KU LEUVEN

Yet Another Residential District Simulator: yards for Controller Development in the Residential Built Environment

Lucas Bex Muhammad Hafeez Saeed Lucas Verleyen
Lieve Helsen Geert Deconinck

09/09/2025

Outline

1. Introduction
2. Design and Features
3. Case Study
4. Conclusion

Residential Demand Response

How to

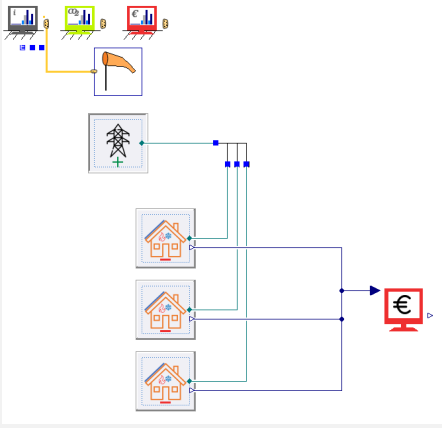
- make energy services meet user needs?
- minimise cost and emissions of energy services?
- valorise energy flexibility?
- ensure reliability of local energy networks?

Considering

- heterogeneous energy service mix
- scalability
- data privacy

District Modeling

Modelica



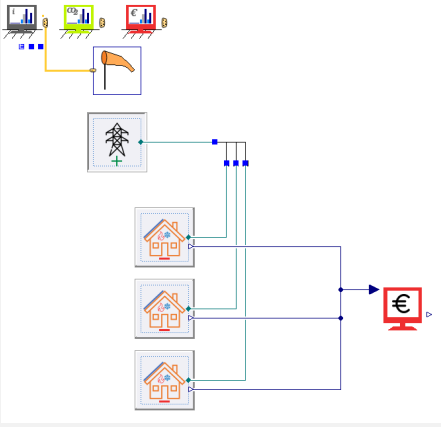
General-Purpose Programming Language

```
import numpy as np
from scipy.integrate import solve_ivp
def thermal_system(t, T, A, f, offset,
    T_ambient, R, C):
    Q_sine = A * np.sin(2 * np.pi * f * t) + offset
    dTdt = (Q_sine - (T - T_ambient) / R) / C
    return dTdt
t_span = (0, 100000)
T0 = 293.15
sol = solve_ivp(thermal_system, t_span, [T0],
    args=(A, f, offset, T_ambient, R, C),
    dense_output=True)
t_eval = np.linspace(t_span[0], t_span[1], 1000)
T_eval = sol.sol(t_eval)[0]
```

District Modeling



Modelica

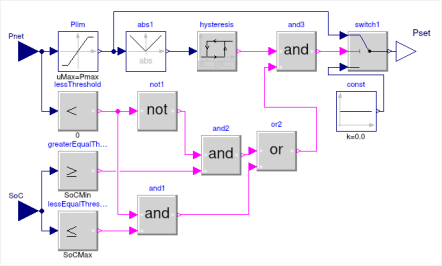


General-Purpose Programming Language

```
import numpy as np
from scipy.integrate import solve_ivp
def thermal_system(t, T, A, f, offset,
    T_ambient, R, C):
    Q_sine = A * np.sin(2 * np.pi * f * t) + offset
    dTdt = (Q_sine - (T - T_ambient) / R) / C
    return dTdt
t_span = (0, 100000)
T0 = 293.15
sol = solve_ivp(thermal_system, t_span, [T0],
    args=(A, f, offset, T_ambient, R, C),
    dense_output=True)
t_eval = np.linspace(t_span[0], t_span[1], 1000)
T_eval = sol.sol(t_eval)[0]
```

