Advanced control for renewable energy systems

Cristian Miron

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI **UNIVERSITÉ DE LILLE1**

Advanced control for renewable energy systems

PHD THESIS – Co-Tutelle

6th November 2018

Author: Cristian MIRON

Co-director: Prof. Nicolai CHRISTOV (University of Lille)

Co-director: Prof. Dumitru POPESCU (Polytechnic of Bucarest)

Supervisor: Prof. Abdel AITOUCHE (Hautes Etudes d'Ingénieur)

Reporters: Prof. Didier MAQUIN (University of Lorraine)

Prof. Mircea IVĂNESCU (University of Craiova)

Acknowledgments

I would like to express my gratitude and appreciation towards my guiding teachers, towards the members and the president of the jury.

I am deeply indebted to Professor Dumitru Popescu for his constant and genuine support along this PhD thesis.

I am profoundly grateful to Professor Nicolaï Christov, without whom this thesis would not have possible.

My sincere appreciation goes to Professor Abdel Aitouche for his involvement and all the provided help.

I would also like to thank Professors Dan Ștefănoiu, Cătălin Petrescu and Ciprian Lupu for their precious advises.

A special thanks to my colleague and friend, Olteanu Severus, with whom i had the pleasure to work with.

I would like to thank Cătălin Dimon, Valentin Tănasă, Tudor-Bogdan Airimițoaie and all of my colleagues for their support and kidness.

I thank my family and friends for being there for me in the difficult moments.

I dedicate this work to my father and my grandfather, my mother and my grandmother. I thank them for all the time and loving they invested in me.

GLOSSARY

PV	Photovoltaic Panel
P&O algorithm	Perturb and Observe algorithm
IC algorithm	Incremental Conductance algorithm
MPPT	Maximum Power Point Tracking
T-S observer	Takagi Sugeno observer
R-S-T	Polynomial R-S-T control
SOC	State Of Charge
DC/DC	Direct Current to Direct Current
GUI	Graphic User Interface
IoT	Internet of Things

LIST OF FIGURES

LIST OF DEVELOPED SOFTWARE

1. Graphic User Interface for calibrating the model of a Photovoltaic Panel
2. MPPT tracking control algorithm on Arduino Uno
3. Battery charging supervisor on Arduino Uno
4. Optimization control supervisor | SCADA system on Raspberry Pi 3
5. Polynomial robust control algorithm on Arduino Due
6. Takagi Sugeno Observer model in Simulink
7. Supervisory system – fuel cell case study model in Simulink

LIST OF PUBLICATIONS

1. C. Miron, D. Popescu, C. Petrescu, Designing control systems with distributed parameters, 2014 18th International Conference on System Theory, Control and Computing (ICSTCC)
2. C. Miron, D. Popescu,A. Aitouche and N. Christov, “Observer based control for a PV system modeled by a Fuzzy Takagi Sugeno model”, in 2015 System Theory, Control and Computing (ICSTCC), 2015 19th International Conference, p. 652-657.
3. C. Miron, N. Christov, S.Olteanu, Energy management of photovoltaic systems using fuel cells, 2016 20th International Conference on System Theory, Control and Computing (ICSTCC)
4. C. Miron, S Frigioiu, S. Olteanu, A. Aitouche, N. Christov, Control Architecture for Low Power Photovoltaic MPPT Modules, 14th European Workshop on Advanced control and Diagnosis
5. S.Olteanu, C. Torous, C. Miron, Model based MPPT control of a small photovoltaic panel, 2017 21st International Conference on System Theory, Control and Computing (ICSTCC)
6. C. Miron, D. Popescu, I. Arsu, Control systems for a heat exchanger with distributed parameters, 2017 21st International Conference on System Theory, Control and Computing (ICSTCC)
7. C. Miron, S.Olteanu, N. Christov, A. Aitouche, Architecture for Embedded Supervisory System of Distributed Renewable Energy Sources, CFP 7th International Conference on Systems and Control, 24-26 October 2018, Valencia, Spain
8. ENERGYLIFE- POSCCE 486/2013 project-Digital Systems for Hybrid Energetic Installations Control
9. SINTELPV -PED 228/2017 project- Intelligent System for Extremal Control of PV panels with Variable Orientation.
10. C. Miron, S.Olteanu, N. Christov, D. Popescu, Advanced control system for photovoltaic energy generation, storage and consumption, Scientific Bulletin UPB, Series A, Applied Mathematics and Physics, 2018

CONTENTS

1. INTRODUCTION

Nowadays renewable energy is a long term solution for replacing the conventional sources of energy. It is known that the sun provides our planet enough energy to sustain the modern life style of its inhabitants. Many countries reoriented their policies regarding the production of energy and embraced the production of "green energy". The use of photovoltaic (PV) arrays and wind turbines has become very popular.

Nevertheless this "free energy" arises new challenges. Some of the big inconveniences of these alternatives are represented by a low conversion rate of the energy and the necessity of using an energy storing system. Another drawback is the reduced transfer efficiency between the PV arrays or/and wind turbines and the consumers.

Since the energy provided by the sun should be sufficient to cover all the world's energy consumption, if captured and stored at reasonable costs, the use of photovoltaic panels will have an increased popularity in time. The "green energy" sector comes with new challenges such as low conversion rates, additional energy storage systems and transfer inefficiency between the PV array and its connected load.

Photovoltaic power stations up to 500kw both in isolated areas and in urban zones will have an exponential development, as many countries reorient their policies regarding the production of energy embracing the use of "green energy". As such, the global tendency is to promote the concepts of energy optimization and energy independence through renewable sources, and among these, the photovoltaic panels are the most favorable. In this sense, the European Union has set the directive 2010/31/UE, in which, by 2021, all new building should be "nearly zero-energy buildings", suggesting that a significant amount of energy should be covered by renewable sources, local or nearby.

Because PV panels still have a reduced conversion rate, a strong and fast power variation and a wide geographical distribution of PV generators, it is implicitly obvious that an optimal energy management is essential. There are two important solutions for achieving this: the first one consists in the construction of a large energy grid, spread on geographical regions with variable power generation conditions, whereas the second approach implies the use of Smart Storage solutions [7]. Three main types of storage exist: Mechanical storage, as for example: water pumping systems (ex: storage by water pumping in Ludington: 110m, 1.87 GW, 15h, 27 million kWh); battery based solutions with different types of batteries used, each with its own advantages and disadvantages [8],[13]; and, finally, hydrogen based storage, bringing a high energy density, good conversion efficiency and physical robustness [10].

The goal of this thesis is to present and compare different control strategies for systems that are powered by renewable sources of energy. A prototype for testing purposes was designed.

This thesis treats different aspects such as PV panel modelling, buck converter modelling, building a non-linear observer, a control algorithm based on maximum power point tracking (MPPT),a polynomial control algorithm, the stability of the system.

Chapter 2 presents different photovoltaic cell models that can be further used in control loops. A graphic user interface is created for facilitating the computation of certain parameters and of the power-voltage / current-voltage characteristics of a PV panel. Furthermore a state space model and a transfer function model of some DC/DC converters are presented.

Chapter 3 focuses on elaborating a Takagi-Sugeno (T-S) observer which will provide the estimated voltage of the PV panel. The latter will later be used in the control block or it can serve for diagnosis purposes.

Chapter 4 compares different classical MPPT algorithms, as well as advanced control algorithms which may be later used to improve the performances of the control loops. A case study on a supervisory control that uses fuel cells is proposed.

Chapter 5 is oriented on a rather practical approach. It presents a distributed control system that is managed via an OPC server. A data acquisition system stores the data sent by each of the control loops and is able to plot data in real time.

Chapter 6 is dedicated to the conclusions.

Chapter 7 presents the code of the developed software and some schematics that were used during simulations.

Chapter 8 lists the bibliography.

De nos jours, l'énergie renouvelable est une solution durable pour remplacer les sources conventionnelles d'énergie. Il est connu que le soleil fournisse à notre planète assez d'énergie pour subvenir aux besoins du mode de vie moderne de ses habitants. Beaucoup de pays ont orienté leurs politiques considérant la production d'énergie et ont adopté la production de « l'énergie verte ». L'utilisation de réseaux photovoltaïques (PV) et d'éoliennes est devenue très populaire.

Cependant, de cette énergie gratuite découle de nouveaux défis. Certains des grands inconvénients de ces alternatives résident dans l'existence d'un faible de taux de conversion de l'énergie et la nécessité de devoir recourir à un système de stockage de l'énergie. Un autre bémol est aussi celui de l'efficacité réduite du transfert entre les réseaux PV et/ou les éoliennes, et les consommateurs.

Etant donné que l'énergie fournie par le soleil devrait couvrir toute la consommation d'énergie du monde entier, si elle est capturée et stockée à coûts raisonnables, l'utilisation des panneaux photovoltaïques aura une popularité croissante dans le temps. Le secteur de « l'énergie verte » s'accompagne, dès lors, de nouveaux défis tendant à résoudre les problèmes relatifs à un taux de conversion faible, la nécessité d'adjoindre des systèmes de stockage d'énergie additionnels et l'inefficacité du transfert entre les réseaux PV et leur charge connectée.

Les centrales électriques photovoltaïques jusqu'à 500kw situées tant dans des zones isolée qu'en zone urbaine sont promises à un développement exponentiel et ce, au regard du fait que beaucoup d'Etats réorientent leurs politiques en considération d'une production d'énergie incluant l'utilisation de « l'énergie verte ». A ce titre, la tendance générale est de promouvoir les concepts d'optimisation de l'énergie et d'indépendance énergétique à travers les sources d'énergies renouvelables, parmi lesquels les panneaux photovoltaïques trouvent une place des plus favorables. Aussi, dans cet esprit, l'Union Européenne a adopté la directive 2010/31/UE, suivant laquelle, d'ici 2021, toutes les nouvelles constructions devront répondre à la norme de « consommation d'énergie quasi nulle » suggérant qu'une quantité significative d'énergie devra être pourvue par des ressources renouvelables locales ou voisines.

Parce que les panneaux PV ont encore un taux de conversion réduit, une forte et rapide puissance de variation, et une large répartition des générateurs photovoltaïques (??), il est implicitement évident qu'une gestion optimale de l'énergie est essentielle. Il y a deux importantes solutions afin d'y parvenir : la première consiste en la construction d'un grand réseau énergétique, réparti sur des régions géographiques avec des conditions de génération de puissance variables alors que la seconde implique l'utilisation de solutions de stockage intelligent [7]. Trois principaux types de stockage existent : le stockage mécanique, comme par exemple, les systèmes de pompes à eau (stockage par pompage de l'eau à Ludington : 110m, 1.87 GW, 15h, 27 million kWh); des solutions à base de batterie, avec différents types de batteries utilisées, chacune avec ses avantages et ses désavantages [8],[13]; et finalement, le stockage à base d'hydrogène, apportant une forte densité d'énergie, une bonne efficacité (bon rendement, you choose) de conversion et une robustesse physique.

L'objectif de cette thèse est de présenter et de comparer différentes stratégies de commandes pour les systèmes alimentés par les sources d'énergies renouvelables. Un prototype destiné à des fins d'essais a été conçu.

Cette thèse traite de différents aspects tels que la modélisation de panneaux PV, la observateur non linéaire, un algorithme de contrôle basé sur une recherche du point de puissance maximum

(Maximum modélisation d'un convertisseur abaisseur DC/DC, construire un Point Power Point Tracking), un algorithme de contrôle polynôme, la stabilité du système.

Le chapitre 2 présente différents modèles de cellule photovoltaïque qui peuvent, en outre, être utilisés dans une boucle de contrôle. Une interface utilisateur graphique est créée pour faciliter le calcul de certains paramètres et de la courbe caractéristique de la tension voltage du panneau PV. De plus, un modèle à espace d'états et un modèle de fonction de transfert de certains convertisseurs DC/DC sont présentés.

Le chapitre 3 se concentre sur l'élaboration d'un observateur Takagi-Sugeno (T-S) qui fournit la tension estimée du panneau PV. Ce dernier sera, plus tard, utilisé dans le bloc de commande ou pourra servir pour les diagnostics.

Le chapitre 4 compare différents algorithmes MPPT classique, ainsi qu'un algorithme de contrôle avancé qui pourra être utilisé plus tard pour améliorer les performances des boucles de contrôles. Une étude de cas sur une commande de supervision utilisant une cellule à combustion est proposée.

Le chapitre 5 est orienté vers une approche plus pratique. Il présente un système de contrôle distribué qui et géré via un serveur OPC. Un système d'acquisition de données enregistre les informations envoyées par chacune des boucles de contrôle et est capable de tracer les données en temps réel.

Le chapitre 6 est dédié aux conclusions.

Le chapitre 7 présente les codes des logiciels développés et certains schémas qui ont été utilisés durant les simulations.

Le chapitre 8 liste la bibliographie.

2. CONSTRUCTION OF THE MATHEMATICAL MODELS

2.1. Photovoltaic panels

Whenever a model is used in a control loop or a supervisory system, its accuracy plays a crucial role. In this chapter we shall present different typologies of photovoltaic cell models.

Before proceeding, a brief reminder of what a solar panel is made of, and how it is functioning will be made.

A photovoltaic panel is composed of several photovoltaic cells (Figure 2-1, Figure 2-2 and Figure 2-3). A PV cell captures the energy provided by the sun and transforms into variable DC current.

On the present market, we can distinguish three types of technologies that are used for manufacturing them.

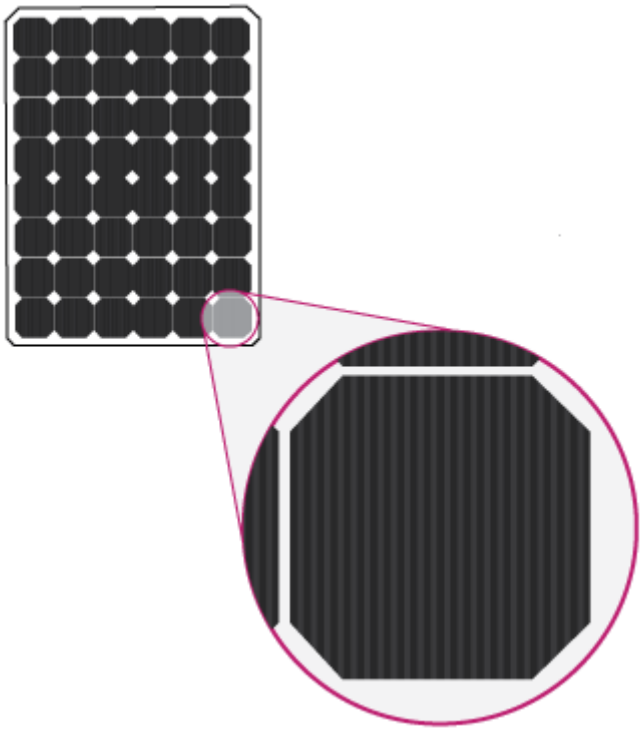


Figure 2-1 Monocrystalline solar cells

Monocrystalline solar cells (Figure 2-1) are created from a single continuous crystal structure. The silicon crystal is formed into bars and cut into wafers. Using only one type of crystal offers a higher conductivity to electrons which generate an electric flow, which leads to a lower temperature coefficient. Therefore, it has an increased efficiency of 10÷23% and a better tolerance to higher operating temperatures, with respect to the other technologies presented below. It also requires less space than its homologues.

However, since the fabrication process is more expensive, its final price might represent a drawback for the buyer. They are particularly sensible to shadowing effects if they do not benefit of the bypass diodes devices.

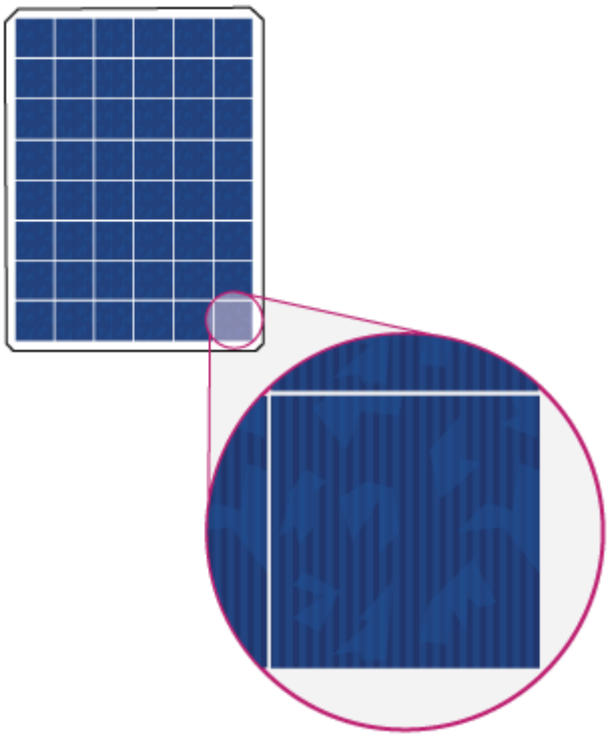


Figure 2-2 Polycristalline solar cells

Polycristalline solar cells (Figure 2-2) are created from fragments of silicon that are melted together to form wafers. Along with the lower purity, the freedom of the electrons to move is diminished as well. This will conduct to a lower efficiency and a higher operating temperature.

They have a lower fabrication cost, due to the simplified technology, which makes them more attractive.

Figure 2-3 Thin film amorphous panels

Thin film amorphous panels (Figure 2-3) are created by depositing substances, such as amorphous silicon, cadmium telluride, copper indium gallium selenide or others, on a solid

surface. They have an efficiency around 7÷13% and the lowest production cost among the three mentioned technologies. Shadowing effects have a smaller impact over this type of panel, in comparison with the previous two others.

We may associate a PV cell with a PN junction diode, since it is a semiconductor with a positive P type layer and a negative N type later, as presented in Figure 2-4 connected in a series / parallel architecture.

When the solar radiation hits the surface of the semiconductor, it sets free electrons that will flow and be collected by the metallic strips. This will represent the positive junction of the PV cell. On the other side, the P-type material, represents the negative junction.

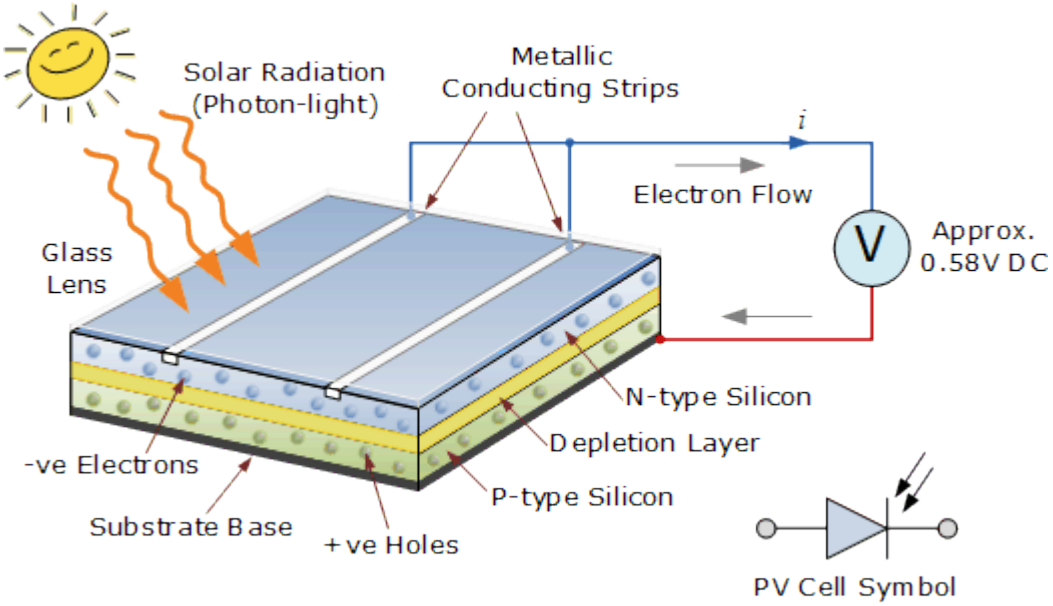


Figure 2-4 PN layer

In order to increase the output voltage of a PV panel, the PV cells are connected in series. To increase the output current, we can either connect the PV cells in parallel, either replace them with others that have a bigger surface area.

Photovoltaic arrays, which are obtained by interconnecting multiple PV panels in a series / parallel configuration, can be used to achieve certain power performances.

For the next part, the following conventions will be made:

I_{pv} is the output current of the PV cell;

I_{ph} is the photogenerated current by the incidence of light;

V is the output voltage of the PV cell;

I_{d1} is the diffusion diode current (obtained with the Shockley diode equation);

I_{d2} is the recombination diode current;

$I_{01}$and I_{02} are the reverse saturation currents or leakage currents of the diode D_1,respectively D_2;

V_{T1} and $V_{T2}$are the thermal voltages of the two diodes;

$V_T = \frac{k_b}{q} T$, where $k_b$is the Boltzmann constant ($1.380650 * 10^{-23}$ J/K), q is the electron charge ($1.602176 * 10^{-19}$C) and T is the temperature of the p-n junction (in Kelvin);

$\alpha_1$and $\alpha_2$are the ideality factors of the two diodes;

$R_s$is a series resistance;

$R_p$is a shunt resistance;

I_{ph_STC} is the light generated current measured in standard test conditions (STC);

K_i is the short circuit current coefficient (measured in mA/K);

K_v is the open circuit current coefficient (measured in mV/K);

$V_{oc,STC}$ is the open circuit voltage at STC;

$I_{sc,STC}$ is the short circuit current at STC;

$\Delta T = T - T_{STC}$ (temperature measured in Kelvin);

G is the insolation or irradiance, while $G_{STC}$is the irradiance measured in STC.

The performances of a PV panel are measured under standard test conditions (STC):

- the photovoltaic cell's temperature $= 25°$;
- the irradiance $= 1000\ W/m^2$;
- air mass of 1.5 spectrum.

2.1.1. One diode photovoltaic panel model

Different models of photovoltaic (PV) cells are presented in several paperwork [4,5,8].

The complexity of the chosen model might have a great impact upon the accuracy of the simulation. Thus, the nonlinear I-V and P-V characteristic curves are affected.

The simplest (ideal) PV cell model, consisting of a current source (I_{ph}) connected in parallel to a diode (D_1), offers an output current (I_{pv}) directly proportional to the radiation level.

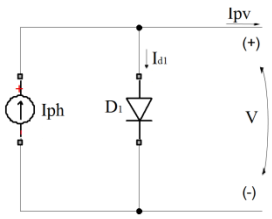


Figure 2-5 Ideal PV cell model

The light generated current can be expressed as:

$$I_{ph} = \left(I_{ph\_STC} + K_i \Delta T\right) \frac{G}{G_{STC}} \tag{1}$$

It can be observed that this term is dependent on both irradiance and temperature. We shall consider, for all further calculus, a constant AM=1.5 atmosphere thickness, which corresponds to a solar zenith angle of 48.2°.

We can compute the value of the output current of the photovoltaic cell, by applying Kirchhoff's current law:

$$I_{pv} = I_{ph} - I_{d1} \tag{2}$$

where the current of the diode is calculated using the Shockley diode equation:

$$I_{d1} = I_0 \left[e^{\frac{V}{V_{T1}}} - 1 \right] \tag{3}$$

The reverse saturation current of the diode can be expressed as following:

$$I_0 = I_{0,STC} \left(\frac{T_{STC}}{T} \right)^3 e^{\left[\frac{qE_g}{\alpha k_b} \left(\frac{1}{T_{STC}} - \frac{1}{T} \right) \right]} \tag{4}$$

where $E_g$is the band gap energy of the semiconductor (usually $1.12\ eV$ for polycrystalline Si at $25°C$), while $I_{0,STC}$ is reverse saturation current at STC.

Most PV panels datasheets do not offer enough information regarding some of the above mentioned parameters, thus an improved equation which takes into account temperature variation replaces the previous one:

$$I_0 = \frac{\left(I_{sc\_STC} + K_i \Delta T\right)}{e^{\left[(V_{oc,STC} + K_v \Delta T)/(\alpha V_T) \right]} - 1} \tag{5}$$

Replacing the light generated current and the current of the diode from eq.(2) with the values from eq.(1), (3) and (5), we obtain the output current of the ideal photovoltaic cell:

$$I_{pv} = \left(I_{ph\_STC} + K_i \Delta T\right) \frac{G}{G_{STC}} - \frac{\left(I_{sc_{STC}} + K_i \Delta T\right)}{e^{\left[\frac{(V_{oc,STC} + K_v \Delta T)}{(\alpha V_T)} \right]} - 1} \left[e^{\frac{V}{V_{T1}}} - 1 \right] \tag{6}$$

This model has the advantage of having a reduced number of parameters, necessary for computing the I-V characteristic curve.

However it presents a big drawback when exposed to environmental variation, fact which makes it unsuitable for real world functioning.

An improved model is obtained by adding a series resistance (R_s), which takes into account the voltage drops and the internal losses. It is one of the most frequently used models in paperwork and PV cell simulations.

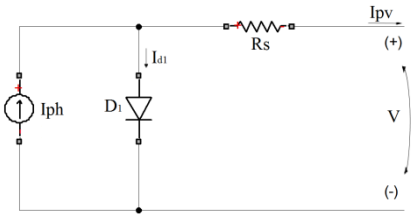


Figure 2-6 Rs PV cell model

The output current of the photovoltaic cell is:

$$I_{pv} = I_{ph} - I_{d1} \tag{7}$$

where the current of the diode becomes:

$$I_{d1} = I_0 \left[e^{\frac{V + I_{pv} R_s}{V_{T1}}} - 1 \right] \tag{8}$$

When the device operates in the current source region, this model lacks precision.

Another model is obtained by adding a parallel resistance (R_p) to the previous structure. The I-V characteristic curve of this model is closer to that of the PV cell than in the other cases, by taking into account the leakage current to the ground when the diode is in reverse bias.

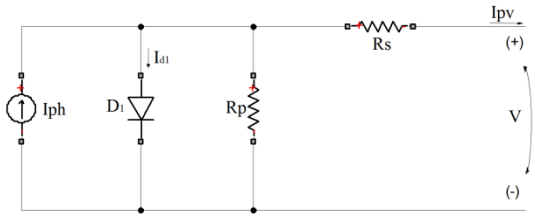


Figure 2-7 Rs-Rp PV cell model

The output current of the photovoltaic cell changes to:

$$I_{pv} = I_{ph} - I_{d1} - \left(\frac{V + I_{pv} R_s}{R_p} \right) \tag{9}$$

It is shown [4] that the one diode PV cell model seems to lack precision when subjected to low voltages, by neglecting the recombination loss in the depletion region of the diode.

2.1.2. Double diode photovoltaic model

Another model is obtained by adding a parallel resistance (R_p) to the previous structure. The I-V characteristic curve of this model is closer to that of the PV cell than in the other cases. It is shown [4] that the one diode PV cell model seems to lack precision when subjected to low voltages, by neglecting the recombination loss in the depletion region of the diode.

In order to solve this problem a trade-off between the accuracy and the complexity of the model is made. Another diode (D_2) is thus added. The two diode PV cell model is presented in Figure 2-8.

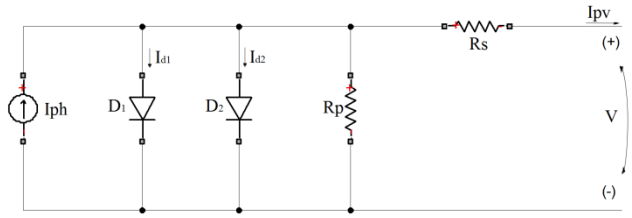


Figure 2-8 Double diode PV cell model

The output current (I_{pv}) equation of the PV module is:

$$I_{pv} = I_{ph} - I_{d1} - I_{d2} - \left(\frac{V + I_{pv}R_s}{R_p}\right) = \\ = I_{ph} - I_{01}\left[e^{\frac{V+I_{pv}R_s}{\alpha_1 V_{T1}}} - 1\right] - I_{02}\left[e^{\frac{V+I_{pv}R_s}{\alpha_2 V_{T2}}} - 1\right] - \left(\frac{V + I_{pv}R_s}{R_p}\right) \quad (10)$$

It can be observed the number of parameters of this model increased to seven: I_{pv}, I_{01}, I_{02}, R_s, R_p, $\alpha_1$and α_2.

In order to reduce the time computation the following simplification is made:

$$I_{01} = I_{02} = \frac{\left(I_{sc\_STC} + K_i\Delta T\right)}{e^{\left[\left(V_{oc,STC}+K_v\Delta T\right)/\{[(\alpha_1+\alpha_2)/p]V_T\}\right]} - 1} \quad (11)$$

where $\frac{(\alpha_1+\alpha_2)}{p} = 1$, $\alpha_1 = 1$ and $p \geq 2.2$.

The values of the R_s and R_p resistances are computed using the algorithm presented in chapter 2.1.3.

The reverse saturation current of the two diodes becomes:

$$I_{01} = I_{02} = \frac{\left(I_{sc\_STC} + K_i\Delta T\right)}{e^{\left[\left(V_{oc,STC}+K_v\Delta T\right)/V_T\right]} - 1} \quad (12)$$

Typical PV power generation systems are configured in series parallel structures in order to achieve certain power demands. The next equation describes output current of such a N_sx N_p size PV module:

$$I_{PV} = I_{ph}N_p - I_{01}N_p\left[e^{\frac{V+IR_s\left(\frac{N_s}{N_p}\right)}{\alpha_1 V_{T1} N_s}} - 1\right] - I_{02}N_p\left[e^{\frac{V+IR_s\left(\frac{N_s}{N_p}\right)}{\alpha_2 V_{T2} N_s}} - 1\right] - \left(\frac{V + IR_s\left(\frac{N_s}{N_p}\right)}{R_p\left(\frac{N_s}{N_p}\right)}\right) \quad (13)$$

The power of a PV array, $P_{PV} = I_{PV}V$, is dependent on the insolation, the cell temperature and the connected load.

Different maximum power point tracking (MPPT) algorithms can be found in literature. Their goal is to optimize the efficiency of the PV array by delivering a maximum power that the PV can achieve with respect to the environmental conditions and load.

The maximum power point should satisfy the following condition:

$$\frac{dP}{dV} \cong 0 \quad (14)$$

$$\frac{dP}{dV} = \frac{d\left(I_{pv}V\right)}{dV} = I_{pv} + V\frac{dI_{pv}}{dV}$$

$$= I_{pv} + V\left[-I_{01}N_p\left(\frac{1}{\alpha_1 V_{T1} N_s}\right)e^{\frac{V+IR_s}{\alpha_1 V_{T1}}} - I_{02}N_p\left(\frac{1}{\alpha_1 V_{T1} N_s}\right)e^{\frac{V+IR_s}{\alpha_1 V_{T1}}} - \frac{1}{R_p\left(\frac{N_s}{N_p}\right)}\right] \quad (15)$$

A sequence of plots will show the differences between the three types of PV models mentioned earlier for different functioning conditions. The PV panel that is simulated has the characteristics taken from the datasheet of a 30W PV, Ameresco Solar 30J.

The I-V curve and P-V curve for STC are presented in Figure 2-9 and Figure 2-10. The same curves are plotted for different values of the irradiation and temperature in Figure 2-11and Figure 2-12.

