Energy Efficient ABMS for a Smart Home Environment

Arfaa Rashid, Rocio Rueda, Jasman Saran, Lamina Zaman
School of Computer Science
University of Windsor
Windsor, Canada

I. Abstract

With the rising concern of energy over usage, this paper concentrates on implementing a multi-agent based system for an energy efficient home management system. The system utilises a combination of the Butler, Sensor, Effector, and Interactor agents to assess the effectiveness of an artificially intelligent smart home. By using the A* search algorithm, the system has been tasked to manage light and temperature levels. Specifically, the algorithm manages heating, cooling, and resting actions to achieve the temperature goal. The algorithm also oversees the usage of light energy by dynamically adjusting the brightness level based on indoor and outdoor light levels. This system is beneficial for homeowners seeking to reduce energy usage while maintaining their personal needs. The implementation represents a step towards sustainable living in residential settings, offering comfort for individuals and families while minimising their environmental footprint.

II. Introduction

In 2019, Canadian households consumed a substantial 1.4 million Terajoules of energy, with residential sectors being key contributors to national energy consumption [1]. To address this challenge, there's a growing inclination towards harnessing artificial intelligence (AI) to develop smart homes utilising agent-based modelling systems [2]. The original simulation from an open source GitHub repository called es-smartHome made by Andrea Montemurro utilises an intelligent agent built on logic in order to model a smart home management system [3]. It allows users to set preferred values for temperature, light brightness, noise, and wind to produce numerical values for each smart home device by using comparisons between indoor and outdoor variables to make decisions. Consequently, while it emulates a real-world environment, implementing this model in reality would result in not accounting for minimising energy usage.

Identify applicable funding agency here. If none, delete this.

The proposed Light Lumificient model simplifies the environment and helps a balanced solution that fulfils user needs while ensuring sustainable energy usage. In addition, the model uses a multi-agent based system (ABMS) with various agents such as Butler, Sensor, Effector, and Interactor agents.

The Sensor agent's task is to get the inputs of the environment by providing information on the content parameters and features, continuously checking variables like outside brightness, inside brightness, inside temperature, outside temperature, outside noise, outside wind, and outside rain. Since the model is deployed as a simulator, in order to get the most accurate environment percepts, an API called OpenWeatherMap API is utilised for getting the current weather forecast in a given city. Another relevant agent, Interactor agent, is used to get user's preferences for desired light and temperature levels. To achieve the user's desired temperature and light levels both efficiently and effectively, the Butler agent is used to make inferences based on the current percepts from the Sensor and Interactor agents in order to search for the most optimal device fixture levels that would maximise energy efficiency. In order for this to be done, two A* search algorithms are employed to determine both a sequence of heating, cooling, and resting actions, as well as brighten and dim options for light fixtures. By considering associated energy costs for heating, cooling, light fixture intensity levels, and heuristics informed by proximity to the user's target, the algorithm swiftly achieves the user's desired temperature and light levels in a cost-effective manner. Lastly, the Effector agent is used to output the optimal results from the Butler agent to ensure the device fixtures perform the desired action.

III. Related Works

This project focuses on the development of a smart home system focused on optimising electric energy consumption, evaluated through a simulation. A multi-agent based system (ABMS) has been deployed for the implementation. This ABMS integrates inputs from various agents to optimise energy usage based on factors including user preferences, environmental conditions, and hardware capabilities.

Several other works have also explored the integration of ABMS for a home energy management system that reduces energy usage and satisfies user preferences. For instance, one study proposed an AI-based home energy management system (AI-HEMS) specifically tailored to control heaters in winter, using sensor data to create prediction models [4]. The main goal of the prediction model is to reduce energy consumption while maintaining user preferences for comfort. This approach incorporates strategies such as comfort-based control focusing on an analysis of environmental data and outing-based control, and predicting outing patterns to adjust heater settings accordingly. Additionally, an

