Høgskulen på Vestlandet

SOLar LAB

icam

Improving HVL SolarLab's Performance

Master Thesis Report

By Bastien Augé
Overseen by Professor Richard J. Grant
From 16 February to 26 June 2026

ACKNOWLEDGEMENTS

Before starting this report, I would like to express my grateful thanks to Richard J. Grant, my supervisor this semester, for his guidance and oversight of my work on my master's thesis. I would also like to acknowledge all the HVL technicians and engineers, especially Harald Moen, for their technical and logistical support, and for giving me their time despite having a lot of work during this spring semester.



Figure 1: HVL Bergen campus and SolarLab exterior installations

TABLE OF CONTENTS

I. INTRODUCTION 5
II. CONTEXT 5
A. Overview of the SolarLab 5
B. Themes of my MSRI 6
C. Deliverables 7
D. State of the art 7
III. ORGANISATION 10
A. Tasks management 10
B. Progress report 10
IV. WORKS AND RESULTS 11
A. Increased maximum production power 11
B. Correlation between weather, radiation and production 14
C. Automated energy management system 18
D. SolarLab crash management 22
E. Data analysis 28
F. SolarLab management 30
G. Other modifications 31
V. CONCLUSION AND PERSPECTIVE 34
VI. BIBLIOGRAPHY 35
VII. APPENDICES 36
Appendix 1: Structure of the SolarLab systems 36
Appendix 2: Calculations for PV wiring modifications 37
Appendix 3: Logic for production forecasting 38
Appendix 4: Calculation made at each step of the model 39
Appendix 5: Extract of the energy management script 40
Appendix 6: Switch and router mounting system 41
Appendix 7: Auto reboot state machine algorithm diagram 42
Appendix 8: Extract of the auto reboot script 43
Appendix 9: Explanation of unexpected production drop/rise 44

TABLE OF ILLUSTRATIONS

Figure 1: HVL Bergen campus and SolarLab exterior installations 2
Figure 2: Key figures of the SolarLab 6
Figure 3: The potential monthly power production of solar energy in Norway [1] 8
Figure 4: Graph of DC input of inverter 1 and 2 during a sun-filled day at the beginning..... 11
Figure 5: Diagram of the two different PV string wiring configurations 12
Figure 6: Box with MC4 connector after wiring modifications 13
Figure 7: Graph of DC input of inverter 1 and 2 during a sun-filled day after modifications ... 13
Figure 8: Key figures of the model 14
Figure 9: Operating steps of the model 15
Figure 10: Graph comparing theoretical and measured output of monocrystalline PV (W) ... 16
Figure 12: Graph comparing actual output, V1 and V2 model for monocrystalline PV 17
Figure 13: Algorithm diagram of the automated energy management system 19
Figure 14: Energy management GUIs 20
Figure 15: Shelly web app interface after the modifications 21
Figure 16: Production forecast integration logic 21
Figure 17: Power consumption of the communication devices 22
Figure 18: UPS inverter model chosen 23
Figure 19: Before and after the UPS inverter installation 23
Figure 20: Before, during and after the switches / router new implementation 24
Figure 21: SwitchBot devices use for auto reboot: Plug Mini, Bot and Hub Mini 24
Figure 22: SwitchBot app 25
Figure 23: Arm supporting the Bot, mounted on the battery cabinet 25
Figure 24: Crash monitor GUIs 27
Figure 25: Key figures of the auto-reboot system 27
Figure 26: Irradiance data from the pyranometer 28
Figure 27: Detailed production data 28
Figure 28: Total production, SOC and controllable loads data 29
Figure 29: Adapters made and installed 31
Figure 30: Crontab programming interface 32
Figure 31: Temperature variations under PVs between 13th and 17th of June 2026 33
Figure 32: SwitchBot Meters inside their support and installation of the under PV support ... 33

LIST OF TABLES

Table 1: Comparison of performance between conventional tilted and vertical PVs [5] 9
Table 2: Model accuracy evaluation 16
Table 3: Summary of updates that need to be carried out 34

I. INTRODUCTION

This report outlines the work carried out during my exchange semester as part of my MSRI at SolarLab, a research and teaching laboratory dedicated to solar energy, which forms part of the Department of Mechanical Engineering and Maritime Studies at HVL Bergen (Norway). Under the supervision of Richard J. Grant, I contributed to the development of this unique platform, which simulates a passive, energy-autonomous dwelling equipped with photovoltaic panels, an energy storage system and sensors.

This project, at the crossroads of technical innovation and education, is aimed to tackle specific challenges: optimising energy production, automating flow management, and enhancing the system's resilience to failures. Through a combination of theoretical and practical approaches, this document outlines how technical, software and hardware solutions have helped to improve the SolarLab.

This report is divided into several sections. The first section sets out the background to SolarLab. It includes a brief overview of the current state of the art. Next, I describe in detail the organisational structure put in place during this semester. Then, the most important section details the work I carried out and the results achieved. Finaly, I proposed some perspectives about the future of the SolarLab.

II. CONTEXT

A. Overview of the SolarLab

The SolarLab is a solar energy test laboratory located on HVL Bergen campus. This laboratory is mostly used for research and teaching on solar energy equipment. This infrastructure also allows students to work on their projects with some real data (production, radiation, …).

The SolarLab is organised around two areas. The first one, and the main one, is the solar photovoltaic (PV) part. The PV part simulates a passive and energy-autonomous house. The second one focuses on solar thermal research. My work during the semester was focused on the PV part. In this report, the reference to SolarLab will only refer to the PV part (of the SolarLab).

It is constituted of a variety of equipment that goes from solar power generation and storage to some sensors, communication devices and equipment that manages the SolarLab.

Here are the main pieces of equipment in the SolarLab:

- Solar power generation:
  - 6x monocrystalline PV for a total of 1680Wp
  - 6x polycrystalline PV for a total of 1650Wp
  - 6x bi-facial vertical PV for a total of 1440Wp
  - 3x Sunny Boy inverter (1.6kW DC – 1.5kW AC each)

- Power storage and management:
  - 1x Tesvolt TS25 battery Lithium NMC[1] (14.40kWh)

[1] NMC: Nickel-Manganese-Cobalt

  - Devices that allowed internet connexion and programming (Raspberry Pi, router, switches, …)
  - 1x Sunny Island bidirectional battery inverter (4,6kW AC)

- Power consumption:
  - SolarLab appliance: HVAC, light, computer, …
  - 3x pilotable load:
    - 1x Amina car charger
    - 1x reinforced socket for an emergency car charger
    - 1x electrical heater to dump the excess energy

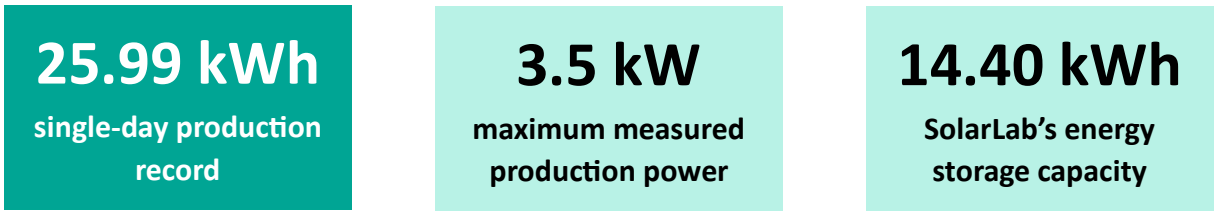


Figure 2: Key figures of the SolarLab

You will find in appendix 1 a diagram with all the equipment of the SolarLab systems. This diagram was one of my first task during my MSRI and I kept updating it during the semester.

In addition, the SolarLab has its own website (https://www.solarlab.online). The website provides an overview of the installations and publishes hourly data on irradiance measured directly by the pyranometer installed on the roof of the passive house.

B. Themes of my MSRI

The SolarLab is an education project, now in its fifth year, led by ICAM students. Equipment has been added gradually. It began with the installation of the most basic components of a solar system: an inverter and solar panels. As successive cohorts of students have worked on the project, the system has been upgraded: energy storage capacity, irradiance data measurement, and TIGO production optimisation system. More recently, controllable loads have been added to make the SolarLab "smarter", along with a programmable, heat able dual-flow ventilation system to ensure an ideal temperature.

Even if I was part of the Department of Mechanical Engineering and Maritime Studies of HVL for my MSRI, my main area of work was within HVL's Bergen SolarLab. I was working as a researcher assistant of Richard J. Grant. The objective of my exchange semester was to improve the SolarLab in a lot of different ways: improvement of the production system, automation of a load management system, research on vertical solar panel, …

In practice, the SolarLab simulates an energy autonomous Norwegian "hytte" (or cabin). These "hytte" are very common small secondary residences that Norwegians own. They are mostly built off grid in the mountains. The SolarLab allowed people to work and develop tools for remote habitat directly within the city of Bergen.

On a day-to-day basis I was working autonomously on my different subjects and got support from the HVL technician/engineer.

C. Deliverables

The aim of this semester was to deliver operational solutions to optimise production, automate energy management and enhance the resilience of the SolarLab. Key deliverables included:

- Technical documents: Procedures for modifying PV cabling, an Excel-based production prediction model, a database of system failures, …
- Python programmes: Automatic management of controllable loads and a restart system with fault detection.
- Hardware installations: back up power system, 3D-printed mounts and enclosures, plug adaptors, temperature sensors.
- Organisational improvements: generic email accounts, network configuration, ...

The work I realised during this semester was designed to be reusable and well-documented; these deliverables ensure the laboratory's autonomy whilst facilitating future work.

D. State of the art

The SolarLab, as an autonomous photovoltaic microgrid, is part of a context in which the challenges of solar system generation, management and resilience are at the centre of current research. This section positions the work carried out during this semester in relation to recent advances in four key areas: solar potential in Norway, smart energy management, east-west-facing vertical panels, and PV output modelling. The sources used are available in *§VI BIBLIOGRAPHY*.

1. Solar energy in Norway

Norway, often perceived as a country where hydropower dominates, has untapped solar potential, particularly for decentralised applications such as remote cabins. A recent study by Mæhlum et al. (2024) [1] assesses the technical potential of solar power on Norwegian buildings, highlighting that although annual sunshine levels are lower than in Southern Europe (around 800–1,000 kWh/m²/year compared with 1,500–2,000 kWh/m²/year in Spain or Italy), several factors compensate for this limitation:

- Cool temperatures: PV panels have a loss of 0.3% to 0.5% per °C above 25 °C, meaning that Norwegian temperatures, often < 20 °C improve the efficiency of the modules [1].
- High albedo: Snow and reflective surfaces increase the output of bifacial panels, particularly in winter.

The main challenge for the PV sector in Norway is the seasonal energy demand. Consumption, particularly for heating, is at its highest in winter, a period when sunshine is at its lowest, necessitating optimised storage and management strategies. The next graph displays the significant variation of the potential solar energy during a year in Norway (top of the next page):

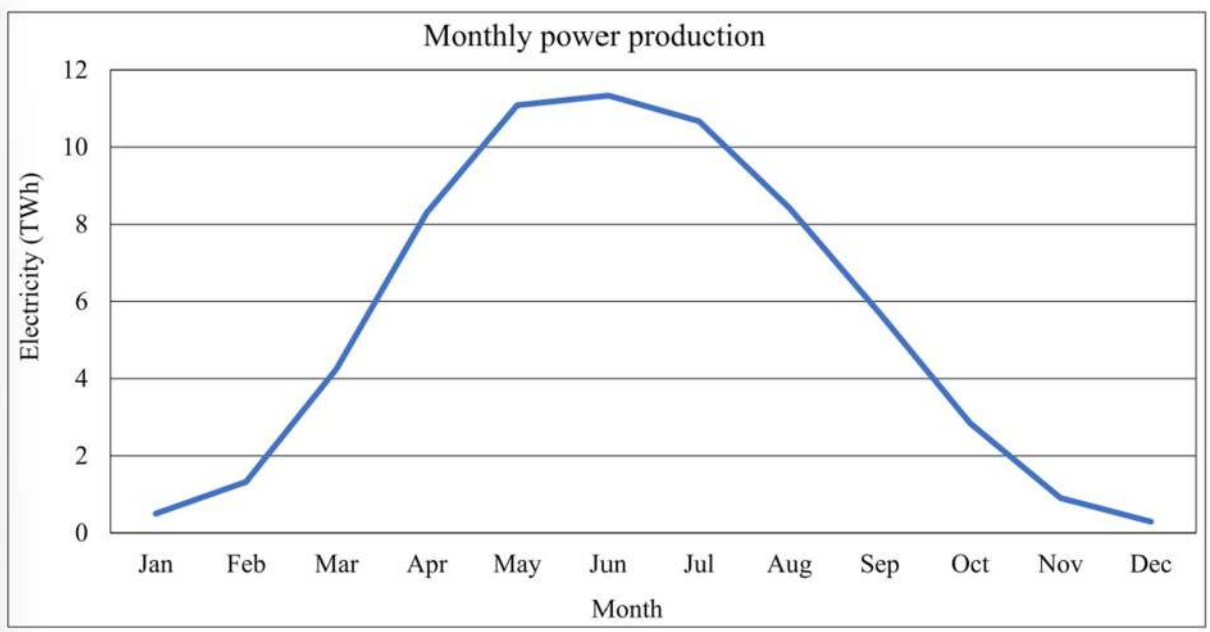


Figure 3: The potential monthly power production of solar energy in Norway [1]

Data from "Our World in Data" confirms that solar power still accounts for just 0.1% of Norway's energy mix (2023), but its potential for self-consumption for isolated buildings is growing rapidly [1].

2. Solar energy production management

Energy management in off-grid PV systems is critical to preventing imbalances between generation and consumption. Li et al. (2020) [2] analyse optimal strategies for systems integrating solar power, energy storage and electric vehicles (V2G[2]), highlighting three major challenges:

- Flow optimisation: Without smart management, up to 20% of the energy generated can be lost due to battery overcharging or uncontrolled dumping [2].
- Integration of controllable loads: Systems such as SolarLab, equipped with charging points and heating systems, require hysteresis logic to prevent frequent switching which causes equipment wear and tear.
- Automation: Manual solutions are not very scalable and are prone to human error [2].

A review by Khan et al. (2023) [3] compares management strategies for hybrid systems, classifying the approaches into three categories:

- Rule-based strategies: Simple system but not very adaptive. For example, fixed State of Charge (SOC) thresholds.
- Optimised strategies: Accurate because they are based on optimal mathematical calculations, but too complex to implement in real time on integrated systems. For example, linear programming.
- Hybrid strategies: combines simplicity and efficiency using part of both strategies.

[2] V2G: Vehicle-to-Grid, bidirectional charging technology that allows electric vehicles to not only draw power from the grid but also send stored battery energy back to it.

3. East-west vertical solar panels

Vertical, east-west facing bifacial panels represent an innovative solution for high-latitude regions, such as Norway. Pavlovic et al. (2023) [4] studied the effectiveness of installing them, demonstrating that this configuration offers:

- A more evenly distributed output throughout the day with two peaks, one in the morning and one in the late afternoon, unlike tilted panels which have a single peak at midday.
- Greater resistance to shading by reflecting light from the ground or buildings on both vertical surfaces, partially offsetting losses due to cast shadows.
- Simplified architectural integration: Ideal for façades or flat roofs, without the need for complex structures.