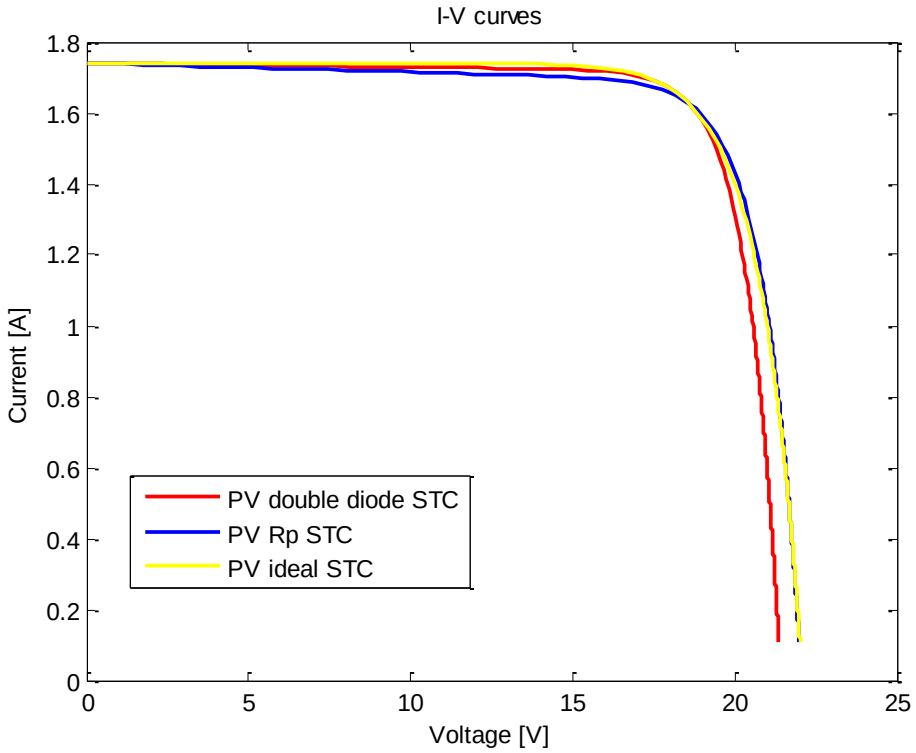


Figure 2-9 :I-V curves for STC

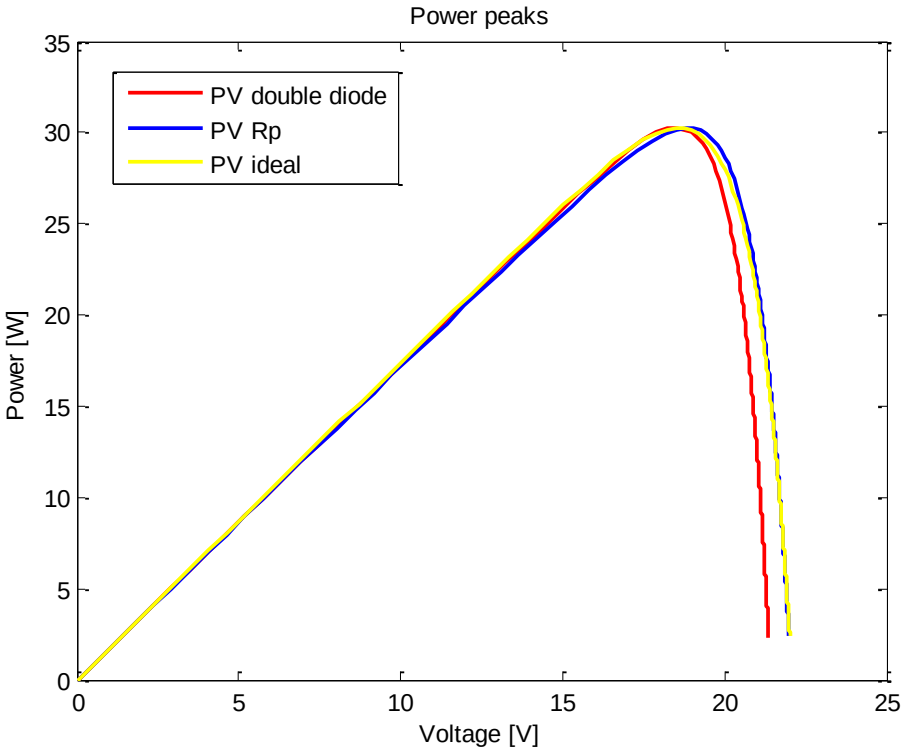


Figure 2-10: P-V curves for STC

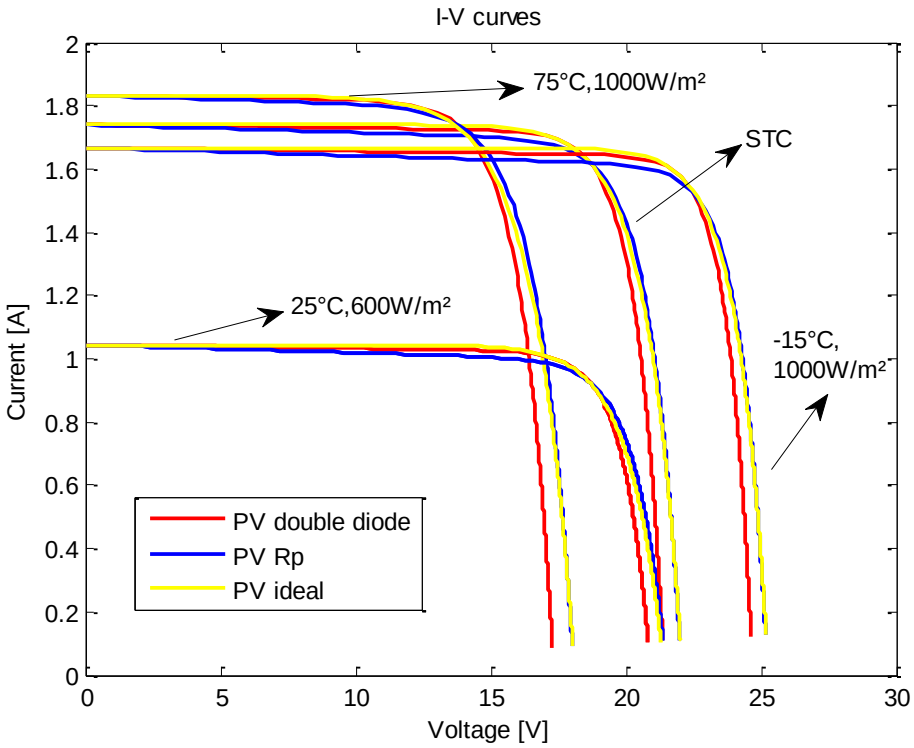


Figure 2-11: I-V curves

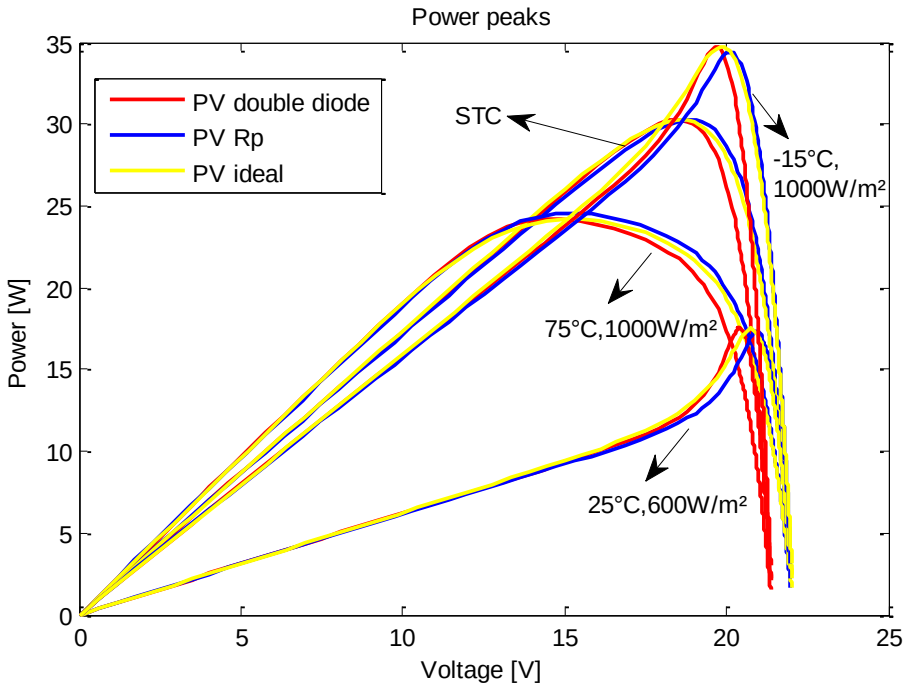


Figure 2-12: P-V curves

2.1.3. Graphic user interface

A solar panel such as FVG100P can be used for many low budget applications.

Having a quick look over the data (Table 2-1) that is provided by the manufacturer one can observe that the values of resistances R_s and R_p are not available.

Table 2-1 FVG100P PV datasheet @STC

Cells in series	36
Maximum rated Power – Pm [W]	9.975
Maximum power Voltage – Vm [V]	17.5
Maximum power Current – Im [A]	0.57
Open Circuit Voltage – VOC [V]	21.00
Short Circuit Current – ISC [A]	0.66
VOC Temperature Coefficient [%/°C]	-0.35
ISC Temperature Coefficient [%/°C]	0.05

A solution for this scenarios is obtained by applying the Newton-Raphson algorithm for finding the values of R_s and R_p described in [1, 2, 8], adapted for the two diode PV cell model. The following graphic user interface (GUI) was developed to facilitate the computation of the two values (Figure 2-13).

After filling in the gaps with the data provided in the datasheet of the PV panel and choosing a suitable value for α_2 (α_1=1, by default) the button that enables the search algorithm can be pressed. The results will be shown among with the I-V and P-V characteristic curves. The simulation is made for a 1500 watts PV array, as in Figure 2-13., while the characteristic curves can be observed in Figure 2-15.

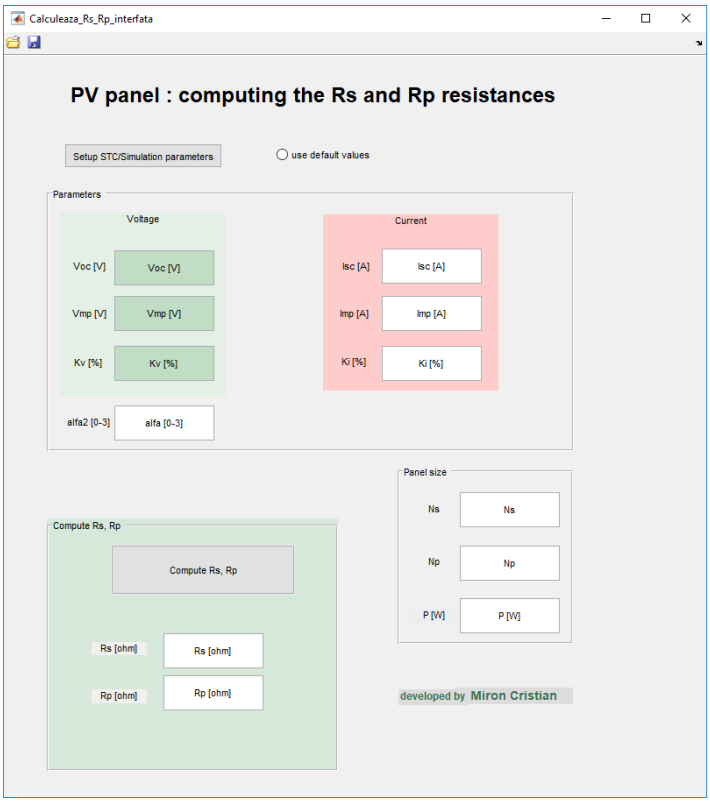


Figure 2-13: Graphic User Interface

For configuring the values of irradiance and temperature, we can either use the STC values by checking the radiobutton (4), either we can set them by clicking the "Setup STC/Simulation parameters" button (3). As presented in Figure 2-14, the two parameters can be manually set.

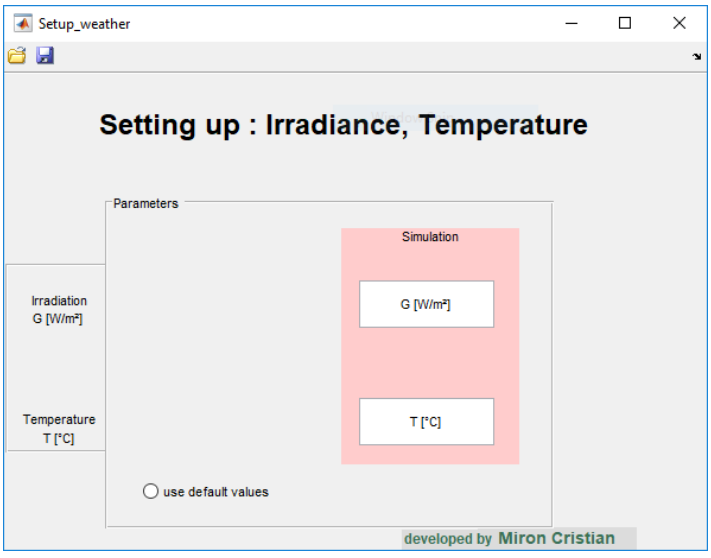


Figure 2-14 Secondary GUI window

Furthermore we can set up the values of the Voc, Vmp and Kv, by filling in the three input boxes from panel (5). We can proceed similarly for setting up the values of Isc, Imp and Ki, from the

other three input boxes present in panel (6). The input box (7) is used for setting the ideality factor of the second diode.

In panel (8) we can decide upon the value of Ns, Np and P, which represents the maximum rated power at STC, for Ns connected PV cells.

In order to compute the values of Rs and Rp, using the Newton-Rhapson algorithm, for a model that is using a single/double diode, we have to push the "Compute Rs,Rp" button (8).

The computed values, after applying all the above mentioned steps, are $R_s = 0.0101\Omega$ and $R_p = 3.061\Omega$.

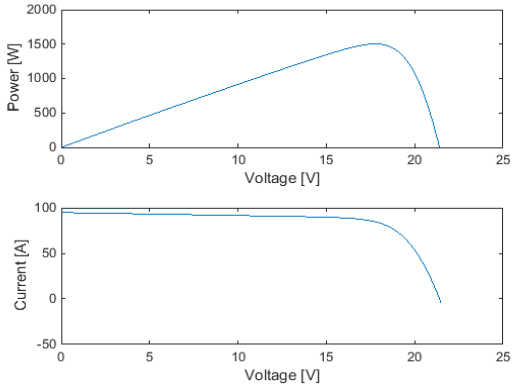


Figure 2-15:I-V and P-V characteristic curves

Moreover, we can save the results by pushing the save button (2), and eventually load them later on, by pushing the open button (1).

2.2 DC/DC converter model

2.2.1 Statespace model

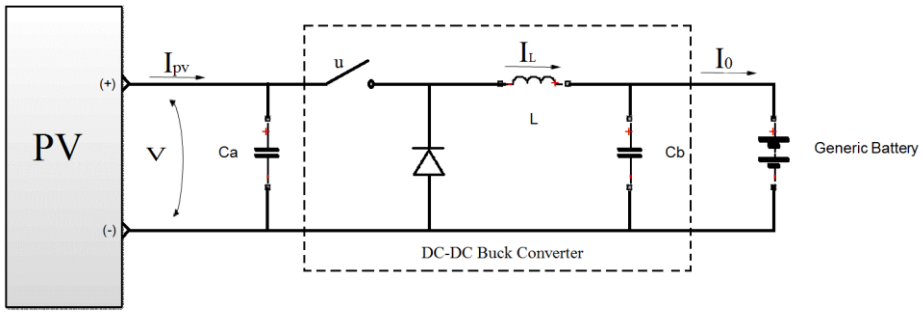


Figure 2-16: DC/DC Buck Converter I

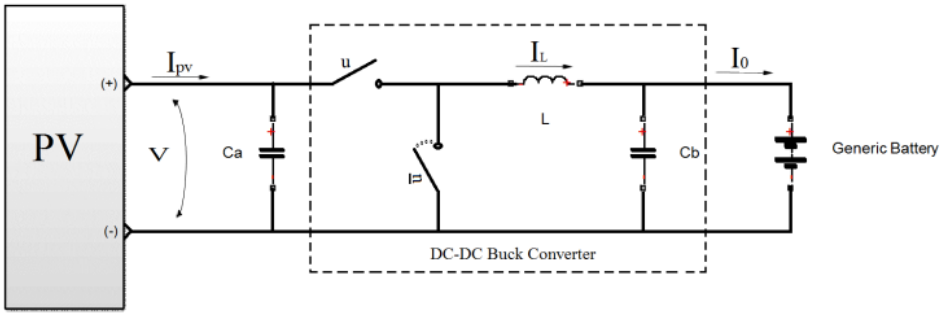


Figure 2-17 DC/DC Buck Converter II

We presume that the PV array is connected to a variable load, a battery in this case. In order to optimize the efficiency of the PV array and to align its voltage to that of the load, a step down converter (DC-DC buck converter) is added, as presented in Figure 2-17.

A synchronous buck converter topology is generally preferred, due to a significant reduction of losses. The buck converter is composed of an inductor (L), a capacitor (C_b), a high-side mosfet that is controlled using a pulse-width modulation (PWM) signal – the duty ratio u, and a low-side mosfet that is controlled via PWM signal –the duty ratio $\overline{u}$. The latter component replaces the diode from an asynchronous buck converter topology (Figure 2-16) and offers the advantage of a lower resistance from drain to source which contributes to a higher conversion efficiency.

The purpose of this scheme is that the power of the PV array is conserved: $P_{out\_buck} = P_{in\_buck} + P_{losses}$. The power losses in a buck converter are caused by the power dissipation in the conduction of the inductor, of the transistors and on the switching losses of the transistors. The voltage drop on the diode is considered, whenever the asynchronous topology is preferred.

Furthermore is presented the dynamic model of the buck converter, using state equations. The model will be calculated for the converter presented in Figure 2-16, and finally generalized for both topologies.

We shall use the next differential equations of the inductor (eq. 16) and of the capacitor (eq. 17):

$$V_L(t) = L\frac{dI_L(t)}{dt} \tag{16}$$

where V_L is the induced voltage on the inductor, L is inductance, $\dot{I_L}$ is the rate of change of the current across the inductor.

$$I_b(t) = C\frac{dV_b(t)}{dt} \tag{17}$$

where I_b is the current that passes through the capacitor,C is the capacitance,$\dot{V_b}$the rate of change of voltage across the capacitor.

If we apply them to the inductor and the two capacitors present in Figure 2-18, we shall obtain:

$$\begin{cases} \dfrac{dI_L}{dt} = \dfrac{1}{L} V_L \\ \dfrac{dV}{dt} = \dfrac{1}{C_a}\left(I_{pv} - I_L u\right) \\ \dfrac{dV_b}{dt} = \dfrac{1}{C_b}\left(I_L - I_{out}\right) \end{cases} \tag{18}$$

In order to obtain the value of the inductor's voltage, V_L, the second law of Kirchhoff is applied between meshes 1 and 2. We shall also consider the internal resistance of the inductor and of the capacitor, as presented in Fig. 2-18.

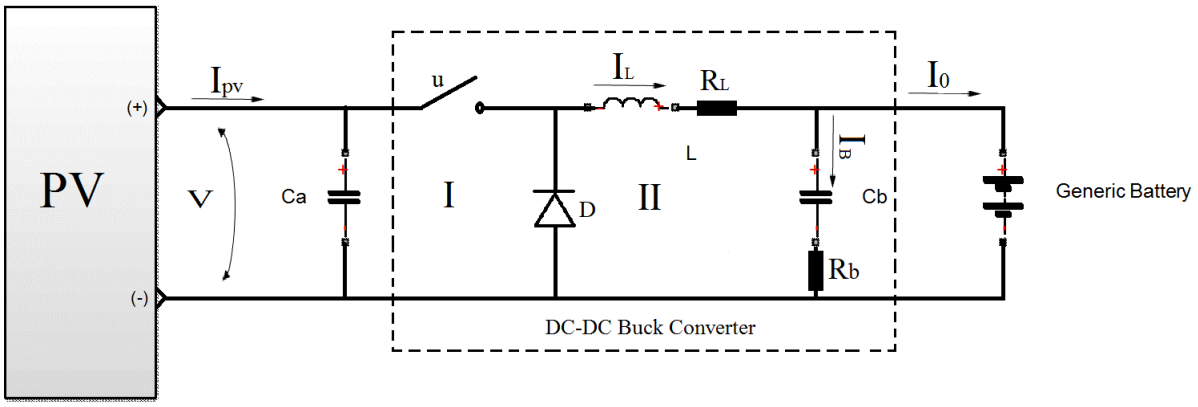


Figure 2-18 The 2nd law of Kirchhoff applied on a DC\DC buck converter

The on state and the off state of the buck converter, when the command u applied on the transistor closes or opens the circuit, are taken into account.

$$V_L + V_{R_L} + V_b + V_{R_b} + V_D - u(V_D + V) = 0 \tag{19}$$

$$V_L = -I_L R_L - V_b - I_b R_b - V_D + u(V_D + V) \tag{20}$$

We shall use the first law of Kirchhoff for calculating the current Ib:

$$I_L = I_0 + I_b \tag{21}$$

(22)

If we reintroduce eq. 21 in eq. 20, we obtain:

$$V_L = -I_L R_L - V_b + (I_0 - I_L) R_b - V_D + u(V_D + V) \tag{23}$$

$$V_L = I_0 R_b - V_b - (R_b + R_L) I_L - V_D + u(V_D + V) \tag{24}$$

Eq. 18 can be rewritten as:

$$\begin{cases} \dfrac{dI_L}{dt} = \dfrac{1}{L}(I_0 R_b - (R_b + R_L)I_0 - V_b - V_D + u(V_D + V)) \\ \dfrac{dV}{dt} = \dfrac{1}{C_a}(I_{pv} - I_L u) \\ \dfrac{dV_b}{dt} = \dfrac{1}{C_b}(I_L - I_0) \end{cases} \quad (25)$$

where :

I_{pv} is the output current of the PV array;

I_0 is the output current of the buck converter;

I_L is the current on the inductor L;

V is the voltage of the PV array on the capacitor C_a;

V_b is the voltage on the capacitor C_b;

R_L is the internal resistance of the inductor L;

R_b is the internal resistance of the capacitor C_b;

u is the PWM signal that controls the transistor(s), that is associated to a given duty cycle;

$\overline{u}$=1-u.

The efficiency of the PV array is also affected by the internal resistances R_L and R_b. This aspect is not usually taken into consideration in MPPT paperwork.

2.2.2. Transfer function model

We shall use a simplified version of eq. 25, in order to compute the transfer function of an asynchronous buck converter topology, in continuous mode. The internal resistances of the inductance and of the capacitor, as well as the voltage drop of the diode, are not taken into account. Furthermore, we shall consider for the moment, the variable *V* as being constant.

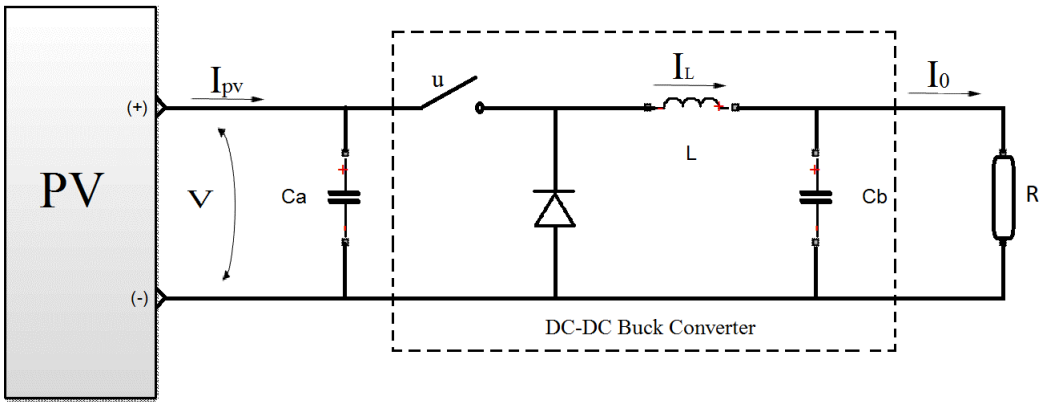


Figure 2-19 Simplified DC\DC buck converter

$$\begin{cases} \frac{dI_L(t)}{dt} = u(t)\frac{V}{L} - \frac{V_b(t)}{L} \\ \frac{dV_b(t)}{dt} = \frac{I_L(t)}{C_b} - \frac{I_0(t)}{C_b} \end{cases} \quad (26)$$

We shall note the resistance of the generic load with *R*.

$$\begin{cases} \frac{dI_L(t)}{dt} = u(t)\frac{V}{L} - \frac{V_b(t)}{L} \\ \frac{dV_b(t)}{dt} = \frac{I_L(t)}{C_b} - \frac{V_b(t)}{RC_b} \end{cases} \quad (27)$$

To convert the differential equations in time *t* into algebraic equations in complex domain *s*, we will apply the Laplace transform.

$$\begin{cases} sI_L(s) = u(s)\frac{V}{L} - \frac{V_b(s)}{L} \\ sV_b(s) = \frac{I_L(s)}{C_b} - \frac{V_b(s)}{RC_b} \end{cases} \quad (28)$$

$$I_L(s) = (sV_b(s) + \frac{V_b(s)}{RC_b})C_b \quad (29)$$

$$s\left(sV_b(s) + \frac{V_b(s)}{RC_b}\right)C_b = u(s)\frac{V}{L} - \frac{V_b(s)}{L} \quad (30)$$

$$V_b(s)\left(s^2C_b + \frac{s}{R} + \frac{1}{L}\right) = u(s)\frac{V}{L} \quad (31)$$

Represented in transfer function form, one gets:

$$\frac{V_b(s)}{u(s)} = \frac{V}{s^2LC_b + s\frac{L}{R} + 1} \quad (32)$$

The *V* variable is not constant so a procedure needs to be implemented in order to solve this.

Considering a slower dynamic in the MPPT block, the Vin can be considered to vary slower than the buck converter's dynamics. As such, we can simply measure its value and divide the obtained command by this measured value.

3. TAKAGI-SUGENO OBSERVER

3.1 TS fuzzy model

Most of the physical dynamical systems, which can be represented with differential equations are nonlinear. When the linearization of a system is not representative, this may lead to its instability in a control loop which is not robust enough to compensate all the modelling uncertainties and/or disturbances. A multi-model approach [2] may present a better accuracy of the whole system, especially when the latter is exposed to parameter variations or structural changes for different functioning points.

To design a T-S fuzzy observer or controller (Figure 3.1), we need a T-S fuzzy model for a nonlinear system. Therefore, the construction of a fuzzy model represents an important and basic procedure in this approach.

This chapter focuses on the approach based on the idea of "sector nonlinearity", "local approximation," or a combination of them to construct fuzzy models.

The idea of using sector nonlinearity in fuzzy model construction first appeared in [8]. This approach guarantees an exact fuzzy model construction. However, it is sometimes difficult to find global sector for general nonlinear systems then local sector nonlinearity is considered. This is reasonable as variables of physical systems are always bounded.

Consider a simple nonlinear system $\dot{x}(t) = f(x)$, where $f(x)$ =0. The aim is to find the global sector such that $\dot{x}(t) = f(x)$ $[a_1\ a_2]$. Figure 3-2 shows the local sector nonlinearity of a function

where two lines become the local sectors under $-d < x(t) < d$. The fuzzy model exactly represents the local region, that is, $-d < x(t) < d$.

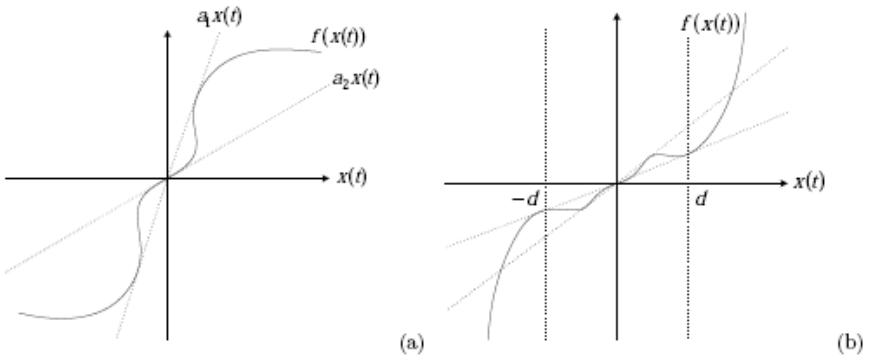


Figure 3-2 Local sector nonlinearity

The plant can be described by TS fuzzy [2, 6] plant model without uncertainties and disturbance and can be written in the following form:

$$\begin{aligned} \dot{x}(t) &= \sum_{i=1}^{r} h_i\big(z(t)\big)\big\{A_i x(t) + B_i u_{inputs}\big\} \\ y(t) &= \sum_{i=1}^{r} h_i\big(z(t)\big) H_i x(t) \end{aligned} \tag{33}$$

where $x(t) \in R^{n\times 1}$ is the state vector, $u_{inputs}(t) \in R^{m\times 1}$ is the input vector, $y(t) \in R^{g\times 1}$ is the output vector, $A_i \in R^{n\times n}$, $B_i \in R^{n\times m}$, $H_i \in R^{g\times n}$ are known matrices with appropriate dimensions and r is the number of rules or sub-models. The functions $h_i(z(t))$ are the weighting functions depending on the state variables $z(t)$ which can be measurable or not measurable variables. These functions verify the so-called convex properties.

$$\sum_{i=1}^{r} h_i\big(z(t)\big) = 1, \qquad 0 \le h_i\big(z(t)\big) \le 1\,, \forall i \in \{1, \dots, nz\} \tag{34}$$

The output equation is chosen to be linear with respect to the state, which is frequently the case in practice.

The normalized h_i functions are given by:

$$h_i\big(z(t)\big) = \frac{w_i\big(z(t)\big)}{\sum_{i=1}^{r} w_i\big(z(t)\big)} \tag{35}$$

The weighting $w_i$functions have the following structure

$$w_i\big(z(t)\big) = \prod_{j=1}^{nz} F_{ji}(z(t)) \tag{36}$$

The membership functions are represented by:

$$\begin{aligned} f_{aj} &= \frac{z_j(t) - zm_j}{zM_j - zm_j} \\ f_{bj} &= 1 - f_{aj} \end{aligned} \tag{37}$$

3.2 TS fuzzy observer

A model-based fuzzy observer design utilizing the concept of the Parallel Distributed Compensation (PDC) is used. The concept of PDC technique in [] is utilized to design fuzzy controllers or fuzzy observers to stabilize fuzzy system. The idea of PDC is to associate a compensator for each rule of the fuzzy model. The resulting overall fuzzy observer in our case is a fuzzy blending of each individual linear observer. The fuzzy observer shares the same fuzzy sets with the fuzzy system.

It is assumed that the fuzzy system model is locally observable. The PDC is also employed to design the following observer structures.

The T-S observer has the following form:

$$\begin{cases} \dot{\hat{x}}(t) = \sum_{i=1}^{r} h_i\big(\hat{z}(t)\big)\big\{A_i\hat{x}(t) + B_i u_{inputs}(t) + L_i(\Delta y)\big\} \\ \hat{y}(t) \ = H\hat{x}(t) \end{cases} \tag{38}$$

where $\hat{x}(t) \in R^{n\times 1}$is the estimated state vector, Δy is the error between the measured signal and the estimated signal, while L_i are the fuzzy observer gains.

3.3 State space model of PV plant

First, a nonlinear PV plant with a Takagi-Sugeno (TS) fuzzy model is presented. Then a model-based fuzzy observer design utilizing the concept of the Parallel Distributed Compensation (PDC) is used, in which observer gains are obtained by solving Linear Matrix Inequalities (LMIs). The control block based on MPPT algorithm will use in real-time the result provided by the nonlinear observer.

Choosing which model or group of models gives the best approximation of the system, with respect to a certain performance criteria, is important.

T-S fuzzy model of PV plant

The system described in eq. 25 is a non-linear system. A synchronous topology of the buck converter will be preferred, therefore the diode will be replaced with a low side mosfet, as presented in Figure 2-17. The approach that is used in this paper takes into account the number

of nonlinearities of the system (nz) and builds a set of linear subsystems (2^{nz}) that approximate the initial system. Each linear subsystem will approximate the non-linear system with a certain weight.

The state space model is rewritten as below:

$$\begin{bmatrix} \frac{dV}{dt} \\ \frac{dI_L}{dt} \\ \frac{dV_b}{dt} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{u}{2C_a} & 0 \\ \frac{u}{2L} & -\frac{1}{L}(R_b + R_L) & -\frac{1}{L} \\ 0 & \frac{1}{C_b} & 0 \end{bmatrix} \begin{bmatrix} V \\ I_L \\ V_b \end{bmatrix} + \begin{bmatrix} -\frac{I_L}{2C_a} & 0 & \frac{1}{C_a} \\ \frac{V}{2L} & -\frac{R_b}{L} & 0 \\ 0 & -\frac{1}{C_b} & 0 \end{bmatrix} \begin{bmatrix} u \\ I_{out} \\ I_{pv} \end{bmatrix} \tag{39}$$

This can be formalized as the following system:

$$\dot{x} \equiv A(x)x + B(x)u_{inputs} \tag{40}$$

where $x = \begin{bmatrix} V \\ I_L \\ V_b \end{bmatrix}$, $A = \begin{bmatrix} 0 & -\frac{u}{2C_a} & 0 \\ \frac{u}{2L} & -\frac{1}{L}(R_b + R_L) & -\frac{1}{L} \\ 0 & \frac{1}{C_b} & 0 \end{bmatrix}$, $B = \begin{bmatrix} -\frac{I_L}{2C_a} & 0 & \frac{1}{C_a} \\ \frac{V}{2L} & -\frac{R_b}{L} & 0 \\ 0 & -\frac{1}{C_b} & 0 \end{bmatrix}$ and $u_{inputs} = \begin{bmatrix} u \\ I_{out} \\ I_{pv} \end{bmatrix}$.

$$y = H \begin{bmatrix} V \\ I_L \\ V_b \end{bmatrix} = Hx \tag{41}$$

where $H = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is chosen with respect to the parameter that will be estimated.

A Takagi-Sugeno (T-S) fuzzy modelling technique [2,6] is used:

$$\begin{cases} \dot{x}(t) = \sum_{i=1}^{8} h_i(z(t))\{A_i x(t) + B_i u_{inputs}(t)\} \\ y(t) = Hx(t) \end{cases} \tag{42}$$

where $z(t) = [z_1(t)\ z_2(t)\ z_3(t)]$ contains the nonlinearities of the system, also known as fuzzy variables, and h_i represents the normalized weighting functions.

