Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Studio di dispositivi attivi basati su nuovi materiali e tecnologie per applicazioni in convertitori SMPPT fotovoltaici: implementazione di algoritmi avanzati e funzioni ausiliarie a bordo del convertitore SMPPT per servizi di tracking e diagnostica energetica

F.G. Della Corte, R. Carotenuto, D. Iero,
M. Merenda, G. Adinolfi, G. Graditi

Report RdS/PAR2018/013

STUDIO DI DISPOSITIVI ATTIVI BASATI SU NUOVI MATERIALI E TECNOLOGIE PER APPLICAZIONI IN CONVERTITORI SMPPT FOTOVOLTAICI: IMPLEMENTAZIONE DI ALGORITMI AVANZATI E FUNZIONI AUSILIARIE A BORDO DEL CONVERTITORE SMPPT PER SERVIZI DI TRACJÌKING E DIAGNOSTICA ENERGETICA

F.G. Della Corte[1], R. Carotenuto[1], D. Iero[1], M. Merenda[1], G. Adinolfi[2], G. Graditi[2]
[1]Università degli Studi Mediterranea di Reggio Calabria - Dipartimento di Ingegneria dell'Informazione, delle Infrastrutture e dell'Energia Sostenibile (DIIES)
[1]ENEA

Dicembre 2018

Report Ricerca di Sistema Elettrico
Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2018
Area: Generazione di energia elettrica con basse emissioni di carbonio
Progetto: B.1.2 "Ricerca su tecnologie fotovoltaiche innovative"
Obiettivo: Sviluppo di tool di progettazione e studio di dispositivi attivi innovativi per convertitori smppt

Responsabile del Progetto: Paola DELLI VENERI, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Studio di dispositivi attivi basati su nuovi materiali e tecnologie per applicazioni in convertitori SMPPT fotovoltaici: implementazione di algoritmi avanzati e funzioni ausiliarie a bordo del convertitore SMPPT per servizi di tracking e diagnostica energetica".

Responsabile scientifico ENEA: Giorgio GRADITI

Responsabile scientifico Università Mediterranea: Francesco G. DELLA CORTE

Indice

Sommario

Durante la terza annualità della ricerca, in linea con le procedure e le soluzioni SMPPT individuate mediante la piattaforma “Pi.Con-RET” sviluppata presso il Centro Ricerche ENEA di Portici, sono stati progettati, realizzati e confrontati sperimentalmente, tre nuovi circuiti SMPPT, tra cui uno basato su MOSFET in silicio, uno su MOSFET in carburo di silicio, ed uno basato sui più innovativi HEMT (High electron mobility transistor) in nitruro di gallio (GaN), una tecnologia emergente che offre dispositivi commerciali con caratteristiche elettriche ai terminali potenzialmente molto utili per applicazioni in ambito fotovoltaico [1] .
Per quanto riguarda lo stadio di controllo del convertitore SMPPT, esso è stato progettato intorno ad un FPGA[2], che svolge prevalentemente la parte di calcolo, e a un microcontrollore [3] con ruolo di supporto dell’attività di calcolo, diagnostica e gestione delle comunicazioni. Attraverso l’abbinamento di queste due tecnologie è possibile la ricerca continua della migliore condizione operativa per il modulo, attraverso la regolazione dei duty-cycle di funzionamento dei MOSFET (o HEMT) di potenza. Sono state, inoltre, integrate funzioni basilari di comunicazione dedicate al monitoraggio dei parametri operativi.
Grazie alla notevole versatilità dei tre circuiti, intenzionalmente ideati e progettati come piattaforme open-hardware per consentire a di ricercatori o personale esperto di intervenire facilmente sui parametri operativi o sperimentare nuovi dispositivi, l’estensione della ricerca ha consentito l’implementazione di nuove funzionalità, senza che ciò richiedesse modifiche sostanziali del progetto dei circuiti stampati, bensì intervenendo quasi esclusivamente sul firmware del microcontrollore e sulla programmazione dell’FPGA.
In pratica, sono stati implementati nuovi algoritmi di inseguimento del Punto di Massima Potenza, più sofisticati rispetto all’algoritmo fondamentale adottato precedentemente. Sono state, inoltre, potenziate le funzionalità di comunicazione fra i convertitori, grazie alle quali è ora possibile la visualizzazione continua, su un server creato ad hoc, dello stato di ciascun convertitore, con produzione automatica di allarmi nei casi di funzionamento anomalo, ad esempio per raggiunti limiti di temperatura.

1. Introduzione

Le attività di ricerca del terzo anno di attività, descritte nel documento "*Realizzazione e caratterizzazione di un prototipo di convertitore SMPPT realizzato con l'integrazione di dispositivi switching basati su materiali innovativi*" [1], hanno compreso la progettazione, la realizzazione e la caratterizzazione, in termini di prestazioni elettriche, di tre convertitori SMPPT: uno realizzato utilizzando MOSFET in silicio come dispositivi switching, uno utilizzando MOSFET in SiC e, infine, uno basato su HEMT in GaN. In accordo ai risultati e alle soluzioni ottenuti mediante la piattaforma Pi.Con-RET, sviluppata presso il Centro Ricerche ENEA di Portici, sono stati selezionati tre SMPPT caratterizzati dagli stessi componenti di potenza (induttanza, capacità di filtro, etc) e la stessa frequenza di switching in modo da poter confrontare il comportamento e le prestazioni delle tre tecnologie switching. Il confronto fra queste soluzioni tecnologiche alternative costituisce un importante risultato e un utile strumento di indagine scientifica su cui basarsi per i futuri sviluppi della ricerca industriale e per lo sviluppo di nuovi circuiti per la conversione ottimale dell'energia solare.
In questo report sono illustrati i risultati delle attività di ricerca svolte nel corso del periodo di estensione della terza annualità. La ricerca prevede due obiettivi:

- lo studio di algoritmi di MPPT alternativi rispetto a quello fondamentale denominato "Perturba e Osserva" (P&O), inizialmente utilizzato per provare il corretto funzionamento dell'hardware progettato durante la terza annualità. In particolare si è optato per la selezione di algoritmi che potessero essere facilmente implementati nel circuito, senza richiedere sostanziali modifiche topologiche dello stesso;
- lo sviluppo di un server sul quale fosse presente un'interfaccia grafica basata su un sito web, con funzioni di diagnostica, in grado di comunicare, in tempo reale le condizioni operative di tutti i convertitori presenti sotto una determinata rete Wi-Fi, con la diramazione di allerte in caso di condizioni critiche (ad esempio in casoi di superamento della massima temperatura di funzionamento dei MOSFET del convertitore).