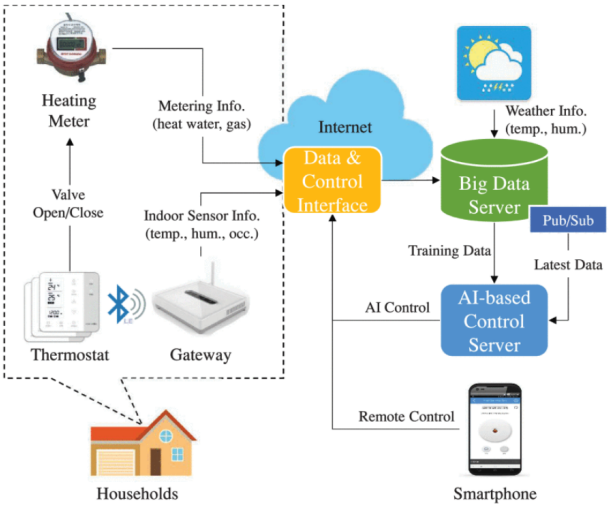

Fig. 1. Overview of AI-HEMS [4]

occupancy-based control is implemented to optimise energy usage based on the occupancy status of a particular room. Real-time monitoring of energy usage, environmental conditions, and weather forecasts help intelligently decide the best way to optimise energy usage. Additionally, factors such as temperature, humidity, and occupancy are also used to adjust the heating according to user preferences. The proposed Light Lumificient model did not factor the occupancy status, however, the real-time environmental factors are considered in the decision-making process.

Another article, talks about a home automation management system made a notable contribution to this field: the design of a graphical user interface (GUI) [5]. This GUI allows users to set parameters such as minimum and maximum temperatures, adjust lighting intensity, and schedule the opening and closing of blinds and windows based on user preferences and information available from sensors. Some of the tests on the

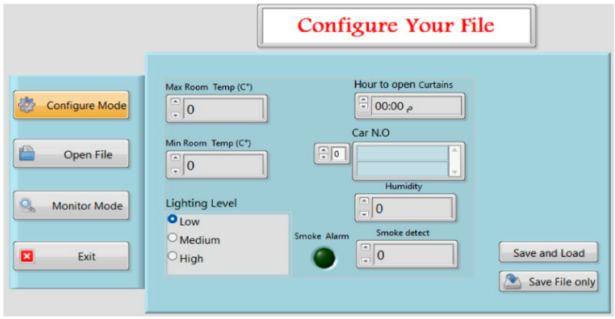

Fig. 2. LabVIEW GUI configuration [5]

proposed system show the ease of use, reliability, and cost-effectiveness compared to alternative solutions. The main objective of the system is to minimise electricity costs while maintaining comfort for the users by implementing optimization techniques for energy usage. Factors such as light intensity, temperature and humidity are considered to control fans, heaters, air conditioners, and light bulbs. This adaptive control strategy reduces electricity uses and aligns the fixtures usage with real-time environmental information available via the systems sensors. The Light Lumificient model aligns closely to this approach also using a GUI based simulator with different Sensors to get temperature and light brightness, as well as user desired temperature and lighting.

Cavone et al. also developed a Multi-Agent System (MAS) including Butler, Sensor, Effector, and Interactor agents [6]. Their model adopted a decentralised approach since it is more resilient to failures occurring and avoids any issues affecting the entire system. Another benefit is that it is more scalable, since it allows for the model to be expanded easier. The Butler agent is the main agent in the system regarding user context and environmental percepts–it recognizes the user's situation using intelligent reasoning, machine learning, service-oriented computing, and semantic web technologies. The sensor

agents gather inputs from the environment to be used for the decision-making process. In addition, Cavone et al. introduced abduction and abstraction strategies. Abduction enables cautious decision making in the absence of data, while abstraction simplifies irrelevant details for the task. This proposed implementation also utilises abstraction in instances where data from a simulated room is unavailable, so a default value is used for decision-making. Abduction is also implemented, such as making API calls to gather information for the sensor agent. This relates to the model proposed in this paper as they both use sensor agents for price optimization.

IV. Illustration / Figure

Figure 1 illustrates the architecture of the agent components taken into consideration based on the suggestions of Cavone et al [6]. The figure also illustrates the interactions between agents. A sample interaction between the Sensor and Butler agent involves the Butler agent receiving percepts from the Sensor agent throughout a given period of time. The Butler agent then performs two A* search algorithms, where the algorithms take values from the Interactor agent (which accepts values based on user preference). The user preferences act as a goal state for the A* algorithm while also considering factors such as energy efficiency for the actual cost. Another interaction between the agents involves the Butler agent communicating or sharing their results with the Effector agent and passing those values to the specific smart home fixtures.

