

PV-OPTIM: A Smart Adaptive Optimization and Control Software

Version 1.1

Configuration Guide

Contents

1. Layer 1 - PV-OPTIM Forecast & Optimization	2
1.1. Prerequisites and dependencies.....	2
1.2. Data files	2
1.3. Components	2
1.3.1. PVForecast.py	2
1.3.2. PVOptim.py	2
2. Layer 2 - PV-OPTIM Monitor & Control.....	3
2.1. Prerequisites and dependencies.....	3
2.2. Data management.....	3
2.3. Configure the connection with the inverter to retrieve data.....	3
2.4. Set-up the smart plugs connectivity	3
2.5. Configure the python scripts	4
2.6. Deploy the orchestration flow.....	4
3. Layer 3 - PV-OPTIM Dashboard.....	8
3.1. Prerequisites and dependencies.....	8
3.2. Configuration	8
3.3. Deployment.....	8

1. Layer 1 - PV-OPTIM Forecast & Optimization

1.1. Prerequisites and dependencies

PV-OPTIM Layer 1 requires Python 3 and the following libraries: *pandas*, *numpy*, *scikit-learn*, *xgboost* and *matplotlib*.

1.2. Data files

Layer 1 uses the following .csv files located in the *data_folder* folder:

- *weather_readings.csv* - weather records for the last 30 days;
- *inverter_readings.csv* – inverter records for the last 30 days with the same timestamp used in *weather_readings.csv*.
- *weather_forecast.csv* – forecast data of the next 24 hours up to 10 days using the same parameters and metrics as in *weather_readings.csv*.
- *tariffs.csv* - tariff rates for the next 24 hours;
- *app_list.csv* – the list of the appliances that will operate the next day;
- *app_constraints.csv* – user preferences and the characteristics of the appliances.

The templates of the .csv files including sample data can be found in the *data_folder*.

Please do not change the templates!

1.3. Components

Layer 1 contains 2 components as follows:

1.3.1. PVForecast.py

Forecast the PV output for the next period depending on the time frame provided in *weather_forecast.csv*. Run the script to see the following metrics: RMSE, MAPE, R^2 and MAE of the forecast.

The forecasted values are saved into *inverter_forecast.csv*.

1.3.2. PVOptim.py

Optimize the schedule for day-ahead based on the PV available power estimated by *PVForecast.py* and the list of appliances and user preferences provided in *app_list.csv* and *app_constraints.csv*. Run the script to see the chart with the optimal schedule and the available PV power estimated for the next day.

The optimal schedule is saved in *optimal_schedule.csv* and the remaining quantities or the deficit is saved in *deficit_surplus.csv*.

The following parameters should be configured in *PVOptim.py*:

- *interval* - represents the interval for optimization in minutes (20, 30, 60). Default 30.
- *pcmax* - maximum load (W) per interval. Default 4500 W.
- *day* – day for optimization. Format 'yyyy-mm-dd'. It should be a day for which the forecast has been made and is included in *inverter_forecast.csv*. Default '2022-08-02'.
- *TNP_load* – total non-programable load per interval or sum of the background consumption per interval (W). Default 200 W.

2. Layer 2 - PV-OPTIM Monitor & Control

2.1. Prerequisites and dependencies

PV-OPTIM Layer 2 requires Python 3 and the following libraries: *pandas*, *numpy*, *scikit-learn*, *xgboost*, *matplotlib*, *mysql.connector* or *mysqlclient*, *urllib*, *pytz*;

2.2. Data management

Layer 2 uses a database management system to store and manage data.

Install and configure MySQL or MariaDB. For full steps and documentation see: <https://www.mysql.com/downloads/> or <https://mariadb.org/>.

