nestli

Release 1.0.0

Aaron Bojarski, Fazel Khayatian, Hanmin Cai

Nov 11, 2022

CONTENTS:

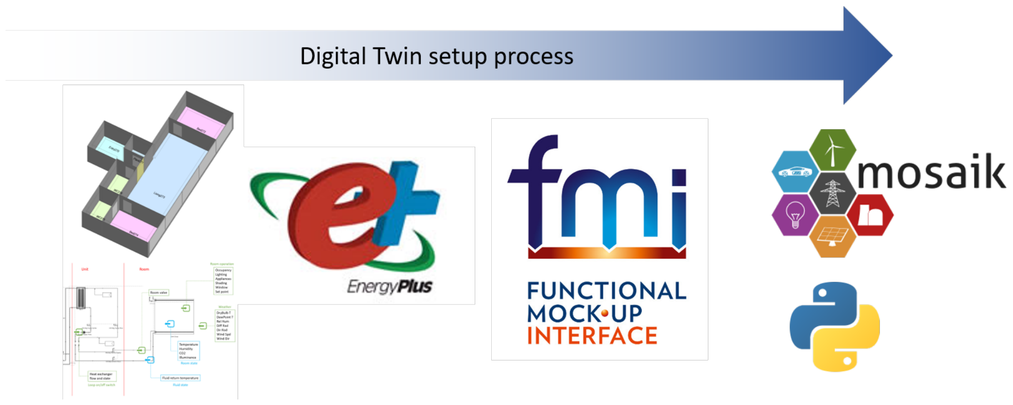
CHAPTER

ONE

NESTLI README

1.1 nestli (Neighborhood Energy System Testing towards Large-scale Integration)

nestli is the virtual sister of the NEST demonstrator at Empa. It is co-simulation environment for benchmarking the performance of BACS (building automation and control systems). In its core, nestli is a calibrated EnergyPlus model of the UMAR living lab. see: https://www.empa.ch/web/nest/urban-mining



The calibrated EnergyPlus model is wrapped into an FMU (fucntional mock-up unit) using the EnergyPlusToFMU tool. see: https://simulationresearch.lbl.gov/fmu/EnergyPlus/export/

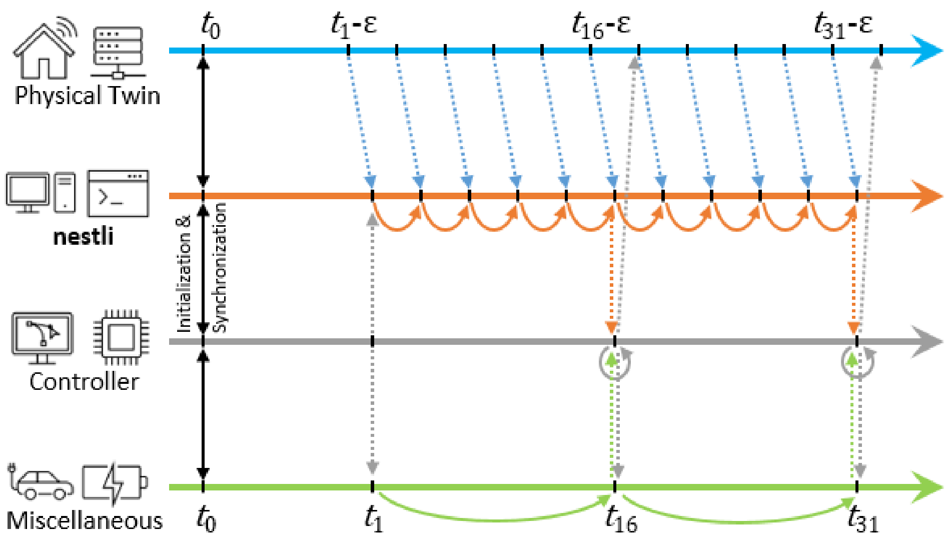
The model is calibrated on measurements that are collected at 1-minute intervals, and thus runs at the same temporal resolution. The HVAC system can be controlled by overriding the setpoint temperature in each room. It is also possible to evaluate the robustness of the controller by manipulating weather conditions and building operation.

1.2 Installation

The installation consists of 3 Steps.

1. install EnergyPlus
2. install Python
3. install the nestli package

Further installation informations are in the pdf documentation. Container-based implementation is also provided using Docker.



1.3 Usage

A simple example of how to use the package is given in the **example** folder. Just run the file nestli_example_run.py where the config example_config.yml specifies all the parameters.