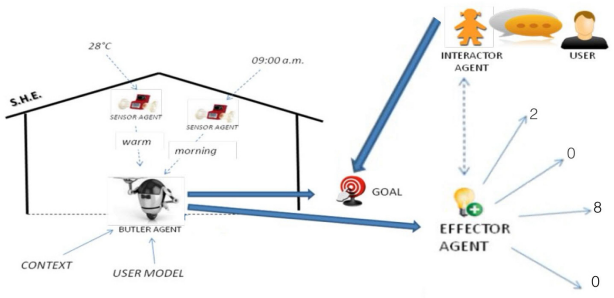

Fig. 3. The multi-agent system architecture. Adapted from [6]

Figure 2 shows the overall simulation. On the left shows the input values taken from the sensors. In the middle, there is a floor plan of the smart room that the butler will consider for spatial percepts, as well as a dropdown menu with different actions for the user (eg. sleep, watch a movie). The right shows the outputted results for both the original simulation (logic results) and the updated simulation (A* results). Also, there is an "Ask Explanation" button that will explain the logic-based results. Lastly, in the top left, there is a profile button showing user preference values for each action.

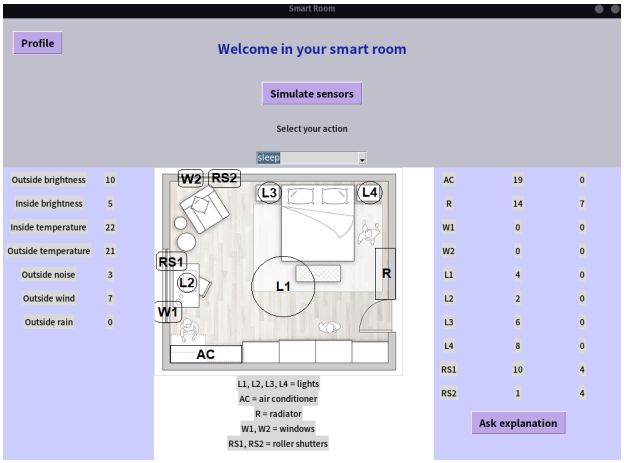

Fig. 4. Overview of proposed simulator

V. Simulation Details

The smart home simulation includes a room with an AC, radiator, and various light sources, each adjustable in brightness and power levels to control total temperature and brightness. For the simulation's graphical user interface, the Python library Tkinter is used. Since the original simulator uses logical comparisons to get device fixture results, the SWI-Prolog and PySwip Python libraries are used to create facts and rules built on Prolog.

Instead of modifying the original simulator's logic, the updated Light Lumificient model utilises both the original logic and formulates a comparison with a new approach to compare viability between the two. The new approach aims to obtain realistic results accounting for both user preferences and energy efficient consumption. In order to do this, the simulator uses two A* algorithms: light and temperature.

For the A* algorithm concerning light, the algorithm aims to find the most optimal brightness value for each light source based on target brightness levels that the user sets for different tasks such as studying, watching a movie, sleeping, listening to music, and cleaning. Each fixture represents light values by an integer from 0 to 10, with 0 representing no light emitted and 10 representing maximum light emitted. To determine the current state, the algorithm calculates both energy cost and heuristic cost. The heuristic cost has two components; it weighs the difference between the current value and user's set

value to prioritise states that are closer to the goal state, and also considers the intensity of each individual light fixture. Intensities are approximated as decimal values between 0 and 1, representing a ratio-based scaling of the contribution of the light source with respect to the total brightness in the room. This is calculated based on the user's location–determined based on the chosen task–and different variables such as the level of outside brightness coming in from the windows. More specifically, the ratios are calculated using the Python library SciPy to find the Euclidean distance between fixture coordinates and the user's relative location. Since ratio values must be higher for lower distances, the distances are inverted and all summed up. Afterwards, each inverted distance is divided by the total sum of inverted distances producing the final ratio results.

Now transitioning to the operational aspects of the algorithm, attention is directed towards comprehending its functionality within the context of various user tasks. For instance, in scenarios where the user is studying, it is presupposed that their location is fixed at their desk. Consequently, the algorithm approximates light intensities with respect to this specific spatial reference. Moreover, it takes into account two external variables, denoted as W1 (window 1) and W2 (window 2), which encompass the effects of external brightness on the overall illumination. These variables are computed based on factors such as the percentage of cloud cover and the time of the day, which contribute to the total brightness within the room. Considering the collective inclusion of variables L1, L2, L3, L4, W1, and W2 for determining total brightness, it is established that the intensity heuristic must allocate a sum of 1 to internal variables, with external variables representing their own distinct contribution factor.