District Control

Modelica



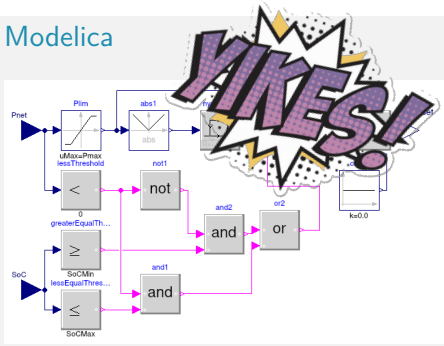
General-Purpose Programming Language

```
function control
  input Real ref;
  input Real SoC;
  input Real SoCMin;
  input Real SoCMax;
  input Real Pmax;
  output Real Pout;
protected
  Boolean nonzero_setpoint;
  Boolean isCharging;
  Boolean can_discharge;
  Boolean can_charge;
algorithm
  nonzero_setpoint := abs(ref) >= 0.05 * Pmax;
  isCharging       := ref < 0;
  can_discharge    := SoC >= SoCMin;
  can_charge       := SoC <= SoCMax;

  if nonzero_setpoint and can_charge
     and isCharging then
    Pout := max(ref, -Pmax);
  elseif nonzero_setpoint and can_discharge
         and not isCharging then
    Pout := min(ref, Pmax);
  else
    Pout := 0.0;
  end if;
end control;
```

District Control

Modelica



General-Purpose Programming Language

```
function control
  input Real ref;
  input Real SoC;
  input Real SoCMin;
  input Real SoCMax;
  input Real Pmax;
  output Real Pout;
protected
  Boolean nonzero_setpoint;
  Boolean isCharging;
  Boolean can_discharge;
  Boolean can_charge;
algorithm
  nonzero_setpoint := abs(ref) >= 0.05 * Pmax;
  isCharging       := ref < 0;
  can_discharge    := SoC >= SoCMin;
  can_charge       := SoC <= SoCMax;

  if nonzero_setpoint and can_charge
     and isCharging then
    Pout := max(ref, -Pmax);
  elseif nonzero_setpoint and can_discharge
         and not isCharging then
    Pout := min(ref, Pmax);
  else
    Pout := 0.0;
  end if;
end control;
```

Dyna-PINN: Physics-informed deep dyna-q reinforcement learning for intelligent control of building heating system in low-diversity training data regimes

Muhammad Hafeez Saeed[a,b,*], Hussain Kazmi[a,b], Geert Deconinck[a,b]

[a] Department of Electrical Engineering, KU Leuven, Belgium
[b] EnergyVille, Thor Park, Genk, Belgium

A Multi-Agent Reinforcement Learning-Based Data-Driven Method for Home Energy Management

Xu Xu, *Member, IEEE,* Youwei Jia, *Member, IEEE,* Yan Xu, *Senior Member, IEEE,* Zhao Xu, *Senior Member, IEEE,* Songjian Chai, *Member, IEEE,* and Chun Sing Lai, *Member, IEEE*

Distributed optimization for scheduling energy flows in community microgrids

Brida V. Mbuwir[*,a,b,c], Fred Spiessens[a,b], Geert Deconinck[a,c]

[a] EnergyVille, Thor Park, Genk 3600, Belgium
[b] VITO, Boeretang 200, Mol, 2400, Belgium
[c] ESAT-Electa, KU Leuven, Kasteelpark Arenberg 10 bus 2445, Leuven 3001, Belgium

Demand Response in Smart Districts: Model Predictive Control of Building Cooling

Philipp Zwickel, Moritz Frahm, Johannes Galenzowski,
Karl-Heinz Häfele, Heiko Maaß, Simon Waczowicz, and Veit Hagenmeyer
Karlsruhe Institute of Technology, Institute for Automation and Applied Informatics, Karlsruhe, Germany
philipp.zwickel@kit.edu

Dyna-PINN: Physics-informed deep dyna-q reinforcement learning for intelligent control of b
regimes

Muhammad Hafeez Saeed[a,b]

[a] Department of Electrical Engineering, KU Leuve
[b] EnergyVille, Thor Park, Genk, Belgium



```
function control
  input Real ref;
  input Real SoC;
  input Real SoCMin;
  input Real SoCMax;
  input Real Pmax;
  output Real Pout;
protected
  Boolean nonzero_setpoint;
  Boolean isCharging;
  Boolean can_discharge;
  Boolean can_charge;
algorithm
  nonzero_setpoint := abs(ref) >= 0.05 * Pmax;
  isCharging       := ref < 0;
  can_discharge    := SoC >= SoCMin;
  can_charge       := SoC <= SoCMax;

  if nonzero_setpoint and can_charge
     and isCharging then
    Pout := max(ref, -Pmax);
  elseif nonzero_setpoint and can_discharge
         and not isCharging then
    Pout := min(ref, Pmax);
  else
    Pout := 0.0;
  end if;
end control;
```