A complementary technical assessment by Al-Nuaimi et al. (2022) [5] compares the performance of east-west systems with a conventional installation:

Criteria	Tilted panels (30°)	Vertical panels (east-west)
Annual generation	+10%	Reference
Winter generation	Reference	+40%
Wind resistance	Medium	High
Maintenance	Moderate	Reduced (less accumulated dirt or snow)

Table 1: Comparison of performance between conventional tilted and vertical PVs [5]

4. Correlation between irradiation and PV output

Modelling PV output as a function of irradiance is a key area for forecasting energy availability. López et al. (2025) [6] analyse the combined impact of irradiance and temperature on output, using statistical and machine learning approaches. Their results show that:

- Linear models are simple but limited. They do not capture the non-linearities associated with:
  - The saturation of output at high irradiance.
  - Variations in panel efficiency as a function of temperature and irradiance [6].
- Polynomial models improve accuracy by incorporating these non-linearities. For example, adding a quadratic term makes it possible to:
  - Reduce the mean error.
  - Improve the general correlation.
  - Have a model better at handling extremes.
- Other more advanced systems such as physical models or models based on machine learning are resource-intensive and more complicated to set up.

III. ORGANISATION

A. Tasks management

During my MSRI, I did not have one main task with a lot of sub-tasks requiring a planning. My project was rather to manage the SolarLab and improve it. I was working on a lot of small tasks and missions at the same time. At the beginning of my MSRI, my tutor gave me some suggestions of missions to do within the SolarLab. Quickly, I was able to autonomously propose my ideas for the next tasks. In addition, my tutor continued to give me a few new subjects to work on. I mostly worked autonomously and was responsible for finding the help I needed from the engineers and technicians at the school when required.

To keep track of the advancement of each task, I created a shared OneNote file with a list of all the tasks I was planning to do. Inside this list I was using a colour code to monitor the priorities of the task. This method allowed me to work effectively and keep a track of what I was doing without losing too much time on the following process.

B. Progress report

To see my progress and ask all my questions easily, my tutor and I decided to have a formal meeting once a week or every two weeks. This enabled us to establish a strong working environment and complete all the informal exchanges we had in the corridor or via Teams.



To keep track of the work carried out at the SolarLab during the years, there is a shared SharePoint site. This site is used to store all documents and files relating to the SolarLab. Within this site, I had my own personal folder, which made it easy to see both work in progress and work that had already been completed.

IV. WORKS AND RESULTS

A. Increased maximum production power

1. Context: initial configuration and issue

The 6 monocrystalline solar panels are connected to inverter 1. There are 2 strings in parallel, each consisting of 3 panels in series. The 6 polycrystalline panels are arranged in the same configuration and connected to inverter 2.

In theory on a nice day, the production curve (DC input of inverter) should resemble a bell curve. However, it was observed that the curve levelled off, creating a flat plateau, at a maximum of 950–1,000W, rather than gradually increasing to reach 1,600W.

The graph below illustrates this trend:

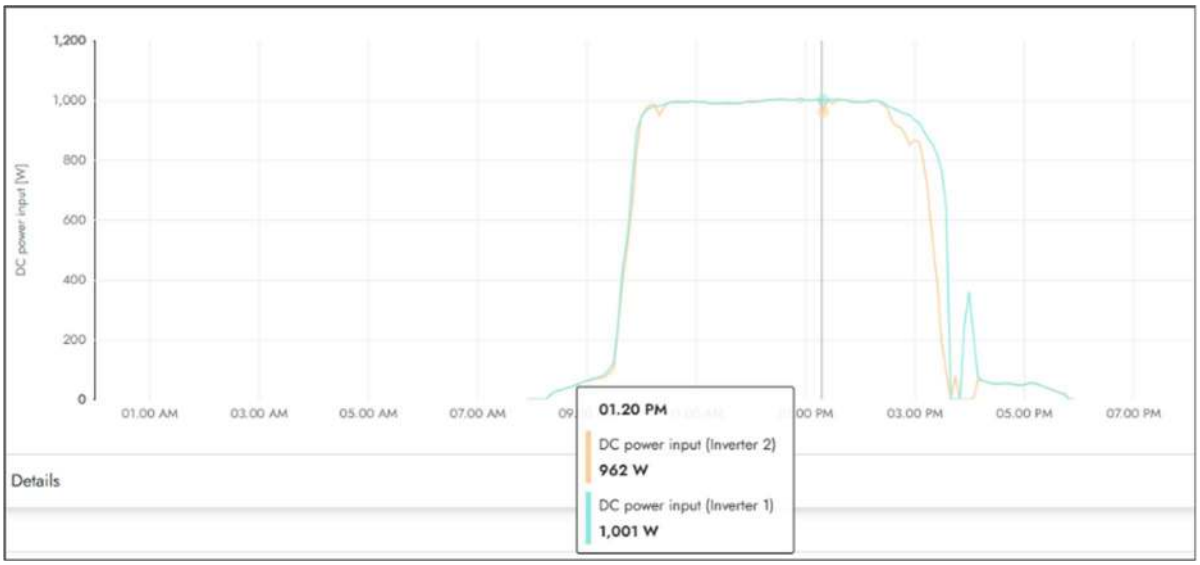


Figure 4: Graph of DC input of inverter 1 and 2 during a sun-filled day at the beginning

In this situation, a lot of energy is lost by not being produced. Therefore, it was important to investigate the cause.

2. Investigation of the cause

My first action for this subject was to look at the datasheet of SolarLab devices involved: Sunny Boy inverter, monocrystalline PV and polycrystalline PV. By doing so, I was able to pick up the rated voltage and current of these devices. For the sake of traceability, I have summarised all my work on this subject in a note. Extracts from this note are available in appendix 2.

By doing some calculations, I was able to determine voltage and current generated by the PV strings to the inverter with the initial configuration. I discovered that, at the rated power production, the PV strings connected to each inverter were generated around 17,5A of current. It was well above the maximum current of the inverters (10A). To solve this problem, I proposed to change the 2 parallel strings of 3 panel each in serial configurations to a full serial

configuration. By doing so, the maximum current is divided by 2 (and the voltage multiplied by 2, but it stayed within the accepted range of the invertor).

In broad terms, the configuration evolves as follows:

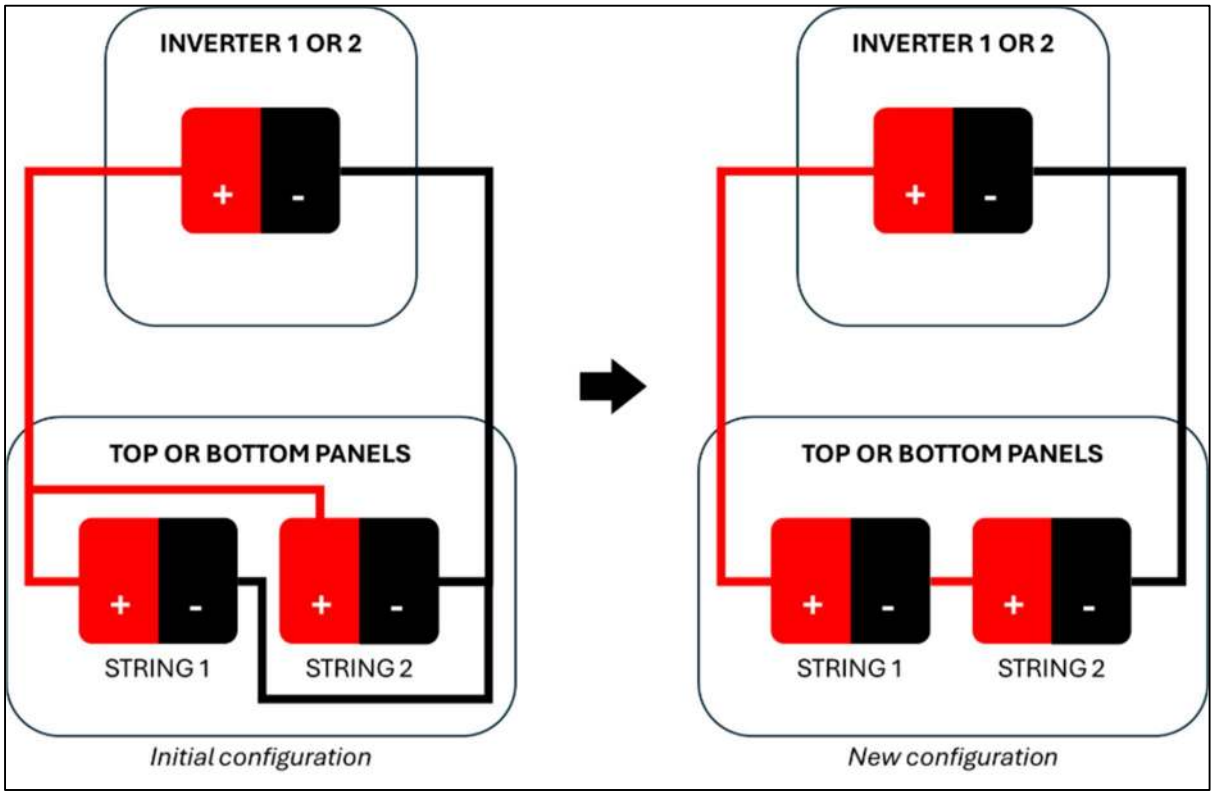


Figure 5: Diagram of the two different PV string wiring configurations

The proposed configuration increases maximum production power (by approximately 60-70%) and ensures operation within the MPP voltage range. The detailed calculations are available in the appendix 2.

3. Implementing the modifications

To implement the change, I wrote a procedure to implement the modification. It includes the following elements:

- Equipment required,
- Safety protocol to use,
- Steps of the wiring modification, with a diagram,
- Settings modifications of the Tigo production optimiser,
- Control checks to performs,

Thanks to the work of a previous ICAM student, the wiring modification was quite easy. Indeed, he built a box with a quick-connect fitting that allows you to easily switch between configurations for the PV strings. I only had to prepare new cable with MC4 connector and then change the wiring configuration.

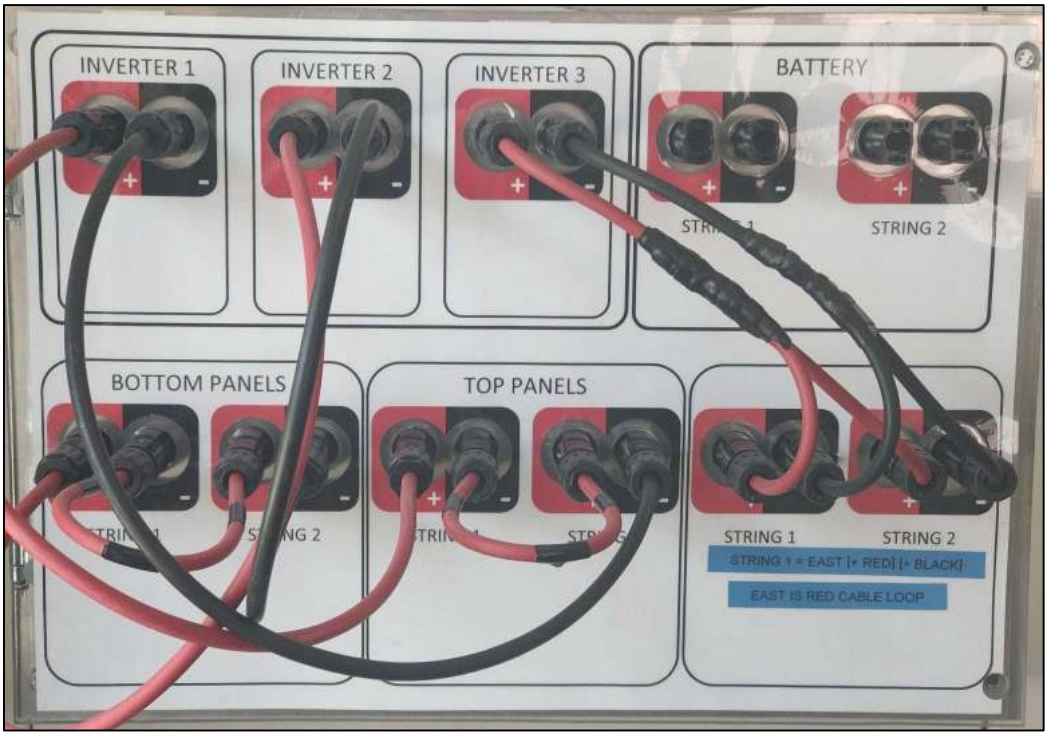


Figure 6: Box with MC4 connector after wiring modifications

4. Result

With the modifications being done to the two-inverter wiring, the maximum current was divided by two and the voltage multiplied by 2 as expected. Consequently, the DC input curve for inverters 1 and 2 resembles a bell curve, with a maximum power output of between 1500 and 1600W.

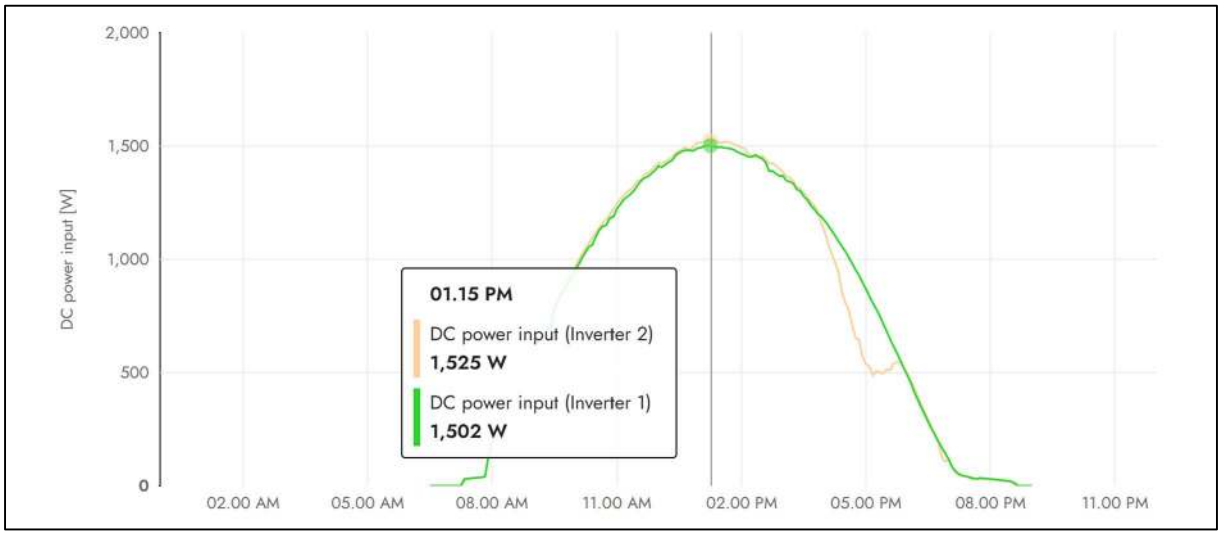


Figure 7: Graph of DC input of inverter 1 and 2 during a sun-filled day after modifications

Modifying the panel wiring makes it possible to optimise output and increase the maximum power output from 2000W to 3000W in the main PV generation section.

B. Correlation between weather, radiation and production

1. Context: reason of building a prediction model

During my first few weeks of my semester, I noticed that SolarLab was operating without any model anticipating the future. Since one of the long-term goals of SolarLab is to operate completely autonomously, I decided to look at the anticipating production in more depth. Indeed, creating a first model forecasting the production based on the weather will allow to adjust the use of the energy of SolarLab.

2. Principle of the model

Before starting working on the mathematical part of the model forecasting the production, I took the time to create a chart summarising all the data input needed, all the step and the main logic between each step. This diagram is available in the appendix 3.

The main data input that will be needed for that worked are:

- GPS coordinates of the PV installation.
- Date and time of the study.
- Information related to the PV implementation.
- Ground albedo.



The solar radiation captured by a PV is divided into 3 components: direct radiation, which travels in a straight line from the sun and generates the maximum amount of energy; diffuse radiation, which is light spread by the atmosphere and clouds, and finally, albedo, which refers to the proportion of light reflected by the surrounding ground surface.

