Reinforcement Learning based HVAC Optimization in Factories

Debmalya Biswas
Philip Morris Products S.A.
Lausanne, Switzerland
debmalya.biswas@pmi.com

ABSTRACT

Heating, Ventilation and Air Conditioning (HVAC) units are responsible for maintaining the temperature and humidity settings in a building. Studies have shown that HVAC accounts for almost 50% energy consumption in a building and 10% of global electricity usage. HVAC optimization thus has the potential to contribute significantly towards our sustainability goals, reducing energy consumption and CO_2 emissions. In this work, we explore ways to optimize the HVAC controls in factories. Unfortunately, this is a complex problem as it requires computing an optimal state considering multiple variable factors, e.g. the occupancy, manufacturing schedule, temperature requirements of operating machines, air flow dynamics within the building, external weather conditions, energy savings, etc. We present a Reinforcement Learning (RL) based energy optimization model that has been applied in our factories. We show that RL is a good fit as it is able to learn and adapt to multi-parameterized system dynamics in real-time. It provides around 25% energy savings on top of the previously used Proportional-Integral-Derivative (PID) controllers.

CCS CONCEPTS

• **Applied computing → Multi-criterion optimization and decision-making**; • **Computing methodologies → Reinforcement learning**.

KEYWORDS

Energy Optimization, HVAC, Sustainability, Reinforcement Learning, Machine Learning

ACM Reference Format:
Debmalya Biswas. 2020. Reinforcement Learning based HVAC Optimization in Factories. In *The Eleventh ACM International Conference on Future Energy Systems (e-Energy '20), June 22–26, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3396851.3402363

1 INTRODUCTION

Heating, Ventilation and Air Conditioning (HVAC) units are responsible for maintaining the temperature and humidity settings in a building. We specifically consider their usage in factories in this work, where the primary goal of the HVAC units is to keep the temperature and (relative) humidity within the prescribed manufacturing tolerance ranges. This needs to be balanced with energy savings and CO_2 emission reductions to offset the environmental impact of running them.

Given their prevalence in not only factories, but homes and office buildings; any efficient control logic has the potential of making significant contributions with respect to their environmental impact. Unfortunately, given the complexity of HVAC units, designing an efficient control logic is a hard optimization problem. The control logic needs to consider multiple variable factors, e.g. the occupancy, manufacturing schedule, temperature requirements of operating machines, air flow dynamics within the building, external weather conditions, energy savings, etc.; in order to decide how much to heat, cool, or humidify the zone.

The HVAC optimization literature can be broadly divided into two categories: (i) understand the recurring patterns among optimization parameters to better schedule HVAC functioning, and (ii) build a simulation model of the HVAC unit and assess different control strategies on the model - to determine the most efficient one. Examples of the first category include [3, 6] which employ a building thermodynamics model to predict the buildings' temperature evolution. Unfortunately, this is not really applicable for our factories where the manufacturing workload varies every day, and there is no schedule to be predicted. It is also worth mentioning that most such models only consider one optimization parameter at a time, i.e. control heating / cooling to regulate the temperature; whereas in our case, we need to regulate both the temperature and humidity simultaneously to maintain the optimal manufacturing conditions. The second category of model-based approaches applies to both PID and RL controllers. PID (Proportional-Integral-Derivative) controllers [7] use a control loop feedback mechanism to control process variables. Unfortunately, PID based controllers require extensive calibration with respect to the underlying HVAC unit, to be able to effectively control them. [2] outlines one such calibration approach for PIDs based on a combination of simulation tools.

Reinforcement Learning (RL) [8] based approaches [4, 9] have recently been proposed to address such problems given their ability to learn and optimize multi-parameterized systems in real-time. An initial (offline) training phase is required for RL based approaches, as training an RL algorithm in live settings (online) can take time to converge leading to potentially hazardous violations as the RL agent explores its state space. [4, 9] outline solutions to perform this offline training based on EnergyPlus based simulation models of the HVAC unit. EnergyPlus [5] is an open source HVAC simulator from the US Department of Energy that can be used to model both energy consumption - for heating, cooling, ventilation, lighting and plug and process loads and water use in buildings. Unfortunately, developing an accurate EnergyPlus based simulation model of an

e-Energy '20, June 22–26, 2020, Virtual Event, Australia

ACM ISBN 978-1-4503-8009-6/20/06…$15.00
https://doi.org/10.1145/3396851.3402363

HVAC unit is a non-trivial, time consuming and expensive process; and as such can be blocker for their use in offline training.

