Received April 5, 2021, accepted April 27, 2021, date of publication April 30, 2021, date of current version May 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076750

Voice-Activated Smart Home Controller Using Machine Learning

LEANDRO FILIPE[1,2], RICARDO SILVA PERES[1,2], (Member, IEEE), AND RUI MANUEL TAVARES[1,2], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, NOVA School of Science & Technology, NOVA University of Lisbon, 2829-516 Caparica, Portugal
[2]UNINOVA—Centre of Technology and Systems (CTS), 2829-516 Caparica, Portugal

Corresponding author: Ricardo Silva Peres (ricardo.peres@uninova.pt)

This work was supported in part by FCT/MCTES (UNINOVA—CTS) under Grant UIDB/00066/2020.

ABSTRACT The emergence of the Internet of Things concept has provided a great vision for the technological future, intending to enable the extraction and comprehension of information from the environment around us, making use of the interaction and cooperation between several technological devices. The example of Smart Homes, in particular, aims to integrate these devices into households, enabling the automation of tasks previously performed by humans, to simplify their daily lives and create a more comfortable environment. However, many of these devices fail to keep their promise, since they were not developed taking into account the frequent change of habits and tastes of the user, being necessary reprogramming of the device to follow the new behaviors. Taking this problem into account, this article presents the design and end-to-end implementation of a voice-activated smart home controller for intelligent devices, deployed in a real environment and validated in an experimental setup of motorized blinds. The architecture of the proposed solution integrates evolvable intelligence with the use of an Online Learning framework, enabling it to automatically adapt to the user's habits and behavioral patterns. The results obtained from the various evaluation tests provide a validation of the operation and usefulness of the developed system. The main contributions of this work are: I) design of a smart home controller's architecture; II) end-to-end implementation of a smart home controller and respective guidelines; III) open-source dataset of user behavior from the smart blinds scenario; IV) comparison between Online and Offline Learning approaches.

INDEX TERMS Internet of Things, machine learning, smart home, evolvable devices, smart blinds, voice-activated devices.

I. INTRODUCTION

In order to keep up with the latest technological trends, many manufacturers are now launching what are called ''smart devices'' on the market. The emergence of these devices has driven Internet of Things' vision of intelligent environments such as smart buildings and smart homes [1]. Increasingly, consumers are bringing home intelligent devices such as televisions, light bulbs, fans, heating systems, or even blinds. To further simplify the integration of all these devices, smart home hubs have emerged, often integrated with a personal assistant, being now possible to carry out a large number of tasks without much effort, often at a distance of a simple voice control (''Alexa, turn on the lights''), or even through simple automations that do not require any input from the user.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Zakarya.

The development of Artificial Intelligence techniques, such as Machine Learning (ML), has further boosted these ideas [2]. The objective of using these techniques in smart homes is to improve their efficiency and responsiveness to the needs and routines of the inhabitants by anticipating them instead of relying on direct commands or pre-programmed scenarios [3]. Furthermore, ML allows the home to recognize different inhabitants and profile their actions individually to learn their behavioral patterns. This information can be later used to detect anomalous behavior which could be a sign of user health or safety risk, or to make energy consumption more efficient in the house [4]. These paradigms have given rise to several new approaches in the context of intelligent home devices, however, despite their ability to drive the evolution of intelligence in these devices and to enable solutions capable of adapting to the habits and preferences of their users, many manufacturers do not fully accompany this

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

transition. Many of these devices are still based on classical and low intelligence programming. Developing smart home solutions with devices like these becomes a complex process, with the software having to be customized for every different home and family, and having to be updated as the family's lifestyle changes. The majority of users do not have the capabilities necessary to address the programming task, and the alternative of hiring professionals to perform such upgrades is quite expensive and inconvenient. The need then arises for a solution that allows devices to observe and adapt to the lifestyle and tastes of the inhabitants of the house in order to learn how to anticipate and accommodate their needs [5].

Taking into account the challenges that arise from the existing "smart devices" described above, it should be clear that a new solution is in need. As so, the following research question is posed:

RQ: *How can evolvable intelligence be implemented in a smart home in order to enable its automatic adaptation to a given user's habits and preferences?*

As a hypothetical solution to this problem, it is proposed the use of Online Learning techniques, an area of ML, to create a smart home controller capable of adapting to the habits and preferences of the user and capable of evolving with the same. In this work, a smart home controller was used to control motorized home blinds, creating a functional and convenient smart blinds solution.

The remainder of this document is structured as follows: In Section II it is presented the related work present in the literature and some challenges and gaps are drawn. Section III presents an overview of the implementation process (Subsection III-B) and the methods and tests used to validate functioning of the proposed solution (Subsection III-C). Next, in Section IV, the results obtained from the various tests will be presented, as well as a discussion and critical evaluation of the same. Additionally, some limitations of this work are raised in Section V. Finally, Section VI presents the conclusions that were drawn from this work, and some suggestions for future work are presented.

II. RELATED WORK

Even before the Internet of Things (IoT) concept appeared, there was already a futuristic vision of intelligent and connected houses capable of anticipating any needs of their residents, where the house's subsystems act autonomously, and always aware of each other.

Varied approaches can be found in the literature, ranging from traditional programming to ML-based solutions, hoping to improve comfort, energy efficiency, or even activity recognition, sharing a common goal of achieving an automated ubiquitous house and improving human life.

In an effort to better comprehend what has been previously done, a thorough analysis of different publications related to smart homes and smart home controllers was carried out. Table 1 summarizes the findings from this assessment, extracting from each article the key information required to address the research questions of the study. This includes the application setting and methods reported by the respective authors, as well as other relevant criteria. Their definitions in the context of the present work are provided below:

- *Real-time*: Applications that can receive data and act on it immediately;
- *Adaptability*: Refers to the ability of a system to learn autonomously and continuously, without human interaction;
- *ML*: Another criterion considered was the presence of ML in the system, and with it, what type of data was used;
- *Validation*: The validation method was also considered since real implementations can be of greater value than simulations;
- *Communication*: To better understand what technologies are being used in such applications, the communication method was analyzed;
- *Voice Control*: Finally, since IoT and smart homes are two of the main focus of this article, the presence of voice control is deemed as a necessary criterion.

From the analysis of Table 1, it is possible to arrive at several conclusions in regards to the current state and trends of smart home research. As can be seen, the Real-Time criterion is one of the most common design properties among the publications included in this analysis, being that more than half have included this aspect in their implementations. It is also worth noting that although several authors mention the importance of real-time and adaptable systems, many fail to implement them. Taking this into account, only the authors that show a concrete implementation of a real-time, adaptable system, are classified as such. Moreover, the lack of adaptable solutions found in the literature is considered a limitation of such systems. This will therefore be an important aspect to consider for the hereby proposed solution. Adaptability should be ensured in order to overcome such limitations from previous work.

As expected, most of the analyzed work uses ML techniques as their main method due to the inherent ability to adapt and learn autonomously. Nonetheless, it is interesting to see the coverage of different techniques that range from simple supervised learning applications to complex multidisciplinary systems. Furthermore, there is a careful consideration when choosing training data, being that real datasets are primarily used. There are although, those that still opt for synthetically generated data due to limitations like data availability, quality, and related issues. Additionally, synthetic datasets can pose as slightly unrealistic and oversimplified scenarios for testing and validation. For example, Dinata *et al.* [19] propose an improvement to the Very Fast Decision Tree (VFDT) [21] algorithm (to which it was called VFDT++), VFDT++), which proved to be the best at predicting a simple luminaire's state (ON or OFF) algorithms. The tests were performed on the dataset CASAS [20] with some modifications and restricted to one month of data, where it is assumed that the dataset is stationary, non-temporal, and there are no

TABLE 1. Overview of some related work present in the literature. Legend: RT - Real-Time; A - Adaptability; ML - Machine Learning; VC - Voice Control.

