

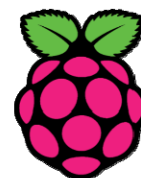
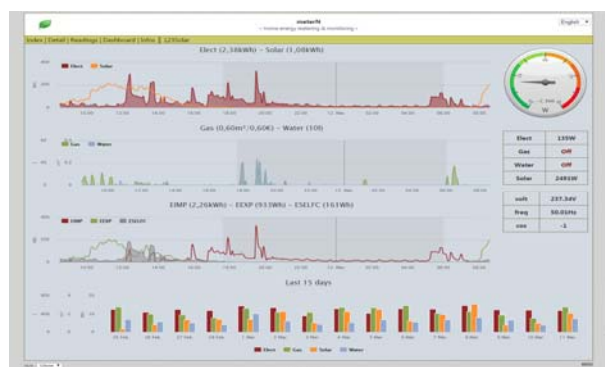
# MONITORAGGIO ENERGETICO

## MANUALE di INSTALLAZIONE

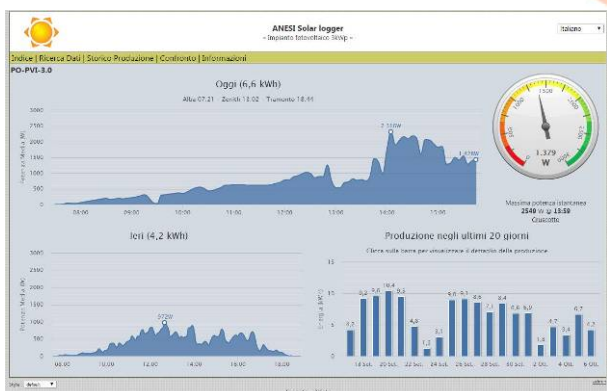
### METERN su Raspberry Pi®

con contatore RS485

### EASTRON SDM120 - SDM220 modbus



RaspberryPi



VER. 2.10

by FLAVIO ANESI

[www.flanesi.it](http://www.flanesi.it)

# INDICE

<b>1</b>	<b>PREMESSA .....</b>	<b>4</b>
1.1	CONVENZIONI TESTO .....	5
1.2	RINGRAZIAMENTI .....	5
<b>2</b>	<b>MATERIALE NECESSARIO .....</b>	<b>6</b>
2.1	HARDWARE.....	6
2.2	SOFTWARE.....	6
<b>3</b>	<b>MISURA CON CONTATORE MODBUS (RS485) .....</b>	<b>8</b>
3.1	MODBUS: PROTOCOLLO UNIVERSALE .....	8
3.2	CONTATORE EASTRON SDM120 MODBUS .....	8
3.3	CONTATORE EASTRON SDM220 MODBUS .....	8
3.4	DOVE ACQUISTARE I CONTATORI.....	9
3.5	COLLEGARE IL CONTATORE ALL'IMPIANTO ELETTRICO.....	9
3.6	LETTURA DATI VIA RS485.....	11
3.7	COLLEGARE IL CONTATORE AL RASPBERRY .....	12
3.7.1	<i>Solo monitoraggio consumi (assenza fotovoltaico).....</i>	<i>12</i>
3.7.2	<i>Inverter Aurora Power-One e contatore SDM120 modbus.....</i>	<i>12</i>
3.7.3	<i>Inverter generico e 2 contatori SDM120 modbus.....</i>	<i>13</i>
3.8	CABLAGGIO RETE MODBUS .....	14
<b>4</b>	<b>INSTALLAZIONE METERN.....</b>	<b>15</b>
4.1	AVVIO DI METERN AL BOOT .....	17
<b>5</b>	<b>INSTALLAZIONE SOFTWARE MODBUS.....</b>	<b>18</b>
5.1	INSTALLARE LA LIBRERIA LIBMODBUS .....	18
5.2	INSTALLARE SCRIPT SDM120C .....	18
<b>6</b>	<b>PARAMETRI COMUNICAZIONE CONTATORE.....</b>	<b>19</b>
6.1	PORTA USB RASPBERRY .....	19
6.2	INDIRIZZO E VELOCITÀ CONTATORE .....	20
6.3	PARITÀ.....	20
6.4	MODIFICA INDIRIZZO E VELOCITÀ CONTATORE .....	21
6.4.1	<i>Modifica indirizzo contatore.....</i>	<i>21</i>
6.4.2	<i>Modifica velocità modbus.....</i>	<i>22</i>
6.5	TEST DI LETTURA CONTATORE.....	22
<b>7</b>	<b>CONFIGURARE I METERS (MISURATORI) .....</b>	<b>23</b>
7.1	MODIFICA FILE POOLER485 PER LETTURA CONSUMI .....	23
7.2	AVVIO FILE POOLER485 PER LETTURA CONSUMI .....	23
7.3	PAGINE DI AMMINISTRAZIONE.....	24
7.4	CONFIGURAZIONE PRINCIPALE.....	26
7.5	CONFIGURAZIONE MISURATORI.....	26
7.5.1	<i>Misuratore 1 - Produzione.....</i>	<i>27</i>
7.5.2	<i>Misuratore 2 – Consumi .....</i>	<i>28</i>
7.5.3	<i>Misuratore 3 – Prelievi.....</i>	<i>29</i>
7.5.4	<i>Misuratore 4 – Immissioni.....</i>	<i>30</i>
7.5.5	<i>Misuratore 5 – Autoconsumo.....</i>	<i>31</i>
7.5.6	<i>Impostazione “Price per unit” .....</i>	<i>31</i>
7.6	CONFIGURAZIONE LAYOUT.....	32

<b>8</b>	<b>CONFIGURARE GLI INDICATORS (INDICATORI)</b>	<b>33</b>
8.1.1	Indicatore 1 - Tensione	33
8.1.2	Indicatore 2 - Corrente	33
8.1.3	Indicatore 3 – Cos $\varphi$ (fattore di potenza)	33
<b>9</b>	<b>AVVIO METERN</b>	<b>34</b>
<b>10</b>	<b>TEST METERN</b>	<b>35</b>
10.1	TEST MISURATORE 1 - PRODUZIONE	35
10.2	TEST MISURATORE 2 - CONSUMI	37
10.3	TEST INDICATORS	38
<b>11</b>	<b>LICENZA D'USO</b>	<b>39</b>
<b>APPENDICI</b>		<b>40</b>
APPENDICE A	GUIDA ALL'USO DELLO SCRIPT SDM120C	41
APPENDICE B	DISPOSITIVI USB (ASSEGNARE UN NOME FISSO)	43
	Adattatori usb diversi	43
	Adattatori usb identici	44
APPENDICE C	AGGIUNTA SENSORI VARI	45
	Impostazioni di base	45
	Piedinatura e numerazione GPIO	45
	Indicatore di processo in esecuzione per meterN	47
	Sensore di temperatura ed umidità DHT22	47
	Sensori di temperatura DS18B20	54
	Sensori di pressione/temperatura/altitudine BMP085 o BMP180	61
APPENDICE D	PROCEDURA DI AGGIORNAMENTO DI METERN	68
APPENDICE E	BACKUP DATI METERN TRAMITE FTP	70
APPENDICE F	INVIO DATI PRODUZIONE E CONSUMO SU PVOUTPUT.ORG	72
	Creazione account su pvoutput.org	72
	Configurazione 123solar	73
	Configurazione dati consumo MeterN	74
APPENDICE G	UPS PER RASPBERRY	75
APPENDICE H	PRESERVARE LA SCHEDA SD DA POSSIBILI DANNI	80
	RamLog	80
	Disabilitare il file swapping	82
	Controllare l'utilizzo della MicroSD di Raspberry PI	83
APPENDICE I	CONFIGURARE LA RETE LAN O WIFI	84

# 1 PREMESSA

La presente guida spiega come installare e configurare il software MeterN sul vostro Raspberry per utilizzarlo come monitor dei consumi domestici.

MeterN è un software free che comprende solo l'interfaccia di visualizzazione via web ed archiviazione dei dati, mentre le eventuali interfacce ed eventuali software per l'acquisizione dei dati vengono demandate ad applicazioni esterne.

Nel nostro caso per rilevare i consumi e comunicarli a MeterN utilizzeremo un contatore EASTRON SDM120C con uscita seriale RS485 (ModBus) che dovrà essere installato nel nostro quadro elettrico di casa e collegato in modo molto semplice al Raspberry per mezzo di un adattatore USB-RS485 come vedremo in seguito. Sarà anche necessario installare sul Raspberry un software che servirà a leggere i dati dal contatore e comunicarli a MeterN.

Nella seguente guida, si presume:

- a) che si utilizzi un Raspberry con Raspbian "wheezy" (vedasi la specifica GUIDA di Walter62 sulla [CONFIGURAZIONE RASPBERRY](#))
- b) che sul Raspberry si abbia già installato e configurato 123solar (vedasi la specifica GUIDA di Walter62 sull'[INSTALLAZIONE di 123SOLAR](#))<sup>1</sup>
- c) che l'utente abbia un minimo di competenze per aprire un file PHP apportarvi delle modifiche e installarlo in una directory del Raspberry

La presente guida è inoltre realizzata per un impianto fotovoltaico e domestico così strutturato:

- 1) impianto fotovoltaico monofase con un solo inverter
- 2) impianto domestico monofase

In altri casi (impianti trifasi o con più inverter) è comunque possibile utilizzare la presente guida, ma alcuni passaggi dovranno essere adattati allo specifico caso. Nello specifico infatti il software 123solar e MeterN gestiscono anche impianti multinverter.

<sup>1</sup> Nel caso non abbiate installato un impianto fotovoltaico ma intendiate usare soltanto meterN per il monitoraggio dei consumi, è possibile anche non installare 123solar, ma la presente guida prevede comunque che alcuni passaggi della guida di Walter62 vengano comunque eseguiti. In tal caso eseguite i capitoli da 1 a 5 (tutto) della "[Guida RPi datalogger rev10](#)" ed eventualmente il capitolo 10 per la parte relativa all'installazione del servizio di invio email utilizzabile anche con MeterN

## 1.1 Convenzioni testo

Nella seguente guida si adotteranno le seguenti convenzioni.

```
sudo nano /etc/init.d/samba
```

Nei riquadri a sfondo azzurro sono illustrati i comandi da eseguire tramite terminale

```
# X-Start-Before: rsyslog
```

Nei riquadri a sfondo giallo sono illustrate le modifiche da apportare al contenuto dei vari file o il comando da inserire nei vari campi di MeterN

## 1.2 Ringraziamenti

Ritengo doveroso ringraziare chi ha collaborato in quest'impresa:

- **Jean-Marc Louviaux:** l'autore dei software [123solar](#) e [MeterN](#)
- **Gianfrd:** l'autore del software [sdm120c](#) per la lettura dei dati dal contatore RS485 e di vari script per l'utilizzo di vari sensori mediante MeterN
- **Ninodifranco, TheDrake** ed altri utenti del forum [energeticambiente.it](#) per i test, gli spunti ed i suggerimenti dati

SE VI INTERESSA ACQUISTARE IL CONTATORE  
EASTRON SDM120modbus o SDM220modbus :





 **Compra adesso** 

SE IL MIO LAVORO VI E' STATO UTILE, OFFRITEMI DA BERE,  
FATEMI UNA DONAZIONE :

**Donazione**

## 2 MATERIALE NECESSARIO

### 2.1 Hardware

<p><b>1 Raspberry Pi</b> Va bene qualsiasi modello</p> 	<p>1 (o 2) contatori monofase modbus <a href="#">EASTRON SDM120-modbus</a> (in precedenza identificato anche come SDM120C) oppure 1 (o 2) contatori monofase modbus <a href="#">EASTRON SDM220-modbus</a></p> 
<p><b>1 scheda SD</b> con installato Raspbian "wheezy" e 123solar<sup>2</sup> - Consiglio una schedina da minimo 8Gb classe 10 per avere spazio per il log dei dati</p>	<p><b>1 (o 2) adattatori USB-RS485<sup>3</sup></b></p> 
<p>Doppino ritorto per RS485 per collegare il contatore al Raspberry (per brevi tratti, va bene anche un buon cavo di rete schermato in cui si utilizzerà soltanto un solo doppino)</p> 	<p>Cavi ed attrezzi vari</p>

### 2.2 Software

Per la corretta installazione e configurazione vi consiglio di avere un PC windows con i seguenti software:

- [WinSCP](#) per la copia dei file sul Raspberry
- [Putty](#) per il collegamento in modalità terminale al Raspberry
- [Notepad++](#) o altro editor di testo per modificare i file di configurazione
- [7Zip](#) o altro software per scompattare file zip e tar

<sup>2</sup> Nel caso non abbiate installato un impianto fotovoltaico ma intendiate usare soltanto meterN per il monitoraggio dei consumi, è possibile anche non installare 123solar, ma la presente guida prevede comunque che alcuni passaggi della guida di Walter62 vengano comunque eseguiti. In tal caso eseguite i capitoli da 1 a 5 (tutto) della "[Guida RPi datalogger rev10](#)" ed eventualmente il capitolo 10 per la parte relativa all'installazione del servizio di invio email utilizzabile anche con MeterN

<sup>3</sup> per gli adattatori USB-RS485, in rete se ne trovano di diversi tipologie. Personalmente mi sono trovato bene con il modello illustrato, che si trova in rete per pochi euro e che viene immediatamente riconosciuto dal Raspberry e che funziona benissimo. Vi consiglio di averne in casa anche uno in più di scorta, in quanto può succedere che saltino e non rispondano più.

Per chi avesse poca dimestichezza nel collegarsi al Raspberry tramite il software Putty o WinSCP consiglio di leggersi le seguenti guide:

- [Accesso via SSH - ovvero come usare i client SSH \(Putty e WinSCP\)](#)
- [Come controllare da remoto il vostro Raspberry Pi usando SSH](#)

Nel caso non abbiate ancora ben chiaro la cosa, in rete si trovano svariate guide a riguardo.



## 3 MISURA CON CONTATORE MODBUS (RS485)

### 3.1 Modbus: protocollo universale

Modbus è un protocollo di comunicazione seriale creato da un importante produttore di PLC nel lontano 1979. A differenza di molti altri protocolli di comunicazione è pubblicato apertamente ed è "royalty-free", ciò ha permesso ad ogni produttore di PLC, Touch Panel ed in genere di schede elettroniche di implementarlo all'interno dei propri dispositivi, rendendolo così lo standard defacto per l'Automazione e la Domotica.

Modbus è un protocollo sicuro, infatti integra il controllo CRC su ogni messaggio per verificare l'integrità dei dati; tipicamente viene utilizzato su linee seriali RS-232 e RS-485, mentre la versione TCP-IP è denominata "Modbus TCP".

Per chi volesse approfondire gli aspetti tecnici di questo protocollo di comunicazione vi consiglio di leggere questa interessante guida: [Il protocollo ModBus](#)

### 3.2 Contatore EASTRON SDM120 modbus

Si tratta di un compatto analizzatore di energia monofase avanzato, dotato di display LCD per la visualizzazione immediata di tutti i parametri energetici della nostra linea elettrica.

Le principali caratteristiche sono:

- Contenitore da 1 modulo DIN
- un solo pulsante (la modifica dei parametri di configurazione è possibile solo via software)
- morsetti per cavi di sezione fino a 6mm<sup>2</sup>
- Corrente max 45A

Qui potete trovare una mia dettagliata recensione:

[Contatore EASTRON SDM120C Modbus per monitoraggio energetico](#)

### 3.3 Contatore EASTRON SDM220 modbus

Un'alternativa al contatore SDM120 è il fratello maggiore SDM220 modbus. Si tratta di un contatore con le medesime funzionalità, ma dispone di:

- Contenitore da 2 moduli DIN
- un display LCD più ampio
- due pulsanti che permettono anche la modifica dei parametri di configurazione
- morsetti per cavi di sezione fino a 16mm<sup>2</sup>
- Corrente max 80A

Nelle seguenti pagine della guida si parlerà sempre di SDM120, ma è possibile utilizzare senza nessuna modifica anche il contatore SDM220.



Qui potete trovare una mia dettagliata recensione:

[Contatore EASTRON SDM220 Modbus per monitoraggio energetico](#)

### 3.4 Dove acquistare i contatori

Potete acquistare i contatori EASTRON SDM120modbus e SDM220modbus nel mio negozio ebay:

SE VI INTERESSA ACQUISTARE IL CONTATORE  
EASTRON SDM120modbus o SDM220modbus :

**Compra adesso**

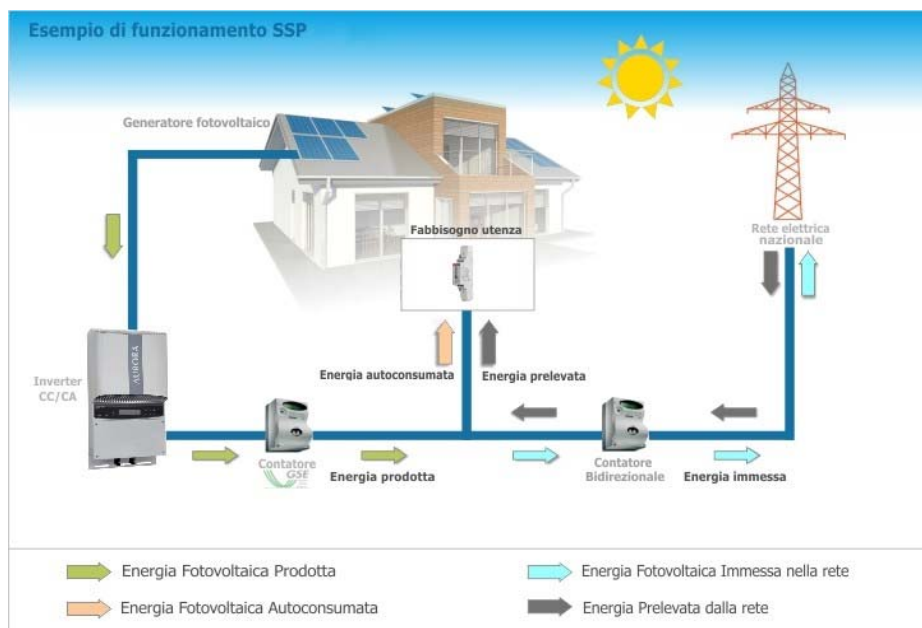
### 3.5 Collegare il contatore all'impianto elettrico

Al fine di poter acquisire i consumi domestici, il contatore andrà installato all'interno del nostro quadro elettrico generale di casa.

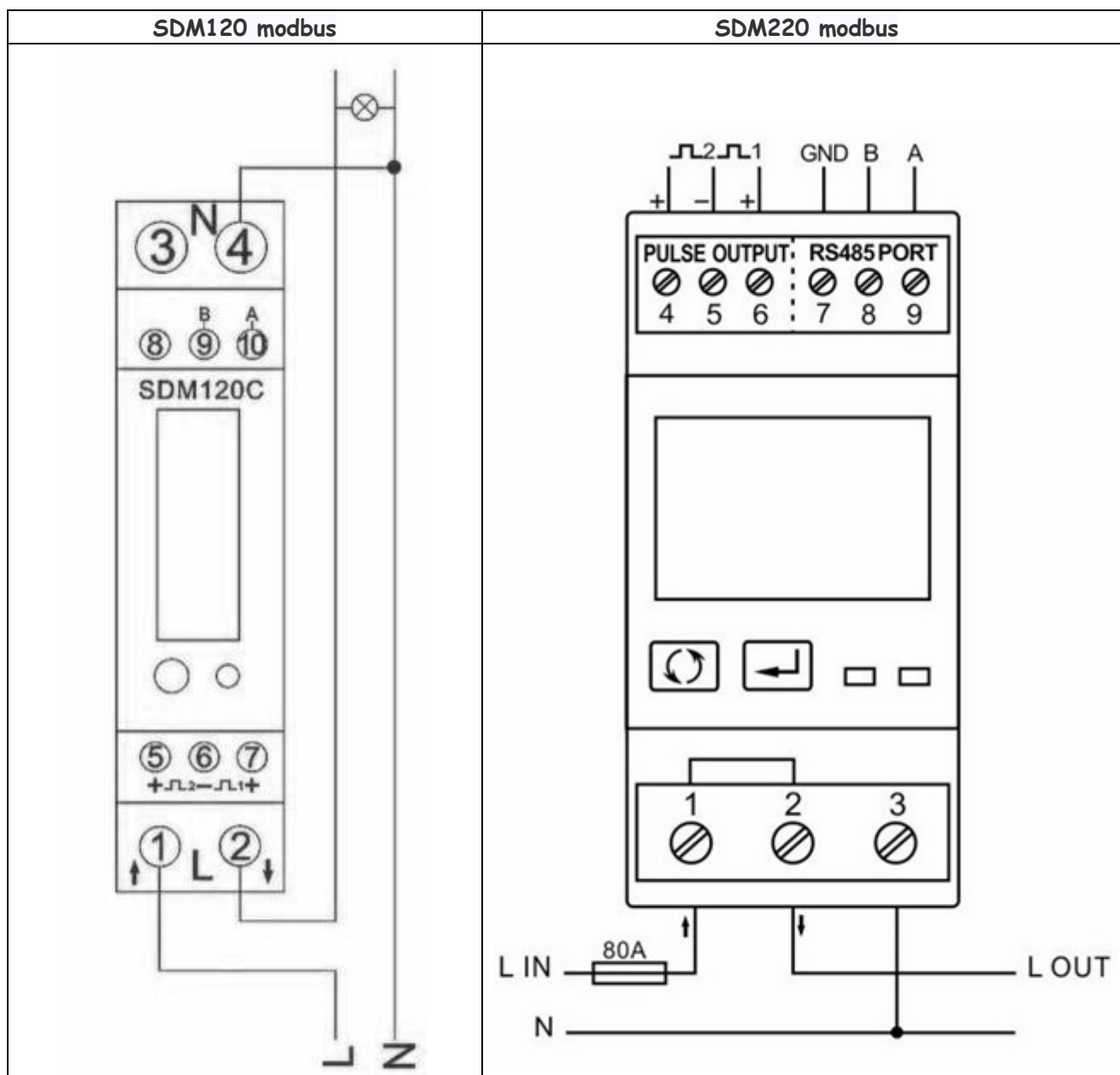
Viste le ridotte dimensioni (un solo modulo da 18mm per il modello SDM120 o due moduli per il modello SDM220) si dovrebbe poter facilmente installare quasi ovunque.