In this work, we propose an efficient RL based HVAC optimization algorithm that is able to learn and adapt to a live HVAC system in weeks. The algorithm can be deployed independently, or as a 'training module' to generate data that can be used to perform offline training of an RL model - to further optimize the HVAC control logic. This allows for a speedy and cost-effective deployment of the developed RL model. In addition, the model output in [4, 9] is the optimal temperature and humidity setpoints, which then rely on the HVAC control logic to ensure that the prescribed setpoint is achieved in an efficient manner. In this work, we propose a more granular RL model, whose output is the actual valve opening percentages of the Heating, Cooling, Re-heating and Humidifier units. This enables a more self-sufficient approach, with the RL output bypassing (removing any dependency and making redundant) any in-built HVAC control logic - allowing for a more vendor (platform) agnostic solution.

The rest of the paper is organized as follows: Section 2 introduces the underlying framework, providing an RL formulation of the HVAC optimization problem. Section 3 outlines the RL logic that is initially deployed to generate the training data for offline training, leading to the (trained) RL model in Section 4 - providing the recommended valve opening percentages in real-time. In Section 5, we provide some benchmarking results of the RL model that has been deployed in one of our factory zones. Initial studies show that we are able to achieve 25% energy efficiency over the previously existing PID controller. Section 6 concludes the paper and provides directions for future work.

2 PROBLEM SCENARIO

2.1 HVAC

Figure 1 shows the energy balance of the HVAC unit. In a simplified way, the HVAC unit has to bring the mix of the Recirculation air and the Fresh air to the temperature and humidity needed to maintain the area temperature and (relative) humidity at the required level. It is easy to monitor the theoretical energy needed by performing the difference between the energy of outgoing air and the incoming air. Comparing this with the amount of energy needed for unit gives the energy efficiency of the HVAC unit. The energies flows can be determined based on the flow of media (air, hot water, cold water, steam) and the temperature difference between the supply and return of the media. And the consumed electrical energy.

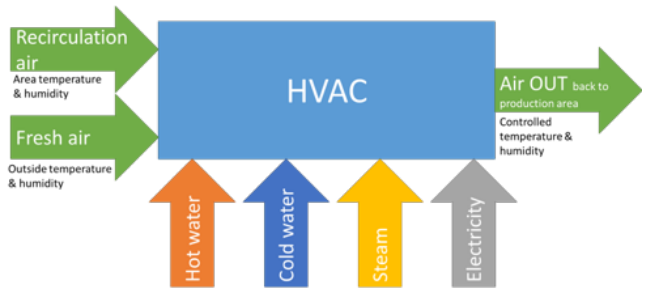

Figure 1: HVAC Control

2.2 Reinforcement Learning (RL)

RL refers to a branch of Artificial Intelligence (AI), which is able to achieve complex goals by maximizing a reward function in real-time. The reward function works similar to incentivizing a child with candy and spankings, such that the algorithm is penalized when it takes a wrong decision and rewarded when it takes a right one - this is reinforcement. For a detailed introduction to RL, the interested reader is referred to [8].

Let us take the analogy of a video game. At any point in the game, the player has a set of available actions, within the rules of the game. Each action contributes positively or negatively towards the player's endgoal of winning the game. For instance, the RL model might compute that running right will return +5 points, running left none, and jumping −10 (as it will lead to the player dying in the game).

2.3 RL Formulation

We now map the scenario to our HVAC setting. At any point in time, a factory zone is in a state characterized by the temperature and (relative) humidity values observed inside and outside the zone.

The game rules in this case correspond to the temperature and humidity tolerance levels, which basically mandate that the zone temperature and humidity values should be within the range: 19−25 degrees and 45 − 55% respectively. The set of available actions in this case are the Cooling, Heating, Re-heating and Humidifier valve opening percentages (%).