The main steps of the model I built are summarised in the table in the appendix 4; calculations are done for each data point request by the user (each data point calculated equal one line within the Excel file). Description of how to operate the model file is done in the *§IV.B.3 Operating principle of the model.*

10,999	**9**	**26**
data points can be calculated simultaneously	**study locations are pre-configured in the file**	**data series can be viewed in the results graph**

Figure 8: Key figures of the model

3. Operating principle of the model

The Excel file where I built my model, is quite easy to operate. Indeed, the user only needs to enter the study parameter on “General parameters” spreadsheet such as the range of the study, the localisation and the characteristics of the PV installation. Most of the fields that are required are built with some drop-down list. After that the user has the possibility to add manually some irradiance data or clearness index.

Then, the model automatically calculated the forecast PV production. All intermediate calculation is available to the user. To visualise the result, the user can select the data he wants to be display by ticking some checkboxes. The graph visualisation built into the tool updates automatically.

Here is the list of steps the user must follow to carry out a study:

Step	Spreadsheets	Comments
1	General parameter	Enter the study parameters in the green cells. The other cells are calculated automatically.
1bis	Drop-down list settings	If necessary, you can add further parameters from the drop-down list in the blue cells.
1ter	External data input	This spreadsheet allows you to enter data manually (under the blue cells). The tool automatically uses this data when you enter it.
-	Weather data	This Excel spreadsheet is used to import weather data, this feature is not yet available. No action is required.
-	Sun position	This Excel spreadsheet is used for calculations and display of results. No action is required.
-	Radiations	This Excel spreadsheet is used for calculations and display of results. No action is required.
-	Production	This Excel spreadsheet is used for calculations and display of results. No action is required.
2	Graphs parameters	Tick the checkboxes of the data to be displayed on the graph.
3	Graphs display	The selected data will be displayed on the graph.
4	Correlation	Copy/paste theoretical and measured data into the cells below the blue ones. There are indicators to help analyse the results.
4bis	Correlation data history	History of data used for correlation.

Figure 9: Operating steps of the model

4. Model evaluation

To assess the accuracy of the model I have created, I analysed the correlation of my model between the predicted output (based on the pyranometer data) and the actual output recorded at SolarLab. Here is the list of criteria I used to test the accuracy of my model:

- Correlation coefficient (|R|): measures how strongly the predicted solar output tracks the trend of actual production, regardless of scale.
- Share of variance (R^2): tells how accurate the actual values are by measuring the proportion of real-world solar fluctuations the model successfully accounts for; it factors in both the pattern and the scale.
- Average error: shows the overall systematic bias of the model, indicating whether it tends to over-predict or under-predict solar energy generation on average.
- Root Mean Square Error (RMSE): measures the average size of the forecasting errors in actual energy units (e.g., W), penalising larger errors more heavily to highlight major misses.
- Relative RMSE: expresses the RMSE as a percentage of average production, allowing to understand the scale of the error relative to total solar output.

To visualise easily the difference between the forecast made by the model and real data production, hereunder is a correlation graph example for 11th of May 2026. The morning was sunny and the afternoon a bit cloudy.

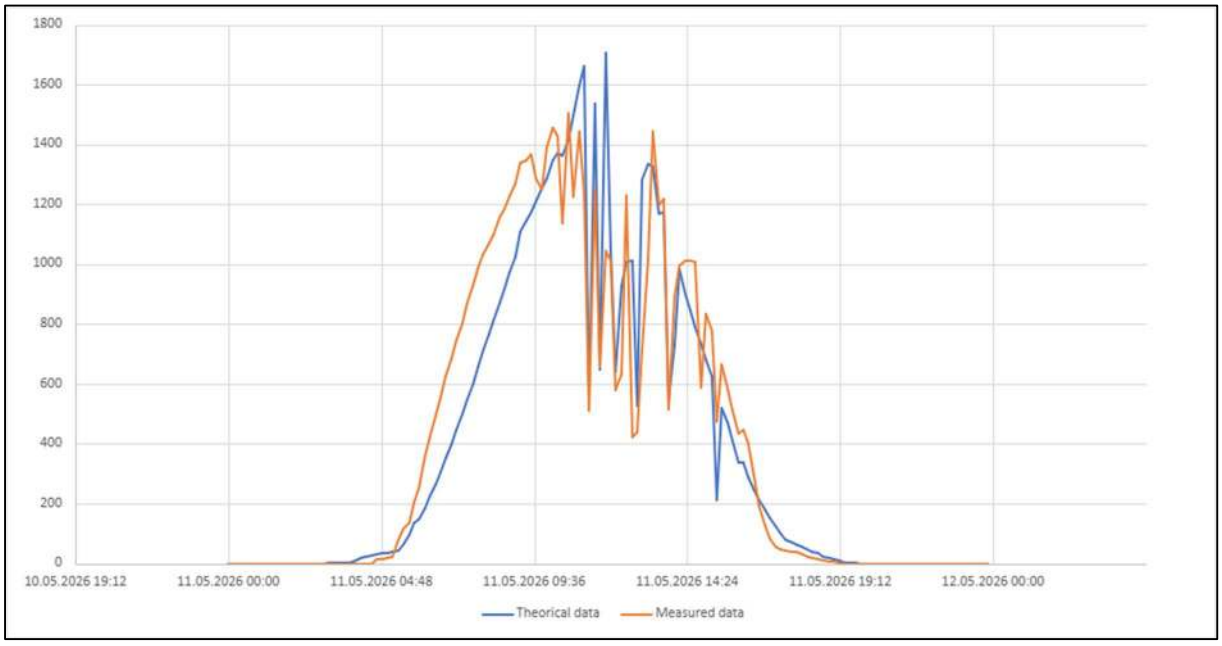


Figure 10: Graph comparing theoretical and measured output of monocrystalline PV (W)

After evaluating the model on multiple production days, here is the model accuracy evaluation (excluding extreme low light day where it is hard to do correlation and not the most significant days):

Criteria			Indications				
Correlation coefficient (|R|):	97,16 %		Good >	85 %	> Fair >	70 %	> Poor
Share of variance (R^2):	94,45 %			75 %		50 %	
Average error:	-9,36 %		Good <	3 %	< Fair <	10 %	< Poor
RMSE:	136,33W			N/A		N/A	
Relative RMSE:	33,75 %			10 %		20 %	

Table 2: Model accuracy evaluation

The first version model performs well over the course of the day as a whole, but poorly when it comes to rapid fluctuations in irradiance.

This problem in dealing with rapid fluctuations is mainly due to the way in which the irradiance measurements are taken by the pyranometer at the input and the power output measurements used for the comparison. Indeed, the pyranometer take measure every 10 minutes but not on round time (for example at 09:07AM and then at 09:17AM) but the power production measurement made is done at round time (for example at 09:10AM and then at 09:20AM). This small-time difference leads to a high relative RMSE in the model's evaluation.

Also, the model tends to underestimate the production when the sun is full, and then to overestimate the production when the light is low. This first version of my model uses a purely linear relationship between the total irradiance received by the PV panels and the power output. In fact, the efficiency of photovoltaic panels is not fixed. It decreases in low-light and increases in high-light conditions. This mainly impact the average error of the model.

5. Improvements and future use

The first model worked quite well, but there is always room for improvement. That is why I decided to work on some model upgrade. To compensate for the underestimating or overestimating production on low or high light situation, I transformed my model from a linear relationship between the total irradiance received by the PV panels and the power output to a 2nd degree polynomial model.

Linear model:

$$power\ output = total\ radiation\ received\ by\ PV * PV\ area * efficiency$$

2nd degree polynomial model:

$$power\ output = (a * total\ radiation\ received\ by\ PV^2 + b * total\ radiation\ received\ by\ PV) * PV\ area * efficiency$$

To determine the optimal values of parameters a and b, I used the Excel Solver to minimise the relative RMSE. The final values are: $a = -0.0003$ and $b = 1.2866$.

With this improve model, here is the new accuracy evaluation (excluding extreme low light day where it is hard to do correlation and not the most significant days):

- Correlation coefficient (|R|): 97,47 % (+ 0,31 %)
- Share of variance (R^2): 95,01 % (+ 0,56 %)
- Average error: -1,87 % (- 7,49 %)
- RMSE: 117,65W (- 18,68W)
- Relative RMSE: 28,63 % (- 5,12 %)

Here is an example data visualisation from multiple day comparing actual output, the V1 model (linear) and the V2 model (second-degree polynomial) for monocrystalline panels (inverter 1):

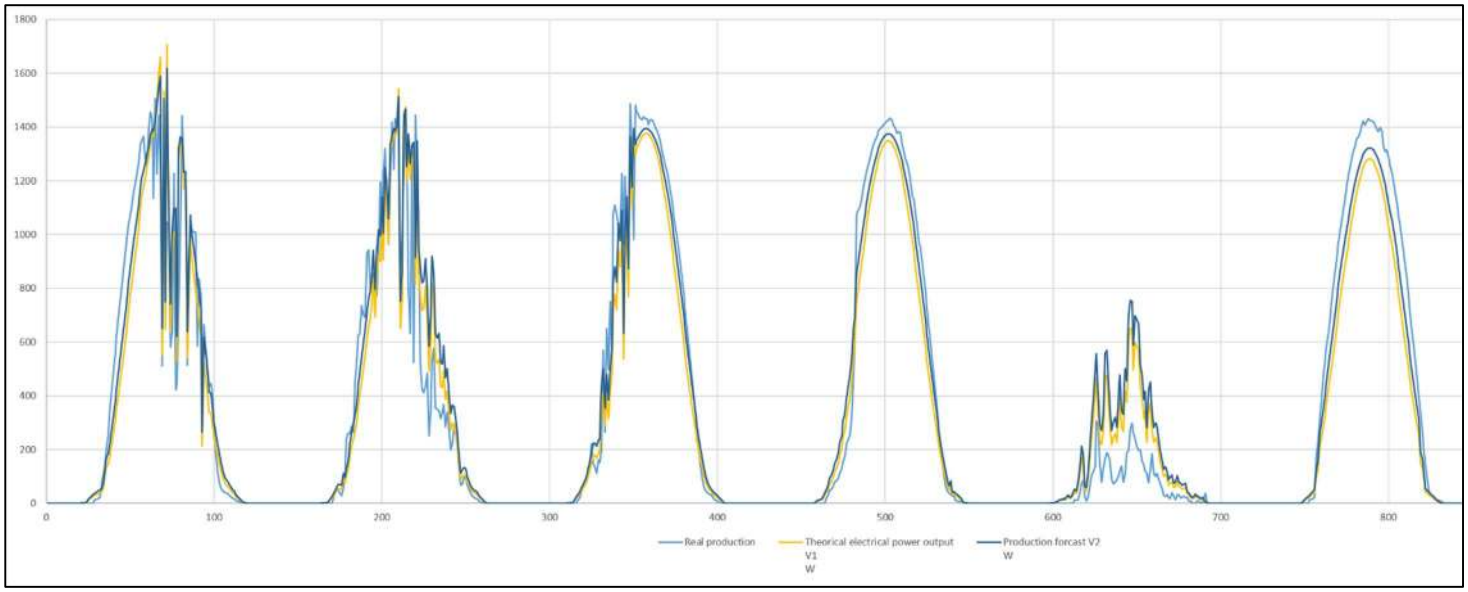


Figure 11: Graph comparing actual output, V1 and V2 model for monocrystalline PV

This V2 model improves a bit the system. However, the model still suffers, to a lesser extent, the same issue as the V1 model. To improve even more the model, a possibility would be to use a model with thresholds to best reflect the actual conditions of the panels.

Thanks to this work, the SolarLab now has a way to forecast the production. Using the same logic, a future automation is possible. This will allow the SolarLab to manage load more effectively. You can see the detail of that in *§IV.C.4 Next evolution*.

C. Automated energy management system

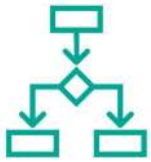
1. Context: need for an automated system

The SolarLab is an off-grid passive house. To work properly, it needs to have a system that manages automatically the usage of the energy. Indeed, when there is no or low sun, all the energy needs to be used only for SolarLab normal operating (heating, irradiation measurement, website actualisation, …). When there is a lot of sun, the goal is to dump the energy produced to keep the battery state of charge (SOC) bellow 95% to ensure that solar production is not interrupted and collect production data that corresponds to the sunlight.

Students before installed 3 pilotable load that were controlled remotely by some Shelly switches[3]. This load corresponds to 1 Amina car charger, 1 reinforced socket used to charge a car as well and 1 electrical heater. In the rest of the document, these loads will be referred respectively as CC1, CC2 and HEATER. When I first arrived, these loads had to be manually switched ON/OFF from the Shelly app based on the production. This mode of operation cannot be kept forever; it must be automated.

2. Operation principle

The last student started working on this subject and proposed some idea to automate the system. Based on the SOC, an energy level (EL) is established and then activates the load, all done thanks to a Python programme on the Rasbery Pi. The energy level allows to simplify programming and implement a hysteresis logic.



In the algorithm diagram, CC2 is deactivated at EL3. The system is designed this way for safety reasons, the maximum current for the three loads is 25A (protection). If two cars (CC1 and CC2) are charging at 10A each and the HEATER is switch ON (9A), there would be a risk of overcurrent (29A > 25A). To prevent a circuit breaker from tripping, the three loads are never activated at the same time.

Thanks to the preparation work done by the previous student, I created this algorithm diagram representing the automatic mode of the programme (top of the next page):

[3] Shelly switch: it is an intelligent switching module that operates via a WI-FI network replacing traditional mechanical switches.

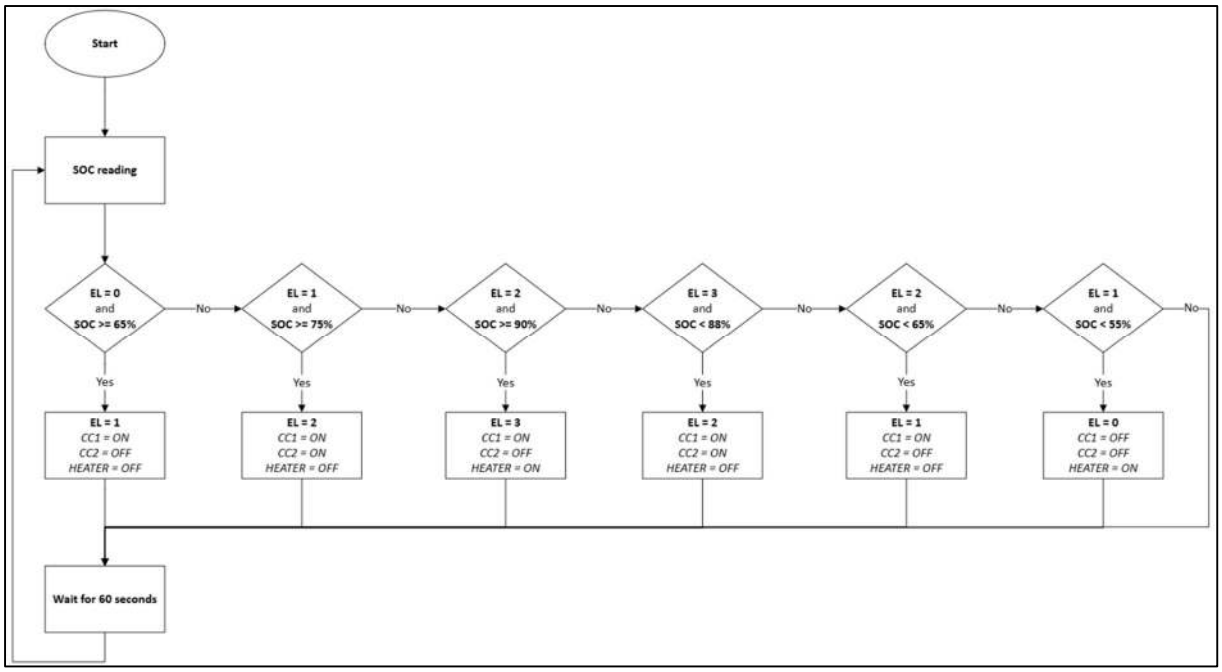


Figure 12: Algorithm diagram of the automated energy management system

3. Development of the Python programme

SolarLab's Raspberry Pi was already running some Python script for other code, using Python for this new automation was therefore quite logic. Based on the preparatory work carried out by the previous student, I developed a fully working code. The final code is structured in many different parts. The main parts are as follows:

- Import of the modules or libraries that are used in the code.
- Definition of the global constants (SOC threshold, IP addresses, credentials, …).
- Definition and initialisation of the global variables.
- Configuration of the external devices:
    - Sunny Island used to read the SOC of the battery (via Modbus TCP request).
    - Connexion to the Shelly devices (thanks to the Shelly HTTP REST API).
- Construction and update of the Graphical User Interfaces (GUIs) tanks to the Tkinter library.
- Main loop, running every 60 seconds:
    - SOC and Shelly state reading.
    - Daily reset of the Energy Level and Auto/Manu mode.
    - Management of the availability and activation of the Manu Mode.
    - Auto mode logic.
    - Shelly state update.
- Programme entry point:
    - Initialisation of the communication with the external devices.
    - Start of the GUIs in the main thread.
    - Start of the main loop in a separated thread.

