Interaction and Behaviour Evaluation for Smart Homes: Data Collection and Analytics in the ScaledHome Project

Matteo Mendula
Dept. of Computer Science and Engineering - University of Bologna
Bologna, Italy
matteo.mendula@unibo.it

Siavash Khodadadeh
Salih Safa Bacanlı
Sharare Zehtabian
Hassam Ullah Sheikh
Dept. of Computer Science University of Central Florida
siavash.khodadadeh@knights.ucf.edu
bacanli@knights.ucf.edu
sharare.zehtabian@knights.ucf.edu
hassam.sheikh@knights.ucf.edu

Ladislau Bölöni
Dept. of Computer Science University of Central Florida
Orlando, FL
lboloni@cs.ucf.edu

Damla Turgut
Dept. of Computer Science University of Central Florida
Orlando, FL
turgut@cs.ucf.edu

Paolo Bellavista
Dept. of Computer Science and Engineering - University of Bologna
Bologna, Italy
paolo.bellavista@unibo.it

ABSTRACT

The smart home concept can significantly benefit from predictive models that take proactive management operations on home actuators, based on users' behavior evaluation. In this paper, we use a small-scale physical model, the ScaledHome-2 testbed, to experiment with the evolution of measurements in a suburban home under different environmental scenarios. We start from the observation that, for a home to become smart, in addition to IoT sensors and actuators, we also need a predictive model of how actions taken by inhabitants and home actuators affect the internal environment of the home, reflected in the sensor readings. In this paper, we propose a technique to create such a predictive model through machine learning in various simulated weather scenarios. This paper also contributes to the literature in the field by quantitatively comparing several machine learning algorithms (K-nearest neighbor, regression trees, Support Vector Machine regression, and Long Short Term Memory deep neural networks) in their ability to create accurate and generalizable predictive models for smart homes.

CCS CONCEPTS

• **Computing methodologies** → **Simulation environments**; *Machine learning approaches*; • **Hardware** → **Temperature optimization**.

KEYWORDS

IoT; energy; optimization; simulation; scaling

ACM Reference Format:
Matteo Mendula, Siavash Khodadadeh, Salih Safa Bacanlı, Sharare Zehtabian, Hassam Ullah Sheikh, Ladislau Bölöni, Damla Turgut, and Paolo Bellavista. 2020. Interaction and Behaviour Evaluation for Smart Homes: Data Collection and Analytics in the ScaledHome Project . In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3416010.3423227

MSWiM '20, November 16–20, 2020, Alicante, Spain

ACM ISBN 978-1-4503-8117-8/20/11…$15.00
https://doi.org/10.1145/3416010.3423227

1 INTRODUCTION

The recent improvements and cost efficiency of IoT devices allowed the augmentation of homes with Internet-connected sensors that can measure various environmental parameters and actuators. Through these domotic sensors and actuators, the users can remotely control many home features, such as doors, window shutters, heating, and air conditioning units, or dehumidifiers. However, to make a home really "smart", we need to ensure that the actions taken by these devices are efficient, synergic, and in line with the inhabitants' preferences.

A precondition of making the right control decisions is the existence of a predictive model of the home - how will the values measured by the sensors change if specific actions are taken? However, developing a closed-form mathematical model which, given a specific home in particular circumstances, predicts the temporal evolution of the sensor readings is a challenging cross-domain modeling task. Developing machine learning models is a compelling alternative that, at least, switches the modeling difficulty with a data collection challenge.

At first glance, gathering data from an IoT augmented home is straightforward: we can log sensor readings and actuator actions. However, the data collected in this fashion will be tied to and

affected by the specific elements of the considered home, environment, geographic area, and inhabitants' preferences. To gather data for a different home, we need to build a home in the target geographic area and put it through several scenarios where some of the inputs (such as weather) are not under our control. Thus, gathering real-world smart home data is expensive and time-consuming. Besides, the publicly available datasets are relatively small and specific to the peculiar home where they were collected.

The ScaledHome project aims to create reusable and small-scale experimental testbeds that replicate a full-scale suburban home, including its sensors and actuators. It provides an environmental modeling system that allows us to position the home in a given geographic location and a weather pattern. For instance, it can model a summer in Orlando, a fall in Milan, or a winter in Norway. The work described in this paper centers around the sensor augmentation and control of the remotely controllable ScaledHome-1 [12] and ScaledHome-2 [4] testbeds, as better detailed in the following parts of the paper, as well as the collection of data and the creation of predictive models using machine learning techniques.

The main contributions of the work done are related not only with the IoT and the smart home area but also with a novel simulation approach and to the application of machine learning techniques to a domestic energy efficiency scenario. To better match the real environmental metrics, we run multiple simulations of real-world scenarios. The technical contributions of this paper are as follows:

- The first presentation of the physical and control implementation of the ScaledHome-2 system.
- The newly designed architecture of our ScaledHome-2 framework environment, by focusing on our framework components that enable the remote performance of experiments.
- A novel technique to create predictive machine learning models for a specific physical home modeled through our testbed. We compare the performance of our models depending on the different machine learning regressors adopted to train them, i.e., K-nearest neighbor, regression trees, SVM regression, and long short term memory (LSTM) deep neural networks.

The rest of the paper is organized as follows. We present the related work in Section 2. We provide a detailed description of our smart home prototype and methods in Section 3. In-the-field experimental evaluation is presented in Section 4, and Section 5 concludes the paper.