After installation, configure the database schema for PV-OPTIM as follows:

1. Connect as *root* and create a new database user *PV_OPTIM* and set its default password to *pv_optim1234*:

```
CREATE USER 'pv_optim'@'localhost' IDENTIFIED WITH  
mysql_native_password BY 'pv_optim1234';
```

2. Open and run the script *db_create_PVOPTIM_schema.sql* to create the database tables.

As a result, the database schema is configured, and the following tables are created (figure 1):

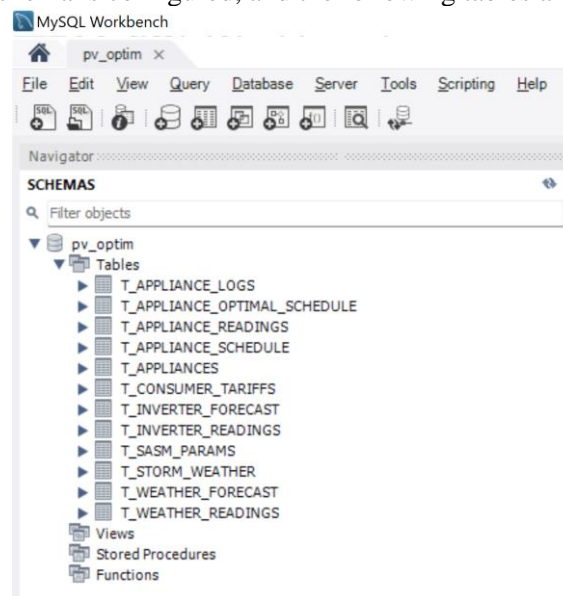


Figure 1 – PV-OPTIM database schema

3. Connect as *PV_OPTIM* and display the tables.
4. Optional – load sample data into PV-OPTIM schema by running *sample_data.sql*. This may take a few minutes. The data set contains records from one PV system of 5 kW collected between 17.02.2022-27.08.2022.

2.3. Configure the connection with the inverter to retrieve data

The current version of PV-OPTIM connects only to Growatt inverters using Growatt API implemented in the *growattServer* python library. Full documentation, methods and examples are available at:

https://github.com/indykoning/PyPi_GrowattServer.

The connection with the inverter and the data collection process is implemented in *inverter.py*. Edit the script and update the username and password used for <https://server.growatt.com/login> page.

2.4. Set-up the smart plugs connectivity

PV-OPTIM can connect to various models of smart plugs from different providers.

The current implementation uses the following TP-Link smart plugs models: HS-110 and KP 115 with energy monitor options. For other models please check the providers' documentation to install and configure the devices and edit the orchestration flow as described in section 2.5.

Configure TP-Link smart-plugs as described in <https://www.tp-link.com/us/home-networking/smart-plug/>

Open your router administration page and see the IP allocated to the smart-plugs. It is recommended to reserve the IP for the smart-plugs using Address Reservation options on the console of the router. Use the allocated IP to configure the orchestration flow in section 2.5.

2.5. Configure the python scripts

Download the python files into a local directory and edit them to provide the required parameters as follows:

1. PVForecastL2.py
 - tz - specify the time zone. Default 'Europe/Bucharest'
 - day - specify the start day for the forecast. Default current day: `dt.datetime.now().strftime('%Y-%m-%d')`
2. PVOptimL2.py
 - interval - represents the interval for optimization in minutes (20, 30, 60). Default 30.
 - pmax - maximum load (W) per interval. Default 4500 W.
 - day - day for optimization. Default current day: `dt.datetime.now().strftime('%Y-%m-%d')`
3. open_weather.py
 - plat - latitude of the PV system
 - plon - longitude of the PV system
 - API_key - your API key to retrieve data from OpenWeather API (<https://openweathermap.org/api>).
4. storm_weather.py
 - plat - latitude of the PV system
 - plon - longitude of the PV system
 - API_key - your API key to retrieve data from StormGlass API (<https://stormglass.io/>).