Author	Application	Methods	RT	A	ML	Data	Validation	Communication	VC
Pan et al. [6]	Device Control	Smart Phone as Remote Controller	✓	✗	✗	–	Simulation and Real Implementation	Bluetooth	✗
Javed et al. [7]	Smart Buildings; HVAC; Energy Saving	Random Neural Networks; Cloud Computing	✓	✗	✓	Real	Real Implementation	GSM	✗
Reyes-Campos et al. [8]	Device Usage Pattern Discovery	Supervised Batch Learning	✗	✗	✓	Real	Simulation	REST	✗
Babu et al. [9]	Smart Home Energy Saving	Fuzzy Logic	✓	✗	✗	–	Real Implementation	ZigBee	✗
Bajpai et al. [10]	Device Control	Speech Recognition	✓	✗	✗	–	Real Implementation	Bluetooth	✓
Madhu et al. [11]	Device Control	Traditional Programming	✗	✗	✗	–	Real Implementation	REST	✗
Abbas et al. [12]	Device Control	Supervised Learning; Decision Trees	✓	✓	✓	Real	Real Implementation	Serial Communication	✗
Di Giorgio et al. [13]	Smart Home Energy Saving	Event Driven Control	✗	✗	✗	Real	Simulation	ZigBee	✗
Natani et al. [14]	Smart Home Activity Recognition	Deep Learning	✗	✗	✓	ARAS dataset [15]	Simulation	–	✗
Safyan et al. [16]	Smart Home Activity Recognition	Supervised Learning; Artificial Neural Networks	✗	✗	✓	Synthetic	Simulation	–	✗
Mozer [5]	Activity Recognition; Energy Efficiency; Comfort Maximization	Reinforcement Learning; Artificial Neural Networks	✓	✓	✓	Real	Simulation and Real Implementation	Wired	✗
Cook et al. [17]	Smart Home Activity Recognition	Episode Discovery; Artificial Neural Networks	✓	✓	✓	Synthetic, Real	Simulation and Real Implementation	Wired	✗
Paulauskaite-Taraseviciene et al. [18]	Lighting Control	Supervised Learning; Artificial Neural Networks	✓	✓	✓	Synthetic	Simulation	–	✗
Dinata et al. [19]	Lighting Control	Online Learning	✓	✓	✓	CASAS dataset [20]	Simulation	–	✗

changes in user habits. As mentioned before, these restrictions to the dataset do not mimic real human behavior and could be improved by introducing more changes in user's behavior.

There is still a large number of researchers that test and validate their results only through simulations of real scenarios. This translates into the possibility of obtaining results that are not replicable in real environments. Although not always possible, the implementation and deployment in a real environment of a system allows for better evaluation and to derive observations with greater certainty and reliability.

Given that IoT has been one of the most popular topics in recent years, it is normal that more recent works adopt wireless communication techniques that are typical of this topic, while older solutions still use wired communications. Unfortunately, voice control is not yet widely adopted. It is believed that through speech recognition techniques implemented in a smart home controller, it may be possible to create a more practical, hands-free solution.

The vast majority of these solutions follow a centralized intelligence architecture where there are several devices responsible for taking information from the surrounding environment and acting on it, connected to a controller that contains all the necessary logic to monitor and automate them. Another option would be to integrate the intelligence directly into the actuating devices, enabling them to locally process the collected data, providing faster responses with no need to consume time in communication, however, architectures of this type are still not very common (at least in ML applications) due to the reduced computational capacity of microcontrollers and lack of quantitative analyses about the performance of common ML algorithms on such devices [22].

For the solution proposed in this article, all these factors were considered. The goal is to develop and deploy in a real environment a real-time, adaptable smart home controller using ML techniques (e.g. Online Learning). A centralized intelligence architecture will be used where the ML model, the voice control, and the user interface will be implemented in a single controller device connected to several actuator devices. The ML model will be trained with data collected from a real environment, and both it and the smart home controller as a whole will be validated in a real environment, under real conditions. Following the IoT vision, it was decided to use wireless communication.

III. MATERIALS AND METHODS

A. SYSTEM DESIGN

The proposed architecture described in this chapter seeks to take advantage of some characteristics of ML, namely, its ability to learn and to distinguish patterns not perceptible to the human being. More specifically, Online Learning techniques were used, in order to obtain an intelligent solution capable of adapting and evolving with the user. The proposed solution also integrates the possibility of voice control,

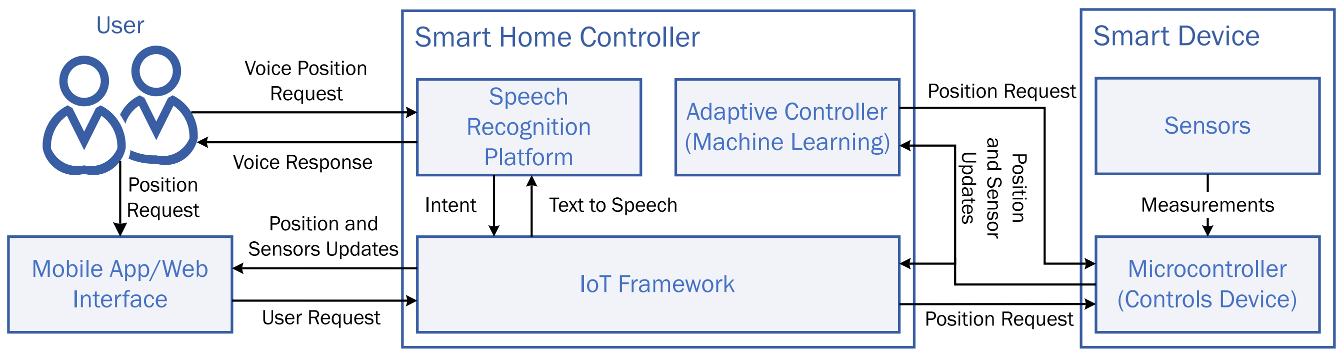

FIGURE 1. Smart home controller architecture overview.

control over a mobile application or through a web interface. All these features must be implemented in a single device (a Single Board Computer) small, light, compact and portable.

In order to predict and adapt to the user's behaviors, the controller must be composed of several modules, responsible for different aspects of the system. As can be seen in Figure 1, the proposed architecture for the smart home controller is composed of three main modules, a Speech Recognition Platform (SRP), an IoT Framework (IoTF) and the Adaptive Controller (AC). The SRP is responsible for converting the user's voice commands into machine perceptive intents. The IoTF serves as the middleman between the SRP and the several integrated smart devices, translating the user's intents into actions recognized by the devices for which they are intended. This module also serves to provide a graphical user interface, via a web page or mobile application. The interaction between the user and the controller is restricted to requests made through the SRP or one of the available graphic interfaces. Finally, the AC contains a ML model that is trained on the behavioral patterns of the user to predict the output of the device being controlled. Although security and privacy issues are not the focus of this work, normal functioning without an Internet connection was considered as a requirement for every module, to avoid having the user's voice processed elsewhere other than in the controller and to avoid having to rely on a good Internet connection. As such, all devices are interconnected using a local area network without Internet connection.

