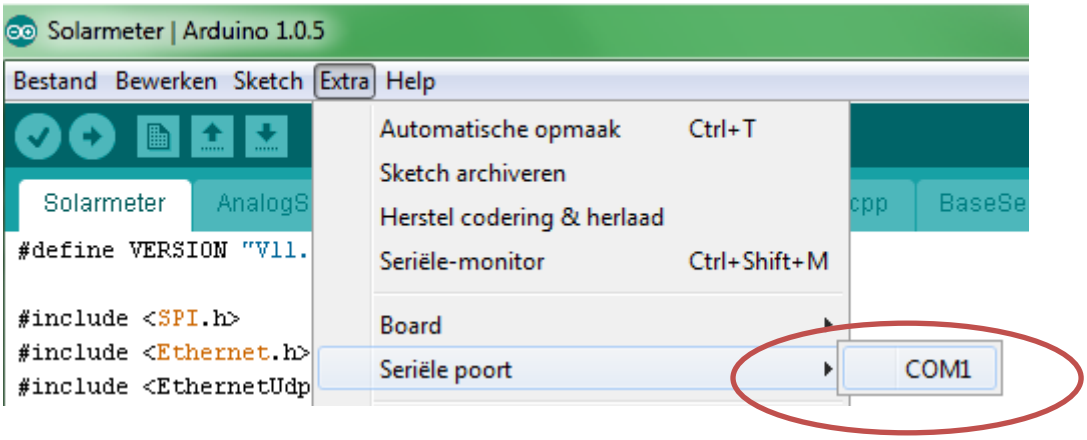
Build and deploy

Once you have configured the system for your needs, all you have to do is build the software and upload it to the Arduino.

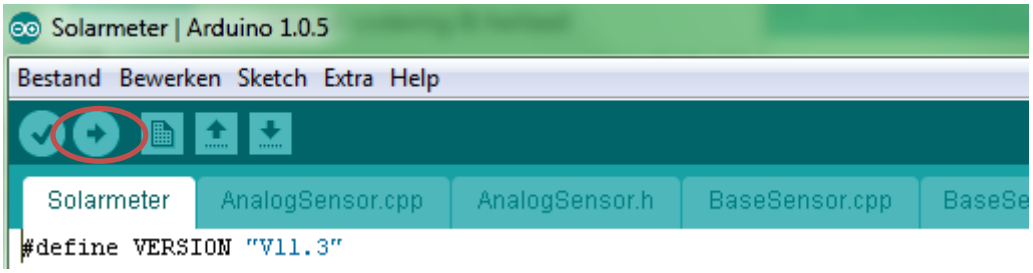
Select the correct Arduino board:



Then select the correct serial port:



Finally press the upload button:



When uploading is finished, you can disconnect the board, put it in your meter cabinet and power up.

Testing

Of course you can test your system by looking into the PvOutput page and see if any data is recorded.

The second method is to log in to Arduino's own web-page. The program will generate a text page containing all values of all sensors:

http://192.168.1.99:555 (or the ip-address you set in the userdefs file)

The ':555' behind the address will tell your web browser to use port 555 instead of the default port 80. This was done because many malware is using port 80 to see if an ip-address is 'alive'.

A sample of this page will look like this:

V11.41
03.07.14 22:40:53
Uptime=4d+0h

ID	SID	Type	Actual	Peak	Today	Total	Factor	TodayCnt	EEprom	ppu	Pulse	Extra
0	2812	2	0	0	10310	4503801	1	10310	10310	1000	398555	
1	2812	24	192	192	-4468	123818588	1	-1117	-1157	250	74940	C=3775 1=347-396:474 2=195-229:296
2	2812	8	0	0	580	12021950	1	58	58	100	215699	

last PvOutput fail=Response timeout @ 03.07.14 10:00:01
DNS status=1
Last NTP update=03.07.14 22:23:47 (in 1x)
mgUpload=03.07.14 00:22:00
MgResponse=201
ExResponse=204
WD ctr=49
WD val=5
Reset Day=3
Free=394

This page will show you:

- The actual version number
- The actual time (synchronized twice a day)
- The uptime in days and hours since the last boot
- A table with for each sensor the SID and the Actual, Today and Peak value for this sensor. If a sensor has extra information, it will be displayed in the 'Extra' column
- The code and time of the last failed upload to PvOutput. 200 is ok, 400-499 is a configuration error. A code of 0 does not mean an error but that PvOutput took too long to respond. The data however is saved.
- The status of the last call to the DNS server. 1=ok, -1 is timeout
- The time of the last synchronization with the time server.
- The time of the upload to Mindergas.nl (if enabled)
- The status of the Mindergas upload (201 is a successful upload, 422 is error)
- The status of the Exosite upload ('204' is not an error but a remark that Exosite recorded the data but returned no data.
- The status of the watchdog.
    - WD ctr counts the number of resets by the watchdog
    - The counter is incremented at every reset but never set to a default value. Therefore you can only use it to see how many resets happened since the last time you checked.

