Pathway towards solving the RTU Case

Vasken, Taso, and more!

TODO

- [x] Decide where to host this file
- [x] Read this file
- [ ] Print or show the figures and tables we can produce today
- [ ] Know where all the scripts are
- [ ] Progressive delivery: each task can be handled independently *as long as* we can define a standard, eg. standard for plotting functions, standard for what a dataframe looks like, the use of haystack, BuildingKit, etc.
- [ ] Run on an existing client
- [ ] Inaugurate RTU taskforce: *requires approuval from JSV*
- Chat with:
  - [ ] Arjun: responsible of SleepCountry and other RTUs
  - [ ] Samuel: knows things
  - [ ] Nicolas: did some RTA automation work
  - [ ] David: AI Onboarding
  - [ ] Steven: M&V process
  - [ ] Claude: Autobot baby scripts to help with mapping or through [[autobot]]
- [ ] Commissioning phase: leave certain zones ON at night; generate better data
- [ ] Fault detection: control intent satisfied?
- [ ] Fault detection: prediction off -> retrain?

Goal

Design a full solution for the RTU case that can be mass-deployed as soon as mapping is complete. No human intervention afterwards except for validating settings and reading reports or logs.

The process begins pre-mapping to decide quickly if the building has potential or has too many issues, mapping is done in [[haystack]], connection to the [[genebility]] pipeline to get utility costs and takes it all the way to the [[conversion-package]]. Aiming to use MPC but can use pre-configured Algos or use a [[genetic-algorithm-for-algo-augmentation]] approach.

The onboarding process templates are already in Asana for hardware installs or via Tridium, so we won't repeat it here more than we need to.

There is also a OneNote session linked to this.

As of 2021-05-19, we are thinking of hosting this information in ClearML after mapping and extraction where the images and tables can be access through the ClearML dashboard, or, alternatively, we can output a PDF or HTML report using those pictures. Not sure if a streamlit dashboard would be good or not here.

This document is organized into different tasks with their respective objectives and needs. Each task also includes a `Script Bank` section which would be scripts that we would be coding and running at that stage. Some of these scripts are currently in production, whereas others are just ideas.

Pre-Mapping Task

Objective: Decide if building meets our requirements and has opportunities for revenue

Need: *Raw* connection data of a few buildings

This process would be run while a BBAI installer is on site with their computer suitcase thing. BBAI personnel may leave device to log a few days if more data is required.

Minimum Requirements

- Check BMS access: manually
- Run `overseer`: data visibility
- Run `curiosity`: writable points
- [TODO need script to] Check variance of data, % points available [if RTUs]

Opportunities for Revenue

- [TODO need approach to] Run on an RTU template -> we're expecting a fan/coils/dampers/temp sensors -> parse raw points into tags (system, zone) -> quick mapping
- [TODO check] Is Fan ON 100% if building not 24/7
- [TODO check] No SP change if building not 24/7
- [TODO check] Is building too good and very well commissioned

Tests to assess if saving potential is {low, medium, high}

Script Bank

```
data_pre_mapping = [
    # See Task 0 in ClearML
    DeployKit.utils_plot.get_heat_map_avg,
    DeployKit.utils_plot.get_heat_map_nan,
    DeployKit.utils_plot.get_msno_nan,
    DeployKit.utils_plot.get_msno_corr_nan,
    DeployKit.utils_plot.get_joyplot_room_temp,
    DeployKit.utils_plot.get_hot_cold_high_low_std_zones,
    DeployKit.utils_plot.get_joy_plot_with_line_dynamic_color,
    DeployKit.utils_plot.get_pairwise_corrs,
    # See MissionControl.stations.comfort
    KitUtils.comfort.temp_power_penalty,
    KitUtils.pythermalcomfort.models.pmv_ppd,
]

data_pre_mapping_TODO = [
    nan_per_point_as_table,  # missing 30% data missing
    nan_per_point_as_table_importance,  # if missing SP, does it matter?
    nan_per_day,
    point_statistics,
    # 20% are constant or missing (and not SP) -> could be trouble;
    #     missing or overwritten
    arundo_process_fdd,
    # summarize findings as counts, top 10 best/worst, zoom in to
    #     a point if need be
    generate_shopping_list_faulty_points_to_fix,
    unsupervised_stuff_cluster,  # "typical day"
    unsupervised_stuff_PCA,  # clusters?
    unsupervised_stuff_UMAP,  # modern clusters?
    setup_fetch_google_pop_times,
    # only for public buildings: retail/commercial/grocery
    script_similar_to_runtime_analysis,  # see with Nicolas
    trends_data_patterns_vs_references,
    # compare data vs a reference that's 24/7 and one with OP
    quick_report_generation,  # pneumatic controls suck
]
```