A thread is a unit of execution that forms part of a larger process. It enables a programme to execute multiple operations concurrently within the same process. In the context of Python, a thread is particularly useful for tasks that need to be executed independently of one another, such as a GUI from a background task. In this case it allowed to avoid mutual interference between the parts of the code.

To build this system I worked by building each module independently before implementing them inside the main code. I started by the most important one and then integrated them to each other when they were working, it was a block-by-block development. This allowed me to quickly create a first version of the programme that did not have all the functionalities but that was allowing me to test and improve it very quickly.

The GUIs I built is as follows:

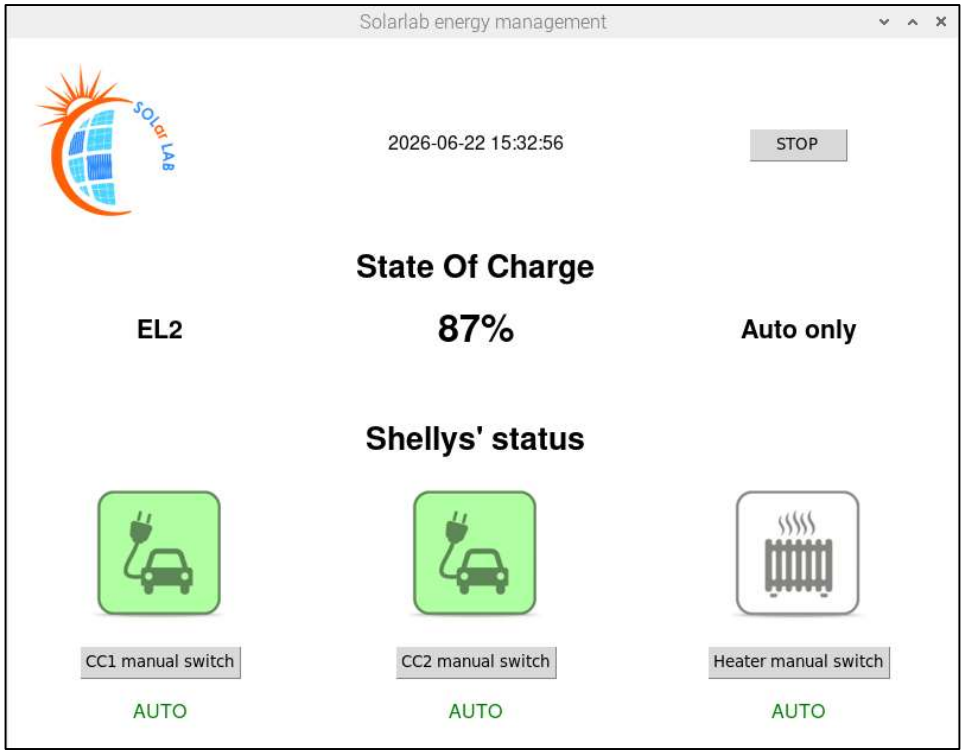


Figure 13: Energy management GUIs

The user interface displays information such as a timestamp of the last time the main loop ran, battery SOC, availability of the manual mode and Shelly state. On the same interface, I've also put switches that allow the user to manually activate the load. The code integrated some safety limitations to protect the charge of the battery. In addition, I built my code in a way that it can detect the manual activations made from the Shelly app.

Whilst working on this, I also took the opportunity to make the Shelly app settings clearer and more organised. The Shelly app now looks like this (top of the next page):

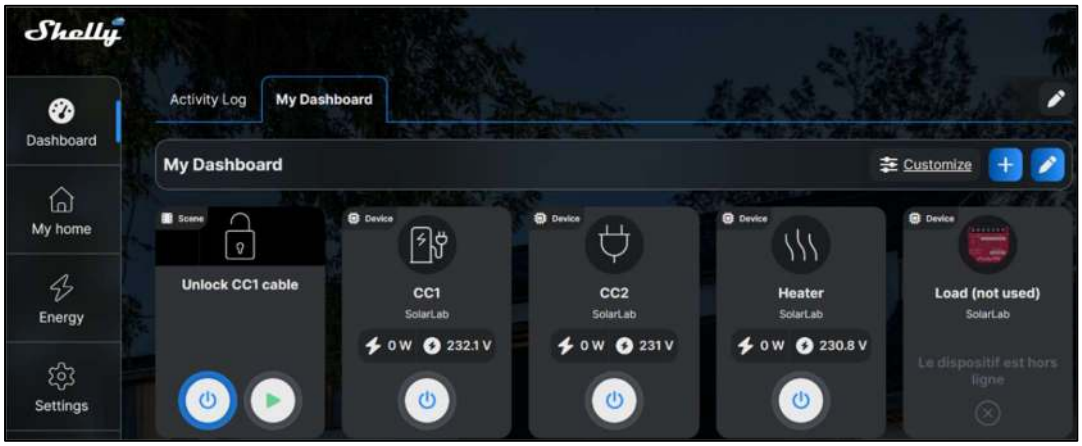


Figure 14: Shelly web app interface after the modifications

With all these functionalities, the final code is quite important, with around 600 lines of code, you can find an extract of some parts of the code in appendix 5. My automated energy management system is now running 24/7 allowing SolarLab to switch load ON and OFF automatically.

4. Next evolution

The first version of my automation works quite well but has some limitation. Indeed, the current system does not include any prevision of production or of consumption. This can lead to situations where the system allows a car to charge and use the energy of the SolarLab even if there is bad weather coming and the SolarLab should ideally keep is energy. The same situation can also happen the other way around. An evolution of the current system would be to integrate variable SOC thresholds based on production forecast realised thanks to an automation of my prevision model and thanks to a weather forecast API. The variable SOC thresholds could be calculated once a day, for example at night. Here is a summary of the logic that can be programmed to make that happen:



Figure 15: Production forecast integration logic

As I did not have time to carry out this evolution during my semester, I wrote a note for the next student, explaining the logic and the tools and API available to carry out the evolution.

D. SolarLab crash management

1. Context: frequent crashes

From time to time, the SolarLab electrical system goes into a blackout. The root cause of this issued is not yet totally identified. The probable causes of this system crash are detailed in *§IV.E.2 Theories crashes cause*

During those crashes (or blackout), all the SolarLab systems go down: measurement of the irradiance with the pyranometer, update of the website, PV production, HAVC and all the other appliances are stopped. When it happens, the SolarLab loses important data, and the SolarLab state cannot be monitored from away. One of my tasks was to make sure that the SolarLab was always up and reboot it when it crashed. Doing so manually had some constraints: a significant delay between the crash and the restart, and a need to check regularly and manually to see if the lab has crashed. Regarding this problem that happened often, my work was to find a solution to keep measurement being made by the pyranometer and data being publish on the website even in a case of a blackout and then to create a system that detects crashes and starts the reboot procedure on its own.

2. Backup power supply

To keep pyranometer data measurement and website update running, a few low consumption devices within SolarLab need to keep a power supply in any cases: the router, the switches and the Raspberry Pi. The goal is to have a backup system capable of running around 12 hours on its own to give plenty of time to reboot SolarLab.

The solution found with the SolarLab team is to install a UPS[4] inverter with a backup battery. I was in charge of dimensioning the system and finding the right component. To do that, I analysed the power consumption of all these devices.

	A	B	C	D	E
1		Estimated average consumption (W)	Max power consumption (W)	Max power consumption with RasberryPi 5 (W)	Comments
2	RaspberryPi power supply (5V1, 2.5A)	6,00	12,75	25,00	*Power consumption of the RaspberryPi and peripherals (excluding the screen).*
3	*RasberryPi 3 model b rev 1.2*				
4	*Smart Setup Hub PMU485*				*25W = anticipation of a possible use of a RaspberryPi 5 with a 5V/5A power supply (maximum power consumption of 25W)*
5	*Pyranometer SMP11*				
6	Router TP-Link Archer MR200	5,00	12,00	12,00	*External power supply: 12V/1A*
7	Switch NETGEAR GS108	3,00	4,92	4,92	*4,92W in worst case, all ports used, line-rate traffic, max PoE*
8	Switch D-LINK	3,00	5	5,00	*Input 5V/1A*
9	TOTAL	**17,00**	**34,67**	46,92	
10	VA With a power factor of 0.8	21,25	43,34	58,65	
11	Time spent off the grid (h)	12,0	5,9	4,3	
12	Usable battery capacity (Wh)	204,00			
13	Usable battery capacity (VAh)	255,00			
14	Usable battery capacity 12V (Ah)	**17**			
15	If sealed lead-acid battery, capacity x2	**34**			

Figure 16: Power consumption of the communication devices

The power consumption is very low and, to keep the system alive for around 12 hours, we need a 12V / 34Ah sealed lead-acid battery. I then looked for a UPS inverter model that meets

[4] UPS: Uninterruptible Power Supply

this requirement. To keep the price low and install a system that is easy to evolve, I decided to use a UPS inverter that doesn't have an internal battery. The maximum power that needs to be supplied is very low, less than 20W, so it's not an important criterion for the UPS inverter selection.

The chosen model is the next on from CyberPower:



Figure 17: UPS inverter model chosen

I was also missioned to do the installation of the new devices inside the SolarLab. I installed wood support (that I also painted in white to make it look nice) to not damage the walls. The battery inverter is currently temporary installed inside a container. The next student will be in charge to build a shelf and a box for that battery.

I also took the occasion to implement new devices to upgrade the implementation of the router and the switches. I used a plastic box fixed to the wall to group communications devices together with a clean cable management. This implementation was initially designed as a 3D model for validation purposes. To fix these devices to the back of the box and allow easy future modification, I used a system of plates with holes and custom-made supports that I designed and 3D printed. Details about the backplates and supports I made are available in appendix 6.

On the next pictures you can see the situation before and after the installation of the devices:



Figure 18: Before and after the UPS inverter installation



Figure 19: Before, during and after the switches / router new implementation

All the devices connected to the UPS inverter can be easily seen on the appendix 1.

At this stage I also set up an automatic message to notify the team when SolarLab crashed. I took advantage of the SMA system that is used to oversee SolarLab energy production to set up this notification system. The only issue with that solution was that the notification can take up to 2 hours to be delivered to the users.

3. SolarLab remote reboot

The action needed to perform a reboot of the SolarLab is quite easy: a double press of the battery button. With that in mind, the idea of creating a system to press the battery button was born. To perform that, Richard J. Grant had the idea to use some SwitchBot devices. This company proposes devices that can be used as small fingers to press buttons remotely.



Figure 20: SwitchBot devices use for auto reboot: Plug Mini, Bot and Hub Mini

Here is the list of all the SwitchBot devices bought.

- Plug Mini, used to track the consumption of the inverter and signal crash with a quick notification to the user when offline.
- Bot, to press remotely the battery button.
- Multiple thermometers to measure new data, for more information see *§IV.G.3. Temperature sensor implementation.*
- Hub Mini, used to connect the other SwitchBot devices to the cloud.

I have then set up the device on the SwitchBot app, see app screenshot on the right. By then I had some issues with the Hub Mini. The hub refused to work as a hub, and I had to contact the SwitchBot support.

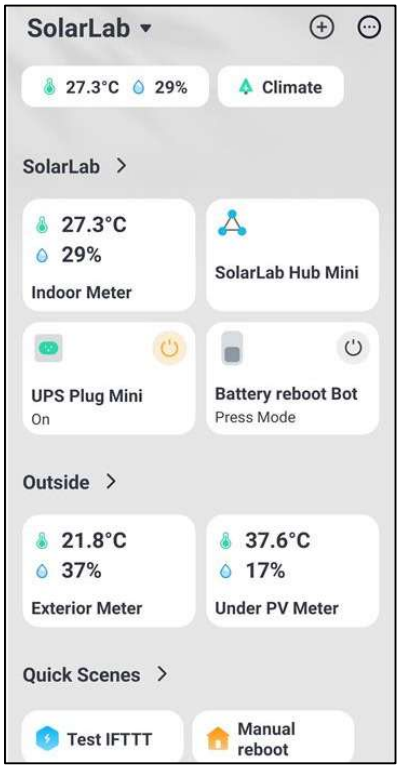


Figure 21: SwitchBot app

On the same app, I also created a “Quick Scenes” allowing the user to perform a manual remote reboot. The logic is as follow: when the user presses the Manual reboot, a notification is sent to the user indicating that a reboot will be performed in 30 seconds. This allows the user to cancel the reboot if it was a miss click. Then, the “battery reboot bot” is activated twice in a row with a ten second gap between the two presses.

The normal way to implement the Bot, is to stick it next to the button that required a press. In our case, we did not want to stick something inside the battery cabinet. To get around this restriction, I designed an arm which fits directly onto the doors and then 3D-printed it. With that system, no modification was required on the battery cabinet.

Here is the arm installed:

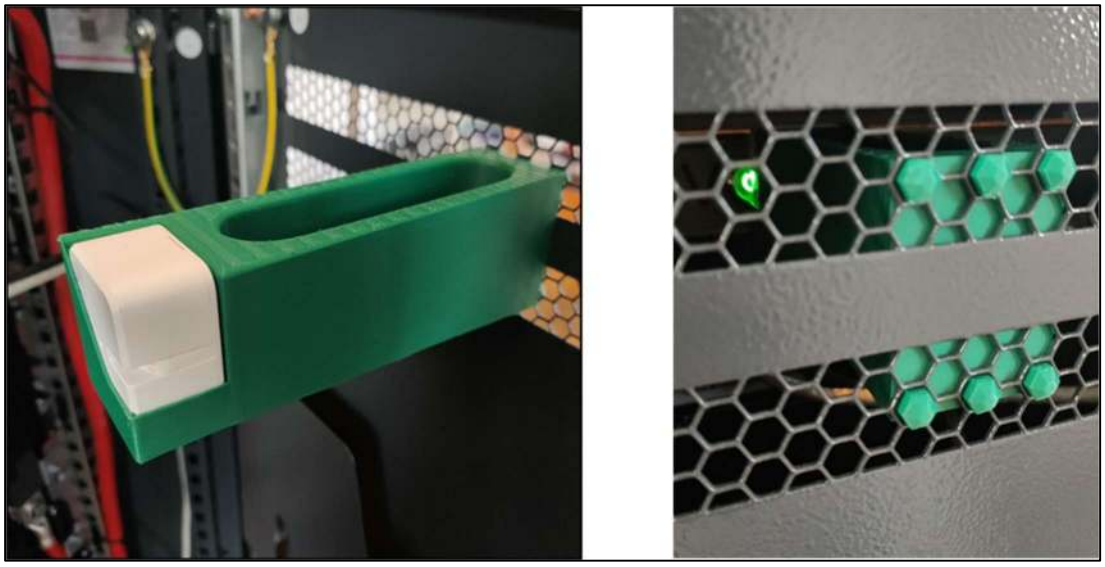


Figure 22: Arm supporting the Bot, mounted on the battery cabinet

4. SolarLab auto reboot

To upgrade this manual remote reboot system, I decided to do some automation in python thanks to the Rasbery Pi. To do so I created a state machine.



