# Sommario

**ComApps** ..... 3
**Le Comapps** ..... 3
***Comapss standard*** ..... 3
***Comapss autoprodotte*** ..... 3
***Struttura delle comapps*** ..... 4
**Creare una comapps** ..... 4
***Comapps pooler*** ..... 4
***Comapps daemon*** ..... 6
Daemon_loop ..... 6
daemon ..... 8
Com_daemon ..... 9
**Configurazione ComApps in MeterN** ..... 11

# ComApps

# Le Comapps

Le comapps sono dei programmi, in qualsiasi linguaggio, che servono per interrogare un dispositivo di misura e leggerne i dati.

Per poterlo fare bisogna innanzitutto che il dispositivo abbia apposite API per poterlo interrogare, una volta che abbiamo i dati li dobbiamo formattare in un apposito formato leggibile da MeterN, ne abbiamo parlato **qui** .

MeterN ha già delle comapss preconfezionate realizzate e messe a disposizione dall'autore, basta solo impostare i propri dati, altre dobbiamo realizzarle da noi, a questo scopo possiamo aiutarci studiando i vari esempi messi a disposizione dall'autore al seguente LINK.

Le comapps devono essere tutte archiviate in una directory, meglio se fuori dalla directory che contiene MeterN cosi in caso di reinsatallazione e/o aggiornamento non verranno modificate o cancellate.

## Comapss standard

- pool123s Se avete installato e funzionante 123solar questa comapss si intefaccia con l'altro applicativo e passa i dati necessari a MeterN
- remotepool123s come il precedente ma legge i dati su un sito remoto dove è installato 123solar
- houseenergy E' una comapps che legge vari meter, imposta vari Meter Id (i meter id sono fissi in quanto servono per calcolare i dati)

## Comapss autoprodotte

Le comapps sono di due tipi, alla fine fanno le stesse cose, ma le fanno in modo diverso:

- Pooler = leggono continuamente i dati direttamente dal meter (un dato alla volta)
- Daemon = sono dei programmi che leggono tutti i dati dal meter li parcheggiano su un file di supporto da dove vengono letti mediante un pooler

per capire la differenza di funzionamento facciamo un esempio comparativo, supponiamo di dover leggere 4 dati da un meter (W prodotti, Wh prodotti, W consumati, Wh consumati).
Con un pooler si devono fare 4 cicli di lettura sul meter (tipicamente un SDM o uno Shelly)
Con un Daemon si fa un solo ciclo di lettura sul meter e poi si legge 4 volte il file di supporto.

Il primo è da preferire se si leggono uno o due dati, il secondo è molto più performante se si devono leggere molti dati inoltre è molto utile nei casi in cui si debbano leggere dati da siti web dove ci sono limiti al numero di letture orarie e/o giornaliere

## Struttura delle comapps

Pooler

Sono file singoli che vengono attivati direttamente da MeterN e si interfacciano direttamente col meter

| Programma | Descrizione | Esempio |
|---|---|---|
| pooler.php | programma che legge i dati dal meter e li passa a MeterN | shelly1.php |

Daemon

qui la cosa si fa più strutturata, dobbiamo sempre avere:

| Programma | Descrizione | Esempio |
|---|---|---|
| com_daemon.php | Avvia il daemon_loop | shelly0_daemon.php |
| daemon_loop.php | legge i dati dal meter | shelly0_loop.php |
| daemon.php | pooler che legge i dati dal file di supporto | shelly0.php |

Ogni pooler o daemon dovrà gestire tanti parametri ( il nome con cui MeterN richiama il dato) tamti quanti sono i dati che si vogliono gestire. questo parametro verrà richiamato da MeterN e sarà inserito nella configurazione dei vari meter.