2 RELATED WORK

Because of the difficulties and time-consuming tasks associated with in-the-field experimentation over real smart homes, only a few related works exist in the literature. Most papers deal with metrics simulation in real-size test-beds. For example, Barker et al. [3] led to a set of heavily instrumented real-size smart homes. They provide a large number of test-beds for collecting data and experimenting with new techniques and algorithms to improve smart home efficiency. Some of the collected data has been useful to run preliminary tests before the completion of ScaledHome-2 since it shares similar approach and goals. The study conducted by Mateo et al. [13] used different kinds of regression models to predict the smart home temperature with an average error of about 0.1°; the developed modeling was applied to large buildings. The results cannot be directly used in a smart home system since additional scaling and tuning are necessary to correlate the results with smaller buildings.

Several studies have considered new solutions to modify the environment in new energy-efficient ways, by employing machine learning techniques to make the proposed solutions react to new scenarios autonomously. Jin et al. [16] have produced a predictive model able to optimize the power consumption for heaters located in a smart home environment. They obtained their model by implementing a recurrent neural network (RNN) and a long short memory model (LSTM), which utilized real temperature and humidity data collected by a real-size physical home. Their proposed optimization scheme saves energy while providing a comfortable environment based on user preferences on temperature and humidity. Han and Lim [9] employed IEEE 802.15.4 and ZigBee to develop a smart energy solution for residential or light commercial environments. It focuses on sensing device control, pricing, demand response, and load control applications. Razghandi and Turgut [14] propose an appliance-level load forecasting algorithm in residential homes instead of the level of the whole household in previous studies. Their model uses LSTM to predict a given appliance's potential load consumption over different time intervals. The Weatherman is the model presented by Chen and Irwin [5] that analyzes energy consumption data as well as wind and solar generation data to predict where a set of coarse energy consumption has occurred. Weatherman's main idea is that a set of weather signatures appear similarly in different environments around the world. Their method can be suitable in the construction of energy consumption-aware environments. The prototype presented by Teich et al. [15] is a neural network aimed at being implemented inside smart homes to maintain a cozy temperature defined by the user in an energy-efficient way. The dynamicity of the model allows it to re-train itself based on new activities inside the home. Dahmen et al. [7] focus on the security aspect of smart homes: their model detects anomalies within the home behaviors to automatically detect threats and to identify proper actions to be taken to guarantee the security of the environment.

Another set of studies has focused on carefully analyzing the relation between home features and human activities. In that sense, Lee et al. [11] have examined the relationship between the behavior of in-home inhabitants and indoor air quality by employing data collected from several sensors and chemical indoor air quality measurements. They gathered data from two smart homes to analyze how indoor air quality affected human behavior inside the house. They have also examined the relationship between indoor air quality and different smart home features (e.g., temperature and humidity). To analyze the data collected, they used various machine learning models such as random forest, linear regression, and support vector regression. Their work has shown a strong relationship between in-home human behavior and corresponding air quality, proposing a potential generalization of this consideration across multiple smart home scenarios. They show that the most impactful feature of human activities is temperature. Kim et al. [10] propose a method built upon different probability-based algorithms to identify social movements inside the house. The target application scenarios are human-centered such as healthcare and education. The research

focused on the recognition of multiple patterns and the ambiguity of different actions. The same motivations drove Cook et al. [6] to design a smart home to predict inhabitant action accurately. Their neural network algorithm facilitated an adaptive and automated environment to enhance the experience of its inhabitants. The meta-predictor proposed by Cook et al. [6] combines the strengths of multiple approaches to predict inhabitant actions. It is an intelligent agent where a weighted voting scheme between predictive algorithms generates the final forecast. Fritz and Cook [8] focused their work on elderly healthcare. They discussed the application of health-assistive smart homes by using data collected from sensors in elderly patients' homes, monitoring them with intelligent algorithms, and predicting potential changes in patients' health status. Alberdi et al. [1] proposed a smart home health-based models in which data was collected over more than two years to develop two models. The first model can detect mobility skills changes, while the second is related to the detection of changes in memory skills. These models act as an early warning system to prevent potentially dangerous symptoms from being harmful.

3 INTERACTION AND BEHAVIOUR EVALUATION IN SCALED HOME

3.1 Problem Formulation

We define the state of our environment as the values of current readings of sensors and actuators; this state could also include other information such as the present time. More formally, we refer to the state of real-world smart home as $X^R = \{x_i^R\}$, in which x_i^R is the value of sensor or actuator i. R denotes a real-world environment. We can divide X^R into three disjoint subsets: I^R, I'^R and O^R. I^R is the sensor readings inside, and I'^R is the actuator states. O^R refers to sensors that are outside the smart home for outside temperature and humidity, etc. The difference between I^R and I'^R is that we do not have any direct control over I^R; however, we can control the values of I'^R by sending signals or commands to the actuators.

Since the collection of data from real-world smart homes could be time-consuming and costly, this pushes the alternative option of simulation usage. The technical challenge here is understanding how exactly inside temperature is impacted by outside variables and inhabitants' actions. Our research work concentrates on building scaled physical models of homes that combine the cost and efficiency of simulation with the realism of physical measurements.

Formally, we define the scaled home state as $X = \{x_i\}$ without superscript R, in which x_i is the value of sensor or actuator i in the scaled home environment. We can divide X into four subsets: I, I', O, and O'. Notice that we have a new subset O' in the scaled home environment that we did not have in the real world. O' denotes the set of actuators that we can control to impact O. Finally, the state of the environment changes through time. We use subscript t to define state in a fixed time step t (e.g., X_t is the state of scaled home at time step t and X_t^R is the state of real-world environment at time step t).