In terms of the implementation, since A* is built on finding the minimum-cost path to the goal, initial values are added to a priority queue, using the built-in queue library in Python. For light, these values are initial total brightness, initial energy cost, initial light fixture values, and shutter status. When considering different paths to take, the algorithm iterates through each light fixture and adds or subtracts one from that fixture value, depending on whether the total brightness is less than or greater than the target. A diagram displaying this is shown here:

Total brightness is calculated based on a weighted sum where the model multiplies the intensity per light with current brightness at each light, and adds outside brightness multiplied by the status of the shutter. Both

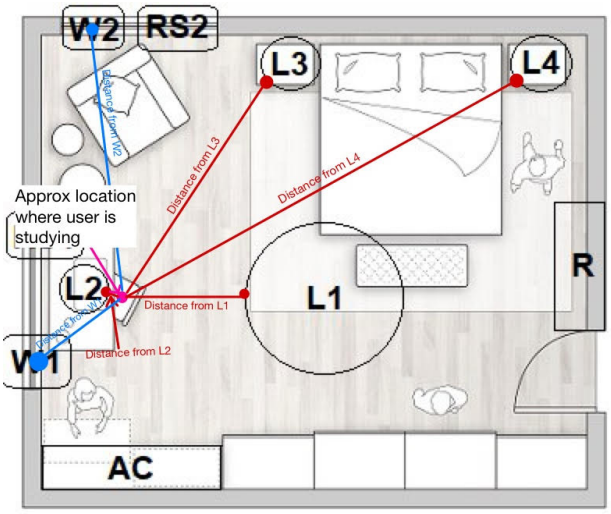

Fig. 5. Floor plan spatial reference

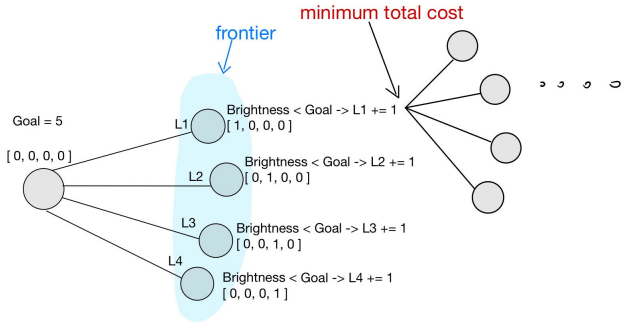

Fig. 6. A* algorithm for light path structure

individual components are multiplied by their own contribution, which is calculated based on the distance from the windows. In short, the equation used is:
totalBrightness = contributionOfInside(intensityL1*lightLevelL1 + intensityL2*lightLevelL2 + intensityL3*lightLevelL3 + intensityL4*lightLevelL4) + contributionOfOutside*(outsideBrightness*statusOfShutter)*

Energy cost is calculated based on an energy consumption rate for each fixture. This can take a value from 0 to 1, where 0 represents a fixture's complete lack of energy consumption and 1 being the maximum energy consumption for that fixture. In short, the equation used is:
*cost = energyConsumptionL1*lightLevelL1 + energyConsumptionL2*lightLevelL2 + energyConsumptionL3*lightLevelL3 + energyConsumptionL4*lightLevelL4*

Once both energy cost and total brightness are

calculated for the current path, total cost is calculated by the formula: f(n) = h(n) + g(n). Function h(n) represents the heuristic cost which is calculated based on how far away the current total brightness is from the goal brightness. Function g(n) represents the actual cost which, in this case, is represented as the energy cost. This process repeats until either the queue is empty or the target brightness is reached.

In sum, the algorithm iteratively evaluates different combinations of light brightness values, considering energy consumption rates and the user's target brightness to minimise energy costs while achieving the desired brightness levels.

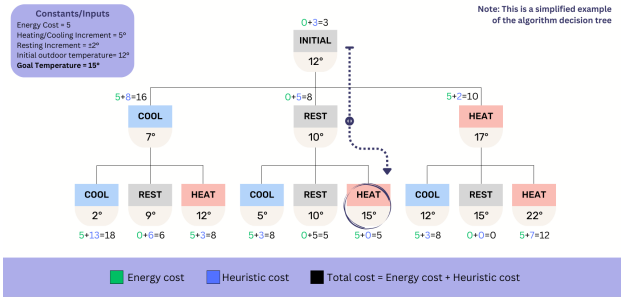

Fig. 7. A* algorithm decision tree for temperature

The A* algorithm concerning temperature is designed similarly to the search algorithm for light brightness. That being said, the algorithms do have differences. The temperature A* algorithm also involves a sequence of heating, cooling, and resting values. Heating and cooling have associated energy costs, and the algorithm's heuristics are informed by proximity to the user's target temperature. Similarly to light brightness, the actual cost is calculated based on the energy cost, determined by the action chosen. This is multiplied by an energy efficiency constant between 0 to 1, to balance weighting between heuristic and actual cost. Taking both user preference and energy efficiency into consideration allows the algorithm to quickly meet the user's set temperature in a cost-effective way.