A state machine is a modelling technique that allows a system or object to exist in a single specific state at any one time, and to change state (also call a transition) only in response to specific events or conditions. In my case, due to the Python version installed being below 3.10, I did not use the "statemachine" library and the match state structure. I recreated the state machine operation with a "if" structure.

The different states of my program are as follows:

- Routine check: everything works as usual, a "ping" to check if the Plug Mini is still connected to the Wi-Fi network is sent every 5 minutes.
- Crash confirmation: when the Plug Mini does not answer to the "Ping", a second control "Ping" is sent 60 seconds later.
- Reboot procedure: if the plug does not answer the second "Ping", the reboot procedure is performed.
- Reboot check: after the reboot is done, the system checks if the Plug Mini is still unreachable. If everything is normal again, the programme goes back to the routine check. If not, the system can in some cases try another time the reboot procedure.
- Hold: If the maximum reboot procedure is reached, the system goes on hold, waiting 1 hour before performing another confirmation "Ping" or the intervention of the user.

The full algorithm diagram for this state machine is available in appendix 7. The final code is structured in many different parts.

The main parts of the code are as follows:

- Import of the modules or libraries that are used in the code.
- Definition of the global constants (delay, number of auto reboot allowed, IP addresses, Token and secret key for the API…).
- Configuration of the external devices:
  - Ping of the Plug Mini thanks to it IP address.
  - Control of the Bot thanks to the SwitchBot API and BLE MAC[5] address
- Construction and update of GUIs tanks to the Tkinter library.
- Main loop, state machine running:
  - An if-statement structure simulating the state machine
- Programme entry point:
  - Initialisation of the communication with the external devices.
  - Start of the GUIs in the main thread.
  - Start of the state machine in separated thread.

[5] BLE MAC: Bluetooth Low Energy Media Access Control

To help the user monitor the situation I also developed a small GUI. It is as follows:

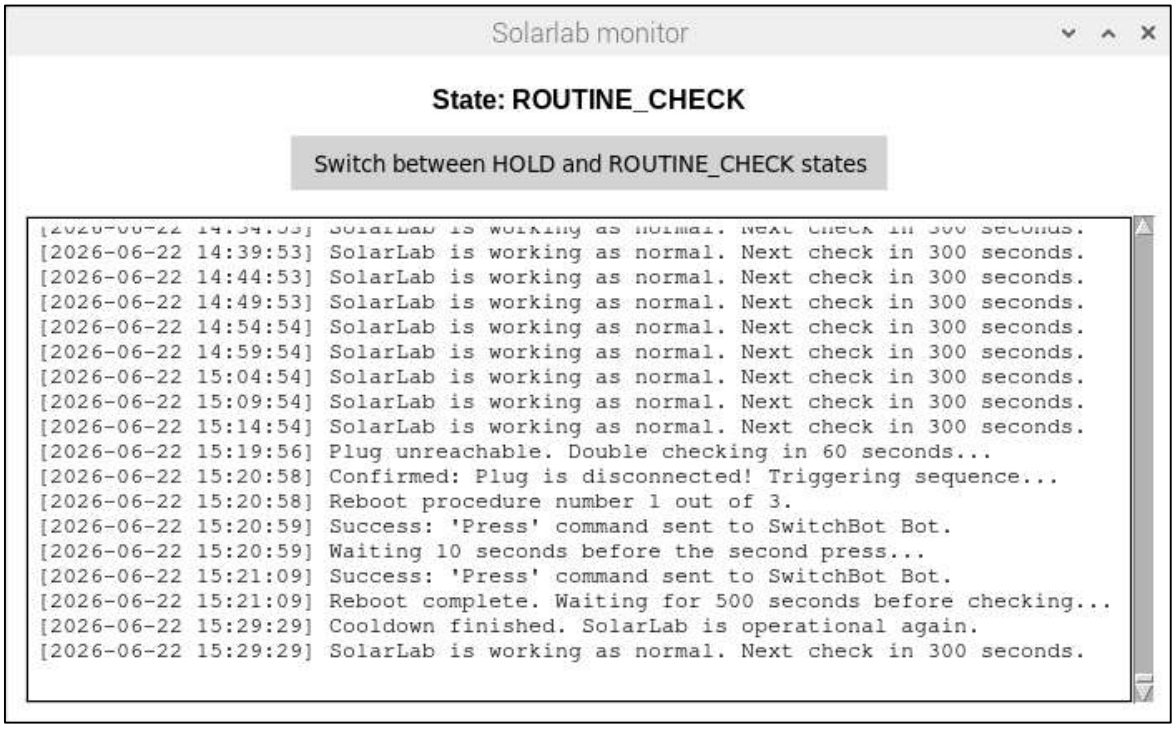


Figure 23: Crash monitor GUIs

This interface displays the current state of the state machine and an area with the last 1500 logs. This allows the user to access a bit more than 5 days of logs information. On the interface I also put a button allowing the user to switch manually between “routine check” and “hold” state.

I developed this code the same way I did for the energy management system. You can find extracts of some parts of the code in appendix 8. Now the SolarLab can reboot itself automatically. The delay between the crash and the reboot being completely done is around 10 minutes.

<10 mins
time between a crash and the complete auto reboot

288
auto checks per day are run to detect crashes

5
is the number of states configured in the script

Figure 24: Key figures of the auto-reboot system

E. Data analysis

1. Example of a typical day

In this part I want to share the data that the SolarLab can collect during a normal day and how the automation done to the system works in real conditions. The data used in this example are from the 22nd of April 2026. The weather of this day was mostly sunny with some few clouds passing from time to time in the sky.

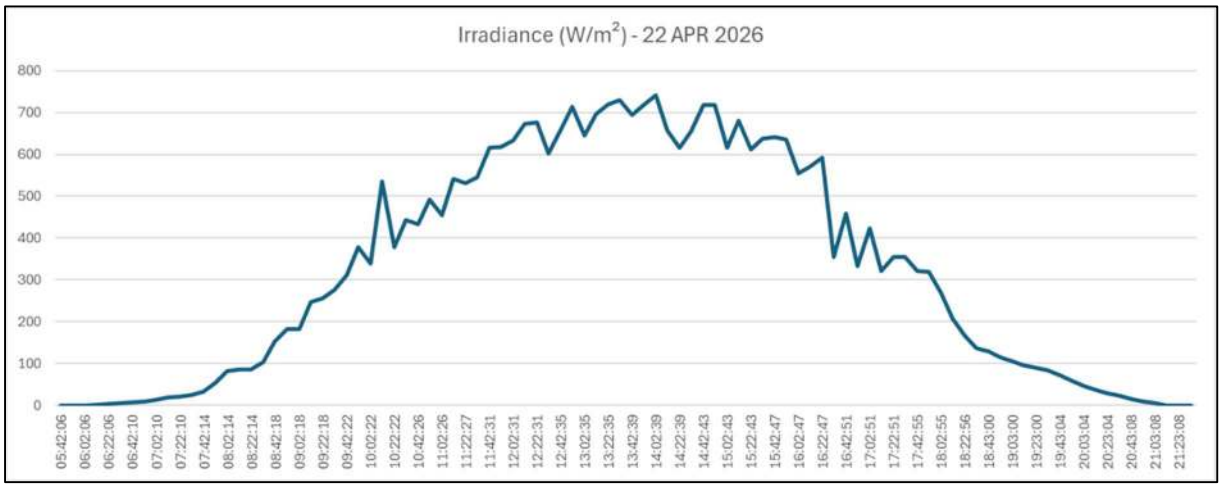


Figure 25: Irradiance data from the pyranometer

This first graph represents irradiance data measured by the pyranometer. During this day the maximum irradiance measured was around 700W/m².

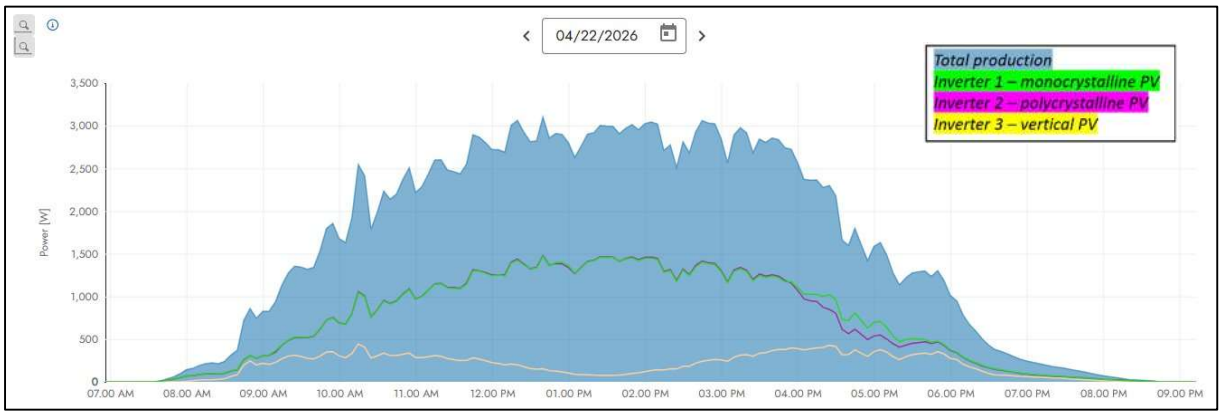


Figure 26: Detailed production data

This second graph displays the detailed production during the day. As was to be expected, the monocrystalline and polycrystalline PV, that have very close characteristics, produced almost the same power during the day. We can still see that around 05:00 PM, the production of the polycrystalline drops a little faster than the monocrystalline. This is due to the cast shadow on the PV. Indeed, a small tree created a shadow on a fraction of the bottom panels which are the polycrystalline ones. During my semester I also did a bit of analyses of the shadows (related to the mountain around Bergen), you can find part of these explanations in appendix 9. This production data also shows very well the production curve created by the vertical PV. The vertical PV are facing East and West. Thanks to this orientation we have two pics of production

during the day: one in the morning and one at the end of the afternoon. This allowed to produce energy when it is generally used and complete the standard PV installation.

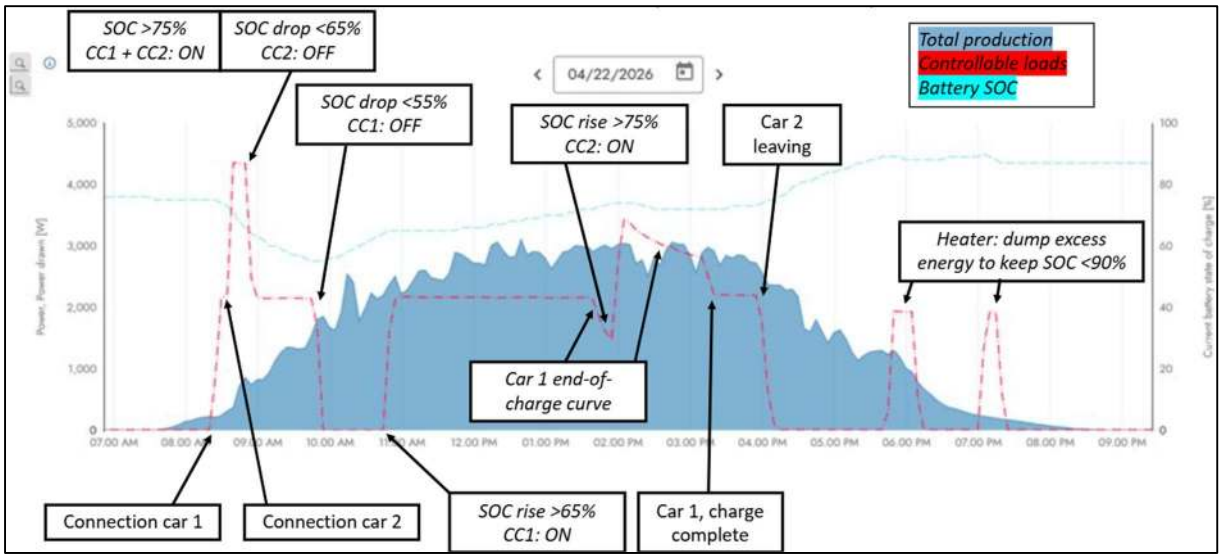


Figure 27: Total production, SOC and controllable loads data

This last graph presents the evolution of the SOC of the SolarLab battery and the power of the controllable loads during the days. With the automated system, loads are activated and deactivated multiple times during the days based on the SOC. The hysteresis logic implement on the code proves its usefulness here to prevent activation and immediate deactivation of load. For example, the CC1 which is ON (when EL is at 1 or more), switch ON when SOC goes higher 65% and only deactivated when SOC drop below 55%.

2. Theories crashes cause

As explained before, the SolarLab tends to blackout time to time. During my period at HVL it happened 11 times without any explanation. As today there is no unique reason to explain it. It is quite hard to find reasons for the crashes. Indeed, when the blackout happens all systems switch off immediately, preventing the devices from keeping an error message in their logs.

To start organising and keeping track of the crash, I created an Excel file reporting the crashes with information on the state of the installation at the moment of the crash such as production, consumption, number of cars charging and for how long, … The goal of this file is to identify on the long-term perspective situations than lead to SolarLab crashes.

The first conclusion of this file outlines a scenario where most crashes occur. This specific scenario takes place during the morning, regardless of the battery's SOC. It is characterised by important fluctuations in solar irradiance, which cause severe variations in photovoltaic power generation. Concurrently, a vehicle is charging on the system, this factor might introduce very short but high-amplitude power spike, but this phenomenon needs to be confirmed. Immediately prior to the system crash, the battery is consistently found in a state of discharge, operating between 1,000 W and 2,000 W. The main theory at this point is that the Sunny Island, which is responsible of generating the micro-grid, is unable to maintain the balance between consumption and generation, causing the microgrid's frequency to fall outside the nominal range and triggering a safety protection system. It is also possible that this phenomenon is amplified due to disruptions in communications at certain times caused by electrical currents.

F. SolarLab management

1. Made available data and tools

One of my duties in the background during my semester at HVL was to make sure that the data collected and measurement tools within SolarLab were available. It is in this context, that I was asked by a professor, Mojtaba Yousefi, to prepare and perform a presentation of all the data available at SolarLab. I made this presentation to one of his classes. The goal was to present to his students' real energy data related to the energy sector. As part of their final project, they had to work on such data. I then shared the data, by creating and setting up specific access accounts for guests, see *§IV.F.3. SolarLab IT tools*.

In addition, I also had to give access to the SolarLab to HVL engineers to enable them to calibrate another irradiance sensor. To make their work easier, I quickly developed some code that allows a reading from the pyranometer to be taken on demand, rather than having to rely on the readings taken every 10 minutes for the website.

2. Conference preparation

In my role as a research assistant to Richard J. Grant, I also had the opportunity to help prepare for a conference presentation he was taking part in in India this semester. The theme of the presentation was "the development of a versatile solar laboratory for educational and research purposes" at the international AMCEHA[6] Conference in India. My task was to trace the entire history of the SolarLab and to provide data analyses to explain how it works.