2. Specifiche generali

In letteratura esistono numerosi algoritmi per l'inseguimento del punto di massima potenza nel contesto fotovoltaico [4, 5]. Con riferimento all'attività di implementazione di algoritmi alternativi per l'inseguimento del Punto di Massima Potenza, è stato prioritariamente dato peso alla semplicità degli stessi, nonché alla loro affidabilità. Infatti, le risorse hardware disponibili nella scheda sono relativamente limitate, sia in termini di program memory del μC (64-Kbyte), numero di Look Up Table dell'FPGA (5.000), che in termini di velocità di calcolo, in particolar modo, del microcontrollore (frequenza massima 120MHz). Per altro, nonostante la varietà, in realtà la gran parte di tali algoritmi è riconducibile ad un numero limitato di schemi di riferimento. Fanno indubbiamente eccezione gli algoritmi di controllo ed ottimizzazione a più variabili, che però richiedono notevoli risorse per la loro efficace implementazione [6]. In definitiva, l'analisi svolta ha condotto all'individuazione di tre tecniche utilmente e agevolmente trasferibili nell'hardware dei convertitori DC-DC precedentemente realizzati. Esse sono: l'inseguimento basato sul calcolo della conduttanza incrementale, quello basato sul metodo Perturba-e-Osserva a tre punti, e, infine, quello basato sul metodo Perturba-e-Osserva a passo variabile.

Relativamente al sistema di comunicazione, la possibilità di che ogni convertitore SMPPT potesse dialogare con un nodo Master e con altri convertitori, utilizzando un ricetrasmettitore integrato a bassa potenza operante con protocolli di comunicazione standard Wi-Fi è stata ulteriormente potenziata per implementare funzionalità a livello di sistema e procedure di diagnostica del circuito, anche da remoto. In aggiunta, è stata prevista ed inserita la possibilità di modifica dell'algoritmo in real-time.

3. Nuovi algoritmi MPPT implementati sui convertitori

In questa sezione sono descritti gli algoritmi implementati sulle piattaforme hardware presentate nel report precedente, relativo alla terza annualità del progetto.

3.1 Algoritmo MPPT a Conduttanza Incrementale

L'algoritmo P&O è diffusamente usato per la sua facilità d'implementazione e la ridotta potenza computazionale richiesta. Esso presenta, però, problemi nell'inseguire variazioni veloci del punto di massimo. Se per esempio, in un pannello fotovoltaico, a seguito della variazione dell'irraggiamento si ha un incremento della potenza, il sistema crederà che la variazione sia dovuta al cambiamento del duty cycle e continuerà a muoversi nella stessa direzione, che in realtà potrebbe essere la direzione opposta alla posizione del nuovo punto di massimo. Il sistema quindi potrebbe impiegare un certo tempo per portarsi intorno al nuovo punto di massimo.
La tecnica della conduttanza incrementale [7, 8, 9] si basa sul fatto che, nel punto di massima potenza, la derivata della potenza rispetto alla tensione è pari a zero. L'algoritmo cerca il punto di massima potenza comparando la conduttanza incrementale di/dv e la conduttanza $-i/v$ finché la tensione non raggiunge il punto in cui la conduttanza incrementale è uguale alla conduttanza della sorgente; nel punto di massimo, infatti, si ha: $dp/dv=0 \implies di/dv=-I/V$. Quando il punto di lavoro del piano PV è sulla destra del MPP, allora $dp/dv<0$, quando invece il punto di lavoro è sulla sinistra del MPP, si ha $dp/dv>0$. Il segno della derivata indica la direzione di perturbazione che conduce al MPP. Attraverso tale algoritmo è, quindi, teoricamente possibile sapere quando il MPP è stato raggiunto e dunque non è necessaria alcuna azione di controllo, mentre nell'implementazione P&O il punto operativo oscilla continuamente intorno al MPP.
L'algoritmo è mostrato nel diagramma di flusso in Figura 1Figura 1; in esso le variazioni incrementali sono approssimate con $dI=I_{in}-I_{in\_prec,mppt}$ e $dV=V_{in}-V_{in\_prec,mppt}$ e, in base al confronto di questi risultati, viene variato il duty cycle, in modo che il punto di lavoro si sposti verso il punto di massimo. Nel punto di MPP, l'algoritmo non effettua alcuna azione di controllo e provvede solo a salvare i valori di tensione e corrente per il confronto nel ciclo successivo. Rispetto alla tecnica P&O, quest'algoritmo offre prestazioni maggiori con condizioni atmosferiche variabili, ma richiede un costo computazionale maggiore.
A posteriori, vengono applicati gli stessi controlli per limitare il duty cycle, già discussi nel caso dell'algoritmo P&O [1].