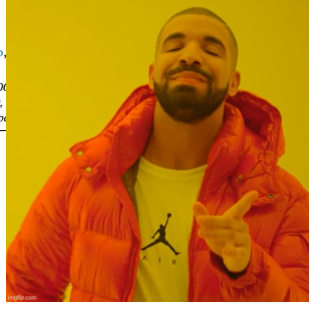
-Based

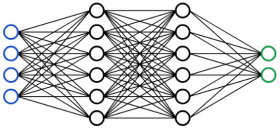
r, IEEE,

Distributed optir
microgrids

Brida V. Mbuwir*[,a,b,

[a] EnergyVille, Thor Park, Genk 3600
[b] VITO, Boeretang 200, Mol, 2400,
[c] ESAT-Electa, KU Leuven, Kasteelpe





Districts:
lding Cooling

zowski,
Veit Hagenmeyer
ormatics, Karlsruhe, Germany

Logos are trademarks of their respective owners. The presenter claims no affiliation with or endorsement by these owners

The *straightforward* option: just use an FMU



The *straightforward* option: just use an FMU

- Modeller must expose all inputs/outputs at top-level
- Low-level FMI interface
- Target language needs good FMI package

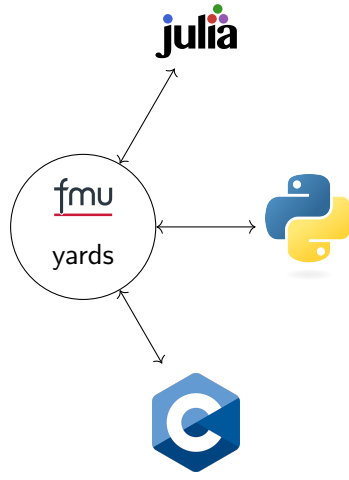




Logos are trademarks of their respective owners. The presenter claims no affiliation with or endorsement by these owners.
Picture: Lucas Bex. "FMIExchange.jl: Foreign Models Through a Familiar Interface". (Eindhoven). July 11, 2024. URL: https://pretalx.com/juliacon2024/talk/XBSX87

Enter yards[1]

- Expose arbitrary model-depth inputs/outputs
- High-level simulation interface
- Runs as a webserver
  - Cross-language controller development
  - Remote simulation
- Easily package custom models
- Based on the BOPTEST framework
  - In-depth comparison → our paper



David Blum et al. "Building Optimization Testing Framework (BOPTEST) for Simulation-Based Benchmarking of Control Strategies in Buildings". In: *Journal of Building Performance Simulation* 14.5 (Sept. 3, 2021), pp. 586–610. ISSN: 1940-1493. DOI: 10.1080/19401493.2021.1986574

[1] always all lowercase

Outline

1 Introduction

2 Design and Features

3 Case Study

4 Conclusion

A Tale of Two Users

Emulator Designer

- Create Modelica model
- Select readable/writable/forecastable variables
- Determine KPIs

A Tale of Two Users

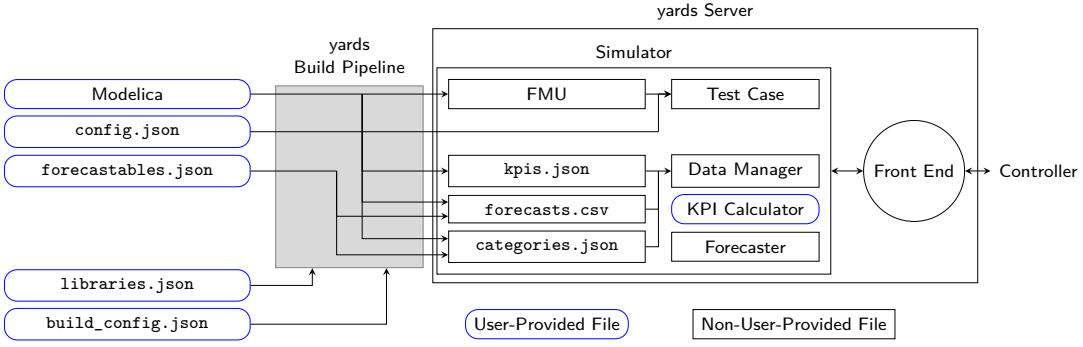
Emulator Designer

- Create Modelica model
- Select readable/writable/forecastable variables
- Determine KPIs

