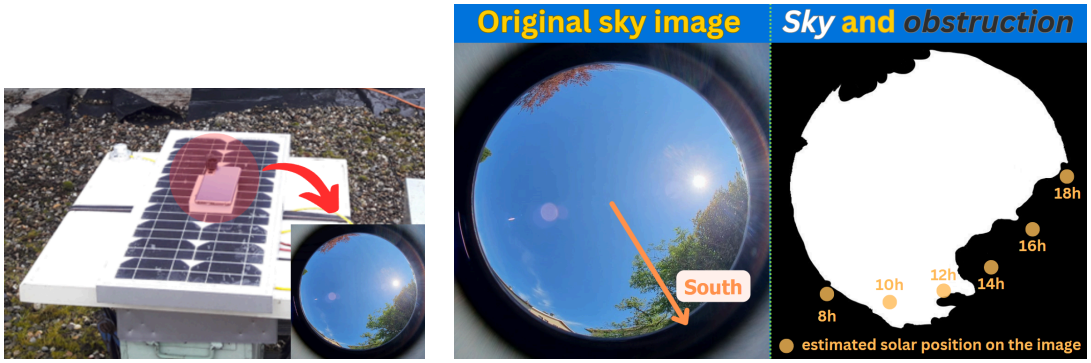
Fisheye Solar Shading Estimator

Updated tutorial and reference guide (2025)

This guide explains how to turn one or more fisheye photographs into a quantitative estimate of the direct solar energy your project can expect, and how to check if your panel, power stage, and battery will sustain your device without outages. It also shows how to prepare your system inputs and read the results.

---

1) How it works



Figures — A fisheye sky photo from the panel's point of view and the sky/obstacle segmentation mask visualization with the sun trajectory.

From the exact place and attitude where your solar panel will sit, you take one or more fisheye photos of the sky. The program calibrates the camera and **automatically** segments each photo into open sky and obstacles using its built-in model.

For the dates you choose, the program **traces the sun's path** across the sky and overlaps that path with your sky/obstacle map. From this it computes a **shading factor** that tells how much of the sun is blocked at each moment.

The incoming energy is then estimated by applying this shading factor to the **direct** and **diffuse** parts of the solar irradiance (they're handled slightly differently under the hood). With your **panel** and **battery** specifications, power-path assumptions, and your chosen consumption profile, the program advances the energy balance over time and reports the **state of charge (SoC)** plus a verdict relative to your **reserve threshold** (your chosen minimum SoC).

*Note: if you use multiple sky photos, they must share the **same orientation and tilt**; they represent different points of view across the panel area to capture partial shading effects.*

2) What you need

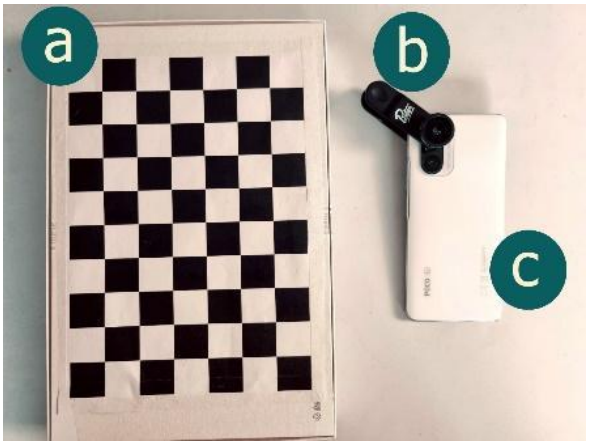


Figure — The tools you need.

You'll need a printed checkerboard (a), a clip-on fisheye lens (b), a smartphone (c) with a compass app set to true heading and a bubble-level app, and a computer to run the program.