1.4 Citation

If you use our tool for your work, please cite the following paper:

Khayatian, F.; Cai, H.; Bojarski, A.; Heer, P.; Bollinger A.; Benchmarking HVAC controller performance with a digital twin. Applied Energy Symposium: Clean Energy towards Carbon Neutrality (CEN2022). https://www.enerarxiv.org/page/thesis.html?id=3819

CHAPTER TWO

INSTALLATION

2.1 Local python Installation

2.1.1 EnergyPlus

Make sure you get the correct version.

1. Install Energyplus 9.3.0
2. Add it to your PATH

2.1.2 Python

1. Install python version 3.8 or higher.
2. Create a virtual environment: `python -m venv path/to/env/nestli-venv`
3. Activate the environment.

2.1.3 nestli package

1. Clone the repository onto you machine
2. cd to the directory where you cloned it into
3. Install the package with: `pip install -e .` This will install the nestli package according to setup.py.

2.2 Docker

To run the simulation in Docker you just need to have Docker installed. Then clone the repository and navigate to it.

You create a Docker image with:

```
docker build . -t nestli
```

CHAPTER

THREE

COMPONENTS

3.1 Simulators

nestli is composed of a number of simulators that communicate data at each time step. The dependencies among simulators are set in the config file (example_config.yml). Figure 1 shows an example of dependency between the simulators based on the example_config.yml located in:

nestli/example/python_controller/example_config.yml

It is important to note that some inputs to a simulator may have circular dependency. Namely, the input to a simulator depends on the outputs from another simulator at the previous time step. For instance in Figure 1, the input from UMAR to PyCTRL is from the previous time step. In Figure 1, such inputs are annotated by (t-1).

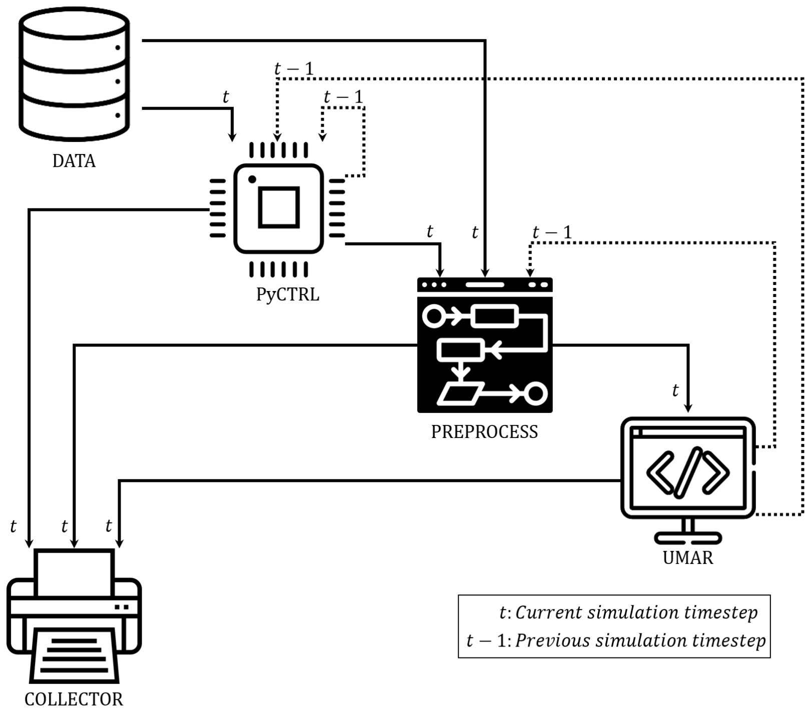

DATA
PyCTRL
PREPROCESS
UMAR
COLLECTOR
t
$t-1$
t: Current simulation timestep
$t-1$: Previous simulation timestep

Fig. 1: A map of dependencies between simulators. This figure explicitly refers to the dependencies between simulators as in the python_controller example.

In the following, each simulator is described in detail:

COLLECTOR: This python function retains the outputs from other simulators at each time-step and saves them into a csv (comma separated value) file at the end of the run. COLLECTOR does not produce any outputs during the run.

ConstantValue: This python function produces a signal with a fixed value. It is particularly useful for parametric or simulation studies when the user desires an input signal with a fixed value.

TabularDataSimPandas: This python function reads the values in the hdf5 files at every time-step and assigns them to the correct input port. Users can override the *.h5 files to manipulate the input signals. However, for some variables (as described in FmuAdapter), users can generate the signals on-the-fly and override the *.h5 values.

FmuAdapter: This python function uses the FMPy library to load FMUs into the nestli environment.

