Pump Gradual Adjustment Room Temperature for Thermia
PGART_T

Ett litet system för att testa om man kan minska elförbrukningen vid uppvärming med Thermias värmepump Atlas genom att justera pumpens börvärde (önskad innetemperatur) enligt ett enkelt regelverk.

Konfigureringsfilen `pgart_control_heating.conf_se` har utförliga förklaringar.

Programmen är skrivna i Python. Systemet är testat i Oracle VM med Debian och i Raspberry Pi.

Princip

Systemet sänker pumpens värmeproduktion när den inte behövs eller när man kan leva med lägre inomhustemperatur.

Systemet styr genom att ändra pumpens börvärde antingen via APIet i online-genesis.thermia.se eller via ModbusTCP. Det fungerar som om du själv ändrade önskad innetemperatur på pumpens display eller via “hjulet” i online-genesis.thermia.se.

Principen är att minska uppvärmningen när den inte behövs. För ett dåligt isolerat hus där värmen försvinner snabbt är det till föga nytta att värma för fullt under hela natten. Här kan en nattsänkning tillämpas. Temperaturen sänks på kvällen och på sena natten kommer pumpen att värma extra för att få huset rimligt varmt igen. Förhoppningsvis blir det totalt sett en reducering av elförbrukningen. För ett mycket välisolerat hus kan dock besparingen vid en nattsänkning vara försumbar. Isynnerhet om man har billig el på natten (förutsätter elavtal med timdebitering.)

Systemet styrs av reglerna i konfigureringsfilen. Regler för att, i timintervall, sänka börvärdet generellt för veckans alla dagar eller för varje veckodag. Under de arbetsdagar ingen är hemma kan börvärdet ju sänkas under några timmar.

Ett otätt och klent isolerat hus som har en värmepump som saknar innetemperaturmätare kan få lägre temperatur än avsett när det blåser. Systemet kan öka börvärdet de timmar det blåser för att försöka kompensera för temperaturfallet pga vindkyleeffekten.

Systemet kan sänka börvärdet när timpriset är högt. Man anger under vilka timintervall som börvärdet kan sänkas. T.ex. Klockan 8-15 och 20-23.

För att det inte ska bli för kallt inomhus sänker systemet dock maximalt två timmar i sträck. En timme utan temperatursänkning mellan två sänkningsperioder. Programmet beräknar i förväg under vilka timmar som börvärdet kan sänkas för att nå optimal besparing.

Istället för att välja timintervall kan man låta systemet sänka börvärdet för ett antal av dygnets dyraste timmar.

Förutsättningar

Systemet laddar `hourly_rate_yyyymmdd.txt` med timpriserna från de program som anges i konfig-filen. När tabellutseendet på den webbsida som systemet hämtar timpriser från ändras måste programmet som laddar också ändras. Man kan själv lägga till program (med samma anropsstruktur) i systemet. Anger man det i konfig-filen så kör systemet programmet.

Systemet kan också köra ett externt program (som har egen anropsstruktur) för att ladda timpriserna i filen.

Vind och temperaturprognoser hämtas från SMHI. Det verkar fungera i Norden och i Baltikum.

Systemet kan också köra ett program som du ordnat för att ladda filen med prognosdata.

Att TCP-Modbus skulle ändras är högst osannolikt men APIet till online-genesis kan ändras efterhand som det upptäcks buggar eller när en ny version så kräver.

Att se upp med

När TCP är konfigurerat i pumpen är allt där vidöppet på LANet.

Kör man mot en pump som inte står på ens eget LAN måste man använda online-genesis såvida inte platsen (LANet) där pumpen står har en applikationsbrandvägg som skyddar pumpen.