We choose $z1 = u$, $z2 = V$ and $z3 = I_L$, obtaining a number of r=8 fuzzy rules.

Taking into account these notations the matrixes A_i and B_i from eq. 34 become:

$$A_i = \begin{bmatrix} 0 & -\frac{z_{1i}}{2C_a} & 0 \\ \frac{z_{1i}}{2L} & -\frac{1}{L}(R_b + R_L) & -\frac{1}{L} \\ 0 & \frac{1}{C_b} & 0 \end{bmatrix}, B_i = \begin{bmatrix} -\frac{z_{3i}}{2C_a} & 0 & \frac{1}{C_a} \\ \frac{z_{2i}}{2L} & -\frac{R_b}{L} & 0 \\ 0 & -\frac{1}{C_b} & 0 \end{bmatrix} \tag{43}$$

In the next table are presented the fuzzy rules which help us define the system described in (36).

Table 3-1 Fuzzy rules

i	Fuzzy sets			Parameters of then-part		
	F_{1i}	F_{2i}	F_{3i}	z_{1i}	z_{2i}	z_{3i}
1	f_{a1}	f_{a2}	f_{a3}	zM1	zM2	zM3
2	f_{a1}	f_{a2}	f_{b3}	zM1	zM2	zm3
3	f_{b1}	f_{a2}	f_{a3}	zm1	zM2	zM3
4	f_{b1}	f_{a2}	f_{b3}	zm1	zM2	zm3
5	f_{a1}	f_{b2}	f_{a3}	zM1	zm2	zM3
6	f_{a1}	f_{b2}	f_{b3}	zM1	zm2	zm3
7	f_{b1}	f_{b2}	f_{a3}	zm1	zm2	zM3
8	f_{b1}	f_{b2}	f_{b3}	zm1	zm2	zm3

3.4 Lyapunov stability

The T-S observer given by (35) is

$$\begin{cases} \dot{\hat{x}}(t) = \sum_{i=1}^{r} h_i\big(\hat{z}(t)\big)\big\{A_i\hat{x}(t) + B_i u_{inputs}(t) + L_i(\Delta y)\big\} \\ \hat{y}(t) = H\hat{x}(t) \end{cases} \tag{44}$$

where $\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} V \\ I_L \\ V_b \end{bmatrix}$, with x_2, x_3 measured and x_1 estimated ,while Δy is the error between the measured signal and estimated signal.

The stability of the observer is assured using a Lyapunov stability method [2,6]. This implies the use of a Lyapunov function such as:

$$\begin{cases} Q = e^T P e \\ \mathrm{e} = \mathrm{x} - \hat{x} \\ P = P^T, P > 0 \end{cases} \tag{45}$$

In order to obtain a bounded error that converges towards zero, the derivative of the function must be negative.

$$\dot{Q} = \dot{e}^T P e + e^T P \dot{e} < 0 \tag{46}$$

The derivative of the error must be computed, using eq. 39 and eq. 42:

$$\begin{aligned} \dot{e} = \dot{x} - \dot{\hat{x}} = &\sum_{i=1}^{r} [h_i(z) - h_i(\hat{z})]\big\{A_i x + B_i u_{inputs}\big\} + \\ &+ \sum_{i=1}^{r} h_i(\hat{z})\{-A_i\hat{x} + A_i x - L_i H(x - \hat{x})\} \end{aligned} \tag{47}$$

$$= \sum_{i=1}^{r} h_i(\hat{z})\{A_i e - L_i H e\} + \omega$$

We consider $\omega = \sum_{i=1}^{r}[h_i(z) - h_i(\hat{z})]\{A_i x + B_i u_{inputs}\}$ as a virtual perturbation that is rejected when $\hat{z} \to z$.

The following quadratic stability condition [3] is used:

$$\dot{Q} + e^T R e - \zeta^2 \omega^T \omega + 2 \propto Q < 0 \tag{48}$$

with $R = R^T$, $R > 0$, $\zeta > 0$, and the decay rate $\propto > 0$.

We reintroduce $eq.$ 44 in eq. 46 :

$$\dot{e}^T P e + e^T P \dot{e} + e^T R e - \zeta^2 \omega^T \omega + 2 \propto Q < 0 \tag{49}$$

And then we replace the value of $\dot{e}$ from eq. 45 in eq. 47 and we obtain:

$$\left(\sum_{i=1}^{r} h_i(\hat{z})\{A_i e - L_i H e\} + \omega\right)^T P e + e^T P \left(\sum_{i=1}^{r} h_i(\hat{z})\{A_i e - L_i H e\} + \omega\right) + e^T R e - \zeta^2 \omega^T \omega + 2 \propto e^T P e \ < 0 \tag{50}$$

$$e^T \sum_{i=1}^{r} h_i(\hat{z})\left\{\left(A_i{}^T - H^T L_i{}^T\right)P + P(A_i - L_i H) + R + 2 \propto P\right\} e + \omega^T P e + e^T P \omega - \zeta^2 \omega^T \omega < 0 \tag{51}$$

The next inequality is used:

$$x^T y + y^T x \leq \lambda x^T x + \lambda^{-1} y^T y, \forall\, \lambda > 0 \tag{52}$$

If we consider $= \zeta^2$, $x = \omega$ and $y = Pe$, eq. 50 becomes:

$$\omega^T P e + e^T P \omega \leq \zeta^2 \omega^T \omega + \zeta^{-2} (Pe)^T (Pe) \tag{53}$$

The left term of the inequality will be replaced by the right term in eq. 49:

$$e^T \sum_{i=1}^{r} h_i(\hat{z})\left\{\left(A_i{}^T - H^T L_i{}^T\right)P + P(A_i - L_i H) + R + \ \ 2 \propto P\right\} e + \zeta^2 \omega^T \omega + \zeta^{-2} e^T P P e - \zeta^2 \omega^T \omega < 0 \tag{54}$$

It can be noticed that the term containing the virtual perturbation is reduced.

$$e^T \sum_{i=1}^{r} h_i(\hat{z})\left\{\left(A_i{}^T - H^T L_i{}^T\right)P + P(A_i - L_i H) + R + 2 \propto P + \zeta^{-2} P^2\right\} e < 0 \tag{55}$$

If we apply the Schur complement lemma we obtain:

$$\begin{bmatrix} \left(A_i{}^T - H^T L_i{}^T\right)P + P(A_i - L_i H) + R + 2 \propto P & P \\ P & -\zeta^2 \end{bmatrix} < 0 \tag{56}$$

For obtaining the gains of the observer this linear matrix inequality (LMI) must be solved. This can be easily done solving the LMI with the help of the open source Yalmip toolbox.

After computing the gains L_1 and L_2, we implement in a simulation environment, such as Matlab/Simulink, the scheme from Figure 2-19 along with the T-S observer. All the presented results are obtained using a variable sampling step. The estimated voltage output of the PV panel approximates well the measured parameter in a short time (Figure 3-). The estimation error converges towards a bounded interval (Figure 3-4) and offers an accuracy superior to 97%.

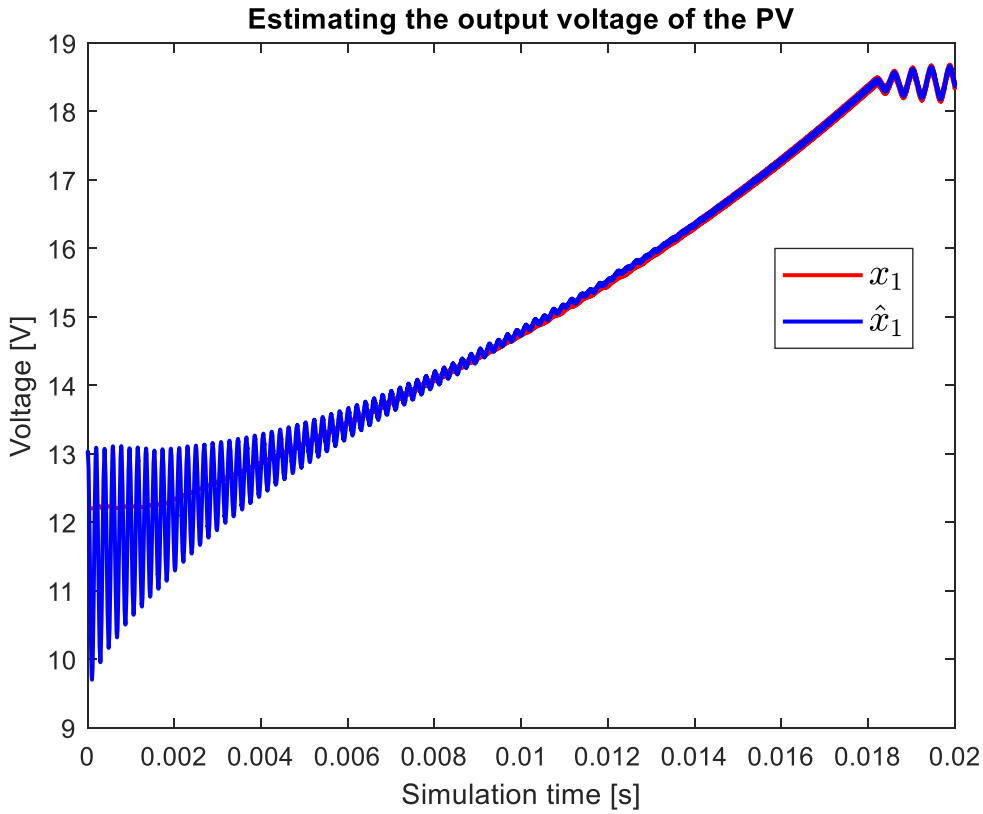


Figure 3-3 Estimating the output voltage of the PV

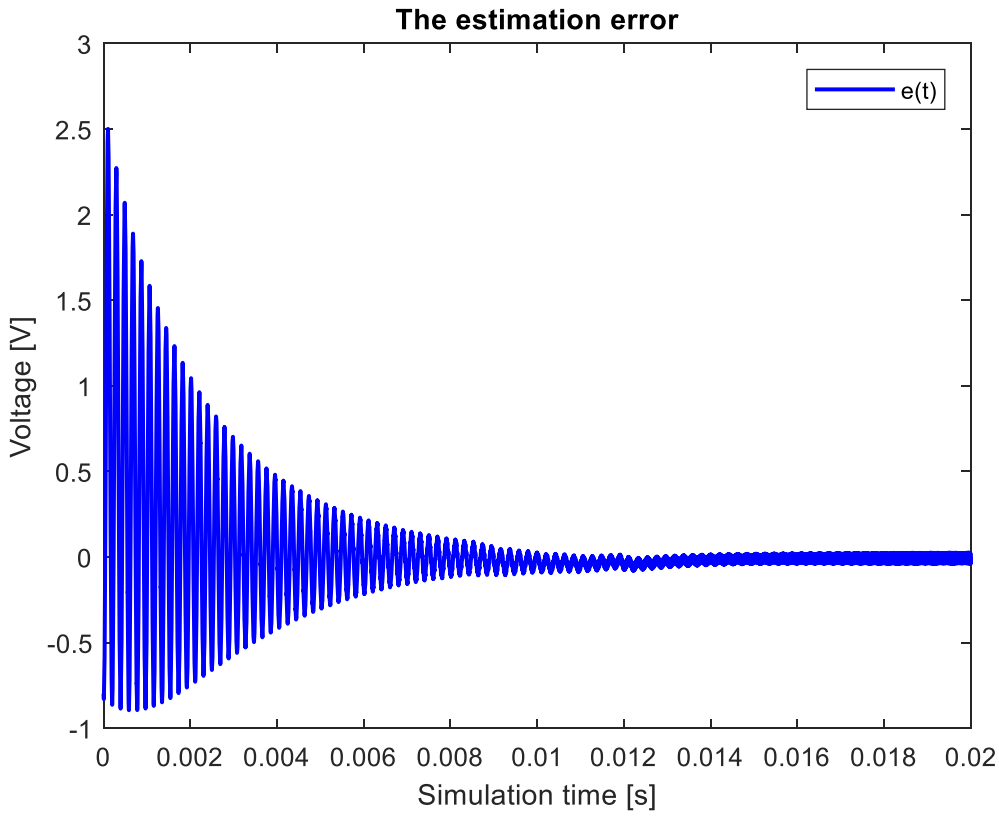


Figure 3-4 The estimation error

3.5. Computing the maximum power point tracking

In order to benefit of the maximum power of the PV panel a MPPT control algorithm will be used. Perturb and observe (P&O) algorithm is a popular one and can be found on many controllers used in actual industry. It is easy to implement and has a very simple concept that is described in Figure 4-1. Details about implementation and how the algorithm works can be found the chapter 4.

Considering that the dynamic of the observer is faster than that of the control, we can use the estimated voltage computed in real-time with the T-S observer for the MPPT control block. In this way we replace the voltage sensor [7] with a software solution or even integrate a fault detection of that sensor.

Using this algorithm the maximum power of the PV panel is achieved under variable irradiance and temperature conditions as it is presented in Figure 3-3.

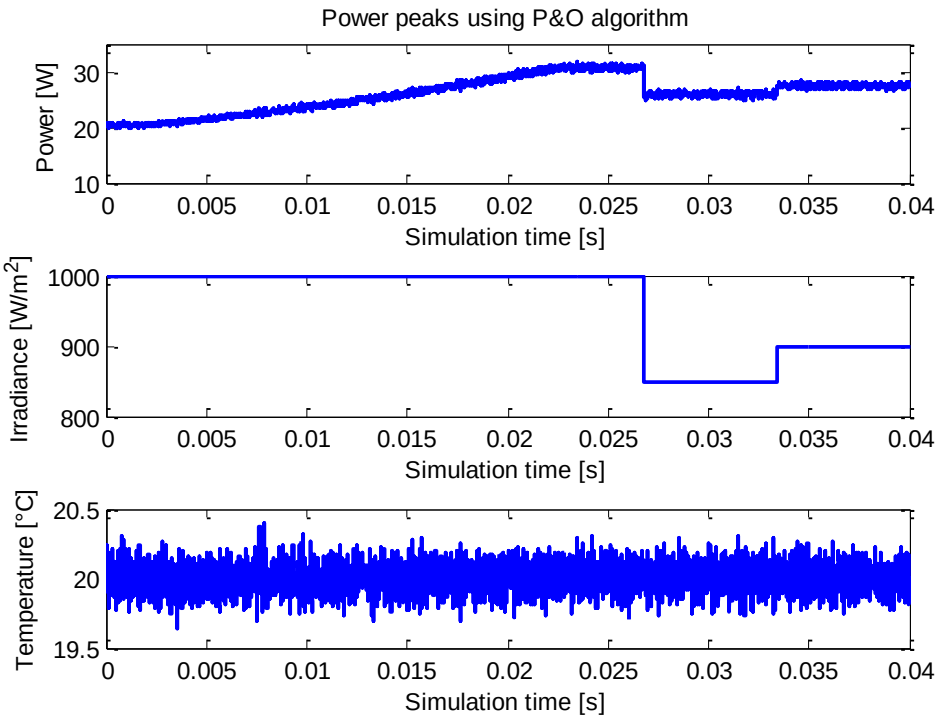


Figure 3-3 Power peaks using P&O algorithm

4. (MPPT) CONTROL ALGORITHMS

4.1. Classic control algorithms

As presented in chapter 2, the performances of a PV panel are affected by meteorological conditions, such as irradiance, temperature, air mass spectrum and shadowing. Furthermore, a step down converter architecture is used in order to maximize the quantity of produced energy of the solar panel.

In order to achieve this latter objective, several maximum power point tracking (MPPT) control algorithms are proposed. Some of the most popular are the hill climbing based techniques. As the name suggests, these algorithms are using the power-voltage characteristic curve. Among them, we shall remind “Perturb and Observe” (P&O), “Incremental conductance” (IC).

The P&O algorithm and different adaptions are widely implemented, due to its simplicity. The system is disturbed at each sampling time and its performances are evaluated.

The duty ratio u that is used to control the high mosfet of the buck converter is increased or decreased, and then the voltage and the power of the PV panel are measured. Comparing the obtained values with the previous ones we can establish if the power of the PV panel is increasing or decreasing, and how the duty ratio must be adjusted with respect to our goal. By modifying u we also modify the value of the output voltage V.

One of the disadvantages of this technique is that once it finds the maximum power point, it will oscillate around it with an amplitude given by the value of the step size. If this step is fixed, then a pre-calibration under different testing conditions is required. It is also sensible to shadowing effect and variable load changes.

In order to construct de MPPT algorithm, a P&O method was implemented based on the logic:

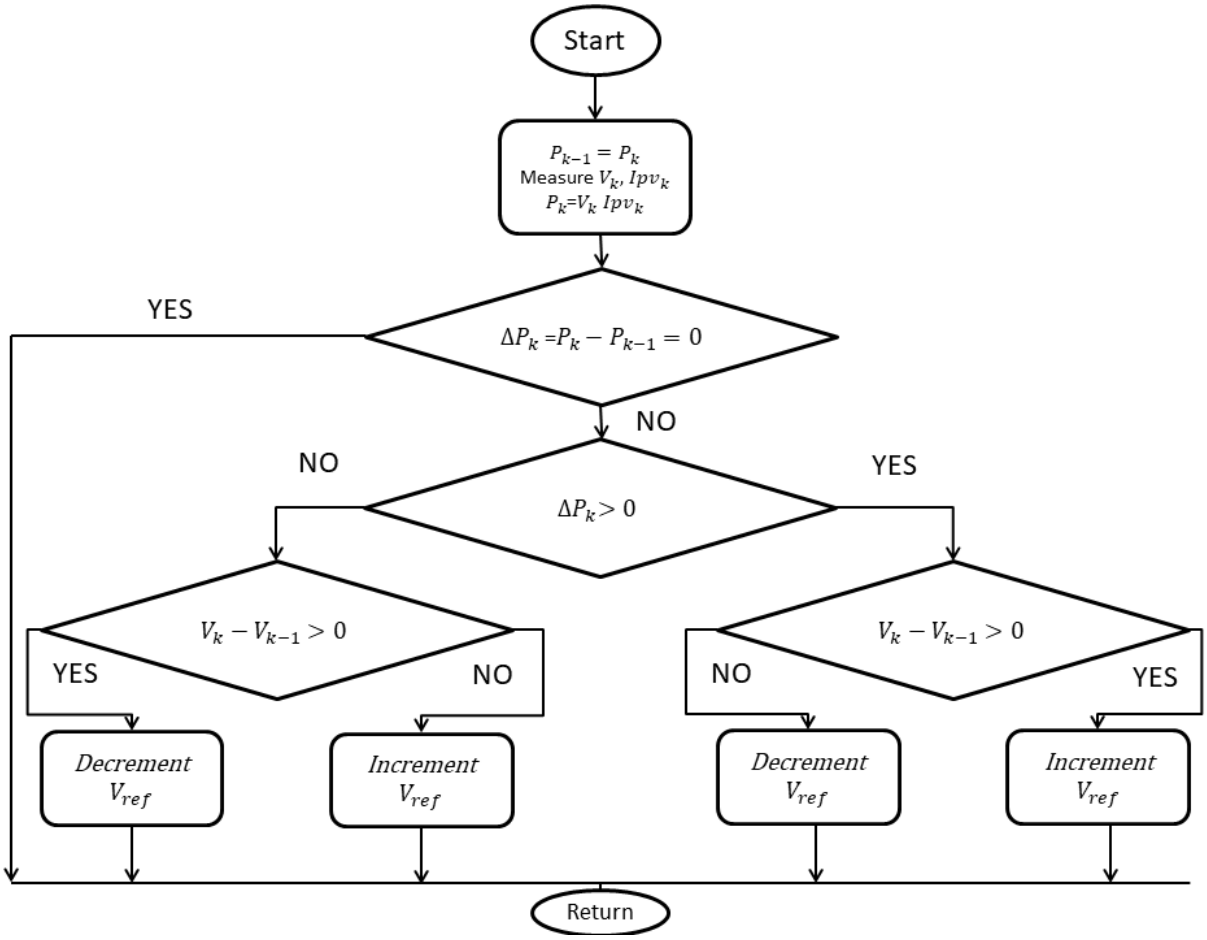


Figure 4-1 The P&O algorithm

The P&O algorithm described in [4,5,6,7] has the advantage of simplicity with satisfying results in tracking.

The IC algorithm is another popular algorithm. It computes the control law with respect to eq.14. If $\frac{dP}{dV} > 0$, the duty ratio increases, else if $\frac{dP}{dV} < 0$, the duty ratio decreases, otherwise the duty maintains its previous value, as presented in Figure 4-2.

It has the advantage of eliminating the oscillations around the maximum power point. One of its drawbacks is that is requires more computing resources.

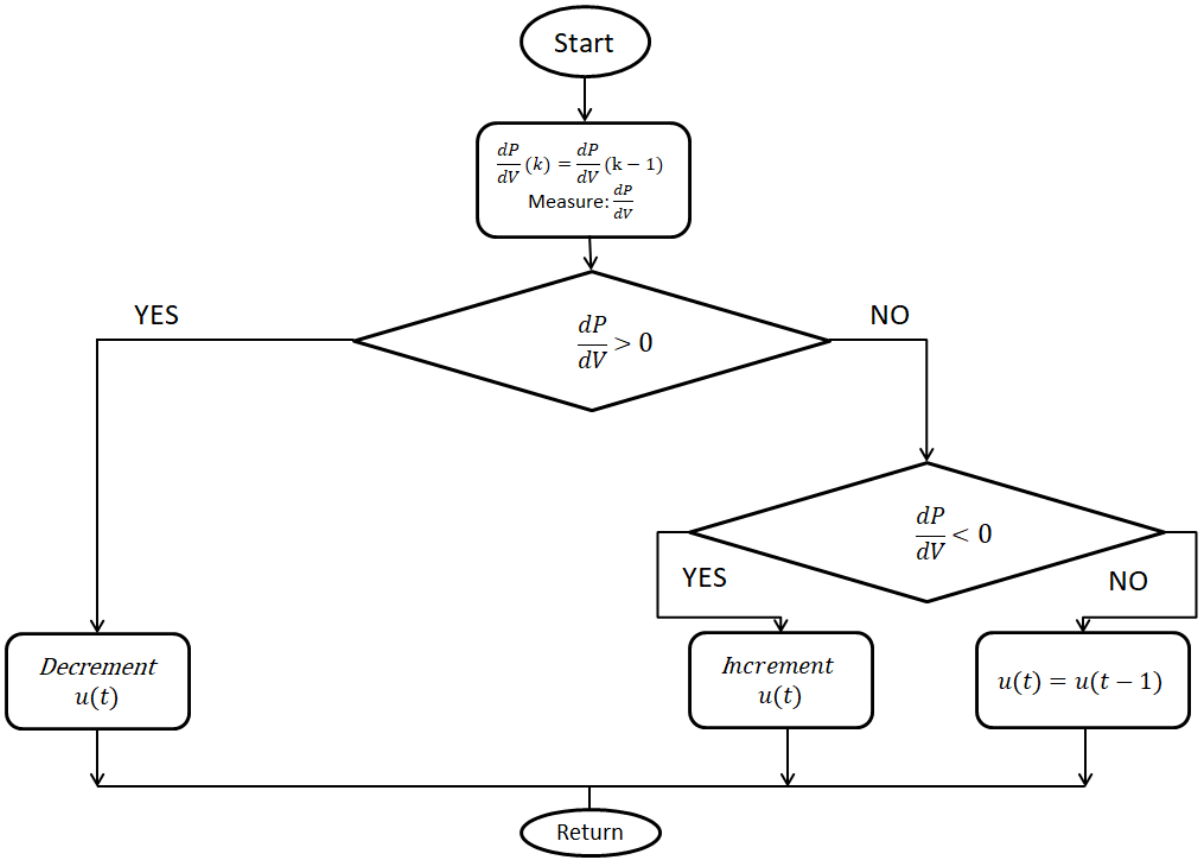


Figure 4-2 The IC algorithm

The next configuration is proposed for the DC-DC converter: the PWM signal will be sent at a frequency of 100kHz, while its inductance $L = 100$ [μH], the internal resistance of the inductor $R_L = 0.1$ [Ω], the capacitance $C_b = 22$ [μF], and the internal resistance of the capacitor $R_b = 1$ [Ω].

A slight advantage can be already noticed. The amplitude of the oscillation when maximum power area is reached is smaller when IC algorithm is used. The difference increases if the step size is chosen for a rather faster conversion rate towards the MPPT. In Figure 4-3 are compared the two algorithms, P&O and IC, with two fixed steps: step1=0.5e-5 and step2=1e-6.

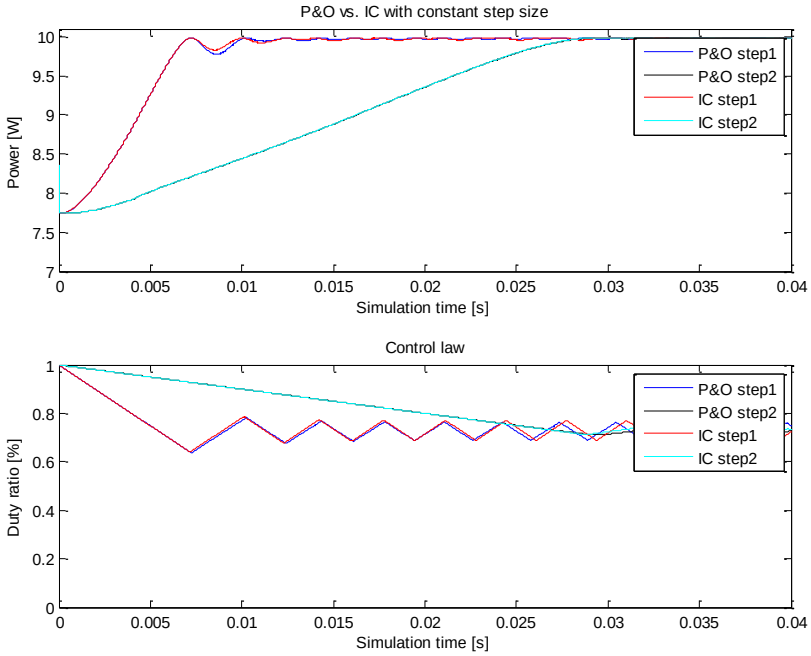


Figure 4-3 Comparing P&O with IC algorithm

The duty ratio is similar for the scenario which uses step1 as step size.

In order to decrease the time needed for finding the maximum power point a variable step time can be used. In several paperwork [7] this objective is achieved by multiplying a constant step size with $\frac{dP}{dV}$.

In spite of the faster initial convergence rate (Figure 4-3) the result is similar to one obtained with IC with constant step size.

An improved IC algorithm is proposed in the following part.

Increasing or decreasing the duty ratio u is accomplished by adding or subtracting a certain value, often called step, from the previous duty ratio u. As presented earlier, if the step size is fixed, then the standard IC algorithm is used. If the step size is variable, such as $step = \frac{dP}{dV} * constant$, then a IC with variable step size is used.

However, an improvement can be added by combining the two methods with respect to the dynamic of the step size.

Measuring the last three values of $\frac{dP}{dV}$ provides information about the search direction:

$$sign\left(\left|\frac{dP(t)}{dV}\right| - \left|\frac{dP(t-1)}{dV}\right|\right) * \left(\left|\frac{dP(t-1)}{dV}\right| - \left|\frac{dP(t-2)}{dV}\right|\right) \quad (57)$$

If eq. 55 is greater than or equal to zero and $\left|\frac{dP}{dV}\right| < err$, then the step size becomes:

$$step(t) = step(t-1) * \text{ratio} \quad (58)$$

where ratio is a finite number and err is the domain in which the acceleration of the search algorithm is increased.

If $\left|\frac{dP}{dV}\right| >$ err the step size becomes constant (c1).

Otherwise if eq. 55 is smaller than zero the step size becomes constant (c2).

For better performances it is advised that (c1) is smaller than (c2).

In Figure 4-4 are compared the above mentioned cases, with three different step sizes.

It can be observed that the improved IC algorithm has a faster conversion rate and oscillations with really small amplitude, aspect that can be confirmed by the control law.

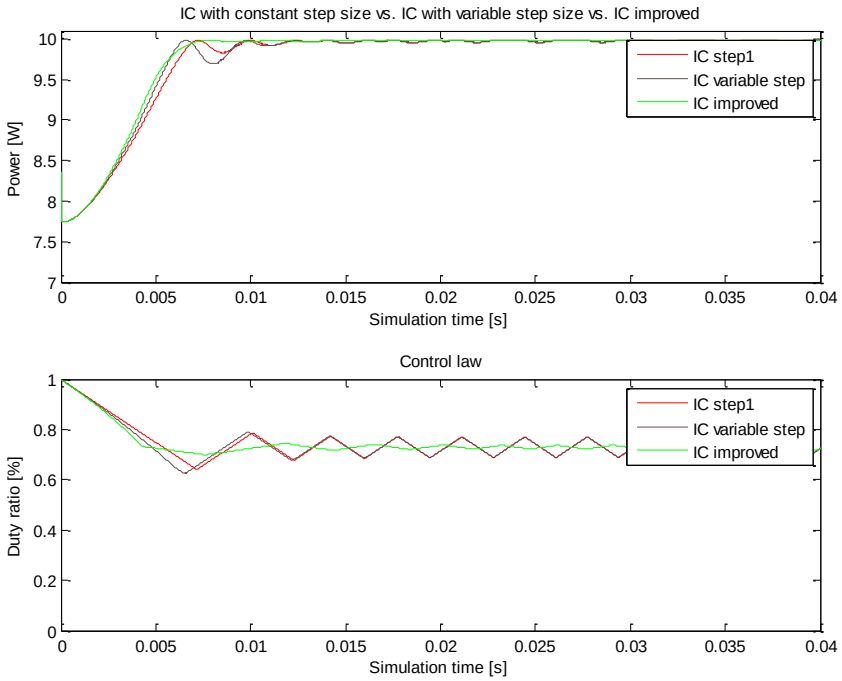


Figure 4-4 Comparing improved IC with other IC algorithms

4.2 Advanced control algorithms (robust polynomial algorithms)

4.2.1 RST regulator for the Buck Converter

4.2.1 RST regulator for the Buck Converter

In Figure 4-5 is presented the general schematic of the architecture. In this simulation, the main load connected to the buck converter is a resistive load (as a heating element or light bulbs). To maintain the continuity of the schematic between the two converters, an auxiliary resistance is added. This can represent a cooling unit or local lighting.

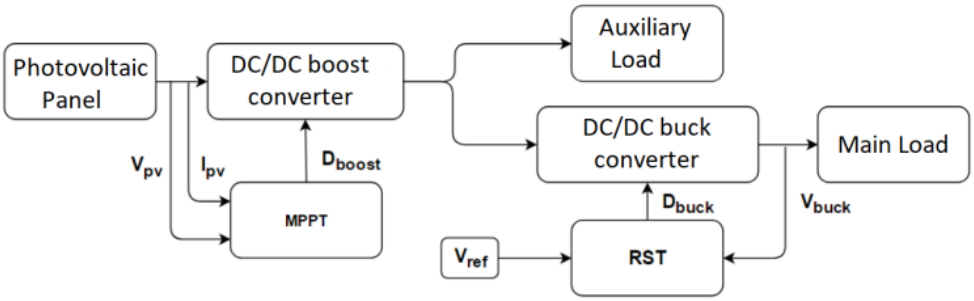


Figure 4-5 The overall system architecture

Considering the transistor in the buck converter, its switching affects the boost converter as well, as a continuous variation of the equivalent impedance. As such, the current oscillates, but its mean value remains constant for a constant input voltage.

The buck converter works with an 32Khz frequency command. The inductance value is chosen accordingly, with a value of 160 [µH]. The condensers, at the output we have 220 [µF], and at the input 100 [µF] to smoothen the input.

The transfer function of the process was presented in eq. 32:

$$H_p(s) = \frac{V}{s^2 L C_b + s\frac{L}{R} + 1} \tag{59}$$

where $L = 160$ [µH], $C_b = 220$ [µF] and $V = 22$ [V].

The mathematical model of the discretized process is an auto-regressive model with exogenous input (ARX).

After using the "zero order hold" discretization method [55] the following discrete transfer function of the model is obtained for a sampling time of 62.5 µs:

$$H_p(z^{-1}) = \frac{z^{-2}(\,1.151132\,) + z^{-1}(1.162124)}{z^{-2}(\,0.971991) + z^{-1}(\,-1.866843) + 1} \tag{60}$$

The dynamic of the process can be observed in figure presented below.