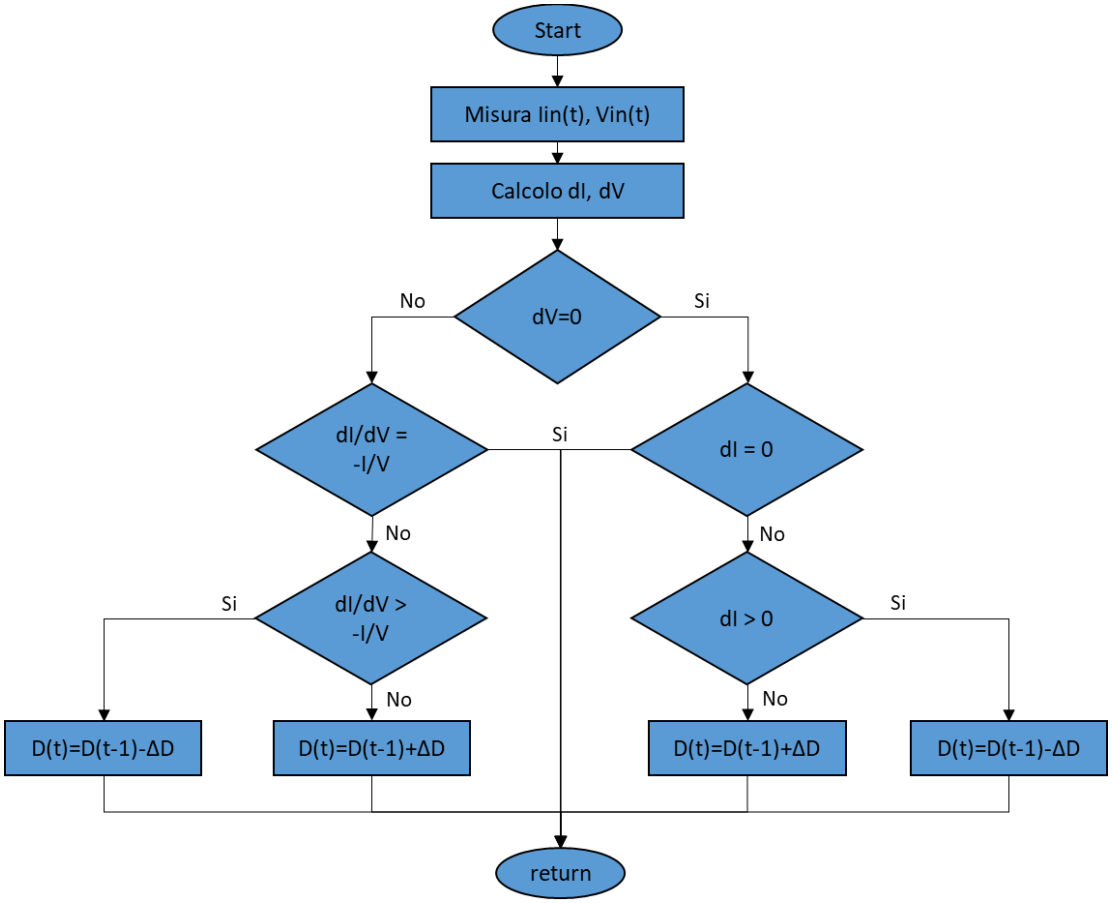

Figura 1. Diagramma di flusso algoritmo della conduttanza incrementale.

3.2 *Algoritmo MPPT P&O a tre punti*

L'algoritmo P&O a tre punti [10] consente di risolvere il problema della divergenza dell'algoritmo P&O in caso di rapide variazioni delle condizioni atmosferiche. Tale algoritmo viene implementato andando ad utilizzare tre punti di lavoro; oltre all'attuale punto operativo (A), vengono utilizzati nel confronto i due punti immediatamente a destra e sinistra (B, C). In questo modo è possibile osservare se siamo in presenza di un punto di massimo e se la variazione del valore di potenza è dovuta al cambiamento delle condizioni atmosferiche.
Se $P_B \geq P_A$ e $P_A > P_C$ si incrementerà il duty cycle, mentre se $P_B < P_A$ e $P_A \leq P_C$ allora il duty cycle verrà diminuito; negli altri casi non verranno effettuate variazioni poiché il cambio di potenza è dovuto ad una variazione dell'irraggiamento.
L'algoritmo implementato è mostrato nel diagramma di flusso in Figura 2;

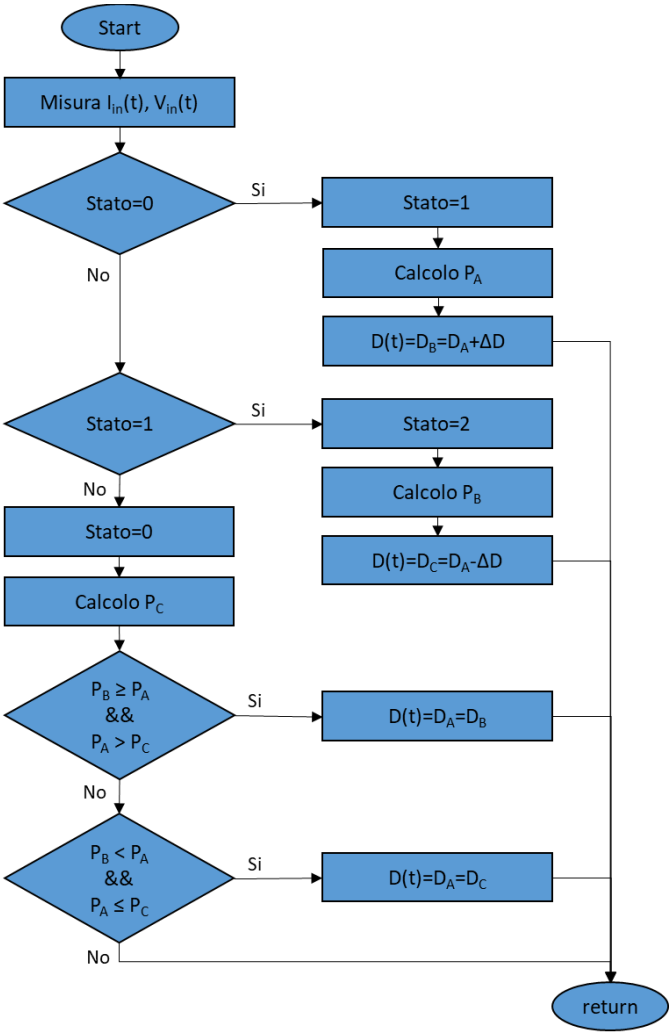

Figura 2. Diagramma di flusso algoritmo P&O a tre punti.

3.3 *Algoritmo MPPT P&O a passo variabile*

Per rendere più efficace l'algoritmo P&O, può essere implementato una variazione del duty cycle variabile in base al punto all'inclinazione della curva P-V [11]. La pendenza della curva, infatti, diminuisce via via che ci si avvicina al punto di massima potenza. Quando si è lontani dal MPP, l'inclinazione è alta e così dovrà essere anche lo step da compiere per avvicinarsi il più rapidamente possibile al massimo. Invece, vicino al MPP, lo step dovrà essere piccolo in modo da non allontanarsi il meno possibile dal punto di massimo. Per applicare l'algoritmo viene, quindi, calcolata una variazione del duty cycle proporzionale al coefficiente angolare della curva, con dei minimi e massimi opportuni per garantire il funzionamento dell'algoritmo.
L'algoritmo implementato è mostrato nel diagramma di flusso in Figura 3;