The smart home controller is capable of controlling most smart devices, however, validation was carried out with the use of developed smart blinds. The blinds are connected to the controller, more specifically to the IoTF and the AC to receive commands and send updates on its status. Both the SRP and the IoTF support other devices, however, the AC only supports the developed smart blinds. This solution integrates four sensors, two brightness and two temperature sensors. Each pair of sensors (one temperature and one brightness) is placed on the interior and exterior portion of the blinds to collect data in real time, both outside and inside the room where the blinds would be installed. This data will then be used as training features for the ML model.

B. IMPLEMENTATION

This subsection describes the implementation of the smart home controller and the smart blinds, detailed in Subsection III-A. The device chosen for the smart home controller was Raspberry Pi 4 Model B (4GB RAM version)[1] for its computational capacity given the physical size of the device, compatibility with other platforms used, native Python language support, among others. This device integrates the SRP, IoTF, and AC modules.

The SRP that was chosen is Rhasspy[2] since it is the best open-source platform found that meets the requirements of the system. This platform supports several communication protocols, including Message Queuing Telemetry Transport (MQTT) and Hypertext Transfer Protocol (HTTP), which were used in the development of this project. It can function completely disconnected from the Internet and offers compatibility with Home Assistant (HA).

For the IoTF, HA[3] was the used platform, as it is a an open-source smart home gateway that runs locally on the device where it is installed, offers integration with various devices and platforms, and is fairly simple to configure. Furthermore, it does not require an Internet connection for all functionalities and allows control of devices through a mobile application or a web-based graphical interface.

Two microcontrollers, a NodeMCU[4] development board, and an Arduino Nano,[5] were used to control the smart blinds, responsible for communication and position control of the blinds respectively. The rotation direction of the blinds is controlled by relays, and the sensors used to measure temperature and brightness are the MCP98085[6] and BH17506,[7] chosen due to their ability to communicate via Inter-Integrated Circuit (I^2C) with the possibility of multiple addresses. To facilitate the integration of the blinds with HA, the ESPHome[8] platform was used, a system specially developed to con-

[1]https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
[2]https://rhasspy.readthedocs.io/en/latest/
[3]https://www.home-assistant.io/
[4]https://nodemcu.readthedocs.io/en/latest/
[5]https://store.arduino.cc/arduino-nano
[6]https://ww1.microchip.com/downloads/en/DeviceDoc/25095A.pdf
[7]https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf
[8]https://esphome.io/index.html

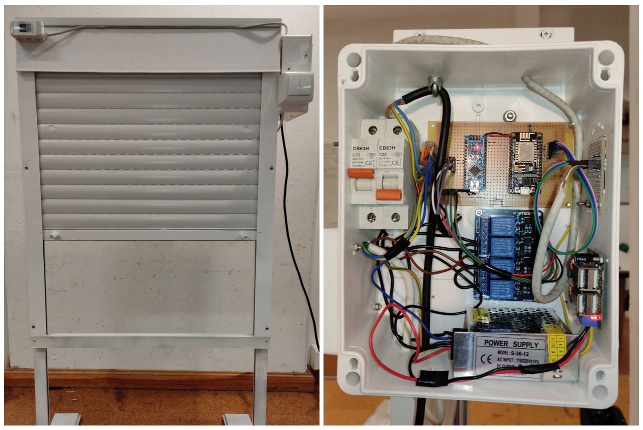

FIGURE 2. Finished prototype. (Left) front view. (Right) electronics enclosure.

trol ESP8266/ESP32 development boards (NodeMCU in this case) through HA. The main programming languages used in the development of the smart blinds were Python and C++.

To choose which ML model to incorporate in the AC it was necessary to collect data on temperature, brightness, and the position of the blinds in a real-life scenario. For this purpose, the first author of this article was used as a study subject and the necessary information was collected using a NodeMCU and temperature and brightness sensors. The sensors used are the same ones that are part of the smart blinds. The AC was developed from scratch using the Python language, one of the most common languages in ML implementations. The operation of this module strongly depends on the use of the Advanced Python Scheduler (APScheduler)[9] and the Eclipse Paho MQTT Python Client[10] libraries.

The total cost of the implementation of the smart home controller and the logic behind the smart blinds was kept under 100€ (one hundred euros), resulting in a low-cost, modular solution, following IoT's vision for smart devices. Two views of the smart blinds' finished prototype can be seen in Figure 2.

1) SPEECH RECOGNITION PLATFORM

As mentioned in Subsection III-A, the smart home controller must work without an Internet connection, which means the same for all its modules. As so, the chosen SRP was the Rhasspy V2.5 platform, *"an open source, fully offline set of voice assistant services"*[2], optimized for communication via MQTT, HTTP, and Websockets, and with built-in support for HA.

Rhasspy also offers a web interface that allows configuring, programming, and testing the voice assistant remotely. It supports several languages including English, and offers multiple services such as Speech to Text, Text to Speech, and Wake Word detection. At a high level, Rhasspy operates by transforming audio data into JavaScript Object Notation (JSON) events (as can be seen in Figure 3).

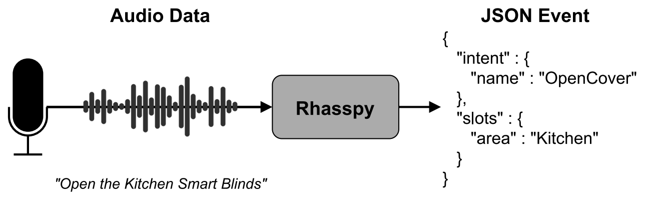

FIGURE 3. Processing of voice commands into JSON events.[12].

```
[OpenCover]
areas = (Kitchen | Bedroom)
open [the] (<areas>){area} [Smart] Blinds

[CloseCover]
close [the] (<OpenCover.areas>){area} [Smart]
    ↪ Blinds

[ControlCover]
(close|open|set) [the] (<OpenCover.areas>){area} [
    ↪ Smart] Blinds to (0..100){position} [
    ↪ percent]
```

Listing 1. Voice command specification excerpt.

Although it supports open-ended speech recognition, the best performance is achieved by defining all the possible voice commands beforehand in a compact, text-based format, specific to this platform. An excerpt of the used voice commands can be seen in Listing 1.

In the example above, each group of sentences specify a different user intent, identified by the name in brackets (`OpenCover`, `CloseCover`, and `ControlCover`). In the first intent, `OpenCover`, the rule `areas` contains the names of all the house divisions where smart blinds are installed, and so, there is no need to program a different intent for each set of smart blinds. In this example two blinds were used, one in the `Kitchen` and one in the `Bedroom`.

For audio capture, the ReSpeaker 2-Mics Pi HAT,[13] an expansion board for Raspberry Pi was used. This board, specifically developed for voice assistant applications, has two microphones and two sound output interfaces. For sound reproduction, a simple three Watt speaker connected to ReSpeaker 2-Mics Pi HAT's JST 2.0 interface was used.

2) IoT FRAMEWORK

For the IoTF, the HA platform was used, based on the Python language, this platform is able to run on a simple system like a Raspberry Pi. It provides the ability to track, control, and automate smart devices, through a customizable Graphical User Interface (GUI), accessible from a mobile app or a web page. Home Assistant has built-in integrations for multiple devices from different manufacturers and also multiple platforms (like Rhasspy and ESPHome, used in this project). As it runs locally, no Internet connection is needed for most functionality. Information about the status of an entity, like historical values of sensors and blinds current position, can be accessed through HA's Lovelace dashboard, available in

[9]https://apscheduler.readthedocs.io/en/latest/
[10]https://www.eclipse.org/paho/
[12]Adapted from https://rhasspy.readthedocs.io/en/latest/whitepaper/
[13]https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/

```
ControlCover:
  action:
    - service: rest_command.rhasspy_speak
      data_template:
        payload: >-
          Setting {{area}} Smart Blinds to {{
              ↪ position}} percent
    - service: cover.set_cover_position
      data_template:
        entity_id: >-
          {{"cover." + area|lower() + "_smart_blinds
              ↪ "}}
        position: >-
          {{position}}
```

Listing 2. Intent script example.