Our goal is to develop a predictive model suitable for smart home optimization and planning. For example, to figure out the best policy for minimizing energy consumption, we can do that by considering every possible action by generating and evaluating a tree of options with our prediction system. We can then choose the most energy-saving and efficient path in the tree. More formally, given:

- two time instants t_i and t_{i+1} where $t_{i+1} > t_i$
- the corresponding $X(t_i)$ and $X(t_{i+1})$ scaled home states
- $f(X)$ as the function meant to predict a scaled home state by a previous one
- $X'(t_{i+1})$ as the future state predicted by f

$$X'(t_{i+1}) = f(X(t_i)) \tag{1}$$

- L_p as the loss function between the predicted state and the actual one in t_{i+1}.

$$L_p(t_{i+1}) = |X'(t_{i+1}) - X(t_{i+1})| \tag{2}$$

- $Lcost$ function represents the energy consumption due to the actuators working time.

Our goal is then to find the best $f(X)$ that minimizes both the $LP = \sum_{i=1}^{n} L_p(t_i)$ and $Lcost$ loss functions.

To enable our prediction system, we need to have access to both outside and inside smart home data. Outside data can involve metrics such as temperature and humidity, while inside features may include actuators state. We can simulate outdoor scenarios through our simulation environment and perform actions to change interior characteristics. It means we can make O close to O^R by controlling O'; in this way, we can also control I' to collect data on how our actions impact I.

3.2 ScaledHome

The ScaledHome-1 prototype by Ling et al. [12] has been significantly enriched with additional equipment and with a framework designed for easy extensibility and flexibility. This paper focuses on the description of the software entities composing this extension and resulting in a further improved version of the ScaledHome-2 [4] prototype; also, the paper originally concentrates on the application of our solution to control inside temperature while minimizing energy consumption.

To those purposes, in ScaledHome-2, we re-designed our framework to collect large amounts of data, suitable to set up predictive models for energy efficiency. There are many different scenarios where the crucial role played by the AC system cannot be underestimated in a smart home. For example, when we want to reach a temperature state that is very far from the current one, other energy-saving actions inside the house are insufficient to achieve it and the only way is to turn on the AC system. Nonetheless, there are some scenarios where the targeted state is close to the current one and we can adjust the state of each room by balancing the appropriate environment features while saving as much energy as possible.

To exemplify in a simple scenario, let us consider a situation where we have just two rooms with different temperatures (T_a and T_b), and we are interested in reaching a target temperature T. If the target temperature is in the $[T_a, T_b]$ range, it is then possible to reduce the heat of the warmest room and increase the coldest, with no economic/environmental cost, by allowing the two rooms to exchange thermal energy. With this consideration in mind, a planner which can make autonomous decisions to accomplish target state requirements would be a significant improvement in the smart home research area. On the other hand, we would need a large

amount of data to automatically build a predictive model to be used by the planner.

Let us emphasize the crucial role of data collected by empirical experiments operated in the real world in this field, in order to guarantee the maximum compliance of simulation scenarios and results. As we are going to demonstrate in Section 3.3, our simulation environment can reproduce with reasonable accuracy the metrics collected from real-world scenarios.

By delving into finer technical details, taking ScaledHome-1 and ScaledHome-2 as the starting basis for the work originally presented here, this paper offers the novel contribution of proposing a reliable and scalable distributed system meant to collect and analyze real-time information about the simulation environment. Moreover, the data collected are processed to extract the information needed by the system to respond to external events autonomously. From now on, for brevity reasons, we use the term ScaledHome to indicate the current version of the prototype that can be described by taking into account two associated new sub-systems:

- **Control system** represents all the characteristics of a real smart home, including those related to automated devices, such as the AC, and those connected with the inhabitants' actions.
- **Experiment management system** provides a transparent interface to set up and run experiments. It includes a remote UI, a data collection service, and an agent to load real-world scenarios.

A central entity is in charge of the interaction between these two systems. While the control system communicates with this primary entity by sending and receiving MQTT messages, the experiment management system interacts by sending REST requests since it does not necessarily need to be close to the physical simulation environment. The code related to both the control[0] and the experiment management[1] systems will be made available to the community to encourage the participation of third-party research teams.

Figure 1 shows the different entities that extend the ScaledHome system:

- **Actuator controller** is the entity in charge of managing all the actuators placed inside the house. It does not contain any business logic since it performs simple actions every time it receives an MQTT command message by the home controller.
- **Sensor controller** manages all the sensors placed in the house, collecting humidity and temperature from each of them every time it receives an MQTT request message by the home controller.
- **Home controller** is the centralized entity that contains all the business logic. It sends and receives MQTT messages depending on the behavior needed to perform the current simulation. It stores the data collected while keeping track of the state of the house during the simulation interval. Also, it provides a discovery service used to find and interact with the two controllers and to guarantee the synchronization of the model with the actual sensors and actuators state.
- **Environment simulation agent** interacts with the home controller to run real-world simulations by taking care of all the scaling aspects due to the mapping of the real state into the simulation one.
- **Action planner agent** provides the house with the performable actions, thus enabling ScaledHome to react to outside environmental changes by taking into account the current distance to the target state and the power saving policy.

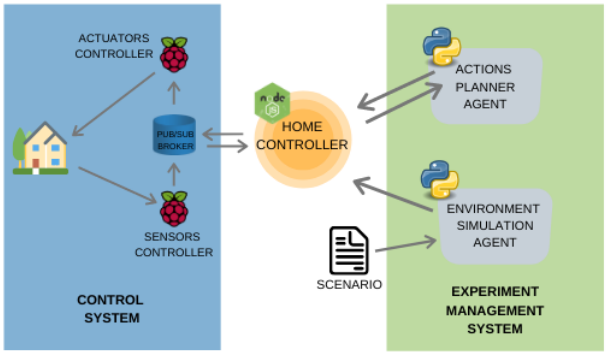


Figure 1: ScaledHome software entities divided into control and experiment management systems.

