Open Software and Manual for designing novel solar cells

Deliverable 4.7



Please consider the environment before printing this document.

This document is shared under Creative Commons License: **CC BY-NC 4.0**



In case of doubt please read the license conditions here:

https://creativecommons.org/licenses/by-nc/4.0/

## Technical References

|  |  |
| --- | --- |
| Project Acronym | GRECO |
| Project Title | Fostering a next generation of European Photovoltaic Society through Open Science |
| Project Coordinator | Universidad Politécnica de Madrid |
| Project Duration | 36 months |

|  |  |
| --- | --- |
| Deliverable No. | 4.7 |
| Dissemination level 1 | PU |
| Work Package | WP4 |
| Task | T4.2.4 |
| Lead beneficiary | P1. UPM |
| Contributing beneficiary(ies) | P10. RLI, P9. HZB |
| Due date of deliverable | 31-05-2020 |
| Actual submission date | 16-05-2020 |

1 PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)

## Document history

|  |  |  |  |
| --- | --- | --- | --- |
| V | Date | Beneficiary | Author |
| 1 | 30/03/2020 | UPM | 1. Martí |
| 2 |  |  |  |

## Table of Contents

[Technical References 3](#_Toc36550431)

[Document history 3](#_Toc36550432)

[Table of Contents 4](#_Toc36550433)

[Executive Summary 5](#_Toc36550434)

[Introduction 5](#_Toc36550435)

[1. The three terminal heterojunction bipolar transistor solar cell 6](#_Toc36550436)

[2. Python 3, Jupyter and Anaconda 7](#_Toc36550437)

[3. Git, GitHub and GitHub Desktop 8](#_Toc36550438)

[4. Lessons learnt in Open Science 9](#_Toc36550439)

[5. Summary 10](#_Toc36550440)

[Glossary- Abbreviations 10](#_Toc36550441)

[Disclaimers 11](#_Toc36550442)

## Executive Summary

In this deliverable we have implemented in open software the code that allows analytically modelling the “three-terminal heterojunction bipolar transistor solar cell”. The code is implemented in Python 3 within a Jupyter Open Notebook scheme. The implementation in the Jupyter environment allows the program to be well documented and self-contained. In this way, the code itself explains, not only what is done in the program in each step, but also how the equations are deduced so that the code and the manual for running it becomes the same thing. The program has been uploaded to GitHub that allows for the development of the code in an open and collaborative framework.

## Introduction

This deliverable is divided in four main sections:

* Section I guides the reader so that he can find information to understand the three-terminal heterojunction bipolar transistor solar cell (3T-HBTSC) in case he is interested in the physics behind this device.
* Section II teaches the reader how to install Python 3 and the Jupyter environment so they can develop further and adapt to their needs the software we have developed. One of the installations tools that allows you to install both Python and Jupyter is called “Anaconda” in GNU-Linux, Windows and Mac operative systems.
* Section III introduces the reader to “Git”, “GitHub” and “GitHub Desktop”, the collaborative framework where he can contribute in an organized way to the open and collaborative development of the software.
* Section IV, as a matter of conclusion, describes the lessons we have learnt regarding Open Science, which is the main motivation of the GRECO project, as a consequence of this work.

If the reader is already familiar with the concept above and wishes to access directly the code, he can do it now by navigating into our GitHub repository:

* https://github.com/amartiv/3\_Terminal\_Transistor\_Solar\_Cell

# The three-terminal heterojunction bipolar transistor solar cell

The “three-terminal heterojunction bipolar transistor solar cell” (3T-HBTSC) is a tandem of two solar cells in which the current is extracted through three terminals (Fig. 1) instead of the conventional two terminals. Extracting current through an additional terminal allows to extract the electric power independently of each cell what offers room for efficiency improvements. It is not the purpose of this deliverable to repeat here the physics behind the operation of this solar cell, furthermore when its operation and model is described in the software itself we have developed. There is by now an existing literature, some of it developed under GRECO, that describes the concept and that we summarize next:

* Martí, A.; Luque, A. Three-terminal heterojunction bipolar transistor solar cell for high-efficiency photovoltaic conversion. Nat. Commun. 2015, 6, 6902–6902.
* Zehender, M.H.; Warren, E.; Tamboli, A.; Martí, A.; Antolín, E. Demonstrating the GaInP / GaAs Three-Terminal Heterojunction Bipolar Transistor Solar Cell. 2019 IEEE 46th PVSC 2019.
* Antolín, E.; Zehender, M.H.; García-Linares, P.; Svatek, S.A.; Martí, A. Considerations for the Design of a Heterojunction Bipolar Transistor Solar Cell. IEEE J. Photovoltaics 2020, 10, 2–7.[GRECO]
* Martí, A.; García‐Linares, P.; Zehender, M.H.; Svatek, S.A.; Artacho, I.; Cristóbal, A.B.; González, R.; Carsten, B.; Ramiro, Í.; Cappelluti, F.; et al. Potential of the three-terminal heterojunction bipolar transistor solar cell for space applications. Eur. Sp. Power Conf. 2019. [GRECO]
* Zehender, M.; Antolín, E.; García-Linares, P.; Artacho, I.; Ramiro, I.; Villa, J.; Martí, A. Module interconnection for the three-terminal heterojunction bipolar transistor solar cell. AIP Conf. Proc. 2018, 2012, 64189.

|  |
| --- |
|  |

Fig. 1. Structure of a “three terminal heterojunction bipolar transistor solar cell” (3T-HBTSC). This figure is also included in the Jupyter Notebook implementing the code and the reader is referred to it for further details.

# Python 3, Jupyter and Anaconda

**Python 3** is the programing language in which our software has been implemented. **Jupyter** is the Open Notebook in which we have written the Python code. The advantage of using Jupyter as Notebook for Python is that allows a unique way of documenting the code. Fig. 1 and Fig. 2 are example that illustrate what we are saying where the reader can appreciate how easily figures and formulas can be visualized in the same code to better illustrate its operation.

|  |
| --- |
|  |

Fig. 2. Example of code as implanted on March 31st 2020 (code might change in updated versions). The potential of Jupyter as front end allows to document the code by even writing the equations in a “human” readable format.

However, what is an advantage of Python, can be turned into a problem. Python allows to use multiple subroutines, already written by others, in order to solve equations, do maths, read data from Excel files, export data, plot graphs etc. However, this milliard of subroutines depends on each other in a complicated way and they need to preserve compatibility all across the system when updated. Here is where Anaconda enters. **Anaconda**[[1]](#footnote-1) is a data science platform that will install Jupyter and Python in your computer (Linux, Mac or Windows) and will keep the subroutines consistently updated for you.

# Git, GitHub and GitHub Desktop

“**Git**”[[2]](#footnote-2) is a version control software. We will explain what is this with an example. Most of us, when we create a new version of a document, we save it with a new name, like “document\_version\_01”, “document\_version\_02”, etc. Moreover, when we write version 2, let’s say, we are afraid of deleting version 1 because, for example, we fear that we might still need something from version 1 that we might have accidentally deleted in version 2. Most of the times, even when the final version is concluded, only the disciplined ones delete the previous versions form their computers. In this way, if the final document is, let’s say 10 MB and you keep 10 previous versions, your computer might well end up in using 100 MB of storage capacity when it would only need 10 MB.

One of the clever things that Git does (it does a lot of clever things!) is that when you store a new version, it manages to store only the changes in a transparent way to the user. For example, if you change “a comma” inside a 10 MB file, storing the new version in the old way would demand another 10 MB of storage space while, by using “Git” protocol, storing the new version will imply a negligible amount of extra storage in your computer.

The second clever thing that “Git” protocol allows you to do is a kind of time travel because it can allow you, for example, to restore the version of your document or code just as it was, say, two months ago.

“Git” is available for GNU-Linux, Windows and Mac users[[3]](#footnote-3).

“**GitHub**”[[4]](#footnote-4) is a kind of cloud storage repository where you can store your codes as well as grant access to others to see it and collaborate with you in its development while keeping a robust control of the new versions that are being created. GitHub is “free”, as in beer, as long as you don’t collaborate, for example, with more than 3 people. But collaborating with as many people as you want will only cost you 4$/month. In addition to the code for modelling the 3T-HBTSC, this deliverable has been also uploaded and made available in our GitHub repository.

“**GitHub Desktop**”[[5]](#footnote-5) is a Graphical Use Interface for managing your development versions that integrates nicely with GitHub. Unfortunately, it is only available for Mac and Windows operative systems. It is discouraging that, having “Git” being originally created for GNU-Linux and shared freely with the world, “GitHub Desktop” had forgotten Linux users. In spite of this, GNU-Linux users have tools like “Git Cola”[[6]](#footnote-6) to manage graphically their Git files.