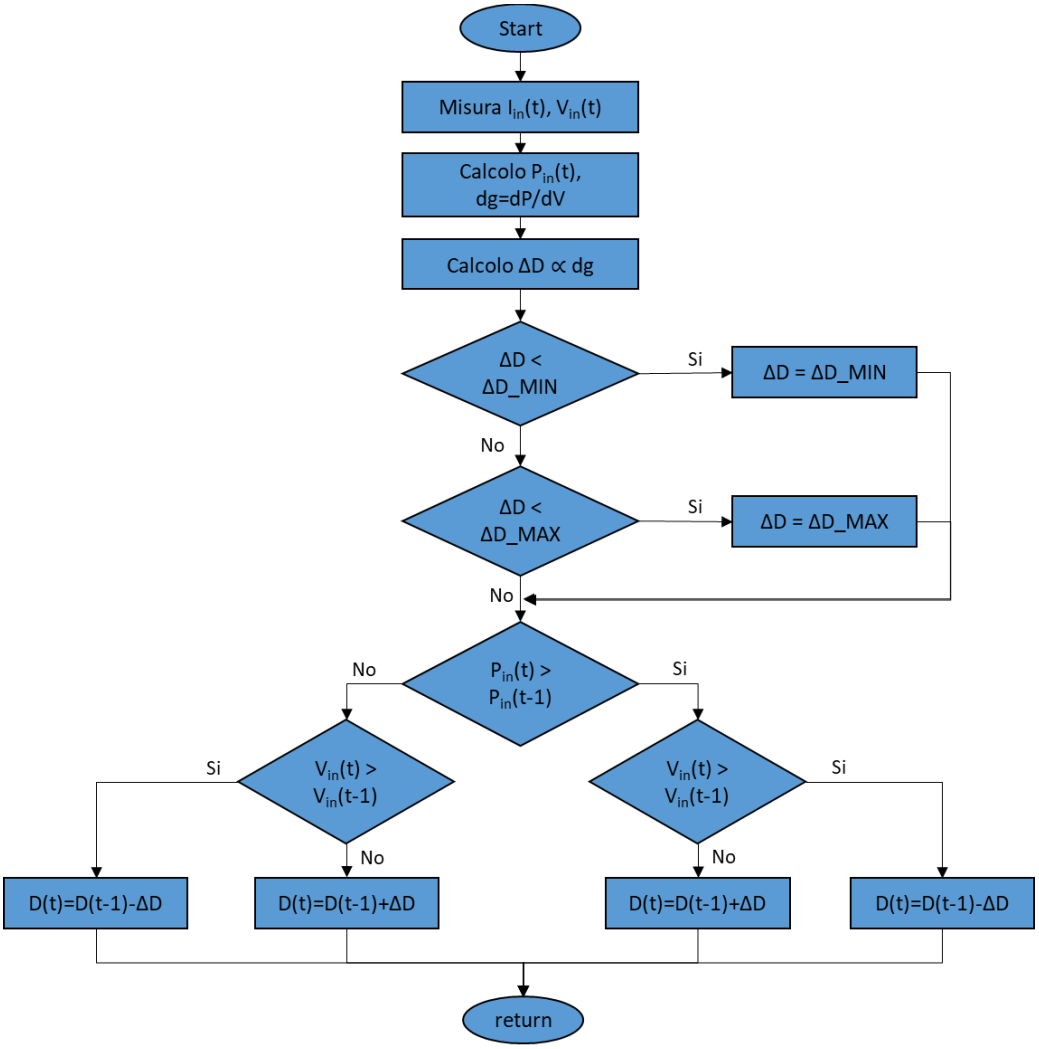

Figura 3. Diagramma di flusso algoritmo P&O a passo variabile.

3.4 *Implementazione degli algoritmi MPPT sul microcontrollore*

Di seguito è presentato un estratto del codice sorgente del microcontrollore relativo all'applicazione dell'algoritmo di MPPT. Il microcontrollore può applicare, in base allo stato di alcuni parametri, i seguenti metodi:

- Algoritmo P&O (standard);
- Algoritmo a conduttanza incrementale;

-
- Algoritmo P&O a tre punti;
- Algoritmo P&O a passo variabile;
- Algoritmo P&O con controllo della tensione di riferimento del DC-DC. In questo caso il microcontrollore calcola la tensione di riferimento e comunica il dato al chip FPGA che effettua il controllo DC-DC in modo da portarsi al valore di tensione fissato dall'algoritmo.

Estratto del codice implementato per gli algoritmi MPPT

```
[...]

Pin=(float)Vin*(float)Iin/100000.0;
Pout=(float)Vout*(float)Iout/100000.0;

if (forceDuty) {
        do_mppt(&dutyM1);
        outputParam(dutyM1, dutyM3, deadTime, freq, forceDuty, select_bootStrap);
} else {
        //algoritmo P&O (DC-DC su FPGA ON)
        if ((Pin>Pin_prec && Vin>Vin_prec_mppt)||(Pin<Pin_prec && Vin<Vin_prec_mppt))
                vref=vref-AUMENTO_VREF;
        else vref=vref+AUMENTO_VREF;
        //controllo min-max
        if (vref<VREF_MIN) vref=VREF_MIN;
        else if (vref>VREF_MAX) vref=VREF_MAX;
}
Pin_prec=Pin; Vin_prec_mppt=Vin; Iin_prec_mppt=Iin;

[...]

/**
 * Applica uno dei metodi MPPT, in base all valore della variabile "tipo_MPPT"
 */
inline void do_mppt() {
        int *duty;
        if (state_DCDC) {
                //buck mode
                duty = &dutyM1;
                dutyM3=0;
        } else {
                //boost mode
                duty = &dutyM3;
                dutyM1=4095;
        }

        if (tipo_MPPT==0) {
                //algoritmo P&O (DC-DC su FPGA OFF)
                mppt_po(duty);
        } else if (tipo_MPPT==1) {
                //algoritmo INC (DC-DC su FPGA OFF)
                mppt_inc(duty);
        } else if (tipo_MPPT==2) {
                //algoritmo P&O passo variabile (DC-DC su FPGA OFF)
                mppt_po_var(duty);
        } else if (tipo_MPPT==3) {
                //algoritmo P&O a tre punti (DC-DC su FPGA OFF)
                mppt_po_3pt(duty);
        } else {
                return;
        }

        //controllo min-max e transizione buck-boost
        if (state_DCDC) {
                //buck mode
                if (dutyM1<120){
                        dutyM1=120;
                } else if (dutyM1>4000) {
                        dutyM1=4095;
                        dutyM3=120;
                        state_DCDC=false; //switch to boost mode
                }
        } else {
```

```
                //boost mode
                if (dutyM3<120) {
                        state_DCDC=true; //switch to buck mode
                        dutyM3=0;
                        dutyM1=4000;
                } else if (dutyM3>4000){
                        dutyM3=4000;
                        dutyM1=4095;
                }
        }
}

/**
 * Metodo MPPT P&O
 * @param duty
 */
inline void mppt_po(int *duty) {
        if ((Pin>Pin_prec && Vin>Vin_prec_mppt)||(Pin<Pin_prec && Vin<Vin_prec_mppt))
                *duty=*duty-AUMENTO_DUTY;
        else *duty=*duty+AUMENTO_DUTY;
}

/**
 * Metodo MPPT P&O a passo variabile
 * @param duty
 */
inline void mppt_po_var(int *duty) {
        dp=Pin-Pin_prec;
        dv=((float)Vin-(float)Vin_prec_mppt)/100.0;
        dg=fabs(dp/dv);

        int passo_duty = MPPT_PO_VAR_COEFF * dg;
        if (passo_duty<AUMENTO_DUTY_MIN) {
                passo_duty=AUMENTO_DUTY_MIN;
        } else if (passo_duty>AUMENTO_DUTY_MAX){
                passo_duty=AUMENTO_DUTY_MAX;
        }

        if ((Pin>Pin_prec && Vin>Vin_prec_mppt)||(Pin<Pin_prec && Vin<Vin_prec_mppt))
                *duty=*duty-passo_duty;
        else *duty=*duty+passo_duty;
}


/* Variabili di supporto per l'algoritmo MPPT P&O a tre punti */
uint8_t mppt_po_3pt_state = 0;
float Pa=0, Pb=0, Pc=0;
int dutyA=0, dutyB=0, dutyC=0;

/**
 * Metodo MPPT P&O a tre punti
 * @param duty
 */
inline void mppt_po_3pt(int *duty) {
        if (mppt_po_3pt_state==0) {
                mppt_po_3pt_state=1;
                Pa = Pin;
                dutyB = dutyA + AUMENTO_DUTY;
                *duty = dutyB;
        } else if (mppt_po_3pt_state==1) {
                mppt_po_3pt_state=2;
                Pb = Pin;
                dutyC = dutyA - AUMENTO_DUTY;
                *duty = dutyC;
        } else {
                mppt_po_3pt_state=0;
                Pc = Pin;
                if (Pb>=Pa && Pa>Pc) {
                        dutyA = dutyB;
                } else if (Pb<Pa && Pa<=Pc) {
                        dutyA = dutyC;
                }
                *duty = dutyA;
        }
}
```

```
/**
 * Metodo MPPT conduttanza incrementale
 * @param duty
 */
inline void mppt_inc(int *duty) {
        dv=((float)Vin-(float)Vin_prec_mppt)/100.0;
        di=((float)Iin-(float)Iin_prec_mppt)/1000.0;

        dg=di/dv;
        g=Iin/Vin;
        dgg=dg+g;

        dve=fabs(dv/Vmedia);
        die=fabs(di/Imedia);
        dge=fabs(dgg/g);

        if (dve<TOLN) {        //dv=0
                if (die<TOLN) {                //di=0
                        //lascia duty invariat
                } else if (di>0) {     //di>0
                        *duty=*duty-AUMENTO_DUTY;
                } else {                       //di<0
                        *duty=*duty+AUMENTO_DUTY;
                }
        } else {               //dv!=0
                if (dge<TOLN) {                //dg=-g
                        //lascia duty invariato
                } else if (dgg>0) {    //dg>-g
                        *duty=*duty-AUMENTO_DUTY;
                } else {                       //dg<-g
                        *duty=*duty+AUMENTO_DUTY;
                }
        }
}
```