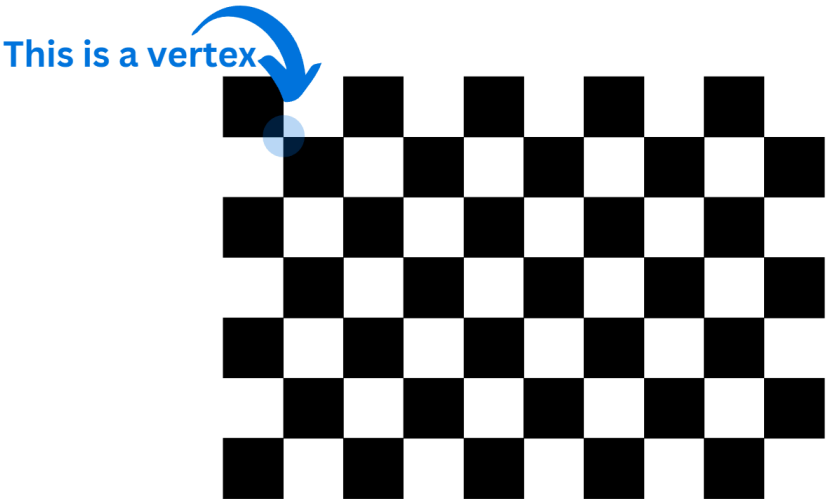


Figure — Explaining what a vertex is.

Calibration pattern (what to print and why).

Print the checkerboard pattern (preferably with **6 inner vertices on the short side and 9 on the long side**). What matters are the **vertices**: the corner points where four squares meet, not the count of squares. Measure the physical **square size in millimeters** on your print and keep that number; the program needs it during calibration. The figure above illustrates what "vertex" means.

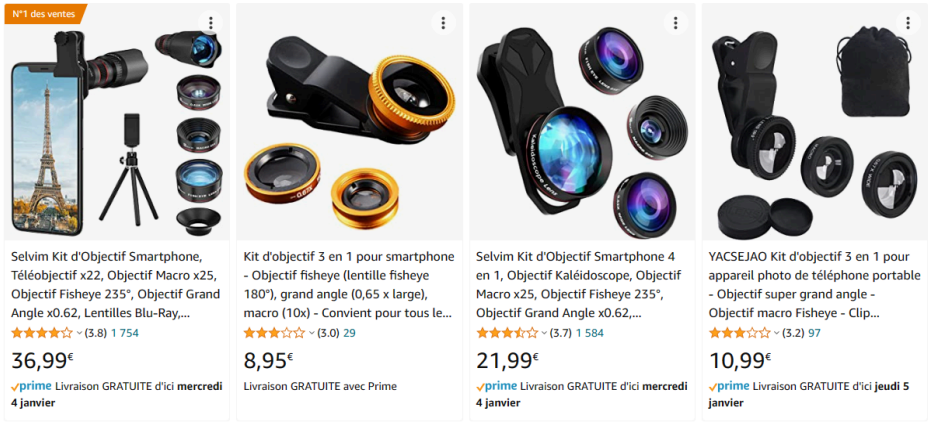


Figure — Pricing of some fisheye lenses.

Clip-on fisheye lenses (cheap and good enough).

Low-cost clip-on fisheyes are widely available; the point is to capture the whole sky dome, not to achieve optical perfection. Expect hobby-grade products and optimistic FOV claims, so buy from a place with easy returns. The PDF shows typical pricing ranges and the kind of product you're looking for (~20€). See above Figure.

MAIN CAMERA	Triple	12 MP, f/1.8, 26mm (wide), 1/1.76", 1.8µm, Dual Pixel PDAF, OIS
		64 MP, f/2.0, 29mm (telephoto), 1/1.72", 0.8µm, PDAF, OIS, 1.1x optical zoom, 3x hybrid zoom
		12 MP, f/2.2, 13mm, 120˚ (ultrawide), 1/2.55" 1.4µm, Super Steady video
	Features	LED flash, auto-HDR, panorama
	Video	8K@24fps, 4K@30/60fps, 1080p@30/60/240fps, 720p@960fps, HDR10+, stereo sound rec., gyro-EIS

Figure — Camera specifications found on GSMArena.

Which phone camera to use (and how to check OIS).
Use the ultrawide camera if you can, but only if that module **doesn't have Optical Image Stabilization (OIS)**. OIS slightly moves lens/sensor elements and breaks the fixed geometry we need for calibration. To check your phone:

- Look up your model on a reliable spec site (e.g., GSMArena) and read the camera table. Identify which modules list **OIS** and which **do not**. Pick a **non-OIS** module. In the figure, you can see that the wide camera has OIS, but the ultrawide doesn't, so you may use the ultrawide camera.

