Alexis GERGONNE

Department of Mechanical and Marine Engineering

End of Master Project

SOLARLAB

Under the supervision of Richard John Grant (Professor Mechanical and Marine Engineering)

Høgskulen på Vestlandet – Bergen – Norway

2023

Table of Contents

I. Acknowledgement 4

II. Introduction 5

1. Context 5

2. Solar Laboratory in Høgskulen på Vestlandet 5

3. Report plan and objectives for the semester 5

4. Important information for future student 6

III. Battery purchase and installation 7

1. Context 7

2. Supplier selection 7

3. Installation 8

IV. Python code 9

1. Context 9

2. Modules importation 10

3. Pyranometer configuration 10

4. Lists initialization 11

5. Date Importation 12

6. Data logging 13

7. Theoretical curves calculation 13

8. Graph design 15

9. Graph creation 16

10. Area for improvement 19

V. SolarLab website 21

1. Website via the Raspberry IP (abandoned solution) 21

2. Buying a domain 22

3. Joomla 22

4. Transmission of files from Raspberry to the server 23

5. SCP 23

6. Automatization with Cron task 24

VI. Various problems 25

1. Wi-Fi connection 25

2. Connection between laptop and raspberry 26

VII. Data analysis 28

1. Context 28

2. Analysis 29

3. Differences between theory and data 32

VIII. Conclusion 33

IX. Sources 35

X. Appendices 36

1. System of the tiny house 36

2. Tesvolt distributors 37

3. Screenshot of Joomla 38

4. Screenshot of SP Page Builder 39

5. Screenshot of Solarlab.online 40

I. Acknowledgement

I would like to thank my manager, Richard John Grant, who supervised me throughout the semester. I would like to thank him for the time he took, for his kindness and welcome.

I would also like to thank the university, the rector Gunnar Yttri, for welcoming me to his establishment, as well as Cathrine Palmstrom.

Finally, I want to thank the people at Icam who made this exchange possible, Christian Courties, Catherine Maury and Jean-Pierre Fradin, who oversees university exchanges.

II. Introduction

1. Context

The global warming is more and more present around the world. The consequences are from bad to worse and no one can deny that we have to make considerable effort to improve the situation, otherwise this planet will be less and less liveable. Indeed, everybody have to reduce his carbon footprint and this includes technology improvement especially in the energy field.

According to the IEA (International Energy Agency), electricity generation alone accounts for 41% of CO2 emissions worldwide. This is due to the different ways in which this energy is created, such as coal-fired power stations. Today, therefore, it is essential to use renewable energies: solar, wind, hydro, etc. According to the 2023 Ember Global Electricity Review, only 12% of electricity comes from renewables energies. If we go into a little more detail, we will see that housing is a major source of greenhouse gas emissions, since 27% of them are due to the production of electricity and heat used in buildings. These data show that our practices need to change.

That is why the passive house concept is now gaining ground. The aim is to use as less energy as possible to heat, cool, light and ventilate a building. This is achieved by a number of means, including insulation, the orientation of the house, the location of windows, and the use of efficient electrical appliances and solar panels.

2. Solar Laboratory in Høgskulen på Vestlandet

The Western Norway University of Applied Science has a branch devoted to renewable energies and a laboratory specifically dedicated to solar energy: the SolarLab. The SolarLab is a passive house located on the university campus. It has been designed to meet all standards, and the aim of the project is to carry out a wide range of calculations and measurements relating to solar energy.

The laboratory has a total surface area of 12 square metres and uses a CMV2 for space heating. The primary goal of this solar laboratory is to operate off-grid, meaning that it relies solely on the electricity produced by the solar panels and stored in the battery.

The passive house's photovoltaic system consists of two types of parts: 6 polycrystalline photovoltaic panels: REC Solar 275 (capacity of 275 W per panel) and 6 monocrystalline photovoltaic panels: ASM6610M 285 - Astroenergy (capacity of 285 W per panel). Three Sunny Boy inverters, each with a capacity of 1500W, are used to convert the direct current (DC) generated into alternating current (AC). You can find a system diagram in appendix 1.

Another battery inverter has been purchased last year, Sunny Island which is specifically designed for energy storage and battery charging. Unfortunately, SMA Sunny Islands are only compatible with a list of identified batteries and the actual Li-On battery is not part of this list. That is why the previous student found the best battery for the tiny house that I will have to buy and install.

3. Report plan and objectives for the semester

In keeping with what previous students have been able to do, the various tasks I will be carrying out during my semester are aimed at improving and developing this SolarLab.

My first task will be to buy and install the battery for this tiny house.

Then the second important task is to be able to display the sun's irradiance on the SolarLab in real time on a screen and, if possible, to be able to access it on any computer on campus. I am going to divide this part in two, with the python code I wrote first and the website I created second.

Finally, a small part will be about the data analysis I was able to record.

4. Important information for future student

In order to ensure follow-up and continuity in the project, I am going to detail a number of points exclusively for the future student. This will ensure a better understanding and in a few days the student will be able to take the project in hand.

So anyone else reading this report can avoid reading the parts written in blue.

All the writing in blue will be intended for future students.

III. Battery purchase and installation

1. Context

In 2022, a Sunny Island, battery inverter has been purchased in order to manage the electricity stored by the Power Oak Battery. Unfortunately the communication between the battery and the inverter was impossible. Because of this we could not launch all the electric system in the tiny house and a new battery had to be identified. Tom, the previous student, did a great work comparing all the battery from the SMA list compatible with the inverter. It was decided to buy the Tesvolt TS25 14,4 Battery.



My work was to find the best supplier.

2. Supplier selection

First of all, I read and took a list that Tom had drawn up before leaving. This list was a good basis on which to start my work. I checked whether all the suppliers he had found still sold the Tesvolt. If they were not, I obviously removed the supplier from the list. I also found some new suppliers that Tom hadn't noticed.

The Tesvolt battery is not widely available and the list of distributors is fairly limited. Thus, thanks to this research stage, I was able to draw up a table of the seven suppliers I found to compare them.

- After a meeting with my manager, I knew the most important points for choosing this distributor:

Firstly, unsurprisingly, the price. This is a significant budget for a university, so we have to be careful. Especially as this type of battery is very bulky and expensive.

Secondly, reviews of the company. We don't want to make another mistake and buy a battery that doesn't work, we need to be sure that the company is reliable and professional. To do this, I checked the reviews on various websites: Google Chrome, TrusPilot, their own website...

The last important point was the country of the company. We would prefer to have a company close to Norway for a number of reasons. The delivery time, because ordering a battery of this size can take a long time if it is sent from the other side of the world. Also, it seemed easy to deal with a European country and it could be easy to trust.

- I then emailed each company to see if they were still available and to get a quote.

As a result, you will find the table with all these points in appendix 2.

Taking all these criteria into account, our choice fell on the French distributor Perma Batteries, which seemed to be a reliable and competent company. The fact that this company is French made it easier for me to talk to them before putting Richard and the salesman in touch to make the purchase.

3. Installation

The delivery time was around three months. The battery should have arrived towards the end of November, at the beginning of December at the latest. But due to customs issues, the delivery took longer and did not arrive before I left Norway. This stage will be the mission of the next student.

As you can see, one of your jobs will be to install the battery. Obviously, you will have to remove the old one first before installing the new one. After talking to Perma Batteries, it is quite possible to arrange calls with them if you need help with the installation. They are French so this may be a good option for you if you have any problems getting started.

It is something I have been looking forward to because it is really practical, interesting and I wanted to see this tiny house up and running and off the grid thanks to my tasks. It is a real regret for me because it should have been one of my main missions during my semester. Even if it wasn't my fault, it was frustrating not to be able to do anything about it. My work at SolarLab was therefore less significant than it should have been.

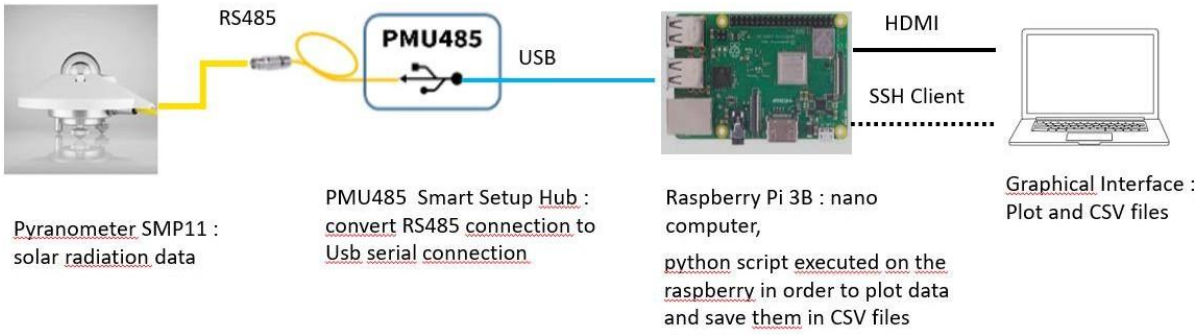
IV. Python code

For the next student: for this part, you need to take the time to read and understand. If anything remains unclear, don't hesitate to ask me questions or make a call.

1. Context

On the roof of the tiny house is a device called a pyranometer (Kipp & Zonen SMP11 Pyranometer) that can read the sun's irradiance. This device sends the data to a nanocomputer, a Raspberry Pi, via another device (Kipp & Zonen PMU485) capable of converting an RS485 connection into a USB 2.0 connection.

The goal is to use the raspberry to save the data and create graphics with them.



Scheme of the system

Tom started writing a program in Python in order to achieve this goal. Unfortunately and despite his efforts, it didn't work and had several errors. Moreover, it was a bit complicated for me to work on the same code that I hadn't written. After several unsuccessful attempts, I decided to start again from scratch. As it happens, it was the right solution because I immediately made progress on this one.

I wrote this code throughout the semester. In fact, each week I modified certain parts of it and improved it to make it as good as possible. It would be impossible and too long to explain all the changes, so I will just explain the final code. But I decided to explain each part separately for a better understanding, especially for the next students.