4. HUB e interfaccia grafica

In questa sezione sono descritte le attività relative all'implementazioni degli algoritmi per la raccolta delle informazioni dagli SMPPT collegati alla medesima sottorete e della rappresentazione grafica delle informazioni acquisite.
Il sistema realizzato di "concentratore" permette l'individuazione dei parametri operativi che garantiscono la massima produzione/efficienza, quali la migliore frequenza di commutazione, il miglior duty-cycle e il massimo dead time dei MOSFET, e provvederà, successivamente, a "suggerire" i parametri suddetti a tutti i sistemi registrati sotto la stessa rete wireless.

4.1 Hub

Lo scopo dell'attività è stato la realizzazione di un hub, un concentratore, sviluppato in linguaggio Python, che permetta di:

- Rilevare autonomamente e periodicamente gli SMPPT collegati alla stessa sottorete
- Creare l'elenco dei dispositivi rilevati, sotto forma di file JSON, con le loro caratteristiche identificative (ID)
- Interrogare, con periodicità di 30s, i dispositivi compresi nell'elenco, richiedendo le informazioni di produzione e conversione della potenza (status e settings)
- Applicare un algoritmo di ricerca del massimo di potenza prodotta e del massimo di efficienza
- Rimappare/riconfigurare i dispositivi con minore produzione, effettuando la trasmissione dei coefficienti di funzionamento del sistema che ha prestazioni maggiori rispetto agli altri sistemi, per mezzo di apposita funzione di scrittura compresa nell'API del sistema.

Il sistema è implementato su un Single Board Computer di tipo Raspberry Pi 3 b+, dotata nativamente di connettività Wi-Fi.

Il codice sviluppato è basato sull'esecuzione di quattro funzioni.

1. Scansiona
2. Acquisisci
3. Trova Massimo
4. Rimappa

La funzione "Scansiona" rileva periodicamente gli SMPPT presenti su una sottorete del tipo 192.168.1.xxx, inserendoli in una lista JSON.

La funzione "Acquisisci" scorre la lista effettuando le richieste previste nelle API per acquisire status e setting dei dispositivi collegati, salvando tutte le informazioni ricevute in un file CSV.
Il file CSV ha come nome file il timestamp di inizio acquisizione, un marcatore temporale che permette di ottenere il riferimento del tempo di acquisizione di ogni singola misura, mentre per riga sono salvate le informazioni per ogni singolo SMPPT, ovvero IP e i risultati delle chiamate a Status e Settings.

La funzione "Trova Massimo" scorre tutti gli SMPPT salvati nel file CSV, calcolando per ognuno la Potenza in uscita Pout e la Potenza in ingresso Pin, e memorizzando frequenza, dead time, e duty cycle dell'SMPPT con valore di Pout maggiore.
Su questo aspetto, possono essere stabilite differenti politiche di massimizzazione.

La funzione "Rimappa" richiama l'API POST e modifica frequenza, dead time, e duty cycle degli *n-1* SMPPT che hanno Pout inferiore all'SMPPT che è stato selezionato dall'algoritmo come riferimento.

I valori ottenuti da ogni singola acquisizione, sono salvati in un file CSV che contiene le informazioni acquisite. Ciò permette la realizzazione di un'interfaccia avanzata di visualizzazione delle informazioni.

4.2 Implementazione del software dell'hub su Single Board computer

Di seguito è presentato un estratto del codice sorgente in linguaggio Python relativo all'implementazione del software dell'hub su Single Board computer di tipo Raspberry. Nel seguito viene riportato un estratto del software sviluppato.