On the one hand, the actuators and sensors controllers have to be placed on a corresponding Raspberry Pi to perform the actual actions and provide real data. On the other hand, the home controller is deployed on either a local machine or a cloud host provider since it works independently by the underlying infrastructure, thanks to the adopted micro-services approach. The environment simulation agent and the action planner agent can be deployed on a different machine because they do not directly interact with the hardware controllers. They have to know the URI location of the home controller to get the smart home state and to provide the required actions to react to the outside environment.

As shown in Figure 2, the ScaledHome testbed consists of six rooms. Each of them has at least a temperature and humidity sensor, a door, and a window. The model has eight windows and seven doors, two of which are entrances to the house. Each door and window is connected to a motor for opening and closing operations, as we wanted to simulate a real human being moving inside the house, and provide the action planner agent an alternative and "green" way to change temperature and moisture inside each room.

A Raspberry Pi3 handles the sensors scattered throughout the house. It is connected to a T-Cobbler Breakout employed to connect signal, power supply, and ground wires to the physical sensors. We used the DHT11 sensors for temperature and humidity since they do not need any additional power supply and work with a 3.3V output already supplied by standard Raspberry Pi3 outlets. According to its datasheet that is summarized in Table 1, DHT11 can measure temperature in the range [0°C, 50°C], and humidity as a percentage in the range [20%, 90%]. In terms of accuracy, the one related to temperature is ±2 °C, while the one for humidity is ±5%.

[0] https://github.com/MatteoMendula/ScaledHome_Control_System
[1] https://github.com/MatteoMendula/ScaledHome_Experiment_Management_System

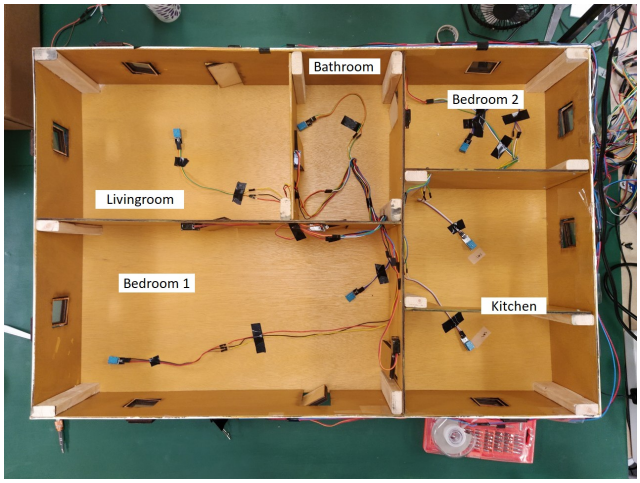


Figure 2: Real and simulation temperature mapping.

	Temperature	Humidity
Measurement range	[0°C, 50°C]	[20%, 90%]
Accuracy	[±1 °C, ±2 °C]	[±1%, ±5%]
Response time	[6s, 30s]	[6s, 15s]

Table 1: Sensor datasheet specifications.

We define actuators as all the devices which can change both the outside and the inside state proactively. According to that definition, the following entities are included in the actuator set:

- a lamp simulating a general heating source like the sun
- a big fan simulating a general cooling source like the wind
- a small fan which simulates the AC system
- 15 motors connected to doors and windows

A second Raspberry Pi3, equipped with a Servo HAT, controls the actuator set, and an additional power supply provides the energy needed to move the motors.

About the entity's interaction, we decided to avoid direct communication between the two controllers by placing a rendezvous point between the two, i.e., a Pub/Sub Broker entity. This broker communicates with another component to support the business logic of the entire system: the home controller, which plays a more complex task in terms of responsibilities and computation duties than the other two controllers. Although the introduction of a new entity increases the complexity of the system, it also guarantees a more significant time and space decoupling. By doing so, the controllers have to perform instructions and collect data. At the same time, all the most complex actions, related to the coordination of the whole system, are delegated to an entity with higher computational abilities.

The home controller acts as a middleware inside the system. It coordinates the entities composing ScaledHome, ensuring availability, and providing a discovery service able to find and communicate with the two controllers. It has been implemented in JavaScript to guarantee good performance in a lightweight Web environment. Also, it collects and stores data every 10 seconds, updating its state, saving a new document inside MongoDB, and appending state data on a CSV file. The GUI interface receives each new record to maintain the user session synchronized with the ScaledHome state. The middleware handles the interaction with the MQTT broker, by sending and receiving MQTT messages. The business logic of the application is self-contained into a central entity that interacts with the two controllers through the broker.

The MQTT messages exchanged are listed below:

- *discovery*: the message published by the home controller to identify if the actuators controller and the sensors controller are active and subscribed to the MQTT broker.
- *discovery_reply*: the message sent by the two controllers when they receive the discovery message. It contains the specific controller identifier.
- $re-connection\_retry$: the middleware sends this message when the controllers are not replying to a fixed number of interested MQTT messages.
- *cmd*: the middleware sends a cmd message to the actuators controller to specify the actions the controller has to perform.
- *request_record*: the home controller publishes a request_record message to obtain updated data about the state of the house from the sensors controller. It can ask a full record, containing the whole state, or it can request a subset of the state of the system, such as the temperature of a room.

Given that one of our main goals is to encourage the usage of ScaledHome by third parties, specific attention has been devoted to simplifying how potentially remote groups of researchers can participate in ScaledHome experiments. Two primary modes are supported:

- **Passive**: observing the ongoing experiments in real-time via the Web interface, streaming data, and video. Also, the researchers can download and process data from past experiments.
- **Active**: remotely controlling the user policy, home policy, and smart grid policy through scripts or a man-in-the-loop model. The researchers can also specify climate patterns or perform real-time control of the weather.