Gather the few **system numbers** you'll enter: battery **capacity (Ah)** and **nominal voltage (V)**; panel **nominal power (W)**; converter/charger **efficiency** (use the product's stated value or, if absent, look up the converter's **IC** datasheet and choose a realistic efficiency at your output voltage and typical load); battery **charge** and **discharge** efficiencies (for lightly-cycled Li-ion, **98–99%** each is typical); converter **maximum power (W)** (e.g., 5 V × 2 A = **10 W**); your device's **consumption profile** (either hourly, in `Consumption_Profile.xlsx`, or **day/night** averages in `Day_Night_Profile.xlsx`); and your **reserve threshold** (the minimum SoC you want to maintain, e.g., **10%**).

3) Capture workflow

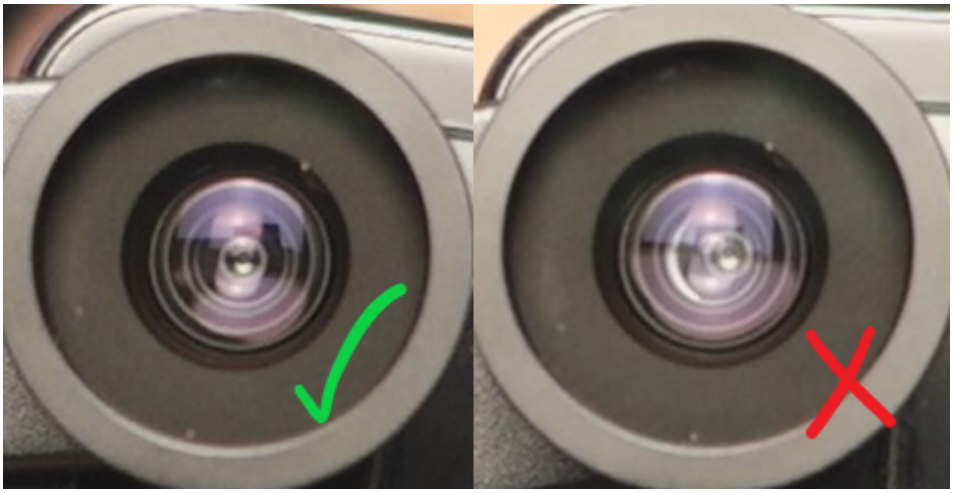


Figure — How to correctly put a fisheye lens.

Start by mounting the clip-on fisheye correctly. Center it over the chosen camera: when you look through the lens, the sensor should appear in the middle (see Figure). Set the phone on a flat, stable surface; modern phones have uneven camera bumps and multiple modules, so the adapter can easily sit crooked if you're not careful. In the preview, check that the dark rim (vignetting) looks **symmetrical**, that's a quick sign you're centered. After you take the sky photo(s), **don't move the lens** so you can use the same setup for calibration.

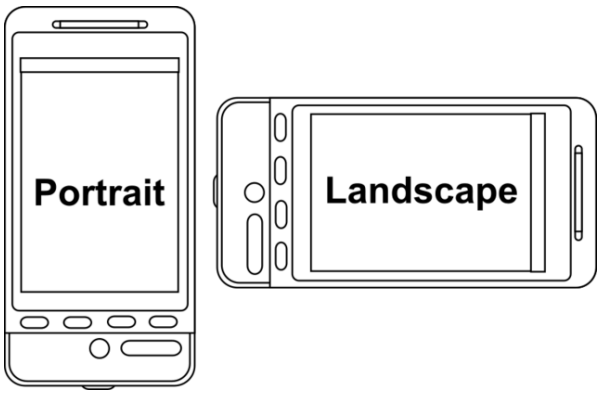
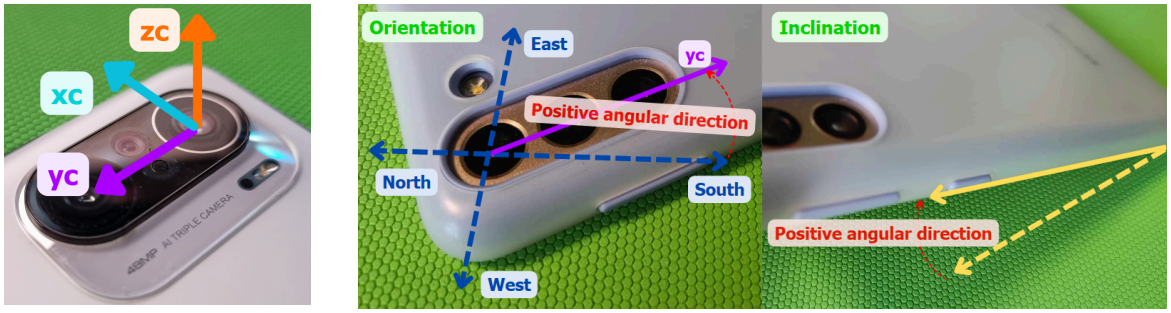


Figure — Portrait and Landscape images.