To understand the rest of this section, you should know that when you run this program, three graphs open in three different windows on the raspberry, one displaying data on a 24-hour scale, the other on a 72-hour scale and finally on a weekly scale. The data displayed starts at the time you launch the program.

Every two minutes the 'images' of these graphs are saved in a folder. At the same time, all the data is recorded every minute.

Now let's look at the parts of my code:

2. Modules importation

```
# FINAL PYTHON CODE - RADIATION - SOLARLAB

######## 1 - IMPORT MODULES ########
import math
import minimalmodbus
import csv
import os
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import matplotlib
matplotlib.use('TkAgg')
```

First of all, all the lines written in yellow starting with a "#" are simple comments for better understanding and to make the code easier to read. I have also used them to separate the different parts.

Modules are libraries of functions and tools that extend the basic functionality of Python.

math: Provides a mathematical function, for example to use cosine and sine in a calculation.

minimalmodbus: a module for communicating with peripherals via the Modbus protocol, in our case the PMU485 device.

csv: Allows you to read and create CSV files, which are useful for backing up our data.

os: Provides functions for interacting with the operating system, such as file and directory management. Once again, this function is useful for sorting saved data.

datetime and timedelta: Used to manipulate date and time objects.

matplotlib.pyplot: A plotting library for creating graphs and visualizations.

matplotlib.animation.FuncAnimation: Used to create animations in graphs.

matplotlib: Specific configuration for using the matplotlib library with a Tkinter (TkAgg) graphical interface.

Without all these modules, we can't run our code. This is necessary because basic python is not capable of doing all the tasks we need.

3. Pyranometer configuration

```
14 ######## 2 - PYRANOMETER CONFIGURATION ########
15 instrument = minimalmodbus.Instrument("/dev/ttyUSB0", 1)
16 instrument.serial.baudrate = 19200
17 instrument.serial.parity = minimalmodbus.serial.PARITY_EVEN
18 instrument.serial.bytesize = 8
19 instrument.serial.stopbits = 1
20 instrument.mode = minimalmodbus.MODE_RTU
```

This part is used to configure the communication with the pyranometer, essential in our case.

- Creation of the minimalmodbus instrument :

instrument = minimalmodbus.Instrument("/dev/ttyUSB0", 1): This line creates an instrument object using the Instrument class of the minimalmodbus module. The instrument is associated with the pyranometer. Here, the device is located at "/dev/ttyUSB0" and is accessible at Modbus address 1. It is simply where the pyranometer is plugged.

- Configuring serial communication parameters :

The following lines configure the serial communication parameters for the instrument. I have taken this part of Tom's code. In fact, all this information was written in the pyranometer datasheet that. So I didn't choose any of the following parameters:

instrument.serial.baudrate = 19200: Sets the baud rate to 19200 bits per second.

instrument.serial.parity = minimalmodbus.serial.PARITY_EVEN: Sets parity to even.

instrument.serial.bytesize = 8: Sets the byte size to 8 bits.

instrument.serial.stopbits = 1: Sets the number of stop bits to 1.

instrument.mode = minimalmodbus.MODE_RTU: Sets the instrument to RTU (Remote Terminal Unit) mode for Modbus communication.

If you want to find this information about devices like the PMU or the pyranometer, you can easily find these documents in my Teams work directory, in useful documents > datasheet.

4. Lists initialization

```
######## 3 - DATES INITIALIZATION ########
# For the real curve
list_time = []
list_radiation = []

list_time_72hours = []
list_radiation_72hours = []

list_time_168hours = []
list_radiation_168hours = []

# For the curve outside the atmosphere
list_radiation_extra = []
list_radiation_extra72 = []
list_radiation_extra168 = []

# For the theoretical curve with clear sky
list_radiation_clearsky = []
list_radiation_clearsky72 = []
list_radiation_clearsky168 = []
```

All these lists will store the radiation data for the last 24 hours, 72 hours and 168 hours. For each graph, there are three different curves: the received data, the theoretical data with a clear sky and the theoretical data outside the atmosphere.

In summary, this section of the code prepares the data structures needed to store the actual solar radiation measurements as well as the theoretical values for different time periods. This data will be used later in the script to create graphs.

There were several reasons for using separate lists for the three data sets (for the last 24 hours, the last 72 hours and one week) rather than a single list:

Firstly, when I wrote this script, I started with the received data. Then I added the other curves. It was easier for me to add a new list for each new curve and not touch the part of the code that was already working.

The second reason is the clarity of the code. By using separate lists, the code becomes more readable and understandable for me and especially for future students. This is because each list has a clear and specific purpose, which makes it easier to understand the role of each set of data. This is essential because this script will be discovered, read and used by several students.

Thanks to this, we have better flexibility for the future, if the needs of the project change and a specific functionality is added for a given period, it is easier to adjust or extend the individual lists without affecting the others.

In addition, simplicity of management was important. I am not a programming expert and the use of separate lists makes it easier to manipulate and manage data specific to each period. If specific operations need to be performed on the data for a particular period, they can be carried out more transparently with separate lists.

5. Date Importation

```
43 ######## 4 - IMPORT DATES ########
44 current_date = datetime.now()
45 yesterday = current_date - timedelta(days=1)
46 three_days_ago = current_date - timedelta(days=3)
47 seven_days_ago = current_date - timedelta(days=7)
48
49 # Format dates as "xx/xx/xxxx"
50 current_date_str = current_date.strftime("%d/%m/%Y")
51 yesterday_str = yesterday.strftime("%d/%m/%Y")
52 three_days_ago_str = three_days_ago.strftime("%d/%m")
53 seven_days_ago_str = seven_days_ago.strftime("%d/%m")
```

The first part is used to obtain specific dates.

current_date = datetime.now(): Thanks to this line we get the current date and time when this line of code is executed.

yesterday = current_date - timedelta(days=1):

three_days_ago = datetime.now() - timedelta(days=3)

seven_days_ago = datetime.now() - timedelta(days=7)

For these three lines, we take the current day and subtract a number of days.

The second part is used to put the date in the right format day / month / year.

6. Data logging

```
######## 5 - DATA STORAGE ########
# Function to save data to a CSV file
def data_csv(current_time, radiation):
    year = current_time.year
    month = current_time.month
    day = current_time.day

    desktop_path = os.path.expanduser("~/Desktop")
    all_data_path = os.path.join(desktop_path, "All data")
    if not os.path.exists(all_data_path):
        os.makedirs(all_data_path)

    month_path = os.path.join(all_data_path, f"{year:04d}/{month:02d}")
    if not os.path.exists(month_path):
        os.makedirs(month_path)

    file_name = f"{day:02d}_{month:02d}_{year:04d}.csv"
    file_path = os.path.join(month_path, file_name)
    with open(file_path, "a") as csvfile:
        file = csv.writer(csvfile, delimiter=';')
        file.writerow([current_time.strftime("%H:%M:%S"), radiation])
```

First step: create folders to store data:

Script creates an "All data" folder on the desktop if it doesn't already exist. As I explained in "Importing modules", it uses the os module to manipulate the file system.

The second step is to create a specific folder for the current month/year:

A folder specific to the month/year is created inside the "All data" folder. The folder format is "MM/YYYY", where MM is the month with two digits and YYYY is the year with four digits.

Finally, the data is saved in a CSV file:

The current time and radiation value are stored in a different CSV file each day, in the format "day_month_year.csv". The file is created when it does not already exist, or opened in append mode (that is why it is written "a") to add new data.

7. Theoretical curves calculation

```python
######## 6 - THEORETICAL CURVE CREATION (extra and clearsky) ########
# Function to calculate the zenith angle based on the time
def calculate_zenith_angle(hour):
    formatted_hour = datetime.now()
    formatted_hour = datetime.strptime(formatted_hour.strftime("%H:%M"), "%H:%M")

    now = datetime.now()
    N = now.timetuple().tm_yday

    delta = 23.45 * math.sin(math.radians((360 / 365) * (284 + N)))

    lat = 60.36780

    w = (((formatted_hour.hour - 12) * 15) + (15/60 * formatted_hour.minute))
    lat_rad = math.radians(lat)
    delta_rad = math.radians(delta)
    w_rad = math.radians(w)

    cos_zenith = (
        math.cos(lat_rad) * math.cos(delta_rad) * math.cos(w_rad)
        + math.sin(lat_rad) * math.sin(delta_rad)
    )

    zenith = math.acos(cos_zenith)
    zenith_degrees = math.degrees(zenith)
    angles_zenith = 1361 * (1 + 0.033 * math.cos(360 / 365 * N)) * cos_zenith

    return angles_zenith, angles_zenith / 1.96
```

For this part I used the courses I attended throughout the semester. I was lucky to be able to take part in the solar energy lecture every week, which was very interesting for my general knowledge as for the project.

I learnt how to calculate the theoretical radiation arriving outside and inside the atmosphere.

Here is an extract from Professor Andrés Olivares's course.

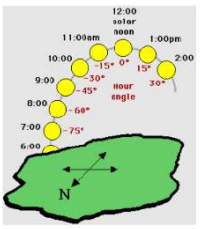
Calculation of the zenith angle

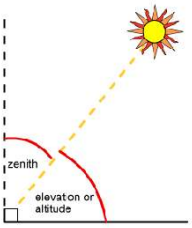
Zenith angle depends on:

- Declination: δ
- Latitude: $\emptyset$
- Hour ange: ω

$$cos(\theta_z) = cos(\emptyset) \times cos(\delta) \times cos\,(\omega) + sin(\emptyset) \times sin(\delta),$$

$$\omega\,(^o) = \frac{time\ i\ klokken - 12}{24\ hours},\ 1\ time\ vinkel = 15°,\ kl\ 13{:}00 = +15°$$





Example Bergen 29.08, kl 11:00

Day number:

$N := 212 + 29 = 241$

Declination:

$delta := 23.45 \cdot \sin\left(\left(\frac{360}{365}\right) \cdot (284 + N)\right) = 8.857$

$delta := 8.857\ deg$

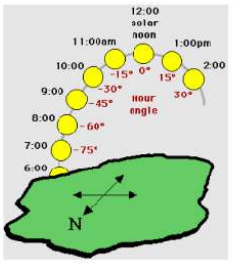
$Lat := 60\ deg$

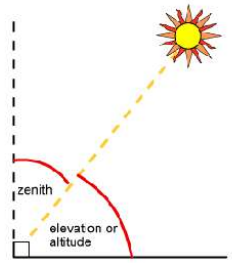
$hour := -15\ deg$

$cos_zenith := \cos(Lat) \cdot \cos(delta) \cdot \cos(hour) + \sin(Lat) \cdot \sin(delta) = 0.611$

$zenith := \operatorname{acos}(cos_zenith) = 52.371\ deg$

$altitude := 90\ deg - zenith = 37.629\ deg$





I have simply transcribed these calculations in python script. That is why at the beginning of the script I imported the maths module.