This requirement has been satisfied by implementing a Web server in NodeJs, able to interact with the user via a graphical interface, and to perform actions as consequences of REST requests. A WebSocket guarantees the constant update of records to the clients subscribed to the ScaledHome simulation system and are interested in receiving new and reliable data.

3.3 The Adopted Methodology: the Temperature Use Case

For simplicity and presentation briefness, in the following parts of the paper, we focus our simulations on temperature only. Since our final goal is to simulate a real day of different cities worldwide, we tested the capacity of ScaledHome to decrease and increase the temperature we have to map into our simulation environment. We have to know how much time is required by the smart home to move from a starting state to a target state to determine the best scaling approach and the associated simulation time interval duration.

To do that, the technique we propose starts with plotting the increasing and decreasing temperature functions. Figure 3 and

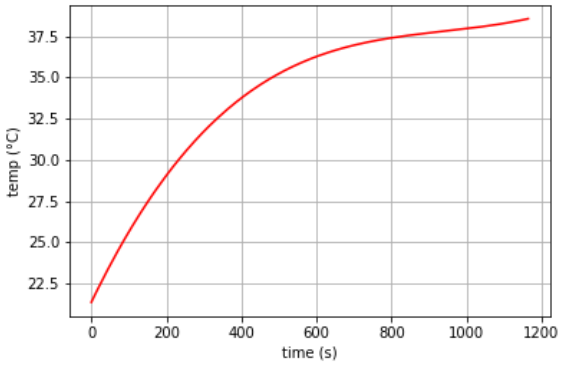


Figure 3: ScaledHome outside temperature increasing function.

Figure 4 illustrates how fast ScaledHome changes its outside temperature.

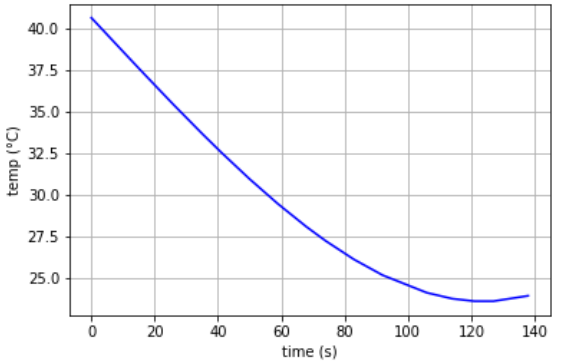


Figure 4: ScaledHome outside temperature decreasing function.

Table 2 shows the boundaries we have found by running several simulations, the reported average values for temperatures, and the corresponding times required to reach each bound.

	Temperature value	Δt
MAX	39±1°C	1100±20 seconds
MIN	21±1°C	150±20 seconds

Table 2: Simulation temperature boundaries.

Thanks to the data obtained by Milano Weather Station, published on Hardvard Dataverse [2], it has been possible to scale the temperatures in Milan in a range reachable by ScaledHome. Given the mathematical representation described in Table 3, we can map the real-world environment and the simulation according to the formula below:

$$SH_i = \frac{M_i - mil_{min}}{mil_{max} - mil_{min}} \times (sh_{max} - sh_{min}) - sh_{min} \qquad (3)$$

Scaling parameters	Mathematical representation
Lower bound in Milan	mil_{min}
Upper bound in Milan	mil_{max}
Lower bound in SH	sh_{min}
Upper bound in SH	sh_{max}
Accepted bias in SH	b
Acutuators current state vector	O'
Target temperature	T_x
Actual temperature	T_a

Table 3: Scaling parameters in Milan and in ScaledHome.

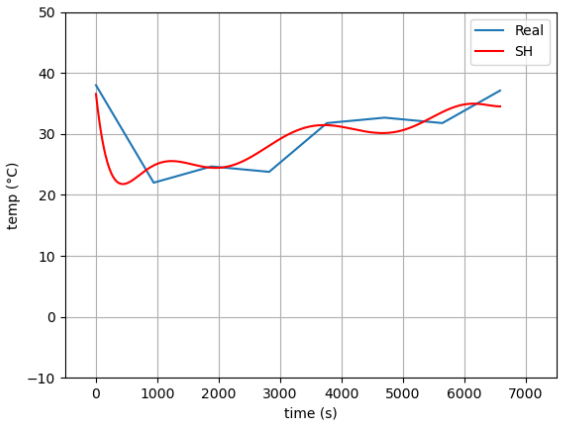
where SH_i is the value inside the ScaledHome system with reference to M_i. The same approach applies to different real-world scenarios for reproducibility purposes.

This consideration was not enough to guarantee the correctness of our mapping because the capabilities of the actuators used to increase or decrease the temperature were not modular since they have only a binary state. To address this issue, we defined a priori an appropriate simulation time interval, which has to last enough to simulate the highest temperature difference. We estimated the duration of the simulation interval by interpolating the increasing and decreasing regression functions. By doing so, both the lamp and the fan have enough time to reach the target temperature. Then, we applied hysteresis concepts to keep the temperature constant whenever the system reaches the target before the end of the predefined time interval. Formally, O' does not change if T_a is in the range $[T_x - b, T_x + b]$. Once T_a reaches $T_x + b$, O' will be set to decrease the inside temperature and vice versa.

