



Arduino Solarmeter

Software manual

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

Table of Contents

Introduction.....	1
PvOutput	1
Downloads.....	3
Installation.....	3
Configuration.....	4
Testing	9

Introduction

The **Arduino Solar Meter** software runs on the Arduino. It has to be downloaded via a host system. This can be any computer that can run the Arduino IDE (Integrated development Environment) program.

The program will be uploading data to the PvOutput site. To do this, an account has to be created. See the next chapter on how to do this.

PvOutput

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:

API Settings

API Access [HELP](#)

The API must be enabled to successfully process requests.

API Key

Your API key is used to update your data automatically. always keep your API key secret.

Read Only Key

Add your own key with read only access to your data, ideal for 3rd party apps

API Referrer

The URL of your webpage. Only applicable if you are embedding [portlets](#).

Registered Systems

System Name	System Id	Status	Add System
Harold65	2812	Active	Edit

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.

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.

— Energy Used — Energy Generated — Power Used — Power Generated — Temperature — Voltage

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
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 Today's 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 connected to one of the variables in the last column. The corresponding energy values will automatically be added to the variable one row above.

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

Downloads

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

Installation

Unzip the file *Arduino-1.0.3-window.zip* or the version that you need and place the files in some user folder.

Unzip the file *Solarmeter_V10.zip* and copy the two folders to the same user folder where you placed the Arduino software.