Estratto del codice implementato

```
import os, glob, time, operator
import subprocess
import requests
import csv
import sys
import json
import shutil
import traceback
import socket
from datetime import datetime


global P_max
P_max = 0
global freq_opt
freq_opt=0
global dt_opt
dt_opt=0
global m1_opt
m1_opt=0
global m3_opt
m3_opt=0
global ip_max
ip_max=0

timeout=1

socket_obj = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

net1 = "192.168.1" #indirizzo sottorete
net2 = net1.split('.')
a = '.'
net3 = net2[0] + a + net2[1] + a + net2[2] + a
stn1 = 1
edn1 = 254
edn1 = edn1 + 1

num_acquis=1 # numero scansioni

from datetime import datetime


path_save= "/home/pi/CSV/"#percorso in cui vengono salvati i file creati

def scan():
    data={'ip_address':[] }
    for ip in range(stn1,edn1):
        addr = net3+str(ip)
        res = socket_obj.connect_ex((addr,80))
        if (res==0):
            print (addr, "this address is live")
            data['ip_address'].append({'ip' : ip})
    return data
#
```

```
def get_status(payload):
    url_prefix = "http://192.168.1."
    IP=str(payload)
    req= "/api/smppt/status"
    url = url_prefix+IP+req
    print(url)
    headers = {'content-type': "application/json"}
    try:
        response = requests.request("GET", url, headers=headers, timeout=timeout)
    except Exception as exc1:
        print (exc1)
        pass
    return response.text

def get_settings(payload):
    url_prefix = "http://192.168.1."
    IP=str(payload)
    req= "/api/smppt/settings"
    url = url_prefix+IP+req
    print(url)
    headers = {'content-type': "application/json"}
    try:
        response = requests.request("GET", url, headers=headers, timeout=timeout)
    except Exception as exc1:
        print (exc1)
        pass
    return response.text

def post_settings(payload, item, value):

    url_prefix = "http://192.168.1."
    IP=str(payload)
    req= "/api/smppt/settings/"
    url = url_prefix+IP+req+str(item)
    headers = {'content-type': "application/json"}
    try:
        response = requests.request("POST", url, data=str(value), headers=headers, timeout=timeout)
    except Exception as exc1:
        print (exc1)
        pass
    print(response.ok)
    return response.ok
#
def smppt_import():
    try:
        startTs = time.time()
        for i in range(num_acquis):
            errCnt = 0
            now = time.time()
            f.write('{}'.format(now))
            f.write('{} '.format(','))
            for x in data['ip_address']:
                print(x['ip'])
                try:
                    resp=get_status(x['ip'])
                    print (resp)
                    json_v = json.loads(resp)
                except Exception as exc:
                    errCnt += 1
                    tb = traceback.format_exc()
                    print ('Error GET Status')


                if errCnt < 1:
                    print("ok")
                    f.write('{}'.format(x['ip']))
                    f.write('{}'.format(','))
                    f.write('{}'.format(json_v['mac']))
                    f.write('{}'.format(','))
                    f.write('{}'.format(json_v['mode']))
                    f.write('{}'.format(','))
```

```
                f.write('{}'.format(json_v['temp1']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['temp2']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['meas']['Vin']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['meas']['Vout']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['meas']['Iin']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['meas']['Iout']))

            else:
                f.write('{}'.format(x['ip']))
                f.write('{}'.format(','))
                f.write('Error')
                f.write('{}'.format(','))
                print ('Too many errors GET Status')

            try:
                resp=get_settings(x['ip'])
                print (resp)
                json_v = json.loads(resp)

            except Exception as exc:
                errCnt += 1
                tb = traceback.format_exc()
                print ('Error GET Settings')

            if errCnt < 1:
                print("ok")
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['mppt']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['algo']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['force']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['freq']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['dt']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['m1']))
                f.write('{}'.format(','))
                f.write('{}'.format(json_v['m3']))

            else:
                f.write('Error ')
                print ('Too many errors GET Settings')

        f.write('\r\n')
    stopTs = time.time()
    timeDiff = stopTs  - startTs
    print ("Time %.3f [s]" % (timeDiff))


  except (ValueError, KeyError, TypeError):
    print ("JSON format error")
#
#
#
#
def acquisisci():
    now = datetime.now().strftime("%Y%m%d_%H%M%S")
    temp = "Test " + str(now)
    print(temp)
    global filename
    filename= path_save + str(now) + ".csv"
    global f
    f= open(filename,"w+")
    f.write('{} '.format('Timestamp,ip,mac,mode,temp1,temp2,Vin,Vout,Iin,Iout,mppt,algo,force,freq,dt,m1,m3'))
```

```
        f.write('\r\n')
        smppt_import()
        f.close()
#
def trova_max():

    with open(filename, newline='') as csvfile:
        reader = csv.DictReader(csvfile)
        P_max=-1
        try:
            for row in reader:

                #print(row)
                Vout=int(row['Vout'])
                Vin=int(row['Vin'])
                Iout=int(row['Iout'])
                Iin=int(row['Iin'])
                Pin=Vin*Iin
                Pout=Vout*Iout
                print("Pmax_init",P_max)
                print("Pout", Pout)
                print("Pin",Pin)
                if (Pout > P_max):
                    P_max=Pout
                    global freq_opt
                    freq_opt=int(row['freq'])
                    global dt_opt
                    dt_opt=int(row['dt'])
                    global m1_opt
                    m1_opt=row['m1']
                    global m3_opt
                    m3_opt=row['m3 ']
                    global ip_max
                    ip_max = int(row['ip'])



        except csv.Error as e:
            sys.exit('file {}, line {}: {}'.format(filename, reader.line_num, e))
#
def rimappa():
    errCnt=0
    for x in data['ip_address']:
        print(data['ip_address'])
        if (x['ip'] != ip_max):
            try:
                resp=post_settings(x['ip'], "freq", freq_opt)

            except Exception as exc:
                errCnt += 1
                tb = traceback.format_exc()
                print ('Error POST freq')

            if (errCnt < 1):
                print("ok POST freq")
            else:
                print ('Too many errors POST freq')

            try:
                resp=post_settings(x['ip'], "dt", dt_opt)

            except Exception as exc:
                errCnt += 1
                tb = traceback.format_exc()
                print ('Error POST dt')

            if errCnt < 1:
                print("ok POST dt")
            else:
                print ('Too many errors POST dt')
```

```
            try:

                resp=post_settings(x['ip'], "m1", m1_opt)
                #print (resp)

            except Exception as exc:
                errCnt += 1
                tb = traceback.format_exc()
                print ('Error POST m1')

            if errCnt < 1:
                print("ok POST m1")
            else:
                print ('Too many errors POST m1')

            try:

                resp=post_settings(x['ip'], "m3", m3_opt)

            except Exception as exc:
                errCnt += 1
                tb = traceback.format_exc()
                print ('Error POST m3')

            if errCnt < 1:
                print("ok POST m3")
            else:
                print ('Too many errors POST m3')
        else:
            print("Only an SMPPT is on the subnet")


while (1):
    data = scan()
    list=data['ip_address']
    print((data['ip_address']))
    try:
        if (data['ip_address'])[0].__contains__('ip'):
            acquisisci()

            trova_max()

            rimappa()
            print("Complete")
    except Exception as exc:
        print("False")


    time.sleep(30)
```