In the same context, I also took part, on an occasional basis, in supervising the preparation for final oral examinations for groups of his students

3. SolarLab IT tools

When I arrived, there was no specific HVL account or email address for the SolarLab. This situation led to constraints such as the need to set up new user accounts or new systems. Indeed, it was always necessary to use the login details of a responsible person at the HVL to log in and configure the systems. Another difficulty was ensuring continuity for ICAM students when technical support services were contacted. This approach is risky if one of the responsible people changes roles.

Therefore, I suggested creating SolarLab email addresses. Two accounts were then created:

- solarlab@hvl.no, used for admin accounts on all the IT tools and to centralise external communications of SolarLab.
- solarlab-data@hvl.no, used for guest accounts on the IT tools supporting the management of multiple accounts.

With that new system in place, I took the opportunity to put together a document summarising all the accounts and passwords used at SolarLab.

[6] AMCEHA: Advanced Materials for Clean Energy and Health Application

G. Other modifications

In this next part, I will present some of the other modifications, upgrades or work I brought to the SolarLab whilst I was there.

1. Realisation of adaptors

At the back of the SolarLab, three P17 reinforced 32A sockets for the pilotable load were installed before my semester. To make these sockets more versatile, I made removable adaptors that allow lower-power devices fitted with a standard plug (type F) to be connected. The adaptors have on one part a P17 plug and at the other side a water-resistant type F socket.



Figure 28: Adapters made and installed

2. Improve Raspberry Pi resilience

As described in the previous part of my report, the Raspberry Pi started to become a very important tool within SolarLab. In case of an issue with the Raspberry Pi currently in use the consequences would be important, and it will take quite some time to re-setup everything. With that in-mind, Richard asked me to find a way to create a clean backup that can be restored in case of an issue.

Thanks to Raspberry Pi documentation, I found a way to created backup of the card. I then created storage folders for backup within the SolarLab SharePoint. I also made a procedure to manage properly the creation, storage and then use of backups to ensure longevity of this system.

Following the same principle of reliability, I have ensured that, in the event of a reboot of the Raspberry Pi, the important programmes restart automatically. To do that I used the “crontab” menu. Crontab (short for “chronograph table”) allows scripts or commands to be ran automatically on a Raspberry Pi at specific dates or times.

```
 GNU nano 5.4                                /tmp/crontab.t6Obvw/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

@reboot sleep 120 && DISPLAY=:0 python3 /home/tom/Desktop/Projet_Web/python_radiation10.py &

*/10 * * * * /home/tom/Desktop/Projet_Web/transfert_script.sh &

@reboot sleep 130 && DISPLAY=:0 python3 /home/tom/Desktop/Shellys_auto/SolarLab_energy_management.py &

@reboot sleep 140 && DISPLAY=:0 python3 /home/tom/Desktop/Auto_reboot/Auto_reboot.py &
```

Figure 29: Crontab programming interface

To simplify future development of Python scripts, I have also created a document listing the serial numbers, MAC addresses, reserved IP (or static IP) addresses, and switch and router connections port for each device in the SolarLab.



A static IP address is a network identifier that never changes. It ensures that a device can always be reached at the same address, eliminating the risk of disconnection after a restart. For a Python script, this is essential. It allows you to hard-code the address into your requests with the certainty that the script is targeting the correct device, thereby ensuring automation without any downtime.

3. Temperature sensor implementation

As indicated in an upper section of the report, I have also set up outdoor and indoor SwitchBot Meter. This connected device measures temperature and humidity. I have set up three of them.

The first one is inside the SolarLab and allows to follow the climate within the passive house. The second one is located just outside and allows to keep a track of the outdoor ambient temperature. The third and last one, is located directly under the monocrystalline PV.

This last one enables new possibilities by tracking PV temperature. Indeed, efficiency of solar panel vary based on its surface temperature. The efficiency of a solar panel decreases by 0.3% to 0.5% for every Celsius degree above 25 °C. At a temperature of 40 degrees, the panels lose between 4.5% and 7.5% of their efficiency. In the future, work of correlation will be possible between PV surface temperature and PV production. Here is an example of temperature graph that can be obtained (top of the next page):

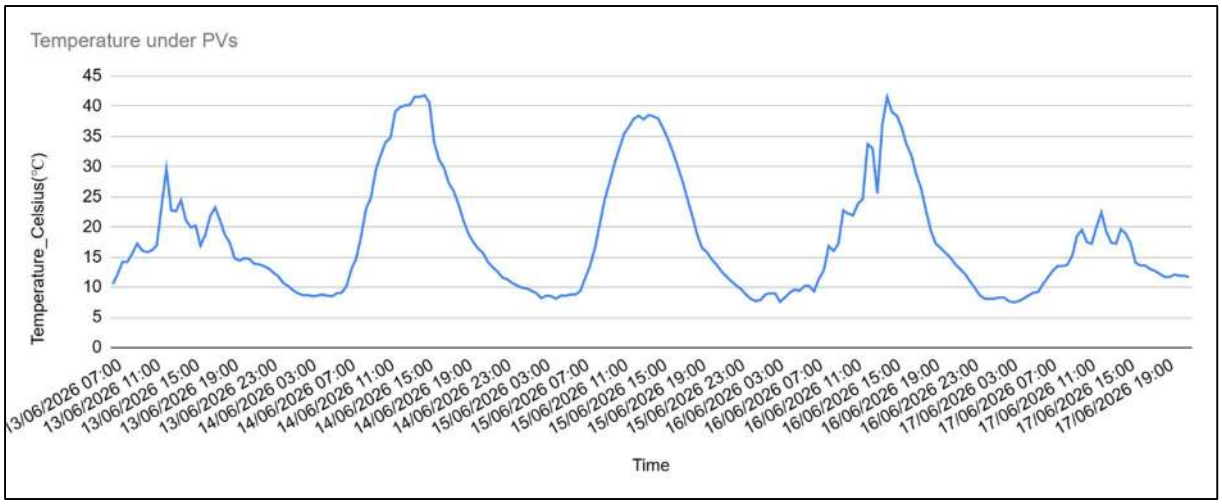


Figure 30: Temperature variations under PVs between 13th and 17th of June 2026

To hold these sensors in place, I designed and 3D-printed some mounts. The first one, which is quite standard, is screwed or glued to the wall and allows the device to measure the ambient temperature.

The second one allows the sensor to be secured against the underside of the PV panels. The bracket is fitted by rotating it into place within the PV mounting rails. It is then locked into position by the sensor itself inserted into it.



Figure 31: SwitchBot Meters inside their support and installation of the under PV support

4. Preparation for future website updates

The SolarLab website is currently running on an old PHP version. To keep it supported and safe, the website host company has recently started charging for old PHP versions website.

One of the main tasks of the student coming after me will be to carry out this main upgrade to have a free of charge website again. To do this, many other tools must update before doing the PHP upgrade. One of my tasks was to prepare this project by identifying the main updates that needed to be carried out. I made a note that summarises that.

Here is a summary table of the updates to be carried out:

	Current version	Proposed version
Joomla	Joomla **4.4.1**	Joomla **5.4.3** (and then **6.0.3**)
Extensions Joomla	Akeeba Backup Core for Joomla! **9.8.3** French **4.4.1.1** shaper_helixultimate **2.0.18** SP Page Builder **3.8.10** System - Helix Ultimate Framework **2.0.18**	Akeeba Backup Core for Joomla! **10.3.2** French **4.4.14.2** shaper_helixultimate **2.2.4** SP Page Builder **5.3.2** System - Helix Ultimate Framework **2.2.4**
PHP	PHP **8.1.34**	PHP **8.4** or **8.5**
IONOS	£7.81 extended support	Free of charge

Table 3: Summary of updates that need to be carried out

V. CONCLUSION AND PERSPECTIVE

This semester at SolarLab has enabled me to tackle challenges relating to optimising the production, management and resilience of the photovoltaic system. The maximum power output was increased from 2,000 W to 3,000 W by reconfiguring the panel strings, thereby resolving the issue of inverter saturation. At the same time, a predictive production model was developed, providing a solid foundation for proactive energy management.

Automation and reliability improvement were at the heart of my contributions, such as:

- Energy management system: automatic activation/deactivation of controllable loads based on SOC, with hysteresis logic to prevent oscillations.
- SolarLab resilience: implementation of a backup power supply and an automatic restart system, reducing the recovery time to 10 minutes following a crash.
- IT tools and data: creation of generic accounts, Raspberry Pi backups, and preparation of website updates to ensure the system's long-term viability.

For the very first time, the SolarLab can operate fully autonomously, managing its charge levels and loads, and even rebooting itself in the event of a crash.

There are several next steps for SolarLab. The first is to migrate the website to a more recent version of PHP so that it no longer requires a paid subscription. Additionally, a step could be to develop an improved version of my energy management system, using production forecasts to optimise energy usage.

Another major project idea would be to establish a partnership with the HVL laboratory working on hydrogen, and to suggest that they use the excess energy from the SolarLab to produce green hydrogen. This would put the excess energy to good use, rather than simply powering an outdoor heater.

Finally, there will always be minor improvements to be made to the SolarLab, such as building a shelf for the UPS system's backup battery or other small supports. Studying the possibility of adding a summer/winter tilt adjustment to the solar panels to optimise their power output is also another idea.

VI. BIBLIOGRAPHY

[1] Mæhlum, E., et al. (2024). Technical potential of solar energy in buildings across Norway: capacity and demand. Solar Energy, 240, 112345. DOI:10.1016/j.solener.2024.112345

[2] Li, W., et al. (2020). Optimal Strategies in Home Energy Management System Integrating Solar Power, Energy Storage, and Vehicle-to-Grid for Grid Support and Energy Efficiency. IEEE Access, 8, 34567-34589. DOI:10.1109/ACCESS.2020.2975678

[3] Khan, M., et al. (2023). Energy management strategies of hybrid renewable energy systems: A review. Energy Exploration & Exploitation, 41(1), 1-25. DOI:10.1177/0309524X231200010

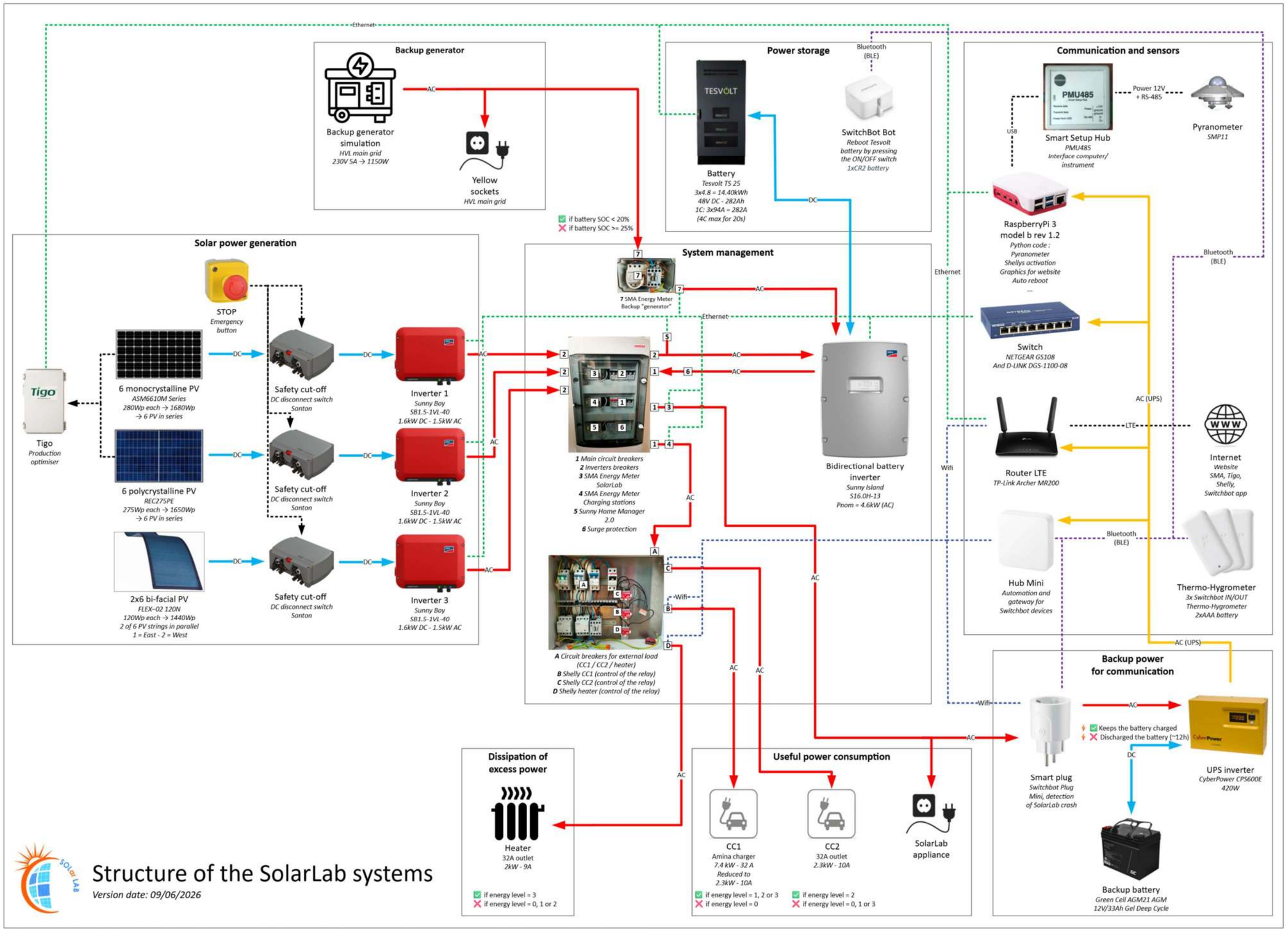
[4] Pavlovic, T., et al. (2023). Determination of the Installation Efficiency of Vertical Stationary Photovoltaic Modules with a Double-Sided "East–West"-Oriented Solar Panel. Energies, 15(3), 1635. DOI:10.3390/en15031635

[5] Al-Nuaimi, A., et al. (2022). East-west oriented photovoltaic power systems: model, benefits and technical evaluation. Renewable Energy, 189, 123-135. DOI:10.1016/j.renene.2022.02.060

[6] López, A., et al. (2025). Analysis of the impact of irradiance and temperature on photovoltaic production: A statistical and machine learning approach. Journal of Cleaner Production, 401, 123456. DOI:10.1016/j.jclepro.2025.123456

VII. APPENDICES

Appendix 1: Structure of the SolarLab systems



Appendix 2: Calculations for PV wiring modifications

2. Configuration and calculation details

a. Inverter operating range (DC input)

The specifications of the inverters used are as follows:

Inverter DC inputs specifications				
Voltage	Minimum	Start	MPP range	Maximum
	50V	*80V*	*160-500V*	*600V*
Current	Maximum		Maximum short-circuit	
	10A		*18A*	

b. Solar panels maximum output

The specifications of the solar panels used are as follows:

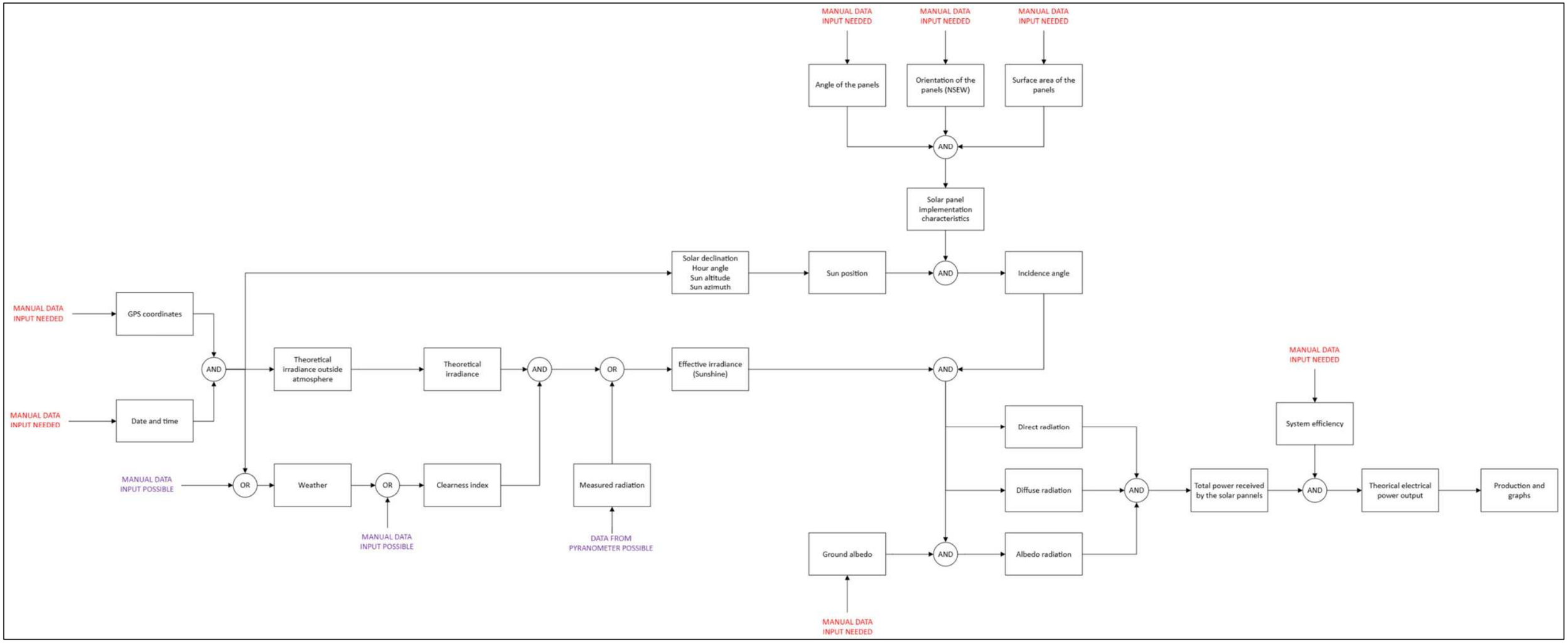
	Solar Panels DC output specifications	
	Monocrystalline	Polycrystalline
Peak power	280Wp	275Wp
Rated voltage	31.55V	31.50V
Rated current	8.98A	8.74A

Output voltage and current from the panels in the initial configuration:

	Output with 2 strings of 3 panels (maximum)	
	Monocrystalline	Polycrystalline
Voltage $U = U_1 = U_2$	U = 3 x 31.55 = 94.65V	U = 3 x 31.50 = 94.50V
Current $I = I_1 + I_2$	I = 8.98 + 8.98 = 17.96A (>10A)	I = 8.74 + 8.74 = 17.48A (>10A)
Peak power P	P = 94.65 x 10 = 946.50W	P = 94.50 x 10 = 945.00W

Output voltage and current from the panels in the proposed configuration:

	Output with 1 string of 6 panels (maximum)	
	Monocrystalline	Polycrystalline
Voltage U	U = 6 x 31.55 = 189.30V	U = 6 x 31.50 = 189.00V
Current I	I = 8.98A (<10A)	I = 8.74A (<10A)
Peak power P	P = 189.30 x 8.98 = 1,699.91W (-> 1,600W)	P = 189.00 x 8.74 = 1,651.86W (-> 1,600W)


MANUAL DATA INPUT NEEDED
MANUAL DATA INPUT NEEDED
MANUAL DATA INPUT NEEDED
Angle of the panels
Orientation of the panels (NSEW)
Surface area of the panels
AND
Solar panel implementation characteristics
Solar declination
Hour angle
Sun altitude
Sun azimuth
Sun position
AND
Incidence angle
MANUAL DATA INPUT NEEDED
GPS coordinates
AND
Theoretical irradiance outside atmosphere
Theoretical irradiance
AND
OR
Effective irradiance (Sunshine)
AND
MANUAL DATA INPUT NEEDED
MANUAL DATA INPUT NEEDED
Date and time
Direct radiation
System efficiency
MANUAL DATA INPUT POSSIBLE
OR
Weather
OR
Clearness index
Measured radiation
Diffuse radiation
AND
Total power received by the solar pannels
AND
Theorical electrical power output
Production and graphs
MANUAL DATA INPUT POSSIBLE
DATA FROM PYRANOMETER POSSIBLE
Ground albedo
AND
Albedo radiation
MANUAL DATA INPUT NEEDED

Appendix 4: Calculation made at each step of the model

Step	Spreadsheets	Column name	Calculation or action made
1	Sun position	Day of the year (n)	Extract day number in year from date study range.
2	Sun position	Equation of Time in min	*= 9,87*SIN(2*RADIANS(360/365*(n))) - 7,53*COS(RADIANS(360/365*(n-81))) - 1,5*SIN(RADIANS(360/365*(n-81)))*
3	Sun position	Longitude Correction (TC)	*= 4*(Longitude - 15*ENT(Longitude/15))*
4	Sun position	True Solar Time (TST)	*= HOUR + MINUTE/60 + (Equation of Time in min + TC)/60*
5	Sun position	Solar declination (δ)	*= 23,45*SIN(RADIANS(360/365*(n-81)))*
6	Sun position	Hour angle (ω)	*= 15*(TST-12)*
7	Sun position	Solar altitude (h)	*= DEGRES(ASIN(SIN(RADIANS(Latitude)) * SIN(RADIANS(δ)) + COS(RADIANS(Latitude)) * COS(RADIANS(δ)) * COS(RADIANS(ω))))*
8	Sun position	Solar altitude (h) (>0)	Keep only positive values for Solar altitude (h)
9	Sun position	Solar azimut (α)	*= MOD(180 - DEGRES(ATAN2(SIN(RADIANS(ω)); COS(RADIANS(ω)) * SIN(RADIANS(Latitude)) - TAN(RADIANS(δ)) * COS(RADIANS(Latitude)))) + 90; 360) MOD(180 - DEGRES(ATAN2(SIN(RADIANS(ω)); COS(RADIANS(ω)) * SIN(RADIANS(Latitude)) - TAN(RADIANS(δ)) * COS(RADIANS(Latitude)))) + 90; 360)*
10	Radiations	Theoretical Irradiance Outside the Atmosphere (ETO) W/m²	*= 1361 * (1 + 0,033 * COS(RADIANS(360 * n / 365))) * SIN(RADIANS(h (>0)))*
11	Radiations	Theoretical Irradiance (Perfect Weather Conditions) W/m²	*= IF(SIN(RADIANS(h (>0))) = 0; 0; ETO * EXP(-0,1 / SIN(RADIANS(h (>0)))))*
12	Radiations	Effective irradiance (with clearness index) W/m²	*= Theoretical Irradiance in Perfect Weather Conditions * Clearness index*
13	Radiations	Irradiance use for the solar panels production W/m²	Use the irradiance from the pyranometer if available; otherwise, use the calculated value.
14	Production	Incidence angle	*= DEGRES(ACOS(SIN(RADIANS(h)) * COS(RADIANS(PV tilt)) + COS(RADIANS(h)) * SIN(RADIANS(PV tilt)) * COS(RADIANS(α - PV orientation))))*
15	Production	Direct radiation	*= IF(COS(RADIANS(Incidence angle)) <= 0; 0; Irradiance use for the solar panels production * COS(RADIANS(Incidence angle)))*
16	Production	Diffuse radiation	*= IF (Theoretical Irradiance (Perfect Weather Conditions)=0; 0; Irradiance use for the solar panels production * (1 - SI(Irradiance use for the solar panels production / Theoretical Irradiance (Perfect Weather Conditions) > 1; 1; Irradiance use for the solar panels production / Theoretical Irradiance (Perfect Weather Conditions))) * (1 + COS(RADIANS(PV tilt))) / 2)*
17	Production	Albedo radiation	*= (Irradiance use for the solar panels production) * Ground albedo * (1 - COS(RADIANS(PV tilt))) / 2*
18	Production	Total radiation	*= Direct radiation + Diffuse radiation + Albedo radiation*
19	Production	Theorical electrical power output W/m²	*= Total radiation * Panel efficiency*
20	Production	Theorical electrical power output W	*= Theorical electrical power output W/m² * Total area*
21	Production	Energy produced per interval kWh	*= (Theorical electrical power output W / 1000) * Interval in hours*
22	Production	Accumulated energy kWh	Cumulation of energy produced per interval
23	Production	The other columns are used to calculate the other side of the PV if it exists in the same way, and then the sum of the two sides.	

Appendix 5: Extract of the energy management script

```python
# ====================================================
# Global constants
# ====================================================
#region Variables and constants
# Protection
SOC_MIN = 23
MAX_AMP = 25
CC1_AMP = 10
CC2_AMP = 10
HEATER_AMP = 9

# Energy level thresholds
SOC_EL0_EL1 = 65
SOC_EL1_EL2 = 75
SOC_EL2_EL3 = 90
SOC_EL3_EL2 = 88
SOC_EL2_EL1 = 65
SOC_EL1_EL0 = 55

# Auto/Manual mode thresholds
SOC_AUTOLOW_MANU = 30
SOC_MANU_AUTOLOW = 25
SOC_MANU_AUTOHIGH = 90
SOC_AUTOHIGH_MANU = 85
MANU_MAX_DISCHARGE = 30

# IP addresses
SUNNY_ISLAND_IP = "192.168.1.10" #SMA Speed Wire Connection IP
SHELLY_CC1_IP = "192.168.1.200"
SHELLY_CC2_IP = "192.168.1.201"
SHELLY_HEATER_IP = "192.168.1.202"

# Loop delay (seconds)
LOOP_DELAY = 60

# Shelly credentials
SHELLY_USER = "admin"
SHELLY_PASSWORD = "admin"
RELAY_ID = 0
```

```python
# ====================================================
# Configuration of external devices
# ====================================================
#region SunnyIsland
class SunnyIslandSOC:
    def __init__(self, ip=SUNNY_ISLAND_IP, port=502, unit_id=3):
        self.ip = ip
        self.port = port
        self.unit_id = unit_id
        self.client = ModbusTcpClient(ip, port=port)

    def read_soc(self):
        try:
            result = self.client.read_holding_registers(30845, 2, unit=self.unit_id)
            if result.isError():
                print(f"Modbus error: {result}")
                return None
            decoder = BinaryPayloadDecoder.fromRegisters(
                result.registers,
                byteorder=Endian.Big,
                wordorder=Endian.Big
            )
            soc = decoder.decode_32bit_uint()
            return soc
        except Exception as e:
            print(f"Exception reading SOC: {e}")
            return None
#Endregion
```

```python
#region Shellys
class ShellyDevice:
    def __init__(self, ip, name):
        self.ip = ip
        self.name = name
        self.url = f"http://{ip}/rpc/Switch.Set"
        self.auth = HTTPBasicAuth(SHELLY_USER, SHELLY_PASSWORD) if SHELLY_USER else None

    def set_state(self, state):
        payload = {"id": RELAY_ID, "on": bool(state)}
        try:
            response = requests.post(
                self.url,
                json=payload,
                auth=self.auth,
                timeout=5
            )
            if response.status_code == 200:
                print(f"{self.name} set to {'ON' if state else 'OFF'}.")
            else:
                print(f"HTTP error {response.status_code}: {response.text}")
        except requests.exceptions.RequestException as e:
            print(f"Connection error for {self.name}: {e}")

    def get_state(self):
        try:
            response = requests.get(
                f"http://{self.ip}/rpc/Switch.GetStatus?id={RELAY_ID}",
                auth=self.auth,
                timeout=5
            )
            if response.status_code == 200:
                data = response.json()
                return data.get("output", False)
            else:
                print(f"Error reading {self.name} state: HTTP {response.status_code} - {response.text}")
                return None
        except Exception as e:
            print(f"Error reading {self.name} state: {e}")
            return None
#endregion
```

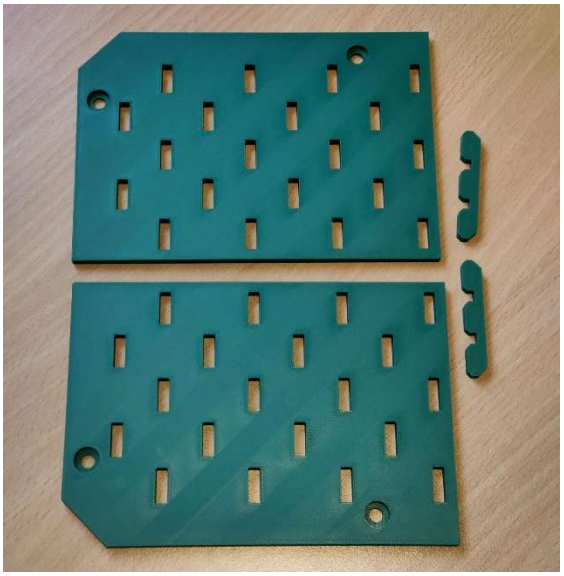
```python
        #region Auto mode logic
        # Energy level with hysteresis
        # Level increase
        if energy_level == 0 and SOC_read >= SOC_EL0_EL1:
            energy_level = 1
        elif energy_level == 1 and SOC_read >= SOC_EL1_EL2:
            energy_level = 2
        elif energy_level == 2 and SOC_read >= SOC_EL2_EL3:
            energy_level = 3

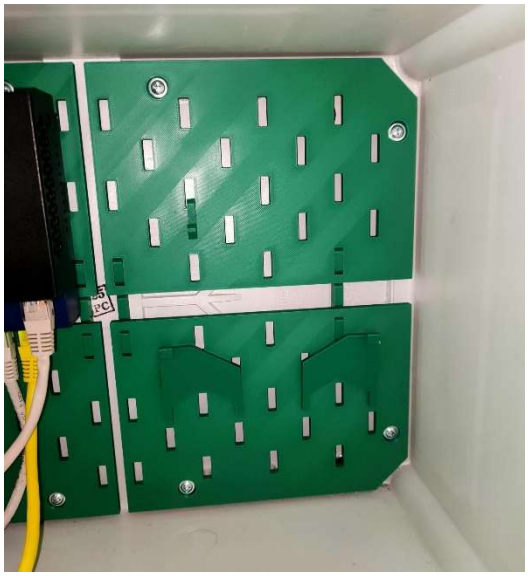
        # Level decrease
        elif energy_level == 3 and SOC_read < SOC_EL3_EL2:
            energy_level = 2
        elif energy_level == 2 and SOC_read < SOC_EL2_EL1:
            energy_level = 1
        elif energy_level == 1 and SOC_read < SOC_EL1_EL0:
            energy_level = 0

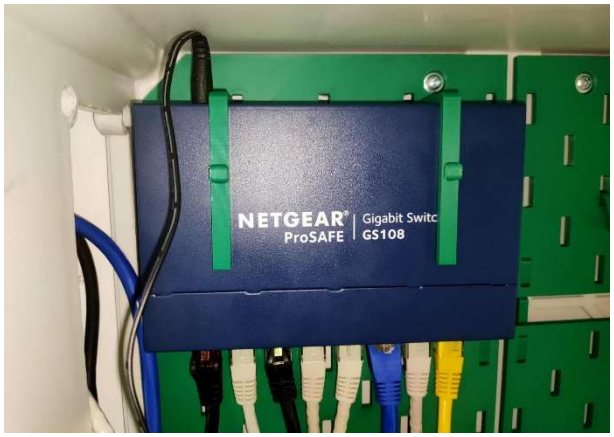
        # Set new states
        SHELLY_CC1_new_state = SHELLY_CC1_mode if SHELLY_CC1_mode == 1 else (1 if energy_level >= 1 else 0)
        SHELLY_CC2_new_state = SHELLY_CC2_mode if SHELLY_CC2_mode == 1 else (1 if energy_level >= 2 else 0)
        SHELLY_HEATER_new_state = SHELLY_HEATER_mode if SHELLY_HEATER_mode == 1 else (1 if energy_level >= 3 else 0)
        #endregion
```