Mapping Task

Objective: Tag the salient building points to our Haystack tags

Need: Haystack finalized for RTUs and a quick templated method to process the building

Once mapping is completed, we are *quite* committed to the project and will want to see it through. This is even more particularly true for the RTU case if the hope is to only have this part require human input – other parts require only validation and verification. At this stage, we can again decide to proceed (as-is, conditionally) or reject the building (painful).

Use of an RTU archetype template

We have seen different approaches to the RTU problem and have designed Algos and methods to control them. Can we also design templates for RTU systems? How many different combinations are there anyway?

Given the raw point names, can we use a easy customizable string parser to sort, filter and tag all the points in the building? We can add this to Needle.

Example: think of a date parser, given string `'2021-09-09'`, you use `datetime.strptime('%Y-%m-%d)` to strip away the year, month and day. Could we do something similar like `%z` would be a zone, `%t` would be a tag and we can define a dictionary for that specific building or client like `{'z_temp': 'ROOM_TEMP', 'AO12': 'COOL_STG_1'}`, and so on. Dictionary can be based on a client: SleepCountry, Walmart, Loblaws, Sobeys, and/or based on a vendor. We would also need to include "writing access level", i.e. which are inputs, outputs, setpoints, and so on.

Work can be linked with Claude Demers-Belanger who's working on Autobot.

- Control methods:
  - ON/OFF all components
  - Setpoint override
  - *Load shed* operation mode
- Needle: RTU Mode TODO
  - `haystack_tags`: custom string parser to haystack tags
  - `equipment_information`: gas heat, electric cool, airflow total, airflow fresh
  - `validate_power_information`: input bills… and some magic

Script Bank

```
data_post_mapping = [
    anaylsis_per_tag,  # for all fans: run ...
    analysis_per_zone,  # clusters, scatter plots, trendlines
    correlation_plots,
    # between zones; zone and HVAC; zone and solar radation / OAT
    faulty_behaviour_stuck_thermostat,  # temp stuck cold, heat always ON
    comfort,  # historical/new, SP based, PMV/PPD
    tigrimite,
    sensor_flat_spike,
    dT_analysis,  # heatmap, slopes; filtered by day/night occ/unocc
    density_based_methods,  # for movements
    hvac_power_vs_oat,
    # ReLU (_/) looking thing -> neutral zone temperature
    joyplot_temperatures,
    setpoint_check_via_std_range_ROOM_TEMP,
    # night setback? or nah?  -> try fitting two gaussians
    equipment_cycling_check,
    # define baseline, show high numbers -> feed Lithium
    #     (cycling reduction algo, load shedding)
    [*data_pre_mapping],
    # should also run all the functions in pre-map stage
]
```

Algo Task

Objective: Starting from a haystack tagged building and some data, generate the BuildingKit config file, validate, launch everything

Need: Script to generate the 99.9% completed BuildingKit config file

This sort of script is being worked on by David and Robert:

`building_kit_config_2_validate = building_kit_configurator(haystack, some_date)`

Not sure about status right now, I think they're still waiting for [[haystack]] to be fully deployed). There's also a BuildingKit configurator streamlit dashboard, but this approach should be Option 2 for RTUs.

Perhaps we would need a service/trigger to say "data is valid, proceed!", and this function would generate the JSON file and notify Samuel Desmeules to validate it.

Once the BuildingKit config file is validated, we can launch everything we have on the ControlKit or [[calvin]] side of things and the conventional approach is considered complete.

Modeling Task

Objective: Assess if trained model is good or not.

Need: Have multiple lenses to see performance at a high-level (store by store: KPIs), mid-level (zone by zone: heatmap) and low-level (specific zone: trendlines, scatter plots).