La presente guida prevede di installare il contatore in modo da misurare il totale dei nostri consumi domestici, mentre tutte le altre grandezze (prelievi, immissioni ed autoconsumo) saranno ricavati per differenza per mezzo del file eflow.php (configurazione indicata al punto 3.6.2).

La corretta posizione di installazione per un classico impianto in "Scambio Sul Posto", sarà dunque la seguente:



Di seguito illustriamo lo schema di collegamento del contatore SDM 120 modbus e SDM 220 modbus.



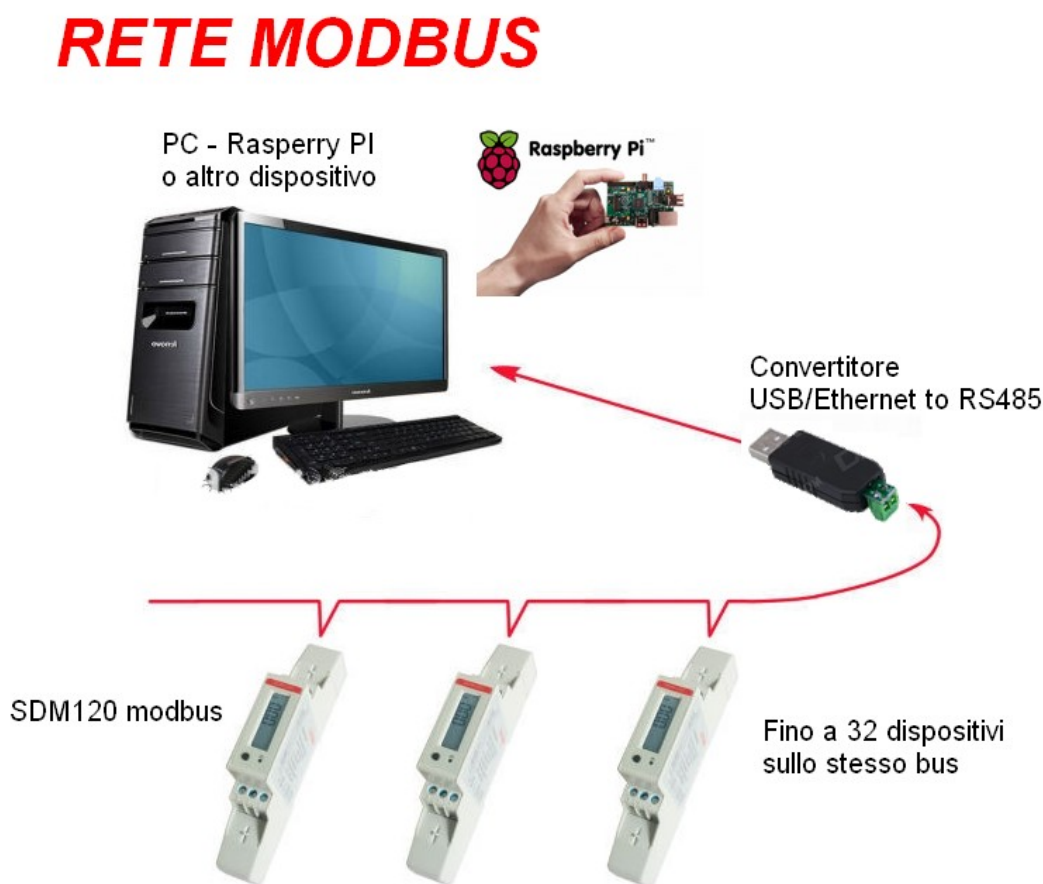
Il precedente schema è solo esemplificativo, in quanto lo schema potrebbe venire variato dal costruttore, pertanto attenetevi a quanto previsto dalle istruzioni d'uso del vostro contatore.

**NOTA:** L'utilizzo di questi contatori comporta la necessità di intervenire sul vostro impianto domestico a 230V. Se non avete un minimo di conoscenza e pratica di impianti elettrici, fate installare il contatore da un vostro elettricista di fiducia.

**Non ci si ritiene responsabili per qualsiasi danno possiate provocare da un uso improprio di quanto riportato nella presente guida**

### 3.6 Lettura dati via RS485

La possibilità di disporre di un'uscita RS485 sul contatore permette di poter installare nel nostro impianto fino a 32 contatori per monitorare quanti e quali carichi vogliamo. Basterà infatti collegarli fra loro con un semplice doppino ritorto e collegarli al Raspberry mediante un economico adattatore USB-RS485, per poter monitorare e loggare tutti parametri energetici che desideriamo, mediante i già noti software 123solar e MeterN:

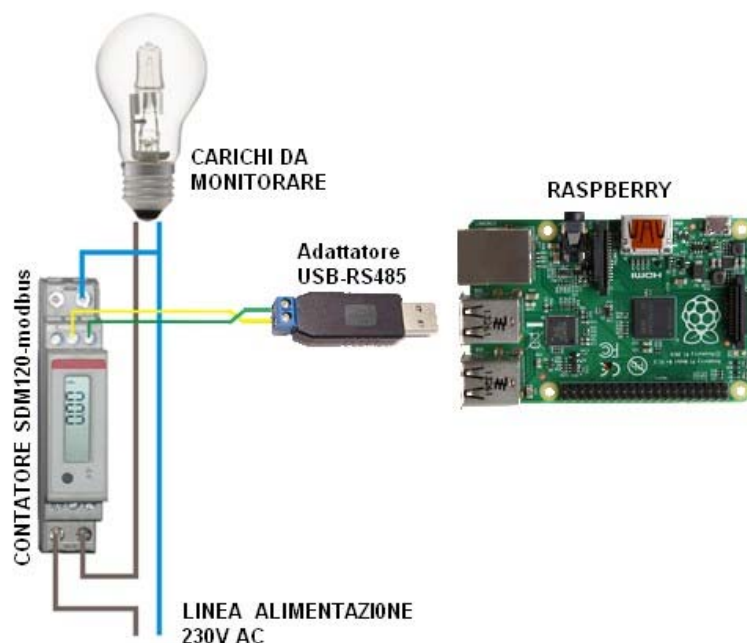


Grazie ad un software sviluppato da Gianfranco Di Prinzio (gianfrdp del forum [energeticamebiente.it](http://energeticamebiente.it)) che ha messo gentilmente a disposizione su [Github](https://github.com), è possibile leggere tutti i valori da questo contatore. Il software prevede inoltre le necessarie integrazioni per poter utilizzare il contatore mediante i software MeterN e 123solar. Infatti la possibilità di poter leggere più contatori sullo stesso bus, permette di utilizzare 123solar anche con inverter non direttamente compatibili, ma leggendo anche la produzione fotovoltaica (e non solo i consumi) mediante un contatore.

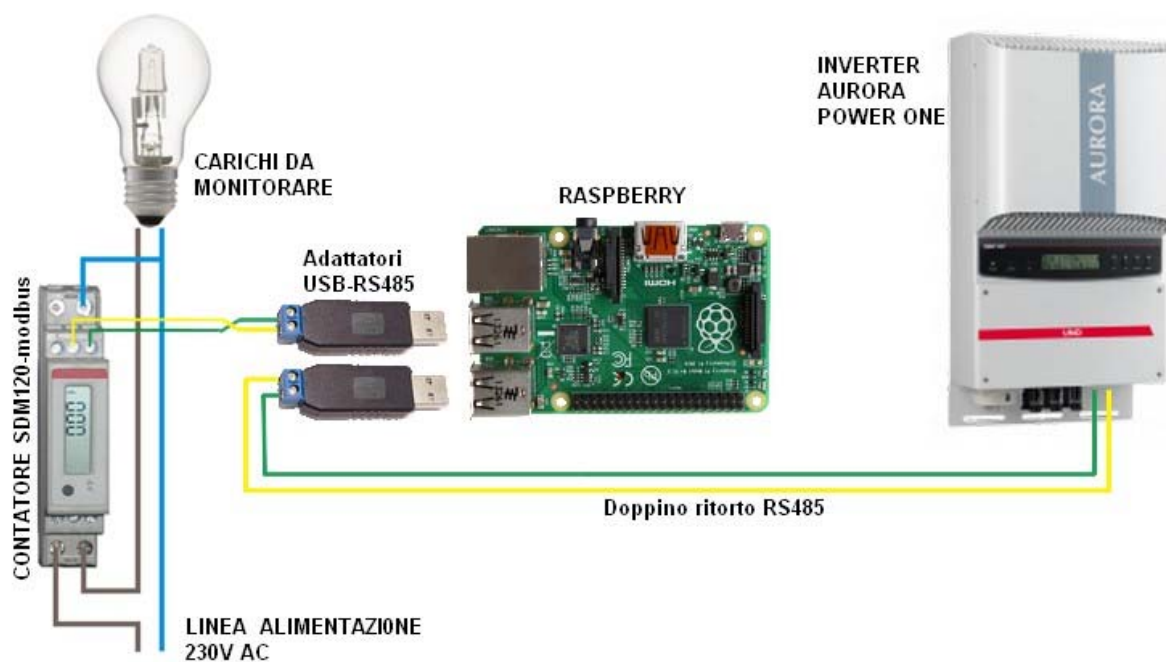
### 3.7 Collegare il contatore al Raspberry

Di seguito si riportano alcuni schemi di collegamento del sistema di monitoraggio mediante Raspberry in diverse configurazioni tipo.

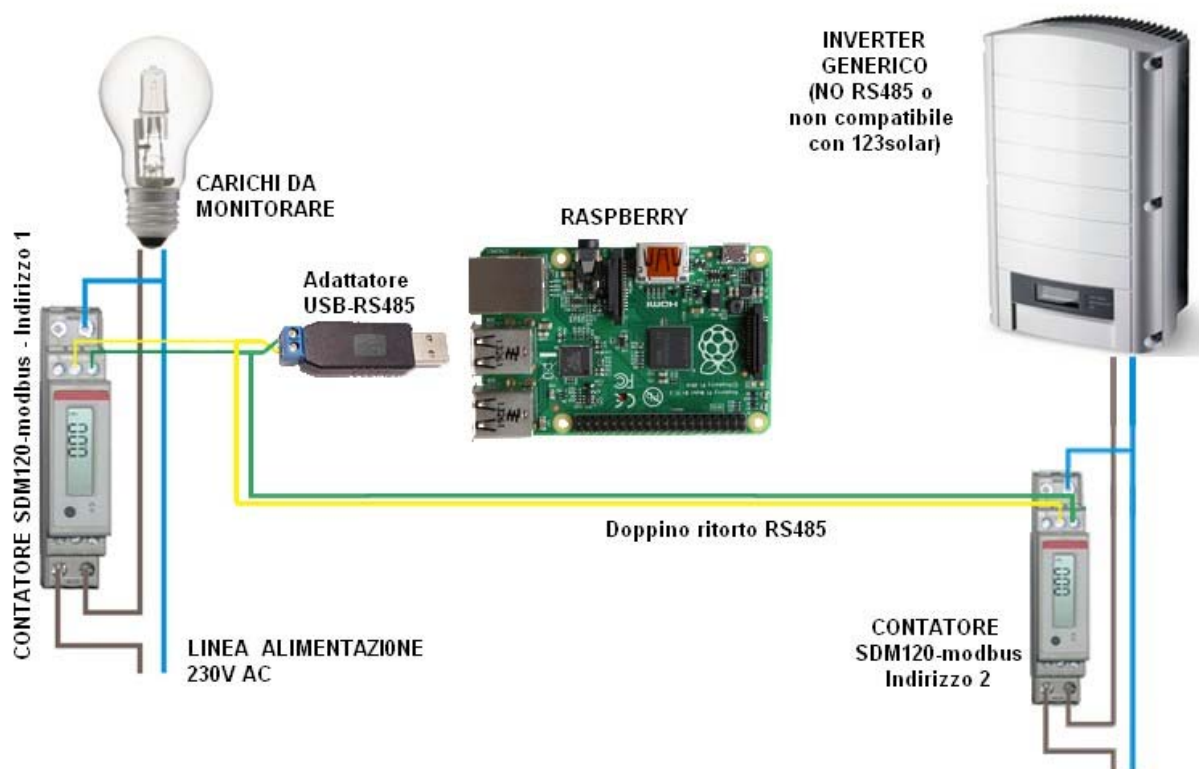
#### 3.7.1 Solo monitoraggio consumi (assenza fotovoltaico)



#### 3.7.2 Inverter Aurora Power-One e contatore SDM120 modbus



### 3.7.3 Inverter generico e 2 contatori SDM120 modbus



In questo caso dovremmo porre particolare attenzione al fatto che i due contatori dovranno avere indirizzi diversi.

Attenersi eventualmente a quanto indicato nell'[APPENDICE A](#) per la modifica dell'indirizzo.

## ATTENZIONE

LA PRESENTE GUIDA È DEDICATA PER LA CONFIGURAZIONE INDICATA AL PUNTO 3.7.2  
CON INVERTER AURORA POWER-ONE (O ALTRO COMPATIBILE CON 123SOLAR) ED UN  
SOLO CONTATORE SDM120/SDM220 MODBUS

PER LA ALTRE CONFIGURAZIONI SONO NECESSARIE DELLE MODIFICHE  
RISPETTO A QUANTO RIPORTATO NELLA GUIDA

### 3.8 Cablaggio rete modbus

Come potete vedere il collegamento nelle varie configurazioni è comunque molto semplice. Basterà collegare mediante un doppino ritorto le uscite RS485 del contatore (identificate con le lettere A e B) con le rispettive lettere riportate sulla chiavetta usb.

**FATE ATTENZIONE A RISPETTARE LA POLARITA' E COLLEGARE IL TUTTO CORRETTAMENTE:**



CONTATORE

ADATTATORE USB-RS485

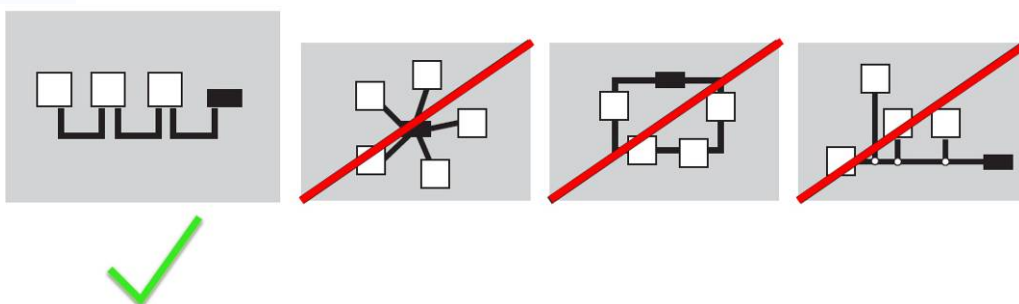
A (morsetto 10)	-----	A (D+)
B (morsetto 9)	-----	B (D-)



Nel caso si installino più dispositivi sullo stesso bus si dovrà creare una catena come l'immagine di seguito:



A differenza della rete di computer LAN o del cablaggio a 230 V di casa, con molti bus la disposizione ed il collegamento dei dispositivi non sono casuali. Con il bus RS485, i singoli elementi vengono collegati in sequenza a un'unica linea continua, come perle in una collana. In questa tipologia, le ramificazioni non sono consentite



Nel caso di reti lunghe andrebbero anche inserite all'inizio ed alla fine delle resistenze di terminazione da 120ohm. Fino a 30-35m personalmente non ho avuto problemi e non ho avuto bisogno delle resistenze di terminazione

Per un approfondimento tecnico sulle reti RS485 suggerisco di leggersi i seguenti documenti:

- [Corretto cablaggio delle reti RS485](#)
- [Guida alla rete RS485](#)



## 4 INSTALLAZIONE METERN

Per installare il software sul Raspberry, basterà seguire passo passo le istruzioni in seguito riportate.

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
..# cd /var/www
```

Scaricare il programma meterN 0.7.7.3 (verificare a [questa pagina](#) che sia l'ultima versione e modificare se necessario)

```
../var/www# wget http://www.123solar.org/downloads/metern/metern0.7.7.3.tar.gz
```

Scompattiamo l'archivio

```
../var/www# tar -xvzf metern0.7.7.3.tar.gz
```

eliminiamo il file tar appena scaricato

```
../var/www# rm -v metern0.7.7.3.tar.gz
```

Abbiamo così finito l'installazione di MeterN.


Collegiamoci al Raspberry con WinSCP per verificare che tutto sia stato installato correttamente.

Nella cartella /var/www/ ora dovremmo trovare anche una cartella metern assieme a quella di 123solar come di seguito:

/var/www				
Nome	Estensione	Dimensi...	Modificato	Diritti
↑ ..			25/06/2014 00:46:22	rwxf-xr-x
123solar			19/08/2014 22:56:26	rwxfwxfwxf
metern			13/09/2014 07:30:11	rwxfwxfwxf

e nella cartella MeterN dovremmo avere:



/var/www/metern				
Nome	Estensione	Dimensi...	Modificato	Diritti
			08/11/2014 17:10:37	rw-r--r--
admin			15/09/2014 08:23:36	rw-rw-rw-
comapps			13/09/2014 11:03:40	rw-rw-rw-
config			13/09/2014 11:02:49	rw-rw-rw-
data			13/09/2014 11:02:15	rw-rw-rw-
images			13/09/2014 11:02:16	rw-r--r--
js			10/09/2014 14:52:29	rw-r--r--
languages			10/09/2014 14:52:20	rw-r--r--
programs			13/09/2014 11:02:16	rw-r--r--
scripts			13/09/2014 07:37:36	rw-rw-rw-
styles			21/10/2014 15:47:17	rw-r--r--
dashboard.php		6.429 B	13/09/2014 07:26:05	rw-r--r--
detailed.php		5.730 B	13/09/2014 07:26:05	rw-r--r--
favicon.ico		1.150 B	13/09/2014 07:26:05	rw-r--r--
index.php		14.753 B	13/09/2014 07:26:05	rw-r--r--
indexdetailed.php		2.393 B	13/09/2014 07:26:05	rw-r--r--
indexreadings.php		6.915 B	13/09/2014 07:26:05	rw-r--r--
README.txt		3.863 B	13/09/2014 07:26:43	rw-r--r--

Dalla versione 0.7.6 in poi, l'autore ha preferito non distribuire assieme al software anche le varie applicazioni per la lettura dei dati.

Dovremo pertanto scaricare il file [comapps\\_examples20150822.tar](#), scompattarlo e copiare il suo contenuto nella cartella /var/www/metern/comapps procedendo come di seguito. (verificare a [questa pagina](#) che sia l'ultima versione e modificare se necessario)

Scaricate pertanto il file del link precedente sul vostro PC e scompattatelo con 7Zip sul desktop. Copiate poi con WinSCP tutto il contenuto della cartella comapps\_examples nella cartella /var/www/metern/comapps sul Raspberry.

Scarichiamo inoltre i file eflowlive sviluppati da ninodifranco che meglio si adattano per l'uso con i contatori modbus. Alcuni dei file precedenti verranno sostituiti.

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue:

```
cd /var/www/metern/comapps
wget http://www.flanesi.it/blog/download/eflowlive_rev_3.1.zip
unzip -o eflowlive_rev_3.1.zip
rm eflowlive_rev_3.1.zip
sudo chmod a+x *
```

Questi file risultano già preimpostati per essere immediatamente utilizzati con il nostro sistema.

Sempre da terminale creiamo inoltre i link simbolici ai vari file:

```
sudo -s
ln -s /var/www/metern/comapps/eflow.php /usr/bin/eflow
ln -s /var/www/metern/comapps/eflowlive.php /usr/bin/eflowlive
ln -s /var/www/metern/comapps/pool123s.php /usr/bin/pool123s
ln -s /var/www/metern/comapps/poolerconsumi.php /usr/bin/poolerconsumi
ln -s /var/www/metern/comapps/pooler485.sh /usr/local/bin/pooler485
```

## 4.1 Avvio di MeterN al boot

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
..# cd /etc
../etc# nano rc.local
```

Editare il file /etc/rc.local inserendo le modifiche in rosso:

```
stty -F /dev/ttyUSB0 19200 &
sudo /usr/bin/curl http://localhost/123solar/scripts/boot123s.php &
sudo sleep 6
sudo /usr/bin/curl http://localhost/metern/scripts/bootmn.php &
exit 0
```

Premere ctrl+O per salvare e ctrl+X per uscire

## 5 INSTALLAZIONE SOFTWARE MODBUS

Per poter leggere i valori del contatore tramite modbus, e necessario installare il software sviluppato specificatamente per i contatori SDM120 e SDM220.