To summarize, given the zone state in terms of the (inside, outside and supply) temperature and humidity values, the RL model needs to decide by how much to open the Cooling, Heating, Re-heating and Humidifier valves. To take an informed decision in this scenario, the RL model needs to first understand the HVAC system behavior, in terms say how much zone temperature drop can be expected by opening the Cooling valve by X%?

Once the RL model understands the HVAC system behavior, the next step is to design the control strategy, or 'Policy' in RL terminology. For instance, the RL model now has to choose whether to open the Cooling value to 25% when the zone temperature reaches 23 degrees, or wait until the zone temperature reaches 24 degrees before opening the Cooling valve to 40%. Note that the lower it opens the valve and the longer it waits, it contributes positively towards lowering the energy consumption. However, it then runs the risk of violating the temperature / humidity tolerance levels as the outside weather conditions are always unpredictable. The above probabilities are quantified by a reward function (Equation 1) in RL terminology, which assigns a reward to each possible action based on the following three parameters:

$$Reward(a) = (w_1 \times SC) - (w_2 \times EC) - (w_3 \times TV) \quad (1)$$

A control strategy is to decide on the weightage of the three parameters: Setpoint Closeness (SC), Energy Cost (EC), Tolerance Violation (TV). The Energy Cost is captured in terms of electricity consumption and CO_2 emission. For instance, a 'safe' control strategy would assign a very high negative weightage (penalty) to Tolerance Violations, ensuring that they never happen, albeit at a higher Energy Cost. Similarly, an 'energy optimal policy' would

prioritize energy savings over the other two parameters. Setpoint Closeness encourages a "business friendly" policy where the RL model attempts to keep the zone temperature as close as possible to the temperature / humidity setpoints, implicitly reducing the risk of violations, but at a higher Energy Cost. We opt for a "balanced" control policy which maximizes Energy Savings and Setpoint Closeness, while minimizing the risk of Tolerance Violations.

The RL formulation described above is illustrated in Figure 2.

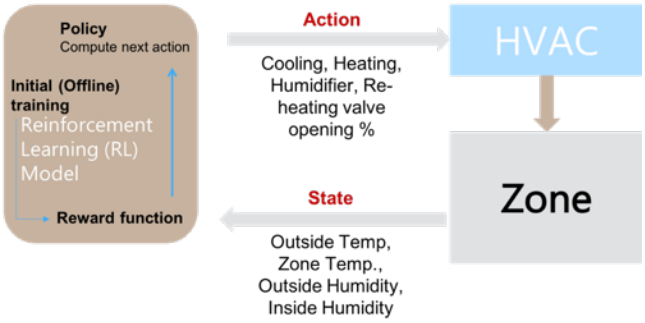

Figure 2: HVAC Reinforcement Learning formulation

3 RL BASED HVAC OPTIMIZATION

We outline a RL algorithm that outputs how much to open the Heating, Cooling, Humidifier and Re-heating valves at time t, based on the current Indoor Temperature and Humidity (at time t), and the previous Heating, Cooling, Humidifier, Re-heating valve opening percentage values, Indoor Temperature and Humidity values at time $t-1$. The *rl_hvac* function runs in real-time computing the new valve opening percentages every 1 min.

```
#Input
it   #(current) Indoor temperature at t
ih   #(current) Indoor humidity at t
pit  #(prev) Indoor temperature at t-1
pih  #(prev) Indoor humidity at t-1
pHV  #(prev) Heating valve opening% at t-1
pCV  #(prev) Cooling valve opening% at t-1
pUV  #(prev) Humidifier valve opening% at t-1
pRV  #(prev) Re-heating valve opening% at t-1

#Output - Valve opening% for this iteration
oHV  #Heating valve opening% at t
oCV  #Cooling valve opening% at t
oUV  #Humidifier valve opening% at t
oRV  #Re-heating valve opening% at t

#static variables
hiW  #Heating step increment
ciW  #Cooling step increment
uiW  #Humidifier step increment
riW  #Re-heating step increment
```

```
#invoked every 1 min
def rl_hvac
    (it, ih, pit, pih, pHV, pCV, pUV, pRV):

    #initialize to previous values
    oHV = pHV
    oCV = pCV
    oUV = pUV
    oRV = pRV
    #Heat and Humidify
    if (it < 22.0) and (ih < 50.0):
        if (it < pit):
            inc = hiW + abs((it - pit)/0.1)
            oHV += inc
        if (ih < p_ih):
            inc = uiW + abs((ih - pih)/0.1)
            oUV += inc
        oRV = 0.0
        oCV = 0.0
    # Cool and Re-heat
    if (it > 22.0) and (ih > 50.0):
        if (it > pit):
            inc = ciW + abs((it - pit)/0.1)
            oCV += inc
        if (ih > p_ih):
            inc = riW + abs((ih - pih)/0.1)
            oRV += inc
        oHV = 0.0
        oUV = 0.0
    # Heat, Cool and Re-heat
    if (it < 22.0) and (ih > 50.0):
        if (it < pit):
            inc = hiW + abs((it - pit)/0.1)
            oHV += inc
        if (ih > pih):
            inc = (ciW + riW)/2)
            + abs((ih - pih)/0.1)
            oCV += inc
            oRV += inc
        oUV = 0.0
    # Cool and Humidify
    if (it > 22.0) and (ih < 50.0):
        if (it > pit):
            inc = ciW + abs((it - pit)/0.1)
            oCV += inc
        if (ih < p_ih):
            inc = uiW + abs((ih - pih)/0.1)
            oUV += inc
        oHV = 0.0
        oRV = 0.0

    return oHV, oCV, oUV, oRV
```