4.3 Implementazione dell'interfaccia grafica

Per ogni SMPPT, i valori ottenuti da ogni singola acquisizione, sono memorizzati in un file CSV che contiene le informazioni acquisite, oltre alla marcatura temporale della singola acquisizione. Ciò permette di avere a disposizione una struttura dati che permette la realizzazione dinamica di un'interfaccia avanzata di visualizzazione delle informazioni acquisite.
I file presenti nell'apposita cartella in cui opera il programma, sono scansionati periodicamente ed i dati in essi contenuti sono inseriti in una struttura dati più adatta alla successiva rappresentazione grafica, con possibilità di selezione delle informazioni visualizzate, e inviati alla dashboard di visualizzazione.

4.4 Implementazione del software per l'interfaccia grafica su Single Board computer

Di seguito è presentato un estratto del codice sorgente in linguaggio Python relativo all'implementazione dell'interfaccia grafica su Single Board computer di tipo Raspberry. In base a detto codice, sono implementate le funzioni riportate nel paragrafo §4.3

```
import os, glob, time, operator
import subprocess
import requests
import csv
import sys
import json
import shutil
import traceback
from scapy.layers.pptp import PPTPOutgoingCallReply
import socket
from datetime import datetime

file_path= "/home/pi/*"

def get_oldest_file(files, _invert=False):
    """ Find and return the oldest file of input file names.
    Only one wins tie. Values based on time distance from present.
    Use of `_invert` inverts logic to make this a youngest routine,
    to be used more clearly via `get_youngest_file`.
    """
    gt = operator.lt if _invert else operator.gt
    # Check for empty list.
    if not files:
        print("Empty")
        return None
    # Raw epoch distance.
    now = time.time()
    # Select first as arbitrary sentinel file, storing name and age.
    oldest = files[0], now - os.path.getctime(files[0])
    # Iterate over all remaining files.
    for f in files[1:]:
        age = now - os.path.getctime(f)
        if gt(age, oldest[1]):
            # Set new oldest.
            oldest = f, age
    # Return just the name of oldest file.
    return oldest[0]

def get_youngest_file(files):
    return get_oldest_file(files, _invert=True)

def check_folder_size(path,sizeLimit):
    if not path:
        path=file_path + "sent_Data"
```

```
    if not sizeLimit:
        sizeLimit=2
    size = subprocess.check_output(['du','-sh', path]).split()[0] #.decode('utf-8')
    unit= size[-1:]
    size= size[:-1]
    size=size.replace(",",".")
    if (unit == "G" and float(size) > 1):
        return True
    else:
        return False


while (1):
    files = glob.glob(file_path)
    print(files)
    filePath = get_oldest_file(files)
    print(filePath)
    if files:

        if os.stat(filePath).st_size == 0:
            print ("File empty")
            time.sleep(5)


        f = open(filePath, 'rb')
        #reader = csv.reader(f)
        print(f)
        reader = csv.DictReader(f)

        print(reader)
        for row in reader:

            Vout=int(row['Vout'])
            Vin=int(row['Vin'])
            Iout=int(row['Iout'])
            Iin=int(row['Iin'])
            Pin=Vin*Iin
            Pout=Vout*Iout

            aa="[{\"timestamp\":\""
            b=row['Timestamp']
            bb="}, {\"mac\":"
            c=row['mac']
            cc="}, {\"ip\":"
            d=row['ip']
            dd="}, {\"temperature_1\":"
            e=row['temp1']
            ee="}, {\"temperature_2\":"
            f=row['temp2']
            ff="}, {\"Pout\":"
            g=Pout
            gg="}, {\"Pin\":"
            h=Pin
            hh="}, {\"freq\":"
            i=row['freq']
            ii="}, {\"dt\":"
            l=row['dt']
            ll="}, {\"m1\":"
            m=row['m1']
            mm="}, {\"m3\":"
            n=row['m3']
            nn="}]"

            payload =  aa + str(b) + bb + str(c)+cc + str(d)+dd+ str(e)+ee  + str(f) + ff + str(g)+gg + str(h)+hh+ str(i)+ii+ str(l)
+ ll + str(m)+mm + str(n)+nn

            url = "https://demo.thingsboard.io/api/v1/yYOYoI9IYt6veSWT63jg/telemetry"
```

```
        headers = {
            'cache-control': "no-cache",
            }
        response = requests.request("POST", url, data=payload, headers=headers)
        print(response.text)


    fileToMove= filePath.split('/',4)
    print ("moving file :"+fileToMove [4])
    pathToMove= file_path + "home/pi/CSV_elaborated/"+fileToMove[4]
    shutil.move(filePath,pathToMove)
    #check if the folder '/home/pi/CSV_elaborated//*' is bigger than 2G
    if check_folder_size(None,None):
        #if bigger then X delete the n older files
        n=2
        for i in range (0,n):
            path=glob.glob("home/pi/CSV_elaborated/*")
            fileToDel= get_oldest_file(path)
            print (fileToDel)
            os.remove(fileToDel)
    else:
        print ("Error")
else:
    print ("No file")
```

Nella seguente figura viene mostrata l'interfaccia grafica sviluppata.