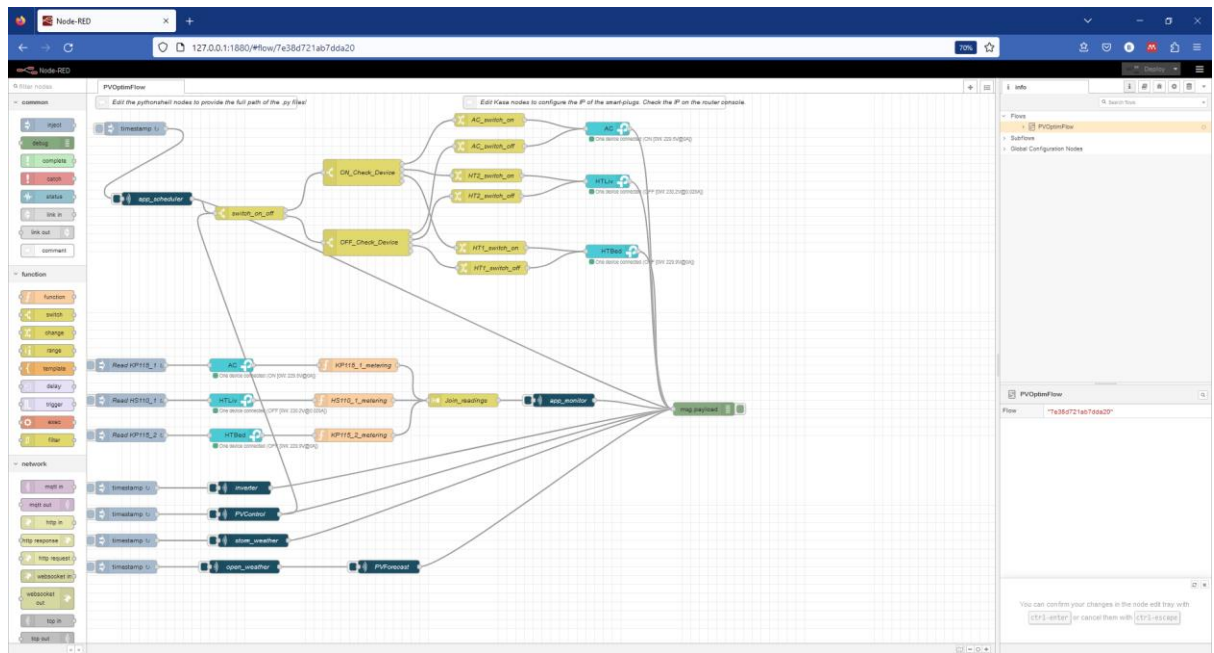
2.6. Deploy the orchestration flow

The orchestration flow enables the connectivity and the processing steps between the smart-plugs, inverter, weather APIs and the python scripts.

The current version of the flow uses TP-Link smart-plugs configured via *Kasa* nodes. To use other smart-plugs from different providers (D-Link, Xiaomi, Broadlink) please see the requirements of the devices, configure their IP and add the corresponding nodes in the flow.

Follow the steps to configure the orchestration flow:

5. Install and configure Node-RED. See full documentation on: <https://nodered.org/#get-started>
6. Run Node-RED and open the console in a web browser. By default, it is accessible on <http://localhost:1880/>
7. Import the orchestration flow of PV-OPTIM provided in the repository in file *PVOptimFlow.json*. Use *Import* option from the Node-RED menu on the right corner of the console, select the file and load it. As a result, the *PVOptimFlow* will be loaded into the console (figure 2):



8. Resolve the dependencies of the nodes by installing *pythonshell* (node-red-contrib-pythonshell) and *Kasa* (node-red-contrib-tplink) nodes using Manage Palette option from the Node-RED menu.
9. Edit the *Kasa* nodes to configure the IP and the name of the smart-plugs (figure 3):

Edit kasa node

Delete Cancel Done

Properties

Name HTBedroom

Device IP 192.168.0.192

Connection poll interval 30000

Event poll interval 30000

Output Payload getMeterInfo

Debug ☐

10. Add/delete *Kasa* nodes to add or remove smart-plugs depending on the number of devices that need to be controlled. The current version uses two KP115 and one HS110 TP-Link smart-plugs with energy monitor option.
11. Edit the *pythonshell* nodes to specify the full path of the python scripts (figure 4):

Edit pythonshell in node

Delete Cancel Done

Properties

Provide a python file with full path

Name

Py file

Virtual Environment Path

Continuous? ☒

Stdin Input? ☒

Continuous: this means the script will continuously produce data (good for trigger once, run forever scripts). This option will always be checked if Stdin Input is checked.