### 5.1 Installare la libreria libmodbus

Il software che useremo per la lettura dei dati dal contatore, si basa sulla libreria libmodbus. E' pertanto necessario come prima cosa, installare questa libreria sul Raspberry.

Per installare e compilare l'ultima versione della libreria [libmodbus](#), eseguire da terminale (Putty) i seguenti comandi:

```
sudo -s  
cd /..  
apt-get update  
apt-get upgrade  
apt-get install libmodbus-dev
```

### 5.2 Installare script sdm120c

Installata la libreria ora non ci resta che installare il [software sdm120c](#). Sempre da terminale digitiamo:

```
cd /home/pi  
git clone https://github.com/gianfrdp/SDM120C
```

Dovremmo ora procedere a compilare il software:

```
cd SDM120C/  
make clean && make  
sudo cp sdm120c /usr/local/bin/
```

Assicuriamoci che all'utente www-data siano concessi i permessi di lettura/scrittura sui dispositivi seriali eseguendo il comando:

```
sudo adduser www-data dialout
```

Riavviamo con il comando:

```
sudo reboot
```

A questo punto tutto è pronto.

## 6 PARAMETRI COMUNICAZIONE CONTATORE

Una volta completati tutti i collegamenti del contatore, è importante capire come quest'ultimo risulta configurato, prima di poter procedere.

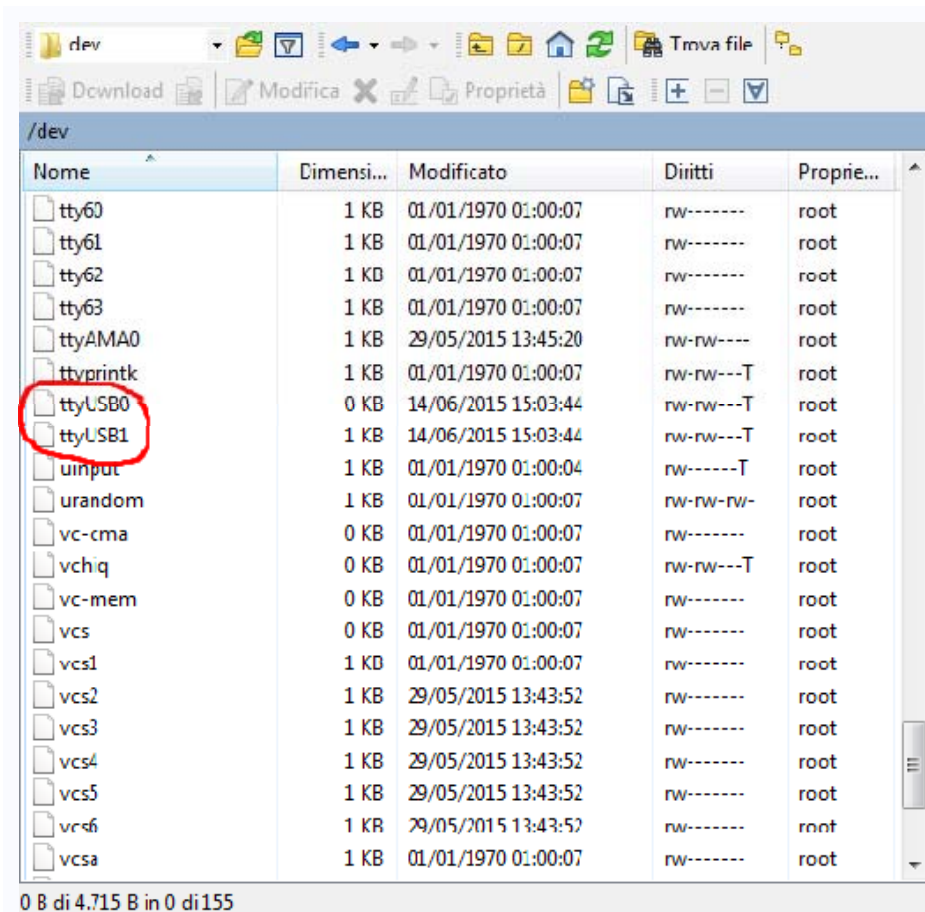
I parametri da individuare ed appuntarsi per le successive configurazioni saranno:

- porta USB con cui il Raspberry identifica l'adattatore USB-RS485
- indirizzo modbus contatore
- velocità di comunicazione modbus del contatore
- parità della comunicazione modbus

### 6.1 Porta USB Raspberry

Nel caso abbiamo collegato al Raspberry un solo adattatore USB-RS485 la porta molto probabilmente sarà *ttyUSB0*.


Se vogliamo però essere certi su quale porta il Raspberry ha connesso l'adattatore USB-RS485, si deve guardare nella directory */dev* del Raspberry mediante WinSCP:



Nel mio caso avendo due adattatori collegati il Raspberry li ha configurati come *ttyUSB0* e *ttyUSB1*.

## 6.2 Indirizzo e velocità contatore

Per individuare questi parametri, basterà premere più volte il pulsante presente sull'sdm120c. In tal modo è possibile visualizzare le varie grandezze misurate dal contatore e ad un certo punto leggere anche l'indirizzo e la velocità :

	Indirizzo modbus (in genere 1)
	Velocità modbus (2400, 4600 o 9600, in genere 2400)
	Parità modbus (O = Odd ; E = Even ; N = None)  <b>NOTA:</b> questo dato è presente solo nelle ultime versioni dei contatori (vedi paragrafo successivo)

Appuntiamoci quindi l'indirizzo la velocità e la parità che useremo in seguito.

## 6.3 Parità

Nel caso in cui il vostro contatore non visualizzi la parità, possiamo individuarla come indicato in seguito. Se invece il vostro contatore visualizza la parità come indicato sopra, saltate questo paragrafo.

In genere la parità impostata di default è Even sui primi dispositivi prodotti, mentre è None su quelli più recenti.

Se sul vostro contatore non è però possibile leggerla direttamente a display, dovremo pertanto procedere per tentativi come illustrato di seguito.

Supponendo di aver individuato come indirizzo 1, velocità 2400 e porta ttyUSB0, digitare da terminale:

```
sdm120c -a 1 -P N -b 2400 /dev/ttyUSB0
```

se il contatore risponde la parità sarà None altrimenti digitare:

```
sdm120c -a 1 -P E -b 2400 /dev/ttyUSB0
```

se il contatore risponde la parità sarà Even altrimenti digitare:

```
sdm120c -a 1 -P O -b 2400 /dev/ttyUSB0
```

se il contatore risponde la parità sarà Odd.

Nel caso che per il vostro contatore l'indirizzo, la velocità o la porta sia diversa, basterà modificare il rispettivo valore ai numeri evidenziati in rosso nelle precedenti righe.

Se tutto è andato per il verso giusto ora dovrete avere:

- Indirizzo modbus
- Velocità modbus
- Porta adattatore USB
- Parità comunicazione modbus

## 6.4 Modifica indirizzo e velocità contatore

Affinché il sistema di monitoraggio funzioni correttamente, è necessario che l'indirizzo modbus del contatore corrisponda al numero di meter che assegneremo successivamente ai consumi (2).

Procediamo quindi ad impostare l'indirizzo del contatore a 2 e la velocità del modbus a 9600, in modo da rendere molto più veloce la lettura dei dati.

Normalmente il contatore arriva preimpostato con indirizzo 1 e con velocità 2400 pertanto, come indicato nell'[APPENDICE A](#), dovremmo eseguire quanto segue.

### 6.4.1 Modifica indirizzo contatore

Per cambiare l'indirizzo del dispositivo da 1 a 2, premere il pulsante frontale sul contatore per 3 secondi, fino a che compare la scritta - SET - sul display, quindi da terminale digitare:

Se parità N :

```
sdm120c -a 1 -s 2 /dev/ttyUSB0  
New address 2  
You have to restart the meter for apply changes
```

Se parità E:

```
sdm120c -a 1 -s 2 -P E /dev/ttyUSB0  
New address 2  
You have to restart the meter for apply changes
```

Riavviare il contatore staccando e riattaccando la fase in ingresso.

## 6.4.2 Modifica velocità modbus

Per cambiare la velocità di trasmissione da 2400 a 9600 premere il pulsante frontale sul contatore per 3 secondi, fino a che compare la scritta - SET - sul display, quindi da terminale digitare:

Se parità N :

```
sdm120c -a 2 -r 9600 -P N /dev/ttyUSB0
New baud_rate 9600
You have to restart the meter for apply changes
```

Se parità E :

```
sdm120c -a 2 -r 9600 -P E /dev/ttyUSB0
New baud_rate 9600
You have to restart the meter for apply changes
```

Riavviare il contatore staccando e riattaccando la fase in ingresso.

## 6.5 Test di lettura contatore

Per essere certi che tutto funzioni correttamente, eseguiamo ora un test di lettura del contatore. A seconda del valore di parità dovremmo digitare il relativo comando di test:

Indirizzo modbus	Velocità modbus	Porta USB	Parità	Comando test
2	9600	tttyUSB0	O	sdm120c -a 2 -P O -b 9600 /dev/ttyUSB0
2	9600	tttyUSB0	E	sdm120c -a 2 -P E -b 9600 /dev/ttyUSB0
2	9600	tttyUSB0	N	sdm120c -a 2 -P N -b 9600 /dev/ttyUSB0

La corretta risposta al comando di test dovrebbe già restituire l'elenco di tutti i valori disponibili:

```
sdm120c -a 2 -P E -b 9600 /dev/ttyUSB0
Voltage: 218.30 V
Current: 0.00 A
Power: 0.00 W
Power Factor: 1.00
Frequency: 50.00 Hz
Import Active Energy: 6409 Wh
Export Active Energy: 0 Wh
Total Active Energy: 6409 Wh
OK
```



I valori ovviamente potranno essere diversi da quelli illustrati e corrispondenti a quelli rilevati in quel momento dal vostro contatore.

Se il contatore non risponde correttamente ricontrollate quanto fatto in precedenza in quanto in queste condizioni MeterN non potrà funzionare.

Per approfondire il funzionamento dello script consultare l'[APPENDICE A](#) della presente guida.

## 7 CONFIGURARE I METERS (MISURATORI)

Non ci resta ora che configurare i vari meters (cioè misuratori) di MeterN.

Prima di procedere alla configurazione tramite interfaccia web, dobbiamo ancora fare qualche modifica a vari file.

### 7.1 Modifica file pooler485 per lettura consumi

Con WinSCP andiamo ad editare il file pooler485.sh che si trova nella cartella /var/www/metern/comapps.

Cerchiamo la seguente riga:

```
CMD="sdm120c -a ${ADDRESS} -b ${BAUD_RATE} -z 10 -i -p -v -c -f -g -P N -w  
5 -q ${DEVICE}"
```

ed andiamo a modificare (se necessario) il parametro della parità della comunicazione modbus (evidenziato in rosso) con il valore che abbiamo individuato precedentemente (inserite N oppure E oppure O).

### 7.2 Avvio file pooler485 per lettura consumi

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
sudo -s  
cd /var/www/metern/config  
nano config_daemon.php
```

Editare il file config\_daemon.php inserendo le modifiche in rosso e cancellando i commenti (//) ad inizio riga come di seguito:

```
<?php
if (isset($_SERVER['REMOTE_ADDR'])) {
    die('Direct access not permitted');
}
// Startup of the com apps daemon as 'http' user if needed
$output = shell_exec('pkill -f pooler485 > /dev/null 2>&1 &');
$output = shell_exec('pooler485 2 9600 /dev/ttyUSB0 > /dev/null 2>/dev/null &');