Keep the phone/lens clean and plan to shoot preferably in **portrait** or **landscape** (no square images). All photos, including **calibrations** and **sky images**, should have the same format.

When taking the photos, do not disturb the fisheye lens to avoid incorrect calibration.



Figures — Coordinate reference of the camera and how to record the phone's orientation and inclination.

In portrait, the bottom of the image aligns with the phone's **yc** axis. You'll use this to record orientation and tilt. Before shooting, note:

- **Orientation (°):** where the **bottom** of the photo (yc) points, e.g., South = 0°, East = +90°, West = −90°.
- **Inclination (°):** tilt relative to horizontal.

*Use a compass app (set to **true heading** and calibrate it) and a bubble-level app. If this feels complicated, a simple, valid pose is yc toward **South** with the phone **flat** (both angles = 0°).* To find **latitude, longitude, elevation**, Google Maps is fine. Note that the program will automatically do an API call to get elevation, and correct yours if it's wrong.

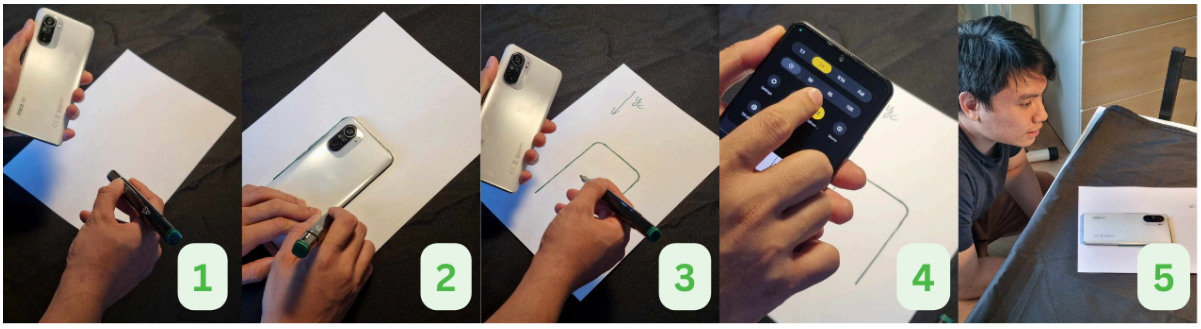


Figure — How to take a photo of the sky

Step by step on how to take photos of the sky.

1. Find a **stable surface** at the panel location. Bring a **pen/tape** to mark the phone's outline. For the simplest setup, aim the **bottom of the photo (yc)** toward **South,** that gives orientation = 0°.
2. Place the phone **face up** so you can read the compass (unlike the figure), center the fisheye, and trace the outline.
3. Note the pose: use a **compass (true heading)** to record the orientation of the **bottom edge (yc)**, and a **bubble-level** to record inclination. Stay in **portrait**.
4. Set a delayed (**5–10s here**) **timer**.

5. Step out of the field of view, and capture the **sky photo.**
   Don't bump the lens afterward, you'll reuse the exact setup for calibration.

Multiple sky photos (same pose).
You may add photos from **different points across the panel area**, for example on each edge of the solar panel, **without changing orientation or tilt**. This simulates partial shading across the panel. Put them all in `SkyImageOfSite/`; the program will ask whether to use one photo or **combine** them.

---

4) Project layout & model files

The repository layout used by the current version is:

```
├── CalibrationImages/          # Camera calibration images
├── DebugData/                  # Debug output files
├── SkyImageOfSite/             # Site sky images for analysis
├── SystemData/                 # System specifications, profiles, and ML models
│   ├── Consumption_Profile.xlsx
│   ├── Day_Night_Profile.xlsx
│   ├── System_Specifications.xlsx
│   └── [ML model files]        # Download from Google Drive
├── omnicalib/                  # Camera calibration library
├── README.pdf                  # Detailed usage instructions
└── *.py                        # Main program files
```

ML models. Download the model weights (e.g., EfficientNet `.pt` files and any meta files) from the Google Drive link referenced in the github repo and place them in `SystemData/`.