NanPlaceholder: This python function generates a NaN signal. It is useful when the user DOES NOT want to override a variable during simulation, thus providing the override port with a NaN signal.

PythonFunction: This python function reads a python-based controller that is written into a python template. A simple implementation of a python based controller can be found in nestli/example/python_controller/PythonController.py

3.2 Resources

3.2.1 Data

nestli's data is based on measurements from the Urban Mining and Recycling (UMAR) unit of the NEST demonstrator at Empa [1]. UMAR is a residential unit (i.e. a flat) with two bedrooms, two bathrooms and one living room/kitchen. The flat is home to two occupants, each with one bedroom while sharing the living/cooking space. The source of air conditioning for the UMAR unit is provided by central heating and cooling networks that operate at 36°C and 16°C, respectively. nestli's data are either directly measured from the sensors or inferred from measurements during the calibration process. Description of the input data is provided in Table 1. The measured/inferred data spans three years from 2019 to 2021. The measurements are collected at 1-minute intervals. The geometric characteristics of the UMAR unit is displayed in Figure 2. In the nestli environment, the data are stored in HDF5 format at the following path nestli/src/nestli/simulators/resources/data

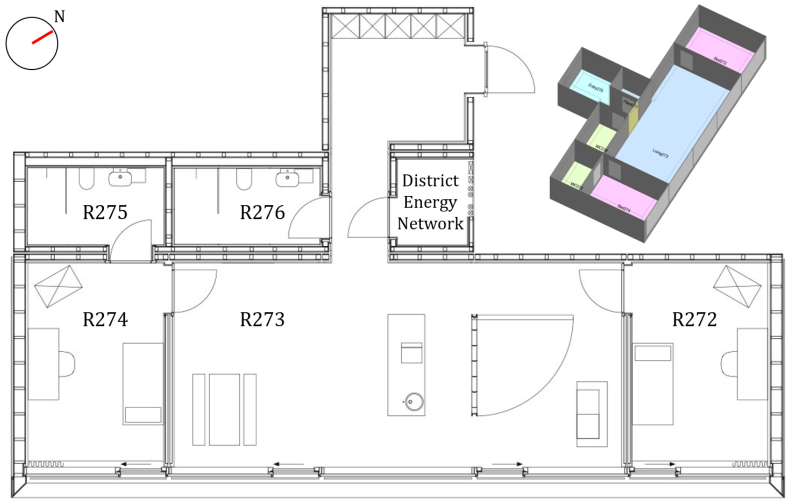


Fig. 2: Plan and axonometric view of the UMAR unit.

Table 1: Description of the input variables

Weather_DryBulb_Temperature	Measured on-site outdoor dry-bulb temperature (℃)
Weather_DewPoint_Temperature	Measured on-site outdoor dew-point temperature (℃)
Weather_Relative_Humidity	Measured on-site outdoor relative humidity (%)
Weather_Diffuse_SolarRadiation	Calculated diffuse horizontal irradiance from on-site measurements of global horizontal radiation (W/m^2)
Weather_Direct_SolarRadiation	Calculated direct normal irradiance from on-site measurements of global horizontal radiation (W/m^2)
Weather_Wind_Speed	Measured outdoor wind speed (m/s^2)
Weather_Wind_Direction	Measured outdoor wind direction (°)
Air_Conditioning_mode	Heating (+1) or cooling (–1) state of the air conditioning system based on measurements of the district network temperature.
District_Network_Temperature	Measured district network fluid temperature (℃)
R272_SetPoint_UpperBound	Nominal upper limit of the comfort bound (℃)
R272_SetPoint_LowerBound	Nominal lower limit of the comfort bound (℃)
R272_Window_Operation	Measured window operation
R272_Shade_Operation	Inferred shading operation from measurements of indoor illuminance values
R272_Occupant_Operation	Inferred internal heat gain from electricity consumption (W)
R273_SetPoint_UpperBound	Nominal upper limit of the comfort bound (℃)
R273_SetPoint_LowerBound	Nominal lower limit of the comfort bound (℃)
R273_Window1_Operation	Measured window operation
R273_Window2_Operation	Measured window operation
R273_Shade_Operation	Inferred shading operation from measurements of indoor illuminance values
R273_Occupant_Operation	Inferred internal heat gain from electricity consumption (W)
R274_SetPoint_UpperBound	Nominal upper limit of the comfort bound (℃)
R274_SetPoint_LowerBound	Nominal lower limit of the comfort bound (℃)
R274_Window_Operation	Measured window operation
R274_Shade_Operation	Inferred shading operation from measurements of indoor illuminance values
R274_Occupant_Operation	Inferred internal heat gain from electricity consumption (W)
R275_SetPoint_UpperBound	Nominal upper limit of the comfort bound (℃)
R275_SetPoint_LowerBound	Nominal lower limit of the comfort bound (℃)
R276_SetPoint_UpperBound	Nominal upper limit of the comfort bound (℃)
R276_SetPoint_LowerBound	Nominal lower limit of the comfort bound (℃)