//$output = shell_exec('/usr/bin/curl http://192.168.1.12/startsdm.php');
//$output = shell_exec("/srv/http/metern/comapps/poolmeters.py live > /dev/null
2>&1 &");
?>
```

Nel caso la vostra porta dell'adattatore USB-RS485 fosse diversa modificate opportunamente la parte /dev/ttyUSB0

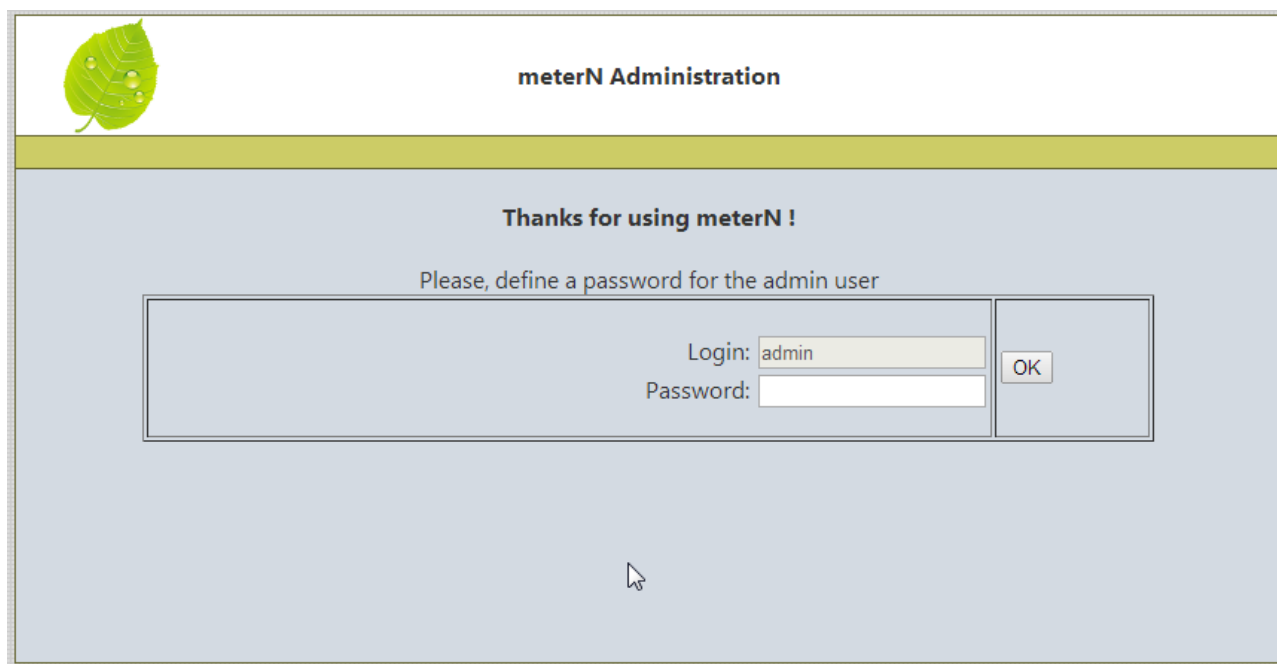
Premere ctrl+O per salvare e ctrl+X per uscire

Il contenuto del file *config\_daemon.php* viene avviato all'avvio di MeterN, pertanto questo ci assicura che il file pooler485 sia in esecuzione quando e solo quando è in esecuzione anche MeterN.

## 7.3 Pagine di amministrazione

Per configurare MeterN procederemo mediante la pagina web di amministrazione, accessibile al seguente indirizzo:

[http://IP\\_RASPBERRY/metern/admin/](http://IP_RASPBERRY/metern/admin/)



**meterN Administration**

Thanks for using meterN !

Please, define a password for the admin user

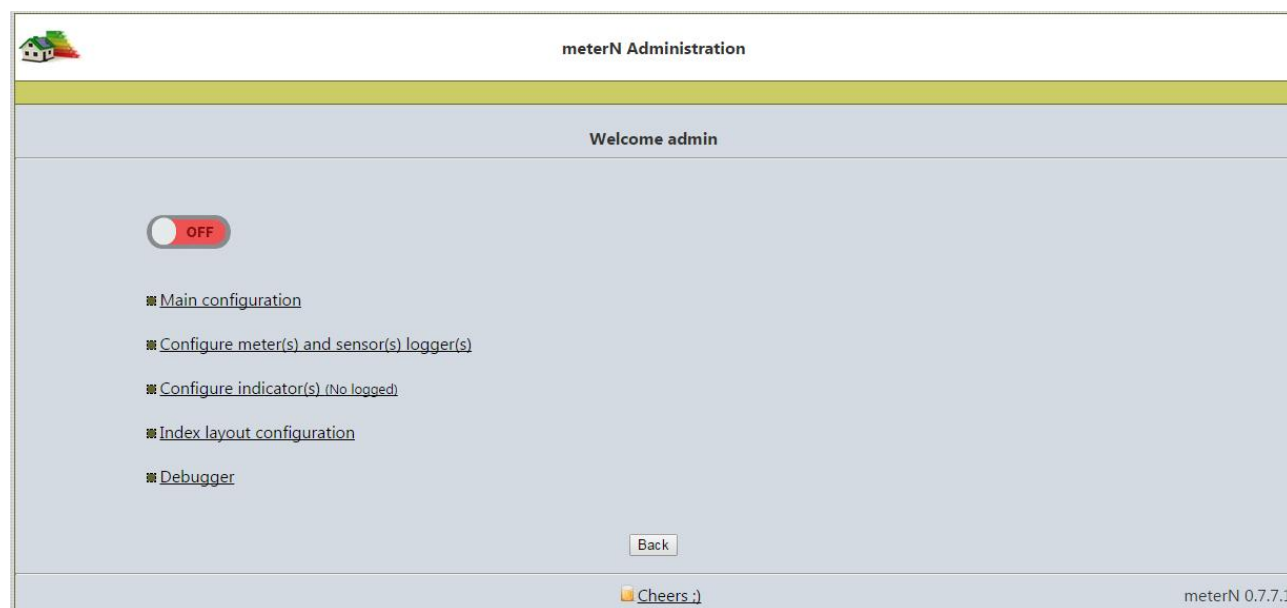
Login: admin

Password:

OK

Al primo accesso dovreste definire una password di accesso al sistema e successivamente verrà chiesto il login a cui risponderete con admin e password che avete appena definito.

Si aprirà quindi il menù di configurazione di MeterN, molto simile a quello di 123solar.



**meterN Administration**

Welcome admin

OFF

- Main configuration
- Configure meter(s) and sensor(s) logger(s)
- Configure indicator(s) (No logged)
- Index layout configuration
- Debugger

Back

Cheers :)

meterN 0.7.7.1

Il pulsante rosso serve per attivare/disattivare MeterN. Per il momento lo lasceremo disattivo, in quanto dobbiamo prima procedere a completare la configurazione del sistema.

## 7.4 Configurazione principale

Selezionare "Main configuration"

compilate i campi come di seguito:

**Number of meters:** inserite

- **2** nel caso vogliate solo monitorare la produzione del fotovoltaico e i vostri consumi domestici (in tal caso la lettura di produzione verrà effettuata direttamente dai valori rilevati da 123solar, mentre i consumi saranno rilevati tramite il contatore ad impulsi);
- **5** se volete monitorare Produzione, Consumi, Prelievi da rete, Immissioni in rete, Autoconsumo (in tal caso la lettura di produzione verrà effettuata direttamente dai valori rilevati da 123solar, i consumi saranno rilevati tramite il contatore ad impulsi, mentre Prelievi, Immissioni ed autoconsumo saranno calcolati in modo automatico con l'impiego del file di MeterN eflow.php)

**Localization:**

Impostate come Timezone **Europe/Rome**, e premete poi sul bottone "Edit Location", per individuare sulla mappa il vostro edificio, cliccatevi e saranno riportate le coordinate geografiche (Latitudine e Longitudine) nei rispettivi campi. Chiudete la mappa cliccando sulla X in alto a destra.

Controllate poi che i restanti campi siano impostati come nella figura precedente.

Salvare cliccando sul bottone "Save config" e quindi cliccare su "Back" per tornare al menu.

## 7.5 Configurazione Misuratori

I misuratori (o meter) di MeterN rappresentano le varie grandezze che vorremo monitorare con il software.

Selezionare ora "Configure your meter(s)/sensor(s)"

## 7.5.1 Misuratore 1 - Produzione

Come già anticipato per la lettura della produzione fotovoltaica utilizzare il file pool123s.php che legge i dati dal software 123solar.

Compilate TUTTI i campi come da figura ed in particolare inserite:

### Main pooling:

Command: pool123s energy

### Dashboard live pooling

Live command: pool123s power

Inserite inoltre la vostra Email nel rispettivo campo se volete abilitare le notifiche via email

Cliccate sul bottone "Save config" per salvare le modifiche

## 7.5.2 Misuratore 2 – Consumi

Selezionate ora il misuratore 2 nel menù a tendina in alto a sinistra

Compilate i campi come da figura ed in particolare inserite:

### Main pooling:

Command: `poolerconsumi 2 energy`

### Dashboard live pooling

Live command: `poolerconsumi 2 power`

Inserite inoltre la vostra Email nel rispettivo campo se volete abilitare le notifiche via email

Cliccate sul bottone "Save config" per salvare le modifiche

### 7.5.3 Misuratore 3 – Prelievi

Selezionate ora il misuratore 3 nel menù a tendina in alto a sinistra

Compilate TUTTI i campi come da figura ed in particolare inserite:

#### Main pooling:

Command: eflow whin

#### Dashboard live pooling

Live command: eflowlive whin

Inserite inoltre la vostra Email nel rispettivo campo se volete abilitare le notifiche via email

Cliccate sul bottone "Save config" per salvare le modifiche



## 7.5.4 Misuratore 4 – Immissioni

Selezionate ora il misuratore 4 nel menù a tendina in alto a sinistra

Compilate TUTTI i campi come da figura ed in particolare inserite:

### Main pooling:

Command: **eflow whout**

### Dashboard live pooling

Live command: **eflowlive whout**

Inserite inoltre la vostra Email nel rispettivo campo se volete abilitare le notifiche via email

Cliccate sul bottone "Save config" per salvare le modifiche

## 7.5.5 Misuratore 5 – Autoconsumo

Selezionate ora il misuratore 5 nel menù a tendina in alto a sinistra

Compilate TUTTI i campi come da figura ed in particolare inserite:

### Main pooling:

Command: **eflow selfc**

### Dashboard live pooling

Live command: **eflowlive selfc**

Inserite inoltre la vostra Email nel rispettivo campo se volete abilitare le notifiche via email

Cliccate sul bottone "Save config" per salvare le modifiche e premete il tasto "Back" per tornare al menù principale.

## 7.5.6 Impostazione "Price per unit"

All'interno di ogni misuratore avrete notato essere presente una voce "Price per unit"

Questo rappresenta il costo pagato o incassato per unità €/Wh ( ed esempio il costo che paghiamo per ogni kWh prelevato dalla rete, oppure il prezzo che il GSE ci riconosce per ogni kWh prodotto dal nostro impianto fotovoltaico se siamo in regime di conto energia.

I valori riportati per questo campo nelle immagini precedenti sono esemplificativi, e dovete pertanto inserire i vostri valori, che potete ricavare secondo le semplici indicazioni seguenti:

- 1 Produzione: nel caso il vostro impianto fotovoltaico sia incentivato secondo uno dei vari conti energia per ogni kWh prodotto, inserite qui l'importo unitario che il GSE vi riconosce per ogni kWh prodotto
- 2 Consumi: non ha molto senso inserire un costo in questo campo, in quanto questi rappresentano i consumi totali della vostra utenza, senza tenere conto dell'eventuale autoconsumo dal

fotovoltaico - lasciate pertanto 0

- **3 Prelievi:** questi rappresentano i vostri prelievi dalla rete e pertanto il consumo che troverete riportato nella vostra bolletta elettrica. In questo caso vi basterà inserire il consumo medio a kWh desumibile dalla bolletta. *(Ad esempio se mediamente ricevete una bolletta bimestrale di 60 € a fronte di un consumo di 240 kWh il valore da inserire sarà:  $60/(240 \times 1000) = 0,00025$  €/Wh - vi consiglio di fare questa media sulle bollette dell'ultimo anno)*
- **4 Immissioni:** valutate se necessario inserire un qualche valore in funzione del sistema di incentivazione riconosciuto dal vostro conto energia
- **5 Autoconsumo:** valutate se necessario inserire un qualche valore in funzione del sistema di incentivazione riconosciuto dal vostro conto energia

## 7.6 Configurazione Layout

Selezionare "Index layout configuration"

Questa pagina vi permette di definire quale sarà l'aspetto dell'interfaccia web di metern ed in particolare:

- **Show graphics in number:** vi permette di definire quanti grafici saranno visualizzati, e quali misuratori (meters) saranno visualizzati sullo stesso grafico
- **Don't fill the serie:** il flag abilita o disabilita il riempimento del grafico. Nel caso in cui si flagga questa casella il misuratore in questione sarà visualizzato come una semplice linea, in caso contrario verrà visualizzata un'area riempita.
- **Show in last 15 days:** potete selezionare quali misuratore visualizzare nel grafico degli ultimi 15 giorni
- **Max power:** definisce il valore di fondo scala per il visualizzatore dei consumi/produzione istantanei

Meter(s)/Sensor(s)	Show in graphic number:	Don't fill the serie:	Show in last 15 days:	Max Power:
#1 Produzione	2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4000 W
#2 Consumi	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4000 W
#3 Prelievi	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5000
#4 Immissioni	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5000
#5 Autoconsumo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5000

Back Save layout

Compilate i campi come da figura.

Cliccate sul bottone "Save layout" per salvare le modifiche e premete il tasto "Back" per tornare al menù principale.

## 8 CONFIGURARE GLI INDICATORS (INDICATORI)

Nelle ultime versioni di MeterN sono stati aggiunti ai meters anche un nuovo tipo di misuratori, chiamati "indicators" cioè indicatori.

La differenza sostanziale è solo che mentre i valori registrati dai meters vengono poi salvati sulla scheda SD (cioè loggati), gli indicatori rappresentano invece dei parametri che vengono solo visualizzati a video nel loro valore istantaneo che assumono in quel momento. Nessuno di questi valori viene salvato.

Vediamo di seguito alcuni esempio di indicatori che possiamo già inserire e che visualizzano gli altri parametri elettrici che ci restituisce il contatore modbus SDM120-modbus o SDM220-modbus. Nei vari appendici sono poi riportati ulteriori indicatori ottenibili aggiungendo al nostro Raspberry vari sensori per la rilevazioni di alcune grandezze come temperatura, pressione atmosferica, umidità, ecc..

Dalla pagina principale di amministrazione selezionare ora "Configure indicator(s) (No logged)"

Ed inserite quindi il numero di indicatori che intendete inserire. Negli esempi di seguito sono **3**.

### 8.1.1 Indicatore 1 - Tensione

Compilate TUTTI i campi come da figura ed in particolare inserite:

Command: `cat /run/shm/metern2.txt | egrep "^2_1\" | grep "*V"`

Indicator#1 Tensione				
Name Tensione	ID 2_1	Value ▾ mode	Command <code>cat /run/shm/metern2.txt   egrep "^2_1\"   grep "*V"</code>	Unit V

### 8.1.2 Indicatore 2 - Corrente

Compilate TUTTI i campi come da figura ed in particolare inserite:

Command: `cat /run/shm/metern2.txt | egrep "^2_2\" | grep "*A"`

Indicator#2 Corrente				
Name Corrente	ID 2_2	Value ▾ mode	Command <code>cat /run/shm/metern2.txt   egrep "^2_2\"   grep "*A"</code>	Unit A

### 8.1.3 Indicatore 3 – Cos φ (fattore di potenza)

Compilate TUTTI i campi come da figura ed in particolare inserite:

Command: `cat /run/shm/metern2.txt | egrep "^2_4\" | grep "*F"`

Indicator#3 Cos φ				
Name Cos φ	ID 2_4	Value ▾ mode	Command <code>cat /run/shm/metern2.txt   egrep "^2_4\"   grep "*F"</code>	Unit

## 9 AVVIO METERN

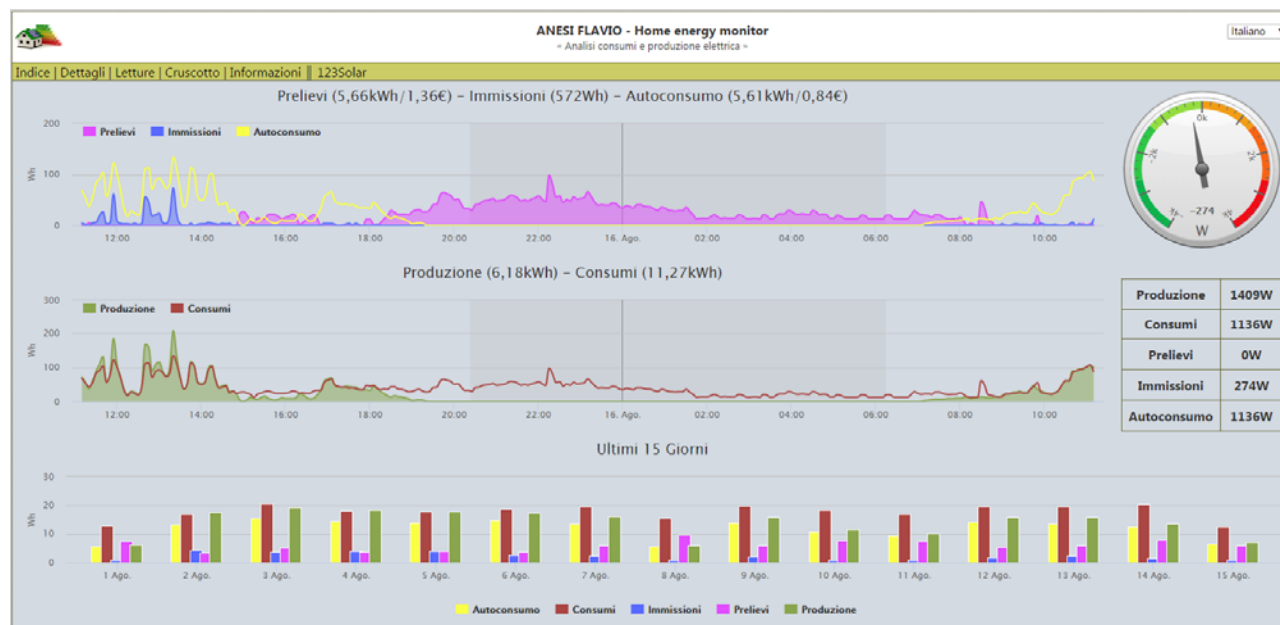
Se avete eseguito tutto correttamente ed anche i test sono andati a buon fine, non ci resta che avviare MeterN.

Dalla pagina di amministrazione , clicchiamo sul pulsante rosso per attivare MeterN



Una volta che appare il pulsante verde ON, non ci resta che aprire la pagina Web di MeterN e goderci il risultato del nostro duro lavoro:

[http://IP\\_RASPBERRY/metern](http://IP_RASPBERRY/metern)



## 10 TEST METERN

E' possibile eseguire alcuni test tramite l'interfaccia web di amministrazione per assicurarci che tutto sia stato configurato correttamente.

Per ogni misuratore eseguiremo il test di lettura dei valore istantaneo e medio utilizzando gli appositi bottoni che si trovano nelle rispettive pagine di configurazione del misuratore.

Collegiamoci con il browser alla pagina web di amministrazione, accessibile al seguente indirizzo:

`http://IP_RASPBERRY/meterN/admin/`

logghiamoci con le credenziali che abbiamo inserito precedentemente (admin e la password inserita in precedenza)

e clicchiamo su "Configure your meter(s)/sensor(s)"

### 10.1 TEST Misuratore 1 - Produzione

Sul primo misuratore ( 1 - Produzione) clicchiamo sul pulsante "Test command"

**meterN Administration**

Select a meter/sensor : 1 (Produzione)

**Meter#1 Produzione**

**Main 5min pooling :**

Name:  Type:  House production:  Skip monitoring:

Meter ID:  Command:  **Test command** Unit Wh Precision 0

Pass over:  Wh Color:  Price per unit:  €/Wh

**Dashboard live pooling :**

Meter ID:  Value:  Live command:  **Test live command** Live unit:

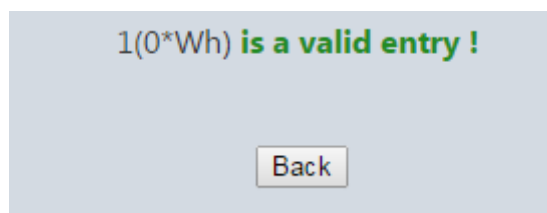
**Notification and report :**

Email:  **Test mail** Report consumption by mail:

Enable Pushover:  **Test Pushover** User key:

Warn if consumption is over:  Wh during the day

Rispondiamo OK alla richiesta di stoppare momentaneamente MeterN e verifichiamo che vi venga restituito:



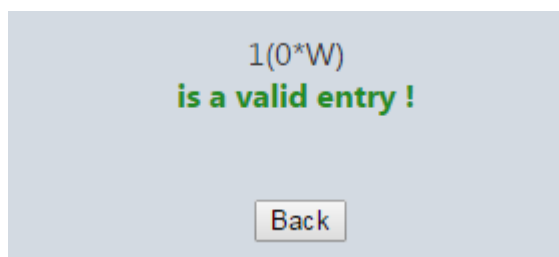
(se avete collegato l'inverter e sta producendo, al posto dello zero dovrete vedere l'energia prodotta)

dal vostro impianto fotovoltaico)

In caso vi venga restituito un errore dovete ricontrollare la configurazione del misuratore e del file pool123s.php al punto 8.1

Eseguiamo ora un secondo test cliccando sul pulsante "Test live command"

Anche in questo caso rispondiamo OK alla richiesta di stoppare momentaneamente MeterN e verificiamo che vi venga restituito:



(se avete collegato l'inverter e sta producendo, al posto dello zero dovreste vedere la potenza erogata dal vostro impianto fotovoltaico)

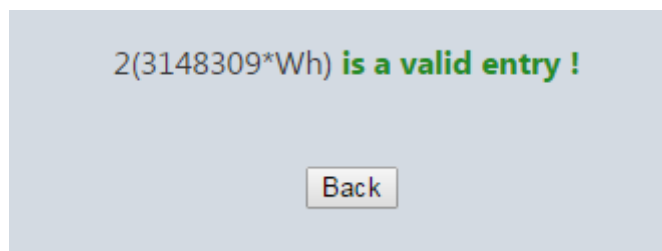
Anche in questo caso se vi viene restituito un errore dovete ricontrollare la configurazione del misuratore e del file pool123s.php al punto 8.1



## 10.2 TEST Misuratore 2 - Consumi

Selezioniamo ora nel menù a tendina in alto a sinistra il secondo misuratore ( 2 - Consumi) e clicchiamo sul pulsante "Test command"

Rispondiamo OK alla richiesta di stoppare momentaneamente MeterN e verifichiamo che vi venga restituito:



In caso vi venga restituito un errore dovete ricontrollare la configurazione del misuratore, del file pooler485.sh al punto 7.1 e 7.2, e del software di lettura del contatore del Capitolo 5.

Eseguiamo ora un secondo test cliccando sul pulsante "Test live command"

Select a meter/sensor : 2 (Consumi) ▼

### Meter#2 Consumi

**Main 5min pooling :**

Name Consumi Type Elect ▼ House consumption ▼ phase 1 Skip monitoring No ▼

Meter ID 2 Command poolerconsumi 2 energy Test command Unit Wh Precision 0

Pass over 0 Wh Color AA4643 Price per unit 0 €/Wh

**Dashboard live pooling :**

Meter ID 2 Value ▼ mode Live command poolerconsumi 2 power Test live command Live unit W

**Notification and report :**

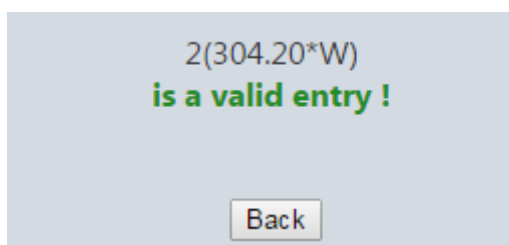
Email vostra email Test mail Report consumption by mail Never ▼

Enable Pushover No ▼ Test Pushover User key

Warn if consumption is over 0 Wh during the day Warn connection lost No ▼

Back Save config.

Rispondiamo OK alla richiesta di stoppare momentaneamente MeterN e verifichiamo che vi venga restituito:



Anche in questo caso se vi viene restituito un errore dovete ricontrollare la configurazione del misuratore, del file pooler485.sh al punto 7.1 e 7.2, e del software di lettura del contatore del Capitolo 5.

### 10.3 TEST Indicators

E' possibile eseguire allo stesso modo dei test anche per i vari indicatori che abbiamo inserito in MeterN.

Collegiamoci con il browser alla pagina web di amministrazione, accessibile al seguente indirizzo:

`http://IP_RASPBERRY/metern/admin/`

logghiamoci con le credenziali che abbiamo inserito precedentemente (admin e la password inserita in precedenza)

e clicchiamo su "Configure indicator(s) (No logged)"

Per ogni indicatore possiamo effettuare la verifica cliccando sul rispettivo pulsante "Test command"

## 11 Licenza d'uso



Quest'opera è distribuita con Licenza [Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 3.0 Italia](http://creativecommons.org/licenses/by-nc-sa/3.0/it/).

Per leggere una copia della licenza visita il sito web:

<http://creativecommons.org/licenses/by-nc-sa/3.0/it/>

o spedisce una lettera a Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

SE IL MIO LAVORO VI E' STATO UTILE, OFFRITEMI DA BERE,  
FATE UNA DONAZIONE :

**Donazione**



## APPENDICI



## APPENDICE A Guida all'uso dello script sdm120c

Cerchiamo ora di capire come funziona questo script e quali sono le varie opzioni disponibili.

Digitando da terminale semplicemente *sdm120c* senza nessuna opzione, il software ci restituisce alcune righe di spiegazione delle varie opzioni disponibili:

```
Usage: sdm120c [-a address] [-d] [-x] [-p] [-v] [-c] [-e] [-i] [-t] [-f] [-g] [-T] [[-m]|[-q]] [-b baud_rate] [-P parity]
[-S bit] [-z num_retries] [-j seconds] [-w seconds] [-1 | -2] device

sdm120c [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w
seconds] -s new_address device

sdm120c [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w
seconds] -r baud_rate device

sdm120c [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w
seconds] -R new_time device

where
-a address      Meter number (between 1 and 247). Default: 1
-s new_address  Set new meter number (between 1 and 247)
-p             Get power (W)
-v            Get voltage (V)
-c            Get current (A)
-f            Get frequency (Hz)
-g            Get power factor
-e            Get exported energy (Wh)
-i            Get imported energy (Wh)
-t            Get total energy (Wh)
-T            Get Time for rotating display values (0 = no rotation)
-d            Debug
-x            Trace (libmodbus debug on)
-b baud_rate    Use baud_rate serial port speed (1200, 2400, 4800, 9600)
                Default: 2400
-P parity       Use parity (E, N, O)
-S bit          Use stop bits (1, 2). Default: 1
-r baud_rate    Set baud_rate meter speed (1200, 2400, 4800, 9600)
-R new_time     Change rotation time for displaying values (0 - 30s) (0 = no rotation)
-m             Output values in IEC 62056 format ID(VALUE*UNIT)
-q             Output values in compact mode
-z num_retries  Try to read max num_retries times on bus before exiting
                with error. Default: 1
-j seconds      Response timeout. Default: 2
-w seconds      Time to wait to lock serial port. (1-30) Default: 0
-1             Model: SDM120C (default)
-2             Model: SDM220
device          Serial device, i.e. /dev/ttyUSB0