Solar declination is calculated as a function of the day of the year (N) using an astronomical formula.

The zenith angle is calculated as a function of time, solar declination and latitude. The calculations involve trigonometric operations, including sines, cosines and cosine arcs. In short, this function calculates the zenith angle as a function of time and solar declination, and then converts this angle into solar radiation.

So thanks to these calculations, we can directly compare the theoretical and actual values.

8. Graph design

```
######## 7 - GRAPHICS DESIGN CREATION ########
######## 24h
# Create a figure for the graph
fig, ax = plt.subplots(figsize=(10, 6))
line, = ax.plot([], [], color='red', linestyle='-', linewidth=2, label='Received Irradiance')
line_extra, = ax.plot([], [], color='orange', linestyle='--', linewidth=2, label='Theoretical Irradiance Outside the Atmosphere')
line_clearsky, = ax.plot([], [], color='purple', linestyle='--', linewidth=2, label='Theoretical Irradiance')

# Set the y-axis scale (0 to 1100) with labels every 100 W/m²
ax.set_ylim(0, 1100)
ax.set_yticks(range(0, 1101, 100))

# Add titles and labels to the axes
ax.set_xlabel("Time")
ax.set_ylabel("Irradiance (W/m²)")
ax.grid(True)
ax.set_facecolor('aliceblue')
ax.legend(handles=[line, line_extra, line_clearsky], loc='upper left')

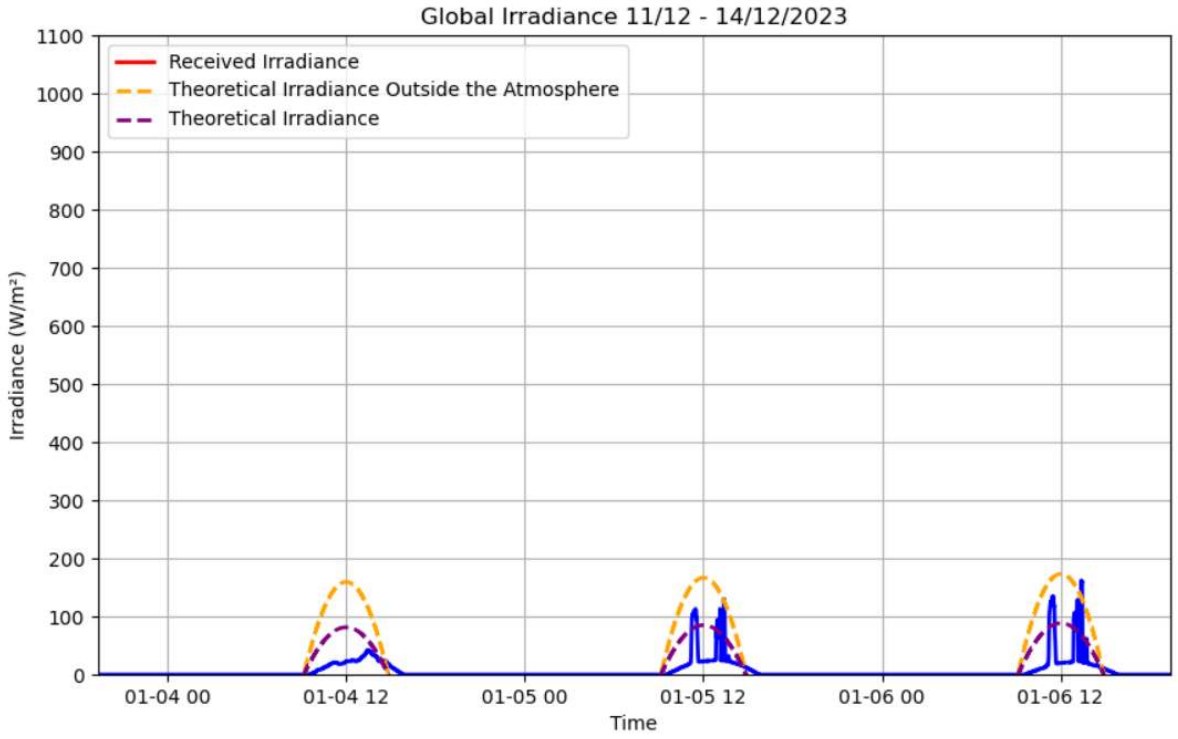
# Create a secondary y-axis for right-side y-axis labels
ax2 = ax.twinx()
ax2.set_ylim(0, 1100)
ax2.set_yticks(range(0, 1101, 100))

```

Here these lines are used to define the design of the graphic. For example, I have given titles to the axes and to each of the curves. I have also sometimes defined the colours and the size of each element. In the last part of this extract, I have created a second y-axis on the right.

I am not going to show it to you so as not to clutter up this report, but I have done exactly the same thing for the other two graphs.

Here is an example of a graph generated by my code:



9. Graph creation

This part is the most complex. It is the part that actually creates the three graphs from the data sent by the pyranometer.

```
######## 8 - GRAPHICS CREATION AND SAVING ########
# Function to save the current graph
def animate(i):
    global list_time, list_radiation, list_time_72hours, list_radiation_72hours, list_time_168hours, list_radiation_168hours
    global list_radiation_extra, list_radiation_extra72, list_radiation_extra168
    global list_radiation_clearsky, list_radiation_clearsky72, list_radiation_clearsky168
    global ax, ax_72hours, ax_168hours

    # Update the real-time graph (24 hours)
    radiation = instrument.read_register(0x0006, functioncode=4)

    # If the radiation value is greater than 2000, replace it with 0
    if radiation > 2000:
        radiation = 0

    # Get the current time
    current_time = datetime.now()
    list_time.append(current_time)
    list_radiation.append(radiation)
```

First I declare the necessary variables.

The function uses several declared lists. As I said earlier, there would have been a way of bringing all this together but I preferred to separate each curve from each graph. So these lists store the solar radiation data for the different time periods, the theoretical data, and the actual data.

radiation = instrument.read_register(0x0006, functioncode=4): This is used to request the reading of register 0x0006 of the pyranometer using code function 4 of the Modbus protocol. I have taken this line from Tom's code.

if radiation > 2000: radiation = 0: If the radiation value is greater than 2000 W/m², it is replaced by 0. I added this line because, although I haven't managed to understand why, sometimes there were peaks at 65000 W/m². This is a value given by the pyranometer when there is a reception error. The problem was that these errors always appeared at night, so I decided to cancel them with this line.

current_time = datetime.now(), list_time.append(current_time), list_radiation.append(radiation): These lines are used to get the current time and make a list of it for titles, for example. I had to put this in the function otherwise I had have date problems.

```
    ############## 24h
    # Calculate the theoretical zenith angle
    zenith_extra, zenith_clearsky = calculate_zenith_angle(current_time.strftime("%H:%M"))
    list_radiation_extra.append(zenith_extra)
    list_radiation_clearsky.append(zenith_clearsky)

    # Update the graph of the last 24 hours
    max_time = current_time
    min_time = current_time - timedelta(hours=24)

    filtered_time = []
    filtered_radiation = []
    filtered_radiation_extra = []
    filtered_radiation_clearsky = []

    for time, rad, rad_extra, rad_clearsky in zip(list_time, list_radiation, list_radiation_extra, list_radiation_clearsky):
        if min_time <= time <= max_time:
            filtered_time.append(time)
            filtered_radiation.append(rad)
            filtered_radiation_extra.append(rad_extra)
            filtered_radiation_clearsky.append(rad_clearsky)

    line.set_data(filtered_time, filtered_radiation)
    line_extra.set_data(filtered_time, filtered_radiation_extra)
    line_clearsky.set_data(filtered_time, filtered_radiation_clearsky)
```

After these list initialisation lines, it is time to calculate the angle of the theoretical zenith:

zenith_theoretical, zenith_theoretical_multiplier = calculate_zenith_angle(current_time.strftime("%H:%M")): Calls the calculate_zenith_angle function (described above) to obtain the angle of the theoretical zenith as a function of the current time. The results are added to the *list_radiation_theoretical* and *list_radiation_theoretical_multiplier* lists.

- Updates the graph for the last 24 hours:

max_time = current_time, min_time = current_time - timedelta(hours=24): Defines a time range of the last 24 hours.

The *for* loop goes through the various lists and filters the data that falls within this range.

The filtered data is stored in new lists beginning with "filtered".

Lines beginning with "line" update the graph data with the filtered data.

```
    # Set a dynamic time range for the x-axis labels (24 hours)
    ax.set_xlim(min_time, max_time)

    current_hour = current_time.strftime("%H:%M")
    ax.set_title(f"Global Irradiance {yesterday_str} - {current_date_str}")
    ax.set_xlabel(f"Time ({current_hour})")

    # Save the data to a CSV file
    data_csv(current_time, radiation)

    # Redisplay the x-axis labels for the even hours of the last 24 hours
    current_hour = current_time.replace(minute=0, second=0, microsecond=0)
    hour_labels = [current_hour - timedelta(hours=i*2) for i in range(13)]  # Even hours
    hour_labels.reverse()  # Reverse the list to match the latest hours
    hour_labels_formatted = [time.strftime("%H:%M") for time in hour_labels]
    ax.set_xticks(hour_labels)
    ax.set_xticklabels(hour_labels_formatted, rotation=45)

    # Add lines on the other hours without displaying the labels
    all_hour_labels = [current_hour - timedelta(hours=i) for i in range(25)]  # All hours
    other_hours = [hour for hour in all_hour_labels if hour not in hour_labels]
    ax.set_xticks(other_hours, minor=True)
```

Next, a step that seems simple but that I had to think about was defining the dynamic time range for the x-axis labels:

ax.set_xlim(min_time, max_time): Defines the time range displayed on the graph's x-axis.