Model choices (pick one):

- **512×512 EfficientNet-b5 or b7** — fastest, but you lose precision. Good for low-end systems.
- **1024×1024 EfficientNet-b5 or b7** — more precise and still reasonably fast. Works on most computers.
- **1024×1024 EfficientNet-b5 or b7 + LGBM** (gradient-boosting post-refinement; one LGBM per base model) — slowest but most precise. Should run on any recent computer.

Tips: b7 is usually a little more accurate but heavier than b5. If you can, use the most precise model (EfficientNet-b7 with LGBM). If you have trouble with this model, try the faster options.

5) Camera calibration

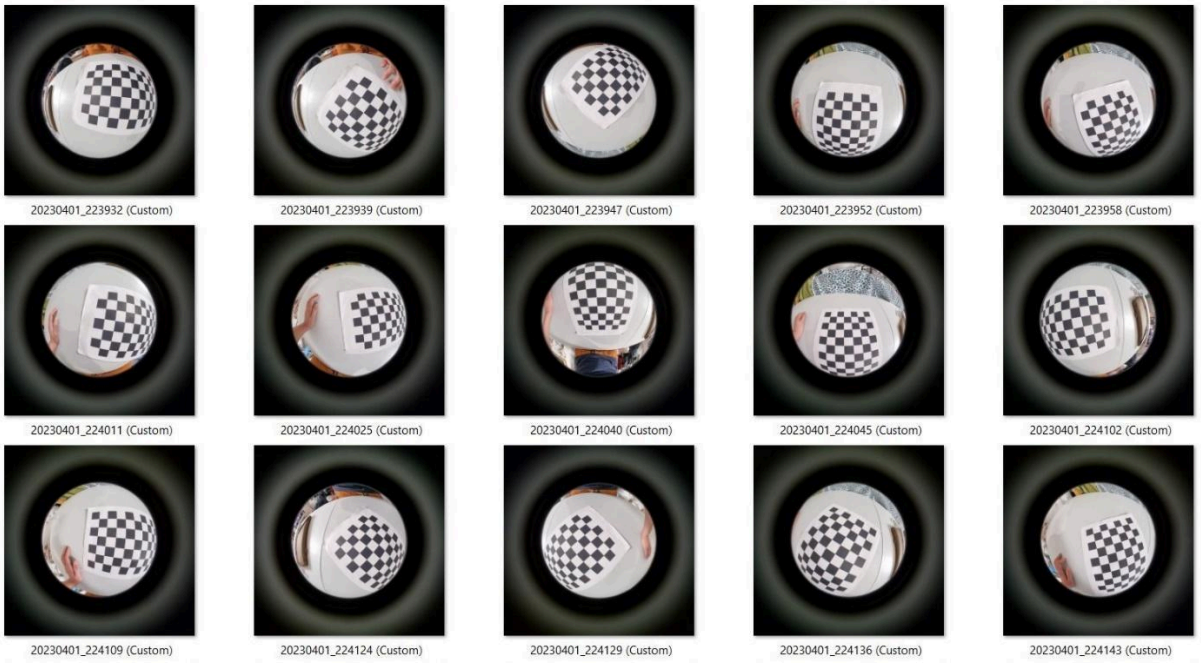


Figure — Bad calibration set: no pattern covers the top left of the field of view.