Serial device is required. When no parameter is passed, retrieves all values
```

Come visibile il software permette la lettura di tutte le grandezze misurate dal contatore ed anche la possibilità di impostare i diversi parametri del contatore stesso, come l'indirizzo, la velocità.

Di default, se non specificato diversamente nella riga di comando, il software assumerà:

- indirizzo (a): 1
- velocità (b): 2400 baud
- Stop bits (S): 1
- Numero di tentativi (z): 1
- Tempo di risposta (j): 2 s

### Opzione -w

Questo parametro, pur essendo l'ultimo arrivato, è molto importante e merita di essere chiarito. E' infatti indispensabile utilizzarlo nel caso in cui si utilizzano più contatori sullo stesso bus e letti da

diversi software (come nel caso in cui si utilizzi un contatore per la produzione con 123solar ed uno per i consumi con MeterN). In tale caso infatti, potrebbero verificarsi delle collisioni fra le due richieste di lettura sullo stesso bus, e questo provocherebbe il blocco delle letture stesse.

L'utilizzo del parametro -w evita questo problema, mettendo in coda la seconda richiesta fino a che la prima non è stata completata e per il tempo in secondi indicato. In tale caso vi suggerisco quindi l'utilizzo di "- w 5"

## ESEMPI

Vediamo di seguito alcuni esempi di utilizzo del software, che si spiegano meglio di molte parole.

Per verificare la versione del software, da terminale digitare:

```
sdm120c | head -n 1 | awk '{print $2}'
```

Per cambiare l'indirizzo del dispositivo da 1 a 2, premere il pulsante frontale sul contatore per 3 secondi, fino a che compare la scritta - SET - sul display, quindi da terminale digitare (ipotizzando la parità del contatore = None):

```
sdm120c -a 1 -s 2 -P N /dev/ttyUSB0  
New address 2  
You have to restart the meter for apply changes
```

Riavviare il contatore staccando e riattaccando la fase in ingresso.

Per cambiare la velocità di trasmissione da 2400 a 9600 del contatore (dopo il parametro -a va messo l'indirizzo del vostro contatore) :

```
sdm120c -a 2 -r 9600 -P N /dev/ttyUSB0  
New baud_rate 2  
You have to restart the meter for apply changes
```

Riavviare il contatore staccando e riattaccando la fase in ingresso.

Quindi, per leggere la sola potenza istantanea del contatore con indirizzo 2 e velocità 9600

```
sdm120c -a 2 -b 9600 -p /dev/ttyUSB0
```

**NOTA:** Una volta modificato l'indirizzo o la velocità del contatore sarà sempre necessario specificare il relativo valore fra le opzioni, in quanto se omessi, il software assume i valori previsti di default e non vi verrà restituita nessuna lettura.

## APPENDICE B      DISPOSITIVI USB (assegnare un nome fisso)

Nel caso abbiate installato due adattatori USB-RS485, per evitare che via siano degli scambi fra le porte USB0 e USB1 degli adattatori, è possibile assegnare un nome fisso ed univoco ad ogni adattatore.

### Adattatori usb diversi

Verificare gli ID dei dispositivi con il comando lsusb:

```
lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
Bus 001 Device 005: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-
Serial (UART) IC
```

Ci sono 2 dispositivi RS485. Creiamo il file /etc/udev/rules.d/10-local.rules

```
sudo nano /etc/udev/rules.d/10-local.rules
```

e andiamo a rinominare i due dispositivi come "metern" e "123solar" inserendo nel file le seguenti righe:

```
ACTION=="add", ATTRS{idVendor}=="067b", ATTRS{idProduct}=="2303", SYMLINK+="metern"
ACTION=="add", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="123solar"
```

CTRL+O per salvare e CTRL+X per uscire

Riavviamo

```
sudo reboot
```

dopo il riavvio si può verificare che tutto funzioni correttamente:

```
ls -l /dev/metern /dev/123solar
lrwxrwxrwx 1 root root 7 Jan 1 1970 /dev/metern -> ttyUSB0
lrwxrwxrwx 1 root root 7 Jan 1 1970 /dev/123solar -> ttyUSB1
```

Naturalmente i vostri dispositivi avranno ID diversi, è pertanto necessario adattare il tutto al vostro caso specifico.

Ora è possibile usare /dev/metern e /dev/123solar nei rispettivi programmi invece di /dev/ttyUSB0 e /dev/ttyUSB1.

## Adattatori usb identici

Se i **dispositivi sono identici** (stesso Vendor id e Product id) allora si deve cercare un altro parametro che li differenzi.

Ad esempio in questo caso sono identici:

```
lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
Bus 001 Device 005: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
```

Eseguire i seguenti comandi e confrontare i risultati:

```
udevadm info -a -p $(udevadm info -q path -n /dev/ttyUSB0)
```

e

```
udevadm info -a -p $(udevadm info -q path -n /dev/ttyUSB1)
```

Nel mio caso uno dei parametri che li differenzia è:

KERNELS=="1-1.2"            per ttyUSB0 (usato per metern)  
 KERNELS=="1-1.3"            per ttyUSB1 (usato per 123solar)

Si dovrà quindi creare il file 10-local.rules:

```
sudo nano /etc/udev/rules.d/10-local.rules
```

e scriverci dentro (sono 2 righe di testo - non 4)

```
ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", KERNELS=="1-1.2",  

  SYMLINK+="metern"  

  ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", KERNELS=="1-1.3",  

  SYMLINK+="123solar"
```

CTRL+O per salvare e CTRL+X per uscire

Riavviamo

```
sudo reboot
```

dopo il riavvio si può verificare che tutto funzioni correttamente:

```
ls -l /dev/metern /dev/123solar
lrwxrwxrwx 1 root root 7 Jan 1 1970 /dev/metern -> ttyUSB0
lrwxrwxrwx 1 root root 7 Jan 1 1970 /dev/123solar -> ttyUSB1
```

Ora è possibile usare /dev/metern e /dev/123solar nei rispettivi programmi invece di /dev/ttyUSB0 e /dev/ttyUSB1.



## APPENDICE C     Aggiunta sensori vari

### Impostazioni di base

Per l'utilizzo di sensori o altro tramite le porte GPIO del Raspberry è indispensabile aver installato [Wiringpi](#) che è la libreria più completa per la gestione dell'interfaccia GPIO del Raspberry PI. E' distribuita tramite GIT e la via più facile per scaricare ed installare la libreria è quindi tramite git-core.

```
cd /home/pi
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
./build
cd ..
```

Ora potete usare la libreria per comandare le porte del raspberry PI  
Per testare la corretta installazione e vedere lo stato della GPIO

```
gpio -v
gpio readall
```

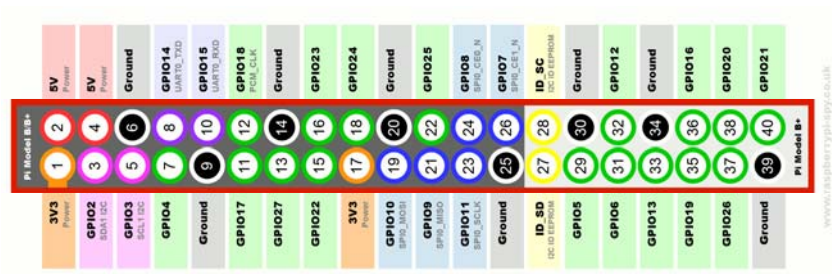
### Piedinatura e numerazione GPIO<sup>4</sup>

Di seguito due immagini che mostrano la piedinatura dell'header presente sul raspberry.  
Le immagini che seguono si riferiscono al modello B+. Di recente è uscito anche il Raspberry 2 ma la piedinatura rimane sempre la stessa.



Numerazione fisica dell'header, serve per identificare un pin sull'header

<sup>4</sup> FONTE: [Raspberry Pi : Usare le linee GPIO](#)



Piedinatura completa, elenca numero fisico (nel cerchio) funzione e nome del pin

Qui in basso troviamo inoltre indicata anche la numerazione secondo lo standard **WiringPi**, utilizzata abbastanza spesso.

GPIO	WiringPi	left bottom P1-01	top P1-02	WiringPi	GPIO
-	-	3V3 Power	5V Power	-	-
-	8	GPIO 0 (SDA)	5V Power	-	-
-	9	GPIO 1 (SCL)	Ground	-	-
4	7	GPIO 4 (GCLK0)	GPIO 14 (TXD)	15	14
-	-	Ground	GPIO 15 (RXD)	16	15
17	0	GPIO 17	GPIO 18 (PCM_CLK)	1	18
-	2	GPIO 21 (PCM_DOUT)	Ground	-	-
22	3	GPIO 22	GPIO 23	4	23
-	-	3V3 Power	GPIO 24	5	24
10	12	GPIO 10 (MOSI)	Ground	-	-
9	13	GPIO 9 (MISO)	GPIO 25	6	25
11	14	GPIO 11 (SCKL)	GPIO 8 (CE0)	10	8
-	-	Ground	GPIO 7 (CE1)	11	7
GPIO	WiringPi	P1-25 bottom right	P1-26 top right	WiringPi	GPIO

## Indicatore di processo in esecuzione per meterN

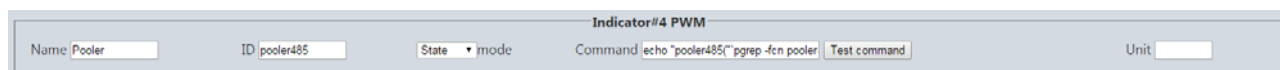
Per creare in MeterN un indicatore di stato che visualizza se un processo è in esecuzione oppure no, è sufficiente utilizzare il seguente comando nella configurazione:

```
echo "YYYY("`pgrep -c YYYY`"*X)"
```

dove YYYY è il nome del processo da monitorare.

Ad esempio, per visualizzare nel cruscotto di meterN se pooler485 è in esecuzione, è possibile utilizzare il comando:

```
echo "pooler485("`pgrep -c pooler485`"*X)"
```

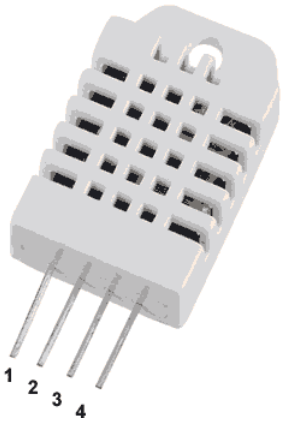



## Sensore di temperatura ed umidità DHT22

Il sensore DHT22 è di dimensioni molto ridotte e opera attraverso una tensione di alimentazione tra 3.3 e 6Vcc comunicando entrambi i dati attraverso un solo pin. E' in grado di misurare temperature che vanno da -40 a 80°C, con una precisione di  $\pm 0.5^{\circ}\text{C}$ , e di rilevare il livello di umidità relativa compresa tra 0 e 100%, con una precisione di  $\pm 2\%$ . Il sensore fornisce uscite calibrate in maniera completamente digitale per le due misure. Non è compatibile col protocollo 1-Wire® pur avendo un'unica uscita.

Infatti se si vogliono collegare più sensori ad un dispositivo bisogna predisporre una connessione per ogni sensore (una diversa linea GPIO del Raspberry per ogni sensore).

Rispetto al modello DHT11, questo sensore è più preciso, più accurato e copre un range più esteso di temperatura e umidità.

Versione base	Versione su scheda										
<div data-bbox="215 302 419 481"> <table border="1"> <thead> <tr> <th colspan="2">DHT22 pins</th> </tr> </thead> <tbody> <tr> <td>1</td><td>VCC</td></tr> <tr> <td>2</td><td>DATA</td></tr> <tr> <td>3</td><td>NC</td></tr> <tr> <td>4</td><td>GND</td></tr> </tbody> </table> </div> 	DHT22 pins		1	VCC	2	DATA	3	NC	4	GND	
DHT22 pins											
1	VCC										
2	DATA										
3	NC										
4	GND										
Necessita della resistenza esterna da 10k (vedi schema sotto)	Non necessità della resistenza di pull-up da 10K in quanto già presente sulla scheda										

**Specifiche tecniche:**

- Tensione operativa: 3.3 ~ 6Vcc
- Range di misura temperatura: -40~80 °C  $\pm$  0.5°C
- Range di misura umidità: 0-100% (relativa)  $\pm$ 2%
- Interfaccia: digitale
- Risoluzione: 1°C, 8bit
- Tempo di risposta: 2s
- Strip connessione: 2.54mm
- Dimensioni: 15 x 25 x 7.7 mm, foro 3mm

Datasheet: [Datasheet sensore DHT22](#) [Altro datasheet](#)

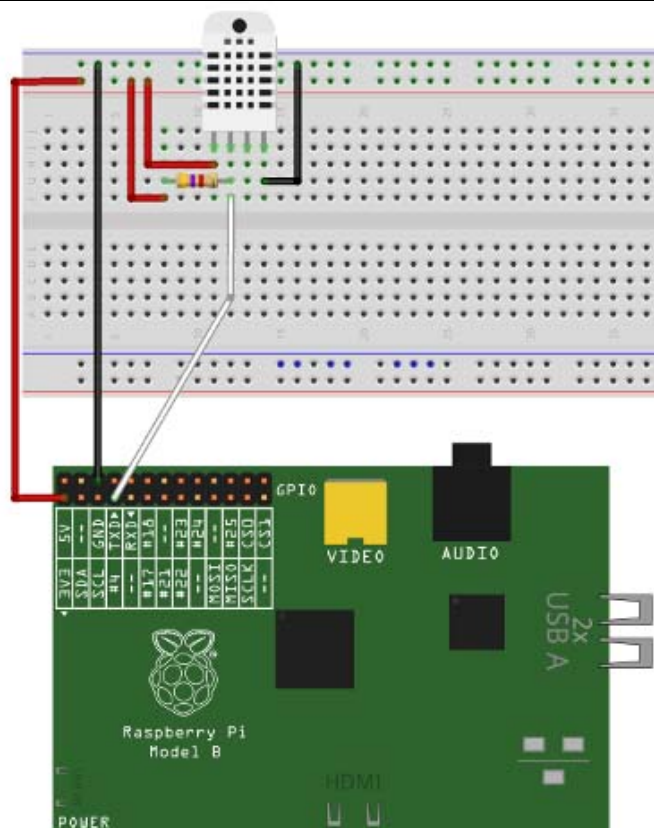
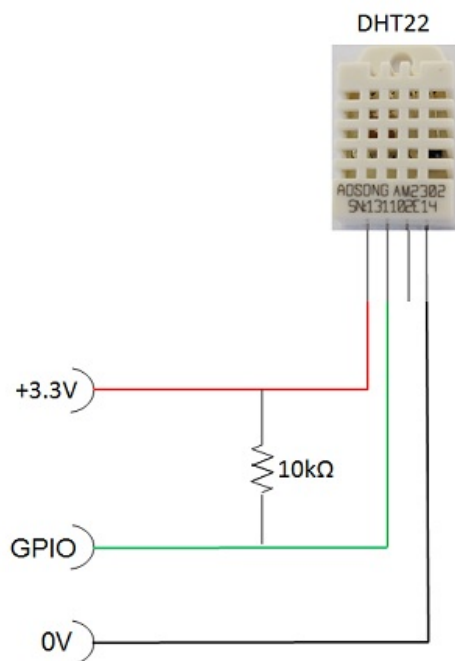
In questo appendice vedremo come rilevare misure di temperatura e umidità con il sensore DHT22 ed il nostro Raspberry Pi con MeterN.

**MATERIALE:**

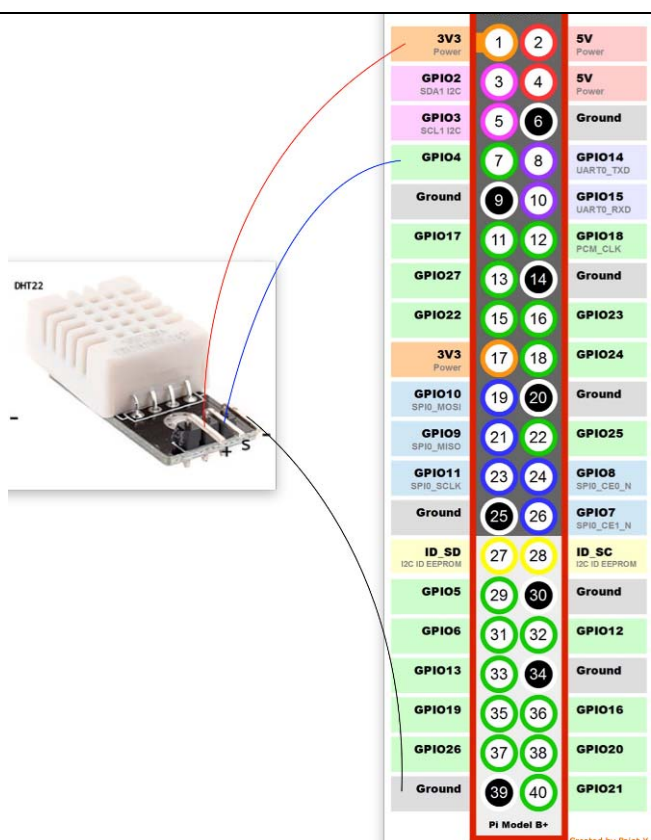
- 1x Sensore DHT22
- 1x Resistenza da 10 k $\Omega$  (non necessaria per la versione su scheda)
- Cavetti vari

L'utilizzo di questo sensore è molto semplice grazie alla libreria per Raspberry Adafruit\_Python\_DHT. Il circuito da realizzare è il seguente:

## Versione base



**Versione con scheda**  
(resistenza già presente sulla scheda)



Installiamo il software necessario per la lettura dei sensori:

```
/home/pi
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT/
sudo apt-get install build-essential python-dev
python setup.py build
sudo python setup.py install
```

Creiamo il file ADHT\_metern.py:

```
nano /usr/local/bin/ADHT_metern.py
```

**ATTENZIONE:** essendo un file python è importante mantenere le spaziature e l'indentatura

```
#!/usr/bin/python
# Copyright (c) 2014 Adafruit Industries
# Author: Tony DiCola

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
import sys

import Adafruit_DHT

# Parse command line parameters.
sensor_args = { '11': Adafruit_DHT.DHT11,
                '22': Adafruit_DHT.DHT22,
                '2302': Adafruit_DHT.AM2302 }
if len(sys.argv) == 4 and sys.argv[1] in sensor_args:
    sensor = sensor_args[sys.argv[1]]
    pin = sys.argv[2]
    id = sys.argv[3]
else:
    print 'usage: sudo ./Adafruit_DHT.py [11|22|2302] GPIOpin# metern_id'
```



```
print 'example: sudo ./Adafruit_DHT.py 2302 4 8 - Read from an AM2302 connected to GPIO #4 and metern id
8'

sys.exit(1)

# Try to grab a sensor reading. Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!
if humidity is not None and temperature is not None:
    print '{0}_1({1:0.1f}*C)'.format(id, temperature)
    print '{0}_2({1:0.1f}*)'.format(id, humidity)
else:
    print 'ERROR'
```

diamo i permessi in esecuzione al file:

```
sudo chmod +x /usr/local/bin/ADHT_metern.py
```

La modalità di utilizzo dello script è:

```
ADHT_metern.py "tipo sensore 21 o 22" "porta GPIO" "meterID"
```

Testiamo il sensore, supponendo che come da schema precedente, si tratti di un sensore DHT22, collegato alla GPIO4 e che l'id metern da assegnare sia 6

```
sudo ADHT_metern.py 22 4 6
6_1(24.2*C)
6_2(74.7*%)
```

**NOTA:** in questo caso la numerazione delle porte GPIO è quella standard

Per evitare possibili errori, non utilizziamo direttamente il file python appena creato, ma creiamo uno script bash come il seguente, che invoca il file python con all'interno un controllo per evitare dati errati ed infine andremo con cron ad invocare il seguente script:

```
nano /usr/local/bin/dht22.sh
```

```
#!/bin/sh
DATA="$(ADHT_metern.py $1 $2 $3)"
#echo $DATA

if [ x"$DATA" != x ] && [ "$DATA" != "ERROR" ]; then
    echo $DATA | cut -f 1 -d\ > /run/shm/metern$3.txt
    echo $DATA | cut -f 2 -d\ >> /run/shm/metern$3.txt
fi
```

Crtl+O per salvare e CRTL+X per uscire

Impostare i permessi in esecuzione:

```
sudo chmod +x /usr/local/bin/dht22.sh
```

Impostare cron per eseguire periodicamente (ogni 15 minuti) la lettura dei valori, creando il seguente file e se necessario modificando opportunamente le parti in rosso.