Stdin Input: when this is checked, input to the node will be fed to the stdin of the scripts. That is, one the very first input, the script will be launched and wait for data from its stdin.

Figure 4 – Specify the full path of the python scripts in the *Pythonshell* nodes

12. Optional: edit the input (inject) nodes to change the time interval for requests. For example, the *timestamp* node corresponding to the *open_weather pythonshell* node can be edited to change the repeat interval to 6 hours (figure 5).

Figure 5 shows the 'Edit inject node' configuration window. The 'Properties' tab is active. The 'Name' field is empty. The configuration table shows two rows: 'msg. payload' is set to 'timestamp' and 'msg. topic' is set to a variable. The 'Inject once after 0.1 seconds, then' checkbox is checked. The 'Repeat' dropdown is set to 'interval' and the 'every' field is set to '0.1' hours. The 'Enabled' checkbox is checked.

Figure 5 – Change the time interval for the requests

13. Deploy the entire flow and check the *Debug* window for messages. Correct the issues if any. As a result, the orchestration flow will run the python scripts, monitor and control the smart-plugs automatically. You may close the browser and let the Node-RED service running in the console.

3. Layer 3 - PV-OPTIM Dashboard

3.1. Prerequisites and dependencies

PV-OPTIM Layer 3 requires Python 3 and the following libraries: *pandas*, *numpy*, *scikit-learn*, *xgboost*, *matplotlib*, *mysql.connector* or *mysqlclient*, *urllib*, *pytz*, *flask*, *flask_sqlalchemy*, *flask_wtf*, *wtf*forms, *WTForms-Alchemy*, *pymysql*, *flask_login*, *flask-bcrypt*.

First, you need to configure Layer 2 before using Layer 3. So, please follow section 2 to set-up and configure Layer 2.

3.2. Configuration

Edit the *run.py* file located in the HR folder of Layer 3 and edit the IP and the port on which the application will run. By default, the application will run on localhost:5000.

3.3. Deployment

Run the application (execute *run.py*) and access the dashboard on the web browser (figure 6).

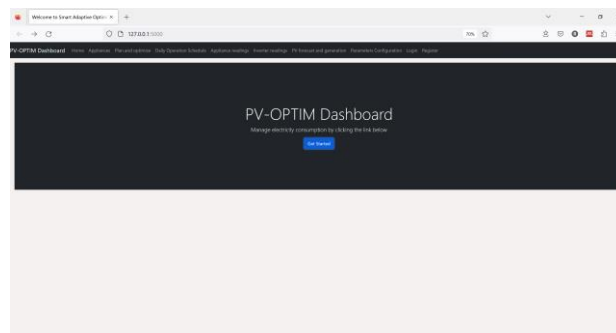


Figure 6 – PV-OPTIM home page

If you loaded the sample data in step 2.2, you can login with the default user account: *pv_optim/pv_optim1234*.

To access the sample data, click *Get Started* and then login with the above credentials.

To create an account, select *Register* from the main menu and provide an username and password. The credentials will be saved into the database in table *T_USERS*.

3.3.1. Appliances page

After login, the Appliances page is shown (figure 7) and you can add a new appliance or update the existing ones (in case you load the sample data). The appliances info are saved in the database table *T_APPLIANCES*.

ID	Name	Device type	Description	Priority	Max operation time	Capacity (kWh)	Actionable	Info	Options
AC	AC Heating/Cooling	Interruptible	Air conditioning	100	60	1000	Yes		Max Update this appliance
HT1	Heating Bedroom	Interruptible	Heating bedroom	100	60	1800	Yes		Max Update this appliance
HT2	Heating Livingroom	Interruptible	Heating Livingroom	100	60	1300	Yes		Max Update this appliance
WH	Water Heater	Shiftable	Water Heater	100	60	1800	Yes		Max Update this appliance
ES	Electric Stove	Shiftable	Electric Stove	100	60	1200	Yes		Max Update this appliance
EO	Electric Oven	Shiftable	Electric Oven	100	60	1800	Yes		Max Update this appliance
WM	Washing Machine	Shiftable	Washing Machine	100	90	1000	Yes		Max Update this appliance
WP2	Well Pump	Interruptible	Well Pump	100	60	900	Yes		Max Update this appliance
PP	Pool Pump	Interruptible	Pool Pump	100	60	1000	Yes		Max Update this appliance