The lines with the titles must be placed in the *for* loop again so that the titles are also updated.

current_hour = current_time.replace(minute=0, second=0, microsecond=0): Resets the hours minutes and seconds to have labels every hour batteries.

*hour_labels = [current_hour - timedelta(hours=i*2) for i in range(13)]:* Creates a list of stacked hours every two hours for the last 24 hours.

hour_labels.reverse(): Inverts the list so that it is ordered from most recent to oldest.

ax.set_xticks(hour_labels): Sets the positions of the x-axis labels.

ax.set_xticklabels(hour_labels_formatted, rotation=45): Rotates the list by 45 degrees to improve readability.

I then do the same thing for the curve for the last 72 hours and the last week. I have decided not to explain these two parts so as not to repeat myself, but it follows exactly the same principle as the last 24 hours.

```
# Create an animation to update the graph in real-time
ani = FuncAnimation(fig, animate, interval=10000)  # The interval is in milliseconds (5 seconds)


# Create an animation to update the graph in real-time (72 hours)
ani_72hours = FuncAnimation(fig_72hours, animate, interval=10000)  # The interval is in milliseconds (10 seconds)



# Display the graph for the last 24 hours in real-time
plt.show()


# Close the Modbus serial port at the end
instrument.serial.close()
```

Now the last part of my code:

ani = FuncAnimation(fig, animate, interval=10000) Creates an animation that updates the graph by calling the animate function at regular intervals defined by the interval argument (in milliseconds). In this case, the interval is 10 seconds.

plt.show(): Displays the graphs

instrument.serial.close(): Closes the Modbus serial port at the end of script execution. As the script runs continuously it never stops.

10. Area for improvement

This part is more a self-assessment of my work than a real need for improvement, so it is up to the next student to improve the code or not.

In writing my report, and with a bit of hindsight, I realised that there were a number of points that I could improve in this code, which is far from perfect.

- Firstly, there are many lists. I have already mentioned why I decided to separate all these lists, but if I had known from the start that there would be three different graphs, each with three curves, I would have grouped some of the data together to write more concise and user-friendly code.

- With regard to the curves, I think I could have taken data every 2 minutes and then made an average of the 5 last data to display one every ten minutes. This would have smoothed the curve, which sometimes has very sharp shapes. It would also have avoided having a programme that displays curves that update every two minutes, which can be a lot for the precision that is desired.

- I would also have been careful about duplicates and unnecessary lines. Whether for the definition of times or lists, I wrote the same lines in the function that creates the curves as in the first part of the code. I think that some of these lines are completely obsolete. By modifying the code little by little I didn't realise these duplicates. Especially as it is easy to change.

- Finally, something I should have done from the start, and I don't know why I didn't think about it before, is the fact that when I run the program, the curves start again at 0. I should have made a code that directly takes the data from the last 24 hours and displays it. Instead, the program runs and

displays the data from that moment on. This is a major flaw because each time you modify the code, when you run it again, you don't see the curves straight away and sometimes you have to wait an hour to see the lines take shape and know whether what you changed is working properly.

V. SolarLab website

The aim of creating graphs with the sun's radiation in real time is to be able to display them at any time on any computer. You can work on them, see them out of curiosity, and show them in class...

For this Tom had thought of several solutions such as using putty and the IP address, which was far too complex for me, especially as you have to be connected to the same network. He was also thinking of using apache software or a program called Xming to record data and communicate with an IRL address. All these solutions were not ergonomic enough. Something much simpler was needed. I immediately thought of creating a website. Thanks to a website, we could see these curves anywhere and at any time.

1. Website via the Raspberry IP (abandoned solution)

At first I wanted to avoid creating a website by buying a domain, knowing that I am not a professional in this field, I thought it was probably possible to create a small website myself.

Nginx

I was looking for ways to do this on forums and came across Nginx.

Nginx is a very powerful open-source web server, often used to host websites, web applications and act as a reverse proxy server. It is famous for its ability to handle a large number of simultaneous connections with low use of system resources.

Thanks to its event-driven architecture, it works at streaming data in real time, which can be particularly useful in scenarios such our graph display.

To summarise, the aim is to create a configuration file in a Nginx folder specifying the path to the directory containing the HTML files and the IP address to which Nginx should listen.

When Nginx is running, it then uses this IP address to create the website described by the html code.

So I used my html writing skills to create my website. I had created several pages and a simple, effective design. This part took up a lot of my time because I wanted to do everything by writing the code myself to have a professional and ergonomic site. So I also wrote some css code for the design of my page. After a few days of struggling with my code, I used a web page editor to get better html and css code for the same result.

Unfortunately, it was only after two weeks that I realized that this solution was clearly not the best one, for a number of reasons. Firstly, when I wanted to test this website with some of my friends, some of them were able to access it but not all, there was a loading problem that I could not explain. In addition, you had to be connected to eduroam (the same connection network as the raspberry) to access the website, which was not very practical either. Finally, the biggest problem is that this type of website is not at all secure. Even if I had managed to create a website using this solution, I had to give up because there were already too many problems and this solution was not permanent.

So I wasted a lot of time on this part, even if it did improve my coding skills.

2. Buying a domain

- After discussion with Mr. Grant, the best solution was to buy a domain and create a real website.

We had to be careful because there are website editors such as NicePage, which I had used for the css code previously, which only offer so-called static sites. In other words, you can't display data that is continuously updated as we would like for our graphics. Having a website hosted on a dynamic server was essential.

- As it happens, Richard was already using Ionos for his personal use and was quite happy with the website up until then.

After looking at the reviews and possibilities on Ionos, I saw that it clearly met our expectations. We could choose a name and create a website. It seemed to have all the options to display our curves without any problems.

This was followed by a few days of budget requests, because a domain like this one is not free.

- Then we thought long and hard about the name of our website, and with what was still available, we opted for solarlab.online.

We still had to choose the 'pack' we wanted to buy from Ionos. As I said earlier, we needed to be sure of two things: that we could send images to the server (the images of our graphics) and that the server could be updated continuously.

Once we bought it, I was able to make all the settings so that the server, the FTP protocol (I will come back to this in the next part) was operational.

Finally, I had to choose which software I was going to use to design the new website. The two best known are Wordpress and Joomla. I knew that I could do everything I wanted with Joomla and it seemed easier on Ionos to install and use Joomla, which is simply why I chose it.

3. Joomla

Joomla is a content management system for creating and managing dynamic websites. There are multiple options and it can be complex to get to grips with when you have never seen this software before.

I have created document to help you understand how it works if you ever want to add or change things on the website.

SP Page Builder

On Joomla you can add extenders to extend the functionality of the website. So I started by adding SP Page Builder.

SP Page Builder is a website editor that greatly simplifies the page creation process. With an ergonomic approach this extension allowed me to design fairly complex page layouts without the need for programming.

However, in all cases I had the code I had written earlier as a back-up, but it was still easier to use SP Page Builder.

You can also have an overview of SP page builder and Joomla in the appendix 4.

Again, don't hesitate to ask me about the SP Page builder features, sometimes creating a new page with a logo and banner can be harder than expected.

This part also took me a long time to get the best possible site. Right up to the last day, I was able to improve certain things with Richard's help. I created several pages, personalised them as much as possible with a banner, a menu, logos...

I invite you to have a look at the site on solarlab.online, or you'll find a screenshot in appendix 5.

Extensions R2H

The R2H extension isn't very complicated. In fact, by installing this on Joomla, I can simply ask the website to update itself whenever I want. Knowing that data arrives every two minutes, I asked the site to update itself every two minutes too. Thanks to this, the website visitor doesn't have to click on the button himself.

4. Transmission of files from Raspberry to the server

Now that the website had been created, the hardest but most interesting part was to send the images of the graphics from the Raspberry to the server so that they could be displayed on the website. As you saw in the part where I explained the code. Every two minutes the three graphics are saved under the same name, overwriting the previous one each time.

The aim of this is to send these three files to the server, also every two minutes, so that the server receives the latest versions of the graphs.

An FTP protocol enables users to send files from their computer to a server. FTP, or File Transfer Protocol, is a way of moving files between a local computer and a remote server. It is particularly useful for managing websites, backing up files and other situations requiring the transfer of data between a client and a server.

5. SCP

My first thought was to use FileZilla, which I knew. I started to configure with this FTP but in the end I learned that on Linux it was easier to use SCP (Secure Copy Protocol) with a script because there was no need for a graphical interface and in a few lines of code the file transfer I wanted was solved.

In addition, SCP is directly integrated into Linux, so I didn't need to install FileZilla on the Raspberry.

Finally, even though FTP now offers secure versions with FTPS, the SCP protocol is more secure because it is based on SSH (Secure Shell).

Below is the script I wrote for this file transfer:

```
#!/bin/bash

#Chemin des fichiers générés sur votre Raspberry Pi
fichier_source1="/home/tom/Desktop/Projet_Web/Frames/frame_last_graph.png"
fichier_source2="/home/tom/Desktop/Projet_Web/Frames/3days.png"
fichier_source3="/home/tom/Desktop/Projet_Web/Frames/1week.png"

# Informations de connexion à votre serveur Joomla
adresse_ip="access980679515.webspace-data.io"
nom_utilisateur="u113983178"
mot_de_passe="!SolarLaboratory23!hvlbergen"

# Chemin de destination sur le serveur Joomla
dossier_destination="/clickandbuilds/SolarLab/images/"

# Utilisez sftp pour transférer les fichiers
expect << EOF
spawn sftp "${nom_utilisateur}@${adresse_ip}:${dossier_destination}"
expect "password:"
send "${mot_de_passe}\r"
expect "sftp>"

# Transférer le premier fichier
send "put ${fichier_source1} ${dossier_destination}\r"
expect "sftp>"

# Transférer le deuxième fichier
send "put ${fichier_source2} ${dossier_destination}\r"
expect "sftp>"

# Transférer le troisième fichier
send "put ${fichier_source3} ${dossier_destination}\r"
expect "sftp>"

send "exit\r"
expect eof
EOF
```

To do this, I used the server connection information I could get from Ionos.