```
nano /etc/cron.d/dht22
```

```
#!/bin/bash

SHELL=/bin/bash
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""

# Minute Hour Day of Month Month Day of Week User Command
# (0-59) (0-23) (1-31) (1-12 or Jan-Dec) (0-6 or Sun-Sat)
2,17,32,47 * * * * root /usr/local/bin/dht22.sh 22 4 6 > /dev/null 2>&1
```

Crtl+O per salvare e CRTL+X per uscire

Per fare in modo che lo script venga invocato anche all'avvio del Raspberry è necessario inserire la seguente riga anche in rc.local

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
nano /etc/rc.local
```

Editare il file /etc/rc.local inserendo le modifiche in rosso:

```
stty -F /dev/ttyUSB0 19200 &
sudo /usr/bin/curl http://localhost/123solar/scripts/boot123s.php &
sudo sleep 6
sudo /usr/local/bin/dht22.sh 22 4 6
sudo /usr/bin/curl http://localhost/metern/scripts/bootmn.php &
exit 0
```

Premere ctrl+O per salvare e ctrl+X per uscire



Infine, su meterN si configurano 2 sensori (temperatura e umidità) con i seguenti comandi:

#### Per la temperatura

**Meter#6 Temperatura**

**Main 5min pooling :**

Name  Type  Skip monitoring

Meter ID  Command   Unit  Precision

Pass over  °C Color  Price per unit  €/°C

---

**Dashboard live pooling :**

Meter ID  Value  Live command   Live unit

---

**Notification and report :**

Email   Report by mail

Enable Pushover ☐ No  User key

Warn if is over  °C during the day

#### Main pooling:

Command: `cat /run/shm/metern6.txt | grep C`

#### Dashboard live pooling

Live command: `cat /run/shm/metern6.txt | grep C`

#### Per l'umidità

**Meter#7 Umidità**

**Main 5min pooling :**

Name  Type  Skip monitoring

Meter ID  Command   Unit  Precision

Pass over  % Color  Price per unit  €/%

---

**Dashboard live pooling :**

Meter ID  Value  Live command   Live unit

---

**Notification and report :**

Email   Report by mail

Enable Pushover ☐ No  User key

Warn if is over  % during the day

#### Main pooling:

Command: `cat /run/shm/metern6.txt | grep %`

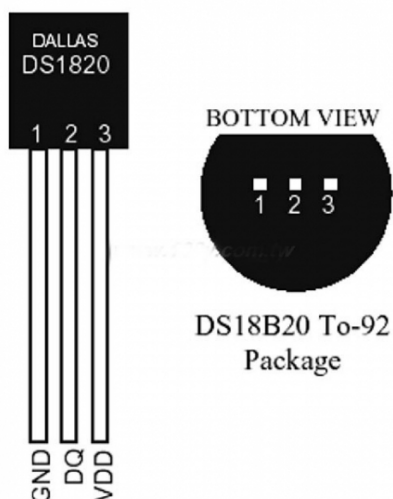
#### Dashboard live pooling

Live command: `cat /run/shm/metern6.txt | grep %`

## Sensori di temperatura DS18B20

### FONTI:

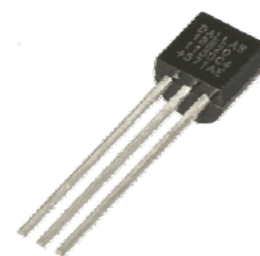
- [Come rilevare la temperatura con Raspberry Pi ed il sensore DS18B20](#)
- [DS18B20 - Sensore di Temperatura Digitale 1-Wire](#)
- [1Wire e il sensore di temperatura ds18s10](#)



Il sensore di temperatura DS18B20 è l'ultimo sensore di temperatura 1-Wire della Maxim IC. Questo sensore ha la capacità di rilevare temperature con precisione da 9-bit a 12-bit in un range di temperature fra un minimo di  $-55^{\circ}\text{C}$  ed un massimo di  $+125^{\circ}\text{C}$  con un'approssimazione di  $\pm 0.5^{\circ}\text{C}$ .

La particolarità di questi sensori di temperatura è rappresentata dal fatto che ciascun sensore ha numero seriale unico di 64-bit che lo identifica memorizzato all'interno di una ROM presente al loro interno. Questa caratteristica permette di poter utilizzare un vasto numero di sensori su di un unico bus di dati; caratteristica di fondamentale importanza nella maggior parte di progetti di data-logging e di progetti basati sul controllo della temperatura.

Il protocollo utilizzato da questi sensori è lo **Unique 1-Wire®**; questa interfaccia richiede l'utilizzo di un unico pin per la comunicazione. Inoltre, per poter funzionare nelle applicazioni tipiche questi sensori di temperatura non necessitano di alcun altro componente esterno. Possono essere alimentati direttamente dalla linea dati ed il range di alimentazione supportato è da 3.0V a 5.5V.

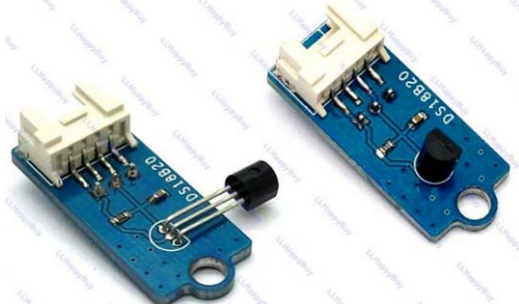




### CARATTERISTICHE TECNICHE

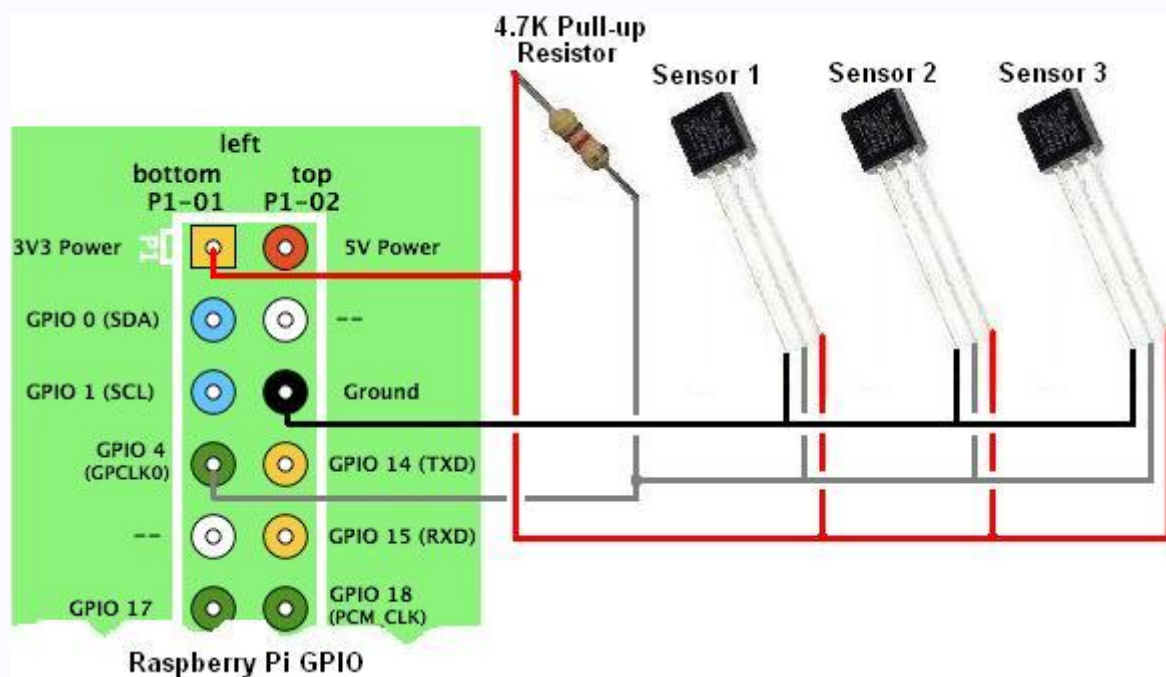
- Alimentazione: da 3.0V a 5.5V
- Calibrato direttamente in  $^{\circ}\text{Celsius}$  (Centigradi)
- Range di temperature misurabili: da  $-55^{\circ}\text{C}$  a  $+125^{\circ}\text{C}$  (da  $-67^{\circ}\text{F}$  a  $257^{\circ}\text{F}$ )
- Accuratezza:  $\pm 0.5^{\circ}\text{C}$  (nel range  $-10^{\circ}\text{C}$  /  $85^{\circ}\text{C}$ )
- Risoluzione del sensore: da 9-bit a 12-bit (impostabile dall'utente)
- Tempo di conversione di temperature in 12-bit word: 750ms
- Ogni sensore ha un indirizzo univoco a 64-bit memorizzato in una ROM interna

Si tratta di un sensore di temperatura tra i più utilizzati sia nel campo hobbistico che in quello professionale, in quanto ha una sensibilità ed una accuratezza della rilevazione di tutto rispetto. Risulta quindi ottimo per ogni tipo di test a livello hobbistico e per la maggior parte delle applicazioni in campo domotico.

E' possibile trovare in commercio questo sensore sia come integrato semplice sia già accoppiato ad altri componenti oppure in versione stagna da pozzetto ad immersione (per boiler ad esempio).

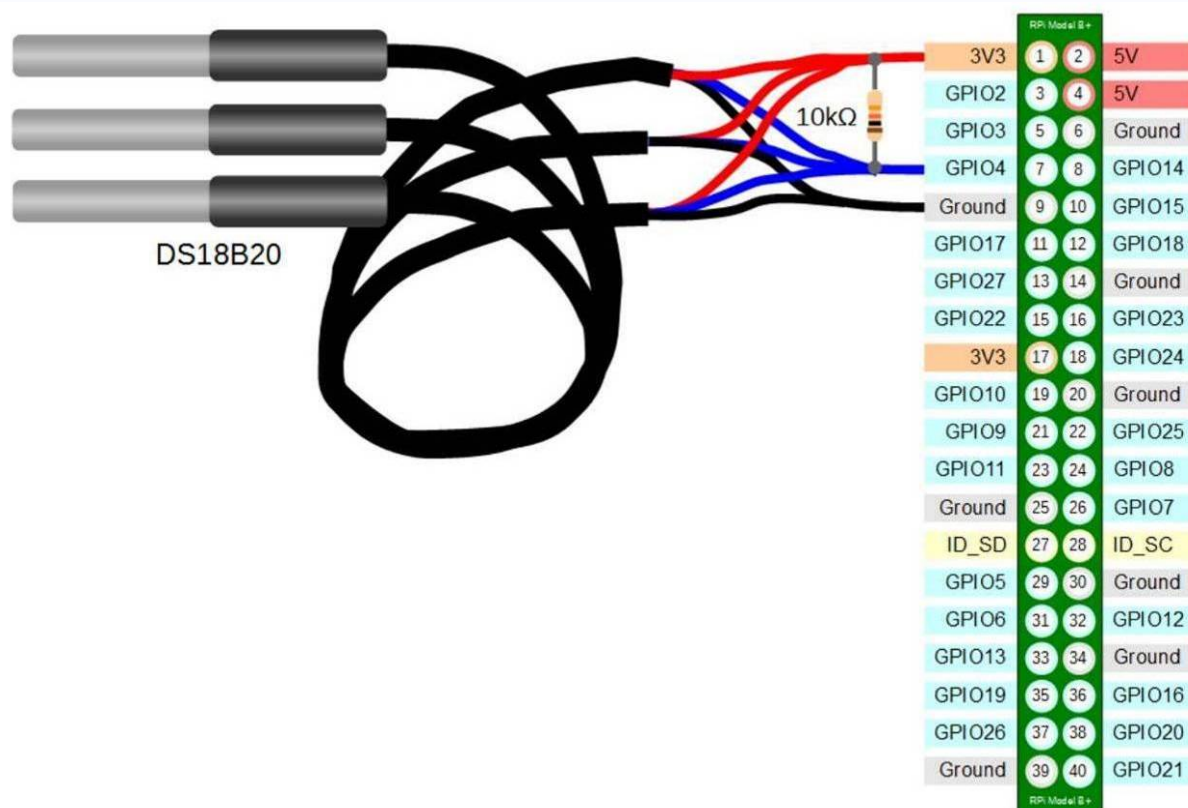
	<p>Questi sensori montati su schedina risultano già dotati della resistenza di pull-up necessaria per il collegamento al Raspberry</p>									
										
	<p>Nel caso invece dei sensori stagni con cappuccio in acciaio occorre collegare esternamente la resistenza di pull-up</p> <p>Per individuare i conduttori di questa tipologia, la regola è:</p> <table><tr><td>Rosso</td><td>----</td><td>VCC</td></tr><tr><td>Nero</td><td>----</td><td>GND</td></tr><tr><td>Altro colore</td><td>----</td><td>DATA</td></tr></table>	Rosso	----	VCC	Nero	----	GND	Altro colore	----	DATA
Rosso	----	VCC								
Nero	----	GND								
Altro colore	----	DATA								

I collegamenti da effettuare tra il sensore DS18B20 ed il Raspberry Pi sono i seguenti:

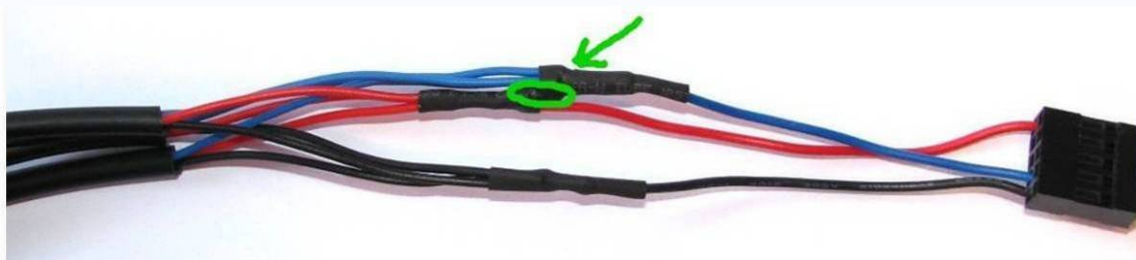


Se la resistenza da 4,7K non dovesse funzionare, provate con una da 10K.

Nel caso dei sensori stagni con cappuccio, una soluzione pratica per effettuare il collegamento e per collegare la resistenza di pull-up è la seguente:



Nella pratica il collegamento può essere realizzato saldando i vari fili dei sensori da collegare ed isolando opportunamente le saldature con del termo-restingente, e collegare la resistenza di pull-up fra il cavo data e +3,3V come di seguito illustrato (la resistenza è nel cerchietto verde).



Il **pin 7 (GPIO4)** è quello utilizzato di default sul Raspberry per il protocollo 1Wire, ma se avete necessità e comunque possibile modificarlo.

Per cambiarlo ed impostare ad esempio il pin 18 (GPIO24) ci sono 2 modi diversi:

1. Per Raspberry B e B+ (A e A+) (o kernel più vecchio del 3.18.0):

```
sudo -s  
echo "options w1_gpio gpiopin=24" > /etc/modprobe.d/w1
```

## 2. Per Raspberry 2:

E' necessario editare il file /boot/config.txt e modificare la seguente riga:

```
# device tree config  
dtoverlay=w1-gpio,gpiopin=24
```

In entrambi i casi sarà necessario riavviare il Raspberry per attivare le modifiche.

## INSTALLAZIONE E CONFIGURAZIONE SENSORE

Innanzitutto è necessario editare il file modules e config.txt

Da terminale è necessario modificare il file /etc/modules,

```
nano /etc/modules
```

aggiungendo le righe

```
w1_gpio  
w1_therm
```

Da terminale

```
nano /boot/config.txt
```

ed inserire il fondo la stringa

```
dtoverlay=w1-gpio,gpiopin=4
```

Premere ctrl+O per salvare e ctrl+X per uscire

Riavviamo il raspberry

```
sudo reboot
```

Installiamo ora il software per la lettura dei sensori:

```
cd /home/pi  
git clone https://github.com/timofurrer/w1thermsensor.git  
cd w1thermsensor/  
python setup.py build  
sudo python setup.py install
```

ricaviamo ora il seriale dei sensori appena collegati:

```
cd /sys/devices/w1_bus_master1/  
ls -l 28-*  
28-000005b8a78e
```

Annotatevi il numero che vi restituirà (parte in rosso) in quanto è il numero seriale del vostro sensore e servirà in seguito per la lettura dei dati.

Creare il file /usr/local/bin/ds18b20.py

```
nano /usr/local/bin/ds18b20.py
```

ed incolliamo il seguente contenuto:

```
#!/usr/bin/env python  
import sys  
from w1thermsensor import W1ThermSensor  
  
address = sys.argv[1]  
sensor = W1ThermSensor(W1ThermSensor.THERM_SENSOR_DS18B20, address)  
temperature_in_celsius = sensor.get_temperature()  
  
print("{0:2f}".format(temperature_in_celsius))
```

Premere ctrl+O per salvare e ctrl+X per uscire

Diamo i permessi in esecuzione al file:

```
sudo chmod +x /usr/local/bin/ds18b20.py
```

Testiamo ora se il sensore risponde correttamente (sostituire la parte in rosso con il vostro seriale):

```
ds18b20.py 000005b8a78e  
26.38
```

Infine configuriamo meterN con questo comando

```
echo "6($(ds18b20.py 000005b8a78e)*C)"
```



Per evitare possibili errori, è meglio però non utilizzare direttamente il file python appena creato, ma creiamo uno script bash come il seguente, che invoca il file python dove potremmo eventualmente anche inserire all'interno un controllo per evitare dati errati, ed infine andremo poi con cron ad invocare il seguente script:

```
nano /usr/local/bin/ds18b20
```

```
#!/bin/sh
DATA="$(ds18b20.py $1)"
echo "$2($DATA*C)" > /run/shm/metern$2.txt
```

Crtl+O per salvare e CRTL+X per uscire

Impostare i permessi in esecuzione:

```
sudo chmod +x /usr/local/bin/ds18b20
```

Per utilizzare il file:

```
ds18b20 "seriale sensore" "meterID"
```

ad esempio:

```
ds18b20 000005b8a78e 6
```

che restituirà

```
6(26.38*C)
```

Impostare cron per eseguire periodicamente (ogni 15 minuti) la lettura dei valori, modificando opportunamente il seriale del vostro sensore.

```
pi@raspberrypi2 ~ $ nano /etc/cron.d/ds18b20
```

```
#!/bin/bash

SHELL=/bin/bash
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""
# Minute Hour Day of Month Month Day of Week User Command
# (0-59) (0-23) (1-31) (1-12 or Jan-Dec) (0-6 or Sun-Sat)
2,17,32,47 * * * * root /usr/local/bin/ds18b20 000005b8a78e 6 > /dev/null 2>&1
```

Crtl+O per salvare e CRTL+X per uscire

Per fare in modo che lo script venga invocato anche all'avvio del Raspberry (q quindi creato subito all'avvio del Raspberry il file temporaneo con la lettura del sensore) è necessario inserire la seguente riga anche in rc.local

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
nano /etc/rc.local
```

Editare il file /etc/rc.local inserendo le modifiche in rosso:

```
stty -F /dev/ttyUSB0 19200 &  
sudo /usr/bin/curl http://localhost/123solar/scripts/boot123s.php &  
sudo sleep 6  
sudo /usr/local/bin/ds18b20 000005b8a78e 6  
sudo /usr/bin/curl http://localhost/metern/scripts/bootmn.php &  
exit 0
```

Premere ctrl+O per salvare e ctrl+X per uscire

Infine, su meterN si configura il sensore come di seguito:

<b>Main 5min pooling :</b>		
Name <input type="text" value="Temp_sonda"/>	Type <input type="text" value="Other"/>	Skip monitoring <input type="text" value="No"/>
Meter ID <input type="text" value="6"/>	Command <input type="text" value="cat /run/shm/metern6.txt   grep C"/> <input type="button" value="Test command"/>	Unit <input type="text" value="C"/> Precision <input type="text" value="1"/>
Pass over <input type="text" value="0"/> C	Color <input type="text" value="FFFFFF"/>	Price per unit <input type="text" value="0"/> €/C
<b>Dashboard live pooling :</b>		
Meter ID <input type="text" value="6"/>	Value <input type="text" value="mode"/> Live command <input type="text" value="cat /run/shm/metern6.txt   grep C"/> <input type="button" value="Test live command"/>	Live unit <input type="text" value=""/>
<b>Notification and report :</b>		
Email <input type="text" value="no@be.org"/> <input type="button" value="Test mail"/>	Report by mail <input type="text" value="Never"/>	
Enable Pushover <input type="text" value="No"/> <input type="button" value="Test Pushover"/>	User key <input type="text" value=""/>	
Warn if is over <input type="text" value="0"/> C during the day		

### Main pooling:

Command: `cat /run/shm/metern6.txt | grep C`

### Dashboard live pooling

Live command: `cat /run/shm/metern6.txt | grep C`



## Sensori di pressione/temperatura/altitudine BMP085 o BMP180

### FONTI:

[Adafruit - Using the BMP085/180 with Raspberry Pi](#)  
[EOS - Pressione e Temperatura: BMP085 e Arduino](#)  
[BMP180 Barometric Pressure Sensor Hookup](#)

Questo sensore è digitale e utilizza il sistema di comunicazione I2C.

Il protocollo I2C utilizza due pin:

- pin SCL da il segnale di clock (Serial Clock Line);
- pin SDA trasmette i dati (Serial Data Line).

Con questo protocollo possono essere collegati fino a 112 dispositivi sulla stessa linea (con indirizzi diversi) e la velocità standard è di 100 kbit/s.

**NOTA:** In considerazione che i sensori BMP180 hanno tutti indirizzo 0x77, potremmo collegare un solo sensore. Se avete bisogno di più sensori BMP180 (quindi tutti allo stesso indirizzo), e se i dispositivi hanno un pin di reset (come ad esempio ha il BMP085), è possibile utilizzare più dispositivi allo stesso indirizzo, ma a scapito di utilizzare un pin GPIO per dispositivo. Quello che si dovrà fare è di mantenere per tutti i sensori il pin GPIO (collegato a XCLR - Reset ) a livello basso, e portarlo a livello alto solo per il dispositivo che si vuole leggere, svincolandolo quindi dal reset e inducendolo a rispondere a qualsiasi richiesta sul bus I2C.

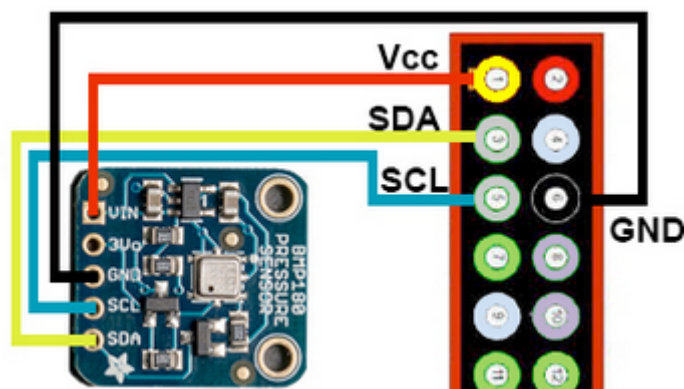
La calibrazione del sensore è fatta durante la produzione e i valori sono salvati nella memoria del sensore. Il BMP180 misura pressione e temperatura con step di 0,1 hPa e 0,1 °C.

I vantaggi offerti dall'utilizzo del sensore sono:

- Range elevato (da 300 a 1100 hPa come da 9000 a -500 m s.l.m);
- Basso errore (0,03 hPa nella modalità ad alta precisione e 0,06 nella modalità a risparmio energetico, equivalenti a 0,25 e 0,50 m);
- Consumo ridotto (5 uA);
- Sensore di temperatura integrato;
- Comunicazione I2C;
- Già calibrato accuratamente in fabbrica.

Il BMP180 misura anche l'altitudine basando le sue misurazioni sulla variazione di pressione e temperatura.

## COLLEGAMENTO DEL SENSORE BMP180



## INSTALLAZIONE E CONFIGURAZIONE SOFTWARE

Dobbiamo ora installare ed attivare gli I2C drivers.

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

```
sudo raspi-config
```

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem      Ensures that all of the SD card storage is available to the OS
2 Change User Password   Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment, Scratch, or the command-line
4 Internationalisation Options Set up language and regional settings to match your location
5 Enable Camera          Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack        Add this Pi to the online Raspberry Pi Map (Rastrack)
7 Overclock              Configure overclocking for your Pi
8 Advanced Options       Configure advanced settings
9 About raspi-config     Information about this configuration tool

<Select>                                <Finish>
```

```
Raspberry Pi Software Configuration Tool (raspi-config)

A1 Overscan              You may need to configure overscan if black bars are present on display
A2 Hostname              Set the visible name for this Pi on a network
A3 Memory Split          Change the amount of memory made available to the GPU
A4 SSH                   Enable/Disable remote command line access to your Pi using SSH
A5 Device Tree           Enable/Disable the use of Device Tree
A6 SPI                   Enable/Disable automatic loading of SPI kernel module (needed for e.g. PiFace)
A7 I2C                   Enable/Disable automatic loading of I2C kernel module
A8 Serial                Enable/Disable shell and kernel messages on the serial connection
A9 Audio                 Force audio out through HDMI or 3.5mm jack
A0 Update                Update this tool to the latest version

<Select>                                <Back>
```

E rispondere YES alle successive richieste ed infine uscire da raspi-config selezionando "Finish"

Installazione il supporto del kernel

```
sudo nano /etc/modules
```

ed modificare come segue

```
snd-bcm2835  
i2c-bcm2708  
i2c-dev
```



```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

e commentare le righe (se ci fossero)

```
# blacklist spi-bcm2708  
# blacklist i2c-bcm2708
```

```

pi@raspberrypi: ~ -- ssh -- 116x32
GNU nano 2.2.6 File: /etc/modprobe.d/raspi_blacklist.conf
# blacklist spi and i2c by default (many users don't need them)
#blacklist spi-bcm2708
#blacklist i2c-bcm2708

[ Read 4 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell

```

```
sudo nano /boot/config.txt
```

ed aggiungere le righe (se non ci fossero)

```

dtparam=i2c1=on
dtparam=i2c_arm=on

```

Premere ctrl+O per salvare e ctrl+X per uscire

Riavviamo ora il raspberry

```
sudo reboot
```

Dopo il riavvio, eseguiamo un test che il bus I2C sia correttamente configurato, digitando da terminale:

```
sudo i2cdetect -y 1
```

considerato che il sensore BMP180 ha come indirizzo 0x77, dovrebbe restituire

```

pi@PiHutTutorials ~$ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  -- 77
pi@PiHutTutorials ~$

```

Segnalando così l'effettivo rilevamento del sensore collegato al raspberry

Completiamo quindi l'installazione:

```

sudo -s
cd /home/pi
apt-get update
apt-get install git build-essential python-dev python-smbus
mkdir /BMP180Code
cd BMP180Code
git clone https://github.com/adafruit/Adafruit_Python_BMP.git
cd Adafruit_Python_BMP
python setup.py install

```

Creiamo il file python pa.py:

```

cd /var/www/MyScripts
nano pa.py

```

```

#!/usr/bin/python
ID = sys.argv[1]
import Adafruit_BMP.BMP085 as BMP085 # Imports the BMP library

# Create an 'object' containing the BMP180 data
sensor = BMP085.BMP085()
print '7_1{0:0.2f}*C'.format(sensor.read_temperature()) # Temperature in Celcius
print '7_2{0:0.2f}*Pa'.format(sensor.read_pressure()*1/100.00) # The local pressure
print '7_3{0:0.2f}*m'.format(sensor.read_altitude()) # The current altitude
print '7_4{0:0.2f}*Pm'.format(sensor.read_sealevel_pressure()*1/100.00) # The sea-level pressure

```

Premere ctrl+O per salvare e ctrl+X per uscire

Testiamo ora il file appena creato per verificare che risponda correttamente con tutte le letture del sensore:

```
sudo python /var/www/MyScripts/pa.py
```

che dovrebbe restituire qualcosa come:

```
7_1(21.00*C)
7_2(900.56*Pa)
7_3(983.70*m)
7_4(900.53*Pm)
```

Impostiamo ora cron per eseguire periodicamente (ogni 15 min) la lettura dei valori

```
nano /etc/cron.d/sensori
```

ed incolliamo i seguenti comandi:

```
#!/bin/bash

SHELL=/bin/bash
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""

# Minute Hour Day of Month Month Day of Week User Command
# (0-59) (0-23) (1-31) (1-12 or Jan-Dec) (0-6 or Sun-Sat)
2,17,32,47 * * * * root sudo python /var/www/MyScripts/pa.py > /run/shm/metern7.txt
```

Premere ctrl+O per salvare e ctrl+X per uscire

Per fare in modo che lo script venga invocato una prima volta anche all'avvio del Raspberry è necessario inserire la seguente riga anche in rc.local

Utilizzando Putty, colleghiamoci al Raspberry ed eseguiamo quanto segue.

```
nano /etc/rc.local
```

Editare il file /etc/rc.local inserendo le modifiche in rosso:

```
stty -F /dev/ttyUSB0 19200 &
sudo /usr/bin/curl http://localhost/123solar/scripts/boot123s.php &
sudo sleep 6
sudo /usr/local/bin/ds18b20 000005b8a78e 6
sudo python /var/www/MyScripts/pa.py > /run/shm/metern7.txt
sudo /usr/bin/curl http://localhost/metern/scripts/bootmn.php &
exit 0
```

Premere ctrl+O per salvare e ctrl+X per uscire

Possiamo quindi inserire i nuovi meters in MeterN.

I sensori inclusi ed utilizzabili nel BMP180 ed il relativo comando da inserire in MeterN sono:

Sensore	Meter_ID	Command
Temperatura in celsius	7_1	cat /run/shm/metern7.txt   egrep "^7_1(\"   egrep "\*C\) \$"
Pressione locale	7_2	cat /run/shm/metern7.txt   egrep "^7_2(\"   egrep "\*Pa\) \$"
Altitudine	7_3	cat /run/shm/metern7.txt   egrep "^7_3(\"   egrep "\*m\) \$"
Pressione a livello del mare	7_4	cat /run/shm/metern7.txt   egrep "^7_4(\"   egrep "\*Pm\) \$"

Per la Pressione locale avremo ad esempio

**Main 5min pooling :**  
Name  Type  Skip monitoring   
Meter ID  Command  Unit  Precision   
Pass over  Test command  Price per unit   
Pa Color  €/Pa

**Dashboard live pooling :**  
Meter ID  Disable  Live command  Live unit   
Test live command

**Notification and report :**  
Email  Test mail   
Enable Pushover  Test Pushover   
Warn if is over  Pa during the day Report by mail   
User key  Warn connection lost

## APPENDICE D      **Procedura di aggiornamento di MeterN**

Vediamo di seguito la procedura da seguire in caso di aggiornamento del software MeterN.

1. Eseguire un backup di tutte le directory di MeterN. Utilizzando WinSCP fatevi una copia in locale sul vostro PC dell'intera cartella `/var/www/metern/`
2. Arrestare l'istanza precedente di MeterN, (fermare MeterN sul pannello di amministrazione mettendo in "OFF")

a. Dal browser:

[http://IP\\_raspberry/metern/admin](http://IP_raspberry/metern/admin)

loggatevi e spegnete metern portando ad OFF il relativo interruttore.

3. Rinominare la cartella esistente di metern in "*metern\_old*"

Potete procedere da terminale o operare con il comodo WinSCP

a. Da terminale:

```
cd /var/www
mv metern metern_old
```

4. Scaricare e decomprimere la nuova versione di MeterN

a. Da terminale:

```
cd /var/www
wget http://www.123solar.org/downloads/metern/metern0.X.X.tar.gz
tar -xzf metern0.X.X.tar.gz
rm -v metern0.X.X.tar.gz
```

5. Nel caso in cui non siano state introdotte particolari modifiche, è possibile utilizzare i file di configurazione salvati in precedenza.  
Copiare "`config_main.php`" e i vari "`config_metX.php`", "`config_layout.php`" e "`config_daemon.php`" dalla directory `/metern_old/config` nella nuova directory `/metern/config`.

Per sicurezza fate un confronto del contenuto dei nuovi file con i vostri file, per controllare che non siano stati aggiunti nuovi parametri. In quest'ultimo caso sarà necessario riconfigurare manualmente meterN

6. Andate in amministrazione MeterN, aprite ogni singola pagina di configurazione, controllate tutti i parametri e salvate (sempre!) la configurazione principale e la configurazione per ogni meters.
7. copiate il contenuto della cartella `/metern_old/data` nella nuova cartella `metern/data`  
Controllate in particolare i permessi dei vari file presenti in questa cartella e che il proprietario sia l'utente `www-data`



a. Da terminale:

```
cp -R /var/www/metern_old/data/ /var/www/metern/  
chown -R www-data:www-data /var/www/metern/data/
```

8. copiate nella cartella comapps tutte le app della vecchia cartella

a. Da terminale

```
cp -R /var/www/metern_old/comapps/ /var/www/metern/
```

9. Cancelliamo e ricreiamo il link simbolico al file pooler485.sh

a. Da terminale

```
rm /usr/local/bin/pooler485  
sudo ln -s /var/www/metern/comapps/pooler485.sh /usr/local/bin/pooler485
```

10. riavviare MeterN dal pannello di amministrazione e verificare che tutto sia correttamente funzionante

## APPENDICE E Backup dati MeterN tramite FTP

Il presente script deriva da una modifica dello script indicato da Walter62 nella sua guida per 123solar.

Lo script è stato infatti modificato per fare il backup della cartella data anche di MeterN oltre che di 123solar.

Il sistema esegue un procedura che salva 30 giorni e poi scarta il più vecchio con il principio "FIFO"

Installate, per prima cosa, il pacchetto lftp.

```
../$ sudo -s
../# cd /..
../# apt-get install lftp
```

Se non lo avete già fatto in precedenza create una directory dove posizionare gli script personali (MyScripts) . Saltate questo passaggio se avete già creato la cartella MyScripts.

```
../$ sudo -s
../# cd /var/www
../# mkdir MyScripts
../# chmod -v 777 MyScripts
```

Creiamo uno script di nome "ftpbackup"

```
../# cd MyScripts
../# nano ftpbackup.sh
```

ed incolliamo le seguenti righe

```
#!/bin/sh
# BEGIN INIT INFO
# Short-Description: backup giornaliero
# Description: Questo file è usato per eseguire una copia giornaliera
# dei dati del datalogger 123solar e di MeterN di 30 giorni con metodo fifo
#
# posizionare il file in /var/www/MyScripts
#### END INIT INFO
# Author: Walter Borin
# Mod: Flavio Anesi
#
# Do NOT "set -e"
TODAY=$(date +"%d-%b-%Y") # Today's date like DD-MMM-YYYY
RMDATE=$(date +"%d-%b-%Y" -d '30 days ago') # TODAY minus X days ago - too old files
FTPUSER=xxxxxxx # User (inserire il vostro UserId)
FTPPW=xxxxxxx # Password (inserire la password per l'ftp)
FTPSERVER=192.168.1.10 # IP server ftp (indirizzo IP del server)
LFTP=/usr/bin/lftp # Path to binary
DATADIR=/var/www/123solar/data # Your data archives are here- 123solar
DATADIR2=/var/www/metern/data # Your data archives are here - metern
```

```

TMPDIR=/var/www/MyScripts # Your temp backup file are here
FTPDIR=/Public/backup # Your backup dir in ftp dir
cd $TMPDIR
tar -czf backupsolar_$TODAY.tar.gz $DATADIR
tar -czf backupmetern_$TODAY.tar.gz $DATADIR2
$LFTP << EOF
open $FTPUSER:$FTPPW@$FTPSERVER
put -O $FTPDIR backupsolar_$TODAY.tar.gz
wait
put -O $FTPDIR backupmetern_$TODAY.tar.gz
wait
cd $FTPDIR
rm -rf backupsolar_$RMDATE.tar.gz
wait
rm -rf backupmetern_$RMDATE.tar.gz
wait
close
exit
EOF
rm -rf backupsolar_$TODAY.tar.gz
rm -rf backupmetern_$TODAY.tar.gz
#echo "Backup del: $TODAY salvato in FTP"
#

```

ctrl+O per salvare e ctrl+X per uscire

Sarà necessario modificare la parte <UserId>, <Password> e <IPserver> con user, password e indirizzo IP del vostro sito ftp

**ATTENZIONE:** sul sito ftp dovete creare la directory <backupdir> sulla radice principale a vostro piacimento (es. "/Public/backup")

Diamo ora i permessi in esecuzione allo script:

```

../$ sudo -s
../# cd /var/www/MyScripts
../# chmod -v 755 ftpbackup.sh

```

a questo punto inseriamo in crontab l'esecuzione automatica all'ora desiderata

```

../$ sudo -s
../# cd /etc
../# nano crontab

```

```

# /etc/crontab: system-wide crontab
..... (parte esistente).....
45 20 * * * root /var/www/MyScripts/ftpbackup.sh

```

ctrl+O per salvare e ctrl+X per uscire

## **APPENDICE F      Invio dati produzione e consumo su Pvoutput.org**

[PVOutput.org](#) è un servizio gratuito online per la condivisione, il confronto e il monitoraggio in tempo reale di un impianto solare fotovoltaico e dei consumi di energia.

Pvoutput.org ha anche delle ottime applicazioni per sistemi Android ed iOS per visualizzare i dati sui rispettivi dispositivi mobili:

**Android:** [PV Output](#)

**iOS:** [PVOutput](#)

I software 123solar e meterN sono già predisposti per poter inviare i dati anche a questo interessante servizio online, che permette fra l'altro di fare dei confronti con altri utenti.

Credo sia importante evidenziare che questo servizio è gratuito, ma i vostri dati saranno liberamente visibili a tutti, a meno di non pagare una piccola iscrizione che vi consentirà di avere dei vantaggi aggiuntivi, fra cui la possibilità di rendere privati i vostri dati e non visibili da tutti.

Per poter usufruire anche di questo servizio sarà necessario fare qualche ulteriore configurazione e prima di tutto sarà necessario crearsi un proprio account sul sito pvoutput.org.


### **Creazione account su pvoutput.org**

Se non abbiamo già creato in precedenza un account, portiamoci sulla pagina di registrazione

<http://pvoutput.org/register.jsp>

Inseriamo i dati richiesti e creiamo il nostro account



**Register Account**  
Choose a login   
Password   
Retype Password   
Email   
  
Can't read the image? Click on it for a new one.  
Type the code shown

Creato l'account dovremo ora loggarci con le nostre nuove credenziali e andare ad abilitare l'API access (1), appuntarci l'API key (2) ed il nostro System ID (3) che andremo poi ad inserire nelle pagine di configurazione di 123solar.

## 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	<a href="#">Add System</a>
<input type="checkbox"/> <a href="#">123solar</a>	<a href="#">10230</a>	Active	<a href="#">Edit</a>

## Configurazione 123solar

Portiamoci ora nella pagina di amministrazione di 123solar:

[http://IP\\_raspberry/123solar/admin/admin.php](http://IP_raspberry/123solar/admin/admin.php)

portiamo sulla pagina "Configure your inverter(s)", ed in particolare alla ultime righe di tale pagina:

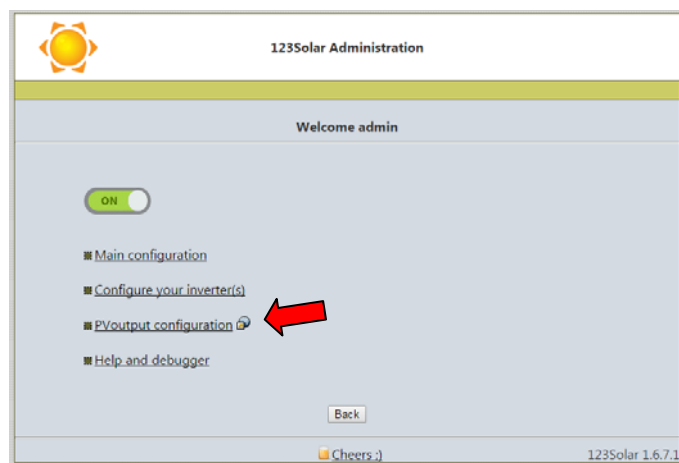
**PVoutput.org :**  
☐ Live Feed  API key  Sys. ID  Consumption

Modificate quindi la riga come di seguito, inserendo il vostro API key ed Sys. ID che vi siete appuntati in precedenza.

**PVoutput.org :**  
☐ Live Feed  API key  Sys. ID  Consumption

Scegliendo inoltre nella casella "Consumption" la voce "meterNhousehold" andremo ad abilitare anche l'invio su Pvoutput.org dei nostri dati relativi al consumo rilevati da MeterN

Nelle nuove versioni di 123solar (dopo 1.6.7) dalla pagina di amministrazione clicchiamo "Pvoutput configuration",



**PVoutput configuration**

Number of PVoutput system(s)

Sys. ID	API key	Consumption	Temperature	Inverter(s)
<input type="text" value="sys ID"/>	<input type="text" value="vostro API key"/>	<input type="text" value="meterN"/>	<input type="text" value="inverter"/>	<input checked="" type="checkbox"/> 1

Modificate quindi la riga come di seguito, inserendo il vostro API key ed Sys. ID che vi siete appuntati in precedenza.

Scegliendo inoltre nella casella "Consumption" la voce "meterN" andremo ad abilitare anche l'invio su Pvoutput.org dei nostri dati relativi al consumo rilevati da MeterN

## Configurazione dati consumo MeterN

Per inviare su Pvoutput.org anche i dati relativi al consumo, sarà necessario editare il file `/var/www/123solar/scripts/pvoconsumption/meterNhousehold.php`