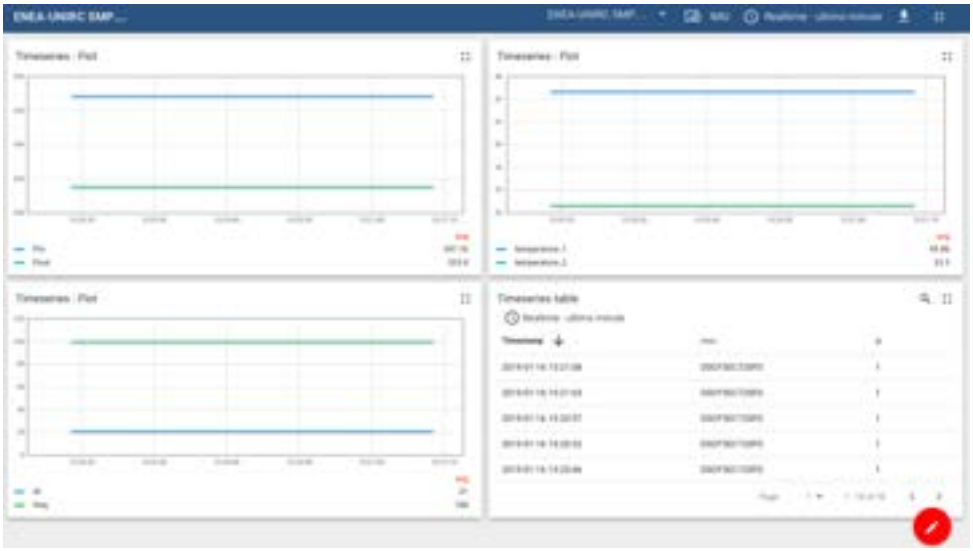

Figura 4. Interfaccia Grafica

5. Conclusioni

Durante il PAR 2018 di progetto sono stati implementati nuovi algoritmi di inseguimento del Punto di Massima Potenza, più sofisticati rispetto all'algoritmo fondamentale adottato precedentemente. Sono state potenziate le funzionalità di comunicazione fra i convertitori, grazie alle quali è ora possibile la visualizzazione continua, su un server creato ad hoc, dello stato di ciascun convertitore, con produzione automatica di allarmi nei casi di funzionamento anomalo, ad esempio per raggiunti limiti di temperatura.

Appendice A

Application Programming Interface (API) per la comunicazione Wi-Fi.

Funzioni di lettura

SMPPT read status

(GET /api/smppt/status)

Test:

```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" -X GET
http://192.168.1.xxx/api/smppt/status
```

Risposta in formato JSON

```
{
 "mac": "xxxxxxxxxxx",
 "mode": [int, bit0=BOOST, bit1=BUCK, bit2=BYPASS],
 "temp1": [signed int, C x 100],
 "temp2": [signed int, C x 100],
 "meas": {
  "Vin": [int, volt x 100],
  "Vout": [int, volt x 100],
  "Iin": [int, ampere x 1000],
  "Iout": [int, ampere x 1000]
 }
}
```

Es. Risposta

```
{
 " mac": "BC184E454DCA",
 "mode": 1,
 "temp1": 2782
 "temp1": 2890
 "meas": {
  "Vin": 2905,
  "Vout": 6543,
  "Iin": 8534,
  "Iout": 3668
 }
}
```

SMPPT read settings

(GET /api/smppt/settings)

Test:

```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" -X GET
http://192.168.1.xxx/api/smppt/settings
```

Risposta in formato JSON

```
{
 "mppt": [int, 0-1 -> 0=dc-dc, 1=mppt],
 "algo": [int, 0-3 -> 0=P&O, 1=INC, 2=P&O variable step, 3=P&O 3-points],
 "force": [int, 0-1 -> 0=calculated duty, 1=force manual duty],
 "freq": [int, 0-7 -> {100,80,120,150,175,200,225,250} kHz],
 "dt": [int, 0-15 -> {312,83,104,125,145,166,187,208,229,250,271,291,312,333,354,375} ns],
 "m1": [int, 0-4095],
 "m3": [int, 0-4095]
}
```

Es. Risposta

```
{
 "mppt": 1,
 "algo": 0,
 "force": 0,
 "freq": 0,
 "dt": 2,
 "m1": 120,
 "m3": 0
}
```

Funzioni di scrittura:

SMPPT write settings

(POST /api/smppt/settings/item)

Test:

```
curl --data "value" http://192.168.1.xxx/api/smppt/settings/item
```

The value to write should be put in the body of the request.

"item" can be:
mppt MPPT ON/OFF [0-1] -> 0=dc-dc, 1=mppt
freq switching frequency [0-7] -> {100,80,120,150,175,200,225,250} kHz
dt dead time [0-15] -> {312,83,104,125,145,166,187,208,229,250,271,291,312,333,354,375} ns
m1 duty cycle MOSFET M1 [0-4095]
m3 duty cycle MOSFET M3 [0-4095]
algo algorithm [0-3] -> {P&O, INC, P&O variable step, P&O 3-points }

Riferimenti bibliografici

[1] F.G. Della Corte et al., “Realizzazione e caratterizzazione di un prototipo di convertitore SMPPT realizzato con l'integrazione di dispositivi switching basati su materiali innovativi“, Report RdS/PAR2017.

[2] Lattice Semiconductor, 2017 iCE40 UltraPlus TM Family Datasheet.

[3] Microchip Technology Inc., 2017. dsPIC33EPXXGS50X FAMILY Datasheet [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70005127d.pdf.

[4] V. Salas, E. Olias, A. Barrado, A. Lazaro, “Review of the maximum power point tracking algorithms for stand-alone photovoltaic systems”, Solar Energy Materials and Solar Cells, vol. 90, no. 11, 2006, pp. 1555-1578.

[5] M.J. Khan, P. Shukla, R. Mustafa, S. Chatterji, L. Matew, “Different types of maximum power point tracking techniques for renewable energy systems: A survey”, *AIP Conference Proceedings* 1715, 020015, 2016.

[6] L. Yi, B. Wang, J. Liu, J. Liu, “Decoupling control of three-phase Z-source photovoltaic grid-connected inverter based on asymmetrical multi-variable PID neural network”, *Acta Energiae Solaris Sinica*, vol. 34, 2013, pp. 1612-1618.

[7] K.H. Hussein, I. Muta, T.Hoshino, M. Osakada, "Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions," *IEE Proc.-Gener, transm. Distrib* vol.142, no.1, .pp.59-64, Jan 1995.

[8] Yeong-Chau Kuo, Tsorng-Juu Liang, Jiann-Fuh Chen, "Novel maximum-power-point-tracking controller for photovoltaic energy conversion system," *IEEE Trans. on Industrial Electronics*, vol. 48, no. 3, pp. 594-601, June 2001.

[9] Fangrui Liu, Shanxu Duan, Fei Liu, Bangyin Liu, and Yong Kang, "A variable step size INC MPPT method for PV systems", *IEEE Trans. on Industrial Electronics*, vol. 55, no. 7, pp.2622~2628, July 2008.

[10] J.A.Jiang et. Al. , "Maximum Power Tracking for Photovoltaic Power Systems," Tamkang Journal of Science and Engineering, Vol. 8, No. 2, pp. 147-153, 2005.

[11] A. Al-Diab and C. Sourkounis, "Variable step size P&O MPPT algorithm for PV systems," *12th International Conference on Optimization of Electrical and Electronic Equipment*, Basov, 2010, pp. 1097-1102.