The RL logic can be explained as follows: Recall that the temperature and (relative) humidity setpoints that we would like to maintain are 22 degrees and 50%; with allowed tolerance ranges of ±3 degrees and ±5% respectively. At every iteration (1 min), the *rl_hvac* function determines which valve(s) to open based on the control logic outlined in Figure 3:

Temperature / (Rel.) Humidity	*> 22.0* degrees	*< 22.0* degrees
> 50%	Cool and Re-heat	Heat, Cool and Re-heat
< 50%	Cool and Humidify	Heat and Humidify

Figure 3: HVAC control logic

Knowing which valve(s) to open, how much to open each valve depends on the reward value, computed as a measure of the 'effectiveness' of the previous (output) valve openings. For instance, let us assume that the indoor temperature is currently 20.5 degrees (below the temperature setpoint), which implies that the Heating valve needs to be opened. During the previous iteration, the indoor temperature was also below the setpoint, say 21.0 degrees, leading the *rl_hvac* function recommendation to open the Heating valve at say 15%.

Given that the current indoor temperature is even lower (20.5 degrees), we infer that the previous Heating valve opening was not sufficiently effective - assigning it a negative reward - and heating more by an amount proportional to the difference between the current and previous indoor temperature. The behavior of the other valves can be explained analogously. This is reinforcement and ensures that the valves are able to efficiently balance the 'setpoint closeness' and 'energy cost' parameters of the reward function (Equation 1).

The remaining parameter of the reward function is the 'tolerance violation' where a penalty needs to be imposed if the indoor temperature / humidity violates the allowed tolerance ranges. A violation in our case implies that the respective valve(s) needs to react faster. This is accommodated by the step increment constants: h_iW, c_iW, u_iW, r_iW. We adjust them in an offline fashion such that their values are adapted if the number of tolerance violations exceeds a certain threshold during a given period.

```
#Check tolerance violations
# and adapt respective increment values
if count((it > 25.0) > n)
    c_iW += 1.0
if count((it < 22.0) > n)
    h_iW += 1.0
if count((ih < 45.0) > n)
    u_iW += 1.0
if count((ih > 55.0) > n)
    r_iW += 1.0
```

4 CONTINUOUS RL OPTIMIZATION

In this section, we extend the RL model to accommodate 'long term rewards', quantified by the Q-value in RL terminology. (Recall that the rewards function in the RL algorithm outlined in the previous section is stochastic, in the sense that it only depends on the last state.) Q-value for a state-action pair (s, a) is defined as a weighted sum of the expected reward values of all future steps starting from the current state s, given that action a is taken at state s.

To accommodate 'long term rewards', we extend our original problem to a continuous space setting. Each episode in this setting corresponds to a period when the indoor temperature and (or) humidity starts moving away from their respective setpoints, to the time that the indoor conditions return to their respective setpoint values - as a result of opening the relevant valve(s).