# Creare una comapps

---

## Comapps pooler

Dobbiamo posizionare il programma nella directory che conterrà tutte le comapps, ma lo dobbiamo fare come amministratore, tutta la procedura andrebbe fatta seguendo il flusso senza interromperla per non perdere i diritti di amministratore senza accorgersene.
Procedere nel seguente modo:

```
sudo -s
cd /var/www/comapps
```

Con il nostro editor preferito creiamo il file:

```
poolshelly.php
```

Al suo interno andremo a inserire il seguente codice:

```
1. #!/usr/bin/php
2. <?php
3.
4. if (isset($_SERVER['REMOTE_ADDR'])) {
5. die('Direct access not permitted');
6. }
7.
8. error_reporting(~E_WARNING);
9.
```

```php
// This script will output a ShellyEM counter into a meterN compatible format
// Configure, then ln -s /var/www/comapps/poolshelly.php /usr/bin/poolshelly and chmod +x poolshelly.php
// Request Main command with 'poolshelly {energy | energy1 | power | power1}'

// Shelly IP
$IP = '192.168.xxx.yyy';
// Shelly request
$url = "http://"."$IP"."/status";
$remotedata = file_get_contents($url);

// meterN config

$METERID    = 1;

// No edit is needed below

if (isset($argv[1])) {

    $GP = null;
    if (!empty($remotedata)) {
        $memarray = json_decode($remotedata, true);
        if ($argv[1] == 'power') {
            $GP = $memarray['emeters'][0]['power'];
            $GP = round($GP, 1);
            echo "$METERID($GP*W)\n";
        }
        elseif ($argv[1] == 'energy') {
            if (isset($memarray['emeters'][0]['total'])) {
                $KWHT = round($memarray['emeters'][0]['total']); // Wh
                echo "$METERID($KWHT*Wh)\n";
            } else {
                die("Abording: KWHT[0] not defined\n");
            }
        }
        elseif ($argv[1] == 'power1') {
            $GP = $memarray['emeters'][1]['power'];
            $GP = round($GP, 1);
            echo "$METERID($GP*W)\n";
        }
        elseif ($argv[1] == 'energy1') {
            if (isset($memarray['emeters'][1]['total'])) {
                $KWHT = round($memarray['emeters'][1]['total']); // Wh
                echo "$METERID($KWHT*Wh)\n";
            } else {
                die("Abording: KWHT[1] not defined\n");
            }
        }
    } else {
        die("Abording: Nessun dato presente\n");
    }
} else {
    die("Usage: poolshelly { energy | energy1 | power | power1 }\n");
}
?>
```

dobbiamo inoltre rendere il file eseguibile dal sistema, per far questo eseguiamo i seguenti passaggi:

```
cd /var/www/comapps
chown www-data:www-data poolshelly.php
chmod a+x /var/www/comapps/poolshelly.php
```

a questo punto dobbiamo creare un link simbolico per poter richiamare il programma

```
ln -s /var/www/comapps/poolshelly.php /usr/bin/poolshelly
```

ora siamo pronti per l'operazione più importante, testare il software creato prima di darlo in pasto a MeterN.

Dalla shell, con riga di comando lanciare tutte le opzioni previste nel software per assicurarsi che diano il risultato voluto, se avete previsto la gestione degli errori, generateli, per assicurarvi che il software li gestisca correttamente, nell'esempio precedente:

```
poolshelly energy
poolshelly energy1
poolshelly power
poolshelly power1
poolshelly
```

l'ultimo genera l'errore di nessuna opzione richiesta e visualizza le opzioni possibili.

---

# Comapps daemon

Come detto in precedenza dobbiamo creare 3 programmi, per tenere in ordine i programmi, se non già fatto, creeremo una directory dove posizionare i com_daemon che in effetti non servono a manipolare i dati ma ad avviare i daemon_loop.

```
sudo -s
cd /var/www/comapps
mkdir -v daemon
```

## Daemon_loop

Iniziamo ora a creare il programma di loop che andrà a leggere il nostro meter nello specifico si tratta di uno Shelly EM con una sola pinza che legge immissioni e prelievi distinguendoli con i valori positivi per i prelievi e negativi per le immissioni, il file lo chiameremo shelly_3_loop.php