3.2.2 FMUs

Currently, there are three FMUs in nestli. The first two are UMAR and PREPROCESS, both of which are integral for executing a simulation. These FMUs are stored at: nestli/src/nestli/simulators/resources/fmu

1. UMAR is an EnergyPlus input file (*.idf) which is wrapped into an FMU by using LBNL's EnergyPlusToFMU software [2]. UMAR is based on FMI Version 2.0 (See Annex 1).
2. PREPROCESS is created in Simulink and wrapped into an FMU by the Matlab/Simulink FMU compiler. PREPROCESS ensures that inputs to UMAR are within reasonable ranges and thus reduces the chances of encountering a fatal error when running the simulation (see Annex 2). PREPROCESS has a baked-in hysteresis controller, which actuates based on the upper and lower thermal comfort bounds. PREPROCESS also has provisions for overriding some variables on the fly during the simulation. The names of these variables always end in "_Override" (e.g., R272_Occupant_Override, R273_Shade_Override, R274_Shade_Override, R275_SetPoint_Override, etc.). The source Simulink file of PREPROCESS is stored at: nestli/fmu_source_files

The third FMU is specifically created as an example and therefore stored at:

3. FMUCTRL is created in Simulink and wrapped into an FMU by the Matlab/Simulink FMU compiler. FMUCTRL is an example of how any controller can be wrapped into an FMU and imported into nestli. The current example of FMUCTRL keeps the indoor air temperature near the lower comfort bound during the heating season and near the upper comfort bound during the cooling season (see Annex 3). The source Simulink file of FMUCTRL is stored at: nestli/fmu_source_files

CHAPTER

FOUR

SIMULATION EXECUTION

nestli uses the Mosaik [3] Python library to coordinate the co-simulation between simulators. A graphical representation of the simulation run is provided in Figure 2.

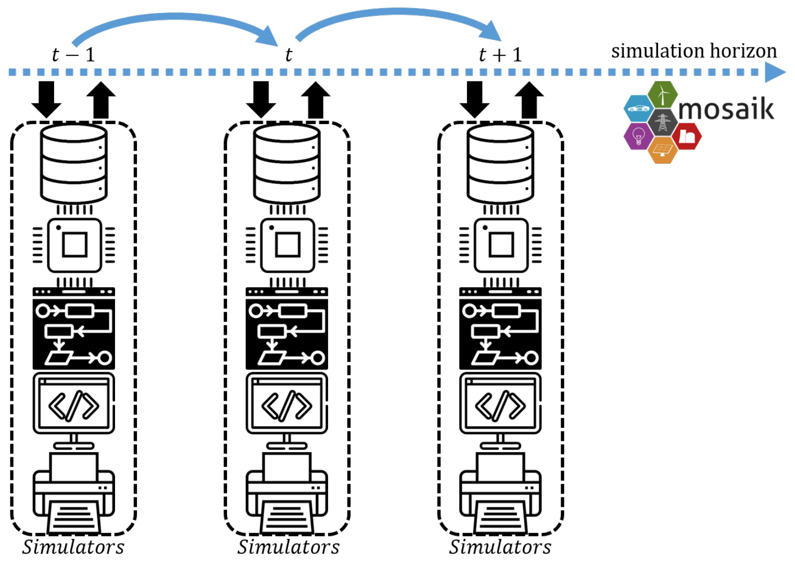


Fig. 1: Co-simulation management by Mosaik. Details of the simulators are provided in Section Components (Figure 1).

Mosaik manages the co-simulation by assigning the correct input data to the relevant input port of simulators at the right time step. Thus, the communication between the simulators during the run is also coordinated by Mosaik. The temporal resolution of nestli simulations is set to 1 minutes due to settings in the EnergyPlus model of UMAR (idf file). As a result, Mosaik manages communications at a sampling time of 1-minute. The PREPROCESS simulator is tightly integrated with the UMAR simulator, and therefore, should also use a 1-minue sampling time. Other simulators can be configured to execute at time-steps other than (and down to) 1-minutes.

CHAPTER FIVE

USAGE

5.1 Overview

The simulation is handled, initialized and started my the Manager class.