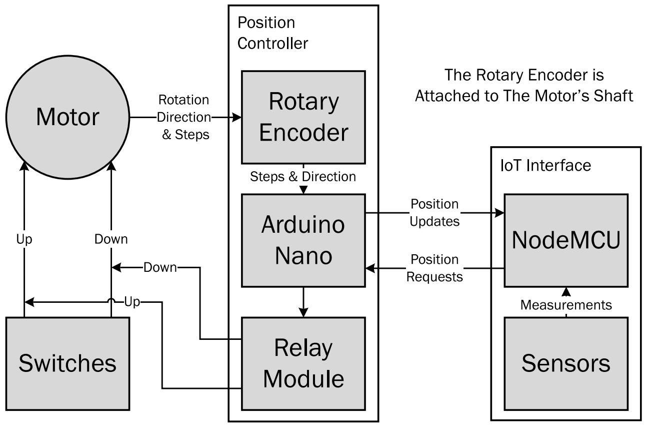

FIGURE 4. Smart blinds' topology.

web page and mobile app format. Intent handling is done by manually adding the intents and desired actions to an intent script file. As with all the configuration files in HA, this file uses the YAML language. Listing 2 contains an example of the user configuration for one of the intents specified in Listing 1.

The variables `area` and `position` are sent from Rhasspy as slots in the JSON event. These slots contain the area of the house the targeted smart blinds are installed and the position they're required to change to. These variables are then used to construct and send a proper response to Rhasspy's Text to Speech service, and to send the requested position to the smart blinds.

3) SMART BLINDS

The logic behind the blinds' control was divided into two modules, the Position Controller, responsible for making the blinds motor actuate to reach the requested position, and the IoT Interface, responsible for communicating with the smart home controller and reading the sensors. Figure 4 presents a simplified version of the implemented smart blinds' architecture.

The communication between these two modules is done with a Universal Asynchronous Receiver-Transmitter (UART) connection. The IoT Interface sends the desired position to the Position Controller, and once it is reached, the latter will inform the first with the final state. The Position Controller is also responsible for notifying the IoT Interface of when the user manually sets the blinds' position through the switches.

For the Position Controller, an Arduino Nano, a rotary encoder, and a three-relay module were used. The Arduino Nano is responsible for the logical part of the module, receiving information from the rotary encoder, and controlling the relay module. The rotary encoder is responsible for translating the motor rotation into steps and direction, perceptible by the microcontroller. Due to the cheap nature of the rotary encoder, some digital debouncing was necessary, for which it was used the code developed by J. Main.[14] Finally, the relay module is responsible for actuating the blinds' motor.

The IoT Interface is composed of a NodeMCU and four sensors. To program the NodeMCU, the ESPHome platform was used, which offers greater control of the ESP8266 and ESP32 microcontrollers through simple but powerful configuration files, and easier integration with HA. Through a simple configuration file in the YAML format, it is possible to connect NodeMCU with all the necessary platforms and sensors. Different protocols were used to communicate with the smart home controller. For the IoTF, ESPHome's proprietary protocol is used due to greater simplicity of integration with HA, and for the AC the MQTT protocol was chosen, making use of Rhasspy's internal MQTT broker. The sensors that integrate this module are responsible for collecting temperature and brightness data, from inside and outside the room where the blinds are installed. For this purpose, two MCP9808 temperature sensors and two BH1750 brightness sensors were connected to the NodeMCU using I^2C.

Each set of smart blinds is designed to work alone and to not depend on other smart blinds. In case the blinds fail to communicate with the smart home controller, the user will be notified in the graphical interfaces and can no longer control that set of smart blinds through voice or application, nevertheless, the smart blinds will remain operational through manual control and, will retain all information about positioning until a new connection is possible with the smart home controller. Furthermore, if the smart blinds fail to reach the desired position at any time, the user will be notified, and all operations regarding that set of smart blinds will come to a halt until the manual override of the user, thus protecting the system against external problems.

4) ADAPTIVE CONTROLLER

The AC is responsible for interpreting the behavioral patterns of the user and make predictions on them. As such, this module aims to collect information from the smart blinds connected to the controller and send the commands with the position predicted by its integrated ML model.

For this module, a discovery concept has been implemented, where blinds publish their information on the discovery topic and are automatically "discovered" by the AC. To each set of smart blinds, a different ML model is created, which is later trained and saved with a user specified frequency. The models undergo a training period (also specified

[14]https://www.best-microcontroller-projects.com/rotary-encoder.html

by the user) where no predictions are sent to the smart blinds. Once the training period has passed and the user has activated predictions, these are sent to the smart blinds specific for that model. The predictions made refer to the position the blinds will be at in the next timestep. However, if the blinds' position is changed in between predictions, the AC will acknowledge a wrong output and prevent additional messages from being sent to the smart blinds for a user specified timeout period. This behavior is based on the implementation made in [5] where the user's override of the controller's prediction is taken as a cue to learn.

As a safety measure, this module is prepared to recognize if one set of smart blinds fail to communicate. In that case, the AC will pause all operations for that specific set of smart blinds until it is able to reconnect. This way, the whole system can continue normal functioning even if one or more set of smart blinds fail.

C. TESTS AND VALIDATION

This subsection serves to describe the methods and scenarios used to validate and analyze the results obtained from the tests performed on different ML models. The tests were performed on three different scenarios in order to evaluate the performance of Online Learning models in comparison with the more traditional Offline Learning approach. These models will later be responsible for predicting the smart blinds' position.

For evaluation, the metrics used are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), being that the latter was given greater importance since it is more sensitive to outlier errors [23]. A rolling window was also used for these two metrics to analyze their value in an observation window equivalent to one week. Both these metrics use the same scale as the data being measured, which in this case, corresponds to the opening percentage of the blinds (%). Additionally, all the evaluations were performed using a filter that restrains the predicted value between 0 and 100, preventing the models from predicting impossible positions. Results without this addition were ignored since only score improvements came from its use.

1) DATASET CHARACTERIZATION

Within the context of ML, regression can be defined as the process of using data to predict a correct real-value continuous quantity [24]. In Online Learning, such process is derived from the analysis of streamed labeled training data, which later enables a model to learn and adapt in a continuous way [25].

For this project, a dataset was created containing temperature and brightness measurements of the inside and outside of a room equipped with the developed smart blinds. These features were labeled with the position the smart blinds were at for that given moment. The measurements were taken by the minute, over a period of approximately three months, following the real behavior of one of the authors. To increase the number of features and facilitate model convergence, additional features were added, like timestamp decomposition features (i.e. quarter, month, day of the month, etc.) and holidays and light flags which indicate whether the day is a holiday and whether the light of the room was on or off when the measurement was taken. Finally, since the purpose of the model is to predict the blinds' position at the next timestep, a shift of the label was performed and the last example (now labelless) was eliminated.

The final dataset is then left with a size of 121,751 examples, containing 14 features and one label. Other three versions of this dataset with larger time intervals between examples were created namely, five, ten, and thirty-minute intervals.