shelly_3_loop.php

```
1. #!/usr/bin/php
2. <?php
3. if (isset($_SERVER['REMOTE_ADDR'])) {
4.     die('Direct access not permitted');
5. }
6. // Beware, only use a tmpfs as /dev/shm (ramfs) !
7.
8. // This script will perform a loop to continuosly  request data to ShellyEM
```

```
    meter and put data into a Shelly_n_log.txt file
 9. // Request the follow data {energy | power | volt | ampere | cosphi | energy1
    | power1 }'
10. // **** Measurements performed with a single clamp that measures import + ed
    export - ******
11. // by Walter62  rev 00   on 02/07/23
12.
13. $IP = '192.168.xxx.yyy'; // Shelly IP for import export energy
14. $CH = 0;                    // Channel "0 = import" or "1 = export" which you
    want use for volt-ampere-cosphi
15. $MID = 3;                   // meterN config
16.
17. // No edit is needed below
18.
19. $url = "http://"."$IP"."/status"; // Shelly request
20. $filelog = "shelly_" . $MID . "_log.txt"; // Temporary file for dataset exaple
    "selly_1_log.txt"
21.
22. while (true) {
23.     $dataarray = array();
24.     $remotedata = "";
25.
26.     $remotedata = file_get_contents($url);
27.     //print_r($remotedata);
28.
29.     if (!empty($remotedata)) {
30.         $memarray = json_decode($remotedata, true);
31.
32.         $GP   = round($memarray['emeters'][0]['power'],1); //W
33.
34.         if ($GP <0){
35.             $GP_0 = 0;
36.             $GP_1 = abs($GP);
37.         } else {
38.             $GP_0 = $GP;
39.             $GP_1 = 0;
40.         }
41.
42.         //channel 0 ==> Import
43.         //$GP_0   = round($memarray['emeters'][0]['power'],1); //W
44.         $KWHT_0 = round($memarray['emeters'][0]['total'],1); // Wh
45.
46.         //Channel 1 ==> Export
47.         //$GP_1   = round($memarray['emeters'][1]['power'],1); //W
48.         //$KWHT_1 = round($memarray['emeters'][1]['total_returned'],1); // Wh
49.         $KWHT_1 = round($memarray['emeters'][0]['total_returned'],1); // Wh
50.
51.         //channel $CH
52.         $VOLT   = round($memarray['emeters'][$CH]['voltage'],1); // V
53.         $AMP    = abs(round (${'GP_'.$CH}/$VOLT,1)); //A
54.         $PHI    = abs(round($memarray['emeters'][$CH]['pf'],2)); // cosphi
55.
56.         $dataarray[0] = $MID . '_0(' . $GP_0 . '*W)';    // n_0(nnn*W)
57.         $dataarray[1] = $MID . '_0(' . $KWHT_0 . '*Wh)';
58.         $dataarray[2] = $MID . '_1(' . $GP_1 . '*W)';
59.         $dataarray[3] = $MID . '_1(' . $KWHT_1 . '*Wh)';
```

```
60.         $dataarray[4] = $MID . '_V(' . $VOLT . '*V)';
61.         $dataarray[5] = $MID . '_A(' . $AMP . '*A)';
62.         $dataarray[6] = $MID . '_PHI(' . $PHI . '*phi)';
63.         $str          = implode(PHP_EOL, $dataarray);
64.         file_put_contents('/dev/shm/' . $filelog, $str);
65.
66.     } else {
67.         if (file_exists($filelog)) {
68.             $now = time();
69.             if ($now - filemtime($filelog)> 5) { // 5 sec
70.                 unlink('/dev/shm/' . $filelog);
71.             }
72.         }
73.     }
74.     usleep(500000);
75. }
76. ?>
```

Dato che con una pinza leggo sia il valore di energia importata che esportata ma con valore negativo, ho disabilitato la lettura del canale e dalla riga 34 alla 40 vado a leggere il valore di energia letto dalla pinza e in funzione che sia positivo o negativo assegno al canale appropriato il valore assoluto quindi positivo per entrambi i canali.