Controller Developer

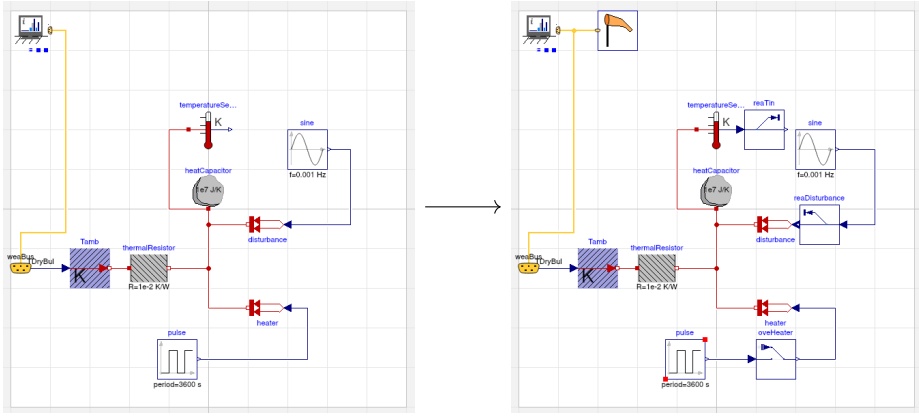
- Develop controller
- Write a `for` loop
- Query KPIs

At a glance

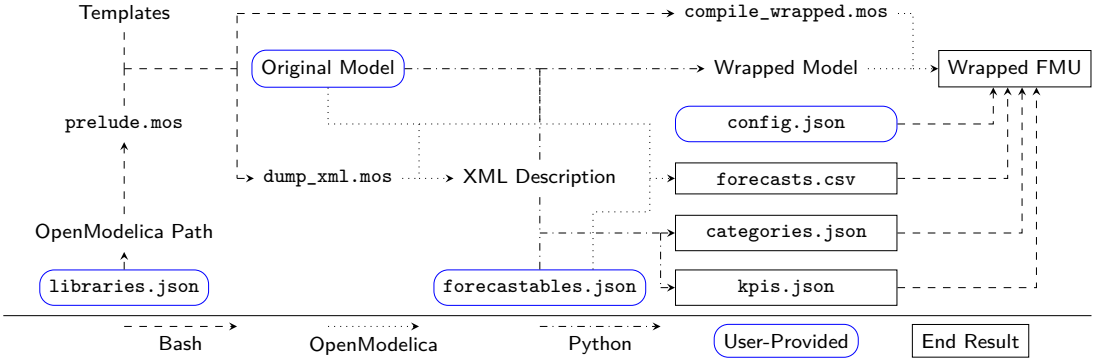

yards Server
yards
Build Pipeline
Simulator
Modelica
config.json
forecastables.json
libraries.json
build_config.json
FMU
Test Case
kpis.json
Data Manager
forecasts.csv
KPI Calculator
categories.json
Forecaster
Front End
Controller
User-Provided File
Non-User-Provided File

From Modelica to yards

- Mark variables for reading/writing in Modelica
- Does not affect model behaviour



From Modelica to yards ctd.



- Combine Modelica model with user-provided metadata files into a wrapper model
- Compile wrapper model to FMU (OpenModelica (default) or manually)
- Generate forecast data
- Workflow runs in Docker container

Custom KPI support

- Implement custom KPI calculator via code injection (or don't)
- Must be implemented in Python
- yards provides helper classes

```
from .baseclass import BaseKPICalculator
class KPICalculator(object):
  # mandatory methods
  def __init__(self, testcase):
    self.base = BaseKPICalculator(testcase)
    self.house_prefixes = ["house1", "house2", "house3"]
  def initialize(self):
    self.base.initialize()
  def get_kpis_absolute(self, scenario=None):
  def get_core_kpis(self, scenario=None):
  # helper methods
  def emissions(self, prefix, injection_scaling_factor=0.5):
  def thermal_discomfort(self, prefix):
  ...
  def get_house_kpis(self, prefix, scenario):
  def get_district_kpis(self, scenario):
```