"*Expect*" is a scripting language and program used to automate interactions with command-line programs. Here to connect to the remote server.

6. Automatization with Cron task

Now that I have the code to send the images to the server, I need to automate it. To do this I'm using Expect, which allows me to run the SCP code every two minutes and send the images every two minutes.

Cron tasks, abbreviated to "cron", are utilities found on Linux. They enable tasks to be scheduled to run automatically at regular intervals. This means that nobody has to run SCP code every two minutes. They are configured in a file called crontab.

All you have to do is open crontab and write the following line:

**/2 * * * * path/to/your/script*

VI. Various problems

1. Wi-Fi connection

I had a lot of problems with the wifi connection in the tiny house. The house is at the far end of the campus and although there is a wifi source not far from the tiny house, the signal inside is very weak.

I am not going to detail all the steps to get the best possible connection, for that you need to see the WIFI settings document that I created.

However, here are a few steps or method that I have taken:

- Connect to Enduroam using the wifi software on Linux wpa_supplicant. Here is an example of the code you need to write on Supplicant on Linux to connect to the wifi network:

ctrl_interface=/var/run/wpa_supplicant GROUP=netdev

update_config=1

country=no #no means norway

network={

ssid="eduroam"

eap=PEAP

key_mgmt=WPA-EAP

phase2="auth=MSCHAPV2"

identity=studentnumber@stud.hvl.no #for example 676065@stud.hvl.no

password="Yourpassword" }

- Connection to the laptop via hotspot when I could not connect to the university wifi.

- Installation of the Asus AC56 antenna to receive the enduroam wifi with better quality and power.

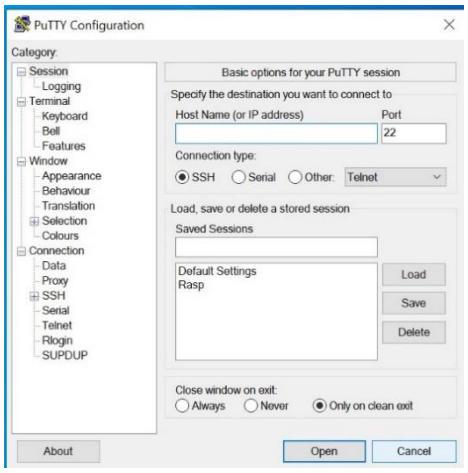
As with everything else in my project, although this last step seemed easy, I had to do a lot of updating and it took me several tries to get this antenna connected and working on the raspberry.

You'll find a document in my work directory specifically for connecting to the wifi.

2. Connection between laptop and raspberry

It is not very practical to work directly on the raspberry with Linux. Firstly because I am not used to this interface but also because it is rather slow, for example when using the Internet. So it is much more practical to connect to it remotely on another computer. This allows you to copy and paste parts of code or other documents from one computer to another. This saves a lot of time.

Putty (abandoned solution)



The last student used Putty to do this. I found this software interesting but not sufficient for my purposes. It is not the easiest to get to grips with, although it is not complicated either. And above all, you can only control one Raspberry window, whereas I wanted to use the whole Raspberry.

VNC Viewer



I immediately turned to VNC Viewer because I found it much easier. All I needed was for the Secure Shell protocol (SSH) connection to be enabled on the raspberry. Then, using the raspberry's IP address, you can

simply connect by entering the username and password of the computer you want to control. However, both computers need to be connected to the same wifi, which sometimes wasn't possible with my connection problems. Nevertheless, I used this solution throughout the semester.

In general, don't hesitate to ask me if there is a problem with the wifi or anything else. I might be able to find a solution.

VII. Data analysis

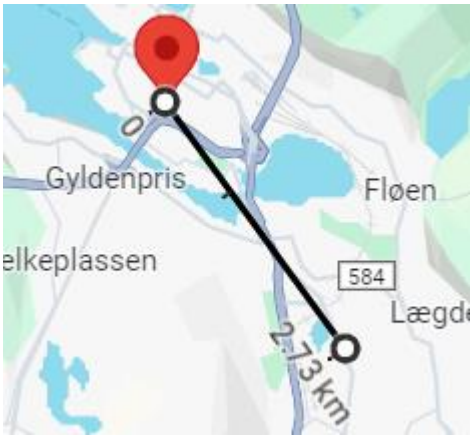
1. Context

In this section, I am going to analyse the data I have been able to collect.

Indeed, the aim of this project is also to understand the impact of the sun's radiation on solar panels and energy production. Unfortunately, with the problem of battery delivery, we were unable to get the tiny house system up and running. However, thanks to the pyranometer and the code I created, the sun's radiation has been recorded for many days and is currently being recorded.

In this analysis, I will compare the data with that of the University of Bergen (UIB). This university has a meteorological centre, and on their website you can download various data, including irradiance from their pyranometer.

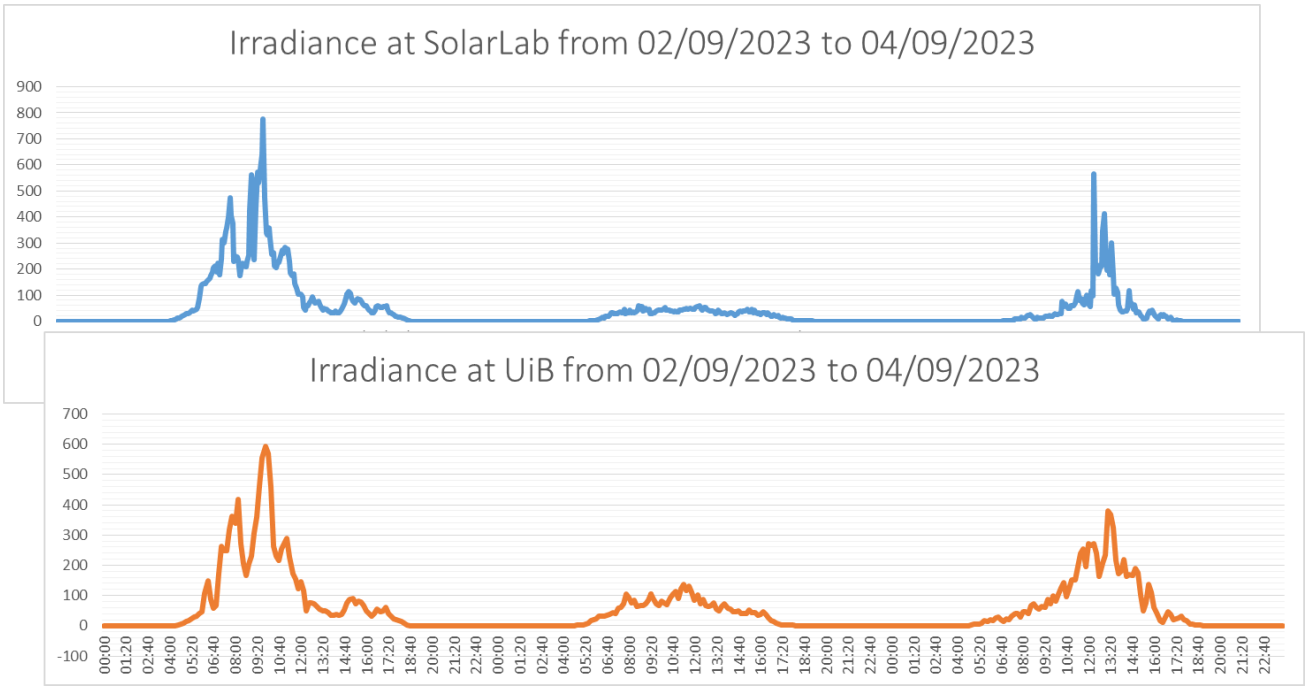
The purpose of this small study is to compare the difference in irradiance according to location within the same city. This will demonstrate the importance of choosing the right location for solar panels, for example.

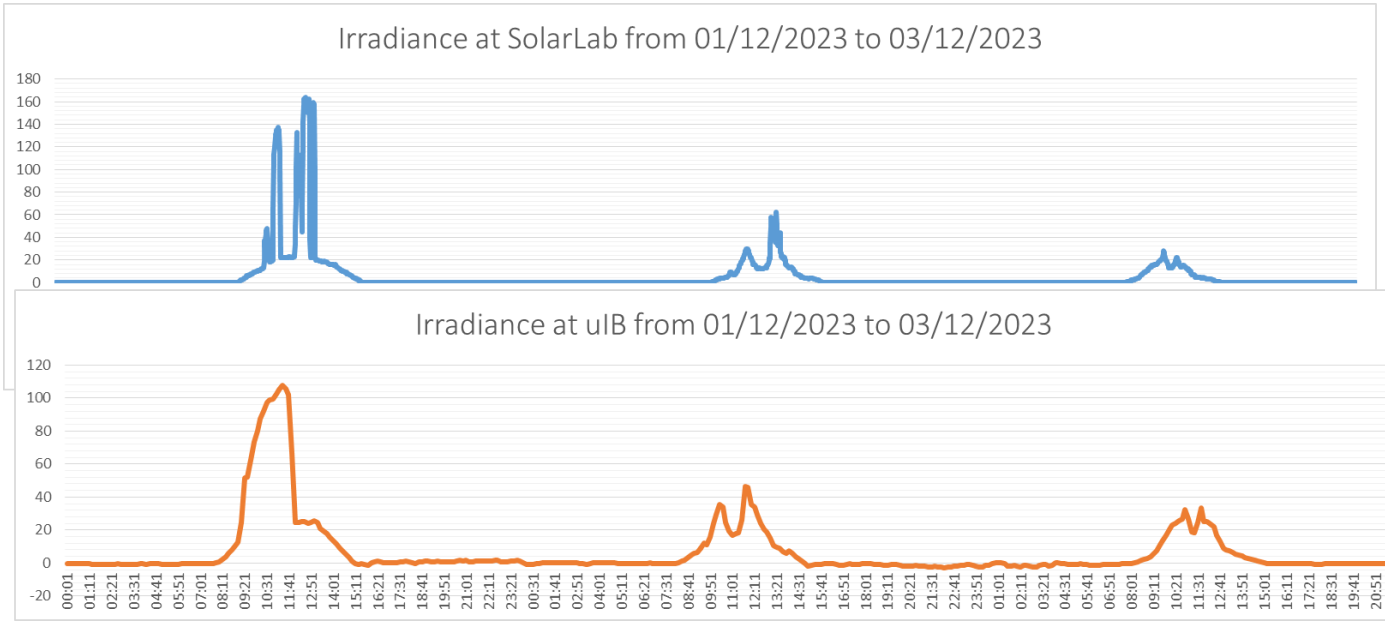


A map extract showing the distance between the two pyranometers, with UiB at top left (Florida) and SolarLab at bottom right (Kronstad).