You now have a folder structure like this:

```
\Arduino-1.0.3
  \Libraries
    <all standard Arduino libraries>
    FlashMini
    MSTimer2
    Time
  \Projects
    \Solarmeter
      AnalogSensor.cpp
      AnalogSensor.h
      BaseSensor.cpp
      BaseSensor.h
      FerrarisSensor.cpp
      FerrarisSensor.h
      Logging.ino
      Mail.ino
      P1Power.cpp
      P1Power.h
      P1GasSensor.cpp
      P1GasSensor.h
      PVoutput.ino
      S0Sensor.cpp
      S0Sensor.h
      Solarmeter.ino
      Temperature.cpp
      Temperature.h
      Time.ino
      userdefs.h
      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 file *userdefs.h* is used to configure the system. Normally you only need to edit this file and do not need to change the others..


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 <code>//#include <SD.h></code> .
//#define USE_LOGGING	Logging to SD card is disabled

PvOutput settings	Meaning
#define PVOUTPUT_API_KEY "xx"	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);	<p>For every S0 sensor you have, add this line and replace the parameters as follows:</p> <p>S1 = a unique name for the sensor</p> <p>p = the digital pin where the sensor is connected</p> <p>n = the number of pulses per kWh that this sensor generates</p> <p>sid = the pvoutput SID where you want the data to be logged</p> <p>v = the variable number of the SID. See PvOutput section for an explanation</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p>
 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);	<p>For every Analog sensor you have (gas, water), add this line and replace the parameters as follows:</p> <p>S2 = a unique name for the sensor</p> <p>a = the analog pin where the sensor is connected</p>

	<p>n = the number of pulses per m³ that this sensor generates</p> <p>sid = the pvoutput SID where you want the data to be logged</p> <p>v = the variable number of the SID. See PvOutput for an explanation.</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p>
FerrarisSensor S3(a1,a2,n,sid,f);	<p>For every Analog sensor you have (gas), add this line and replace the parameters as follows:</p> <p>S3 = a unique name for the sensor</p> <p>a1 = the analog pin where the left sensor is connected</p> <p>a2 = the analog pin where the right sensor is connected</p> <p>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="</p> <p>sid = the pvoutput SID where you want the data to be logged</p> <p>The FerrarisSensor always logs to v3 and v4</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p>
Temperature T1("station",sid,f)	<p>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://gratisweerdeata.buienradar.nl/#Station to find this number or go directly to xml.buienradar.nl and lookup a nearby station and its number.</p> <p>sid = the pvoutput SID where you want the temperature to be logged.</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p> <p>The Temperature sensor always logs to v5.</p> <p>Set the scaling factor (f) to 10 if you want to log in correct units.</p>
P1Power P1(&Serial,sid,v,f);	<p>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.</p> <p>&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".</p> <p>If you have an Arduino MEGA, you can select one of 4 serial ports named "&serial0" to "&Serial3"</p> <p>sid = the pvoutput SID where you want the temperature to be logged.</p> <p>v = the variable number of the SID. See PvOutput for an explanation.</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p>

P1GasSensor P1Gas(&P1,sid,v,f);	<p>This sensor can only be used if you also have the P1Power sensor defined.</p> <p>&P1 = The name of the P1Power sensor</p> <p>sid = the pvoutput SID where you want the temperature to be logged.</p> <p>v = the variable number of the SID. See PvOutput for an explanation.</p> <p>f = the scaling factor. The value calculated by the sensor will be divided by the factor before uploading to PvOutput.</p>
BaseSensor* sensors[] = {&S1,&S2,&S3};	<p>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.</p>

Temperature sensor settings	Meaning
//#define USE_GRAADDAGEN	If T1 is defined, it will log the actual temperature of the selected weatherstation.
#define USE_GRAADDAGEN	<p>If T1 and G1 (Analog Sensor) are defined, today's gas usage and average temperature are used to calculate a factor. This factor is calculated as: $F = (24/\text{hour}) * (\text{Gas} / (18 - T_{\text{avg}}))$</p> <p>The factor gives you an indication of the isolation factor of your house.</p> <p>See also http://www.mindergas.nl/degree_days/explanation for an explanation (dutch only)</p>

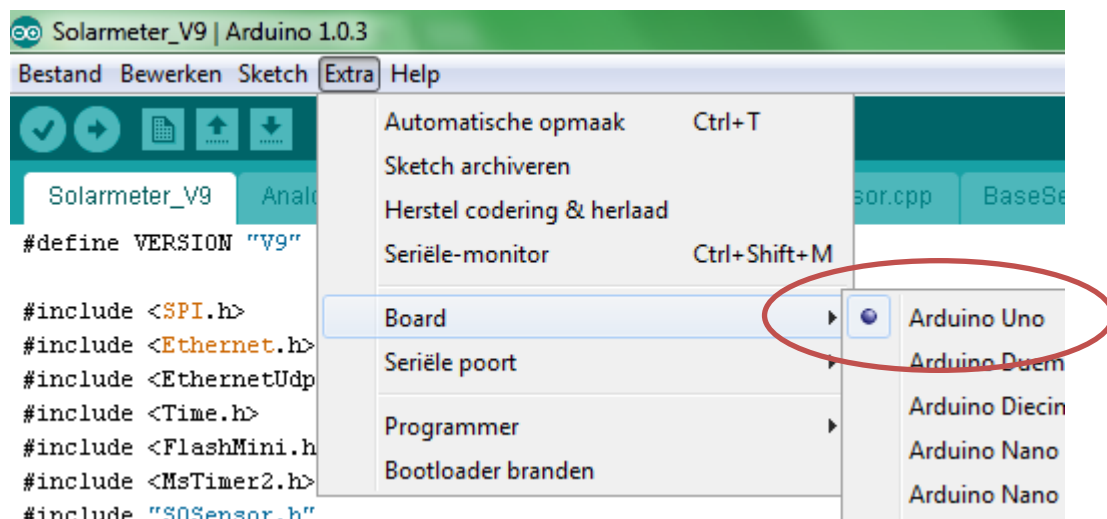
Mail settings	Meaning
#define USE_MAIL	A mail will be sent once a day
//#define USE_MAIL	Mail function is disabled
#define MAIL_TIME 21	Defines the time of the mail in hours. In this example the mail will be sent at 21:00
#define MAIL_TO "someone@somemail.com"	The destination address
#define MAIL_FROM "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.

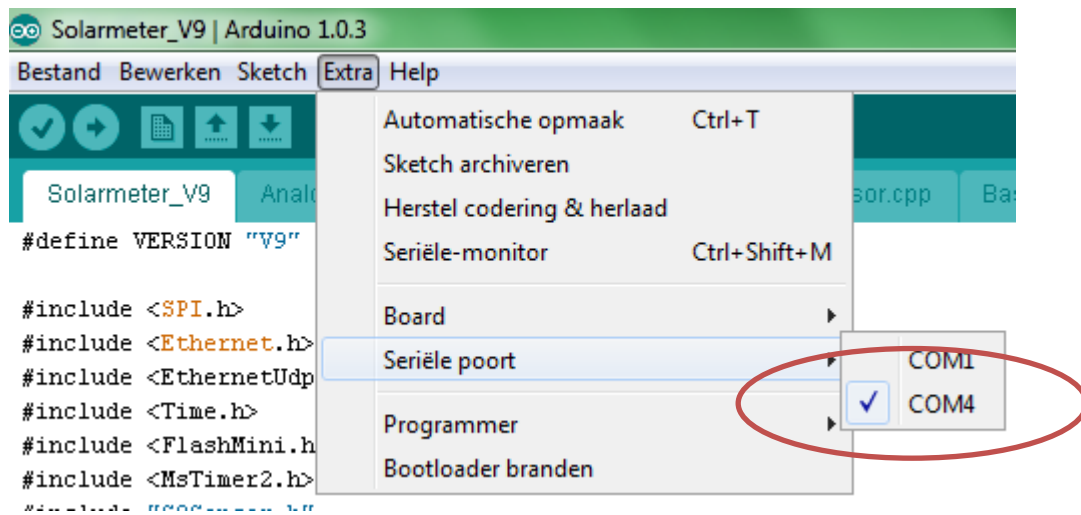
Build and deploy

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

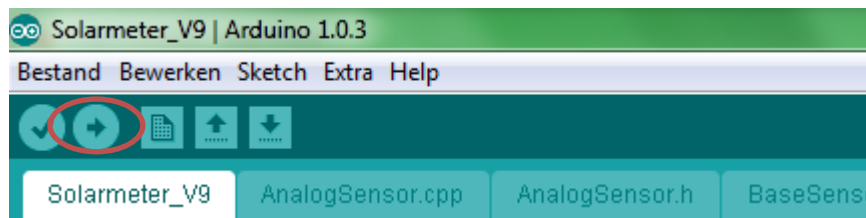
Select the correct Arduino board:



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>

A sample of this page will look like this:

← → ↺ ⬆ 192.168.1.99											
V10 26.01.13 09:55:43 Uptime=3d+17h											
ID	SID	Type	Actual	Peak	Today	Factor	TodayCnt	EEprom	ppu	Pulse	Extra
0	2812	2	399	399	211	1	211	28	1000	9010	
1	2812	24	-250	-250	1360	1	340	339	250	-57420	C=9462 1=290-351:597 2=187-229:370
2	2812	8	1137	1137	1470	1	147	28	100	31650	
3	2812	5	-31	0	0	10	60	54	1000	0	Average=-3.79 GDFactor=17.87
PvOutput response=200 DNS status=1 Last NTP update=25.01.13 22:00:00											

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 status of the last upload to PvOutput.. 200 is ok, 0 is no response, 400-499 is configuration error.
- 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 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 °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 m3)
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.

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

<http://192.168.1.99/save>

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

<http://192.168.1.99?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