Interacting with yards through the REST API

From Python (simplified for readability):

```
put(f'{url}/initialize', json={'start_time': start, 'warmup_period': 0})
put(f'{url}/scenario', json={'electricity_price': 'highly_dynamic'})
put(f'{url}/step', json={'step': step})
control = MyController()
for _ in range(0, stop, step):
    measurements = put(f'{url}/measurements')
    forecast = put(f'{url}/forecast', json={"point_names": ["TDryBul"], "
    horizon":3600*24, "interval":3600})
    control_signals = control.compute_control(measurements, forecast)
    post(f'{url}/advance', json=control_signals)
mykpis = get(f'{url}/kpi_absolute')
```

Interacting with yards through the REST API

From Python (simplified for readability):

```
put(f'{url}/initialize', json={'start_time': start, 'warmup_period': 0})
put(f'{url}/scenario', json={'electricity_price': 'highly_dynamic'})
put(f'{url}/step', json={'step': step})
control = MyController()
for _ in range(0, stop, step):
    measurements = put(f'{url}/measurements')
    forecast = put(f'{url}/forecast', json={"point_names": ["TDryBul"], "
    horizon":3600*24, "interval":3600})
    control_signals = control.compute_control(measurements, forecast)
    post(f'{url}/advance', json=control_signals)
mykpis = get(f'{url}/kpi_absolute')
```

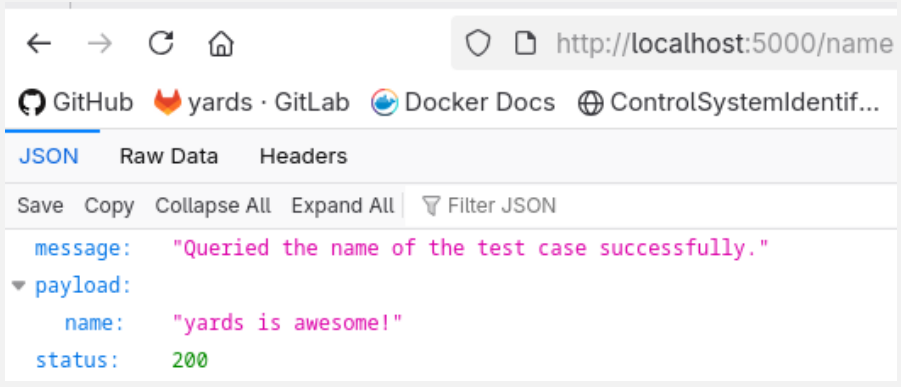
From the terminal:

```
$ curl -s -X PUT "$URL/step" -H "Content-Type: application/json" -d "{\"step
    \": 604800}"
$ curl -s -X POST "$URL/advance" -H "Content-Type: application/json" -d "{}"
$ curl -s "$URL/kpi"
```