In terms of the implementation, the algorithm iterates through the three sequences: heating, cooling, and resting, and determines the current temperature based on the action chosen. For example, if the option is currently heating, then a constant for a heating incremental value is added to the current temperature, and costs are calculated to determine the quality of this state. If there are better ways to reach this state, this solution sequence is abandoned. Otherwise, this process repeats until the target temperature is reached.

VI. Discussion

A. Original versus the new model

The original es-smartHome model is purely logic-based and goal-oriented. It considers both inside and outside percepts and changes device values to reach the goal set by the user, without considering the associated energy costs. Effector values are strictly determined from a simple comparison between current and goal values. A sample comparison in the Prolog knowledge base is made below for temperature, where the outputted AC and radiator values are determined through comparing inside brightness with Y_temp (a preferred temperature based off user preferences): *preference(study, temp, 20, [ac, r, w1, w2]).*

The radiator is set to Y_temp when the inside temperature is less than Y_temp, and AC to Y_temp otherwise. The query is shown below: *setInsideEffectors_temp(X_temp_inside, Y_temp) :- (X_temp_inside* $<$ *Y_temp* $\rightarrow$ $setEffectors([r], Y\_temp), setEffectors([ac], 0);$ $setEffectors([ac], Y\_temp), setEffectors([r], 0)).$

This results in an unrealistic and high cost approach. Instead, the proposed Light Lumificient model considers energy costs associated with each action in order to meet user requirements in a realistic and cost-effective way. However, the complexity of the original simulation does allow for more options in how to reach the user's goal, some of which are inherently more cost-effective. For example, the original simulation has windows and roller shutters that allow the outside environment to have a greater effect over indoor temperature and light. Since the Light Lumificient model simplifies the environment and variables to achieve efficient functionality at a small scale, it cannot take advantage of windows and shutters to minimise AC, radiator, and light usage. Overall, the model performs well and meets expected benchmarks, but in the future, it could be improved to consider more variables that increase cost-effectiveness further.

B. Limitations and Improvements of Model

The model successfully simulates a smart home system and effectively reaches user goals. However, some limitations exist. Firstly, the model only consists of A* algorithms for light and temperature levels–other variables that may affect energy efficiency such as wind

speed and noise levels are not considered.

In addition, the A* calculations for each light fixture is a ratio-based approximation relative to the light fixture and the user's location depending on an action. Hence, this heuristic may not be one-hundred percent accurate for situations where the user is constantly moving. Hence, it is ideal to have sensors constantly predicting user location.

Since this model is based on a simulation of a smart home, it may be limited in showing accurate percepts in comparison to real life smart home systems. In addition, the simulation is based on a two-dimensional environment, hence spatial recognition of objects is less precise than in a three-dimensional space. To model the environment more accurately, the simulation could be developed on another software supporting 3D modelling such as Unity3D. The simulation could also calculate heuristics such as light fixture intensity using more advanced equations considering shadows, correlated colour temperature, log-median luminance, and contrast.

Lastly, the simulation accepts integer values for all of the variables, for instance the light-based variables range between 0 - 10 units and outside temperature variables range between 0 - 30 units. Additionally, the effector agent's values range between 0 - 10 units for the light fixtures and 0 - 50 units for the temperature-based fixtures. This method reasonably simulates results in an isolated testing environment, but, in reality, temperatures fluctuate based on location so a user may require unconstricted temperature ranges. Additionally, double precision variables may be more accurate for the implementation.