To run a simulation, you need to start a Manager instance with a config file. The config file specifies all simulation parameters.

5.1.1 Script

The following python script is all that is needed for a simulation.

```
example_manager = Manager("example_config.yml")
example_manager.build_simulation()
example_manager.run()
```

5.1.2 Config file

The config file is a yaml file in which the simulation parameters are set. It first specifies some general parameters.

```
OUTPUT_FOLDER_PATH: ""
START: '2022-05-12'
DURATION: 10
RESOLUTION: 60
```

The DURATION (days) has to be an integer and the RESOLUTION (seconds) must be the same as the model resolution.

After that all the simulators are specified. A PATH can either be absolute or relative to the config file folder path or if it is in the nestli package you can see the example below.

```
# The Simulators specify all the models in the simulation. Currently the following␣
↪simulators are available:
#   FMU:  This will create a simulator from an FMU.
#         PATH specifies the folder where the FMU is contained and NAME the filename.
#   TABULAR_DATA: This will create a simulator from DATA in tabular form. The supported␣
↪format is hdf5.
#         PATH specifies the folder where the .h5 files are contained. The file name␣
↪must correspond with the variable name of the data it containes.
#   NAN_PLACEHOLDER: This simulator will create a NaN value to all connected entities.
#   COLLECTOR: This is a simple Data collector. It will save the data of all signals you␣
↪connect to it and write them to a file after the simulation.
```

(continues on next page)

(continued from previous page)

```
#         No additional information is neccessary.
#   PYTHON_FUNCTION: This will create a controller from a python function. You specify␣
↪the Path to the python class.

SIMULATORS:
    SIM1:
        NAME: "UMAR"
        TYPE: "FMU"
        PATH: "$nestli$./simulators/resources/fmu"
        NUMBER_OF_MODELS: 5
```

Last all the connections between the simulators are defined. For each simulator you specify all the outgoing signals.

```
MAPPINGS:
    SIGNALS_OUTGOING_FROM_UMAR:
        FROM: SOURCE
        MAPPINGS:
            # A simple mapping
            MAPPING1:
                TO: DESTINATION
                VARIABLES:
                    SOURCE_VARIABLE_NAME_1: DESTINATION_VARIABLE_NAME_1
                    SOURCE_VARIABLE_NAME_2: DESTINATION_VARIABLE_NAME_2
            # A mapping with circular dependency and multiple models and thus an index␣
↪has to be specified
            MAPPING2:
                TO: DESTINATION
                INDEX: 0
                CIRCULAR_DEPENDENDY: TRUE
                INITIAL_VALUES:
                    SOURCE_VARIABLE_NAME: 24.5669
                VARIABLES:
                    SOURCE_VARIABLE_NAME: DESTINATION_VARIABLE_NAME
```

5.2 Run Simulation

5.2.1 Local python

To run in locally you can just run the python file you created.

5.2.2 Docker

To run with Docker you first need to create an image with:

```
docker build . -t nestli
```

This is further described in the installation section.

You run a container specifying your source file path with PATH_TO_PROJECT_FOLDER with:

```
docker run -it -v "PATH_TO_PROJECT_FOLDER:/example_folder" nestli
```

This folder must contain a python file called nestli_example_run.py and your config file and other files your simulation needs. It will be copied to the container and the results will get copied back to it.

CHAPTER

SIX

RELEASE NOTES

6.1 0.2.0

- added full python controller functionality and template
- updated occupant data
- updated fmu
- changed duration to days

6.2 0.1.0

First usable version of nestli with example for behavior of occupants

CHAPTER

SEVEN

REFERENCES

[1] Richner, Peter, et al. "NEST-A platform for the acceleration of innovation in buildings." Informes de la Construccion 69.548 (2017): e222.

[2] Nouidui, T. S., D. M. Lorenzetti, and M. Wetter. "EnergyPlusToFMU." (2013).

[3] Schütte, Steffen, Stefan Scherfke, and Martin Tröschel. "Mosaik: A framework for modular simulation of active components in smart grids." 2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS). IEEE, 2011.