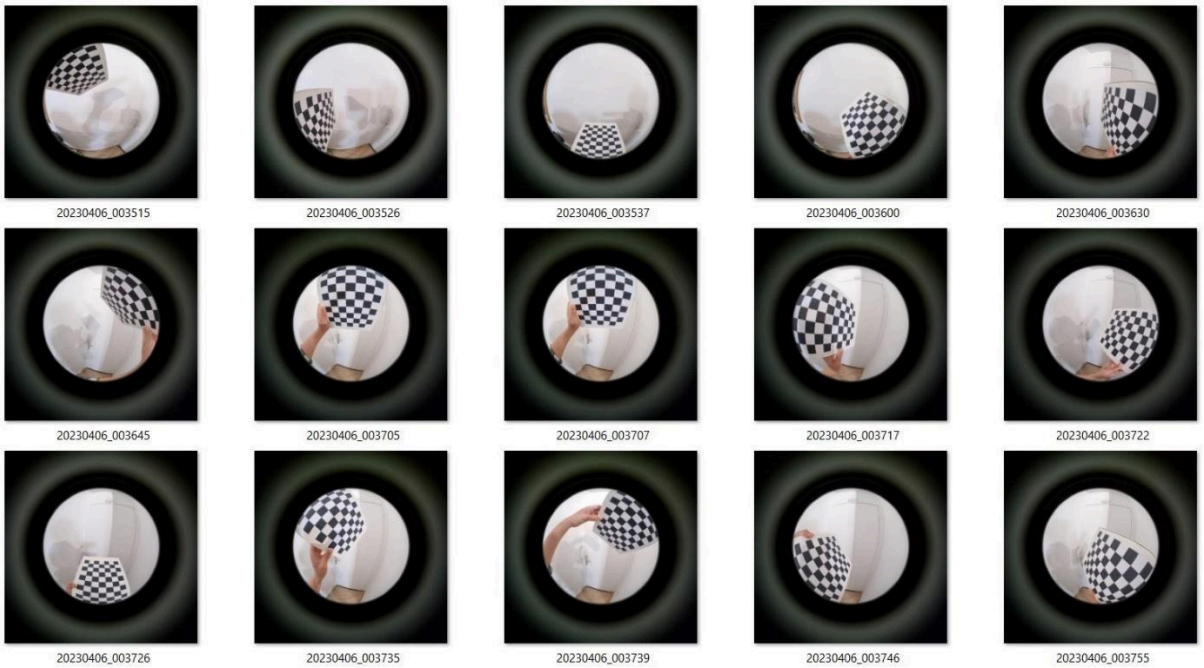


Figure — Good calibration set: all vertices are visible, and all parts of the field of view is covered

Calibrate the **same phone + lens + pose** you used for the sky photo(s). If your sky was shot in **portrait**, all calibration photos must also be in **portrait**. Don't take a single photo—take **many** so the checkerboard covers the **center, edges, and all four corners** of the frame. This broad coverage is what stabilizes the fisheye model.

For each image, here are some recommendations:

- Use **even lighting**—avoid flares and hard shadows on the pattern.
- Hold the board so **all inner vertices (6×9)** are clearly visible.
- For **edge coverage**, push the outer squares **slightly beyond the frame** while keeping the vertices visible.
- Keep the **phone fixed in portrait** (or landscape if your sky images are in landscape format) and **move the board**, not the phone—this keeps orientation consistent.

After calibration, open `DebugData/fov_test.jpg`. The red circle should line up with the fisheye rim. If it doesn't, your set likely missed some regions. Compare your shots with from the two **Figures**: the bad set leaves corners uncovered, while the good set shows vertices visible everywhere and full coverage of the image.

---

6) Preparing inputs

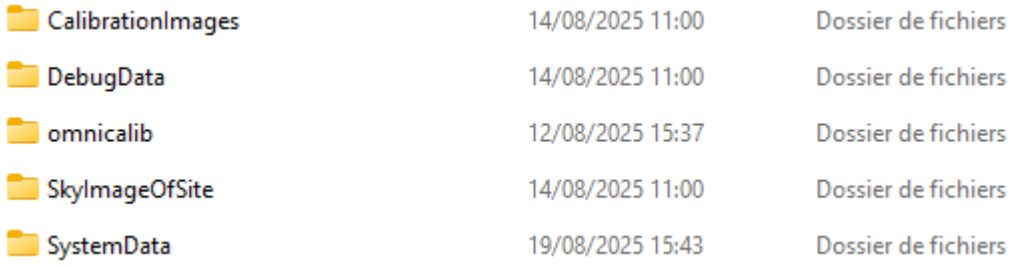


Figure — List of folders present in the program. You may complete CalibrationImages, SkyImageOfSite, and SystemData