C. Results

Figure 7 graph shows the results of both light and temperature A* search algorithms on various devices (including air conditioner, radiator, windows, light fixtures, and roller shutters) and compares them to the original simulator's logic-based results. The original simulation set a level of intensity for each device based on logic-based results (blue bars), while our simulation sets these levels based on the A* search algorithm (pink bars). Each device value is scaled to an integer number between 0 and 10 (the exceptions being the air conditioner and radiator, which are scaled from 0 to 50). The A* algorithm produces significantly more energy efficient results in comparison to the original logic-based results. For instance, the logic result in the example below requires the use of two light fixtures. The A* results, on the other hand, only uses one light fixture. Additionally, the logic results require the radiator to be turned up far higher than the A* results require. Interestingly, Light Lumificient's method of considering user preferences to prioritise certain lights depending on the selected activity would allow for both improved energy efficiency and user satisfaction, rather than forcing a trade-off between the two. The original simulation did not consider energy at all in favour of user preference, so it could be hypothesised that each change made to prioritise energy efficiency would result in slightly worse user satisfaction. However, this change was entirely positive and improved both results significantly.

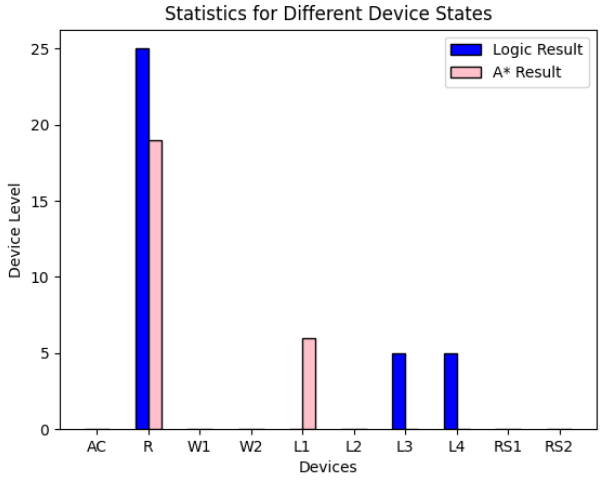

Fig. 8. Double-bar graph for comparing logic and A* results

VII. Conclusion

In conclusion, by using a combination of Butler, Sensor, Effector, and Interactor agents, the system effectively optimises energy consumption while meeting user preferences for light and temperature levels. The A* search algorithm was utilised for two tasks: managing light and temperature variables. Regarding light brightness, the algorithm successfully determines the most suitable brightness levels for different tasks, taking into account both user preferences and energy efficiency considerations. Regarding temperature, the algorithm manages heating, cooling, and resting actions to reach target temperatures while minimising energy costs. This way, our simulation aligns with the United Nations' Sustainable Development Goal 7: Affordable and Clean Energy.

References

[1] S. C. Government of Canada, "Households and the Environment Survey: Energy Use, 2019,"

The Daily - , https://www150.statcan.gc.ca/n1/daily-quotidien/220502/dq220502b-eng.htm (accessed Feb. 1, 2024). [2] C. Feijóo et al., “Harnessing Artificial Intelligence (AI) to increase wellbeing for all: The case for a new technology diplomacy,” Telecommunications Policy, vol. 44, no. 6, Jul. 2020. doi:10.1016/j.telpol.2020.101988 [3] A. Montemurro, “Andmon97/ES-smarthome: An intelligent agent written in Prolog for managing the Smart Environnement of a room.,” GitHub, https://github.com/andmon97/es-smartHome (accessed Feb. 21, 2024). [4] K. Kwon, S. Lee and S. Kim, ”AI-Based Home Energy Management System Considering Energy Efficiency and Resident Satisfaction,” in IEEE Internet of Things Journal, vol. 9, no. 2, pp. 1608-1621, 15 Jan.15, 2022, doi: 10.1109/JIOT.2021.3104830 [accessed Mar. 24, 2024]. [5] Elkholy, M.H.; Senjyu, T.; Lotfy, M.E.; Elgarhy, A.; Ali, N.S.; Gaafar, T.S. Design and Implementation of a Real-Time Smart Home Management System Considering Energy Saving. Sustainability 2022, 14, 13840. https://doi.org/10.3390/su142113840 [accessed Mar. 24, 2024]. [6] D. Cavone, B. De Carolis, S. Ferilli, and N. Novielli, “An Agent-based Approach for Adapting the Behavior of a Smart Home Environment,” WOA, 2011. [7] Chao Ruan, Li Zhou, Liangzhuang Wei, Wei Xu, Yandan Lin, Prediction model for indoor light environment brightness based on image metrics, Displays, Volume 82, 2024, 102662, ISSN 0141-9382, https://doi.org/10.1016/j.displa.2024.102662 [accessed Mar. 24, 2024].