2) SCENARIO A: OFFLINE

For the Offline Learning scenario, five models were selected from the scikit-learn [26] library, Random Forest Regressor (RFR), Extra Trees Regressor (ETR), Passive Aggressive Regressor (PAR), K-Nearest Neighbors Regressor (KNNR), and Linear Regression (LR). The dataset was divided into two parts, the training and the test sets, with two thirds and one third of the total dimension respectively. A 10-fold cross-validation was performed on the training set to obtain training results. The models were then retrained with the same data and only predictions were made on the test set, to validate the performance of the models on unseen data. The same tests were conducted on the remaining three datasets (with five, ten, and thirty-minute intervals).

3) SCENARIO B: ONLINE

To obtain a faithful comparison, the models used in this scenario are equivalent adaptations of the models used in Scenario A, with some exceptions. The Adaptive Random Forest Regressor (ARFR), Hoefding Tree Regressor (HTR), Hoefding Adaptive Tree Regressor (HATR) from the Python scikit-multiflow [27] library, and the PAR, KNNR and LR models from the creme [28] library were used, keeping the hyperparameters of the models of the two scenarios as close as possible. For this scenario, the dataset was split in the same way as in Scenario A. To evaluate model performance, the prequential method was used on the training set. Similar to Scenario A, the trained models then perform only predictions on the test set to ensure good performance with unseen data. It should also be noted that for this scenario, the metrics used on the training set are slightly different. Since the models are trained incrementally, first predictions are very uneducated, as such, a one-week rolling window was applied to the MAE and RMSE metrics. Finally, these tests were repeated for the remaining three datasets.

4) SCENARIO C: TIME AND SPACE COMPLEXITY ANALYSIS

In Online Learning problems there is a speed-quality trade-off where complex models offer better results for more extensive training, as opposed to simple and fast models that operate at lower rankings while being able to handle the large flow of data received rather quickly [29]. With that being said, in this

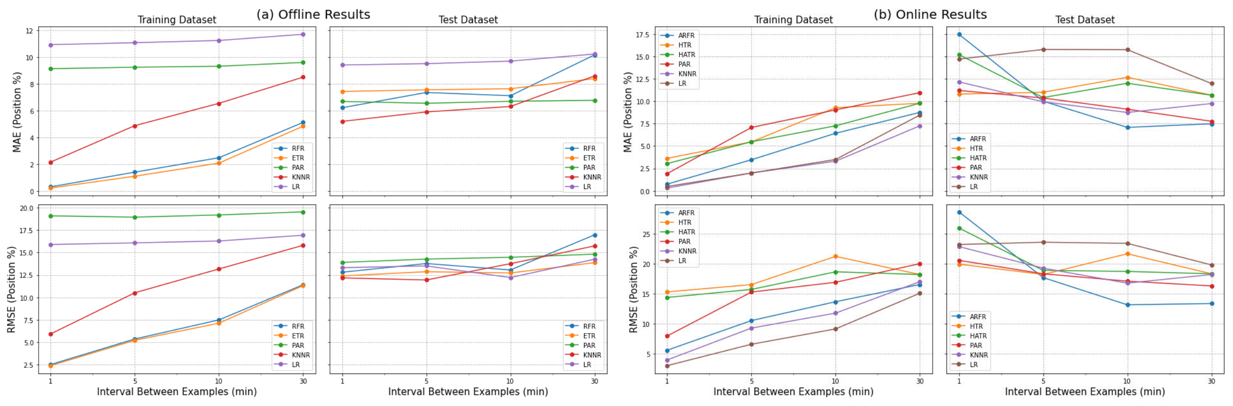

FIGURE 5. **Results obtained from: (a) scenario A; (b) scenario B.**

scenario, a time and space complexity analysis of the used online models is made. For this, all the models were trained 5-fold with an increasing number of samples and the respective training time and size in memory were measured. To maintain coherence was used the same training process of Scenario B. The results were analyzed using the big-O Python library[15] that estimates the complexity of Python code, and the size of the models was measured using the Pympler Python library.[16] All the models were trained on the same machine, a double core Intel®Xeon®CPU @ 2.30GHz with 13GB of RAM.

5) SCENARIO D: HOLDOUT

This scenario intends to compare the two ML techniques in a more realistic use case, were with the use of Offline Learning there would be a necessity to retrain the models periodically. Here, the offline models are trained with one month of data and are used to predict the following month. The real values of the predicted month are kept to train the models again, and thus continuing the cycle. For the online models, a prequential evaluation was conducted on a total of three months of data. To better understand the results, only the two best performing online models and equivalent offline models were studied. A thirty-minute interval between examples was used.

IV. DISCUSSION OF RESULTS

At first, both the models from Scenario A and B were evaluated only on the one-minute interval dataset, however, while analyzing the training results, it was noticed that the results from the analysis on the test set were much worse than the training results, which indicate the presence of overfitting. Furthermore, the models almost always predicted the value of the label of the previous example. It was then hypothesized that because of the great amount of consecutive same-labeled examples, the models would forget previously learned information upon receiving new examples, i.e. they suffered from catastrophic interference [30]. This hypothesis is then backed up by conducting the same tests on datasets with an increased interval time between examples.

[15]https://pypi.org/project/big-O/
[16]https://pypi.org/project/Pympler/

As can be seen in Figure 5a, the results obtained from Scenario A tend to worsen with the increase in interval time for the training set. This is partly due to a decrease in data quantity. However, the already higher error results from the test set only show a slight decrease in quality that suggests the reduction of overfitting. As for Scenario B, Figure 5b shows the same worsening of results seen in the previous scenario, but on the test set, a significant improvement in results can be seen for higher time intervals. This can be explained due to the fact that Online Learning models learn incrementally, and as such, the streamed data cannot be shuffled, as opposed to the data from Scenario A. The aforementioned behavior of predicting the value of the previous label can no longer be seen in the thirty-minute interval dataset. Typically only the prequential method is used to perform the validation in Online Learning, however, due to the suspicion in the presence of overfitting and catastrophic interference, it was necessary to conduct further analysis on unseen data, with no further training.

Comparing the results obtained from Scenarios A and B, as expected, the average results for the test set with a thirty-minute interval are better on the offline models, however, the best scoring online model (i.e. ARFR), offers a lower RMSE score on both the ten and thirty-minute interval datasets than any of the offline models. Taking a look at Figure 6, it is possible to see the evolution of the online model's RMSE, where a slight decrease in value over time is present, which indicates the models are adapting and effectively learning. This behavior is much better seen in datasets with lower intervals due to the higher example quantity, however, as mentioned before, these are not very realistic due to the learning problems encountered.

As for the better performing online models, the ARFR got the lowest metrics values for both the ten and thirty-minute interval datasets, being the first the better performing one (with 7.078% MAE and 13.172% RMSE), followed by the PAR online model (with 7.760% MAE and 16.316% RMSE). Although ARFR performs better than PAR, it has a significantly higher noise level, as it can be seen in Figure 7. The smoother response to the opening and closing of the blinds

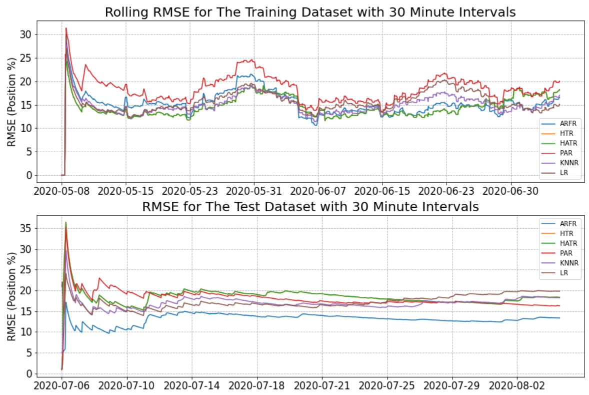

FIGURE 6. Evolution of the online models' RMSE for the training and test datasets with 30-minute intervals between examples.