Let us now focus on one such episode (in a continuous space setting). Given that the stochastic RL algorithm (in Section 3) always starts opening the valves at 0.0%, the temperature and (or) humidity deviation from the setpoint keeps increasing, until the valve opening percentage reaches the tipping point, after which the deviation starts decreasing again until it becomes 0. This episodic behavior is illustrated in Figure 4.

For the sake of simplicity, we have only shown the Temperature - Cooling curve, however an analogous behavior can be anticipated for the other scenarios, including those involving Humidity. The energy cost in Figure 4 corresponds to the shaded region. Given this behavior, it is easy to see that if we knew the Cooling tipping point beforehand to be 22.3 degrees, we could have opened the Cooling valve earlier to the tipping point - leading to a lower energy cost (depicted by dashed shaded region in Figure 4). The caveat here is that the tipping point needs to be estimated properly for all the valves, otherwise opening a valve to more than the tipping point percentage might actually lead to a higher energy cost.

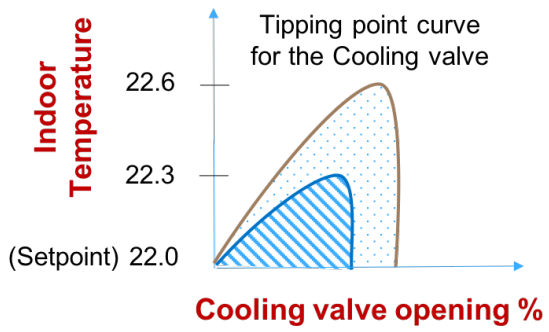

Figure 4: Valve tipping point computation

In the sequel, we show how the data generated by the RL algorithm in Section 3 can be used as training data, to develop a model to predict the 'tipping point' of the valves for each state of the factory zone. The (Section 3) RL algorithm data can be considered as consisting of the following input and values, for each time point t: (Indoor Temperature, Indoor Humidity, Heating valve opening%, Cooling valve opening%, Humidifier valve opening%, Re-heating valve opening%). We apply the filtering below (only illustrated for humidity) on this data - to extract the training data:

```
# loop through rows of the generated data
for i in range(1, dataset.shape[0]):
    cRow = dataset.iloc[i]  # current row
    pRow = dataset.iloc[i - 1]  # previous row
    if cRow['InHumidity'] > 50.0 and
    cRow['InHumidity'] < pRow['InHumidity']:
        # write row to training data
    if cRow['InHumidity'] < 50.0 and
    cRow['InHumidity'] > pRow['InHumidity']:
        # write row to training data
```

The filter aims to identify 'episodes', focusing on the time points where the indoor humidity starts converging towards the setpoint, after a period of increased deviation from it. Needless to say, the valve opening percentages at these time points correspond to the 'tipping points' for those states. We train four models: *h_model*, *u_model*, *r_model*, *c_model* based on this training (filtered) data, to predict the 'tipping point' values of the Heating, Humidifier, Re-heating, Cooling valves respectively. The trained models are then embedded in our RL algorithm as follows (we only show the 'Heat and Humidify' block for illustration):

```
#Heat and Humidify
if (it < 22.0) and (ih < 50.0):
    if (oHV == 0.0):
        oHV = h_model.predict(it, ih)
    elif (it < pit):
        inc = hiW + abs((it - pit)/0.1)
        oHV += inc
    if (oUV == 0.0):
        oUV = u_model.predict(it, ih)
    elif (ih < p_ih):
        inc = uiW + abs((ih - pih)/0.1)
        oUV += inc
    oRV = 0.0
    oCV = 0.0
```

With this update, the RL algorithm is able to bootstrap the valve opening percentages, so that each episode will start with the respective 'tipping point' values (instead of starting from 0.0%) provided by the trained models - leading to a lower energy cost (Figure 4).

5 BENCHMARKING RL VERSUS PID

The proposed RL model has been deployed in a zone of our factory in Romania. The designated zone has five (similar) HVAC units, where the schematics of a HVAC unit is illustrated in Figure 5.