CHAPTER

EIGHT

ANNEX

Table 1: The input and outputs of UMAR. (t: current timestep, t-1: previous timestep)

Inputs	Outputs
DryBulb_Temperature (t)	R272_Air_Temperature (t)
DewPoint_Temperature (t)	R273_Air_Temperature (t)
Relative_Humidity (t)	R274_Air_Temperature (t)
Diffuse_SolarRadiation (t)	R275_Air_Temperature (t)
Direct_SolarRadiation (t)	R276_Air_Temperature (t)
Wind_Speed (t)	DistrictHeating_Supply_Temperature (t)
Wind_Direction (t)	DistrictHeating_Return_Temperature (t)
R272_Occupancy (t)	DistrictHeating_Return_Flow (t)
R273_Occupancy (t)	DistrictCooling_Supply_Temperature (t)
R274_Occupancy (t)	DistrictCooling_Return_Temperature (t)
R272_Heating_Flow (t)	DistrictCooling_Return_Flow (t)
R272_Colling_Flow (t)	
R273_Heating_Flow (t)	
R273_Colling_Flow (t)	
R274_Heating_Flow (t)	
R274_Colling_Flow (t)	
R275_Heating_Flow (t)	
R275_Colling_Flow (t)	
R276_Heating_Flow (t)	
R276_Colling_Flow (t)	
DistrictHeating_Flow (t)	
DistrictCooling_Flow (t)	
R272_Shade (t)	
R273_Shade1 (t)	
R273_Shade2 (t)	
R273_Shade3 (t)	
R274_Shade (t)	
R272_Window (t)	
R273_Window1 (t)	
R273_Window2 (t)	
R274_Window (t)	
R272_SetPoint_Temperature (t)	
R273_SetPoint_Temperature (t)	
R274_SetPoint_Temperature (t)	
R275_SetPoint_Temperature (t)	

continues on next page

Table 1 – continued from previous page

Inputs	Outputs
R276_SetPoint_Temperature (t)	
DistrictHeating_Temperature (t)	
DistrictCooling_Temperature (t)	
DistrictHeating_Switch (t)	
DistrictCooling_Switch (t)	

Table 2: The input and outputs of PREPROCESS. (t: current timestep, t-1: previous timestep)

Inputs	Outputs
Weather_DryBulb_Temperature (t)	DryBulb_Temperature (t)
Weather_DewPoint_Temperature (t)	DewPoint_Temperature (t)
Weather_Relative_Humidity (t)	Relative_Humidity (t)
Weather_Diffuse_SolarRadiation (t)	Diffuse_SolarRadiation (t)
Weather_Direct_SolarRadiation (t)	Direct_SolarRadiation (t)
Weather_Wind_Speed (t)	Wind_Speed (t)
Weather_Wind_Direction (t)	Wind_Direction (t)
Air_Conditioning_mode (t)	R272_Occupancy (t)
District_Network_Temperature (t)	R273_Occupancy (t)
R272_Simulated_Air_Temperature (t-1)	R274_Occupancy (t)
R272_SetPoint_UpperBound (t)	R272_Heating_Flow (t)
R272_SetPoint_LowerBound (t)	R272_Colling_Flow (t)
R272_SetPoint_Override (t)	R273_Heating_Flow (t)
R272_Window_Operation (t)	R273_Colling_Flow (t)
R272_Window_Override (t)	R274_Heating_Flow (t)
R272_Shade_Operation (t)	R274_Colling_Flow (t)
R272_Shade_Override (t)	R275_Heating_Flow (t)
R272_Occupant_Operation (t)	R275_Colling_Flow (t)
R272_Occupant_Override (t)	R276_Heating_Flow (t)
R273_Simulated_Air_Temperature (t-1)	R276_Colling_Flow (t)
R273_SetPoint_UpperBound (t)	DistrictHeating_Flow (t)
R273_SetPoint_LowerBound (t)	DistrictCooling_Flow (t)
R273_SetPoint_Override (t)	R272_Shade (t)
R273_Window1_Operation (t)	R273_Shade1 (t)
R273_Window1_Override (t)	R273_Shade2 (t)
R273_Window2_Operation (t)	R273_Shade3 (t)
R273_Window2_Override (t)	R274_Shade (t)
R273_Shade_Operation (t)	R272_Window (t)
R273_Shade1_Override (t)	R273_Window1 (t)
R273_Shade2_Override (t)	R273_Window2 (t)
R273_Shade3_Override (t)	R274_Window (t)
R273_Occupant_Operation (t)	R272_SetPoint_Temperature (t)
R273_Occupant_Override (t)	R273_SetPoint_Temperature (t)
R274_Simulated_Air_Temperature (t-1)	R274_SetPoint_Temperature (t)
R274_SetPoint_UpperBound (t)	R275_SetPoint_Temperature (t)
R274_SetPoint_LowerBound (t)	R276_SetPoint_Temperature (t)
R274_SetPoint_Override (t)	DistrictHeating_Temperature (t)
R274_Window_Operation (t)	DistrictCooling_Temperature (t)
R274_Window_Override (t)	DistrictHeating_Switch (t)