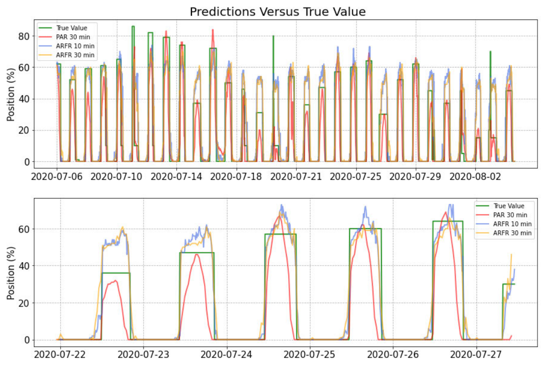

FIGURE 7. Predictions versus true value for ARFR and PAR online models. Second plot presents a zoomed portion of the first plot.

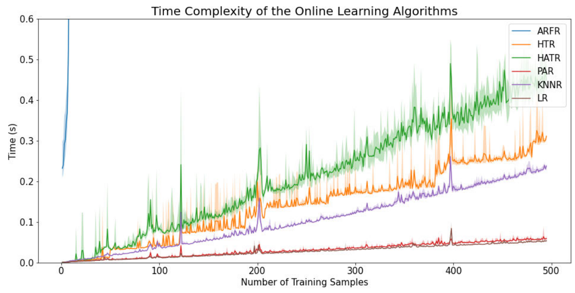

FIGURE 8. Time complexity estimation of the Online Learning algorithms. Traces represent the average value, while the highlighted sections represent the range of measured values.

TABLE 2. Time and space complexity analysis of the Online Learning algorithms. Legend: n - Number of training samples; k - Positive real number ($k > 1$); c - Positive real number ($0 < c < 1$).

Algorithm	Time Complexity	Space Complexity
ARFR	$\Theta(n^k)$	$\Theta(n^c)$
HTR	$\Theta(n)$	$\Theta(n^c)$
HATR	$\Theta(n \cdot log(n))$	$\Theta(n^c)$
KNNR	$\Theta(n \cdot log(n))$	$\Theta(n^c)$
PAR	$\Theta(n)$	$\Theta(1)$
LR	$\Theta(n)$	$\Theta(1)$

present in the PAR model, can also be considered a point in favor in this type of applications.

The obtained results from Scenario C were processed and plotted. Figure 8 shows an estimation of the time complexity of the Online Learning algorithms, this is, the training duration in seconds over the number of training samples, while Figure 9 shows an estimation of the space complexity for the same algorithms, measuring the size of the models in bytes. In Figure 8, traces represent the average of the 5 values measured, while the highlighted area represents the range of measured values. Figure 9 on the other hand, only shows traces since there is no size variance over the 5 runs. The average values of each algorithm were further analyzed to extract an estimation of the complexity in Big-O notation form (actually, since this estimation is based on the average value, this is called Big-Θ notation), present in Table 2. The process consists of fitting the measured values to common complexities and selecting the best fitting one. As mentioned before, these are only empirical estimations and do not necessarily represent the real Big-O notation for these algorithms, although, relevant observations can be derived from these results.

With that being said, since the ARFR is an ensemble model, it is safe to assume a greater complexity when compared with the linear PAR, both for time and space. This assumption is further backed up by the results present in Table 2. The ARFR has a polynomial time complexity, while the PAR is of linear time complexity. This not only means that generally, the ARFR will take longer times to train than PAR, but also that with the increase in the number of samples the performance difference between the two algorithms will be greater. It is also worth noting that while the PAR is one of the fastest algorithms (taking approximately 60 milliseconds to train with 495 samples), the ARFR is the slowest by far (taking almost 1 minute to train with 495 samples).

However, analysis of the time complexity does not suffice. In Online Learning, the model will only train with a single sample at a time, as such, the training time should not vary significantly. In contrast, the same does not occur with the space occupied by some models. Despite being trained incrementally with just one sample, the models have to retain information about what was learned. For some algorithms this does not mean an increase in size, however, specifically for tree-based algorithms, as they learn the trees will create more leaves and grow bigger, thus occupying more space. Even worse, ensemble algorithms like ARFR base their predictions upon multiple decision trees, adding complexity to the algorithm. These claims are supported by Figure 9, where it is possible to see a greater complexity of the tree-based models over other models. Just like before, the ARFR is the most resource-intensive algorithm, although in regards to space complexity this algorithm is not alone, as other three algorithms share the same complexity. Nevertheless, the comparison between ARFR and PAR shows a significant difference not only in resource expenditure but also

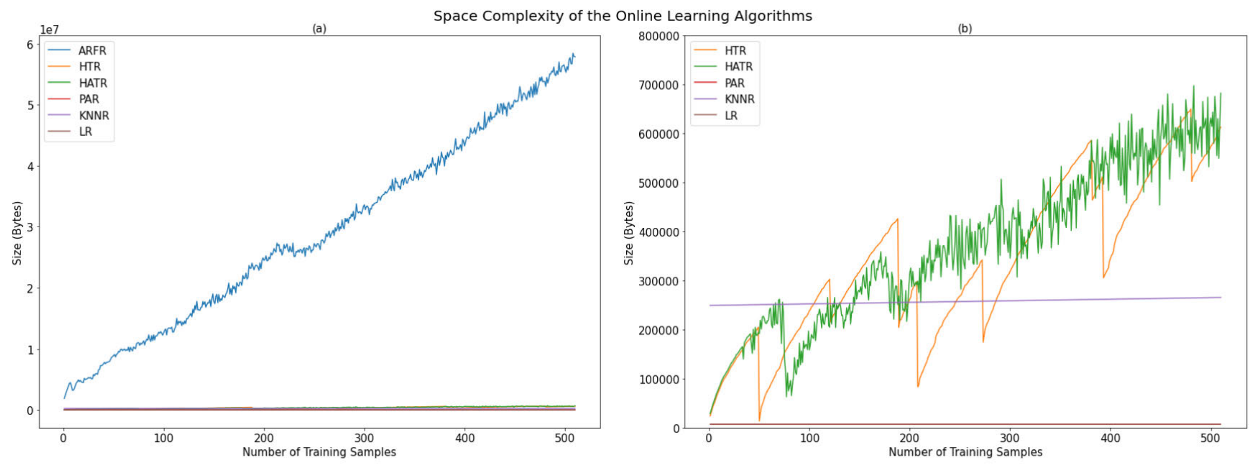

FIGURE 9. Space complexity estimation of the online learning algorithms. Plot (b) shows a zoomed in portion of plot (a).

space complexity, being that the ARFR is of a fractional power complexity (easily reaching a size of 55 MB with just 500 samples), while the size of PAR remains constant with the increase in the number of training samples (approximately 7 KB), which is perfect for IoT applications known for its limited computational and storage/memory resources.

It is also possible to see the effect that pruning has on the size of HTR. Pruning is a technique in ML that reduces the size of decision trees by removing sections of it that are non-essential, reducing the complexity of the final model. As can be seen in Figure 9b, pruning is marked by the big slopes present in the HTR trace, greatly reducing the memory size of the model. These slopes can also be seen in Figure 8, although with an increase in training time since pruning includes more steps to the algorithm.