To do this, I chose two different periods, the first days of September and the first days of December. Thanks to this, I will also show the differences according to the seasons.

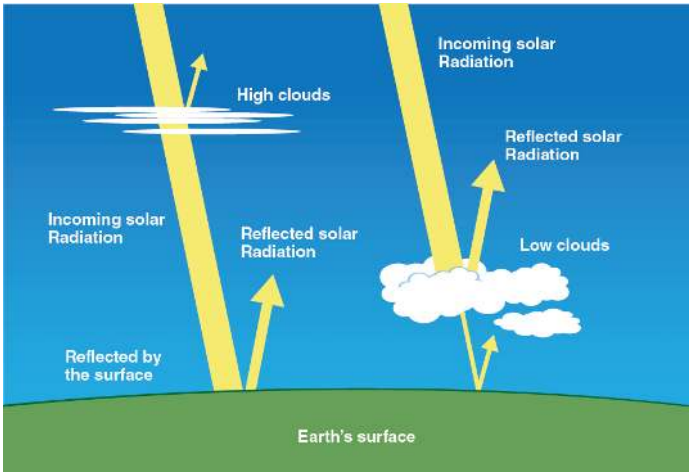
2. Analysis





The graphs are offset so that the curves can be aligned for comparison.

There are a number of reasons for the differences in these graphs. Firstly, a simple reason for the small differences is simply the clouds. As shown in the figure below, clouds block the passage of the sun's rays, and we all know that a few hundred meters can take you from shadow to light. As it happens, our two pyranometers are 2.73 km apart. So we can say that this is enough to make a difference on sunny but cloudy days.



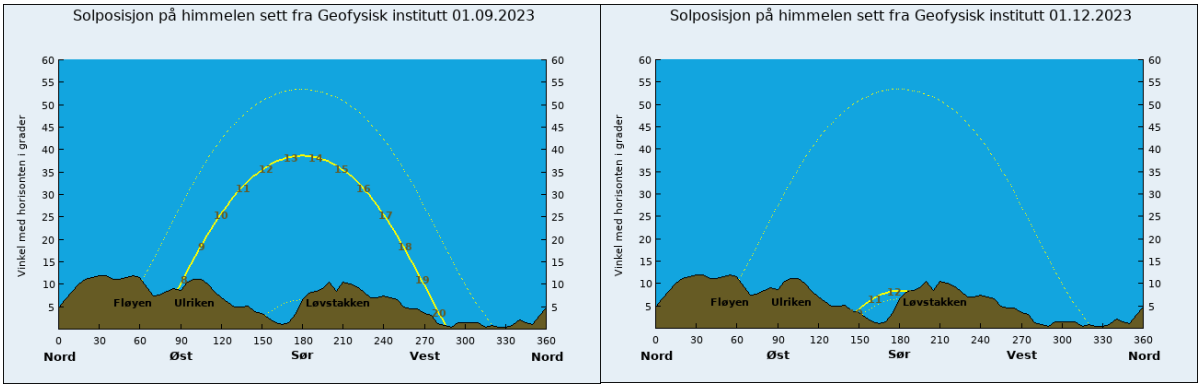
A second difference which is not insignificant if we want to compare these curves and the precision of the devices. Indeed, different devices can be more or less efficient and accurate at receiving the sun's rays. What is more, these devices don't send data at the same time frequency. So the UiB curve seems much smoother. Certain small peaks found on the SolarLab curves may therefore be smoothed out and absent from the UiB graph.

The various reasons already mentioned explain the fluctuation during the day, but real differences can be observed, for example at sunrise and sunset.

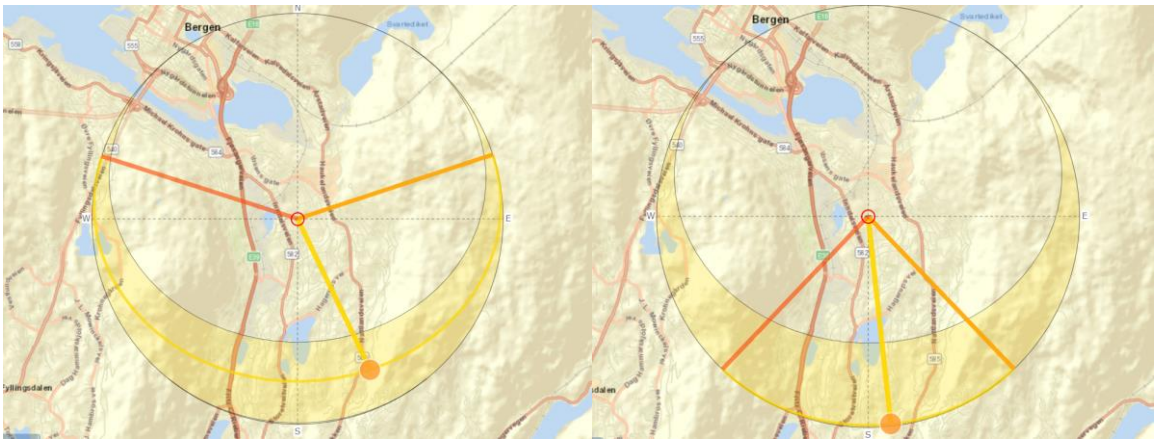
The graphs on the previous page show that sunrise appears earlier in Florida (UiB neighbourhood) than in Kronstad (SolarLab neighbourhood). This is due to the position of these sites in relation to the surrounding environment. Bergen is surrounded by seven mountains that have a great impact on the reception of the sun's rays. In fact, these mountains will obstruct the sun's rays at the beginning and end of the day.

The UiB Meteorological Centre have created a graph showing the sun's path according to the time of day and date. The graph is very interesting, and makes us realize that the sun is hidden by these mountains for many hours, especially in winter.

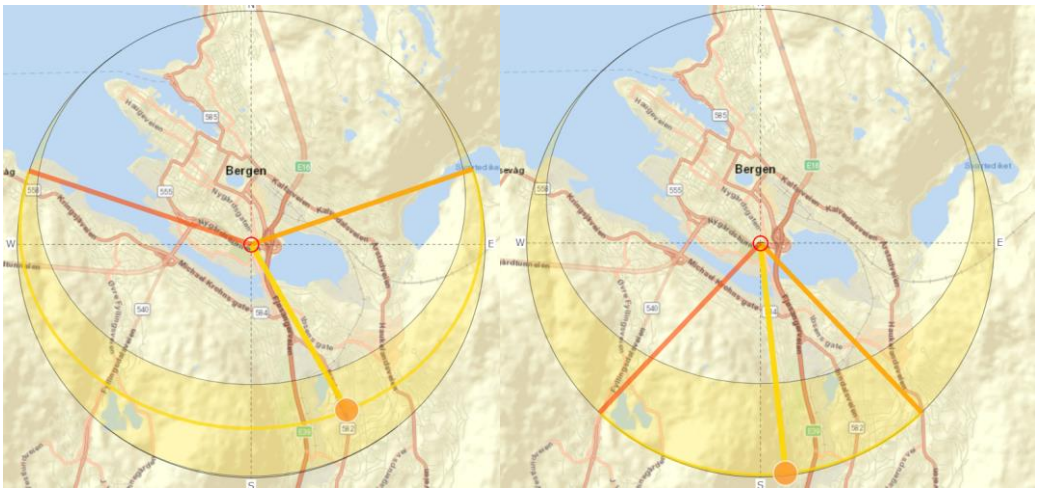
For example, at the first peak of the curve in December in Florida, the curve drops off completely at around 1pm. It turns out that this drop in radiation occurred every day. If we look at the image below, we can see that at around 1pm the sun goes behind the mountains, so we have the explanation for this drop.



These graphics are rather difficult to create, as they require a topographical map of the surrounding area and the transition from 3D to 2D is not easy to do. Unfortunately, I wasn't able to recreate this with SolarLab. However, there is a site that lets you see the sun's trajectory without taking the environment into account.



Sun trajectory at SolarLab in summer (a) and winter (b)



Sun trajectory at UiB in summer (c) and winter (d)

Figure (*a)* and (*c)* show a clear difference between sunshine at Florida and Kronstad. In Kronstad, for both sunrise and sunset, there are mountains between the sun's position and the pyranometer (dark orange and

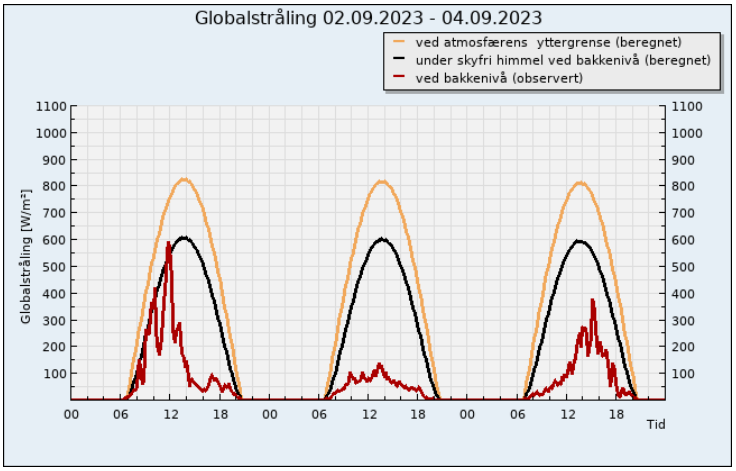
light orange lines). In Florida, however, the sunset is almost as long as the sun itself (which confirms the previous image), so solar panels positioned at this point would produce more electricity.