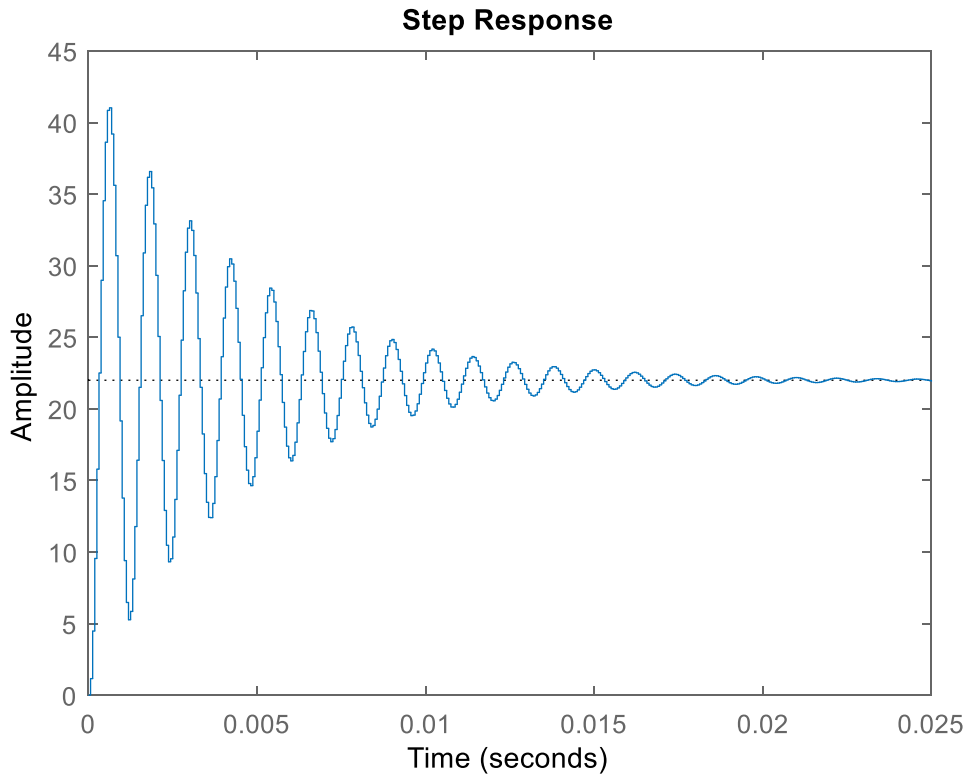


Figure 4-6 Step response of the plant model

A polynomial R-S-T algorithm ([53],[55],[56]) is proposed for controlling the plant model above. The demanded performances are adjusted through some trials simulating the behavior of the global system, including the photovoltaic generator, the converters, variable solar irradiation and temperature.

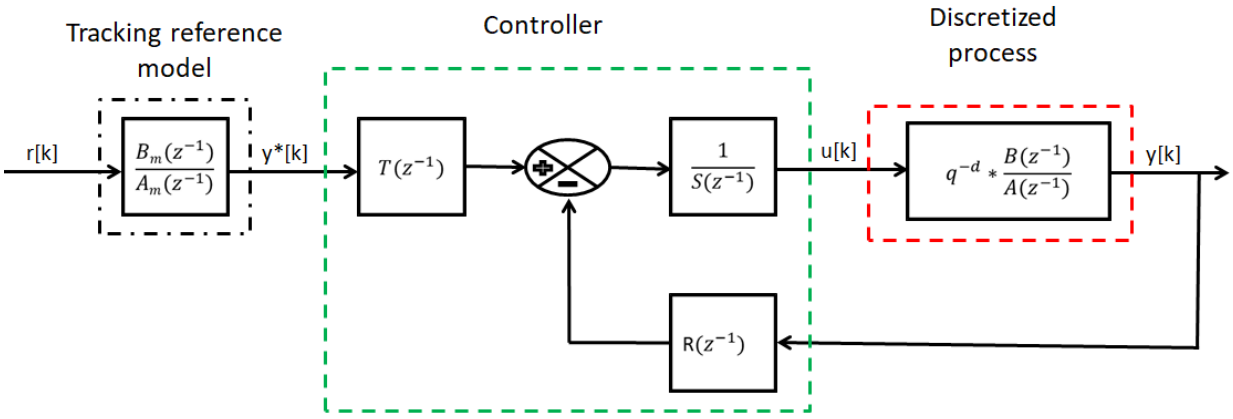


Figure 4-7 RST feedback diagram

This control structure takes into account the discretized transfer function of the process, the R-S-T polynomials used for controlling the plant in closed loop and rejecting the disturbances, and the trajectory generator which imposes a desired trajectory.

The general RST command form is the following:

$$u(k) = \frac{T(q^{-1})}{S(q^{-1})} r(k) - \frac{R(q^{-1})}{S(q^{-1})} y(k), \qquad \forall\, k \in \mathbb{N} \tag{61}$$

Where the polynomials of the controller are defined as below:

$$\begin{cases} R(z^{-1}) = r_0 + r_1 z^{-1} + \ \cdots + \ r_{nr}\, z^{-nr} \\ S(z^{-1}) = 1 + s_1\, z^{-1} + \ \cdots + \ s_{ns}\, z^{-ns} \\ T(z^{-1}) = t_0 + t_1\, z^{-1} + \ \cdots + \ t_{nt}\, z^{-nt} \end{cases} \tag{62}$$

The polynomials of the plant are defined as below:

$$\begin{cases} B(z^{-1}) = b_1\, z^{-1} + b_2\, z^{-2} + \ \cdots + \ b_{nb}\, z^{-nb} \\ A(z^{-1}) = 1 + a_1\, z^{-1} + \ \cdots + \ a_{na}\, z^{-na} \end{cases} \tag{63}$$

The coefficients of the plant can be substituted with the values obtained in eq. 58.

Further we can compute the transfer function of the closed loop system

$$H_{cl}(z^{-1}) = \frac{z^{-d}\, T(z^{-1})\, B(z^{-1})}{A(z^{-1})\, S(z^{-1}) + z^{-d}\, B(z^{-1})\, R(z^{-1})} = \frac{z^{-d}\, T(z^{-1})\, B(z^{-1})}{P(z^{-1})} \tag{64}$$

where the poles of the closed loop system are characterized by the polynomial:

$$P(z^{-1}) = p_1 * z^{-1} + p_2 * \ z^{-2} + \ \cdots + \ p_{nb} * \ z^{-np} \tag{65}$$

The polynomial P contains the dominant and auxiliary poles of the closed loop system:

$$P(z^{-1}) = \ A(z^{-1})\, S(z^{-1}) + z^{-d}\, B(z^{-1})\, R(z^{-1}) = P_D(z^{-1}) P_F(z^{-1}) \tag{66}$$

$P_D(z^{-1})$ corresponds to the dominant closed loop poles of the system, and is responsible with for performances of the controller. $P_F(z^{-1})$ stands for the auxiliary poles of the closed loop system, which are used for filtering purposes, for the improving the robustness of the system. As mentioned in [30], the auxiliary poles are faster than the dominant poles, meaning that the a real part of the roots of $P_F(z^{-1})$ is inferior to one of $P_D(z^{-1})$.

A second degree transfer function is usually used for imposing the dominant poles, thus the desired behavior of the closed loop system.

$$H_{imposed}(s) = \frac{{\omega_0}^2}{s + 2\zeta\omega_0 s + {\omega_0}^2} \tag{67}$$

where , ζ is the damping factor and $\omega_0$is the natural frequency.

Taking into account the value of the sampling period (62.5 μs), a damping factor $\zeta = 0.8$ and a natural frequency $\omega_0 = 10000$rad/s are chosen.

After the discretization we obtain:

$$H_{imposed}(z^{-1}) = 1 - \ 1.164793z^{-1} + 0.416862z^{-2} \tag{68}$$

Then we impose these poles to the polynomial P:

$$P(z^{-1}) = H_{imposed}(z^{-1}) \tag{69}$$

Furthermore, an integrator is added in the *Hs* imposed part of the polynomial *S* in order to obtain a null steady state error.

$$H_S(z^{-1}) = 1 - 1z^{-1} \tag{70}$$

The coefficients of the R and S polynomials are determined by solving the next equation:

$$x = M^{-1}p \tag{71}$$

where:

$$\begin{aligned} x^T &\stackrel{\text{def}}{=} [1\ s_1\ s_2\ \dots\ s_{nb+d}\ r_0\ r_1\ \dots\ r_{na-1}] \\ p^T &\stackrel{\text{def}}{=} [1\ p_1\ p_2\ \dots\ p_{na+nb+d}] \end{aligned} \tag{72}$$

and $M_{[na+nd+d+1],[na+nb+d+1]}$ is the Sylvester matrix, associated to the coprime polynomials A and B.

The values are computed using the algorithm presented in section 7.2.:

$$R(z^{-1}) = 0.228370\ - 0.409277z^{-1} + 0.192297z^{-2} \tag{73}$$

$$S(z^{-1}) = 1\ - 1.007634z^{-1} + 0.007634z^{-2} \tag{74}$$

The coefficients of the T polynomial are determined from the following expression:

$$T(z^{-1}) = K * P(z^{-1}) \tag{75}$$

where:

$$K(z^{-1}) = \begin{cases} \dfrac{1}{B(1)}, B(1) \neq 0 \\ 1, \quad B(1) = 0 \end{cases} \tag{76}$$

After introducing eq. x in y we obtain:

$$\begin{aligned} T(z^{-1}) = 0.432291 - 1.127883\ z^{-1} + \ 1.144695\ z^{-2} - 0.536612\ \ z^{-3} \\ + 0.098899z^{-4} \end{aligned} \tag{77}$$

The transfer function of the tracking reference model is:

$$H_m(z^{-1}) = \frac{z^{-1}\ B_m(z^{-1})}{A_m(z^{-1})} = \frac{z^{-1}\ (b_{m1} + b_{m2}\ z^{-1})}{1 + a_{m1}z^{-1} + a_{m2}\ z^{-2}} \tag{78}$$

To obtain the polynomials *Am* and *Bm* of the trajectory generator, another transfer function of second degree is used. A damping factor $\zeta = 0.8$ and a natural frequency $\omega_0 = 12000$rad/s are chosen.

The same procedure is applied as for computing the polynomials *R* and *S*.

$$B_m(z^{-1}) = 095495\ z^{-1} + \ 0.073072z^{-2} \tag{79}$$

$$A_m(z^{-1}) = 1 - 1.280762\ z^{-1} + 0.449329\ z^{-2} \tag{80}$$

After obtaining all the parameters of the above mentioned polynomials the performances of the control algorithm are tested in simulation, as presented in Figure 4-8 and in Figure 4-9 .

Firstly, the closed loop control is tested in WinReg software for reference tracking and rejection of the disturbances.

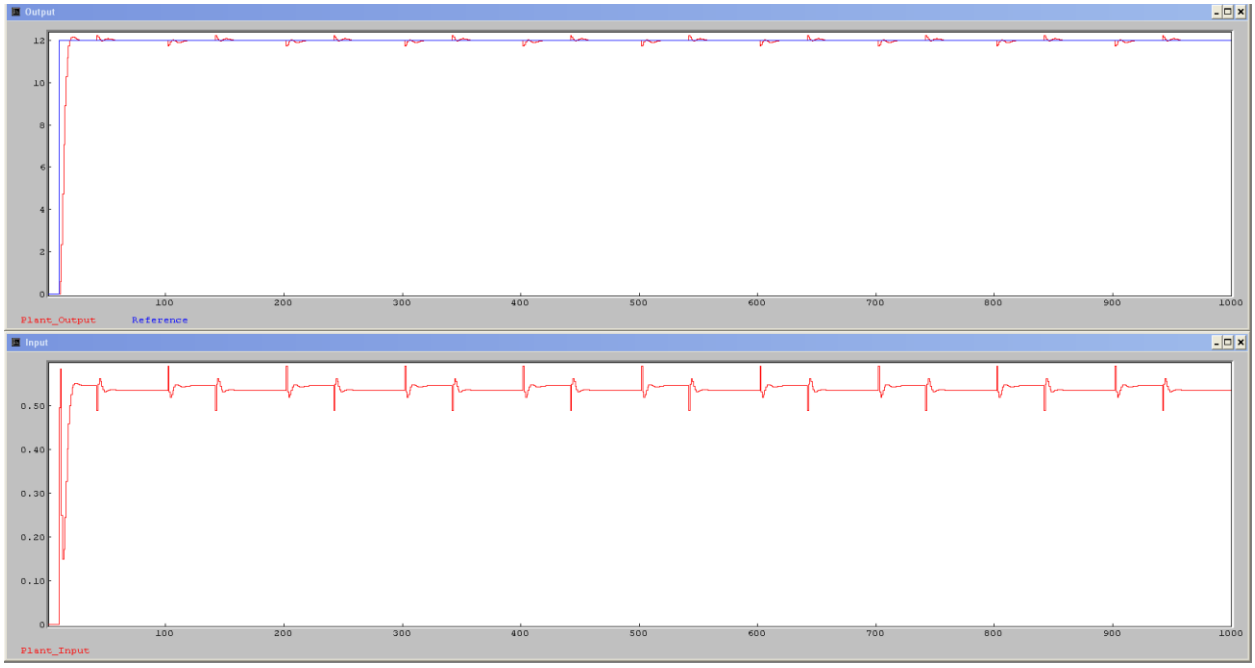


Figure 4-8 Reference tracking and rejecting of disturbances in WinReg

Secondly, the control loop is tested on the model of the DC/DC buck converter.

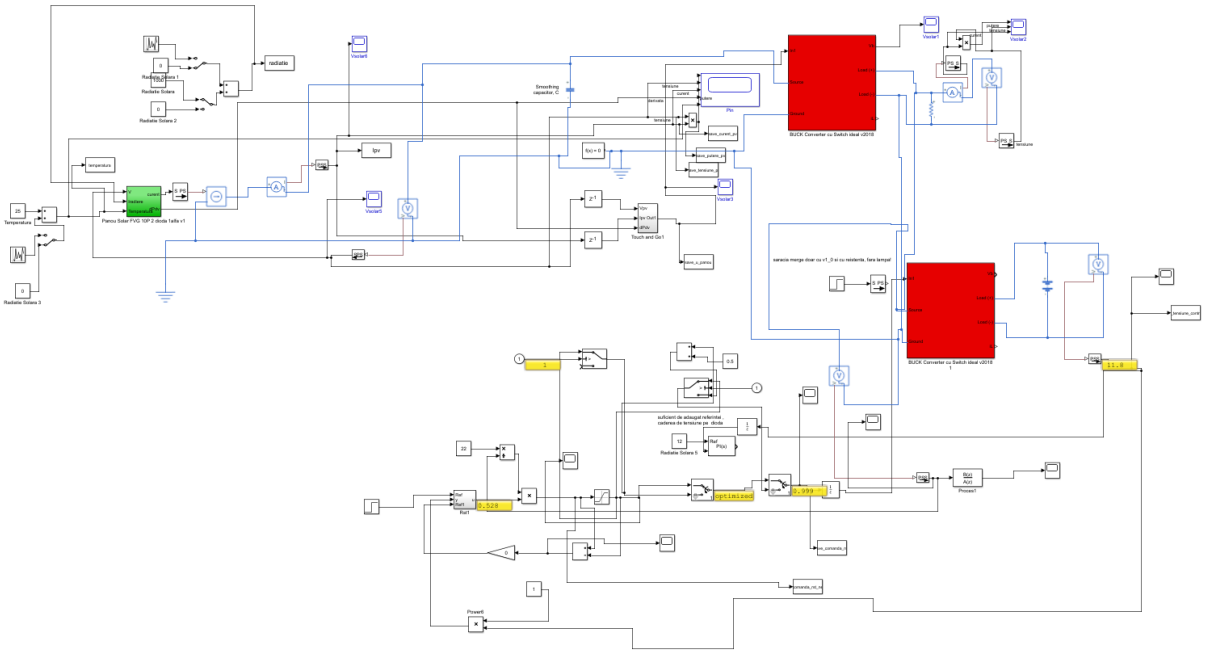


Figure 4-9 Validating control loop in Simulink

During the simulation, two types of disturbances are being applied to the control loop. After the reference is reached, the element of execution, the transistor from the Sziklai pair, is being disconnected from the controller. Later on, an additive disturbance is applied to the control law.

We can observe that in Figure 4-10 that the MPPT control algorithm is harvesting the maximum power that the PV can provide, with respect to the second control loop which perturbs it as well.

Furthermore we can observe the two types of disturbances proposed earlier. The first one which is triggered during the time interval [0.4s,0.41s] is not well rejected, because the control law is saturated at maximum value, while the static error is superior to 13.3%. The second one which is triggered during the time interval [0.8s,0.85s] is well rejected.

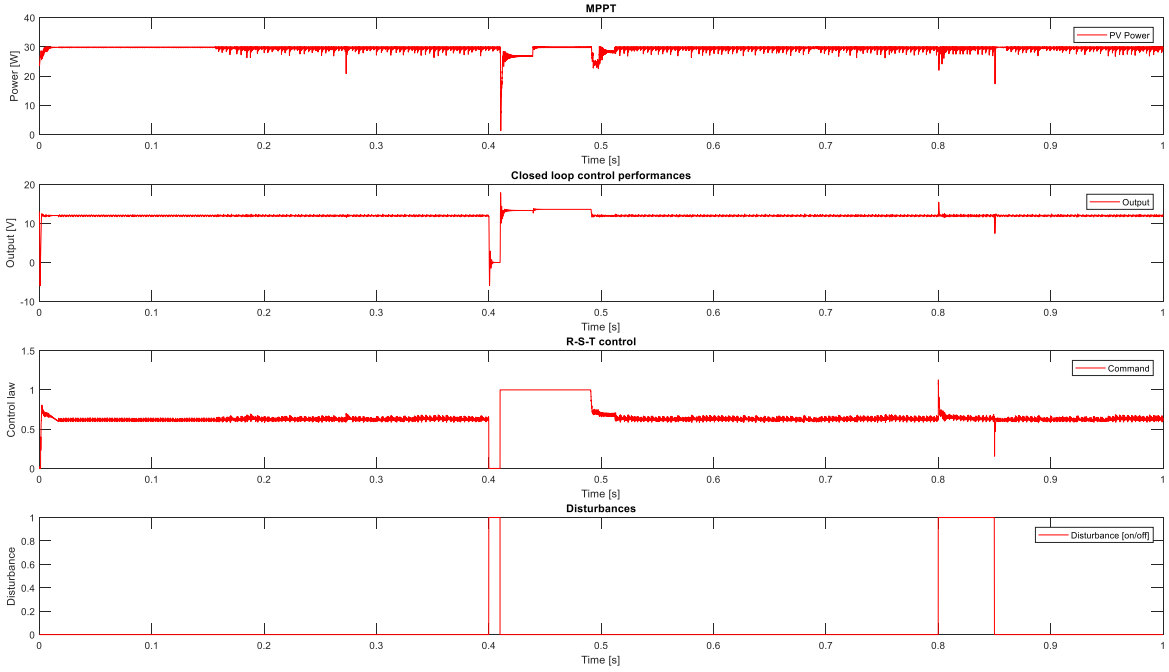


Figure 4-10 Closed loop control with the R-S-T control polynomial

4.2.2 Robust R-S-T control algorithm

An output sensitivity functions is used to analyze the performances of the controller:

$$S_{yp}(z^{-1}) = \frac{A(z^{-1}) * S(z^{-1})}{A(z^{-1}) * S(z^{-1}) + z^{-d} * B(z^{-1}) * R(z^{-1})} \tag{81}$$

We shall consider the margin modulus:

$$\Delta M \stackrel{\text{def}}{=} \min_{\omega \in R} \left|1 + H\left(e^{j\omega}\right)\right| \tag{82}$$

$$\Delta M|_{db} = -\max_{\omega \in R} \left|S_{yp}\left(e^{j\omega}\right)\right|\Big|_{db} \tag{83}$$

A bigger value of ΔM implies a better robustness. In order to achieve it, the value of *Syp* must decrease.

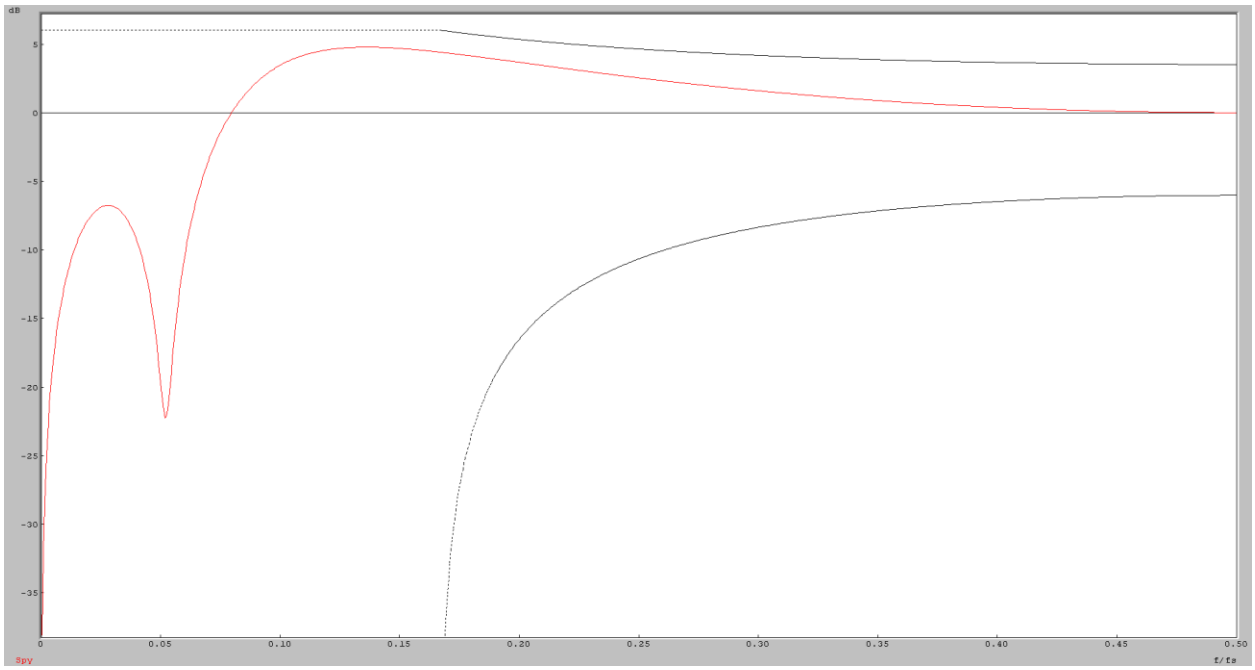


Figure 4-11 Sensitivity function – Syp (R-S-T)

In the Figure 4-11 is presented the sensitivity function *Syp*. After computing with the WinReg Robustness Analysis routine we obtain $\Delta M = -4.78\ dB$.

The performances in regulation and tracking can be improved tracking by introducing auxiliary pairs of poles/zeros and/or introducing pre-specifications for the *R* and *S* polynomials.

Furthermore we shall create a robust R-S-T polynomial controller.

An extra pair of complex poles to the tracking, imposing another second degree transfer function with a damping factor $\zeta = 0.8$ and a natural frequency $\omega_0 = 6000$rad/s

Poles and zeros of the system can be pre-specified in the polynomial H_S and H_R as presented below:

$$S = H_S * S'; R = H_R * R' \tag{84}$$

The following poles will be imposed:

$$H_S(z^{-1}) = (1 - z^{-1})(1 - 0.2z^{-1})(1 - 0.4z^{-1})(1 - 0.15z^{-1}) \tag{85}$$

The values of all the polynomial are recomputed:

$$R(z^{-1}) = 0.456929 - 1.129520z^{-1} + 1.032453z^{-2} - 0.420929z^{-3} + 0.077288z^{-4} - 0.004831z^{-5} \tag{86}$$

$$S(z^{-1}) = 1 - 1.273248z^{-1} + 0.085684z^{-2} + 0.256612z^{-3} - 0.074769z^{-4} + 0.005721z^{-5} \tag{87}$$

$$T(z^{-1}) = 0.432291 - 1.127883z^{-1} + 1.144695z^{-2} - 0.536612z^{-3} + 0.098899z^{-4} \tag{88}$$

$$B_m(z^{-1}) = 0.095495 + 0.073072z^{-1} \quad (89)$$

$$A_m(z^{-1}) = 1 - 1.280762z^{-1} + 0.449329\, z^{-2} \quad (90)$$

As for the first R-S-T computed controller, the closed loop control is tested in WinReg software for reference tracking and rejection of the disturbance for both robust and standard controller. In Figure 4-12 we can observe a slight improvement related to the rejection of the disturbances, with the tradeoff of having a higher amplitude of the control signal.

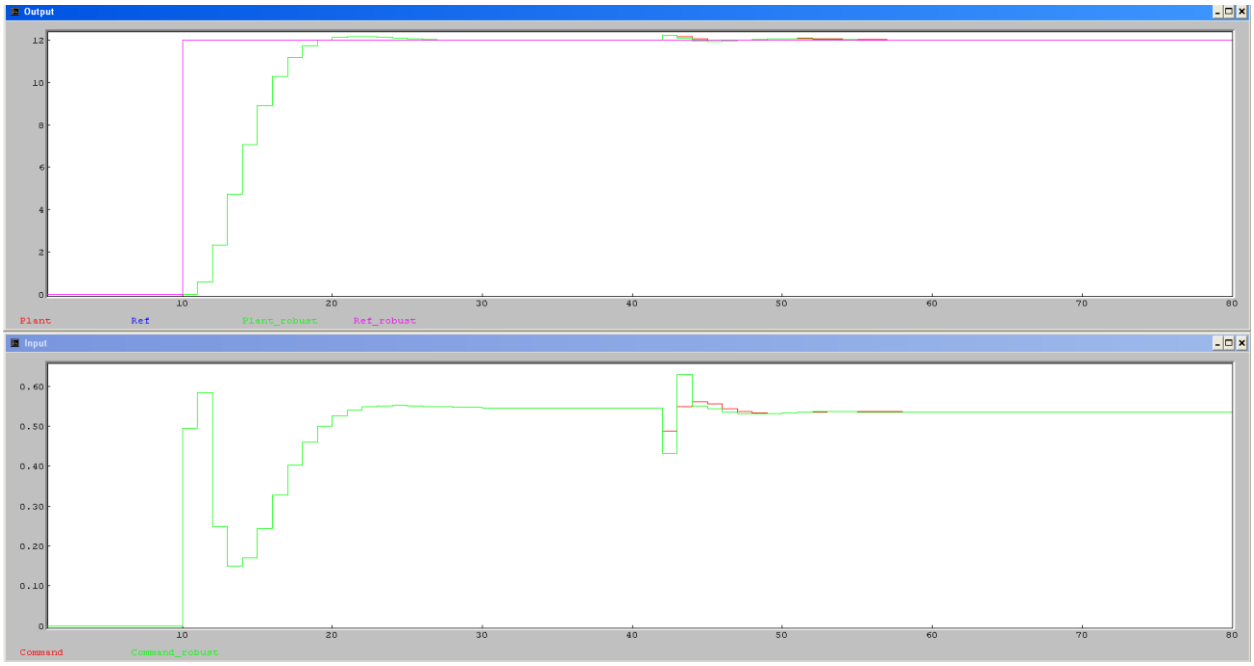


Figure 4-12 Reference tracking and rejecting of disturbances in WinReg for robust controller and standard controller

Next we shall analyze the sensitivity functions of the two controllers. As presented in Figure 4-13

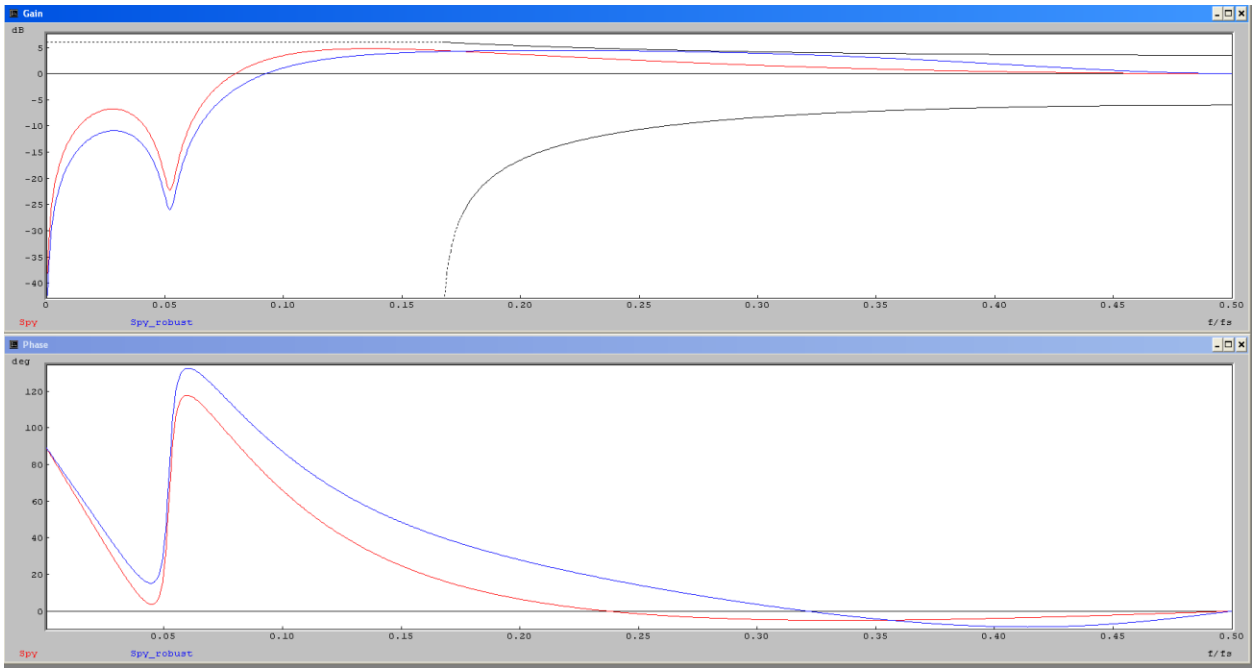


Figure 4-13 Sensitivity functions – Syp (R-S-T) of robust controller vs standard controller

An improved value of $\Delta M = -4.44\ dB$ is obtained.

Then, as for the previous regulator, the control loop is tested on the model of the DC/DC buck converter, as presented in the figure below.

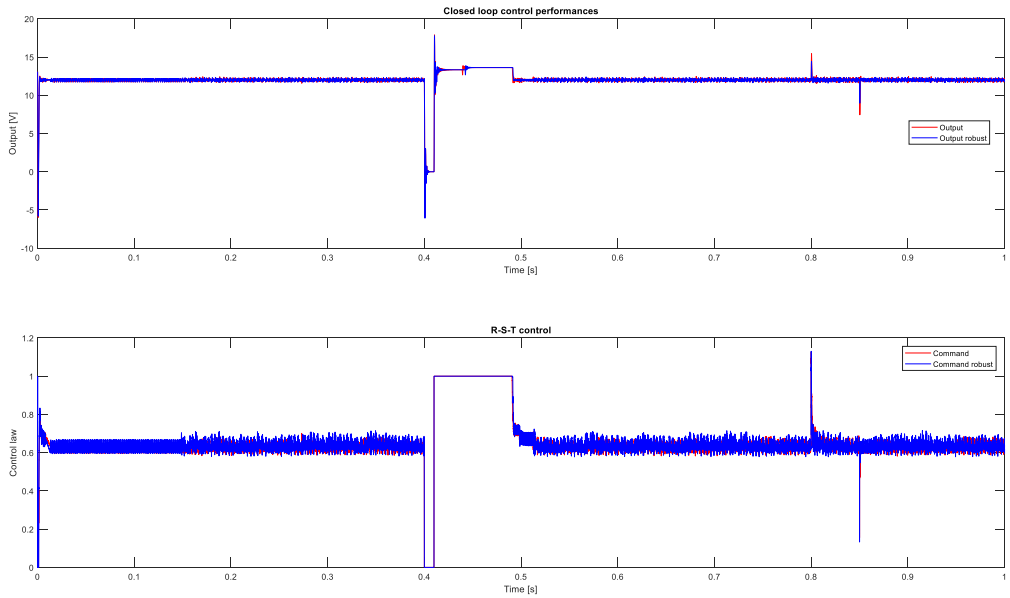


Figure 4-14 Closed loop control with the robust R-S-T control polynomial

We can observe in Figure 4-15 and Figure 4-16 that the rejection of both types of disturbances is slightly improved.