Showing 1 to 9 of 9 rows | 12 rows per page

[Add new appliance](#)

Figure 7 – Appliances page

3.3.2. Plan and optimize page

To plan the operation of the appliances on a specific day (day-ahead), click on *Add appliance to plan* and provide your preferences regarding the *start time*, *end time*, *number of cycles* (number of operations) and *duration*. If you plan to use an appliance with less than its full capacity, then set *Maximum Power coefficient* between 0.1 and 1. Click *Save Schedule* to return.

After planning the operations of the appliances, click *Optimize the plan* (figure 8).

ID appliance	Operation	Start time	Last start time	Priority	Power coefficient	No of cycles	Duration	Options
EO	1	2022-08-26 13:00:00	2022-08-26 14:00:00	1	1.0	1	30	Delete schedule Update schedule
WH	1	2022-08-26 12:30:00	2022-08-26 18:00:00	1	1.0	1	60	Delete schedule Update schedule
AC	1	2022-08-26 11:00:00	2022-08-26 18:00:00	1	1.0	3	30	Delete schedule Update schedule
ES	1	2022-08-26 11:00:00	2022-08-26 15:00:00	1	1.0	1	30	Delete schedule Update schedule
WP2	1	2022-08-26 09:00:00	2022-08-26 13:00:00	1	1.0	6	10	Delete schedule Update schedule

Plan the operation of the appliances. To optimize the daily schedule, please click Optimize the plan!

Figure 8 – Optimize the plan

Select day, time interval for optimization (in minutes) and maximum power load (in kW) accepted by your system. In case you loaded the sample data, you may select 26th of August 2022 or 29th of July 2022 to see an optimized schedule (figure 9).

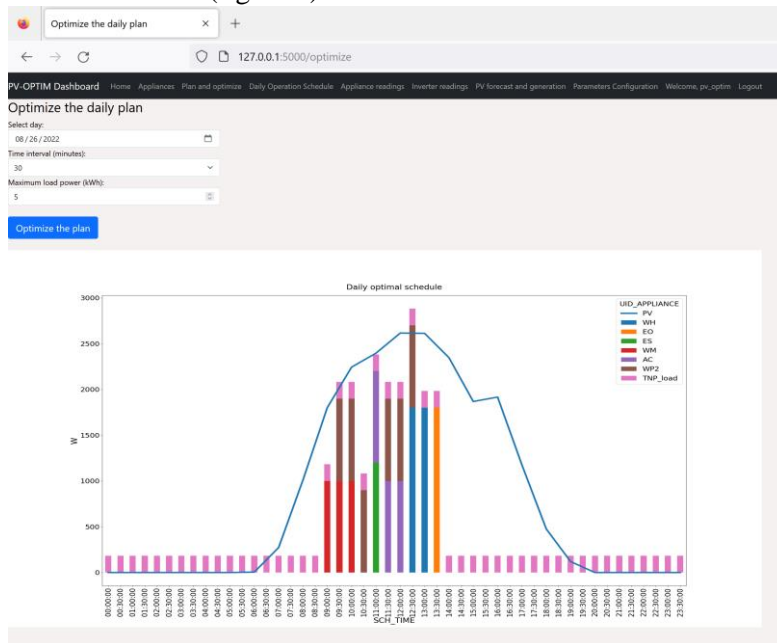


Figure 9 – See the optimal schedule on chart

The optimal schedule is saved into the database and can be accessed and edit (optionally) in the *Daily operation Schedule* page (see section 3.3.3).

3.3.3. Daily Operation Schedule page

After optimization, the optimal schedule can be see and updated. Click on Edit/Confirm schedule to make adjustments of the start time, end time or maximum end time. To enable real time control, choose Yes in the Set active: list (figure 10). To actually control an appliance you need to configure properly the orchestration flow as described in section 2.6.