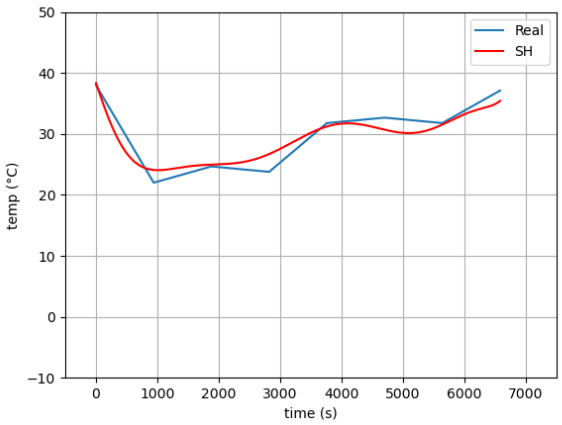
In addition to that, we introduced new temperature sub-goals by applying inference on the data we already had. By doing so, even if the temperature cannot be changed gradually, the trend is more smooth, and it follows the real temperature more accurately. It improves the smoothness of the temperature trend and the system's ability to adapt to changes in the corresponding real-world scenario during the given time interval. In this regard, the simulation time interval will change depending on the real-world situation we need to simulate. A larger temperature interval will lead to a longer time frame. Basically, instead of reaching the goal using the entire available time interval, we divided each interval into multiple subintervals obtaining at the same time the same number of sub-goals. The approach adopted provides a flexible way to map real-world scenarios in an easy-to-manage simulation environment.

Figure 6a and Figure 6b show as a final result a considerably accurate mapping of real-world temperatures in Milan into the ScaledHome temperature range. Also, we evaluate the improvement achieved by applying hysteresis techniques at this time. In particular, Figure 6a displays that the ScaledHome system, without the usage of sub-goals, does not adjust its outside temperature target quickly enough to catch the Milan temperature trend.

As can be seen in Figure 6b, the division of each goal into six subgoals leads to better matching between the real and the simulated temperature. It is reasonable to think that the increase in the number of sub-goals will guide to even more accurate matching.

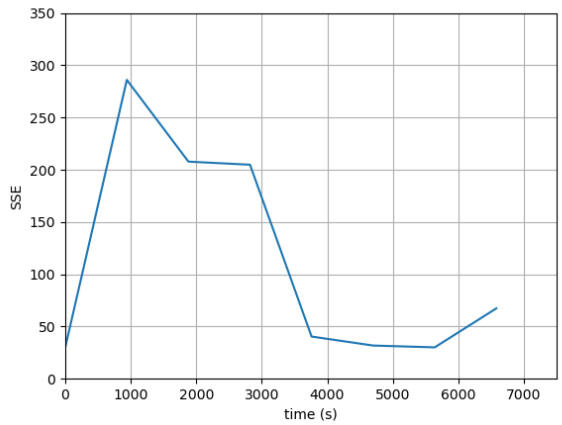


(a) Without the adoption of hysteresis techniques and zero sub-goals.

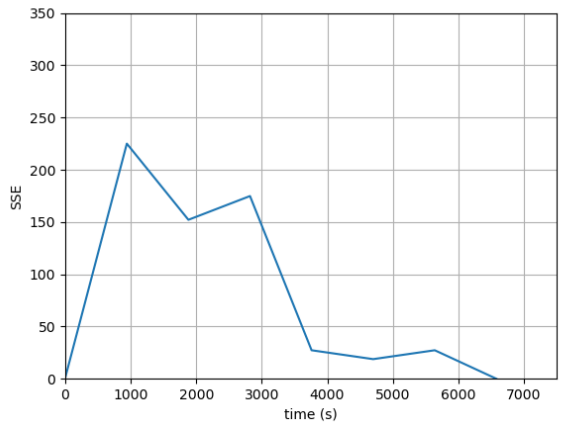


(b) With the adoption of hysteresis techniques and six sub-goals for each temperature target.

Figure 5: Real and simulated temperature mapping with and without the adoption of hysteresis techniques and sub-goals.



(a) Without the adoption of hysteresis techniques and zero sub-goals.



(b) With the adoption of hysteresis techniques and six sub-goals for each temperature target.

Figure 6: Real and simulated temperature mapping with and without the adoption of hysteresis techniques and sub-goals.

4 IN-THE-FIELD EXPERIMENTAL EVALUATION

Our goal consisted of finding a reliable machine learning model that predicts a future state based on the current state inside the house. This kind of forecasting is needed to select among all the possible action vectors, which would reach the goal in the shortest amount of time while minimizing energy consumption. In other words, we want to change the inside state of the house, reducing the usage of inside actuators such as the AC system and the heater.

To obtain a predictive model able to meet the aforementioned requirements, four different machine learning regressors have been trained:

- K-nearest neighbor (KNN)
- SVM regressor
- Deep Neural Model (DNN)
- Long short-term memory (LSTM)

While we developed the DNN model from scratch, the others are provided in ready-to-use packages by scikit-learns and Keras. In particular, scikit-learn offers the library containing KNN and SVR, while Keras allows the developer to build the desired LSTM network by configuring a built-in function. Regarding the DNN model, we have identified five layers with 128 neurons each as best hyperparameters. Besides, we selected the Huber loss and the Relu activation function as the most suitable ones.

We fed those models with the same dataset collected by running several simulations where we changed the outside metrics accordingly to the scaled temperatures obtained from one day selected from the Milan temperature dataset [2]. We also randomly altered the state of the inside actuators, giving the models the knowledge to learn the correlation between actuators state and the other interior features.

In this regard, each record r_i contains the features we considered as inputs while the corresponding future state record r_{i+t} includes the target features we wanted to predict. The t value is a settable parameter, as well as input and target features. We introduced this kind of flexibility to consider some variables as hyperparameters of the models since we do not know a priori which are the most impactful ones for our predictive purposes.