```
pi@raspberrypi ~ $ nano /var/www/123solar/scripts/pvoconsumption/meterNhousehold.php
```

andando a modificare le parti evidenziate in rosso:

```
<?php
// For meterN Electrical household meter
// You'll need to setup the following variables :
$meterndir = '/var/www/meterN/'; // meterN csv data path
$metnum    = 2; // meterN household meter number
$passomn   = 0; // pass-over of your meterN counter
//
```

ctrl+O per salvare e ctrl+X per uscire

Non ci resta che verificare che i nostri dati vengano correttamente inviati su PVoutput.org

## APPENDICE G      UPS per RASPBERRY

**FONTE:** [Raspberry Pi UPS by using a USB power bank](#)

La mancanza improvvisa di tensione, può creare dei problemi al nostro sistema di monitoraggio, introducendo nelle letture dei picchi non realistici e, cosa non da poco, può portare a danneggiare irrimediabilmente il sistema operativo del Raspberry al punto da rendere inutilizzabile la schedina.

Per scongiurare questi problemi, è quindi molto importante dotare il nostro Raspberry di una fonte alternativa di alimentazione.

La soluzione più semplice ed economica è quella di alimentare il Raspberry per il tramite di un power-bank, che anche in assenza di tensione di rete provvederà a tenere acceso il Raspberry per diverse ore.

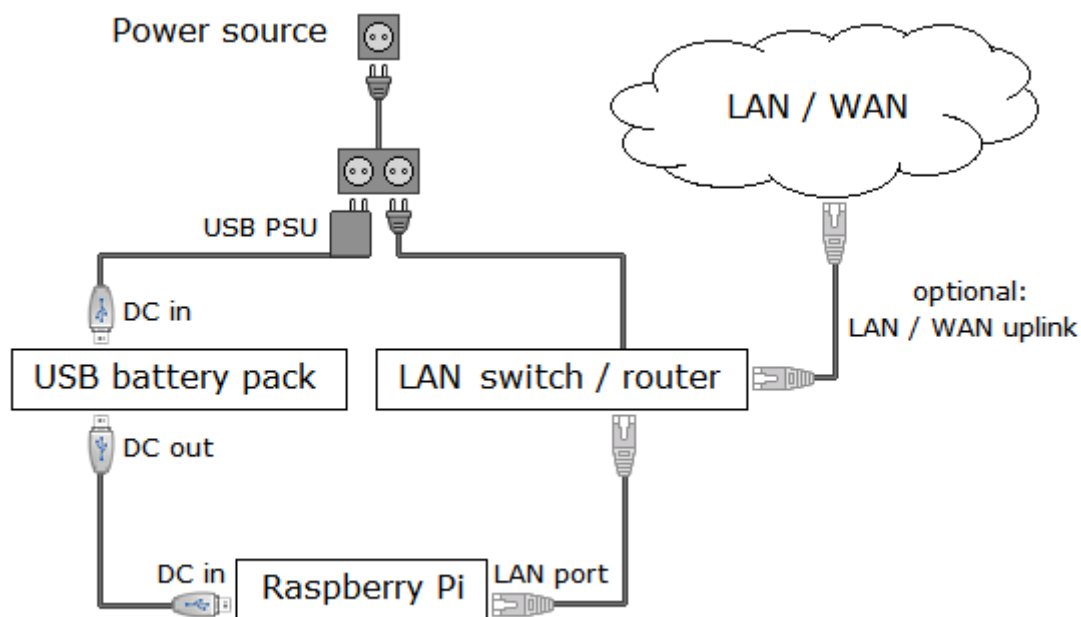


Personalmente ho adottato questa soluzione con una finezza in più, un piccolo software che in caso di assenza di rete, "stima" la durata della batteria, ed in caso di assenza prolungata di alimentazione provvede a spegnere via software il Raspberry, scongiurando qualsiasi problema.

Particolare attenzione dovrà essere posta alla scelta del power-bank. Si dovrà infatti scegliere un power bank che può rimanere sempre alimentato e che sia dotato di uno switch interno che in caso di carica completa provvede ad alimentare il nostro Raspberry direttamente tramite la rete.

Personalmente ho utilizzato l'[EasyAcc Classic Gen2 5200mAh Power Bank](#) che si trova in rete per circa 15€ e che ben si presta a tale scopo, garantendo un'autonomia di circa 6 ore ed essendo dotato di una presa microUSB di ingresso e di una USB in uscita.

Lo schema di collegamento sarà quindi il seguente:



Con tale configurazione sarà anche possibile utilizzare il software [upsd \(Uninterruptible Power Supply Daemon\)](#) che permette, come visto, di "stimare" la durata della batteria ed in caso di assenza di alimentazione prolungata spegnere il Raspberry.

Il software rileva la mancanza di rete attraverso la presa LAN, ecco perché è importante che vi sia il collegamento LAN (basta anche il solo collegamento ad uno switch LAN anche senza rete internet)

### Installazione software

Da terminale (Putty) creiamo inoltre il link simbolico:

```
wget http://raspi-ups.appspot.com/upsd/upsd_1.2-1.deb
sudo dpkg -i upsd_1.2-1.deb
```

Per eventualmente disinstallare il programma basterà digitare sempre da terminale:

```
sudo dpkg -r upsd.
```

Per controllare se il software è stato correttamente installato basterà digitare da terminale:

```
upsd -i
```

che in caso positivo vi restituirà le seguenti informazioni:

```
IS_RUNNING=YES
```



```
PID=2305
LAST_UPDATED=2015-08-16 15:47:13
POWER_OUTAGE=NO
BATTERY_CHARGE_LEVEL=100
BATTERY_REMAIN_TIME=06:40:53
BATTERY_LOW=NO
```

Per leggere invece la documentazione completa del software:

```
man upsd
```

ed anche

```
man upsd.conf
```

## Configurazione software

Al fine di garantire un corretto funzionamento ed in particolare una corretta stima della durata della carica del power-bank, è necessario effettuare alcune configurazioni che sono chiaramente spiegate nel manuale (man upsd.conf)

Con WinSCP andiamo ad editare il file upsd.conf che si trova nella cartella /etc/upsd/.

Dovremo andare ad inserire i seguenti parametri:

BATTERY\_RUN\_TIME : indica il tempo in secondi di durata della batteria

Come indicato nella guida potrà essere determinato in due modi:

- Misurandolo: a power-bank completamente carico, togliere l'alimentazione e lasciare acceso il Raspberry alimentato mediante la batteria ed attendere fino alla scarica completa, cronometrando la durata.
- Calcolandolo: la guida fornisce un sistema approssimativo di calcolo:

```
runtime = 3600 * Wh / W
           Wh = battery capacity in watt-hours (Wh)
           W = average system power consumption in watt (W)
           Raspberry Pi Model A: 1.5 W
           Raspberry Pi Model A+: 1 W
           Raspberry Pi Model B: 2.5 W
           Raspberry Pi Model B+: 1.75 W
           Raspberry Pi Model 2 B: 2 W
```

Pertanto con il power-bank indicato da 5200mAh oppure da 19,24 Wh nel caso di un Raspberry B+ avremo:  $3600 * 19.24 / 1.75 = 39579s$

Personalmente ho adottato questo sistema in quanto non mi interessa la precisione, mi basta che il sistema si spenga in condizioni di sicurezza ed ho anche adottato un consumo di 2W per il mio Raspberry B+, superiore a quello indicato, ottenendo:  $3600 * 19.24 / 2 = 34362s$

**BATTERY\_CHARGE\_TIME:** indica il tempo impiegato in secondi per una completa ricarica

Anche in questo caso la guida indica i due modi per determinare questo parametro:

- Misurandolo: a power-bank completamente scarico, collegare l'alimentazione e attendere fino alla carica completa, cronometrando la durata.
- Calcolandolo: la guida fornisce un sistema approssimativo di calcolo:

```
chargetime = 3600 * Wh / (min(As, Ai) * V)
Wh = capacità batteria in watt-ora (Wh)
As = corrente massima erogata dall'alimentatore, in ampere (A)
Ai = corrente massima del connettore di input, in ampere (A)
V = tensione della batteria, in volt (V)
```

Pertanto con il power-bank indicato da 5200mAh avremo:

- Wh = 19.24 Wh
- As = 2,0A (corrente erogata dal mio alimentatore)
- Ai = 2,1A
- V= 5V

E quindi  $\text{chargetime} = (3600 * 19.24) / (2,0 * 5) = 6926 \text{ s (115 min)}$

**BATTERY\_LOW\_LEVEL:** indica il livello in % di carica della batteria sotto il quale il software la considera scarica e procede con lo spegnimento.

Personalmente ho lasciato il 30% già predisposto per garantire un discreto margine di sicurezza.

Ricavati tutti i parametri necessari possiamo quindi procedere ad editare con WinSCP il file upsd.conf che si trova nella cartella /etc/upsd/ inserendo i dati indicati in rosso

```
# This is the configuration file for the upsd daemon program.
# Please see the upsd.conf(5) man page for further documentation.

# Enabled switch.
ENABLED = YES

# The network interface used to determine power outages.
INTERFACE = eth0

# The check interval for power outages, in milliseconds (ms).
CHECK_INTERVAL = 1000

# The update interval of the upsd.status file, in seconds (s).
UPDATE_INTERVAL = 10

# The amount of time the fully charged battery of the UPS can power the system, in seconds (s).
BATTERY_RUN_TIME = 34362

# The amount of time the empty battery of the UPS needs to become fully charged, in seconds (s).
BATTERY_CHARGE_TIME = 6926
```

```
# The charge level under which the battery will be considered empty, in percent (%).  
# Set this to 0 to disable this feature.  
BATTERY_LOW_LEVEL = 30  
  
# The time limit after a loss of power at which the battery will be considered empty, in seconds (s).  
# Set this to 0 to disable this feature.  
BATTERY_LOW_TIME = 0
```

Tali dati dovranno ovviamente essere adattati alla capacità del vostro power-bank con quanto indicato sopra.

Inutile dire che la stima della durata sarà tanto più precisa, quanto preciso sarà il dato che andremo ad inserire.

Ci rimane solo da configurare cosa deve eseguire il programma in caso di raggiungimento del livello di scarica previsto (30%), e cioè lo spegnimento del Raspberry.

Per fare ciò, dovremmo editare con WinSCP il file `low_battery` che si trova nella cartella `/usr/lib/upsd`

```
#!/bin/bash  
  
# This script is executed by upsd if there is a power outage AND the battery  
# charge level is considered to be low. Whenever this happens, you have to  
# expect an abrupt loss of power very soon. It thus usually is the best idea  
# to properly shut down the system here.  
  
# Uncomment the line below to properly shut down the system. Note that upsd  
# must be running as root, otherwise this will not work.  
  
shutdown -h now
```

Basterà quindi togliere il commento (#) davanti all'ultima riga.

Abbiamo poi a disposizione ulteriori due file sempre nella cartella `/usr/lib/upsd` in cui inserire dei comandi che il sistema eseguirà nelle due condizioni:

- power\_back : comandi da eseguire al ritorno dell'alimentazione dopo una interruzione. Si potrà ad esempio inserire un comando di invio di una email a segnalare la mancanza di tensione
- power\_outage : comandi da eseguire appena si rileva una mancanza di alimentazione

## APPENDICE H Preservare la scheda SD da possibili danni

### FONTI:

- [STOPPING SD CARD CORRUPTION ON RASPBERRY PI'S RASPBIAN](#)
- [Limit SD Card writes on Raspberry Pi using Ramlog](#)

Siamo tutti a conoscenza del fatto che le schede SD hanno un numero seppur grande ma comunque limitato di scritture. Possiamo quindi immaginare come un sistema di monitoraggio e salvataggio dei dati, "stressi" particolarmente la scheda SD.

Anche se tutto il sistema di monitoraggio è stato studiato per limitare le scritture sulla scheda SD andando a memorizzare i dati istantanei in ramdisk, vi sono comunque numerosi altri servizi che vanno a scrivere in continuazione sulla scheda.

Fra questi vi sono tutti i log di sistema, che solitamente per il Raspberry vengono salvati nella cartella /var/log

Vediamo di seguito un sistema per spostare in ramdisk anche i log di sistema, ma senza però perdere questi dati in caso di spegnimento del Raspberry.

### RamLog

Ramlog, è un utile tool per evitare di far lavorare troppo la sd, sperando di allungarle la vita. Questo servizio all'avvio crea un disco virtuale in memoria (ramdisk), vi copia i files contenuti in /var/log e poi lo monta al posto di /var/log, in questo modo tutte le modifiche ai files verranno effettuate in ram senza continuare a scrivere sulla scheda SD, rischiando di danneggiarla.

Inoltre in fase di spegnimento, il sistema scriverà su sd i files contenuti nella /var/log in memoria, in modo da non perdere nemmeno in caso di riavvio i nostri log.

Per installarlo eseguire da terminale:

```
sudo apt-get install lsof rsync
wget http://www.tremende.com/ramlog/download/ramlog_2.0.0_all.deb
sudo dpkg -i ramlog_2.0.0_all.deb
rm ramlog_2.0.0_all.deb
```

Ora dobbiamo modificare alcuni files di configurazione:

```
sudo nano /etc/init.d/ramlog
```

Ctrl+O per salvare e Ctrl+X per uscire

Aggiungiamo le seguenti due linee nella parte iniziale del file di configurazione (la parte che inizia con: #BEGIN INIT INFO). Includere anche i caratteri #

```
# X-Start-Before: rsyslog
# X-Stop-After: rsyslog
```

Ora editiamo il file `/etc/init.d/rsyslog`

```
sudo nano /etc/init.d/rsyslog
```

aggiungiamo 'ramlog' alla fine di queste due linee esistenti:

```
# Required-Start: $remote_fs $time ramlog
# Required-Stop: umountnfs $time ramlog
```

Ctrl+O per salvare e Ctrl+X per uscire

Dopo aver modificato i file di configurazione, eseguire:

```
sudo inserv
sudo reboot
```

Dopo il riavvio del sistema, riavviamo un'altra volta:

```
sudo reboot
```

Dopo il secondo riavvio controlliamo eventuali errori nel file log di ramlog :

```
sudo cat /var/log/ramlog
```

Se troviamo una riga tipo la seguente:

```
Aug 09 10:52:29 Starting ramlog-tmpfs 2.0.0: [ OK ]
```

significa che ramlog è partito e funziona regolarmente

Potrebbero invece esserci dei messaggi di errore in quanto è necessario che Ramlog sia avviato prima di altri demoni. Per esempio Samba. In tal caso è necessario editare il file appropriato in `/etc/init.d`

Per esempio per Samba:

```
sudo nano /etc/init.d/samba
```

Aggiungere 'ramlog' alla fine delle seguenti linee:

```
# Required-Start: $network $local_fs $remote_fs ramlog
# Required-Stop: $network $local_fs $remote_fs ramlog
```

Queste linee assicurano che ramlog sia avviato prima che l'altro demone (come samba) venga avviato.

Dopo aver modificato i file di configurazione, eseguire nuovamente:

```
sudo insserv  
sudo reboot
```

Dopo il riavvio possiamo anche controllare le partizioni montate per verificare che ramlog sia effettivamente partito.

Da terminale digitiamo:

```
df
```

e dovremmo vedere la partizione ramlog-tmpfs:

```
root@raspberrypi:/var# df
File system      1K-blocchi    Usati Disponib.  Uso% Montato su
rootfs           7515692 3958504   3206564    56% /
/dev/root        7515692 3958504   3206564    56% /
devtmpfs         219744      0    219744     0% /dev
tmpfs            44784      236    44548     1% /run
tmpfs            5120       4     5116     1% /run/lock
tmpfs            89560      16    89544     1% /run/shm
/dev/mmcblk0p1   57288     9928    47360    18% /boot
/dev/root        7515692 3958504   3206564    56% /var/log.hdd
ramlog-tmpfs     223912    70048    153864    32% /var/log
```

## Disabilitare il file swapping

Una ulteriore protezione per evitare la corruzione della scheda SD è di disabilitare lo swapping

Il Raspberry usa dphys-swapfile per controllare lo swapping.

Questo crea dinamicamente una partizione di swap. Utilizzeremo questo tool per disabilitare lo swap.

Eseguire i seguenti comandi da terminale per disabilitare lo swap:

```
sudo dphys-swapfile swapoff  
sudo dphys-swapfile uninstall  
sudo update-rc.d dphys-swapfile remove
```

Fatto ciò potremo eseguire il comando `free -m` per verificare l'uso della memoria:

```
root@raspberrypi:/var# free -m
              total        used         free       shared    buffers     cached
Mem:           437          250           186           0          15         183
-/+ buffers/cache:           51          385
Swap:            0            0            0
```

Troveremo che lo swap è stato disabilitato.

Anche in caso di riavvio questa soluzione sarà permanente.

## Controllare l'utilizzo della MicroSD di Raspberry PI

**FONTI:** [Controllare l'utilizzo della MicroSD di Raspberry PI , ovvero ma quanto mi scrivi... :\)](#)

Vediamo ora come sia possibile verificare se vi sono altri programmi che scrivono dei log troppo spesso sulla nostra scheda SD.

Con l'utilizzo di Ramlog e la disabilitazione dello swapping abbiamo già limitato l'uso della SD, ma di seguito vediamo come sia possibile investigare se ci sia scappato ancora qualcosa.

Innanzitutto si installano i tools necessari con il comando:

```
apt-get install inotify-tools
```

Per lanciare il monitor in tempo reale (su tutto il file sistem) ed ottenere un elenco (log) di tutti gli accessi in scrittura ai file, si utilizza il seguente comando (tutto su un'unica riga):

```
echo 16384 > /proc/sys/fs/inotify/max_user_watches && inotifywait -m -r --format "%T %w %e %f" --timefmt="%F %T" -e modify,move,create,delete,attrib --exclude='/(dev|run(/lock|/shm)|proc|var/log)' /
```

Se invece preferite una statistica su un periodo prefissato (nell'esempio 5 minuti = 300 secondi) per capire dove concentrare le ricerche, è possibile eseguire il comando:

```
echo 16384 > /proc/sys/fs/inotify/max_user_watches && inotifywatch -t 300 -r -e modify,move,create,delete,attrib - --exclude='/(dev|run(/lock|/shm)|proc|var/log)' /
```

**NOTA:** nelle esclusioni ho già inserito la cartella /var/log in quanto l'abbiamo già spostata in ram con l'utilizzo di ramdisk

Di seguito un esempio di output "bello pulito":

```
# inotifywatch -t 300 -r -e modify,move,create,delete,attrib --exclude='/(dev|run(/lock|/shm)|proc)' /
Establishing watches...
Finished establishing watches, now collecting statistics.
total modify filename
1 1 /var/www/metern/data/csv/
1 1 /var/www/123solar/data/inv1/csv/
```

**NOTA:** Se volete escludere dal controllo anche la dir /tmp o altre cartelle perchè li avete già spostati in ram o altrove, basta aggiungere nelle esclusioni:

```
--exclude='/(dev|run(/lock|/shm)|proc|var/log |tmp)'
```

Questo tipo di comandi, a differenza del classico iotop e similari, facilita l'individuazione e nella

rimozione o spostamento in RAM dei singoli file frequentemente scritti ed allunga la vita della nostra MicroSD. Nel mio caso, avevo 2 o 3 programmi che scrivevano continuamente gli stessi file e delle entry nei log di sistema che venivano scritte troppo frequentemente. Con un monitoring chirurgico sono riuscito ad eliminare tutto il superfluo con grande soddisfazione.

Buona caccia!

## **APPENDICE I      Configurare la rete LAN o WIFI**

Per la configurazione della rete LAN e del wifi vi suggerisco questa interessante utility da installare sul raspberry:

[wicd-curses](#)

E' dotata di una pseudo interfaccia grafica in modalità testo e permette di ricercare le reti wifi, connettersi ad una rete selezionata dalla lista, impostare IP statici per LAN e wifi ed altro. Personalmente la trovo semplice e pratica, oltre ad essere in italiano.

Nel link sopra trovate le istruzioni per l'installazione e l'uso.

SE VI INTERESSA ACQUISTARE IL CONTATORE  
EASTRON SDM120modbus o SDM220modbus :

 **Compra adesso**

SE IL MIO LAVORO VI E' STATO UTILE, OFFRITEMI DA BERE,  
FATEMI UNA DONAZIONE :

 **Donazione**