In winter, the difference is less obvious, between figures (*b)* and (*d)* as both locations are hindered by the mountains, especially as the sun is particularly low at this time of year. This is reflected in our graphs on the page 30. In summer, there is a clear difference in sunrise between Kronstad and Florida, much less so in winter.

Thanks to Excel, I was able to calculate the difference in the areas under the curves. As we have hours on the abscissa, the calculation is not perfect. To do this I used an approximate trapezoidal method. I then came across a rather abstract value due to the abscissa times. But I was still able to compare the two values for our two locations and I could see that in summer the pyranometer in Florida received 3% more radiation than in Kronstad.

3. Differences between theory and data

As explained in the code explanation section of my report, it is possible to calculate the theoretical radiation at a given point. There are two reasons for the differences. The first point that explains the difference between the orange and black curves below is the absorption of light rays by the atmosphere. Light rays are more powerful outside the atmosphere. The theoretical values that can be obtained at the earth's surface are therefore given by the black curve. This varies according to the time and date and therefore the position of the sun.



The real curve does not follow the theoretical curve because of clouds, natural obstacles such as mountains and other obstacles such as buildings.

Sometimes the red curve can pass over the black curve. This can be explained by an approximation in the theoretical calculations but also, and above all, by reflections. Buildings, windows or other objects can reflect the sun's rays onto the device, which will then record higher values than normal.

VIII. Conclusion

For my final year in my school, I knew I had to take the opportunity to do my scientific memoir in another country. For me, working in a new university is the best way to develop my knowledge, expertise and interpersonal skills.

After thinking long and hard about the different projects and destinations that Icam offered me, I finally chose to go to Høgskulen på Vestlandet, in Bergen. The university seemed to me to be of a good standard and the project interesting.

When I arrived and became aware of the project, I was very interested in this tiny house. In my daily life, I try to have the best possible impact on the environment, so the goal of making this house off grid was a motivating project.

Unfortunately, I soon realized that my work on this project would involve programming, setting up electronic devices and installing electrical systems. I have to admit that I did not expect this when I chose this memoir. An error in the information provided by Icam meant that I had to work on a subject that I had not really expected. But even though it was not the field in which I have the most expertise, I immediately thought that it was a good opportunity to improve my knowledge in an area that was different from the ordinary.

And as expected, it was really more interesting than I thought it would be. To be honest, it was sometimes quite difficult for me to find the solution to whatever problem I was facing. Indeed, I am not the best at programming, it is something I am not used to doing and basic problems became directly difficult to solve. But with a lot of patience and self-sacrifice, I managed to make a lot of progress. Today, I am proud of the site I have created, especially with the graphics sent by the raspberry. Although it may seem like an easy task, it was not for me.

Finally, the highlight is the battery, which unfortunately never arrived. I would have loved to install it, make all the connections and start up the system. I would also have liked to do more calculations to compare theory and practice.

Turning now to the semester in general, I have to say that I really enjoyed it. Since my office was in the tiny house, it is true that I was often alone. I did not work in teams or anything like that. But I am used to it, it is not a problem. I think I am a really independent person who knows how to organize his own schedule. Maybe for someone else it might be more difficult.

I really appreciate that my manager lets me organize my days. Mr Grant was interested in results and the progress of the project, which was enough for me. I liked that trust, I didn't have any pressure on my work and I think it is better that way.

I also enjoyed the lectures Mr. Grant asked me to attend. I was interested in my project, but also in my knowledge in general. I know this was something the previous student lacked, so I am glad to have had this opportunity. I also enjoyed meeting other people and having a different day to everyone else.

Finally, I would like to say a few words about my experience in Norway, even though it had nothing to do with the project, I was able to meet some extraordinary people and see some incredible landscapes. My life in Norway was perfect and I already know I will miss it. Thanks to Mr. Grant for giving me a few days off to enjoy this beautiful country.

IX. Sources

SolarLab: https://solarlab.online/

HVL: https://www.hvl.no/

Tom Bernard's Report

Andrés Olivares lectures

Website:

Joomla: https://forum.joomla.fr/

Analysis part:

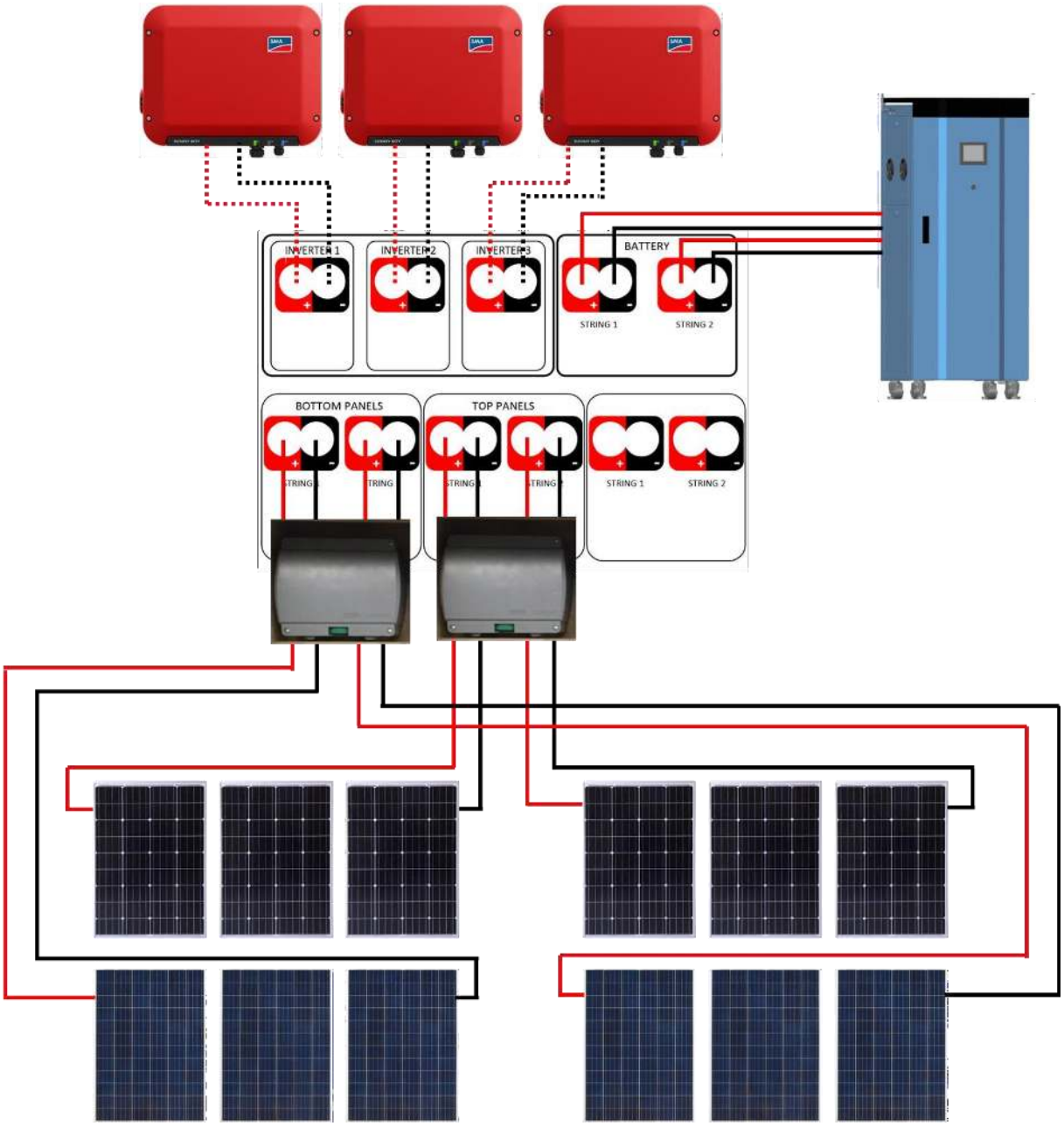
Suncalc: https://www.suncalc.org/#/27.6936,-97.5195,3/2024.01.11/21:19/1/3