Objective 2: Can we design simpler physics-based models to capture enough of the trend and use it immediately?

Need 2: Finalize the RC modeling approach, think of transfer learning methods and/or include *a priori* assumptions based on Standards and engineering judgement or experience.

For the models that are in production now. We need to beef up the validation methods. From simple visualizations to validating if the model handles test cases correctly (heating will result in rise of temperature) to looking into model's mapped domain and distribution shifts.

For the resistance-capacitance (RC) model in Impetus personal repo, it needs some rework to rely on TensorFlow's (or other) automatic differentiation methods. Not too happy as-is yet. There's still quite a bit of work left here to also link it strongly with BuildingKit and Haystack. We're always against time.

For the transfer learning part, can we use information from other RTU buildings in the same region or of a same archetype? There was some work done trying to do this in the One Model to Rule them All project. Perhaps we should push this sort of idea further blending in physics-informed neural nets or Universal ODEs.

Script Bank

```
model = [
    select_model(rc, ml... based on data available / effort required),
    model_train_metrics_plots,
    validate_model_metrics(test_cases),
    validate_model_prediction_plots(test_cases),
    model_report,  # hosted in ClearML? html file? pdf? emailed?
]
```

Optimal Control Task

Objective: Hands-free optimal control approach.

Need: Easy way for ONB to validate approach and assess if results are good/correct.

Working on Gargamel to split the optimizer from the model. This approach is still the simplest method and most flexible. Working on assuring convergence and speed.

For Walmart, I used an MPC method, however, I didn't assume the controls were discrete since we lumped many zones together. Full explanation of the method is in [[project_impetus]]. As a discrete-controls problem, MPC isn't as fast since it has to use a branch-and-bound type method. We're also looking into using RL, but the problem with RL is that if the cost/fitness/reward function changes, you need to retrain the model. There are multi-objective approaches but we haven't tested it.

There's still quite a bit of work when it comes to defining objective functions and constraints in a clean way which depends on getting the equipment use (energy, power, $, CO2-eq), cycling rates or rate of change, comfort (temperature in setpoint, PMV/PPD, rate of change of temperature), etc etc.

Then we need to have a conversation about uncertainty and robustness.

And another about indoor air quality, but might not be too imposed in an RTU building.

```
control = [
    get_model,
    DataKit.encode_decode,
    # encode status: heat stg 2 > heat stg 1 > fan
    set_up_config,  # comfort weight, parameters; setpoints for comfort
    validate_config,  # print it, stamp it; confirm with client?
    commissioning_active_learning,  # algo to modify SP to get better data
]
```

Monitoring Task

Objective: We need better monitoring! And not spam inboxes! Also need better release methods with stronger confirmation.

Need: . . .

Monitoring isn't only about faults and scripts crashing. We also need to check if we're performing well relative to a baseline.

We also need to, not only confirm that a control action is written, but if the control intent was satisfied: maintain comfort, heat/cool the space in the expected time frame. Otherwise, other our model/algo is wrong or something else went wrong.

Script Bank

```
analysis_continuous = [
    setup_monitoring_plugins,
    check_actual_equals_sent,  # defective sensor/equipment, failed write
    prediction_vs_actual_live,
    guardian_module,  # include Robert
    savings_to_date_HDD_CDD_normalized,
    # compared to some baseline, HDD/CDD normalized
]
```

Reporting and Conversion Task

Objective: Wouldn't it be nice if we had reports be generated weekly?

Need: Presentation template and what would the client want to see?

We need BBAI performance KPIs.

Are we reducing equipment runtime while respecting thermal comfort? And if yes, is our solution improving, stable or declining?

We have a RTA tool. A low-hanging fruit is to execute it in the background every week (frequency is configurable) and raise a flag if necessary. For example, BBAI reduces runtime 20% last week and 12% this week. Should we retrain the model? Change the algos parameters? Is there a defective equipment? Did a module (writer, HERMES, MPC) crash?

Script Bank

```
report = [
    runtime_analysis,
    opportunity_summary,
    savings_to_date_m&v_process,
    year_projection,
    capex_projection,
    # by reducing cycling, we increased the useful lifetime
    #     of equipment -> $$$
]
```