Ora che abbiamo il file compilato lo salviamo in:

```
/var/www/comapps
```

### daemon

Ora passiamo alla stesura del programma che si occuperà di leggere i dati e renderli disponibili a MeterN, lo chiameremo shelly_3.php, il contenuto dovrà essere il seguente:

shelly_3.php

```
1. #!/usr/bin/php
2. <?php
3. if (isset($_SERVER['REMOTE_ADDR'])) {
4.     die('Direct access not permitted');
5. }
6. // This script will read output file of the ShellyEM counter and will expos
   the data to MeterN
7. // Configure, then ln -s /var/www/comapps/shelly_n.php /usr/bin/shelly_n and
   chmod +x shelly_n.php
8. // Request Main command with 'shelly_1 {energy | power | volt | ampere |
   cosphi | energy1 | power1 }'
9. // by Walter62  rev 00   on 02/07/23
10.
11. $MID = 3;  // meter to call - shelly_n = shelly_3
12. $filelog = "shelly_" . $MID . "_log.txt"; // Temporary file for dataset exaple
    "selly_2_log.txt"
13.
14. // No edit is needed
15.
```

```
$help = "Usage: shelly_3 { energy | power | volt | ampere | cosphi | energy1 | power1 }\n";

if (!isset($argv[1])) {
    die($help);
}

if ($argv[1] == 'power'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_0\(" | grep "*W)"');
}elseif ($argv[1] == 'energy'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_0\(" | grep "*Wh)"');
}elseif ($argv[1] == 'power1'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_1\(" | grep "*W)"');
}elseif ($argv[1] == 'energy1'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_1\(" | grep "*Wh)"');
}elseif ($argv[1] == 'volt'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_V\(" | grep "*V)"');
}elseif ($argv[1] == 'ampere'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_A\(" | grep "*A)"');
}elseif ($argv[1] == 'cosphi'){
    $outstr =  exec('cat /dev/shm/'.$filelog .' | egrep "^'. $MID .'_PHI\(" | grep "*phi)"');
} else{
    die($help);
}

echo "$outstr";
?>
```

## Com_daemon

Adesso abbiamo i due programmi che che leggono i dati dal contattore e li trasmettono a meter su richiesta, dobbiamo però assicurarci che ad ogni avvio del sistema, o meglio, ad ogni avvio di MeterN si avvii anche il programma daemon_loop (il daemon lo avvia MeterN quando gli serve il dato).

Dato che è più pulito avviare il programma quando MeterN è in esecuzione ci appoggiamo al com_daemon e quindi creiamo il software necessario:

shelly_3_daemon.php

```
#!/usr/bin/php
<?php
/*
Check shelly_3_loop.php path bellow
Then ln -s  /path to/shelly_3_daemon.php /usr/bin/shelly_3_daemon
Start and stop the daemon via metern/config/config_daemon.php
And request values with shelly_3 command
*/
```

```
 9.
10. if (isset($_SERVER['REMOTE_ADDR'])) {
11.     die('Direct access not permitted');
12. }
13.
14. if (file_exists('/dev/shm/shelly_3_daemon.pid')) {
15.     $cdpid = (int) file_get_contents('/dev/shm/shelly_3_daemon.pid');
16.     exec("ps -ef | grep $cdpid | grep shelly_3_daemon", $ret);
17.     if (!isset($ret[1])) {
18.         $cdpid = null;
19.         unlink('/dev/shm/shelly_3_daemon.pid');
20.     }
21. } else {
22.     $cdpid = null;
23. }
24.
25. if (isset($argv[1])) {
26.     if (($argv[1] == 'start' || $argv[1] == 'stop') &&
    file_exists('/dev/shm/shelly_3_log.txt')) {
27.         unlink('/dev/shm/shelly_3_log.txt');
28.     }
29.     if ($argv[1] == 'start') {
30.         if (is_null($cdpid)) {
31.             $command = 'php /var/www/comapps/shelly_3_loop.php' . '> /dev/null
    2>&1 & echo $!;';
32.             $cdpid     = exec($command);
33.             file_put_contents('/dev/shm/shelly_3_daemon.pid', $cdpid);
34.         } else {
35.             echo "shelly_3_daemon seem to be running as $cdpid";
36.         }
37.     } else if ($argv[1] == 'stop') {
38.         if (!is_null($cdpid)) {
39.             $command = exec("kill $cdpid> /dev/null 2>&1 &");
40.         unlink('/dev/shm/shelly_3_daemon.pid');
41.         }
42.     } else {
43.         echo "Usage : shelly_3_daemon {start | stop}\n";
44.     }
45. } else {
46.     echo "Usage : shelly_3_daemon {start | stop}\n";
47. }
48. ?>
```