- WD val shows the last checkpoint in the code when the reset occurred.
  1 = Process new day event, collect Mindergas data
  2 = Process new hour event, save sensor data and send mail
  3 = Process new minute event, calculate sensor data
  31 = Send to Mindergas.nl
  32 = Send to PvOutput.org
  33 = Reset peak data
  34 = Send to Exosite.com
  4 = read P1 port
  5 = refresh webpage
- Reset day will show the day of the month that the meter values were reset. This is used to prevent multiple counter resets if your Arduino reboots more than once in a day.
- Free is the number of free ram in the cpu. This number should remain constant after a while. If not, there is a memory leak in the software…

The values in the table shown at each sensor mean the following:

ID	The number of the sensor. Used to change the
SID	The System ID where the data is logged to.
Type	The variable where the data is logged to. The Ferraris sensor and the P1 sensor are type 24
Actual	The actual production/consumption in real units (Wh/h=W or L/h or ^{O}C) but before scaling. The temperature shown in row 3 in the example is shown as -31 but send to PvOutput as -3.1
Peak	The maximum of Actual during the last 5 minutes (or 10 if you changed the interval)
Today	The total for today in real units (Wh or litre)
Total	The absolute meter value in real units (Wh or litre)
Factor	The scaling factor
TodayCnt	The total number of pulses counted today
Eeprom	The last saved value of TodayCnt. This number will be loaded when the Arduino is powered on.
PPU	The conversion factor from pulses to real units, pulses per unit
Pulse	The time between the last two pulses in ms.

Extra info that is sent by the Ferraris sensor:

C=nnnn	The counter for the threshold adjustment. When nnnn reaches 10000 the thresholds will be adjusted
1=aaa-bbb:ccc	The current data for sensor 1. aaa is the lower threshold, bbb the higher threshold and ccc is the last read analog value
2=aaa-bbb:ccc	The current data for sensor 2. aaa is the lower threshold, bbb the higher threshold and

	ccc is the last read analog value

Extra info sent by the Temperature sensor:

average	The average temperature for today
gdFactor	The gas usage factor

Extra info sent by the P1 sensor:

PowerUsage	The actual power used
PowerSolar	The actual power generated
GasUsage	The current meter value of the gas meter
M1	The current meter value of energy used, high tariff
M2	The current meter value of energy used, low tariff
M3	The current meter value of energy generated, high tariff
M4	The current meter value of energy generated, low tariff

Updating day values

Normally, the measured day values of each sensor are stored in EEprom every hour. This is done so the values will not be lost when the Arduino is reset.

Values are only written once per hour because the EEprom has a limited number of write cycles.

Save actual values

If you want to force the saving of values because you want to update the software of have to reset the Arduino for another reason, you can do that by specifying a parameter on the webpage:

http://192.168.1.99:555/save

Change an individual value

If one value is incorrect for some reason, you can change this by entering:

http://192.168.1.99:555?1=1234

This will set the *TodayCnt* value of the selected sensor to the new value.

The number behind the "?" is listed in the "ID" column.

Note that the counter values are saved, not the Actual values

Change the absolute value of a meter

The absolute value for each sensor has to be entered manually (except for smart meters, they already send absolute values)

To set the value, enter the following:

http://192.168.1.99:555?A=1234567

The character behind the "?" is 'A' for sensor 0, 'B' for sensor 1, etc.

Reset all values

If, for some reason, the daily values are not correct and you want to simulate the start of a new day, use the reset function:

http://192.168.1.99:555/reset

This will reset all counters to 0 (or to the actual meter reading in case of a P1 meter)

If you did the reset accidentally, just press the reset button on the Arduino and the last saved values will be restored.

To make the change permanent, execute the */save* command or wait until the next hour (where the values will be saved anyway)

Restart the Arduino

If, for some reason, you want to restart the Arduino, you can do so by entering the next command:

http://192.168.1.99:555/restart

Update the time

Since 11.41 it is also possible to force an update of the realtime clock.

http://192.168.1.99:555/ntp

Test uploading of gas value

By entering the command http://192.168.1.99:555/gas the current value of your gasmeter (if enabled) will be uploaded to mindergas.nl immediately.

If you have any questions or problems, please contact me via the issues page on http://solarmeter.codeplex.com

If you know dutch, it is also possible to check out the forum on www.wijhebbenzon.nl. Many threads are related to this software.

Enjoy!