Installation

Untar/unzip `pgart.tar.gz` i `din-user/`

```
pgart
├── bin
│   ├── pgart_calc_hourly_rate_adj_func.py
│   ├── pgart_control_heating.py
│   ├── pgart_env_func.py
│   ├── pgart_experimental_calc_costs.py
│   ├── pgart_experimental_conv_eon_files_to_generic_func.py
│   ├── pgart_get_hourly_rates_elprisetjustnu_se.py
│   ├── pgart_get_hourly_rates_entsoe_eu.py
│   ├── pgart_get_hourly_rates_herrforsnat_fi.py
│   ├── pgart_get_hourly_rates_minspotpris_no.py
│   ├── pgart_get_indoor_temp_raspberry_pi.py
│   ├── pgart_get_monthly_rates_elbruk_se.py
│   ├── pgart_get_smhi_forecasts_func.py
│   ├── pgart_get_smhi_forecasts.py
│   ├── pgart_lang_func.py
│   ├── pgart_misc_func.py
│   ├── pgart_read_control_params_func.py
│   ├── pgart_thermia_modbus_func.py
│   └── pgart_thermia_online_genesis_func.py
├── etc
│   ├── pgart_control_heating.conf_en
│   ├── pgart_control_heating.conf_se
│   ├── pgart_language.conf
│   └── pgart_mail_params.conf
├── var
│   ├── local
│   │   └── explanation_summary_run_log.txt
│   ├── log
│   └── stat
└── xtra_if_systemd
    ├── help_install_systemd.txt
    ├── pgart_control_heating.service
    └── pgart_control_heating.timer
```

Drift

Om du har ett gmailkonto som konfigurerats med ett "app password" kan programmet sända felmeddelanden dit.

Följande tre konfigurationsfiler måste uppdateras initialt:

- `pgart_language.conf.`
- `pgart_control_heating.conf_se/_en.`
- `pgart_mail_params.conf.`

Programmet ska köras en gång i timmen för att logiken skall fungera. Via crontab eller systemd.

`Crontab:`
Startas programmet 1-2 minuter in i timmen så minskas risken för uteblivna svar.
Lägg in följande rad:

```
2 * * * /usr/bin/python3 /home/din-user/pgart/pgart_control_heating.py >>
/home/din-user/pgart/var/log/hourly_run.log
```

`systemd:`
Instruktioner finns i `xtra_if_systemd.`

Det följer med ett program som skapar en fil med kostnaderna per timme för tim- och månadspriser baserat på elförbrukningen. Programmet läser filer med förbrukningsdata som förutsätts ha skapats av ett externt program. Är EON nätleverantör finns kan man använda ett program här men det kräver en del manuellt arbete. Se mera under “Experimentella Datafiler”.

Med följande kommando skapas en kostnadssammanställning.

```
Python3 pgart_experimental_calc_costs.py --fromymd <yyyymmdd> --
toymd <yyyymmdd>
```

Ordinära Datafiler

`din-user/pgart/var/local/hourly_rate_yyyymmdd.txt`
För timprisstyrning behövs en fil med “day-ahead”priserna. Systemet kör de program som anges i parametern “`pgm_create_hourly_rates`” för att skapa sådana filer men kan även köra ett helt externt program för att skapa motsvarande timprisfiler.

Utdrag från en standardiserad .
Timme : pris (“i öre eller cent med två decimaler”)/kWh:

```
00: 305.16
06: 255.30
17: 360.55
23: 202.71
```

`din-user/pgart/var/smhi_forecast.txt`
En stor json-fil med utsikterna nedladdat från SMHI. Prognoserna för nästkommande 24 timmar. En mellanfil som bara används när systemet skapar `forecast_short.txt` filen.

`din-user/pgart/var/forecast_short.txt`
En standardiserad fil med prognoser som behövs för beräkna vindkylans påverkan. Programmet skapar filen genom att extrahera data från `smhi_forecast.txt`. Systemet kan också köra ett användarinstallerat program för att skapa filen.

Filen har en rad per timme:
dag_timme : prognos-temperatur, prognos-vind-m/s
`2023-01-17_20: 0.8, 6.8`

`din-user/pgart/var/windchill_calculations.txt`
För posterna i `forecast_short.txt` skapar programmet en ny fil med poster som visar hur mycket lägre temperaturen verkar vara pga vindkyleeffekten.

Filens har en rad per timme: (extra blanka för tydlighets skull)
dag och timme temp vind m/s beräknad vindkyletemp diff verklig- och vindkyletemp

`2023-01-17_20 : 0.8  6.8  : -4.8  -5.6`

`din-user/pgart/var/log/windchill_stats.log`
Här visas värdena som används vid beräkningen av ett nytt börvärde pga vindkyleeffekten.
En lång rad som här delats upp för läsbarhetens skull.

```
2023-02-05_14:02
h: 14  last_indoor_t: 20
forec_t: 1.1  forec_wind: 5.8
factor: 0.8
windchill_t: -4.0  diff: -5.1
wanted_inc: 2.0  got_inc: 2.0 (max 3)
```
Temperaturhöjningen inom gränsvärdena.