come per il pooler rendiamo i files eseguibili dal sistema, per far questo eseguiamo i seguenti passaggi:

```
sudo -s
cd /var/www/comapps
chown www-data:www-data shelly_3.php
chown www-data:www-data shelly_3_loop.php
chown www-data:www-data shelly_3_daemon.php
chmod a+x /var/www/comapps/shelly_3.php
chmod a+x /var/www/comapps/shelly_3_loop.php
chmod a+x /var/www/comapps/shelly_3_daemon.php
```

creiamo i link simbolici per poter richiamare i programmi

```
ln -s /var/www/comapps/shelly_3.php /usr/bin/shelly_3
ln -s /var/www/comapps/shelly_3_loop.php /usr/bin/shelly_3_loop
ln -s /var/www/comapps/daemon/shelly_3_daemon.php /usr/bin/shelly_3_daemon
```

ora siamo pronti per testare il software creato prima di darlo in pasto a MeterN.

Dalla shell, con riga di comando lanciare tutte le opzioni previste nel software per assicurarsi che diano il risultato voluto, se avete previsto la gestione degli errori, generateli, per assicurarvi che il software li gestisca correttamente, nell'esempio precedente:

```
poolshelly energy
poolshelly energy1
poolshelly power
poolshelly power1
poolshelly
```

l'ultimo genera l'errore di nessuna opzione richiesta e visualizza le opzioni possibili.

# Configurazione ComApps in MeterN

Quando tutto funziona possiamo abilitare i vari programmi in MeterN, per far ciò dobbiamo modificare un file di sistema (sistema di MeterN) in questo modo:

```
cd /var/www/metern/config
nano allowed_comapps.php
```

alle voci esistenti aggiungere altre voci all'array dopo l'ultima esistente

```
$ALLWDCMD[n] = 'poolshelly energy';
$ALLWDCMD[n] = 'poolshelly power';
......
......
$ALLWDCMD[n] = 'shelly_3 energy';
$ALLWDCMD[n] = 'shelly_3 power';
$ALLWDCMD[n] = 'shelly_3 volt';
$ALLWDCMD[n] = 'shelly_3 ampere';
$ALLWDCMD[n] = 'shelly_3 cosphi';
......
......
$ALLWDCMD[n] = 'shelly_3_daemon start';
$ALLWDCMD[n] = 'shelly_3_daemon stop';
```

la struttura è la seguente:

1. [*n*] = il numero tra parentesi quadre deve essere un numero progressivo che segue quelle già presenti
2. poolshelly = link simbolico che richiama il nostro programma
3. energy = opzione da richiamare

inserire tante voci quante sono le opzioni che MeterN dovrà richiamare.

Bene, ora che tutte le ComApps sono utilizzabili da MeterN ci occuppiamo di istruire MeterN ad avviare tutte le

com_daemon ad ogni avvio, per far questo basta inserirle nel file di configurazione di MeterN che si chiama “config_daemon.php”, procediamo come segue:

```
cd /var/www/metern/config
nano config_daemon.php
```

si aprirà il file in modifica, se non è mai stato usato prima troveremo al suo interno due righe commentate che danno un esempio di come inserire il comando, nell'esempio sotto sono le righe 5 e 9.

```
 1. <?php
 2. if(!defined('checkaccess')){die('Direct access not permitted');}
 3. // Manage com. apps daemon as 'http' user if needed
 4. if (is_null($PID)) { // Stop Daemon
 5.         //$out = exec("com_daemon stop");
 6.         $out0 = exec("shelly_3_daemon stop");
 7.         $out1 = exec("solax_daemon stop");
 8. } else { //Start
 9.         //$out = exec("com_daemon start");
10.         $out0 = exec("shelly_3_daemon start");
11.         $out1 = exec("solax_daemon start");
12. }
13. ?>
```

come mostrato nell'esempio, dove si vuole far avviare due com_daemon, si dovranno inserire per ogni com_daemon una riga nella sezione “stop” e una nella sezione “start”, nellesempio sono le rige 6 e 7 per lo stop e 10 e 11 per lo start

Fonte:
**Wiki - Progetti**

Autore:
**Walter62**

Ultimo aggiornamento: **2026/06/20 15:31**