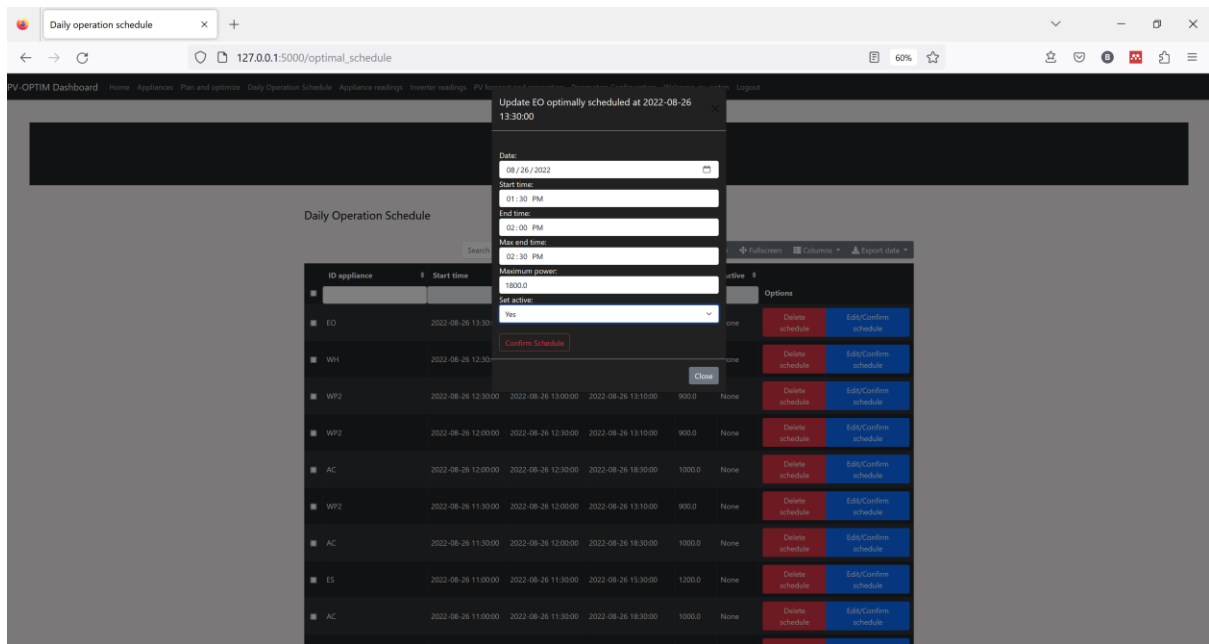


Figure 10 – Edit the optimal schedule to activate real time control for an appliance

In case you need to add a new appliance to the daily schedule, click on *Add appliance to the daily operation schedule* and provide the details regarding start time, end time.

The daily schedule is automatically saved in the database and will be controlled by *PVControl* in case you configure the orchestration flow in section 2.2.

3.3.4. Appliance readings page

To see the readings for an appliance, select a day and an appliance from the list and click on *Show readings*. The readings are collected only for the appliances connected to the smart-plugs configured in section 2.2.

3.3.5. Inverter readings page

To see the PV generation, load and battery SOC (if existing), select a day and click *Show readings* (figure 11). In case you loaded the sample data, you can select a day between February-August 2022.



Figure 11 – Inverter readings page

3.3.6. PV Forecast and generation page

To display a comparison between the PV output and the forecast, select a day and click *Show readings* (figure 12).