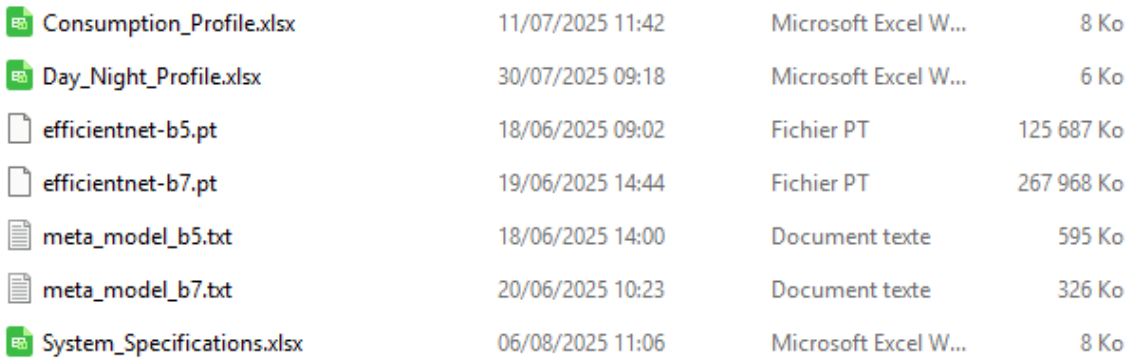


Figure — What files should be present in the SystemData folder

Open `SystemData/System_Specifications.xlsx` and fill a row with your **location** (lat, lon, elevation), **date range** for evaluation, **checkerboard details** (6×9 inner corners, square size in mm), panel and converter ratings, and **battery** capacity/nominal voltage. Set your **reserve threshold**—the minimum SoC you want to maintain (for example 10% to protect battery health and margin).

For the load profile you have two options:

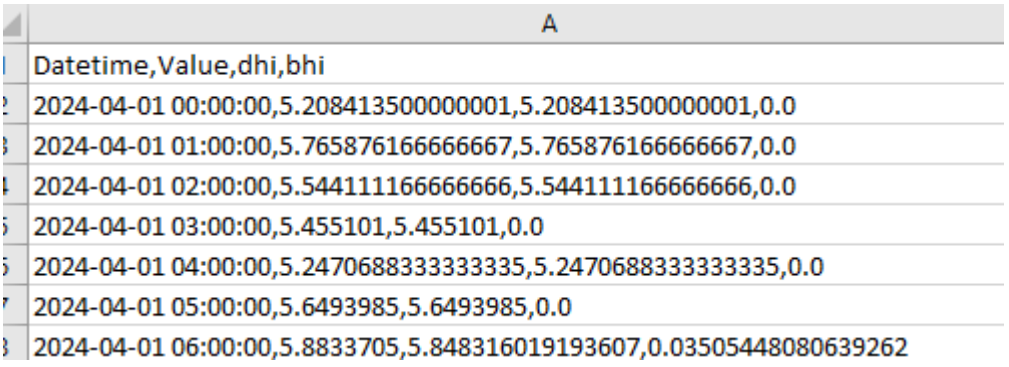
- **Hourly profile (`Consumption_Profile.xlsx`)** for devices whose consumption varies through the day; the 24 entries represent average **W** (or **Wh per hour**) for each clock hour.

- **Day/Night profile (`Day_Night_Profile.xlsx`)** for simpler systems; provide one average draw for when the sun is above the horizon (day) and one for when it is below (night). If both files are present, the program asks which one to use.

Place the files where the program expects them. Put your **checkerboard calibration photos** in `CalibrationImages/`. Put your **sky photo(s)** into `SkyImageOfSite/`. The **ML model files** (EfficientNet b5/b7 and, if you plan to use it, the LGBM refinement model that matches your choice) must also be in `SystemData/` so the automatic sky segmentation can run. The program reads from these folders and writes diagnostics and plots to `DebugData/`.

---

7) Custom irradiance data (optional)



	A
1	Datetime,Value,dhi,bhi
2	2024-04-01 00:00:00,5.208413500000001,5.208413500000001,0.0
3	2024-04-01 01:00:00,5.765876166666667,5.765876166666667,0.0
4	2024-04-01 02:00:00,5.544111166666666,5.544111166666666,0.0
5	2024-04-01 03:00:00,5.455101,5.455101,0.0
6	2024-04-01 04:00:00,5.2470688333333335,5.2470688333333335,0.0
7	2024-04-01 05:00:00,5.6493985,5.6493985,0.0
8	2024-04-01 06:00:00,5.8833705,5.848316019193607,0.03505448080639262

Figure — Example CSV format: comma-separated values with a single header row (first line contains column names); no extra metadata or preamble lines.

By default, the program retrieves irradiance for your location from NASA POWER. If you already have measured or modeled data, you can use your **own CSV or Excel file** instead—no coding required. Place the file in `SystemData/`.