Moreover, we studied the correlation between the different metrics inside the house for the same reason. In the following, Figure 7 and Figure 8 illustrate the relationship between inside temperature and humidity. By doing so, we have been able to treat the inside features as training parameters, by taking the most relevant metrics as a starting point during the hyperparameter space exploration. To explore the hyperparameter space, we developed an automated greedy tuner able to find the most suitable ones for our purposes (e.g., the best number of neighbors in K-nearest neighbors).

As regards to SVR, we had to combine one SVR model for each target variable because it considers just one target value at the time. Concerning DNN, we adopted a batching approach to speed up computation. In that sense, we choose as batch size the same t parameter used to identify the target records in the previous models, exploring different sizes as we have done with the other hyperparameters. We applied the same method to design the LSTM network, exploring the batch size and the best sequence length.

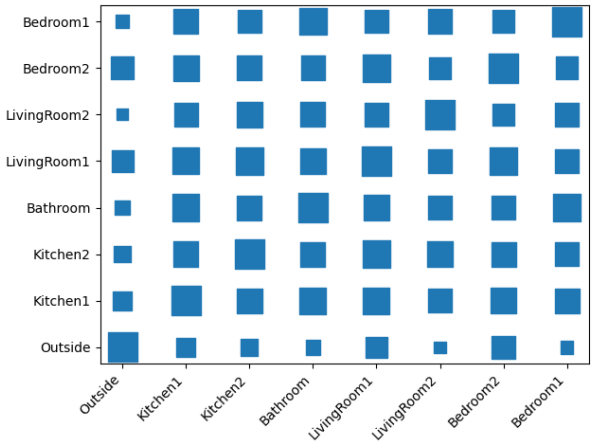


Figure 7: Temperature heatmap.

For each implemented model, we considered both mean squared error (MSE) and accuracy. Regarding the latter, we defined a tolerance range to validate the predicted value, increasing a score variable every time our prediction was inside the tolerance range.

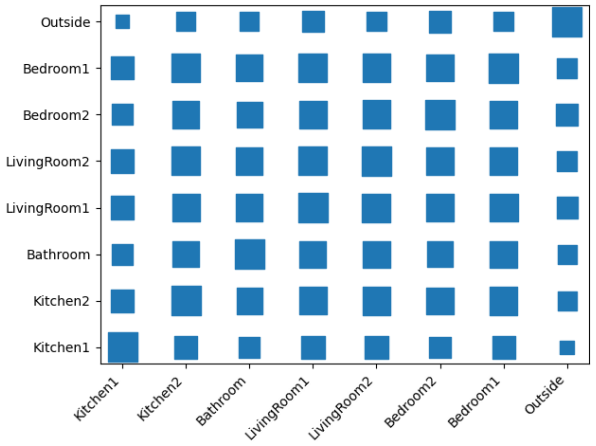


Figure 8: Humidity heatmap.

Since temperature is the main feature we predicted, and the accuracy of the available sensors is 1°C, we set the tolerance range for each actual target value y_i to $[y_i - 1, y_i + 1]$.

With a dataset consisting of about 300 records, we identified DNN as the model with the highest accuracy, achieving an accuracy score of approximately 89%. While KNN and LSTM models reached 70% and 60%, SVR became the model with the lowest accuracy of 20%.

We then decided to increase the size of the dataset to find out if LSTM would achieve better results. Since LSTM performs the learning process on sequences instead of single values, we expected to reach better performance with a more significant amount of data. With approximately 4000 records, the best model was KNN, which achieved about 95% accuracy, while SVR was still the worst by barely reaching 20%. DNN performs almost the same, and LSTM improves its score by an accuracy score of about 87%, as expected.

Table 4 reports the scores in terms of accuracy achieved by each model, differentiating each of them by their score obtained through training on either the only training set or the validation and training sets merged.

Model	300 records	4000 records
KNN	70% and 71% (val.)*	95% and 96% (val.)*
SVR	42% and 47% (val.)*	22% and 19% (val.)*
DNN	79% and 63% (val.)*	89% and 88% (val.)*
LSTM	64% and 27% (val.)*	87% and 85% (val.)*

Table 4: MM models accuracy comparison.
(*) score achieved by training on training set and validation set merged.

We find that the most models are improving their accuracy with more data. However, the SVR model has both a significantly lower accuracy than the other models, and the accuracy decreases with more data. A possible reason for this behavior is that the linear model behind the SVR approach cannot represent the complex nonlinear relationship between the inputs and output features.

Additionally, we simulate the recursive application of our predictive model on already forecasted home states, by chaining multiple

predictions sequentially. To do that, we selected KNN as the most accurate among the other models. Figure 9 shows the actual bedroom temperature trend and the corresponding recursive prediction obtained by our machine learning model. As can be seen, it follows the real trend during the first 6000 seconds, and then it does not accurately match it because the other related features deviate too much from the actual behavior.