UiB website: https://www.uib.no/en/rg/meten/57124/radiation-observatory

SunPath 3D: https://andrewmarsh.com/apps/staging/sunpath3d.html

X. Appendices

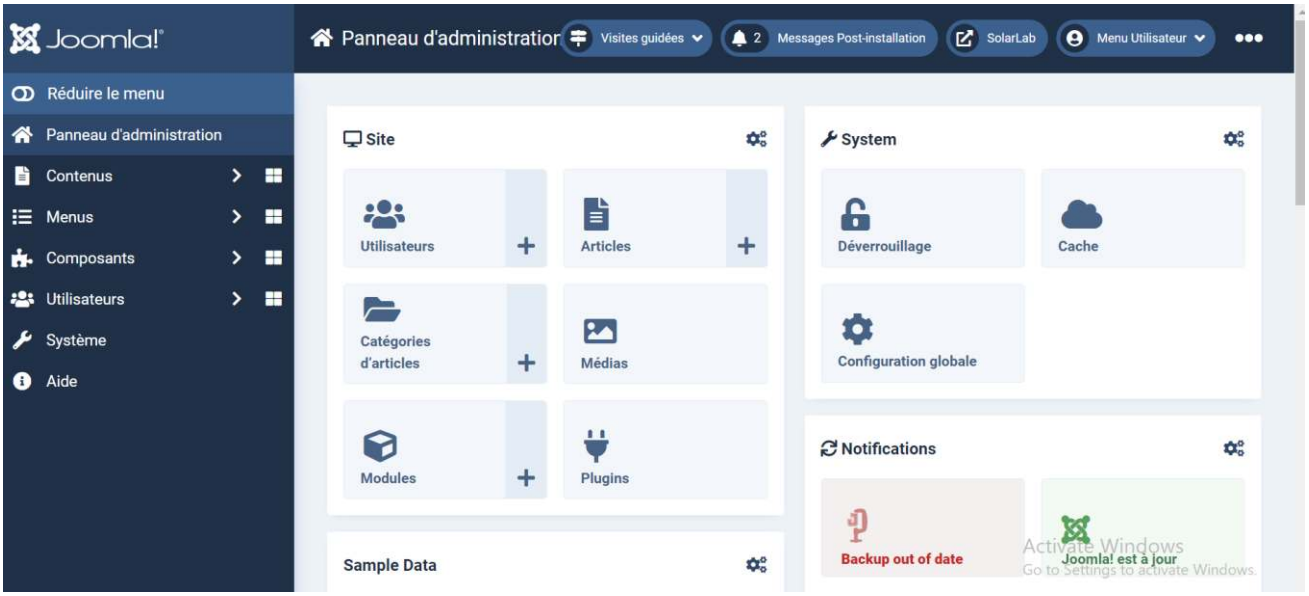
1. System of the tiny house

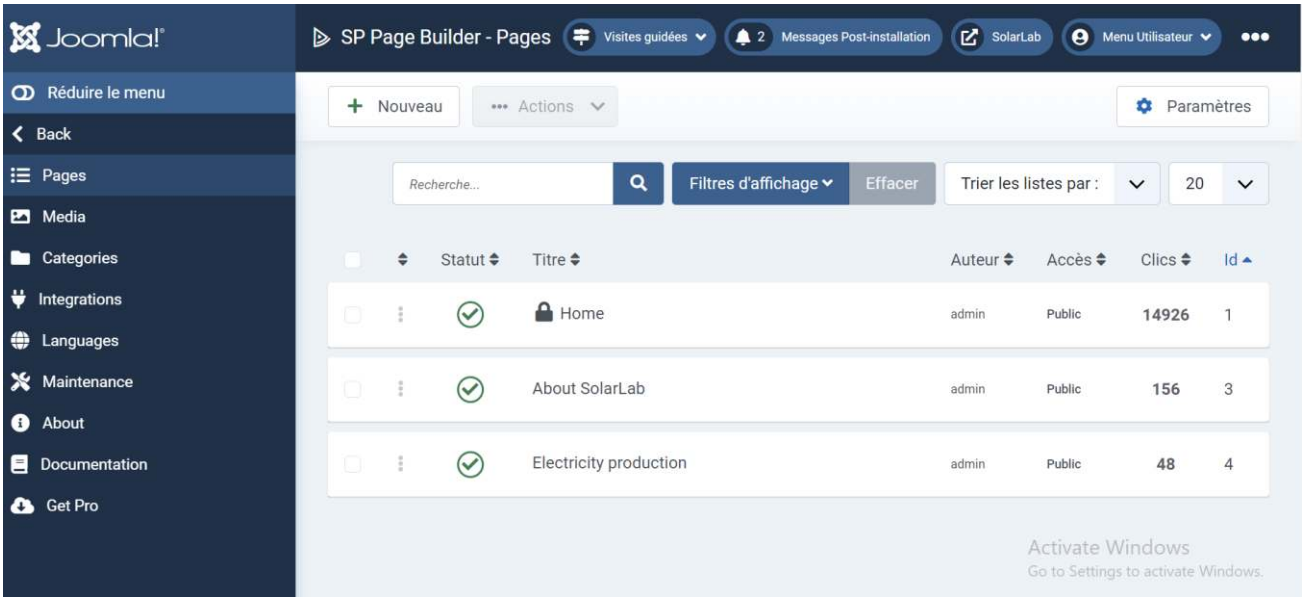


Tesvolt TS25 14,4 Battery - List of distributors

Country	Distributors	Energy (kWh	Price (€)	Price ((NOK)	All included	Comments	Souces	Reviews	Contact	Quotation or answer TOM	Quotation or answer Alexis
France	***Perma batteries***	14,4	14950	168,871	no VAT		https://perma-batteries.com/en/product/tesvolt-ts-	4,9/5 (86 reviews)	contact@perma-batteries.com (Julien Allera)	Tom : yes	Email sent
Spain	***Suministros del sol***	14,4	?	?		ASK FOR THE PRICE	https://suministrosdelsol.com/en/baterries-tesvolt/644-	4,5/5 (66 reviews) 3.7/5 (5 reviews)		Tom : Not available at this moment	Email sent
Bulgary	***Nencom***	14,4	14279	166			https://en.nencom.com/products/tesvolt	5/5 (14 reviews) 4,5/5 (10 reviews)	1@nencom.com	Tom : No answer	Email sent
Uk	***Proinso***	14,4	?	?			https://www.proinso.net/tesvolt-lithium-ion-batteries/	Not on Trustpilot		Not sent	Email sent
Norway	***Getek***	14,4	?	?			https://getek.no/tesvolt/	5/5 (2 reviews) O review on Trustpilot	morten@getek.no	Tom : Waiting for quotation	Email sent
France	***Capenergie***	14,4	?	?		ASK FOR THE PRICE	http://www.capenergie.fr/catalogue/photovoltaique/aut	5/5 (5 reviews) Not on Trustpilot		Not sent	Email sent
South Africa	***PerpetualSolar***						https://perpetualsolar.co.za/product/tesvolt-lithium-	Not on Trustpilot		Not sent	

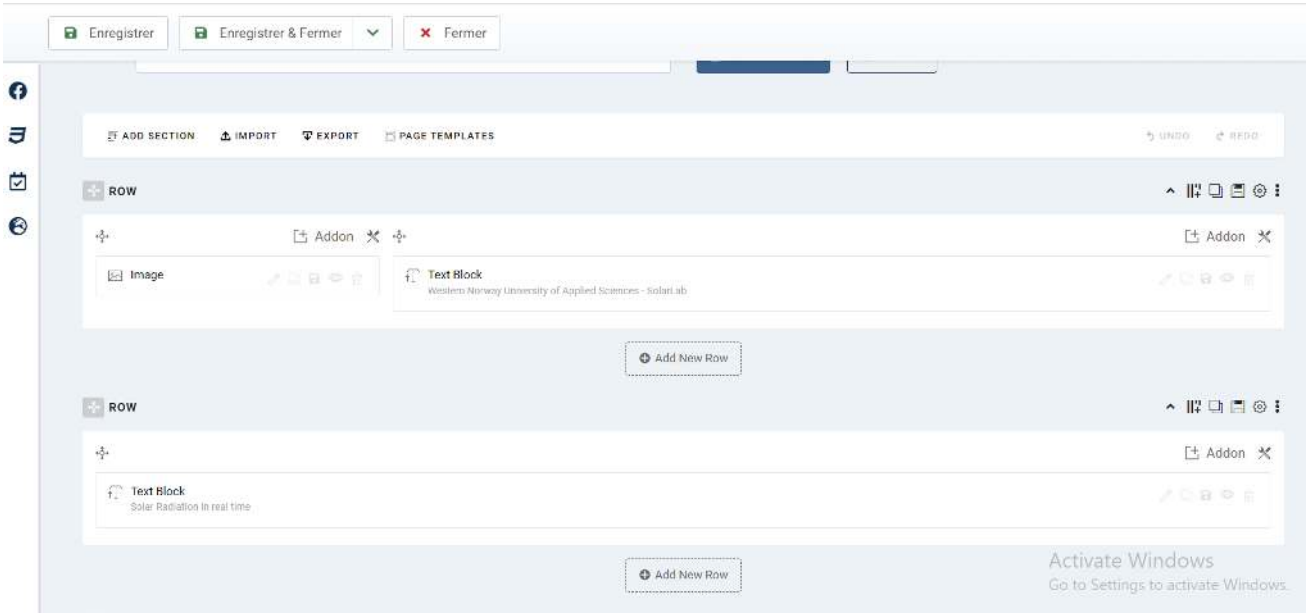
3. Screenshot of Joomla

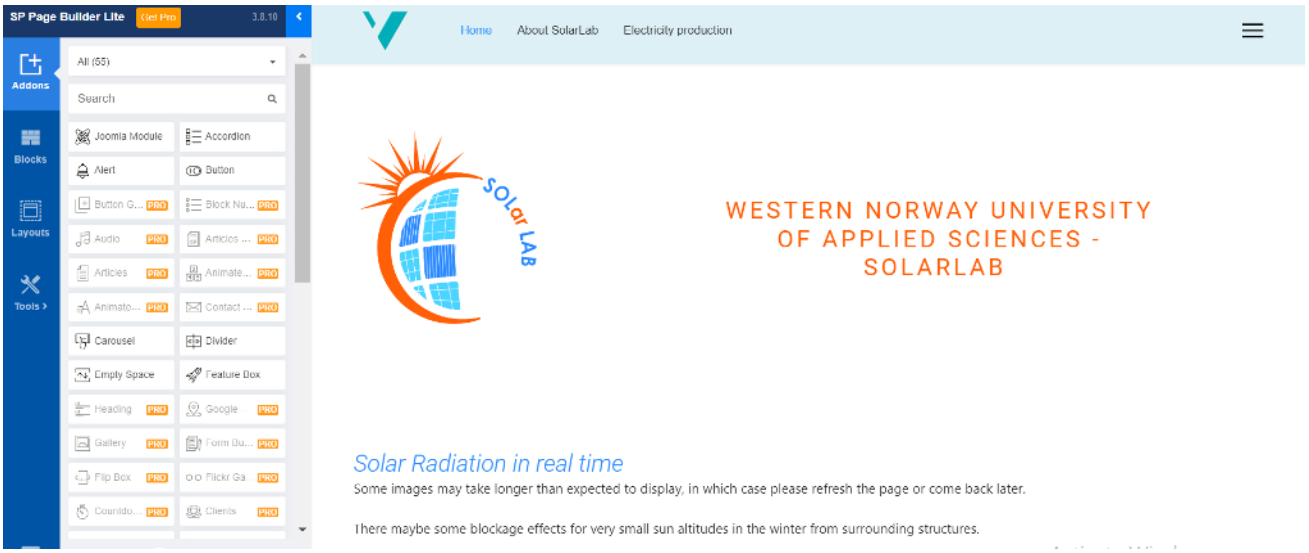




4. Screenshot of SP Page Builder







5. Screenshot of Solarlab.online