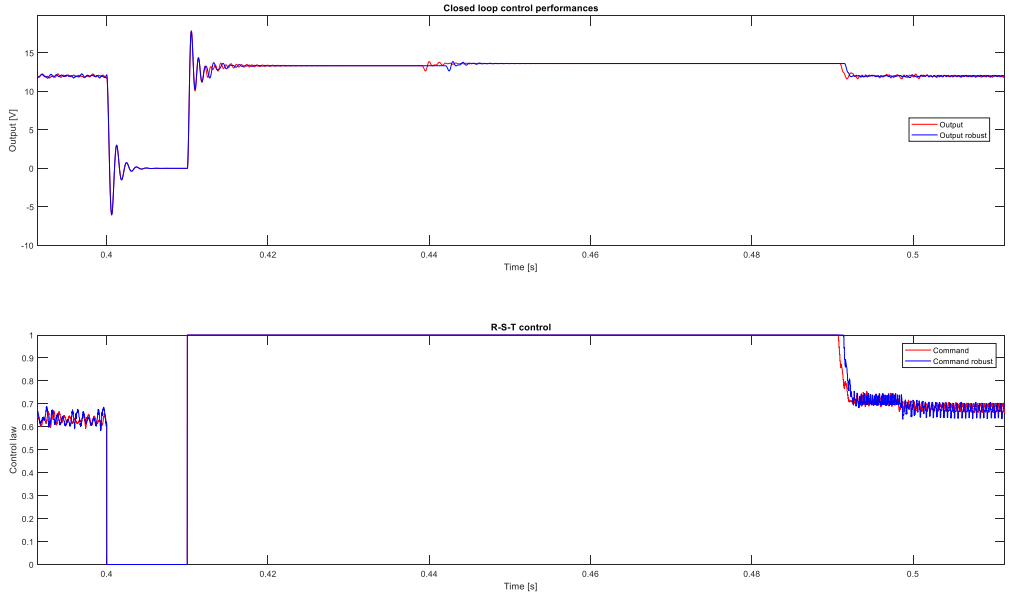


Figure 4-15 Rejection of disturbances (scenario I) – robust vs. standard RST

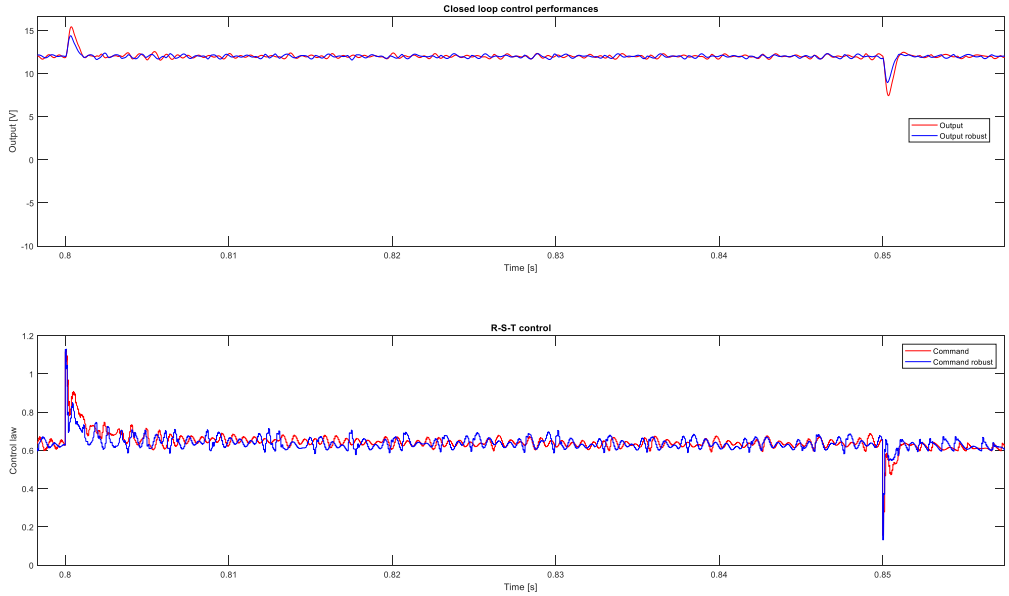


Figure 4-16 Rejection of disturbances (scenario II) – robust vs. standard RST

4.2.3 Robust R-S-T control algorithm with anti-windup

During the first type of perturbation we can observe that the command is saturated. Therefore an anti-windup technique can be used to avoid this kind of scenarios. The error between the control law generated by the controller and its value after the saturation block can be multiplied with a gain and be reintroduced in the control loop, as presented in Figure 4-17

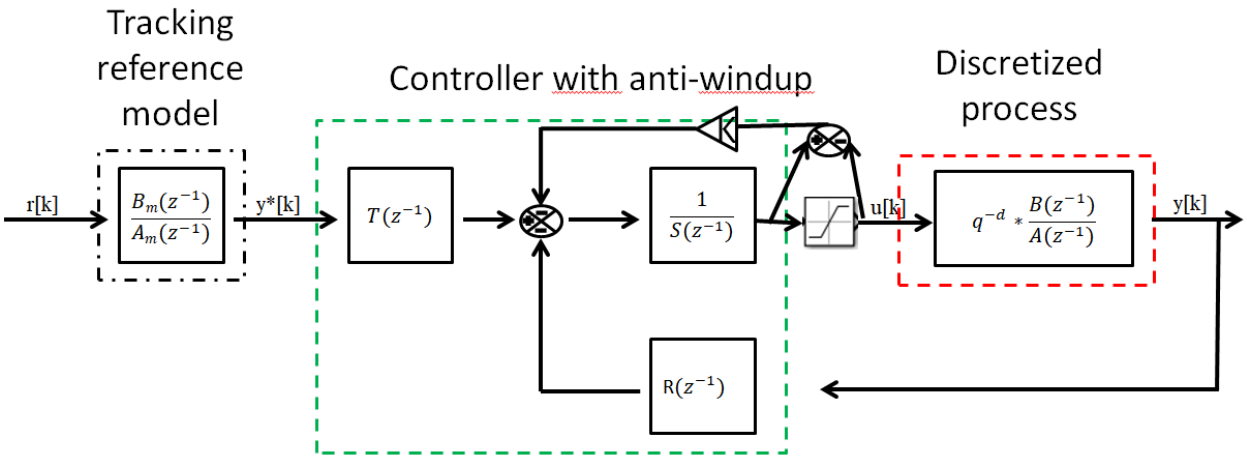


Figure 4-17 RST feedback diagram with anti-windup

The new controller is tested within the same conditions. The results presented in Figure 4-18 confirm us that the new control law is improved. The command is not saturated anymore in the first testing scenario, aspect which leads to a fast rejection of the disturbance.

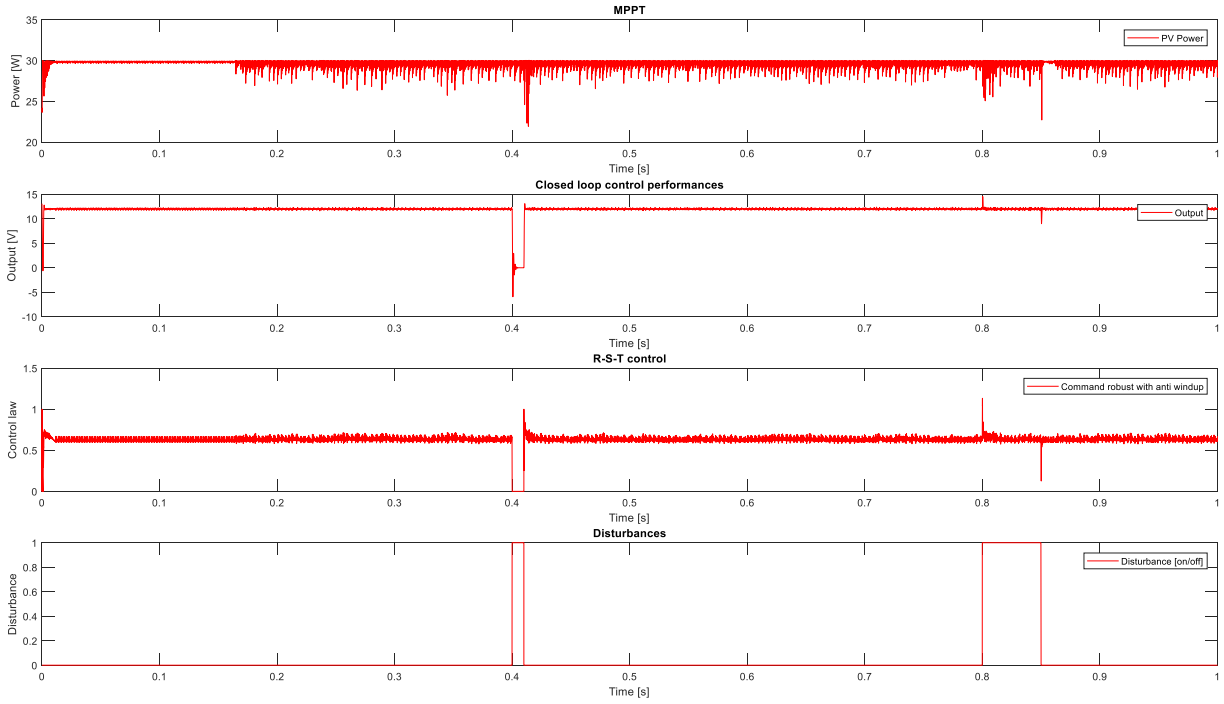


Figure 4-18 Closed loop control with the robust R-S-T control polynomial with anti-windup

4.2.4 Implementation of the robust R-S-T control algorithm

In Figure 4-19 is described the implementation of the DC/DC step down converter. The Arduino Uno measures the voltage of the panel delivered via a voltage divider circuit, and the current of the panel provided by a Hall effect current sensor. It controls the DC/DC converter via a 32Kh frequency output pin which is connected to a Sziklai pair or "complementary Darlington".

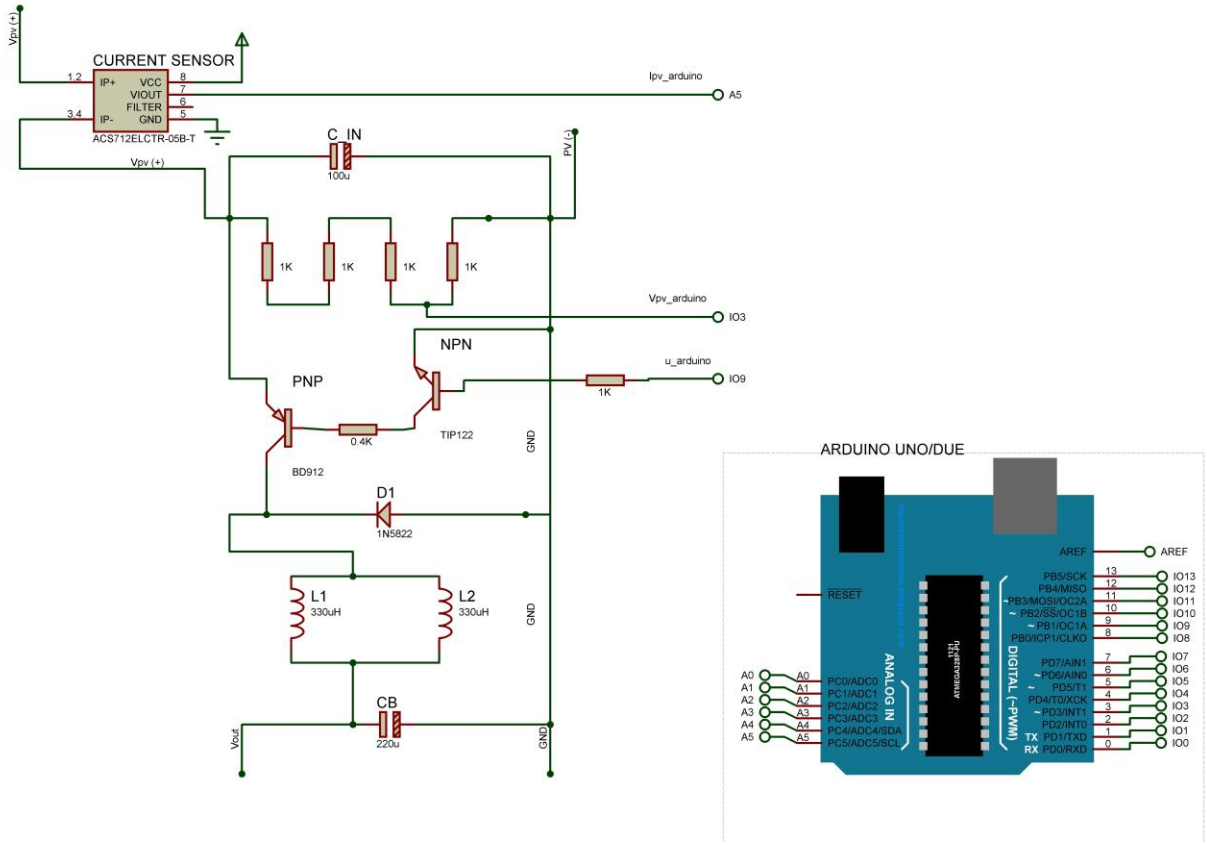


Figure 4-19 Hardware implementation DC/DC

For the achieving the imposed performances a loop that runs at a frequency of 16kHz was used. First the code was run on an Arduino Uno architecture. The frequency of the control loop was limited to maximum 3kHz. The performances of the regulator were not satisfying if the output was measured with an oscilloscope. Therefore a change of architecture was required, which lead to the use of Arduino Due.

The obtained results are presented in Figure 4-20. The reference of 12V is well tracked by the controller. There are some spikes in measurement data, because of the really high resolution that was used to measure with the oscilloscope.

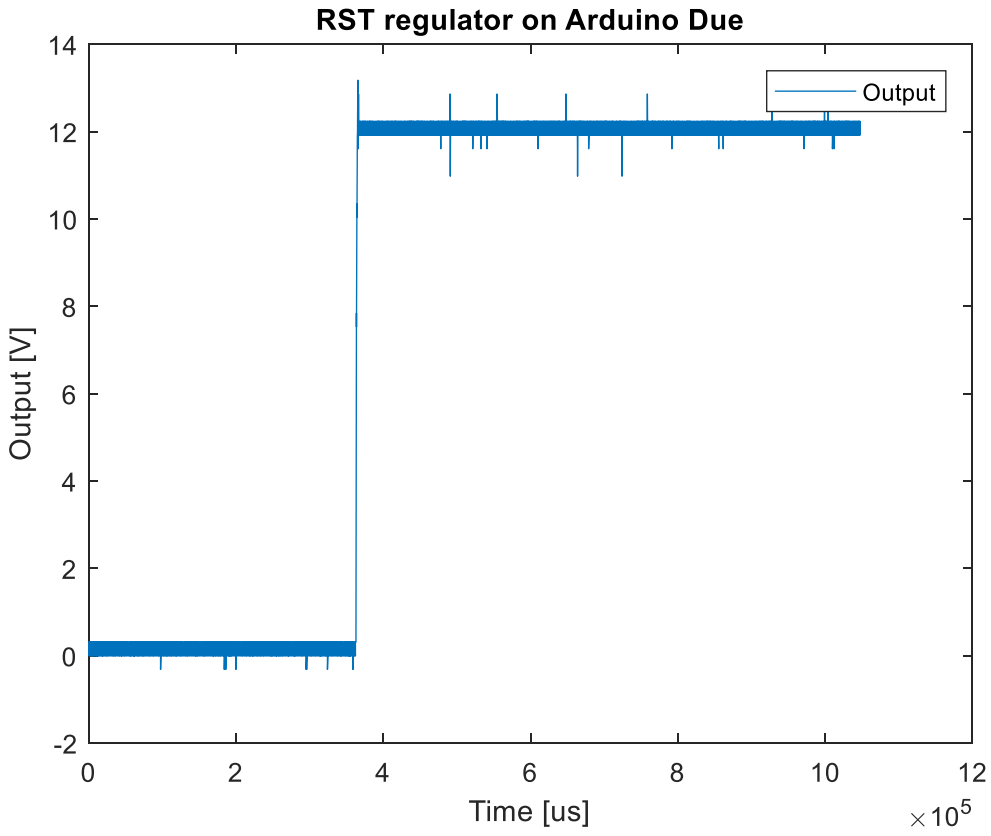


Figure 4-20 Implementation of RST controller on Arduino Due

4.3 Advanced control algorithms – Fuel cell case study

4.3.1 Electrolyzer Unit

In order to obtain the hydrogen by means of electricity, the electrolyzer is the most spread solution. There are two main types of electrolyzers: PEM electrolyzers, that act in a way like a reversed Fuel Cell and alkaline electrolyzers, being simpler and a more accessible solution. The energy efficiency of an electrolyzer can reach an efficiency of 80% and a 99.5% purity of obtained hydrogen. Here we have considered an alkaline electrolyzer, A good reference, on which this article relies, is [11]. The electrolyzer is a 1.5 KW model that activates after a certain minimal current is reached. Temperature variation is taken into account, as in Figure 4-21.

$$U_{elec,cell} = U_{rev} + \frac{r_{1+}r_2 T}{A} I + k_{elec} \ln \frac{k_{T1} + \frac{k_{T2}}{T} + \frac{k_{T3}}{T^2}}{A} I + 1) \quad (91)$$

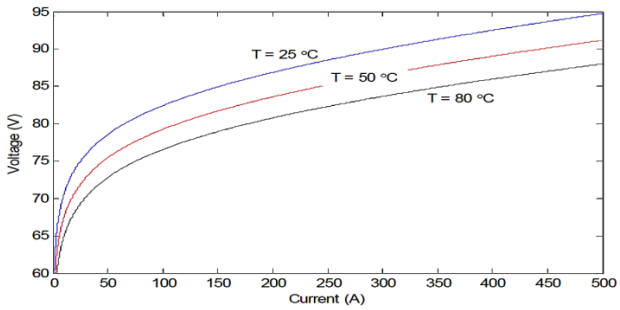


Figure 4-21 U-I characteristics of the electrolyzer at different temperatures

The coefficients are computed as follows: r_1 and r_2 are experimentally obtained constants; k_{T1}, k_{T2}, k_{T3} are temperature related constants. U_{rev} is the reversal voltage.

4.3.2 Fuel Cell Systems

FCs are of different types, but two major categories stand out: FCs based on a Proton Exchange Membrane (P.E.M.), and solid oxide FCs. The first class distinguishes itself through low working temperatures (around 100 degrees Celsius), whereas solid oxide may reach 1000 degrees [12]. A PEM FC is chosen in this work.

From a mathematical modelling point of view, the FC model has two sides: a gaseous part and an electro-chemical part. A detailed modelling of the gaseous part is done in [14]. This is expressed through the equations:

$$\begin{aligned}
\frac{dp_{sm}}{dt} &= \frac{R_{O_2} \cdot T_{st}}{V_{sm}} \cdot \left(W_{cp} - W_{sm,out}\right) \\
\frac{dp_{rm}}{dt} &= \frac{R_a \cdot T_{rm}}{V_{rm}} \cdot \left(W_{ca,out} - W_{rm,out}\right) \\
\frac{dp_{O_2,ca}}{dt} &= \frac{R_{O_2} \cdot T_{st}}{V_{ca}} \cdot \left(W_{O_2,ca,in} - W_{O_2,ca,out} - W_{O_2,reacted}\right) \\
\frac{dp_{v,ca}}{dt} &= \frac{R_v \cdot T_{st}}{V_{ca}} \cdot \left(-W_{ca,out} + W_{v,ca,gen}\right) \\
\frac{dp_{H_2,an}}{dt} &= \frac{R_{H_2} \cdot T_{st}}{V_{an}} \cdot \left(W_{H_2,an,in} - W_{H_2,an,out} - W_{H_2,reacted}\right)
\end{aligned} \tag{92}$$

The electrochemical side, on the other hand is determined by the Nernst equations:

$$\begin{aligned}
V_{dc_stack} &= V_{open} - V_{ohmic} - V_{act} - V_{con} \\
V_{open} &= N_O \left[V_O + \frac{RT}{2F} ln\left(\frac{PH_2\sqrt{PO_2}}{PH_2O\sqrt{PO}}\right)\right] \\
V_{ohmic} &= I_{dc} R_{FC} \\
V_{act} &= N_O \frac{RT}{2\alpha F} ln\left(\frac{I_{dc}}{I_O}\right)
\end{aligned} \tag{93}$$

Where the variables are presented in the following table:

Table 4-1

		Variable	Description
R_v	Vapor gas constant	R_{H_2}	Hydrogen gas constant
R_{O_2}	Oxygen gas constant	$C_{d,an}$	Hydrogen purge nozzle discharge coefficient
M_v	Molar mass of vapor	$W_{sm,out}$	Mass flow exiting the supply manifold[kg/s]
k_{term}	Constants representing the linearization coefficient describing a valve[kg/(s*Pa)]	$W_{ca,out}$	Mass flow exiting the cathode[kg/s]
$W_{rm,out}$	Mass flow exiting the return manifold[kg/s]	$W_{O_2,ca,in}$	Mass flow of oxygen entering the cathode[kg/s]
$W_{H_2,reacted}$	Mass flow of oxygen reacted inside the anode[kg/s]	$\overline{R}$	Universal gas constant
$W_{O_2,reacted}$	Mass flow of oxygen in the cathode that reacts with the electrons and ions to form vapor	$W_{v,ca,gen}$	Mass flow of generated water inside the cathode

4.3.3 Control Algorithm

The MPPT can be achieved in the presence of climate variations and/or variable loads if proper control algorithms are used. Two popular algorithms are the "Perturb and Observe" (P&O) algorithm and the "Incremental Conductance" (IC) algorithm.

The P&O algorithm described in [5] has the advantage of simplicity with satisfying results in tracking. The IC algorithm computes the control law with respect to eq. 14. If $\frac{dP}{dV} > 0$, the duty ratio increases, else if $\frac{dP}{dV} < 0$, the duty ratio decreases, otherwise the duty maintains its previous value.

However, in order to increase the performances, an IC algorithm with a variable step time is proposed. The constant step size is multiplied by $\frac{dP}{dV}$, within certain desired boundaries.

In the following scenario a PV array of 1500 watts is connected to a variable load, via a buck converter. The irradiation value changes at the half time of the simulation, from 1000 W/m^2 to 700 W/m^2, at the same time as the consumer. The performances of a P&O algorithm are compared to those of an IC with variable step size. It can be noticed that the latter converges faster towards the maximum power point and, unlike the P&O algorithm, rejects the disturbances generated by the dynamics of the load.

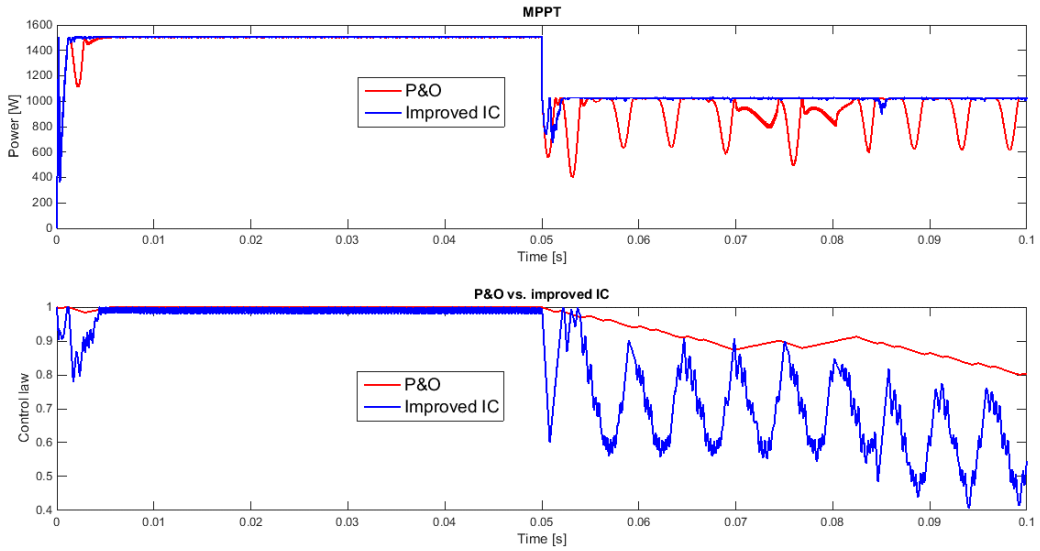


Figure 4-22 Comparison between P&O and IC with variable step size

4.3.4 Supervisory System

In this section, the supervisory system is developed and a certain test scenario is proposed. Let us presume a system composed of a PV array, a buck converter, two banks of batteries, an electrolyzer, a fuel cell, and a set of variable loads. The professional simulation software used are AMESim and Matlab in co-simulation. It has been shown [1]-[5] that under certain meteorological conditions, such as low irradiation or high temperature values, the PV array may not produce enough energy with respect to the consumers. The solution proposed in this article is a supervisory system that manages how the system should respond to external disturbances like variable meteorological conditions and variable loads.

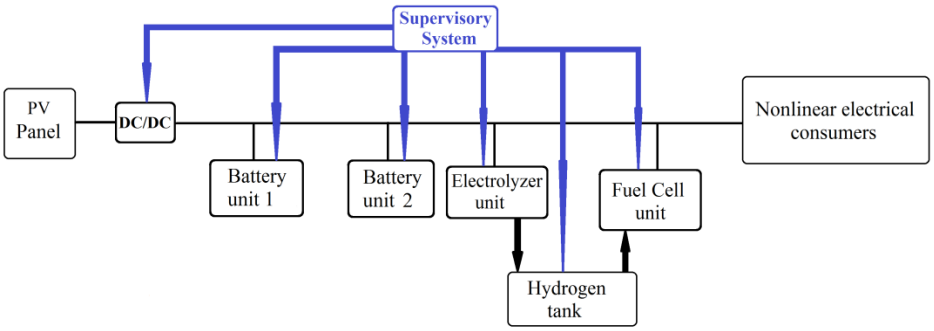


Figure 4-23 System configuration

The PV array provides energy to the consumers via the buck converter. The surplus of energy is stored first in the banks of batteries and, if the batteries are full or other conditions arise (conditions that increase the overall system performance), then through the electrolyzer, the hydrogen tank is filled.

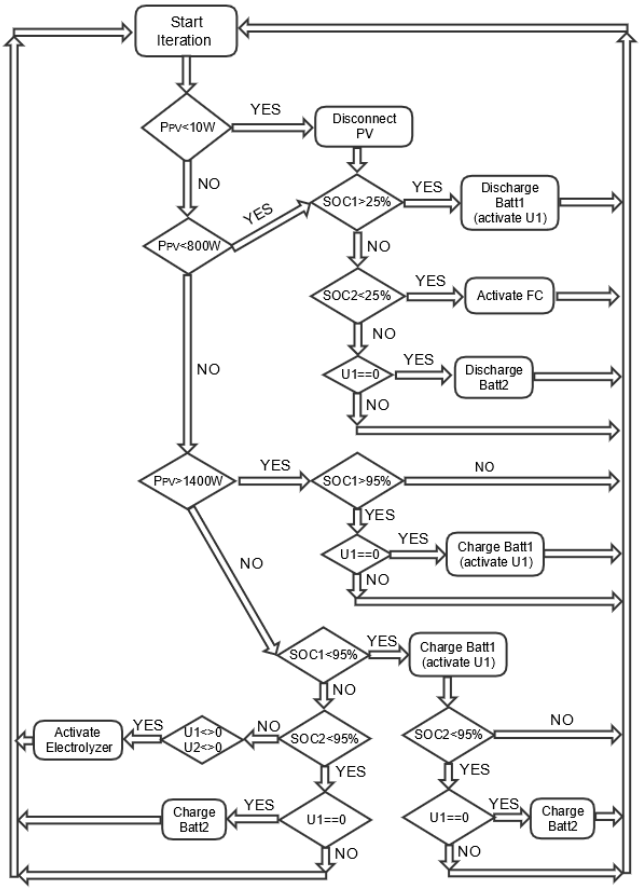


Figure 4-24 System configuration

When the PV array's power becomes insufficient, the battery banks are added to the system to sustain the difference of demanded energy. In order to increase the lifespan of the batteries and to facilitate their replacement in case of failure, two sets of battery banks will be used alternatively. The first one is used, until it is fully charged or until it reaches 25% of its state of charge. Its place is taken by the second set of battery banks, which is used under similar conditions. Furthermore, a third source of energy is used whenever the state of charge of the battery banks becomes lower than 25%. The fuel cell will use the hydrogen tank to produce energy and supply it to the system. The simulations done tested the propose scenarios on a period of 10s.

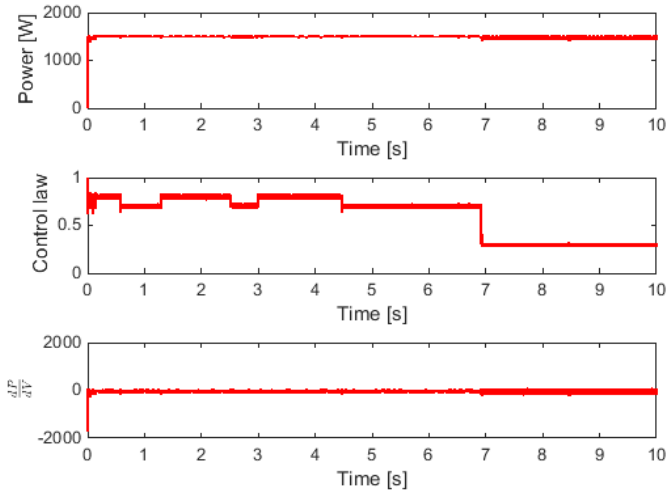


Figure 4-25 Power, Control Law, dPdV

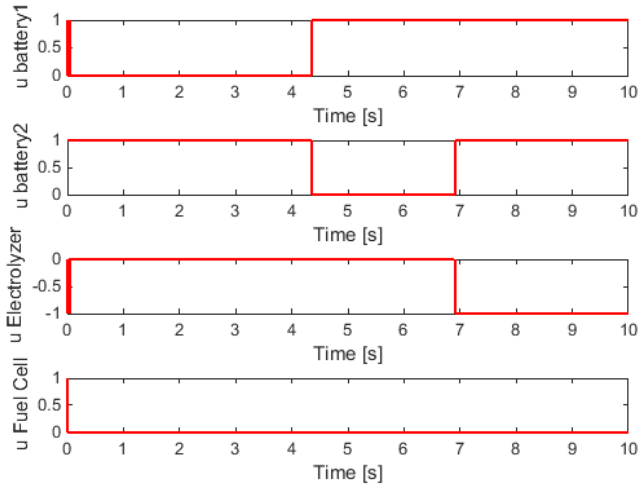


Figure 4-26 Commands

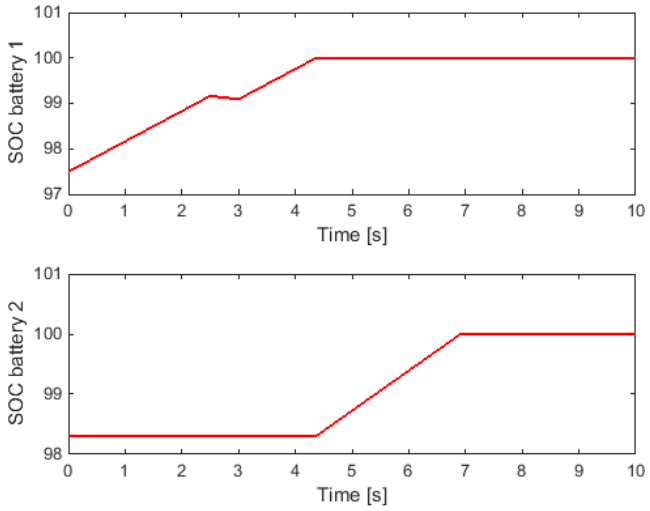


Figure 4-27 SOC bat1, SOC bat2

Scenario 1: Different state parameter evolutions.

In Figure 4-25, Figure 4-26 and Figure 4-27, a first scenario is presented, where a constant maximum power is maintained by means of the DC-DC converter control law described. In the battery evolution figure, one can notice the alternating functioning of the two batteries, this increasing their life-span. At second 7, when the two batteries are fully charged, the electrolyzer starts to generate hydrogen. Another important case is visible between seconds 2.5s and 3s, when an over-demand of the consumer loads discharges the active battery 1. These consumers are nonlinear in behavior.

In the following three figures, a scenario where there is no photovoltaic power (under 10W).