Interacting with yards through the REST API

From the webbrowser

please don't actually do this



Outline

1 Introduction

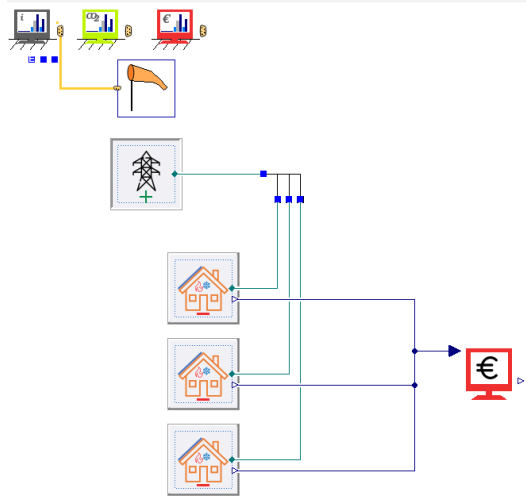
2 Design and Features

3 Case Study

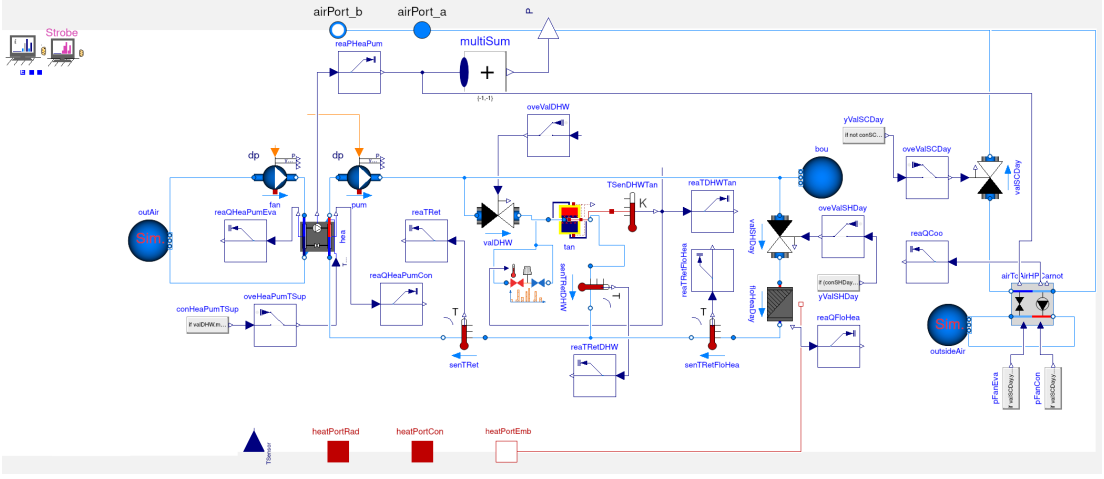
4 Conclusion

Setup - District

- TinyCluster: 3 buildings
- Heat pump for space heating
- Cost-optimal control

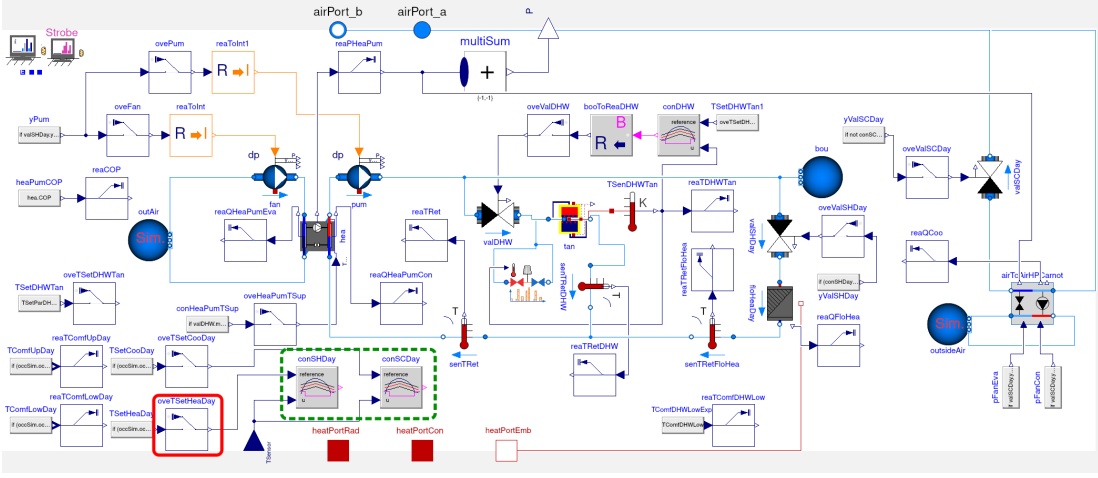


Setup - Heating circuit



Setup - Default Controller C0

- Heating setpoint with subsidiary hysteresis controller
- Setpoint based on building occupancy