Appendix 6: Switch and router mounting system

To secure all the devices to the back of the box, I've created a system of backplates with mounting points. The backplates consists of four symmetrical sections, each secured by two screws. The plates are connected by brackets to stiffen the whole assembly. Doing it this way makes it easy to create small brackets with hooks to secure the devices. This system will make it easy to replace the devices in the future should they be upgraded. Below are some pictures of the several sub-assemblies:



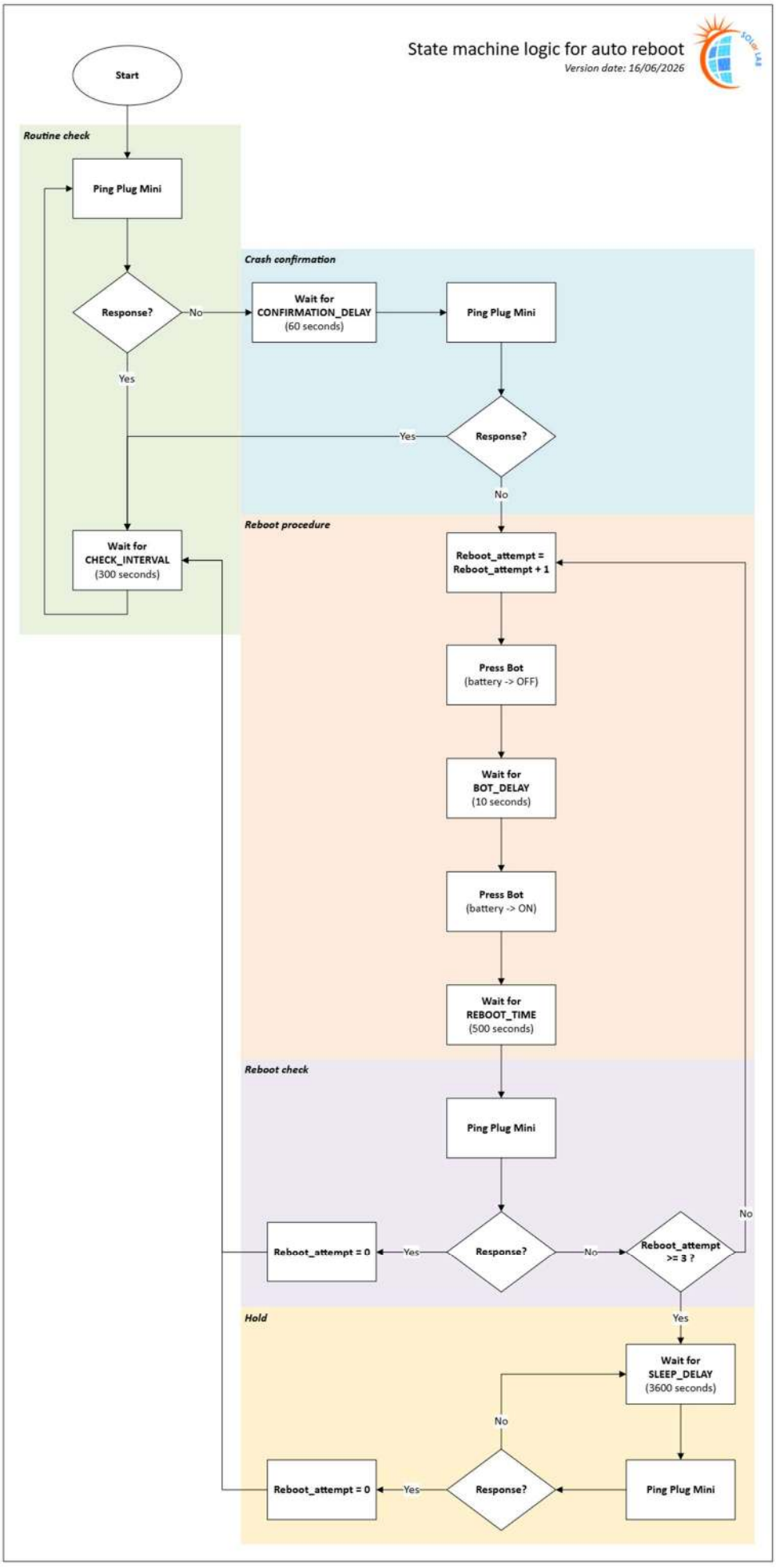








Appendix 7: Auto reboot state machine algorithm diagram



Appendix 8: Extract of the auto reboot script

```
# --- SWITCHBOT API CONFIGURATION ---
BOT_DEVICE_ID = "E77601865337"
TOKEN = "TOKEN HERE"
SECRET = "CODE HERE"

# --- NETWORK CONFIGURATION ---
PLUG_IP = "192.168.1.203" # Static IP of the Plug Mini on the router

# --- TIMING CONSTANTS (IN SECONDS) ---
CHECK_INTERVAL = 300              # Time between normal plug connectivity checks
CONFIRMATION_DELAY = 60           # Delay before the second ping confirmation
BOT_DELAY = 10                    # Delay between the first and second bot actions
REBOOT_TIME = 500                 # Time the installation need to reboot
SLEEP_DELAY = 3600                # Delay between check on hold mode

# --- NUMBER OF REBOOT ---
MAX_REBOOT_ALLOWED = 3
Reboot_attempt = 0

# --- Maximum number of log lines ---
MAX_LOG_LINES = 1500
```

```
# --- GRAPHICAL USER INTERFACE ---
root = tk.Tk()
root.title("Solarlab monitor")
root.geometry("700x400")
root.configure(bg="white")
root.minsize(400, 300)

# Frame for the status and button
top_frame = tk.Frame(root, bg="white")
top_frame.pack(pady=10, padx=10, fill=tk.X)

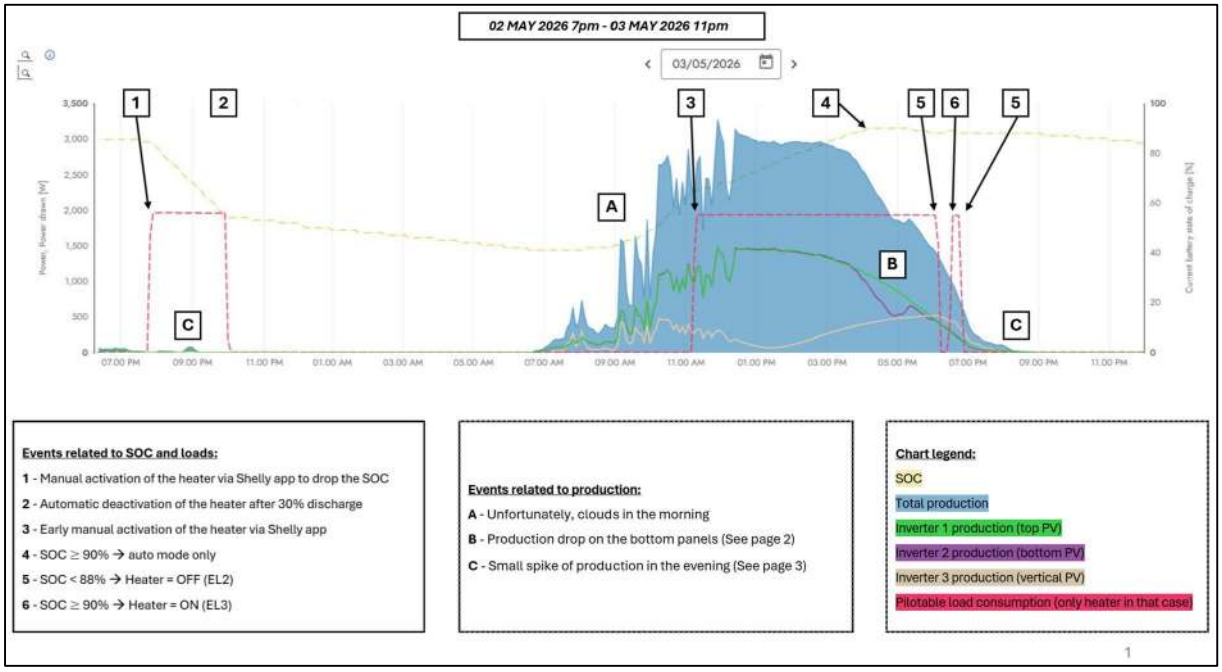
# Label to display the status
state_label = tk.Label(
    top_frame,
    text=f"State: {state}",
    bg="white",
    font=('Arial', 12, 'bold')
)
state_label.pack(pady=5)

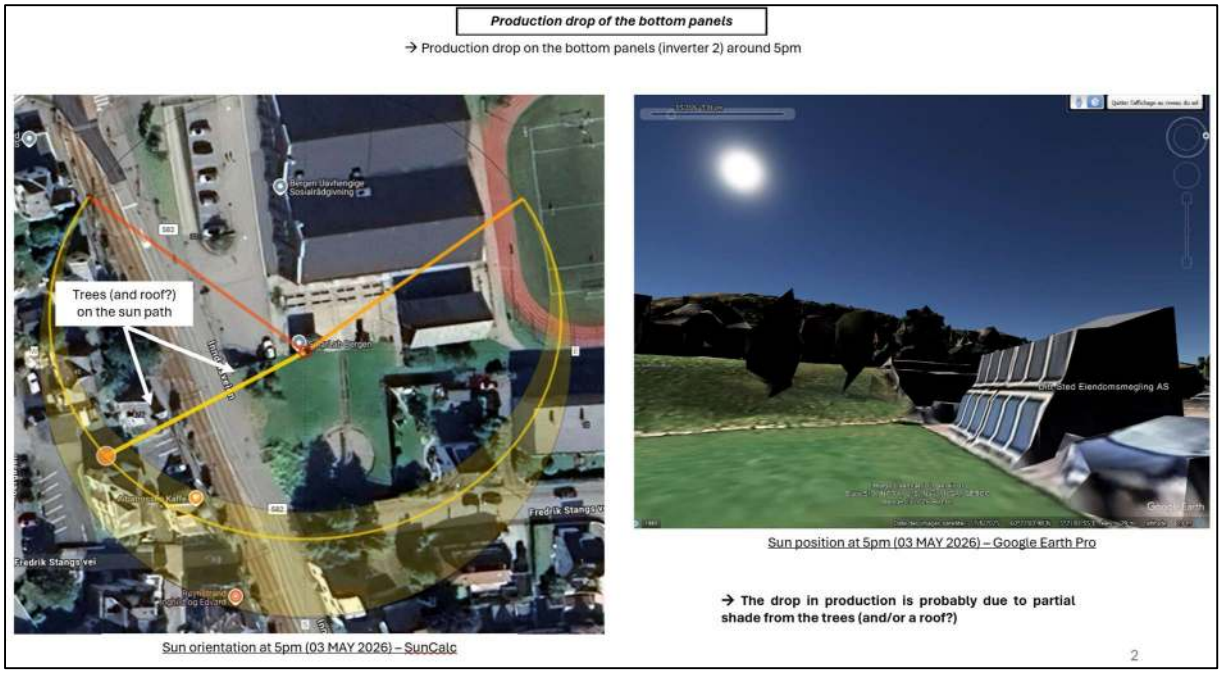
# Button to switch the state
toggle_button = tk.Button(
    top_frame,
    text="Switch between HOLD and ROUTINE_CHECK states",
    command=toggle_state,
    bg="lightgray",
    relief=tk.FLAT,
    padx=10,
    pady=5
)
toggle_button.pack(pady=5)

# Logs section (with scrolling)
log_area = scrolledtext.ScrolledText(
    root,
    width=80,
    height=12,
    wrap=tk.WORD,
    bg="white",
    font=('Courier', 10),
    padx=5,
    pady=5,
    borderwidth=1,
    relief=tk.SOLID
)
log_area.pack(padx=10, pady=(0, 10), fill=tk.BOTH, expand=True)

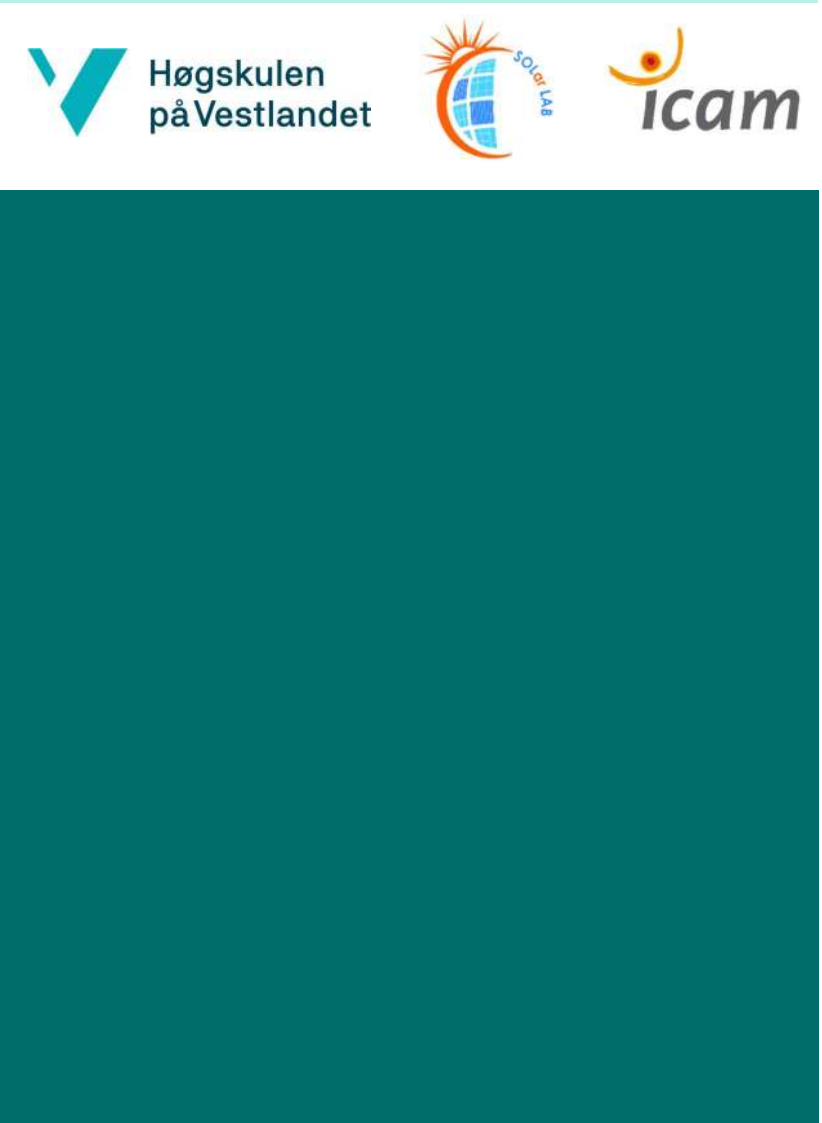
# Redirects stdout to the log area
sys.stdout = ConsoleOutput(log_area, max_lines=MAX_LOG_LINES)
```

Appendix 9: Explanation of unexpected production drop/rise








Høgskulen
på Vestlandet
SOLAR LAB
icam