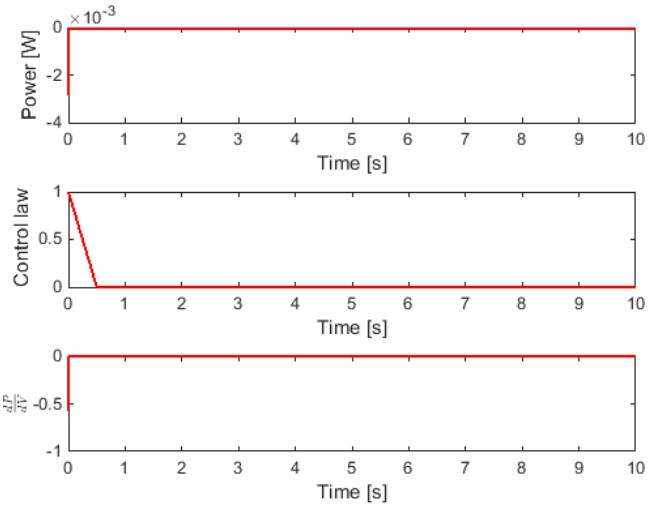


Figure 4-28 Power, Control Law, dPdV

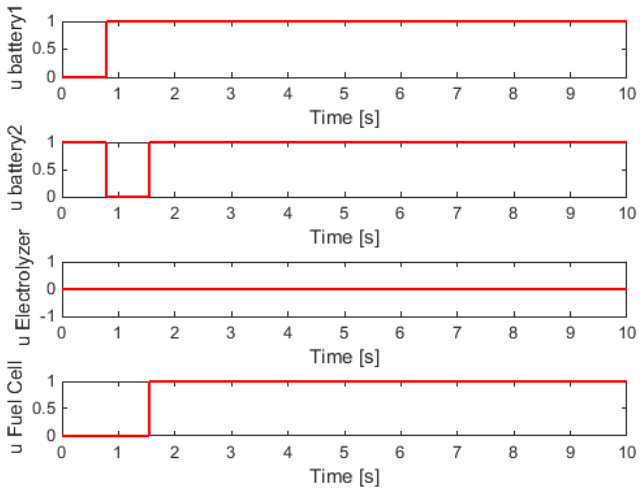


Figure 4-29 Commands

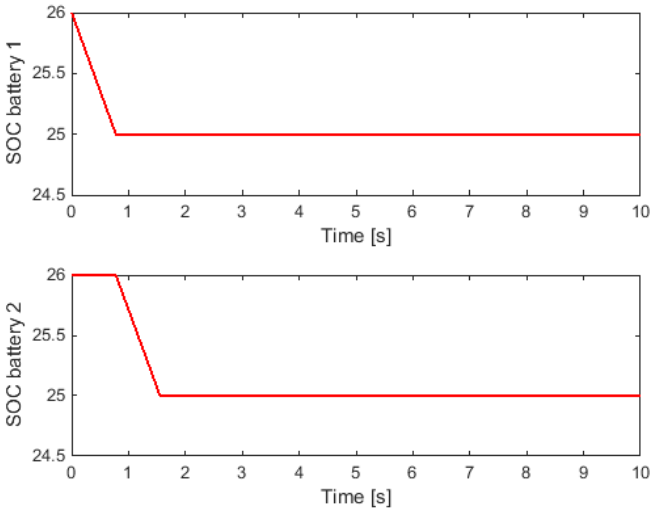


Figure 4-30 SOC bat1, SOC bat2

Second Scenario-System parameters evolutions.

Figure 4-28, Figure 4-29 and Figure 4-30 emphasize the tendency of the supervisor to prioritize the use of only one battery, unless it is absolutely necessary; as such, battery one starts to discharge, and battery 2 activates only when battery 1 reaches 25% state of charge at instant 0.8s. Also, at 1.5s, as the two batteries become depleted, the Fuel Cell is activated to compensate for the power shortage. Although not visible in the present simulations, if the hydrogen tank is filled after a certain value, the batteries will start to charge as well.

5. ARCHITECTURE FOR EMBEDDED SUPERVISORY SYSTEM OF DISTRIBUTED RENEWABLE ENERGY SOURCES

The production of energy through renewable energy sources has become accessible even for urban areas, for example in neighborhoods with near-zero energy buildings. Usually for such cases, each generator is optimized using a Maximum Power Point (MPPT) controller. For this, we propose a hardware architecture that employs microcontroller nodes for local control and microprocessor boards as Ethernet server/client. The first will deal with the acquisition of the input signals required for the MPPT algorithm, whereas the latter deals with the generation of the control signal that is being sent to the buck/boost converter. Some papers deal with similar idea, as [7].

Whenever the energy demand modifies, with respect to the consumer, we can adjust our energy production strategy. This can be easily done with the help of a supervisor. A hardware architecture based on Raspberry PI 3 is proposed for this case study.

In article [1], a home automation solution is described, using a Raspberry Pi that employs Wi-Fi to communicate with the base system. The microprocessor solution together with adjacent sensors manages the essential data about energy consumption, as voltage, current, power and displays statistics about these data. The data is stored in a database that will be further used for statistics that will be shown on an LCD attached. Here, the software solution was implemented using Qt libraries for the display. On the other hand, our work aims at a more industrial level solution.

Another example is the project "openenergymonitor.org" [15] that developed their own hardware as well as integrating other HMI, one of their devices being based also on Raspberry Pi (EmonPi). They targeted users that are also small home users. Other large non-portable solutions are also present. Also a web platform is developed, all this targeting the development of an IoT solution.

Processor based boards have become more powerful and accessible, especially those using ARM based solutions. Their communication capabilities are high yet their response times are limited, thus giving them the potential to act as system supervisors.

In the field of Ethernet based industrial communication, OPC protocol has seen a rise in interest [3], [4]. We have aimed towards Free Opc UA (a version of OPC UA implemented in Python) that uses TCP/IP for communication, thus its destination being industrial communication in SCADA frameworks.

The major advantages of the developed system, as a whole are its simplicity, modularity and the low cost of the solution.

5.1 General Architecture

5.1.1 Architectures Description

A robust, portable architecture was desired, that targeted the high potential industrial IoT area. A Raspberry Pi board is used to implement the high level communication protocols. The choice of raspberry Pi is just as an example, all the software being platform independent, thus any ARM or

Intel compatible device being acceptable. In order to achieve this quality of multiplatform compatibility, Python has been chosen as programming language.

One important aspect of the current architecture is its scalability. As such, each Raspberry Pi, which is an OPC client/server is capable to be connected via serial communication to a few microcontrollers (controlling a generator). All the geographically dispersed nodes in the OPC network follow a connection of the form: one server/ many clients, as [5], [6]. This can be seen as a smart grid.

In this work, a single general control supervisor is chosen, acting as an OPC client, implemented in Matlab Simulink. The importance of a supervisor is to optimize each controller in accordance with demand, production, price and future prognosis.

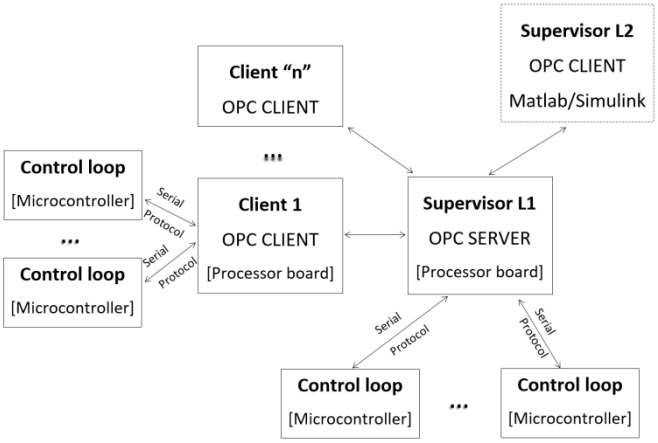


Figure 5-1 Global view of the solution

Python is not a time efficient programming language, yet it compensates through its portability especially when visual interfaces are concerned, as plots and displays of values. It supports an automatic memory management unit and has a diverse set of libraries for different functionalities. It is an interpreter based language thus giving it a multiplatform support. It offers ”dynamic typing”, “reference counting” and “cycle detecting garbage collector”. Also important is its “late binding” that combines names of methods and variables at execution time.

It has multiple implementations, the most common being CPython. Other popular ones are: Jpython, that permits Java classes, IronPython for .NET and PyPy that is a faster and more efficient version. To improve response time, code written in other languages can be incorporated for time dependent parts of the code.

5.1.2 Communication Protocols

OPC UA (Unified Architecture), is a communication protocol of the type M2M (Machine to Machine) that has seen an increase in usage since it appeared in 2008, being developed by the OPC foundation. OPC UA, being the successor of the 1996 OPC (OLE for Process Control), is very different than the original. The main concept being to get rid of COM/DCOM object, of the Windows operating system, to be replaced by the SOA (Service Oriented Architecture) for platform inter-operability, while improving security. It was developed to improve upon the capabilities of the classic OPC, so even thou they are similar, this protocol brings some

advantages, as for example: better finding the availability of OPC servers on the network, monitoring and reporting of client data values when some criteria is reached. OPCUA is an architecture based on services, that integrates OPC functionalities in an easily expandable framework. All the COM OPC specifications are mapped in a “Unified Architecture”.

This is possible by the use of two protocols TCP/IP (opc.tcp://Server) or http (http://Server) for web services.

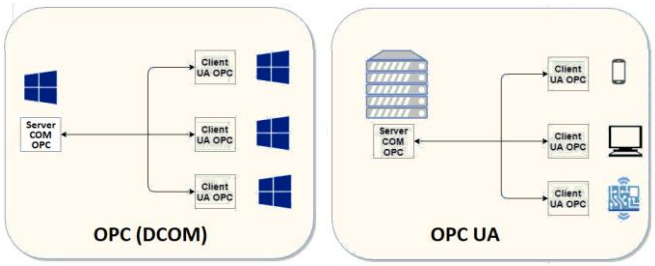


Figure 5-2 Server/client „OpcUa”

One important aspect of industrial communication is the security of data that is transmitted. The OPC UA offers data encryption, authentication schemes and audit. Messages can be encrypted on a 128/256 bit level, the encryption being on a session level. For authentication, each UA client is identified through OpenSSL certificates.

Events are also present in this protocol, for client notification for example. Regarding the addressing space, all data is represented hierarchically, thus creating information that is simple to discover and to manage. Classic OPC can be integrated through the aid of COM/proxy wrappers.

OPC UA being object oriented, has a simple to manage tree of data, even for complex information schematics.

Finally, as the OPC UA is constituted by multiple levels, all innovations regarding transport protocols, encryption, security services, can be incorporated while it maintains compatibility with other versions.

The binary protocol is the best performance wise, using the least amount of resources. Because of its open structure, OPC UA was used with different API’s developed in different programming languages as: C, C++, Java, for commercial SDK and some Open source SDK as well in C, C++, Java and Python.

One such solution is ASNeG, developed in C++. That offers an Open Source OPC UA server application and a web server as well. In .Net implementations, ANSI C is used for low level operations, as socket manipulation. Java implementations are similar to those .Net. The project Eclipse Milo offers an Open Source implementation, pure Java of the client UA 1.03 and corresponding server specifications.

Python implementations, offers a high level of abstraction from the OPC UA client and server. FreeOpcUa is another Open Source solution developed for C++ and Python and it is the solution adopted here. FreeOpcUa is developed using PEP8 standard, easy to use being still in testing and expansion. A server or client can be thus written in just a few lines of code. It is a mixture of high level objects and low level calls in one single application. The majority of low level code is auto generated from xml specifications. A problem that the server resents is that when the

addressing space for XML is being highly used, time consumption increases and thus the starting time of the server increases. The solution is to create a link to the cache file of the server constructor and the portion that contains the addressing space will be created at the first run of the server.

5.2. General scheme

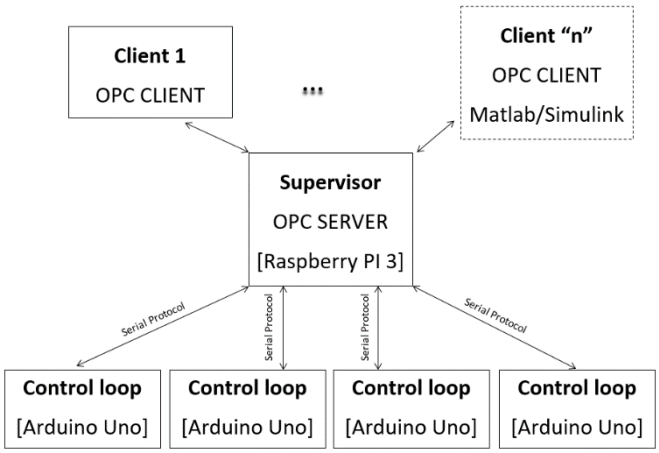


Figure 5-3 Practical test platform

The physical implementation of one node was done, with respect to the scheme presented in Figure 5-3.

The OPC protocol is used for long distance communication, whereas serial communication is used locally. Matlab is used as an OPC client in order to develop supervisory algorithms.

5.2.1. OPC Server

The core of the presented node is represented by the OPC Server. This is implemented on the Raspberry PI 3, that is running on a free operating system, such as Raspbian. The code of the server was developed using "Python", which is an object oriented programming language.

It also plays another important role, the one of the supervisor level 1. It provides the setpoint of energy production for each control loop, with respect to the imposed performances. The performances can be chosen from an offline database or can be received from another supervisor – level 2.

Data, such as the values of temperature, solar irradiation or wind speed, that are crucial for the regulation of solar panels, might be sent to the control loop.

Any of the recorded data can be stored for later use. Therefore, a lightweight disk-based database, such as SQLite can be installed on the OPC server. The main advantage is that it does not require the installation of a server in order to function. Queries can be made locally and plots can be easily drawn.

5.2.2. OPC Client

Client Python on any OS

The client can be represented by a script written in Python programming language, that is running on any compatible operating system and hardware. Multiple users can communicate with the OPC server and have the access to read and write to certain predefined variables of each process. Graphic user interfaces can be added for plotting data and facilitating the visualization of the monitored or controlled variables in real time.

Arduino client

Arduino can also play the role of an OPC client, in the presented work, or even of a server, if a dedicated network module is present. The integrated OBD library can facilitate an easy coding of the program.

Client/ Supervisor Level 2 – Matlab/Simulink

The energy production strategy is often correlated with the power consumption and the prices. A supervisor that centralizes the data from all the local supervisors can be used to optimize the control strategy with respect to the imposed performances and costs. It can connect to the OPC servers from each node, via OPC protocol.

Furthermore, models, control algorithms and estimators, can be tested and improved via Matlab/Simulink that can connect to the data provided by the control loops to the local supervisor.

5.2.3. Control Loop

One of the main goals of this control loop is to extract the maximum amount of power [15,16] from each solar panel, with respect to the power set point, provided by the supervisor, and the weather conditions.

Another goal might be to implement the recharging control loop, if energy storage systems are used.

Both goals can be easily achieved with a rather cheap dedicated solution, such as Arduino Uno.

The communication is done with the Supervisor Level 1 via serial protocol. The default configuration of the Raspberry Pi 3, supports up to four USB connections, that can be also used for communicating via serial protocol with the Arduino architecture. The control loops can provide information to the supervisor, concerning values of different parameters, such as tension and current.

5.2.4. Real time data acquisition and plotting

The connexion between a client and the OPC server is established and plot of real time data is generated.

Figure 5-4 presents plotted data from the local OPC node which was generated using "matplotlib" and "numpy" python libraries. Thus one can connect to a node and visualize the evolution of the recorded parameters.

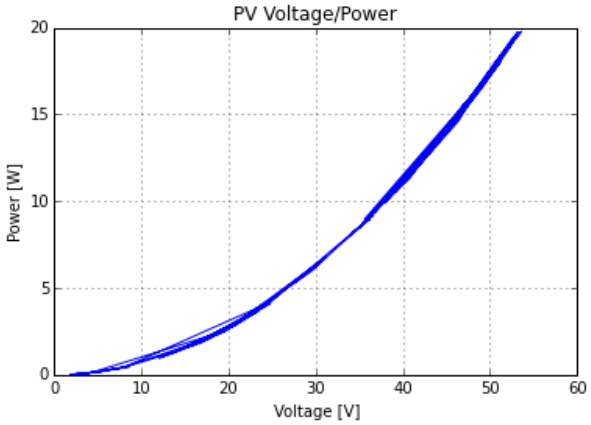


Figure 5-4 Real time data plotting

5.2.5. Implementation

A configuration similar to the one presented in Figure 5-5 has been chosen.

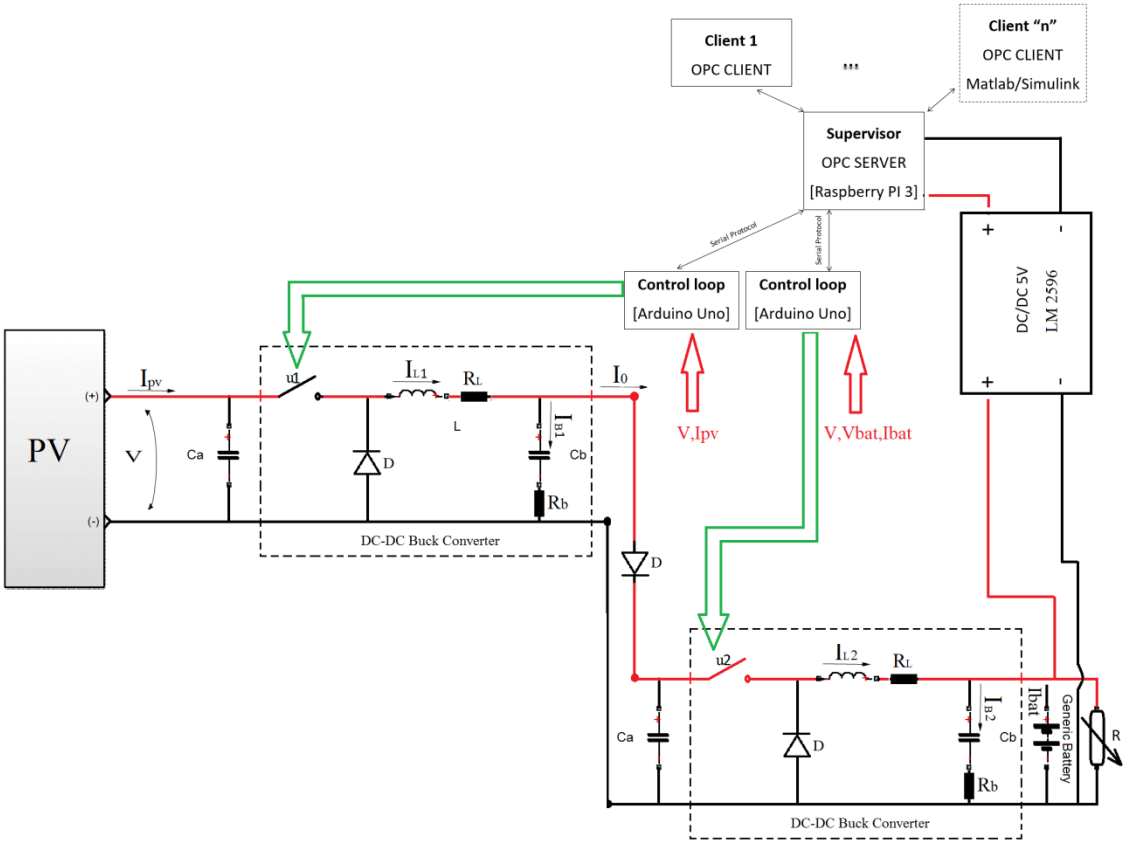


Figure 5-5 Implementation schematic

The access to the server can be done remotely, via internet connection or intranet, through a freeware application, such as VNC. This implies that such a connection is available via built-in WIFI modem, built-in LAN or an additional GSM modem. A routine has been implemented to lunch, after each reboot, the script of the supervisor.

The communication with the two control loops is established and the database starts recording the received data.

The first control loop deals with the performance of the MPPT implemented algorithm. It can switch between different algorithms if a request of such type is received from the supervisor. Options such as receiving and replacing the values of some parameters which are tunable, or overwriting the value of the control law with another imposed value, were implemented. These features can be very useful during testing process and not only. The parameters that first control loops sends to the supervisor are V, the voltage of the solar panel, I_{pv}, its current, and the duty cycle *u1* of the MMPT algorithm which controls the mosfet of the buck converter described in Figure 2-16.

The second control loop is the control system of the battery. The whole architecture is autonomous and is self-powered. This control loop sends to the supervisor the V_{bat}, the voltage of the battery, I_{bat}, its current, and the duty cycle *u2* which is used for controlling the buck converter that controls the charging of the battery.

Data representing the above mentioned parameters of the two control loops, recorded during four days, can be observed in Figure 5-6. The plot was generated using the application SQLite The solar panel reaches its maximum power production during evening, because of constraints of orientation.

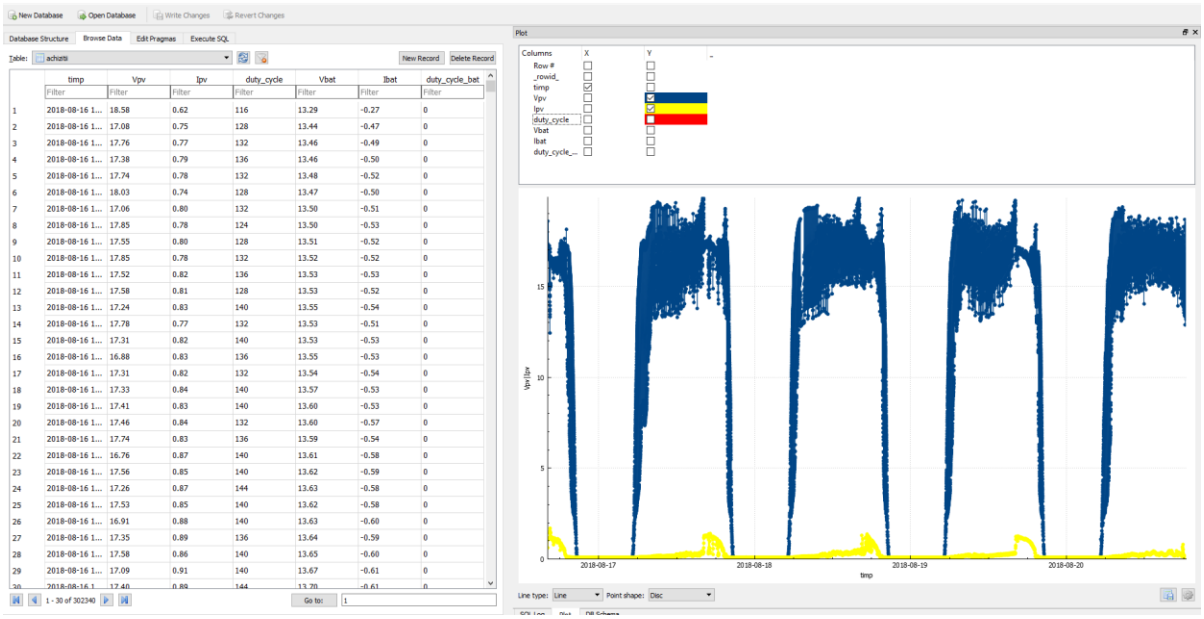


Figure 5-6 Recordings

Data sent by the first control loop, containing the values of the parameters Vpv and Ipv, can be observed in Figure 5-7.

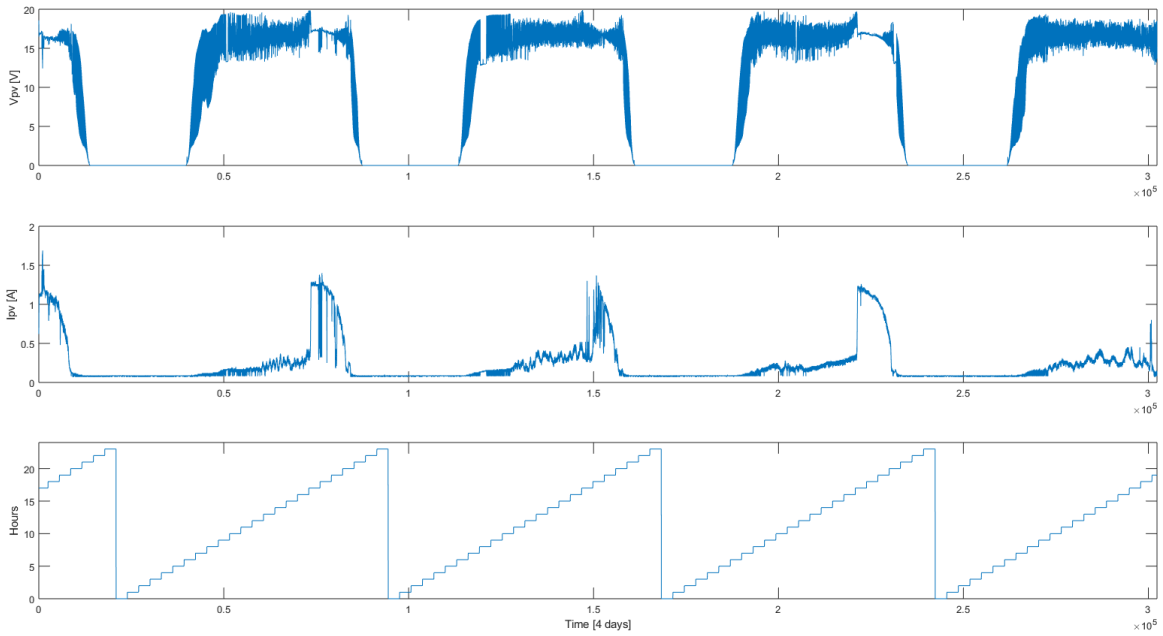


Figure 5-7 PV voltage and current evolution in time

Data sent by the second control loop, containing the values of the parameters Vbat and Ibat, can be observed in Figure 5-8.

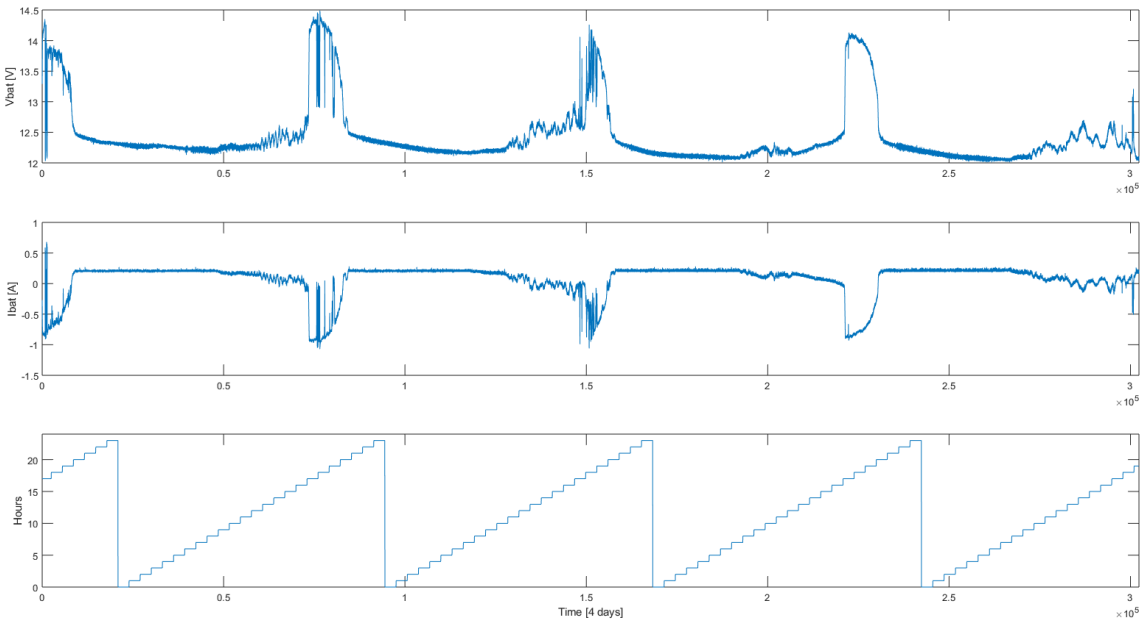


Figure 5-8 Battery voltage and current evolution in time

This platform can communicate with other similar platforms and regulate the total production of energy with respects to the demand and the imposed performances. Relays are used for activating/deactivating static and variable loads, in order to have a more realistic testing scenario.

Some challenges were encountered because of the hardware measurement precision. All the measurements performed with Arduino Uno development board are sensible to voltage variations. In order to solve this issue, at the power on of the system, the offset for the voltage and current sensor is calculated with respect to the current voltage value of the Arduino hardware. Software filtering is also applied for some scenarios.

The stocking space is of 128GB for each Supervisor. Taking into account that for the above mentioned parameters and the current date and time, a recording reaches about 4,5 MB for a sampling time of 1 second, a whole year of recordings will require only about 1,65 GB of storage space.

6. CONCLUSION

The goal of this thesis was to combine both theoretical and practical aspect of control theory applied on systems that function on “green energy”.

It started by presenting the standard models of PV cells and compared their performances in simulation, in order to test them later on in different control loops. The models were built in Matlab and Simulink programming environments, in which a software with a graphic user interface was developed.

In the next part were presented different typologies of DC/DC step down converters, which were later on tested in simulation. Prototypes were developed, tested and validated in real-time closed loop control systems.

Different MPPT control algorithms are presented as a control solution, to improve the efficiency of the energy transfer between a PV panel and the consumer.

A Takagi-Sugeno fuzzy observer is built and tested in a closed loop control system.

A robust polynomial controller with anti-wind up mechanism is presented as a control solution for optimizing the energy transfer between a solar panel and a load or a storage solution. It is tested in both simulation and on a self-made prototype.

A case study presents a supervisory system that manages the transfer of energy between an array of solar panels, some energy storages solutions and the consumers, with the help of a co-simulation between Matlab/Simulink and AmeSim software. Comparisons between the performances obtained with different MMPT algorithms are made.

A prototype, consisting of a a supervisory system, manages the power generation of the solar panel, the recharging system of a lead-acid battery and assures the required power to different variable loads. It works in real time and can be easily controlled remotely via intranet or internet connection.

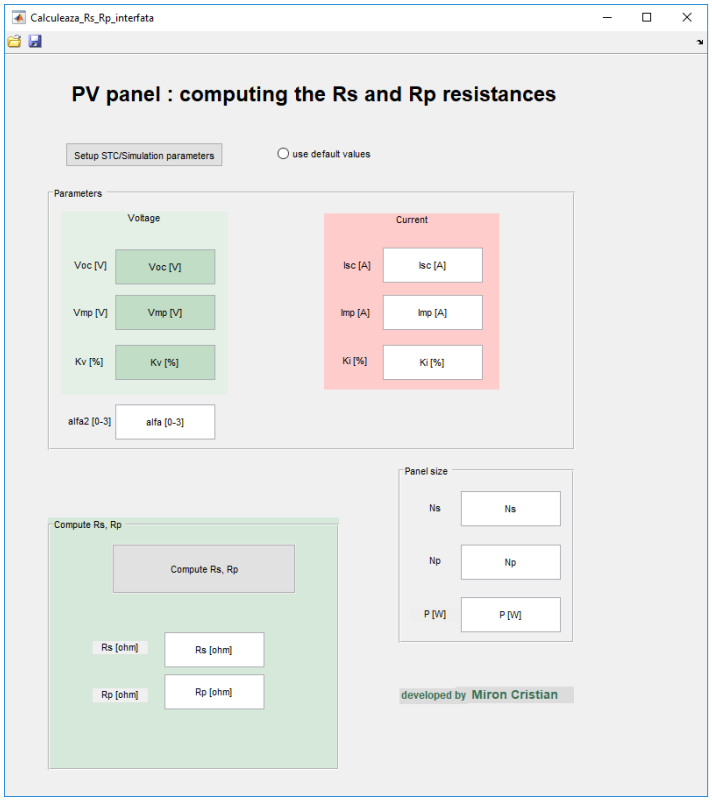
This these presents a solution to the growing need of IoT solutions in the area of renewable energies. The solution is embedded, portable, and scalable, dedicated especially for intelligent energy management in smart neighborhoods.

This work can be further used to develop cheap smart grid network solutions for domestic use or research purposes.

7. ANNEXES

All the written code can be found in the here, along with hardware schematics.