Setup - Python controllers

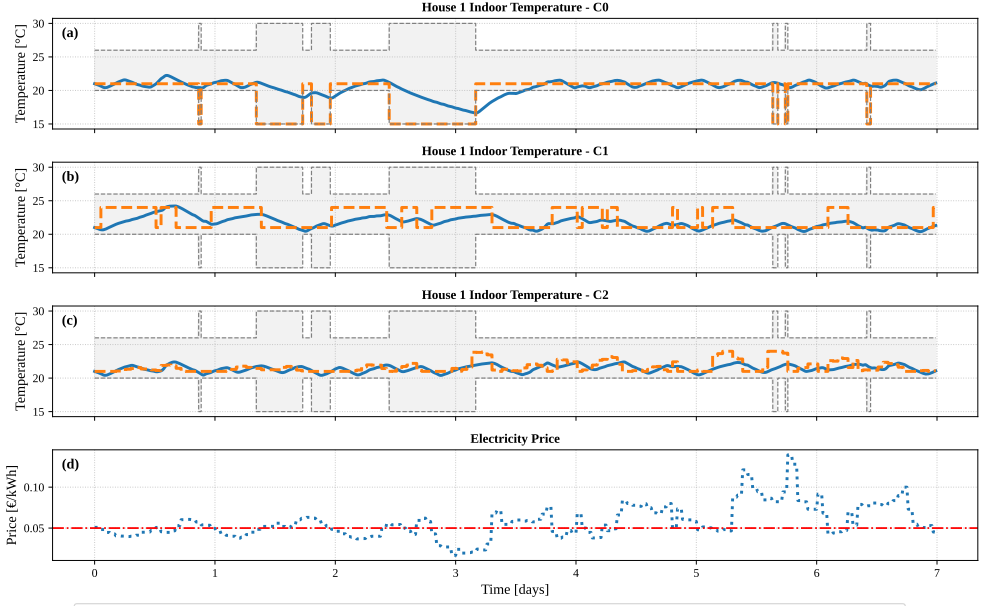
- C1: `if price > threshold then TsetHigh else TSetLow`
- C2: `if Δprice > threshold then TsetHigh else TSetLow`

Sample controller code for C1

```
class PriceBasedController:
    def __init__(self, T_min=273.15 + 21, T_max=273.15 + 24, price_thresh=0.05):
        self.T_min, self.T_max, self.thresh = T_min, T_max, price_thresh

    def step(self, t):
        resp = requests.put(
            f"{self.url}/forecast",
            json={
                "point_names": ["PriceElectricPowerHighlyDynamic"],
                "horizon": 0,
                "interval": 1,
            },
        ).json()["payload"]
        p = resp["PriceElectricPowerHighlyDynamic"][0]
        T_set = self.T_min if p > self.thresh else self.T_max
        cmds = {f"house{i}_hva_oveTSetHeaDay_u": T_set for i in (1, 2, 3)}
        cmds.update({f"house{i}_hva_oveTSetHeaDay_activate": 1 for i in (1, 2, 3)})
        return cmds
```

Results


House 1 Indoor Temperature - C0
(a)
Temperature [°C]
House 1 Indoor Temperature - C1
(b)
Temperature [°C]
House 1 Indoor Temperature - C2
(c)
Temperature [°C]
Electricity Price
(d)
Price [€/kWh]
Time [days]
Indoor Temperature
Temperature Setpoint
Comfort Bounds
Electricity Price
Price Threshold

Results KPIs

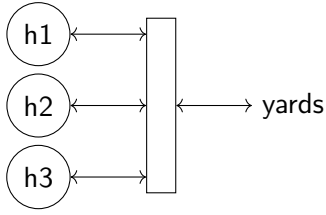
```
curl http://localhost:5000/kpi
```

Metric		C0	C1	C2
Operational Costs [€]	inj[2]	**0.49**	0.24	0.36
	off[3]	45.25	**43.00**	43.31
Operational Emissions [kgCO_2]	inj	**0.32**	0.15	0.26
	off	**42.90**	44.10	43.93
Energy Use [kWh]	inj	**10.25**	4.76	8.11
	off	**760.5**	773.5	775.2
Thermal Discomfort [Kh]	SHC[4]	18.05	**0.00**	**0.00**
	DHW[5]	0.00	0.00	0.00

[2] injection
[3] offtake
[4] space heating and cooling
[5] domestic hot water

Possible further work

- Data-driven MPC using `JuMP.jl` and `ControlSystemIdentification.jl`
- Deep Q-learning using `PyTorch` and `Stable Baselines`
- Multi-agent simulations with central coordinator



Outline

1 Introduction

2 Design and Features

3 Case Study

4 Conclusion

Conclusion

- Cross-language collaborations
- Allow advanced controller design and prototyping on high-fidelity models
- Built-in controller comparison mechanisms
- Strongly customisable
- Open source

Links

Try out yards



- https://gitlab.kuleuven.be/positive-energy-districts/yards

Contact me



- https://www.linkedin.com/in/lucas-bex/
- `cas.bex@kuleuven.be`