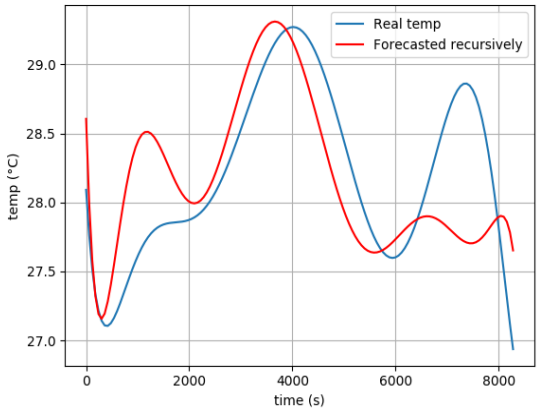


Figure 9: Bedroom temperature and recursive prediction.

```
def recursive_prediction(self, prediction_index):
    x, y = self.data_set_parts['test']['x'], self.
     data_set_parts['test']['y']
    fig, ax = plt.subplots()
    #index of the column to be predicted
    t6_index = 2
    time_counter = 0
    time = []
    t6_y = []
    y_hat_recursive = [x[0].reshape(1,35)]
    t6_y_hat = []
    for index in range(0,x.shape[0]):
        if (index % prediction_index == 0):
            time.append(time_counter)
            time_counter += 60
            t6_y.append(y[index][t6_index])
            temp = self.scaler.transform(self.predict(
     y_hat_recursive[len(y_hat_recursive)-1]))
            y_hat_recursive.append(temp)
    for el in y_hat_recursive:
        a = self.scaler.inverse_transform(el)
        t6_y_hat.append(a[0][t6_index])
    p1 = np.polyfit(time,t6_y,9)
    p2 = np.polyfit(time,t6_y_hat[:len(t6_y_hat)-1],9)
    line1, = ax.plot(time, np.polyval(p1, time))
    line2, = ax.plot(time, np.polyval(p2, time),'r-')
    ax.set(xlabel='time (s)', ylabel='temp (ÂřC)', title=
     'Bedroom temperature recursive prediction')
    ax.legend((line1, line2), ('Real temp', 'Forecasted
     recursively'))
    ax.grid()
    plt.show()
```

Listing 1: Recursive prediction inside the ScaledHome Experiment Management System.

Listing 1 shows how the recursive prediction of temperatures works inside the ScaledHome Experiment Management System.

This kind of approach, combined with interpolation techniques, can be taken into account when it is not possible to get recent data at the same frequency as required by the action planner agent.

5 CONCLUSIONS

This paper describes the development of a predictive model for smart home environments. We used ScaledHome, a small scale IoT testbed for the simulation of real-world scenarios and collection of data. Using this data, we trained a machine learning model that can reliably predict the new environment states function of the estimated current state and the actions taken by the actuators.

Future work will include the improvement of the predictive capabilities of the model using future state-space exploration with heuristics and reinforcement learning techniques. We also plan to improve the testbed by adding new sensors measuring other aspects of the smarthome environment and actuators modeling user actions.

ACKNOWLEDGEMENT

The support for this work was provided by the National Science Foundation REU program awards 1560302 and 1852002. Any opinions, findings, and conclusions and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] A. Alberdi, A. Weakley, M. Schmitter-Edgecombe, D. J. Cook, A. Aztiria, A. Basarab, and M. Barrenechea. 2018. Smart Home-Based Prediction of Multidomain Symptoms Related to Alzheimer's Disease. *IEEE Journal of Biomedical and Health Informatics* 22, 6 (2018), 1720–1731.
[2] ARPA. 2015. mi_meteo_8162.csv.
[3] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. 2012. Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes.
[4] T. Burns, Gr. Fichthorn, S. Zehtabian, S. S. Bacanli, M. Razghandi, L. Bölöni, and D. Turgut. 2020. IoT Augmented Physical Scale Model of a Suburban Home. In *IEEE ICC 2020 Workshop on Convergent IoT (C-IoT)*. 1–6.
[5] D. Chen and D. E. Irwin. 2017. Weatherman: Exposing weather-based privacy threats in big energy data. *IEEE Big Data Conference* (2017), 1079–1086.
[6] D. J. Cook, M. Youngblood, E. O. Heierman, III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. 2003. MavHome: an agent-based smart home. In *IEEE PerCom'03*. 521–524.
[7] J. Dahmen, B. L. Thomas, D. J. Cook, and X. Wang. 2017. Activity Learning as a Foundation for Security Monitoring in Smart Homes. *Sensors* 17, 4 (2017).
[8] R. Fritz and D. J. Cook. 2017. Identifying Varying Health States in Smart Home Sensor Data : An Expert-Guided Approach.
[9] D.-M. Han and J.-H. Lim. 2010. Smart home energy management system using IEEE 802.15.4 and zigbee. *IEEE Transactions on Consumer Electronics* 56 (2010).
[10] E. Kim, S. Helal, and D. J. Cook. 2010. Human Activity Recognition and Pattern Discovery. *IEEE Pervasive Computing* 9 (2010), 48.
[11] W. Lee, S. Cho, P. Chu, H. Vu, S. Helal, W. Song, Y.-S. Jeong, and K. Cho. 2016. Automatic agent generation for IoT-based smart house simulator. *Neurocomputing* 209 (2016), 14–24.
[12] J. Ling, S. Zehtabian, S. S. Bacanli, L. Bölöni, and D. Turgut. 2019. Predicting the temperature dynamics of scaled model and real-world IoT-enabled smart homes. In *IEEE GLOBECOM'19*.
[13] F. Mateo, J. J. Carrasco, A. Sellami, M. Millán-Giraldo, M. Domínguez, and E. Soria-Olivas. 2013. Machine learning methods to forecast temperature in buildings. *Expert Systems with Applications* 40 (2013), 1061–1068.
[14] M. Razghandi and D. Turgut. 2020. Residential Appliance-Level Load Forecasting with Deep Learning. In *IEEE GLOBECOM'20*.
[15] T. Teich, F. Roessler, D. Kretz, and S. Franke. 2014. Design of a Prototype Neural Network for Smart Homes and Energy Efficiency. *Procedia Engineering* 69 (2014), 603–608.
[16] J. Wenquan, U. Israr, A. Shabir, and K. Dohyeun. 2019. Occupant Comfort Management Based on Energy Optimization Using an Environment Prediction Model in Smart Homes. *Sustainability* 11 (2019).