7.1. Graphic User Interface for calibrating the model of a Photovoltaic Panel



7.1.1. Generating the main GUI window

The code for this GUI is presented in the matlab script below:

```
function varargout = Calculeaza_Rs_Rp_interfata(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',   @Calculeaza_Rs_Rp_interfata_OpeningFcn,
...
                   'gui_OutputFcn',     @Calculeaza_Rs_Rp_interfata_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end

function  Calculeaza_Rs_Rp_interfata_OpeningFcn(hObject,  eventdata,  handles,
varargin)

handles.output = hObject;

guidata(hObject, handles);


function varargout = Calculeaza_Rs_Rp_interfata_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;



function edit_Voc_CreateFcn(hObject, eventdata, handles)

if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function uipushtool3_ClickedCallback(hObject, eventdata, handles)
buton=get(handles.radiobutton,'Value');
% assignin(radiobutton,'buton',buton);
if buton==1

evalin('base','load(''valori_initiale'')');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
edit_Voc=num2str(evalin('base', 'Voc'));
set(findobj('Tag','edit_Voc'),'String',num2str(edit_Voc));

edit_Vmp=num2str(evalin('base', 'Vmp'));
set(findobj('Tag','edit_Vmp'),'String',num2str(edit_Vmp));

edit_Kv=num2str(evalin('base', 'Kv'));
set(findobj('Tag','edit_Kv'),'String',num2str(edit_Kv));

edit_alfa=num2str(evalin('base', 'alfa2'));
set(findobj('Tag','edit_alfa'),'String',num2str(edit_alfa));

edit_Isc=num2str(evalin('base', 'Isc'));
set(findobj('Tag','edit_Isc'),'String',num2str(edit_Isc));

edit_Imp=num2str(evalin('base', 'Imp'));
set(findobj('Tag','edit_Imp'),'String',num2str(edit_Imp));

edit_Ki=num2str(evalin('base', 'Ki'));
set(findobj('Tag','edit_Ki'),'String',num2str(edit_Ki));

edit_Ns=num2str(evalin('base', 'Ns'));
set(findobj('Tag','edit_Ns'),'String',num2str(edit_Ns));
```

```
edit_Np=num2str(evalin('base', 'Np'));
set(findobj('Tag','edit_Np'),'String',num2str(edit_Np));

edit_P_max=num2str(evalin('base', 'P_max'));
set(findobj('Tag','edit_P_max'),'String',num2str(edit_P_max));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

else
evalin('base','uiload');

% evalin('base','load(''valori_initiale'')');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
edit_Voc=num2str(evalin('base', 'Voc'));
set(findobj('Tag','edit_Voc'),'String',num2str(edit_Voc));

edit_Vmp=num2str(evalin('base', 'Vmp'));
set(findobj('Tag','edit_Vmp'),'String',num2str(edit_Vmp));

edit_Kv=num2str(evalin('base', 'Kv'));
set(findobj('Tag','edit_Kv'),'String',num2str(edit_Kv));
try


edit_alfa=num2str(evalin('base', 'alfa2'));
set(findobj('Tag','edit_alfa'),'String',num2str(edit_alfa));
catch err
    cod_eroare='Undefined function or variable ''alfa2''.'
    if (strcmp(err.message,cod_eroare))
        alfa2=1.2
        assignin('base','alfa2',alfa2);
        edit_alfa=num2str(evalin('base', 'alfa2'));
        set(findobj('Tag','edit_alfa'),'String',num2str(edit_alfa));
    else
    end
end
edit_Isc=num2str(evalin('base', 'Isc'));
set(findobj('Tag','edit_Isc'),'String',num2str(edit_Isc));

edit_Imp=num2str(evalin('base', 'Imp'));
set(findobj('Tag','edit_Imp'),'String',num2str(edit_Imp));

edit_Ki=num2str(evalin('base', 'Ki'));
set(findobj('Tag','edit_Ki'),'String',num2str(edit_Ki));

edit_Ns=num2str(evalin('base', 'Ns'));
set(findobj('Tag','edit_Ns'),'String',num2str(edit_Ns));

edit_Np=num2str(evalin('base', 'Np'));
set(findobj('Tag','edit_Np'),'String',num2str(edit_Np));

edit_P_max=num2str(evalin('base', 'P_max'));
set(findobj('Tag','edit_P_max'),'String',num2str(edit_P_max));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

function edit_Voc_Callback(hObject, eventdata, handles)
```

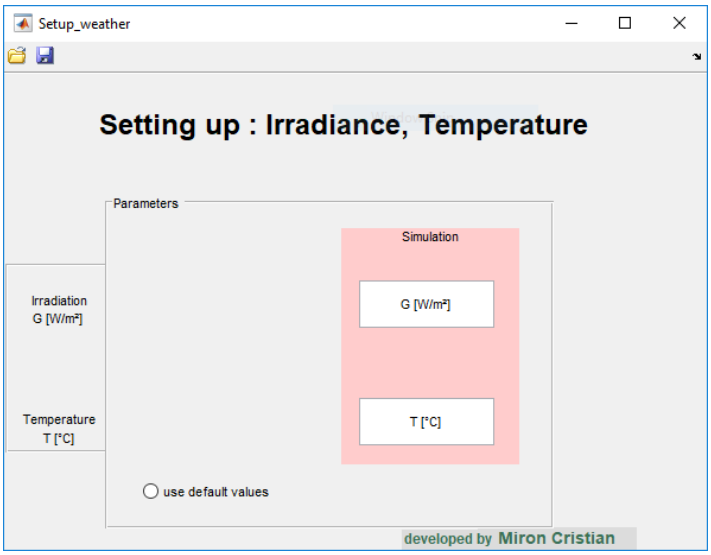
```
edit_Voc=str2double(get(hObject,'String'));
assignin('base','Voc',edit_Voc);

function edit_Vmp_Callback(hObject, eventdata, handles)

edit_Vmp=str2double(get(hObject,'String'));
assignin('base','Vmp',edit_Vmp);


% --- Executes during object creation, after setting all properties.
function edit_Vmp_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_Kv_Callback(hObject, eventdata, handles)

edit_Kv=str2double(get(hObject,'String'))
assignin('base','Kv',edit_Kv);

function edit_Kv_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_alfa_Callback(hObject, eventdata, handles)
edit_alfa=str2double(get(hObject,'String'));
assignin('base','alfa2',edit_alfa);

function edit_alfa_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_Isc_Callback(hObject, eventdata, handles)

edit_Isc=str2double(get(hObject,'String'));
assignin('base','Isc',edit_Isc);



function edit_Isc_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit_Imp_Callback(hObject, eventdata, handles)

edit_Imp=str2double(get(hObject,'String'));
assignin('base','Imp',edit_Imp);

% --- Executes during object creation, after setting all properties.
function edit_Imp_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_Ki_Callback(hObject, eventdata, handles)

edit_Ki=str2double(get(hObject,'String'));
assignin('base','Ki',edit_Ki);

function edit_Ki_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_Ns_Callback(hObject, eventdata, handles)
edit_Ns=str2double(get(hObject,'String'));
assignin('base','Ns',edit_Ns);

function edit_Ns_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_Np_Callback(hObject, eventdata, handles)

edit_Np=str2double(get(hObject,'String'));
assignin('base','Np',edit_Np);

function edit_Np_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_P_max_Callback(hObject, eventdata, handles)
edit_P_max=str2double(get(hObject,'String'));
```

```
assignin('base','P_max',edit_P_max);


function edit_P_max_CreateFcn(hObject, eventdata, handles)

if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --------------------------------------------------------------------
function uipushtool4_ClickedCallback(hObject, eventdata, handles)

[Voc]=evalin('base','Voc');
[Vmp]=evalin('base','Vmp');
[Kv]=evalin('base','Kv');
[alfa]=evalin('base','alfa');
[Isc]=evalin('base','Isc');
[Imp]=evalin('base','Imp');
[Ki]=evalin('base','Ki');
[Ns]=evalin('base','Ns');
[Np]=evalin('base','Np');
[P_max]=evalin('base','P_max');

configuratie={'Voc','Vmp','Kv','alfa','Isc','Imp','Ki','Ns','Np','P_max'};
uisave(configuratie,'configuratie');

% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)

% --- Executes on radiobutton press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

run Setup_weather.m


% --- Executes on radiobutton press in radiobutton.
function radiobutton_Callback(hObject, eventdata, handles)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

run calculeaza_Rs_Rp_script_2d_2alfa_Newton_Rhapson.m
edit_Rs_Callback()
edit_Rp_Callback()
run calculeaza_Rs_Rp_script_grafic_2alfa.m


function edit_Rs_Callback(hObject, eventdata, handles)
edit_Rs=num2str(evalin('base', 'Rs'));
set(findobj('Tag','edit_Rs'),'String',num2str(edit_Rs));



function edit_Rs_CreateFcn(hObject, eventdata, handles)
```

```
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_Rp_Callback(hObject, eventdata, handles)
edit_Rp=num2str(evalin('base', 'Rp'));
set(findobj('Tag','edit_Rp'),'String',num2str(edit_Rp));


function edit_Rp_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

7.1.2. Generating the secondary GUI window



The user can introduce the values of the STC parameters before starting the simulation. The default values are G=1000 W/m2 and T=25°C.

```
function varargout = Setup_weather(varargin)
% SETUP_WEATHER MATLAB code for Setup_weather.fig

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Setup_weather_OpeningFcn, ...
                   'gui_OutputFcn',  @Setup_weather_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Setup_weather is made visible.
function Setup_weather_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Setup_weather (see VARARGIN)

% Choose default command line output for Setup_weather
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Setup_weather wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Setup_weather_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;




% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function uipushtool1_ClickedCallback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function uipushtool3_ClickedCallback(hObject, eventdata, handles)
buton=get(handles.radiobutton1,'Value');
% assignin(button,'buton',buton);
if buton==1

evalin('base','load(''valori_initiale_m'')');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
edit2=num2str(evalin('base', 'Gstc'));
set(findobj('Tag','edit2'),'String',num2str(edit2))

edit3=num2str(evalin('base', 'T'));
set(findobj('Tag','edit3'),'String',num2str(edit3))

edit4=num2str(evalin('base', 'G'));
set(findobj('Tag','edit4'),'String',num2str(edit4))

edit7=num2str(evalin('base', 'Tstc'));
set(findobj('Tag','edit7'),'String',num2str(edit7))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

else
uiimport

evalin('base','load(''valori_initiale_m'')');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
edit2=num2str(evalin('base', 'Gstc'));
set(findobj('Tag','edit2'),'String',num2str(edit2))

edit3=num2str(evalin('base', 'T'));
set(findobj('Tag','edit3'),'String',num2str(edit3))

edit4=num2str(evalin('base', 'G'));
set(findobj('Tag','edit4'),'String',num2str(edit4))

edit7=num2str(evalin('base', 'Tstc'));
set(findobj('Tag','edit7'),'String',num2str(edit7))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

function edit2_Callback(hObject, eventdata, handles)
edit2=str2double(get(hObject,'String'))
assignin('base','Gstc',edit2);


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
```

```
function edit3_Callback(hObject, eventdata, handles)
edit3=str2double(get(hObject,'String'))
assignin('base','T',edit3);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
edit4=str2double(get(hObject,'String'))
assignin('base','G',edit4);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
edit7=str2double(get(hObject,'String'))
assignin('base','Tstc',edit7);


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
function uipushtool4_ClickedCallback(hObject, eventdata, handles)

Gstc=evalin('base','Gstc');
Tstc=evalin('base','Tstc');
G=evalin('base','G');
T=evalin('base','T');

configuratie={'Gstc','Tstc','G','T'}
uisave(configuratie,'configuratie');
```

```
% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
```

7.1.3. Newton – Raphson research algorithm

For computing the values of the resistors Rs and Rp the following Newton – Raphson method is implemented in the script below:

```
function                    [y,P,Rs1,Rp1,gasit1,V1,derivata]                    =
calculeaza_Rs_Rp_scrip(I,V,T,G,Rs,Rp,gasit)
gasit=0;
marja_eroare=0.0007
Voc=evalin('base','Voc');
Vmp=evalin('base','Vmp');

Isc=evalin('base','Isc');
Imp=evalin('base','Imp');

I=0; % valoarea calculata a curentului

Kv=evalin('base','Kv');
Ki=evalin('base','Ki');

Ns=evalin('base','Ns');
Np=evalin('base','Np');
P_max=evalin('base','P_max');

Gstc=evalin('base','Gstc');
Tstc=evalin('base','Tstc');

G=evalin('base','G');
T=evalin('base','T');



% alfa1=evalin('base','alfa1');
Imp=Imp*Np;
Isc=Isc*Np;
P_max=evalin('base','P_max')*Np;
assignin('base','Imp',Imp);
assignin('base','Isc',Isc);
% assignin('base','P_max',P_max);

Rs=0.01;
Rp=Vmp/(Isc-Imp)-(Voc-Vmp)/Imp;

assignin('base','Rs',Rs);
assignin('base','Rp',Rp);

assignin('base','I',I);
assignin('base','gasit',gasit);

delta_T=T-Tstc;
```

```
T=T+273.15;
alfa1=1;
try
alfa2=evalin('base','alfa2');
catch err
    if (strcmp(err.message,'Undefined function or variable ''alfa2''.'))
        alfa2=1.2
    else
    end
end
assignin('base','alfa1',alfa1);
assignin('base','alfa2',alfa2);

P_anterior=100000;
P_curent=90000;



while   (((std([P_curent,P_max])>=marja_eroare)   &&Rs<=1.1   &&   alfa2<=8   &&
alfa1<=3))
 if (Rs>=1)
     alfa2=alfa2+0.1;
     Rs=0;
     assignin('base','Rs',Rs);
 end
derivata=0;

P_anterior=P_curent;

Rs=evalin('base','Rs');
Rp=evalin('base','Rp');

gasit=evalin('base','gasit');


VT=(1.3806503/1.60217646*10^-4)*T;

I01=(Isc+Ki*delta_T)/(exp((Voc+Kv*delta_T)/(Ns*VT))-1);
% I01=(Isc+Ki*delta_T)/(exp((Voc+Kv*delta_T)/(alfa1*Ns*VT))-1);

% I02=(Isc+Ki*delta_T)/(exp((Voc+Kv*delta_T)/(alfa2*Ns*VT))-1);

Ipv=(Isc+Ki*delta_T)*G/Gstc;


k=1;
I=0;


for V=0.1:0.01:Voc

x0=0.001;
fx1=x0-Ipv+I01*(exp((V+x0*Rs)/(alfa1*Ns*VT))+exp((V+x0*Rs)/(alfa2*Ns*VT))-
2)+(V+x0*Rs)/Rp;

   while abs(fx1)>0.00001 && x0>=0    %abs(fx1)>0.001 && x0>=0
```

```
        fx0=x0-
Ipv+I01*(exp((V+x0*Rs)/(alfa1*Ns*VT))+exp((V+x0*Rs)/(alfa2*Ns*VT))-
2)+(V+x0*Rs)/Rp;

fpx0=1+I01*((Rs/(alfa1*Ns*VT))*exp((V+x0*Rs)/(alfa1*Ns*VT))+(Rs/(alfa2*Ns*VT)
)*exp((V+x0*Rs)/(alfa2*Ns*VT)))+Rs/Rp;
        x1=x0-fx0/fpx0;
        fx1=x1-
Ipv+I01*(exp((V+x1*Rs)/(alfa1*Ns*VT))+exp((V+x1*Rs)/(alfa2*Ns*VT))-
2)+(V+x1*Rs)/Rp;  %verific daca noua val a sol fct fx=0 este inf marjei de
eroare
        x0=x1;


   end
I=x0;
P=I*V;
I_salvat(k)=I;
P_salvat(k)=P;
V_salvat(k)=V;
k=k+1;
end

[val,poz]=max(P_salvat);
P=P_salvat(poz); % pentru a gasi exact Imp si Vmp

%  plot(P_salvat);
P_curent=P;


Id1=I01*(exp((Vmp+Imp*Rs)/(alfa1*VT*Ns))-1);
% Id2=I02*(exp((Vmp+Imp*Rs)/(alfa2*VT*Ns))-1);
Id2=I01*(exp((Vmp+Imp*Rs)/(alfa2*VT*Ns))-1);
    Rs=Rs+1e-3;

    Rp=(Vmp+Imp*Rs)/((Ipv-Id1-Id2)-P_max/Vmp);

    if(Rp<0)
        Rs=1;
       P_curent=0;

    end
    if(alfa2>=8)
        alfa1=alfa1+1;
        alfa2=1;
    end

y=I;
date_salvate=[V_salvat' I_salvat' P_salvat'];
assignin('base','Rs',Rs);
assignin('base','Rp',Rp);
assignin('base','alfa1',alfa1);
assignin('base','alfa2',alfa2);
assignin('base','gasit',gasit);
assignin('base','I',y);
assignin('base','P',P);
assignin('base','date_salvate',date_salvate);
```

```
% end
Rp_anterior=Rp;
end

P_eroare=P-P_max
assignin('base','P_eroare',P_eroare);
sprintf('marja_eroare_impusa=%d;',marja_eroare)
sprintf('Rs=%d;',Rs)
if(Rp<0)
sprintf('Rp=%d;',Rp_anterior)
else
    sprintf('Rp=%d;',Rp)
end

sprintf('P_eroare=%d;',P_eroare)
```

7.2. MPPT tracking control algorithm on Arduino Uno

The code was written for Arduino Uno hardware and can be compiled and tested with the open source software “Arduino IDE”. It reads the voltage from the PV panel from a resistor divider and the current from a current sensor ACS712 (5A). It computes the output of the P&O / Incremental conductance algorithm and sends it to the transistor. The communication is made with the Raspbery Pi 3 supervisor via serial port. The messages must respect a certain convention in order to be taken into account. Whenever data is received via serial port, it is analyzed and if the input format is respected, it is passed to the main loop. The supervisor can override the computed value of the control algorithm or change the step size.

```
#include <SPI.h>

#include <SD.h>

String inputString = "";         // a String to hold incoming data

String string1 = "";

String string2 = "";

String string3 = "";

String string4 = "";

boolean stringComplete = false;  // whether the string is complete

float data,data1,data2,data3;

int despartitor[3];

int index,j;


String date_totale;
```

```
int pin_pwm=9; // 9  sau 5 sau 6


int duty_cycle=1;

int n;

const int analogIn = A5;
const int analogVin = A4;
const int  analogVpv=A3;
float mVperAmp = 0.100; // use 100 for 20A Module and 66 for 30A Module 185 pt 5a
float ACS_RawValue,ACS_Voltage,ACS_Amps,Putere_consumata;
float ACSoffset = 2.516;
float tensiune_Arduino=4.979;
float Vbat_RawValue,Vbat;
float Vpv_RawValue,Vpv;

float P;
float P_vechi;
float deltaP;
float deltaP_vechi;
float deltaVpv;
float Vpv_vechi;
float deltaVpv_vechi;


int amplificare;

float conditie;
float timp_esantionare;
```

```
bool semnal_validare;

unsigned long time1;
unsigned long time2;

File myFile;

void setup(){
 Serial.begin(9600);
 pinMode(pin_pwm, OUTPUT);
 TCCR1B = TCCR1B & B11111000 | B00000001; // setez la 32khz atentie la pin 9
// TCCR0B = (TCCR0B & 0b11111000) | B00000001; //setez la 62 khz pin 5 sau 6
   analogWrite(pin_pwm,0); // inchid alimentarea si masor valoarea senzorului pentru a o folosi ca decalaj

 // reserve 200 bytes for the inputString:
  inputString.reserve(200);
  string1.reserve(200);
  string2.reserve(200);
  string3.reserve(200);
  string4.reserve(200);
  index=0;

  delay(2000);

 n=1000;

  for(int i=0;i<n;i++)
  {
 ACSoffset+=(((analogRead(analogIn) / 1024.0) * tensiune_Arduino ) );
```

```
 delay(2);
  }
 ACSoffset=ACSoffset/n;
ACSoffset=ACSoffset*0.998;

 analogWrite(pin_pwm,100);
 semnal_validare=1;
delay(2);

string4 = "*";

string4.concat('0');
string4.concat(";");
string4.concat("0");
string4.concat(";");
string4.concat('0');
string4.concat(";");

}

void loop(){
  /*
  Serial.print("Time: ");
  time1 = millis();
  Serial.print(time1);
 */

ACS_RawValue=0;
ACS_Voltage=0;
ACS_Amps=0;
Vbat_RawValue=0;
```

```
Vbat=0;
Vpv=0;

// print the string when a newline arrives:
 if (stringComplete) {
 //   Serial.println(inputString);
   data= inputString.toFloat();

   data1=string1.toFloat();
   data2= string2.toFloat();
   data3= string3.toFloat();

   inputString = "";
   stringComplete = false;
   semnal_validare=not(semnal_validare);
 }

 n=200;
 for(int i=0;i<n;i++)
 {

ACS_RawValue+= analogRead(analogIn);
ACS_Voltage+= (analogRead(analogIn) / 1024.0) * tensiune_Arduino; // Gets you mV
ACS_Amps+= (((analogRead(analogIn) / 1024.0) * tensiune_Arduino - ACSoffset) / mVperAmp);

//citesc tensiunea bateriei
Vbat_RawValue+= analogRead(analogVin);
Vbat+=(analogRead(analogVin)/1024.0)*tensiune_Arduino;

//citesc tensiunea panoului
Vpv_RawValue+= analogRead(analogVpv);
```

```
Vpv+=(analogRead(analogVpv)/1024.0)*tensiune_Arduino;

//delay(1);
 }

ACS_RawValue=ACS_RawValue/n;
ACS_Voltage=ACS_Voltage/n;
ACS_Amps=ACS_Amps/n;
Vbat_RawValue=Vbat_RawValue/n;
Vbat=Vbat/n*4;
Vpv_RawValue=Vbat_RawValue/n;
Vpv=Vpv/n*4;
Putere_consumata=ACS_Amps*Vpv;


 // trebuie implementat algoritmul mmmpt
   P=Vpv*ACS_Amps;
   deltaP=P-P_vechi;
   deltaVpv=Vpv-Vpv_vechi;
 //  deltaI=ACS_Amps-I_vechi;

   timp_esantionare=0.2;

   conditie=deltaP/(timp_esantionare); //corectie facuta

   //conditie=ACS_Amps+(Vpv*(deltaI/deltaVpv));

   amplificare=4;

   if (data1==1) // modific pasul de inaintare
   {
```

```
   amplificare=data2;
   }

   //1 dP>0 dV>0
   if ((conditie>0.2) & (deltaVpv>0))
   {
    duty_cycle=duty_cycle-amplificare;
    }

   //2 dP<0 dV>0
   if ((conditie<-0.2) & (deltaVpv>0))
    {
      duty_cycle=duty_cycle+amplificare;
    }

    //3 dP>0 dV<0
   if ((conditie>0.2) & (deltaVpv<0))
   {
    duty_cycle=duty_cycle+amplificare;
    }

    //4 dP<0 dV<0
   if ((conditie<-0.2) & (deltaVpv<0))
    {
      duty_cycle=duty_cycle-amplificare;
    }

if (duty_cycle<0)
{
duty_cycle=0;
}
```

```
if (duty_cycle>255)
{
duty_cycle=255;
}

if (data1==2)
 {
  duty_cycle=data3;
 }

analogWrite(pin_pwm,duty_cycle);

   P_vechi=P;
   deltaP_vechi=deltaP;
   Vpv_vechi=Vpv;
   deltaVpv_vechi=deltaVpv;

semnal_validare==1;
if(1==1)
{
string4 = "*";

string4.concat(Vpv);
string4.concat(";");
string4.concat(ACS_Amps);
string4.concat(";");
string4.concat(duty_cycle);
string4.concat(";");

Serial.println(string4);
```

```
}
}


void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    index=index+1;
    // add it to the inputString:
    inputString += inChar;
    if (inChar == ';') {
      j=j+1;

      if (j==1) {

      string1=inputString;
      inputString="";

      }


     if (j==2) {
     string2 = inputString;
     inputString="";
    }

    if (j==3) {
     string3 = inputString;
     inputString="";
    }
```

```
  }


  if (inChar == '\n') {
    stringComplete = true;
    index=0;
    j=0;
  }
 }
}
```

7.3. Battery charging supervisor on Arduino Uno

The code was written for Arduino Uno hardware and can be compiled and tested with the open source software “Arduino IDE”. It reads the voltage from the PV panel from a resistor divider, the voltage of the battery and the current of the battery from a current sensor ACS712 (5A).It controls the DC/DC buck converter that charges the battery. The communication is made with the Raspbery Pi 3 supervisor via serial port. The messages must respect a certain convention in order to be taken into account. Whenever data is received via serial port, it is analyzed and if the input format is respected, it is passed to the main loop. The supervisor can override the computed value of the control algorithm or change the step size.

```
#include <SPI.h>
#include <SD.h>



int pin_pwm=9;


int duty_cycle=1;


const int analogIn = A5;
const int analogVin = A4;
const int  analogVpv=A3;
float mVperAmp = 0.185; // use 100 for 20A Module and 66 for 30A Module
float ACS_RawValue,ACS_Voltage,ACS_Amps,Putere_consumata;
float ACSoffset = 2.516;
```

```
float tensiune_Arduino=4.979;
float Vbat_RawValue,Vbat;
float Vpv_RawValue,Vpv;

boolean stringComplete = false;  // whether the string is complete

String inputString = "";        // a String to hold incoming data
String string1 = "";
String string2 = "";
String string3 = "";
String string4 = "";
float data,data1,data2,data3;
int despartitor[3];
int index,j;

String date_totale;

bool semnal_validare;

File myFile;

void setup(){
 Serial.begin(9600);
 TCCR1B = TCCR1B & B11111000 | B00000001; // setez la 32khz
   analogWrite(pin_pwm,255); // inchid alimentarea si masor valoarea senzorului pentru a o folosi ca decalaj


  delay(2000);
```

```
 for(int i=0;i<150;i++)
 {

ACSoffset+=(((analogRead(analogIn) / 1024.0) * tensiune_Arduino ) );

 delay(2);
 }

 ACSoffset=ACSoffset/150;
 ACSoffset=ACSoffset*0.993;
 ACSoffset=2.5;
/* Serial.print("ACSoffset = " ); // shows pre-scaled value
 Serial.print(ACSoffset);
 */
delay(2);

/*
Serial.print("Initializing SD card...");

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  if (SD.exists("test.txt")) {
    Serial.println("test.txt exists.");
  } else {
    Serial.println("test.txt doesn't exist.");
  }
```

```
 // open a new file and immediately close it:
 Serial.println("Creating test.txt...");
 myFile = SD.open("test.txt", FILE_WRITE);
 myFile.close();

 // Check to see if the file exists:
 if (SD.exists("test.txt")) {
   Serial.println("test.txt exists.");
 } else {
   Serial.println("test.txt doesn't exist.");
 }

 // delete the file:
 Serial.println("Removing test.txt...");
// SD.remove("test.txt");

 if (SD.exists("test.txt")) {
   Serial.println("test.txt exists.");
 } else {
   Serial.println("test.txt doesn't exist.");
 }

myFile = SD.open("test.txt", FILE_WRITE);

 String string1 = "Vbat;" ;
 String string2= string1 + "Vbat2;";
 String string3= string2 + "Vbat3;";
 myFile.println(string3); // write number to file

 String val0 = String(ACSoffset);
String val1 = val0 + ";" ;
```

```
//String val2= ACSoffset + ";" ;
//String val3= ACSoffset + ";" ;

 myFile.println(" 1;"); // write number to
myFile.println("2;"); // write number to file
myFile.println("3;"); // write number to file
   myFile.close(); // close file
 */
}

void loop(){

ACS_RawValue=0;
ACS_Voltage=0;
ACS_Amps=0;
Vbat_RawValue=0;
Vbat=0;
Vpv=0;


 if (stringComplete) {
 //   Serial.println(inputString);
   data= inputString.toFloat();

   data1=string1.toFloat();
   data2= string2.toFloat();
   data3= string3.toFloat();


   //data1=data1+1;
   //data2=data2+1;
```

```
   //data3=data3+1;




 //   date_totale=String(data1); //  + ';' + data2 + ';' + data3 + ';'
//    date_totale=date_totale + ";" + char(data3) + ";";

   // Serial.println(String(data1) + ";" + String(data2) + ";" + String(data3) + ";"); // + "\r\n"
//    Serial.println(data1+1);

//    Serial.println(data1+1);
//    Serial.write(data1+1);
//    Serial.println(data2+1);
 //   Serial.write(data2+1);

 //   Serial.println(data3+1);



//    Serial.write(data3+1);
   // clear the string:
   inputString = "";
   stringComplete = false;
   semnal_validare=not(semnal_validare);
 }


 for(int i=0;i<150;i++)
 {
```

```
ACS_RawValue+= analogRead(analogIn);
ACS_Voltage+= (analogRead(analogIn) / 1024.0) * tensiune_Arduino; // Gets you mV
ACS_Amps+= (((analogRead(analogIn) / 1024.0) * tensiune_Arduino - ACSoffset) / mVperAmp);

//citesc tensiunea bateriei
Vbat_RawValue+= analogRead(analogVin);
Vbat+=(analogRead(analogVin)/1024.0)*tensiune_Arduino;

//citesc tensiunea panoului
Vpv_RawValue+= analogRead(analogVpv);
Vpv+=(analogRead(analogVpv)/1024.0)*tensiune_Arduino;

delay(2);
 }

ACS_RawValue=ACS_RawValue/150;
ACS_Voltage=ACS_Voltage/150;
ACS_Amps=ACS_Amps/150;
Vbat_RawValue=Vbat_RawValue/150;
Vbat=Vbat/150*4;
Putere_consumata=ACS_Amps*Vbat;

/*

myFile = SD.open("test.txt", FILE_WRITE);
String string1 = "volti";
 myFile.println(Vbat + string1); // write number to file
   myFile.close(); // close file

 Serial.print("ACS Raw Value = " ); // shows pre-scaled value
 Serial.print(ACS_RawValue);
```

```
Serial.print("\t ACS_mV = "); // shows the ACS_Voltage measured

Serial.print(ACS_Voltage,3); // the '3' after ACS_Voltage allows you to display 3 digits after decimal point

Serial.print("\t ACS_Amps = "); // shows the ACS_Voltage measured

Serial.println(ACS_Amps,3); // the '3' after ACS_Voltage allows you to display 3 digits after decimal point


 Serial.print("\t Vbat_raw = "); // shows the ACS_Voltage measured

Serial.print(Vbat_RawValue,3); // the '3' after ACS_Voltage allows you to display 3 digits after decimal point

Serial.print("\t Vbat = "); // shows the ACS_Voltage measured

Serial.println(Vbat,3); // the '3' after ACS_Voltage allows you to display 3 digits after decimal point

Serial.print("\t Putere_consumata = "); // shows the ACS_Voltage measured

Serial.println(Putere_consumata,3); // the '3' after ACS_Voltage allows you to display 3 digits after decimal point

*/


// delay(2500);


// sistemul de genstionare a reincacarcarii acumulatorului

//Vpv=17;

if  ((Vpv>Vbat) && (Vbat<14))

  { if  (ACS_Amps<1)

    { if (duty_cycle>=1 ) { //logica inversa

    duty_cycle=duty_cycle-1;

    }

      analogWrite(pin_pwm,duty_cycle); }

    else if (ACS_Amps>2) {

    if (duty_cycle+1<255 && duty_cycle>=1 ) { //logica inversa

    duty_cycle=duty_cycle+1;

    }

    analogWrite(pin_pwm, duty_cycle);
```

```
    }
   }
   else if ((Vpv>Vbat) && (Vbat>=14))
   { if (duty_cycle+10<255) {
     duty_cycle=duty_cycle+1;
     }
     analogWrite(pin_pwm, duty_cycle);
     }


    else {
     duty_cycle=255;
  analogWrite(pin_pwm, 255);
    }
/*  Serial.print("\t duty_cycle = ");
  Serial.println(duty_cycle);
*/

  Serial.println('*'+String(Vbat) + ";" + String(ACS_Amps) + ";" + String(duty_cycle) + ";"); // + "\r\n"
}


void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    index=index+1;
    // add it to the inputString:
    inputString += inChar;
    if (inChar == ';') {
     j=j+1;
```

```
  if (j==1) {

  string1=inputString;
  inputString="";

  }




 if (j==2) {
 string2 = inputString;
 inputString="";
}

if (j==3) {
 string3 = inputString;
 inputString="";
}



 }
 // if the incoming character is a newline, set a flag so the main loop can
 // do something about it:
 if (inChar == '\n') {
  stringComplete = true;
  index=0;
  j=0;
 }
}
```

}

7.4. Optimization control supervisor | SCADA system on Raspberry Pi 3

The code below is written in python programming language. It represents the code that will run on the Raspberry Pi configuration.