# Lessons learnt in Open Science

* Lesson 1: Opening your software into a collaborative framework such as Github **can be a painful process for beginners.** Some beginners will be afraid of showing poor programing skills if everybody can see their codes and feel as if they were continuously examined. On the other hand, feeling everybody can watch your code forces you to do it better.
* Lesson 2: The first group of people you probably will want to collaborate with are the mates in your team. For effective collaboration, all of them will have to learn and not all of them will be motivated to do so or will appreciate the advantages. Most of them will feel they have better things to do (and perhaps they have). Be ready to dedicate extra effort to, first learn by yourself and then, train others.
* Lesson 3: **It is not sufficient with acquiring a superficial level of knowledge of these tools** in order to effectively use them: an advanced level is need which increases the difficulty in implementing them since more time is needed to learn them.
* Lesson 4: **We recommend that collaborative tools, and computer tools that are in general necessary for the implementation of open science are thought, at least, at graduated level**. Later, when researchers have already their habits, it will be likely too late for changing them. Alternatively, or complementarily, aggressive professional **formation courses should be given** to adequately train personnel in the required computer skills.
* Lesson 5: Taking the initiative of opening your software at individual level is an step, but the initiative should likely be taken, at least, at institutional level in order to provide a uniform and consistent framework and platforms for their developments.
* Lesson 6: For using efficiently platforms like GitHub, a monthly fee is needed. Some researcher are reluctant to pay this fee and initiate the process of opening their software collaboratively because they fear their Institutions will not cover these expenses arguing it can be used for particular use, or the complicated paperwork needed to justify the expense against an inquisitive auditor, or the difficulty to charge the expense to a new project once the project that originated the software has finished. Perhaps **the creation of an European free platform, similar to GitHub, would be advisable for covering for life the development of software originated within a European Project at least, while this software is being developed.**
* Lesson 7: The spread in operative systems (GNU/Linux, Windows, Mac…) adds an additional complication since not all the tools are developed for all the operative systems. For example, GitHub Desktop only works in Mac and Windows which prevents GNU/Linux users from using it which is paradoxical since most of the Open tools existing for developing Open software seem to have been promoted by GNU/Linux developers.
* Lesson 8: Making your software “open” does not mean that the software you use is “opened”. For example, when you make freely available a pdf document, you probably have used Microsoft Word to initially create it an export it to pdf format. Microsoft Word is not Open software. In order to make your openness to progress an step forward, the tools you develop your software with should also be open. Because of this, in our example, we have chosen Jupyter and Python 3. However, we had no previous experience with these tools. Learning them has added additional pain and added additional developing times because, for example, algorithms we were able to implement in a few minutes (for example, those related to symbolic calculations) have now taken us weeks. Besides, we are not sure they work as well as the former ones, because this is only accumulated experience time can say. **Nevertheless, although it will depend on case by case basis, we recommend to make the move**. Now, more people will be able to test our software, which will increase its robustness. In the process, we have learnt new algorithms to do things (like plots, reading from Excel files…) that we ignored before. Open developing tools are much better documented that closed ones: you will find the instructions to do whatever you want to do and likely hundred of tutorials. **In essence, you are actually benefiting from others having opened their developments!** And not less, most of these tools are free as in beer.

# Technical summary

* To access the software we have developed for this deliverable, go to:

https://github.com/amartiv/3\_Terminal\_Transistor\_Solar\_Cell

* To install Jupyter and Python 3, we recommend the Anaconda environment:

https://www.anaconda.com/

* To Open a GitHub account, go to:

https://github.com/

* To Download GitHub Desktop, go to:

https://desktop.github.com/

* To use the git protocol outside the GitHub ecosystem (but also inside if you wish), go to

https://git-scm.com/

and surf for your particular need, depending on your operative system.

## Glossary- Abbreviations

* 3T-HBTSC: Three terminal heterojunction bipolar transistor solar cell.

## Disclaimers

‘Fostering a Next Generation of European Photovolatic Society Through Open Science' GRECO is a Collaborative Project funded by the European Commission under Horizon 2020. Contract: 787289, Start date of Contract: 01/06/2018; Duration: 36 months (3 years).

The authors are solely responsible for this information and it does not represent the opinion of the European Community. The European Community is not responsible for any use that might be made of the data appearing therein.

The use of “he” or “him” in this document does not imply any gender discrimination.



1. https://www.anaconda.com/ [↑](#footnote-ref-1)
2. https://git-scm.com/ [↑](#footnote-ref-2)
3. https://git-scm.com/downloads [↑](#footnote-ref-3)
4. https://github.com/ [↑](#footnote-ref-4)
5. https://desktop.github.com/ [↑](#footnote-ref-5)
6. https://git-cola.github.io/ [↑](#footnote-ref-6)