Taking into account the final results of the tests performed in Scenario B, the models added to the AC's implementation are the ARFR and the PAR, giving the user the possibility of choosing between models and the respective time interval between examples. It is recommended to use the PAR (with a thirty-minute interval) as it has a lower time and space complexity (as seen in the results obtained from Scenario C) and lower noise. This also translates in a more resource-efficient solution.

Finally, as can be seen in Figure 10, the tests performed on Scenario D show better results for the online models than those obtained by the offline models, especially between the ARFR and RFR models. It is also possible to see a decline in the value of the RMSE over time for the online models, which suggests that better results could be obtained with more training. In addition, there is a decrease in the quality of the offline models' results for examples further away from the training date, which indicates a lack of adaptation. Although it may not be sufficient to conclude with certainty that Online Learning techniques are strictly superior to more conventional Offline Learning techniques, this experiment serves to demonstrate the better adaptability of online models when compared to offline models.

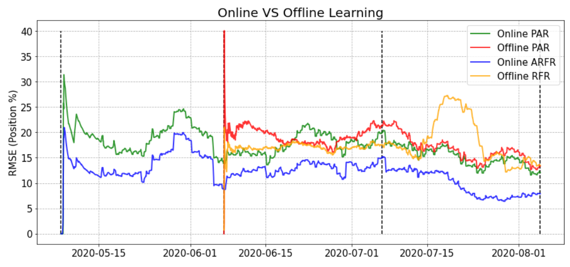

FIGURE 10. Online versus offline models.

As an indication of good performance, the achievement of a 5% average error (represented by MAE) after two months of training would mean the validation of the proposed hypothesis. However, although this milestone was not met, the obtained results appear to be quite promising. This shows the benefits associated with the use of Online Learning algorithms in the adaptive control of motorized blinds. It is foreseeable that with the increase in training examples, the proposed models would converge in a more accurate and better-performing solution.

V. LIMITATIONS

The restriction to three months of data may not be sufficient to cover for changes in user's habits since almost all the data was collected during only one quarter of the year, which limits the importance of this feature.The increase in interval time between examples greatly reduces data quantity, posing as a challenge for the models to learn with such a scarce resource. Increasing the data acquisition period would mean a larger dataset and a better validation would be possible, as well as the identification of other possible problems. The presence of additional features regarding user actions and location (like in [5] and [17]) is believed to also help in a problem of this nature.

Furthermore, during data acquisition, the outside sensors were placed inside a closed box with a see-through screen. The box was placed in a location exposed to direct sunlight. This originates some incoherence in the collected temperature data given the presence of a green-house effect inside the box. To overcome this problem, the placement of the temperature sensor inside a Stevenson screen in a shady place would be optimal.

Finally, the tests of Scenario C were performed in Google Colab[17] and thus were restricted to the 24 hour continuous running period that this service offers. Although the data gathered from this test was sufficient to make observations upon it, more data would help reach conclusions with greater certainty.

VI. CONCLUSION AND FUTURE WORK

This study proposes an architecture that focuses on the use of Online Learning to develop a smart home controller capable of controlling multiple connected devices according to the user's preferences and habits. The study of this controller was applied to a real case scenario, using developed smart blinds as a smart device. Needless is to say that the techniques used here would easily be adapted to any other smart home device. Here the presence of a common control platform for all devices opens up the possibility of more complex systems and the ability to communicate between all devices of a smart home, making the IoT's vision for intelligent environments a step closer to being accomplished.

Nonetheless, the results obtained from the various evaluation tests provide a validation of the operation and usefulness of the developed system. Additionally, a time and space complexity analysis of the used Online Learning algorithms was made. From these, and answering the research question raised in Section I, it is possible to verify that with the use of Online Learning techniques, evolvable intelligence can be implemented in a smart home controller, therefore creating a solution capable of automatic adaptation to the user's habits and behavioral patterns. Moreover, it is expected that the flaws and limitations of the architecture of the ML models tested here, will serve to drive any future work in this area. The datasets generated and analyzed in this study, are made publicly available for reproducibility purposes. This data can be found here: https://git.io/JtI5u.

To summarize, the main contributions of this work are:

I Design of a smart home controller's architecture;
II End-to-end implementation of a smart home controller and respective guidelines;
III Open-source dataset of user behavior from the smart blinds scenario;
IV Comparison between Online and Offline Learning approaches.

As future work, several modules present possibilities for improvements. In the case of the smart blinds themselves, hardware improvements would be beneficial for the accuracy of the blinds' position control. Also regarding this module, hardware redundancies could be considered in future work, where there could be a second device, ready to take action in case of failure of the first. Further research on the subject of SRPs that work entirely without Internet access could boost the development of more of these platforms, with better performance than the one used here. Concerning the ML models used, several different techniques could improve their performance, being one possibility the use of a hybrid architecture, where at a first moment, while there was still little data, the AC would be trained with offline models, and after a certain time, the transition to online models would take place to adapt the controller to changes in habits and preferences of the user. The investigation of other ML techniques such as Reinforcement Learning in the context of smart homes would be another possibility for future work. Additionally, although the system is not connected to the Internet, security and privacy issues are still relevant and should be explored in future work.

[17]https://colab.research.google.com/

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7123563/

[2] B. Qolomany, A. Al-Fuqaha, A. Gupta, D. Benhaddou, S. Alwajidi, J. Qadir, and A. C. Fong, "Leveraging machine learning and big data for smart buildings: A comprehensive survey," *IEEE Access*, vol. 7, pp. 90316–90356, 2019.

[3] D. Liciotti, M. Bernardini, L. Romeo, and E. Frontoni, "A sequential deep learning application for recognising human activities in smart homes," *Neurocomputing*, vol. 396, pp. 501–513, Jul. 2020.

[4] D. Schweizer, M. Zehnder, H. Wache, H.-F. Witschel, D. Zanatta, and M. Rodriguez, "Using consumer behavior data to reduce energy consumption in smart homes: Applying machine learning to save energy without lowering comfort of inhabitants," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 1123–1129.

[5] M. C. Mozer, "The neural network house: An environment that adapts to its inhabitants," in *Proc. AAAI Spring Symp. Intell. Environ.*, Dec. 1998, pp. 110–114.

[6] M.-S. Pan and C.-J. Chen, "Intuitive control on electric devices by smartphones for smart home environments," *IEEE Sensors J.*, vol. 16, no. 11, pp. 4281–4294, Jun. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7433919/

[7] A. Javed, H. Larijani, A. Ahmadinia, R. Emmanuel, M. Mannion, and D. Gibson, "Design and implementation of a cloud enabled random neural network-based decentralized smart controller with intelligent sensor nodes for HVAC," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 393–403, Apr. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7740096/

[8] J. Reyes-Campos, G. Alor-Hernández, I. Machorro-Cano, J. O. Olmedo-Aguirre, J. L. Sánchez-Cervantes, and L. Rodríguez-Mazahua, "Discovery of resident behavior patterns using machine learning techniques and IoT paradigm," *Mathematics*, vol. 9, no. 3, p. 219, Jan. 2021. [Online]. Available: https://www.mdpi.com/2227-7390/9/3/219

[9] V. S. Babu, U. A. Kumar, R. Priyadharshini, K. Premkumar, and S. Nithin, "An intelligent controller for smart home," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2654–2657. [Online]. Available: http://ieeexplore.ieee.org/document/7732459/

[10] S. Bajpai and D. Radha, "Smart phone as a controlling device for smart home using speech recognition," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 0701–0705. [Online]. Available: https://ieeexplore.ieee.org/document/8697923/

[11] G. M. Madhu and C. Vyjayanthi, "Implementation of cost effective smart home controller with Android application using node MCU and Internet of Things (IOT)," in *Proc. 2nd Int. Conf. Power, Energy Environ., Towards Smart Technol. (ICEPE)*, Jun. 2018, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/8659128/

[12] A. F. Abbas and M. Z. Abdullah, "Design and implementation of tracking a user's behavior in a smart home," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, Feb. 2021, vol. 1094, no. 1, Art. no. 012008. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/1094/1/012008

[13] A. D. Giorgio and L. Pimpinella, "An event driven smart home controller enabling consumer economic saving and automated demand side management," *Appl. Energy*, vol. 96, pp. 92–103, Aug. 2012, doi: 10.1016/j.apenergy.2012.02.024.