```
def receptie(input):
 return;


def is_number(s):
   try:
      float(s)
      return True
   except ValueError:
      pass



   return False


import sqlite3
import time
import datetime
#timp=str(datetime.datetime.utcnow())
timp=str(datetime.datetime.now())
nume='exemple'+timp+'.db'
conn = sqlite3.connect(nume)


test="-1;-1;-1;\r\n"
missed=test.rstrip(';').split(';')


import serial
import time
```

```
ser2 = serial.Serial('/dev/ttyACM0', 9600, timeout=2)
ser = serial.Serial('/dev/ttyACM1', 9600, timeout=2)

import sys
sys.path.insert(0, "..")
import time
import datetime
from datetime import datetime


from opcua import ua, Server


if __name__ == "__main__":

    # setup our server
    server = Server()
    server.set_endpoint("opc.tcp://192.168.0.50:4840/freeopcua/server/")


    # setup our own namespace, not really necessary but should as spec
    uri = "http://examples.freeopcua.github.io"
    idx = server.register_namespace(uri)

    # get Objects node, this is where we should put our nodes
    objects = server.get_objects_node()




    # populating our address space
```

```
    myobj = objects.add_object(idx, "MyObject")
    #myvar1 = myobj.add_variable(idx, "tensiune", 10)
    #myvar2 = myobj.add_variable(idx, "curent", 100)

    myvar1 = myobj.add_variable(idx, "Vpv", -1)
    myvar2 = myobj.add_variable(idx, "Ipv", -1)
    myvar3 = myobj.add_variable(idx, "duty_cycle", -1)

    myvar4 = myobj.add_variable(idx, "Vbat", -1)
    myvar5 = myobj.add_variable(idx, "Ibat", -1)
    myvar6 = myobj.add_variable(idx, "duty_cycle_bat", -1)

    myvar1.set_writable()    # Set MyVariable to be writable by clients
    myvar2.set_writable()
    myvar3.set_writable()

    myvar4.set_writable()
    myvar5.set_writable()
    myvar6.set_writable()

    # starting!
    server.start()

    c = conn.cursor()
    conditie=0

# Create table
    c.execute('''CREATE TABLE achizitii
          (timp text,Vpv text, Ipv text, duty_cycle text,Vbat text,Ibat text,duty_cycle_bat text)''')

    #initializare
```

```
while conditie==0 :
    read_serial=ser.readline()

    read_serial2=ser2.readline()
    if ( (ser.inWaiting()>0) & (len(read_serial.rstrip(';').split(';'))==4) & (ser2.inWaiting()>0) & (len(read_serial2.rstrip(';').split(';'))==4)):

        #if ( (read_serial=='\r\n') :
        #   my_list=test.rstrip(';').split(';')
        #else :
        #   my_list=read_serial.rstrip(';').split(';')
        my_list=read_serial.rstrip(';').split(';')
        caracter_start=my_list[0]

        my_list2=read_serial2.rstrip(';').split(';')
        caracter_start2=my_list2[0]

        if ( (caracter_start[0][:1]=='*') & is_number(my_list[1]) & is_number(my_list[2]) & (caracter_start2[0][:1]=='*') & is_number(my_list2[1]) & is_number(my_list2[2]) ) :

            caracter_start=my_list[0].rstrip('*').split('*')
            caracter_start2=my_list2[0].rstrip('*').split('*')

            myvar1.set_value(caracter_start[1])
            myvar2.set_value(my_list[1])
            myvar3.set_value(my_list[2])

            Pv=myvar1.get_value()

            myvar4.set_value(caracter_start2[1])
            myvar5.set_value(my_list2[1])
```

```
            myvar6.set_value(my_list2[2])


            Vbat=myvar4.get_value()


            #Ibat=myvar5.set_value(my_list2[1])
            #duty_cycle_bat=myvar6.set_value(my_list2[2])


            Pv_anterior=Pv
            #Ipv_anterior=Ipv
            #duty_cycle_anterior=duty_cycle
            Vbat_anterior=Vbat
            #Ibat_anterior=Ibat
            #duty_cycle_bat_anterior=duty_cycle


            conn.commit()
            ser.flushInput()
            ser2.flushInput()
            print(str(Pv) + ';' + str(myvar2.get_value()) + ';' + str(myvar3.get_value()) + ';' )
            print(str(Vbat) + ';' + str(myvar5.get_value()) + ';' + str(myvar6.get_value()) + ';' )
            c.execute(" INSERT INTO achizitii values (?,?,?,?,?,?,?)",(timp,myvar1.get_value(),myvar2.get_value(),myvar3.get_value(),myvar4.get_value(),myvar5.get_value(),myvar6.get_value()))
            conditie=1
            time.sleep(2)


    try:
        timp_inregistrare=1000000
        count = 1
        while True:
            timp_esantionare=abs(datetime.now().microsecond-timp_inregistrare>900000)


            time.sleep(0.85)
```

```
        read_serial=ser.readline()
        read_serial2=ser2.readline()

        timp=str(datetime.now())

        if ( (ser.inWaiting()>0) & (len(read_serial.rstrip(';').split(';'))==4) & (ser2.inWaiting()>0) & (len(read_serial2.rstrip(';').split(';'))==4)):

            my_list=read_serial.rstrip(';').split(';')
            caracter_start=my_list[0]

            my_list2=read_serial2.rstrip(';').split(';')
            caracter_start2=my_list2[0]

            if ((len(caracter_start)>=1) & (len(caracter_start2)>=1)) :

                if ( (caracter_start[0][:1]=='*') & is_number(my_list[1]) & is_number(my_list[2]) & (caracter_start2[0][:1]=='*') & is_number(my_list2[1]) & is_number(my_list2[2]) ) :

                    caracter_start=my_list[0].rstrip('*').split('*')
                    caracter_start2=my_list2[0].rstrip('*').split('*')

                    #if ( (my_list[0] != '') & (my_list[1] != '') & (my_list[2] != '')):
                    myvar1.set_value(caracter_start[1])
                    myvar2.set_value(my_list[1])
                    myvar3.set_value(my_list[2])

                    Pv=myvar1.get_value()

                    myvar4.set_value(caracter_start2[1])
```

```
                myvar5.set_value(my_list2[1])
                myvar6.set_value(my_list2[2])

                Vbat=myvar4.get_value()

                print(str(Pv) + ';' + str(myvar2.get_value()) + ';' + str(myvar3.get_value()) + ';' )
                print(str(Vbat) + ';' + str(myvar5.get_value()) + ';' + str(myvar6.get_value()) + ';' )

                c.execute(" INSERT INTO achizitii values (?,?,?,?,?,?,?)",(timp,Pv,myvar2.get_value(),myvar3.get_value(),Vbat,myvar5.get_value(),myvar6.get_value()))

                #INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

                #myvar2.set_value(my_list[1])
                Pv_anterior=Pv

                Vbat_anterior=Vbat


                conn.commit()
                ser.flushInput()
                ser2.flushInput()
                time.sleep(0.1)


                timp_inregistrare=datetime.now().microsecond



    finally:
        #close connection, remove subcsriptions, etc
```

conn.close()

server.stop()

7.5. Takagi Sugeno Observer model in Simulink/Matlab

The script for computing the gains of the observer can be found in the following Matlab script:

```
clear all
clc
run initializare_variablie_T_S_observator.m

z1_max=1;  %u
z1_min=0.05;

z2_max=22.1;
z2_min=0;

z3_max=4;
z3_min=-1;

% %%%%%%%%%%%%%%%%%%%%%% LMI 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


A1=[0 -z1_max/(2*Ca) 0; z1_max/(2*L) -((Rb+RL)/L) -1/L; 0 1/Cb 0];


%
% %%%%%%%%%%%%%%%%%%%%%% LMI 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A2=[0 -z1_max/(2*Ca) 0; z1_max/(2*L) -((Rb+RL)/L) -1/L; 0 1/Cb 0];





%_______________________Declararea variabilelor___________________________%

p1=sdpvar(3,3,'symmetric');
p3=sdpvar(1,1,'symmetric');
C=[0 1 0;0 0 1];

q11=sdpvar(3,2,'full');
q13=sdpvar(3,2,'full');
q21=sdpvar(3,2,'full');
q23=sdpvar(3,2,'full');
q31=sdpvar(3,2,'full');
q33=sdpvar(3,2,'full');
q41=sdpvar(3,2,'full');
q43=sdpvar(3,2,'full');

alfa=500;
```

```
matrice1=[A1'*p1+p1*A1-C'*q11'-q11*C+p1+2*alfa*p1           p1; ...
                  p1                          -(p3*eye(3,3))      ];
matrice2=[A2'*p1+p1*A2-C'*q21'-q21*C+p1+2*alfa*p1          p1; ...
                  p1                          -(p3*eye(3,3))    ];
% matrice3=[A3'*p1+p1*A3-C'*q31'-q31*C+p1           p1; ...
%                   p1                           -(p3*eye(3,3))     ];
% matrice4=[A4'*p1+p1*A4-C'*q41'-q41*C+p1           p1; ...
%                   p1                           -(p3*eye(3,3))    ];


LMI=    [[matrice1]<0];
LMI=LMI+[[matrice2]<-1e-5];
% LMI=LMI+[[matrice3]<-1e-5];
% LMI=LMI+[[matrice4]<-1e-5];

LMI=LMI+[p1>0];
LMI=LMI+[p3>0];


solvesdp(LMI)

% % options = sdpsettings('verbose',1,'solver','bmibnb');
% sdpsettings('solver','sedumi')
% sdpsettings('solver','sdpt3')
% sdpsettings('solver','fmincon')
% solvesdp(LMI,[],sdpsettings('solver','sedumi'))

P1=double(p1);
P3=double(p3);

Q11=double(q11);
L1=inv(P1)*Q11;


Q21=double(q21);
L2=inv(P1)*Q21;


Q31=double(q31);
L3=inv(P1)*Q31;


Q41=double(q41);
L4=inv(P1)*Q41;




conditie1=real(eig(double(matrice1)));
conditie2=real(eig(double(matrice2)));
% conditie3=real(eig(double(matrice3)));
% conditie4=real(eig(double(matrice4)));

if eig(double(P1))>=0
else
    display('P1<0 !');
```

```
end

if eig(double(P3))>=0
else
    display('P3<0 !');
end


if conditie1<0
else
    display('conditia 1 nu este indeplinita!');
end

if conditie2<0
else
    display('conditia 2 nu este indeplinita!');
end

% if conditie3<0
% else
%     display('conditia 3 nu este indeplinita!');
% end
%
% if conditie4<0
% else
%     display('conditia 4 nu este indeplinita!');
% end
```

In Figure 7-1 and Figure 7-2 are presented the main Simulink schemes used for simulating the presented results.

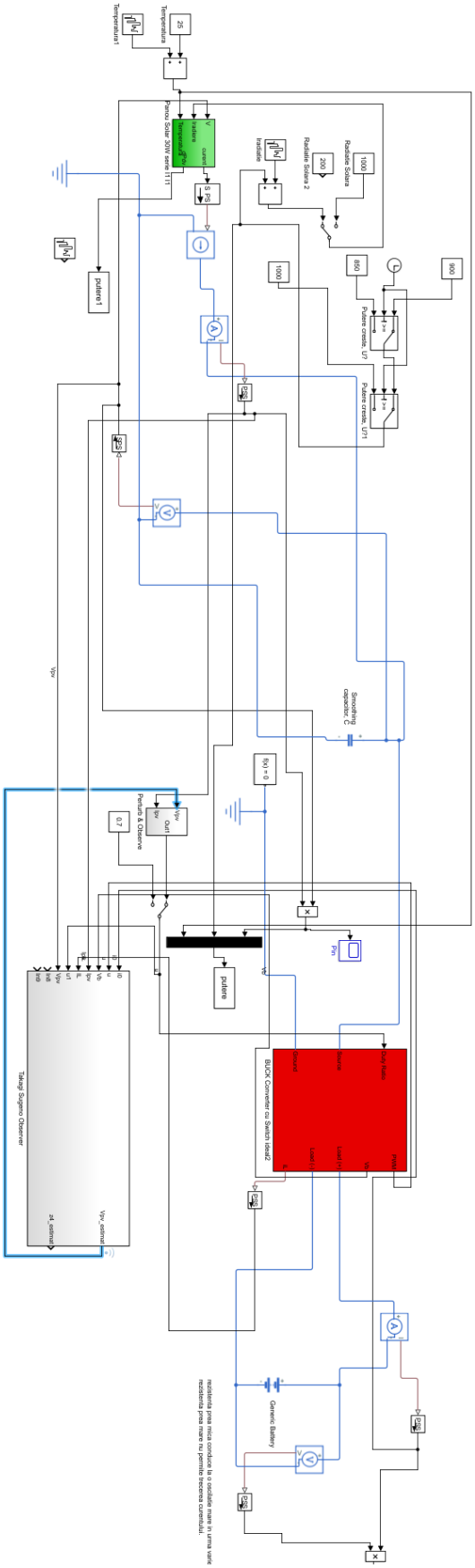


Figure 7-1 Simulink TS global control schematic

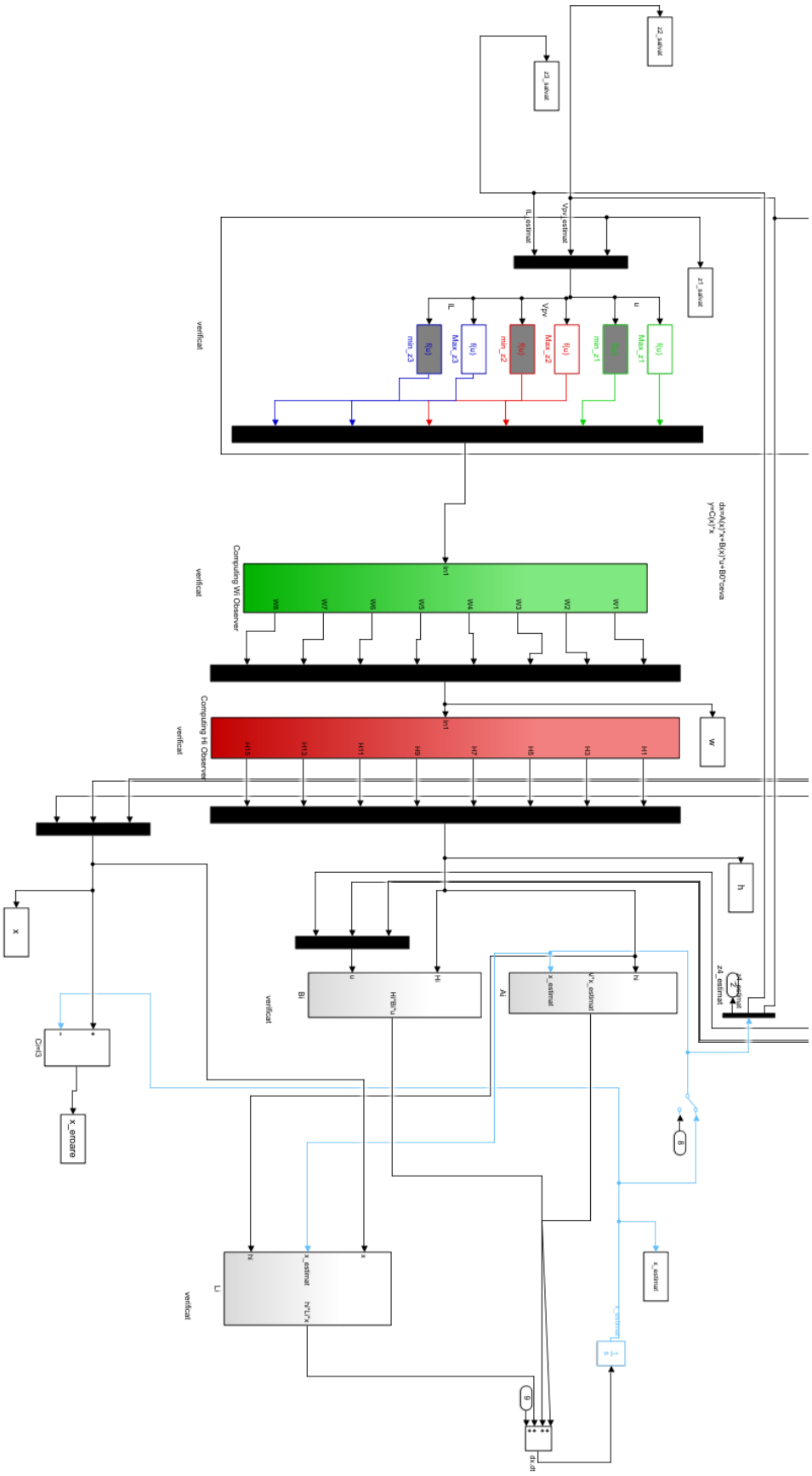


Figure 7-2 Simulink TS observer implementation

7.6. Supervisory system – fuel cell case study model in Simulink

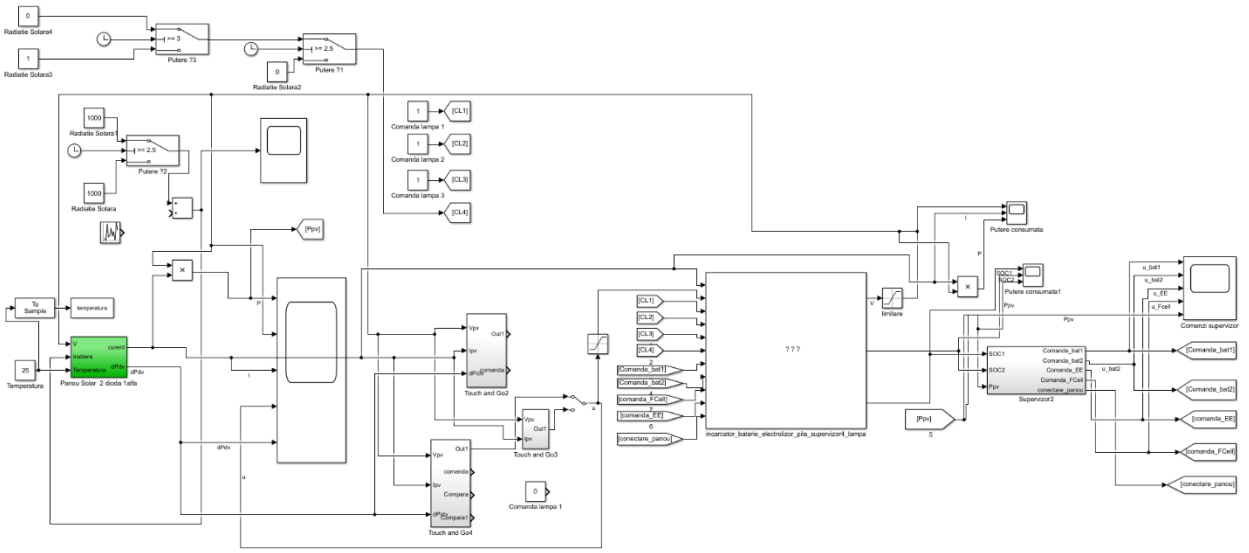


Figure 7-3 Supervisory system – fuel cell

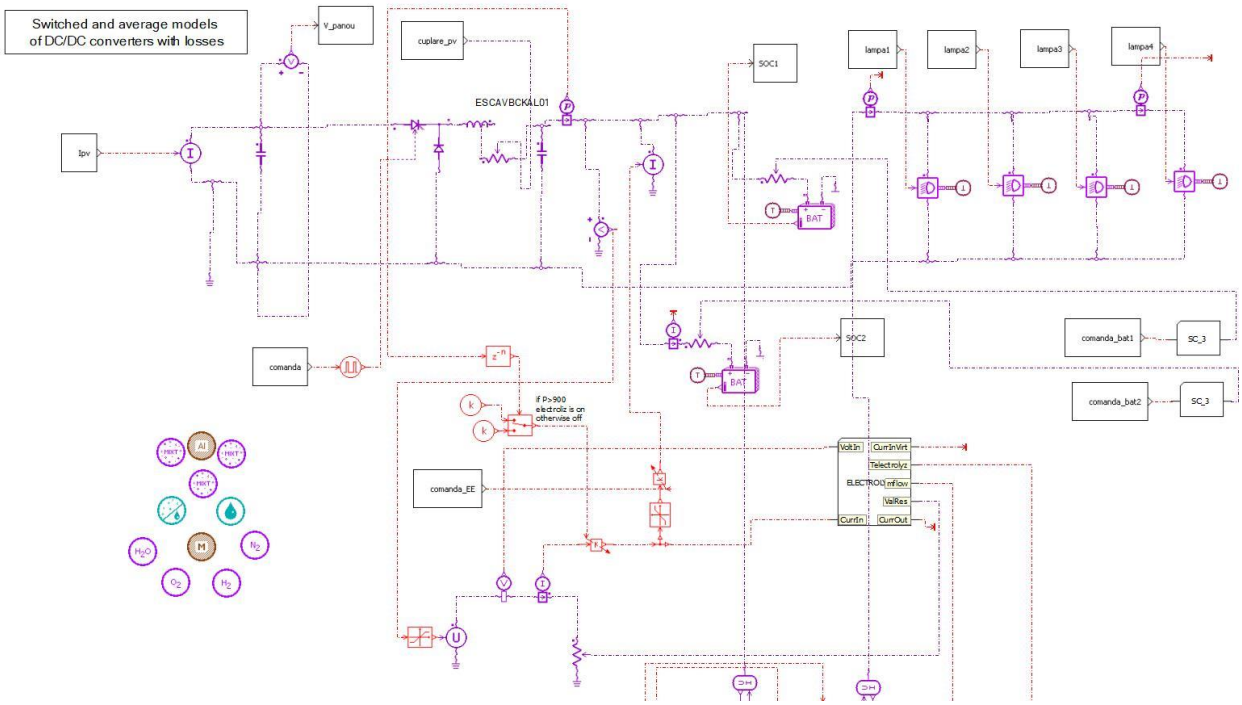


Figure 7-4 DC/DC converter and fuel cell in AMESIM

7.7. Computing R-S-T polynomials in Matlab

```
% Calculare param RST

% definim functia de transfer a sistemului B/A
t_esantionare=1/320000
L = 165e-6; %300e-6; % 300
C = 220e-6; %440e-6;
R = 10;%10;
s = tf('s');
V_max_sistem=22;
H = V_max_sistem/(s*s*L*C + L/R * s +1);

% calculam raspunsul sistemului la treapta
% stepinfo(H)

%trasam raspunsul la treapta al sistemului
% step(H)

 ts=1/16000%16000;%1/32000
%cum alegem valorile lui tzeta si omega
wn=10000
tzeta=0.7

hd2 = c2d(H,ts,'zoh');
hd = tf( hd2.Numerator, hd2.Denominator, ts, 'variable','z^-1');
A = cell2mat(hd.Denominator);
B = cell2mat(hd.Numerator);

% scriem functia  impusa
Qc = tf(wn*wn,[1 2*tzeta*wn wn*wn])
Q = c2d(Qc,ts,'zoh')

%trecem in z-1
Q = tf(Q.Numerator,Q.Denominator,ts,'variable','z^-1')

%notam polinomul caracteristic
P = cell2mat(Q.Denominator)
p = P;

% determinam lungimile vectorilor
na = length(A) - 1;
nb = length(B) - 1;
np = length(P) - 1;

ns = nb - 1; % dim polinom S
nr = na - 1; % din polinom R
if(np <= na+nb-1)
    display('P OK');
end


a=filt(A,1,ts);
aux=filt([1 -1],1,ts);
a=a*aux;
a=cell2mat(a.Numerator);%S'=(1-z^-1)*S
```

```
na = length(a) - 1;
nb = length(b) - 1;


ns = nb - 1; % dim polinom S
nr = na - 1; % din polinom R
if(np <= na+nb-1)
    display('P OK');
end

%introducem in matricea A, partea impusa a lui S=HS*S' , iar in B, partea
%impusa a lui R=HR*R'

 M = [ [1 a(2:end) 0]' [0 1 a(2:end)]' [0 b(2:end) 0 0]' [0 0 b(2:end) 0]' [0
0 0 b(2:end)]'];

p = [P zeros(1,length(M)-length(P))]'; % p =[P zeros(1,length(M)-length(P)]
x = M\p;
S = x(1:ns+1)' % determinam coeficientii polionmului S
temp = filt(S,1,ts)* aux;

S=cell2mat(temp.Numerator)

R = x(ns+2:end)' % determinam coeficientii polionmului R

bsum = sum(B);
T = P/bsum

wn2=12000
tzeta=0.8

Hm = wn2^2/(s^2+2*tzeta*wn2*s+wn2^2)

Hm2 = c2d(Hm,ts,'zoh');
hdm = tf( Hm2.Numerator, Hm2.Denominator, ts, 'variable','z^-1');

Am = cell2mat(hdm.Denominator)
Bm = cell2mat(hdm.Numerator)

tf(conv(A,S),P)
[Gm,Pm,Wcg,Wcp]=margin(tf(conv(A,S),P))
```

8. BIBLIOGRAPHY

[1] Chian-Song Chiu, “T-S Fuzzy Maximum Power Point Tracking Control of Solar Power Generation Systems” in IEEE Transactions on Energy Conversion, vol. 25, no. 4, Dec. 2010.

[2] Z. Lendek, T. Guerra, R. Babuška and B. De Schutter, “Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models” in Studies in Fuzziness and Soft Computing, vol. 262, chapters II,III,IV, Springer, 2010.

[3] S. Olteanu, A. Aitouche, L. Belkoura, A. Jouni, “Embedded P.E.M. Fuel Cell Stack Nonlinear Observer by means of a Takagi-Sugeno Approach” in Studies in Informatics and Control, ISSN 1220-1766, vol. 24 (1), p. 61-70, 2015.

[4] K. Ishaqu, Z. Salam, H. Taheri, “An improved two-diode photovoltaic (PV) model for PV system,” Power Electronics, Drives and Energy Systems (PEDES) & 2010 Power India, 2010 Joint International Conference, p. 1-5, Dec. 2010.

[5] N. Shannana, N.Yahayab and B. Singhc, “Single-Diode Model and Two-Diode Model of PV Modules: A Comparison,” in 2013 IEEE International Conference on Control System, Computing and Engineering, Dec. 2013, Penang, Malaysia.

[6] Z. Lendek, T. Guerra and R. Babuška, “On non-PDC local observers for TS fuzzy systems” in Fuzzy Systems (FUZZ), in IEEE International July 2010.

[7] A. Rivai and N. Abd. Rahim, “A low-cost Photovoltaic (PV) array Monitoring System” in IEEE Conference on Clean Energy and Technology (CEAT) 2013.

[8] A. V. Kumar, A. M. Parimi and K.Rao , “Implementation of MPPT Control Using Fuzzy Logic in Solar-Wind Hybrid Power System” in SPICES, IEEE,2015.

[9] K. Ishaqu, Z. Salam, H. Taheri, “An improved two-diode photovoltaic (PV) model for PV system,” Power Electronics, Drives and Energy Systems (PEDES) & 2010 Power India, 2010 Joint International Conference, p. 1-5, Dec. 2010.

[10] N. Shannana, N.Yahayab and B. Singhc, “Single-Diode Model and Two-Diode Model of PV Modules: A Comparison,” in 2013 IEEE International Conference on Control System, Computing and Engineering, Dec. 2013, Penang, Malaysia.

[11] G. Hsief, C. Tsai and H. Hsief, “Photovoltaic Power-Increment-Aided Incremental-Conductance Maximum Power Point Tracking using Variable Frequency and Duty Controls” IEEE on Power Electronics for Distributed Generation Systems, 2012

[12] G. Kish, J. Lee, P. Lehn, “Modelling and control of photovoltaic panels utilizing the incremental conductance method for maximum power point tracking ”, IET Renewable Power Generation, 2011

[13] F. Kazan, S. Karaki, R. Jabr and M. Mansour, “Maximum Power Point Tracking Using Ripple Correlation and Incremental Conductance”, Universities Power Engineering Conference (UPEC), 2012 47th International, p. 1-6

[14] N. Zakzouk, A. Abdelsalam, A. Helal,”Modified Variable-step Incremental Conductance Maximum Power Point Tracking Technique for Photovoltaic Systems”, Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE

[15] M. Villalva, J. Gazoli, and E. Filho, “Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays”, in Power Electronics, IEEE Transactions on 2009, p. 1198 – 1208.

[16] K. Ishaqu, Z. Salam, H. Taheri, “An improved two-diode photovoltaic (PV) model for PV system,” Power Electronics, Drives and Energy Systems & 2010 Power India, 2010 Joint International Conference, 2010.

[17] N. Shannana, N.Yahayab and B. Singhc, “Single-Diode Model and Two-Diode Model of PV Modules: A Comparison,” in 2013 IEEE International Conference on Control System, Computing and Engineering, 2013.

[18] G. Kish, J. Lee, P. Lehn, “Modelling and control of photovoltaic panels utilizing the incremental conductance method for maximum power point tracking ”, IET Renewable Power Generation, 2011.

[19] M. Villalva, J. Gazoli, and E. Filho, “Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays”, in Power Electronics, IEEE Transactions on 2009, pp. 1198 – 1208.

[20] G. Hsief, C. Tsai and H. Hsief, “Photovoltaic Power-Increment-Aided Incremental-Conductance Maximum Power Point Tracking using Variable Frequency and Duty Controls” IEEE on Power Electronics for Distributed Generation Systems, 2012.

[21] S. O. Amrouche, D. Rekioua and T. Rekioua, "Overview of energy storage in renewable energy systems," *2015 3rd International Renewable and Sustainable Energy Conference*, Marrakech, 2015, pp. 1-6.

[22] N. Garimella and N. K. C. Nair, "Assessment of battery energy storage systems for small-scale renewable energy integration," *TENCON 2009 - 2009 IEEE Region 10 Conference*, Singapore, 2009, pp. 1-6.

[23] K. Agbossou, M. Kolhe, J. Hamelin and T. K. Bose, "Performance of a stand-alone renewable energy system based on energy storage as hydrogen," in *IEEE Transactions on Energy Conversion*, vol. 19, no. 3, pp. 633-640, Sept. 2004.

[24] D. M. Ali and S. K. Salman, "A Comprehensive Review of the Fuel Cells Technology and Hydrogen Economy," *Proceedings of the 41st International Universities Power Engineering Conference*, Newcastle-upon-Tyne, 2006, pp. 98-102.

[25] Øystein Ulleberg, “Modeling of advanced alkaline electrolyzers: a system simulation approach”, International Journal of Hydrogen Energy, Volume 28, Issue 1, January 2003, pp 21-33.

[26] Y. Wang, K. S. Chen, J. Mishler, S. C. Cho and X. C. Adroher, “A review of polymer electrolyte membrane fuel cells: Technology, applications, and needs on fundamental research”, Applied Energy, Volume 88, Issue 4, April 2011, pp. 981-1007, ISSN 0306-2619.

[27] R. Atia and N. Yamada, "Sizing and Analysis of Renewable Energy and Battery Systems in Residential Microgrids," IEEE Transactions on Smart Grid, vol. 7, no. 3, pp. 1204-1213, May 2016.

[28] S. C. Olteanu, A. Aitouche, L. Belkoura and A. Jouni, “Embedded P.E.M. Fuel Cell Stack Nonlinear Observer by means of a Takagi-Sugeno Approach”, Studies in Informatics and Control, ISSN 1220-1766, vol. 24 (1), pp. 61-70, 2015.

[29] C.Lupu,D.Popescu,C.Dimon, Soluții Practice de Conducere a Proceselor Neliniare, Editura Politehnica Press, București 2010

[30] I.D.Landau, Gianluca Zito, Digital Control Systems-Design, Identification and Implementation, Springer-Verlag London Limited, 2006.

[31] I.Dumitrache, T. Dragomir, Automatica Volumul II, Editura Academiei Române, București, 2013.

[32] D.Popescu, R.Dobrescu, F.Ionescu, D.Stefanoiu, Modelare in Ingineria Proceselor Industriale, Editura AGIR, Bucuresti 2011.
[33] M.Voicu, Teoria Sistemelor, Editura Academei Romane, Bucuresti 2008.
[34] K.Y.Astrom,R.M. Murraay, Feedback Systemes : An introduction for Scientists and Engineers, Princeton University Press,2008.
[35] D.Thanth, A.Borole, *Wi-Fi Based Smart Energy Meter*, International Journal of Research in Computer and Communication Technology, Vol. 4, No. 4, April-2015.
[36] L.Durkop, J.Imtiaz, H.Tresk, L.Wiefnieski, J.Jasperneite, Using OPC-UA for the Autoconfiguration of Real-time Ethernet Systems, inIT
[37] A. Oulasvirta, G.Abowd, User Interface Design in the 21st Century.
[38] J.Imtiaz, J.Jasperneite, Scalability of OPC-UA Down to the Chip Level Enables Internet of Things, inIT.
[39] S.Gruner, H.Ali, M.Sattar, J.Pfrommer, F.Palm, *RESTful Industrial Communication with OPC UA*, Transactions on Industrial Informatics.
[40] M.Hoffmann, C.Buscher, T.Meisen, S.Jeschke, Continuous integration of field level production data into top-level information systems using the OPC interface standard.
[41] S.Kumar, A.Dixit, M.Mittal, A design of cost effective hybrid solar interface and management unit, june 2016.
[42] http://www.toptal.com/twitter-bootstrap/speeding-up-development-bootstrap.
[43] http://www.opto22.com/documents/2061_High_Performance_HMI_white_paper.pdf
[44] https://opcfoundation.org/about/opc-technologies/opc-ua/
[45] https://media.readthedocs.org/pdf/python-opcua/latest/python-opcua.pdf
[46] http://www.commsvr.com/UAModelDesigner/
[47] http://www.pyqtgraph.org/
[48] http://guide.openenergymonitor.org/
[49] Kamyar Mehran,Takagi-Sugeno Fuzzy Modeling for Process Control, Industrial Automation, Robotics and Artificial Intelligence (EEE8005)
[50] Z. Binder, “About a multimodel control methodology, algorithm, multi-processors,implementation and application." 8th IFAC World Congress 981-986, 1991.
[51] C. Miron, D. Popescu,A. Aitouche and N. Christov, “Observer based control for a PV system modeled by a Fuzzy Takagi Sugeno model”, in 2015 System Theory, Control and Computing (ICSTCC), 2015 19th International Conference, p. 652-657.
[52] C. Miron, S Frigioiu, S. Olteanu, A. Aitouche, N. Christov, Control Architecture for Low Power Photovoltaic MPPT Modules, 14th European Workshop on Advanced control and Diagnosis
[53] S.Olteanu, C. Torous, C. Miron, Model based MPPT control of a small photovoltaic panel, 2017 21st International Conference on System Theory, Control and Computing (ICSTCC)
[54] C. Miron, S.Olteanu, N. Christov, A. Aitouche, Architecture for Embedded Supervisory System of Distributed Renewable Energy Sources, CFP 7th International Conference on Systems and Control, 24-26 October 2018, Valencia, Spain
[55] C. Miron, D. Popescu, C. Petrescu, Designing control systems with distributed parameters, 2014 18th International Conference on System Theory, Control and Computing (ICSTCC)
[56] C. Miron, D. Popescu, I. Arsu, Control systems for a heat exchanger with distributed parameters, 2017 21st International Conference on System Theory, Control and Computing (ICSTCC)
[57] C. Miron, N. Christov, S.Olteanu, Energy management of photovoltaic systems using fuel cells, 2016 20th International Conference on System Theory, Control and Computing (ICSTCC)