For this zone, we first present (Figure 6) the indoor conditions and HVAC valve opening percentages of running the HVAC with a PID controller for a week (around 10, 000 readings, corresponding to a reading every minute).

For the same zone, we then ran the benchmarking with the HVAC units controlled by the RL model. We ensured that the manufacturing workload was similar for both scenarios. We first ran the RL algorithm outlined in Section 3 for a week. Following this, the filter outlined in Section 4 was applied to generate training data. The four models: *h_model*, *u_model*, *r_model*, *c_model* were then trained using AWS Sagemaker Autopilot [1] to produce XGBoost based Regression models with a validation error (MSE from hyper-parameter tuning) of around 25.0. With the models trained, we ran the RL algorithm presented in Section 4 for another week (with a similar production workload). The results are presented in Figure 7.

Comparing the average valve opening percentages (highlighted by the red bounding boxes in Fig. 6 and Fig. 7), we can see that all the RL based valve opening percentages are lower; ranging from 10% savings for the Heating valve to almost 45% savings for the Re-heating valve - leading to 25% savings on average.

6 CONCLUSION

In this work, we considered the problem of HVAC energy optimization in factories, which has the potential of making a significant environmental impact in terms of energy savings and reduction in CO_2 emissions. To address the problem complexity, we outlined a RL based HVAC controller that is able to learn and adapt to real-life factory settings, without the need for any offline training. To the best of our knowledge, this is one of the first works to report on a live deployment of an RL-HVAC model, in an actual factory. We provided benchmarking results that show the potential to save upto 25% in energy efficiency.

Note that we have considered energy savings as proportional to the valve opening percentages (the lower the better). In reality, the energy consumption and CO_2 emissions of the different valves may not be proportional, i.e. depending on the underlying mechanism, opening the Heating and Cooling valves by the same percentage may not consume the same amount of energy, and emit the same amount of CO_2. We leave this as future work to adapt the RL logic to accommodate the energy consumption and CO_2 emissions aspects.

ACKNOWLEDGMENTS

I would like to thank Christian Hochreutener, Petrica Stancu, Mihnea-Alexandru Manea, Corrado Iorizzo, Antonio Bruno and Fady Fadel for their support in deploying the models in our factories.

REFERENCES

[1] Amazon. 2020. *Amazon SageMaker Autopilot.* Retrieved May 25, 2020 from https://aws.amazon.com/sagemaker/autopilot/
[2] Carlos Blasco, Javier Monreal, Ignacio Benítez, and Andrés Lluna. 2012. Modelling and PID control of HVAC System according to Energy Efficiency and Comfort Criteria. *Sustainability in Energy and Buildings* 12 (2012), 365–374.
[3] Yudong Ma, Francesco Borrelli, Brandon Hencey, Brian Coffey, Sorin Bengea, and Philip Haves. 2012. Predictive Control for the Operation of Building Cooling Systems. *IEEE Transactions on Control Systems Technology* 20, 3 (2012), 796–803.
[4] Takao Moriyama, Giovanni De Magistris, Michiaki Tatsubori, Tu-Hoa Pham, Asim Munawar, and Ryuki Tachibana. 2018. Reinforcement Learning Testbed for Power-Consumption Optimization. In *Proceedings of the 18th Asia Simulation Conference (AsiaSim).* 45–59.
[5] US Department of Energy. 2020. *EnergyPlus.* Retrieved May 25, 2020 from https://energyplus.net/
[6] Frauke Oldewurtel, Alessandra Parisio, Colin N. Jones, Manfred Morari, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Katharina Wirth. 2010. Energy Efficient Building Climate Control using Stochastic Model Predictive Control and Weather Predictions. In *Proceedings of the American Control Conference.* 5100–5105.
[7] F. Peacock. 2020. *An Idiot's Guide to the PID Algorithm.* Retrieved May 25, 2020 from https://www.pidcontrol.net/index.html
[8] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
[9] Tianshu Wei, Yanzhi Wang, and Qi Zhu. 2017. Deep Reinforcement learning for building HVAC control. In *Proceedings of the 54th Annual Design Automation Conference (DAC).* 1–6.