continues on next page

Table 2 – continued from previous page

Inputs	Outputs
R274_Shade_Operation (t)	DistrictCooling_Switch (t)
R274_Shade_Override (t)	
R274_Occupant_Operation (t)	
R274_Occupant_Override (t)	
R275_Simulated_Air_Temperature (t-1)	
R275_SetPoint_UpperBound (t)	
R275_SetPoint_LowerBound (t)	
R275_SetPoint_Override (t)	
R276_Simulated_Air_Temperature (t-1)	
R276_SetPoint_UpperBound (t)	
R276_SetPoint_LowerBound (t)	
R276_SetPoint_Override (t)	

Table 3: The input and outputs of FMUCTRL. (t: current timestep, t-1: previous timestep)

Inputs	Outputs
AirConditioning_Mode (t)	R272_SP_Override (t)
SetPoint_UpperBound (t)	R273_SP_Override (t)
SetPoint_LowerBound (t)	R274_SP_Override (t)
Clock (t)	R275_SP_Override (t)
R272_Simulated_Air_Temperature (t-1)	R276_SP_Override (t)
R273_Simulated_Air_Temperature (t-1)	
R274_Simulated_Air_Temperature (t-1)	
R275_Simulated_Air_Temperature (t-1)	
R276_Simulated_Air_Temperature (t-1)	