[14] A. Natani, A. Sharma, and T. Perumal, "Sequential neural networks for multi-resident activity recognition in ambient sensing smart homes," *Appl. Intell.*, Jan. 2021. [Online]. Available: http://link.springer.com/10.1007/s10489-020-02134-z

[15] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, "ARAS human activity datasets in multiple homes with multiple residents," in *Proc. 7th Int. Conf. Pervasive Comput. Technol. Healthcare Workshops (PervasiveHealth)*, May 2013, pp. 232–235.

[16] M. Safyan, S. Sarwar, Z. U. Qayyum, M. Iqbal, S. Li, and M. Kashif, "Machine learning based activity learning for behavioral contexts in Internet of Things (IoT)," *Program. Comput. Softw.*, vol. 46, no. 8, pp. 626–635, Dec. 2020. [Online]. Available: http://link.springer.com/10.1134/S0361768820080204

[17] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An agent-based smart home," in *Proc. 1st IEEE Int. Conf. Pervas. Comput. Commun. (PerCom)*, Mar. 2003, pp. 521–524. [Online]. Available: http://ieeexplore.ieee.org/document/1192783/

[18] A. Paulauskaite-Taraseviciene, N. Morkevicius, A. Janaviciute, A. Liutkevicius, A. Vrubliauskas, and E. Kazanavicius, "The usage of artificial neural networks for intelligent lighting control based on resident's behavioural pattern," *Elektronika ir Elektrotechnika*, vol. 21, no. 2, pp. 72–79, Apr. 2015. [Online]. Available: http://eejournal.ktu.lt/index.php/elt/article/view/8772

[19] I. B. P. P. Dinata and B. Hardian, "Predicting smart home lighting behavior from sensors and user input using very fast decision tree with kernel density estimation and improved Laplace correction," in *Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst.*, Oct. 2014, pp. 171–175. [Online]. Available: http://ieeexplore.ieee.org/document/7065885/

[20] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, Jul. 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6313586/

[21] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association Computing Machinery, 2000, pp. 71–80, doi: 10.1145/347090.347107.

[22] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria, "Machine learning on mainstream microcontrollers," *Sensors*, vol. 20, no. 9, p. 2638, May 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/9/2638

[23] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?–arguments against avoiding RMSE in the literature," *Geoscientific Model Develop.*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014. [Online]. Available: https://gmd.copernicus.org/articles/7/1247/2014/

[24] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[25] R. S. Sutton and S. D. Whitehead, "Online learning with random representations," in *Machine Learning Proceedings 1993*. Amsterdam, The Netherlands: Elsevier, 1993, pp. 314–321. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9781558603073500472

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[27] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *J. Mach. Learn. Res.*, vol. 19, no. 72, pp. 1–5, 2018. [Online]. Available: http://jmlr.org/papers/v19/18-251.html

[28] M. Halford, G. Bolmier, R. Sourty, R. Vaysse, and A. Zouitine. (2020). *Creme, a Python Library for Online Machine Learning*. [Online]. Available: https://github.com/creme-ml/creme

[29] H. Oosterhuis and M. de Rijke, "Balancing speed and quality in online learning to rank for information retrieval," in *Proc. ACM Conf. Inf. Knowl. Manage.* New York, NY, USA: Association Computing Machinery, Nov. 2017, pp. 277–286, doi: 10.1145/3132847.3132896.

[30] M. McCloskey and N. J. Cohen, *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem* (Psychology of Learning and Motivation), vol. 24, G. H. Bower, Ed. New York, NY, USA: Academic, 1989, pp. 109–165. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079742108605368

LEANDRO FILIPE was born in Almada, Portugal, in 1997. He received the B.S. and M.Sc. degrees in electrical and computer engineering from the NOVA University of Lisbon, Portugal, in 2018 and 2021, respectively.

Since 2019, he has been working a Teaching Assistant with the NOVA School of Science & Technology. He is currently a Ph.D. Student with the NOVA University of Lisbon. His research interests include artificial intelligence, machine learning, and intelligent and predictive systems.

RICARDO SILVA PERES (Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the NOVA University of Lisbon, Portugal, in 2015 and 2019, respectively, with a specializing in the application of artificial intelligence in smart manufacturing.

Since 2014, he has been working as a Researcher with the UNINOVA—Centre of Technology and Systems focusing on the development of intelligent and predictive manufacturing systems. He is currently an Invited Professor with the NOVA University of Lisbon, integrating the Department of Electrical Engineering, NOVA School of Science & Technology. He has participated in several national and international research projects, including FP7 PRIME, H2020 PERFoRM, H2020 OpenMOS, H2020 GO0D MAN, and H2020 AVANGARD. He is the author of several publications in highly ranked international scientific journals and conference proceedings (peer-reviewed). His research interests include predictive manufacturing, industrial artificial intelligence, cyber-physical systems, and multi-agent systems. He has also been a member of the IEEE IES Technical Committee on Industrial Agents, since 2018, and the IEEE Standards Association P2805.1/2/3 Edge Computing Nodes Working Group, since 2019.

RUI MANUEL TAVARES (Member, IEEE) was born in Oeiras, Portugal, in 1975. He graduated from the NOVA University of Lisbon, Portugal, in 1998, where he received the M.Sc. and Ph.D. degrees, in 2001 and 2010, respectively.

He has been with the Department of Electrical and Computers Engineering (DEEC), NOVA School of Science & Technology, NOVA University of Lisbon, since September 1998, where he is currently an Assistant Professor. Since 1998, he has also been working as a Senior Researcher with the UNINOVA—Centre of Technology and Systems (CTS). From 1998 until 2001, he worked as an Assistant Researcher with the Interoperability Supported by Standards group (GRIS), UNINOVA, where he has actively participated in several national and joint European cooperative projects in science and technology (e.g., ESPRIT IV 22056—FunSTEP). Since 2001, he has also been with the Analog Microelectronics Design Group of CTS/UNINOVA, where he participated in, and has been actively participating in several national and joint European cooperative projects in science and technology (e.g., H2020—Proteus). His research interests include electronic design automation of analog circuits and systems (EDA/CAD), and the design of low-power and low-voltage analog integrated circuits. He has published several articles in international leading conferences, international journals, and a book, and has been serving as a reviewer for many IEEE Conferences. He is also a member of the IEEE Circuits and Systems (CAS) Society and IEEE Solid-Sate Circuits (SSC) Society. He participated in the organizing committee of the Seasonal Schools: CAS4IoT'2016 and CAS4IoT'2018; and the International Conference IEEE ISCAS'2015.