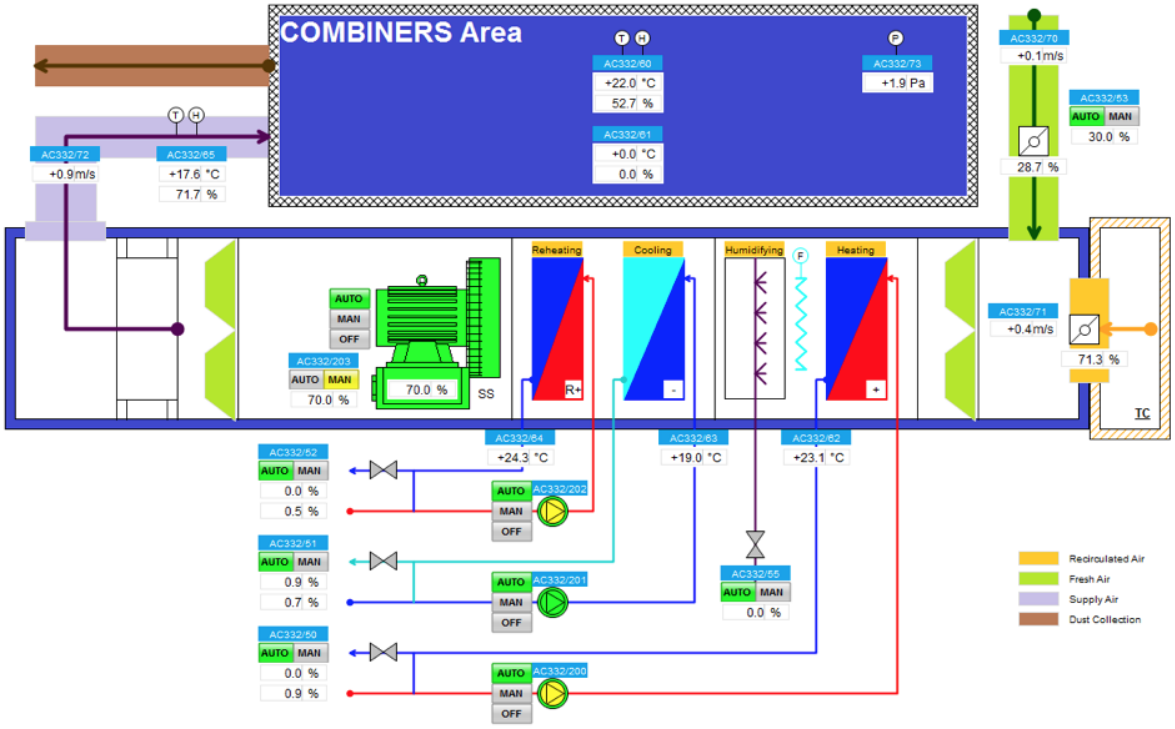

Figure 5: HVAC schematics

	Room Temperature ValueY	Room Humidity ValueY	Heating Valve ValueY	Cooling Valve ValueY	Humidifier Valve ValueY	Reheat Valve ValueY
mean	22.027662	50.037363	9.372753	10.498550	30.770681	20.468397
std	0.066739	0.922815	20.184126	14.498391	29.895784	31.726365
min	21.743055	46.518517	0.839120	0.607639	0.000000	0.491898
25%	21.990740	49.479164	1.012731	0.839120	0.000000	0.665509
50%	22.019676	50.086803	1.215278	5.049190	29.300000	1.157407
75%	22.063078	50.665508	4.166667	17.100695	49.960000	34.143517

Figure 6: PID based HVAC control readings

	Room Temperature ValueY	Room Humidity ValueY	Heating Valve ValueY	Cooling Valve ValueY	Humidifier Valve ValueY	Reheat Valve ValueY
mean	22.009329	49.529951	8.446135	7.832123	25.714461	11.139734
std	0.152894	1.419767	12.799755	9.436951	23.573753	21.848293
min	21.151621	44.531250	0.839120	0.578704	0.000000	0.491898
25%	21.931133	48.726852	0.925926	0.781250	2.157418	0.578704
50%	22.019676	49.469906	1.591435	3.327546	24.203690	0.954861
75%	22.092014	50.173611	11.400463	12.413195	38.818600	9.143518

Figure 7: RL based HVAC control readings