Your file should contain a **timestamp column** and irradiance data in **W/m²**. You can provide either a single **GHI** column, or a pair of columns for **direct** and **diffuse** light:

- One GHI column (Global Horizontal Irradiance). The program will **split it into direct and diffuse** using a standard method (Erbs). This requires the `pvlib` package, which is installed with `requirements.txt`.
- Or two columns: **DNI + DHI** (Direct Normal + Diffuse Horizontal), or **BHI + DHI** (Beam Horizontal + Diffuse Horizontal).

When you run the tool, it will **list the available files** in `SystemData/` and ask you to pick one. If needed, it will ask you to **confirm which columns** contain the data. Clear headers like “timestamp”, “GHI”, “DNI”, “BHI”, and “DHI” make this easy. Make sure your timestamps **cover the date range** you set in `System_Specifications.xlsx`; hourly cadence is typical.

Note: if your data aren’t hourly, the program resamples to hourly resolution by averaging the values within each hour.

---

7) Running the program

Once your files are in place, run the tool from the repository folder. Make sure Python is installed, install the dependencies with `pip install -r requirements.txt`, then start it with `python main.py`. The first launch can take a moment while Python initializes.

The program guides you interactively. It performs a short setup pass: it **reads** your scenario row from `System_Specifications.xlsx`, **loads** the selected load profile, **opens** the sky photo(s) and the calibration set, and **confirms** that the required model files are available. If something is missing or inconsistent, the program catches the issue and **waits for you to fix it** (for example, by correcting a file or path) before continuing, nothing is discarded; you stay in control. Processing then follows four stages:

- **Camera & lens calibration** — builds the fisheye model from your checkerboard set and checks the field of view.
- **Sun path & site irradiance** — computes the sun’s trajectory for your dates and loads the site irradiance used for the simulation.
- **Shading from sky images** — overlaps the sun path with the ML-generated sky/obstacle masks to get **direct** and **diffuse** shading factors.
- **Energy balance & battery SoC** — applies shading to irradiance, simulates panel → converter/charger → battery → load, and outputs the SoC timeline and verdict.

On the first run, all stages execute. On later runs, only the stages affected by your recent changes are recomputed; if nothing changed, you can choose which stages to run. If several sky photos are present, you’ll be asked whether to process just one or to **combine** them (same orientation/tilt) to simulate partial shading across the panel. You’ll also choose the model option you want (see Section 3).

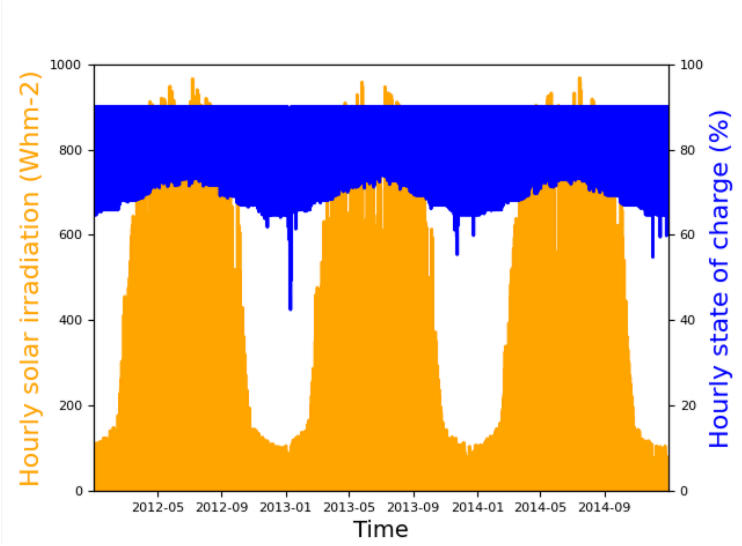


Figure — Example of hourly_soc_estimation.png

At the end, a text file named like `YYYYMMDD-HHMM-verdict.txt` appears in the repository root. It reports the **minimum SoC** over your evaluation period and a short conclusion against your **reserve threshold**, in other words, whether the system can run continuously under your assumptions. For a quick visual, open `DebugData/hourly_soc_estimation.png` (names may differ depending on your selected consumption profile). Over long periods the hourly points merge visually, but you can still see the overall SoC trend. If you want the full data, open `SystemData/sov_ev.xlsx`, which logs hour-by-hour **SOC**, **Energy flow**, **Consumption**, and **Irradiation (after shading)** so you can make your own plots.