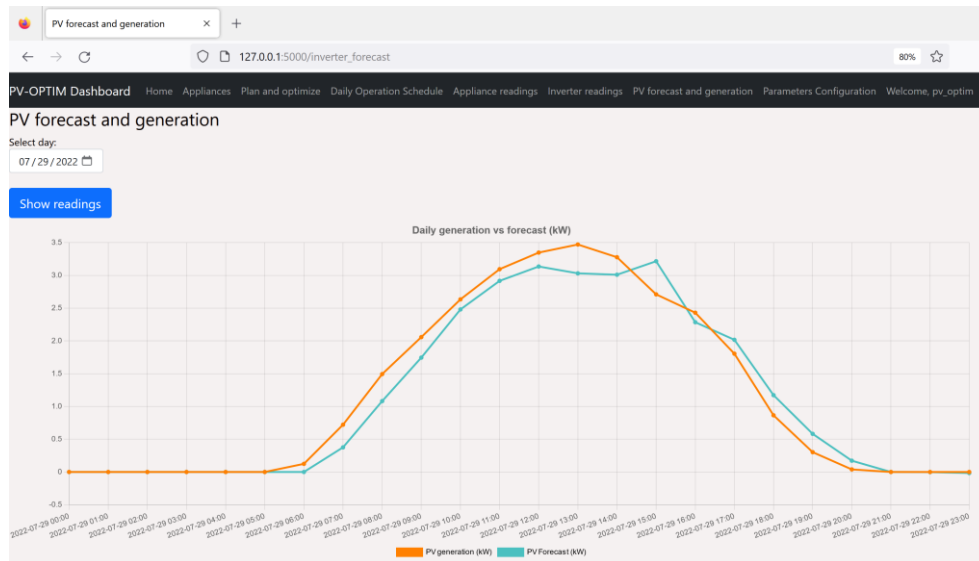


Figure 12 – Comparison between hourly forecast and generation

In the bottom of the page, the estimated PV power is provided as a minimum, average and maximum value (figure 13).

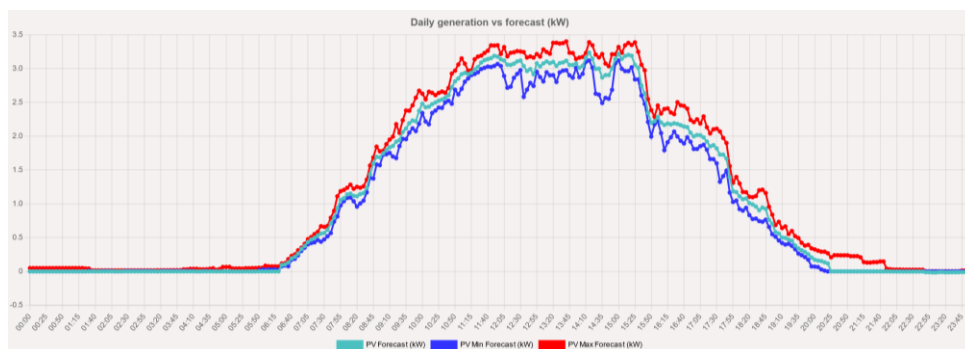


Figure 13 – Estimated values of the PV output (minimum, average and maximum)

3.3.7. Parameters Configuration page

The real time control (PV-OPTIM Control) is using the following parameters to decide whether to switch on/off appliances (figure 14):

- *papp_to_control* – the minimum value (W) of an appliance power that will be controlled. Devices with consumption below the specified value will not be turned off;
- *pbat_discharge* – The power limit from battery (W) that is allowed for the consumption. If the total load exceeds this value, then some appliances will be switched off;
- *sasm_control* – To enable PV-OPTIM Control set its value to 1, to disable it set the value to 0;
- *sasm_load_rate* – The power limit (W) allowed to exceed the PV generation. If the total load exceeds this value, then some appliances will be switched off;

- *sasm_soc_charged* – The SOC state (%) that allows extra loads when battery is charging;
- *sasm_soc_discharged* – The SOC state (%) that allows extra loads when battery is discharging.

Name	Param value	Param value	Options
papp_to_control	200.0	Appliance consumption used for PV-OPTIM control	Edit value
pbat_discharge	400.0	Minimum power consumption from battery to switch OFF (PBAT>=PBAT_DISCHARGE)	Edit value
sasm_control	1.0	Allow PV-OPTIM control: Yes=1, No=0	Edit value
sasm_load_rate	1000.0	Power consumption exceedent: ON<= load_rate	Edit value
sasm_soc_charged	60.0	SOC value to switch ON: SOC>=soc_charged	Edit value
sasm_soc_discharged	80.0	SOC value to switch OFF: SOC<=soc_discharged	Edit value

Showing 1 to 6 of 6 rows 10 rows per page

Figure 14 – Configure PV-OPTIM Control parameters