`din-user/pgart/var/log/action.log`
Här loggas vissa steg vid bl.a. beräkningen av temperaturen. Bra vid felsökning och för att se programmets beslutsvägar.

`din-user/pgart/var/log/summary_run.log`
En rad per körning. (Om pumpen saknar rumsgivare är temperaturvärdet 200!)

Den riktiga loggfilen inleds med förklaringar.

```
2023-03-13_14:02 000010 No_schema    out:7   room:200 sensor:20.3
pump:22 new:22 hr_rate:0 off      windchill:1 set_hour

2023-03-13_15:02 001000 No_schema    out:8   room:200 sensor:20.6
pump:22 new:21 hr_rate:0 off      windchill:1 reset_hour

2023-03-13_16:02 000001 Set_manually out:8   room:200 sensor:20.2
pump:20 new:20 hr_rate:0 off      windchill:0 off
```

`din-user/pgart/var/settings_status.txt`
I filen sparas data som systemet behöver mellan körningarna.

Experimentella Datafiler

Programmet som gör kostnadssammanställningen för elförbrukningen för timpris och månadspris kräver så klart timförbrukningsdata. För att använda funktionen måste förbrukningsdata alltså finnas i `hourly_comsumption_yyymmdd.txt.`

Står EON för leveransen kan man manuellt ladda ned csv-filer med timförbrukningsdata från sitt konto. Något gratis API finns inte.

`your-user/pgart/var/local/export-2022.csv`
En fil nedladdad från EON. Namnet på de nedladdade filerna blir export-2023.csv, export-2023 (1).csv osv. En fil per dag men filen saknar datumuppgift!

Ett utdrag ur export-2022.csv:
Förbrukning per timme för 2022
"Klockslag";"Förbrukning för 2022(kwh)"
"00:00";2.127
"01:00";1.248

Du måste döpa om de nedladdade filerna till typ `eon_yyyymmdd.csv` så systemet vet dagen.
`your-user/pgart/var/local/eon_yyyymmdd.csv`

`din-user/pgart/var/local/hourly_comsumption_yyymmdd.txt`
En standardiserad fil med timkonsumtionen. Systemet omvandlar EON's csv-filer till detta format.

Utdrag ur en sådan fil:
Timme, förbrukning i kWh
00: 2.127
06: 4.933
17: 2.436
23: 0.902

`din-user/pgart/var/local/monthly_rate.txt`
En standardiserad fil med månadspriser behövs också för att funktionen ska fungera. Systemet skrapar dem från elbruk.se.

Ett utdrag:
år-månad : pris (öre/kWh)
2023-02: 52.03
2023-01: 70.38

`din-user/pgart/var/stat/el_charges_frånyyyymmdd_tillyyyymmdd.txt.`
Skapas av `pgart_experimental_calc_costs.py`

Utdrag ur en sådan fil:
2022-12-17

Kl	kWh	tim pris	medel pris	timpris kostnad	medelpris kostnad	diff
00	2.127	3.05	2.71	6.49	5.77	0.72
06	4.933	2.55	2.71	12.59	13.39	-0.80

```
17  2.436    3.61    2.71     8.78      6.61    2.17
23  0.902    2.03    2.71     1.83      2.45   -0.62
```

Dygnets förbrukning: 57.28 kWh
Kostnad för hela dygnet med timpriset: 166.10 kr, med månadspriset: 155.48 kr, skillnad: -6.40%

Pumpinställning

Aktivera Modbus TCP:

1. Sätt pumpen i driftsläge av.
2. Aktivera serviceläge genom att klicka på hänglåset och mata in 607080.
3. Installationsmenyn, aktivera BMS.
4. Systeminställningar, på sida 2, välj TCP och behåll port 502.

Resursanvändning

Styrprogrammet kör en gång i timmen. En ny temperatur sänds till pumpen endast om värdet behöver ändras.