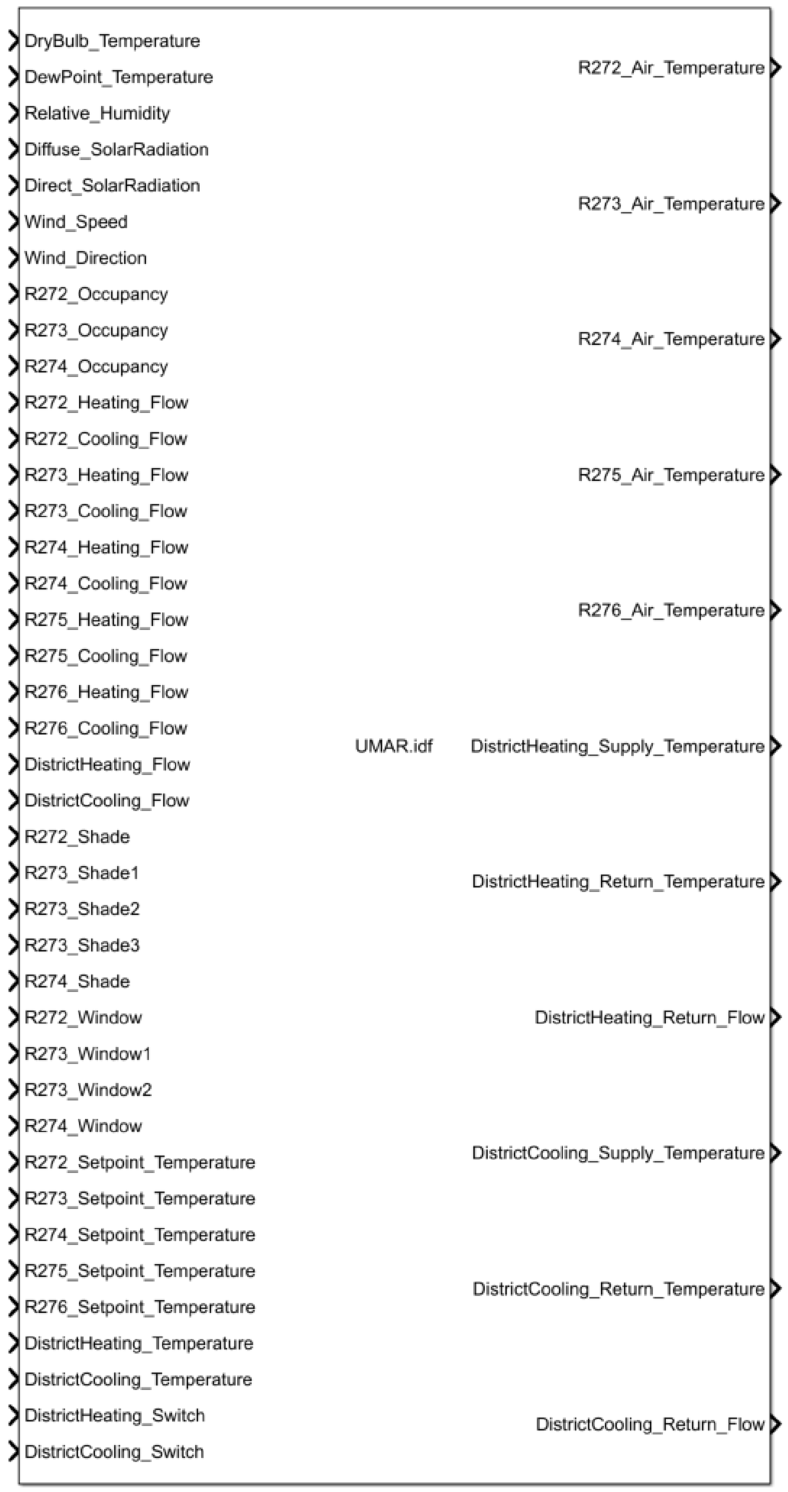


Fig. 1: The input and outputs of UMAR.

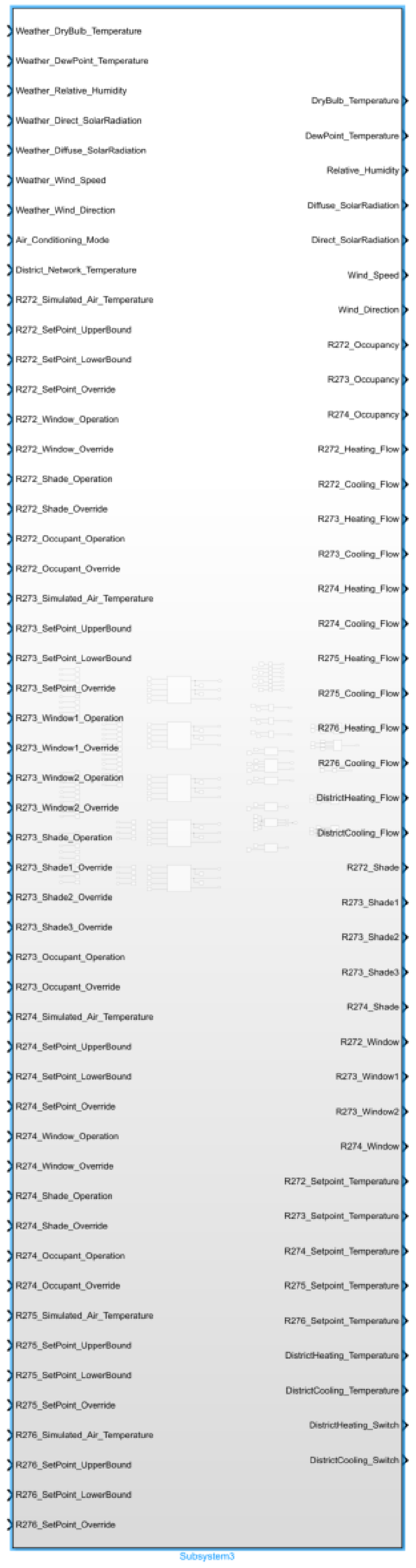


Fig. 2: The input and outputs of PREPROCESS.

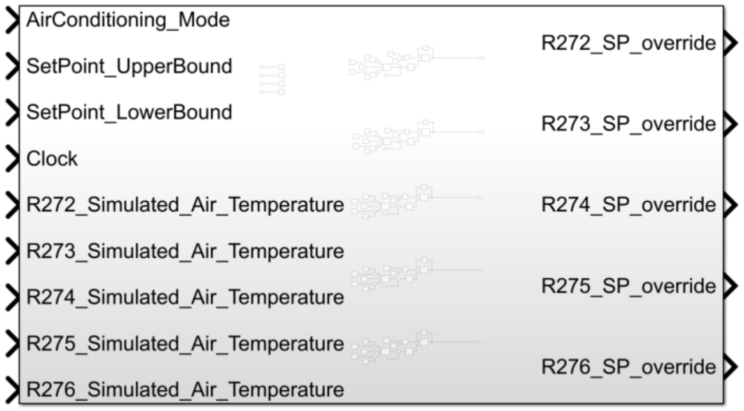

AirConditioning_Mode
SetPoint_UpperBound
SetPoint_LowerBound
Clock
R272_Simulated_Air_Temperature
R273_Simulated_Air_Temperature
R274_Simulated_Air_Temperature
R275_Simulated_Air_Temperature
R276_Simulated_Air_Temperature
R272_SP_override
R273_SP_override
R274_SP_override
R275_SP_override
R276_SP_override

Fig. 3: The input and outputs of FMUCTRL.