Integration of Mobile Devices in Home Automation with use of Machine Learning for Object Recognition

Rui Passinhas
ISCTE – Instituto Universitário de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
rjsps@iscte-iul.pt

Rui Neto Marinheiro
ISCTE – Instituto Universitário de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
rui.marinheiro@iscte-iul.pt

Paulo Nunes
ISCTE – Instituto Universitário de Lisboa
Instituto de Telecomunicações
Lisbon, Portugal
paulo.nunes@iscte-iul.pt

ABSTRACT

The number of smart homes is increasingly expanding, with even more connected devices and available control options. Mobile devices have unfortunately been up to now generally regarded as mere remote controls in these environments.

This paper addresses this shortcoming, by presenting a novel integration architecture and prototype where the potential of mobile devices sensors can be better explored in home automation platforms, in particular by detecting objects in the information collected by their cameras that subsequently allow for users to interact with them in an intuitive way. The detection is performed at the mobile side, using a lightweight machine learning solution.

The obtained accuracy and processing time are comparable to that obtained at server side. But the advantage here is that the interactive experience of users can be dramatically improved, with the absence of round-trip time required if server processing would be used.

CCS CONCEPTS

• **Computer systems organization~Sensors and actuators**
• Human-centered computing~Ubiquitous and mobile devices
• Computing methodologies~Object recognition

KEYWORDS

Internet of Things, Smart Homes

ACM Reference format:

Rui Passinhas, Rui Neto Marinheiro and Paulo Nunes. 2020. Integration of Mobile Devices in Home Automation with use of Machine Learning for Object Recognition. In *Proceedings of 10th Euro American Conference on Telematics and Information Systems 2020 (EATIS 2020). ACM, Aveiro, Portugal, 8 pages.* https://doi.org/10.1145/3401895.3401925

1 Introduction

The main goal of the system proposed here is to promote the integration of mobile devices into current home automation solutions, considering the device as a sensor, in particular the use of its camera in a user-friendly way in order to perform interactive tasks based on image recognition.

EATIS 2020, November 25–27, 2020, 3810-193 Aveiro, Portugal

ACM ISBN 978-1-4503-7711-9/20/05.
https://doi.org/10.1145/3401895.3401925

Analyzing the related work in the smart homes sector, it is noticeable that an effort is being made in progressively using the mobile device to interact with the home automation platforms and control our home appliances in a user-friendly way. Still and all, there is no bridge between the mentioned subjects and this paper aims to fill that gap, providing the device the ability to perform on its own image recognition tasks, from an intuitive application, and consequently take actions in real-time in a smart home environment.

This integration strives to go beyond the scope of the normal home automation mobile applications as far as the interaction with the user happens and as well as the concept of object recognition with real-time detection is attached.

Thus, the system presented consists in a sequence of modules which provide the foundation to the whole workflow that needs to occur since back from the user to the end smart home devices. In a high-level system architecture overview, the system presented consists in a mobile application where the user interacts with the home devices. This application is where the object recognition happens in real time, analyzing the surrounding environment and enabling the user interactions with the devices. Then, the application interacts with a message broker, which connects to the home automation platform and consequently triggers events in the actuators of home devices themselves, according to the existing rules.

In order to validate the proposal presented here, a prototype has been implemented where real case scenarios were considered, analyzing the identification and interaction with the smart home devices, measuring the obtained processing delay and accuracy.

Results show that the proposed solution stresses that an inference machine learning model can be used in real-time processing, integrated inside an android application providing instantaneous visual feedback. This also leads to real-time action triggering in automation platforms, that provide home device integration, and consequently built a seamless and uninterrupted information workflow.

The integration proposed here significantly contributes to demonstrate that: (i) a better integration of mobile devices in home automation is possible, in particular by exploring the potential of their sensors; (ii) lighter machine learning models, implemented at the mobile device side, can be used real-time detection, to improve user interactive experience.

The remaining part of the current paper is organized as follows. Section 2 presents a literature review in this area. Section 3 discusses the design of the architectures proposed here. Section 4 details the implementation of the prototype used for the validation and how it was used to perform tests and obtain results. Finally, Section 5 presents general conclusions about the current contribution and Section 6 promising future work.

2 Related Work

In the field of the Internet of Things (IoT), a myriad of proposals has suggested the integration of smart mobile devices in home or building automation solutions such as [11][10]. The authors in [11] have targeted a low-cost mobile approach to Smart Homes. The proposed system aims to access and control devices in a Smart Home using a smartphone app. With the integration of wireless communication and cloud networking, the goal is to provide users with the possibility to control all the electrical smart appliances, devices and sensors using a friendly interface in a smartphone from remote locations. The proposed system is composed by a base station implemented in an Arduino Mega connected to Wi-Fi and multiple satellite stations based in Arduino Uno boards with Radio Frequency modules to communicate with the base station.

The authors in [10] developed a mobile Android app than can access information of all the appliances in a smart home and allows to interact with them, manually or automatically from scheduled events. Implementing both AES and RSA algorithms, the app was designed taking into concern common security issues. The system is composed of two main blocks, an outdoor environment and an indoor environment. The outdoor one consists of the end user and the application cloud server whereas the indoor one has the access point, the hosts and all the nodes. Even though communication between the two environments is done using the Internet with encrypted information, communication inside indoor environment uses Zigbee. The application can be accessed in real-time in any remote location and has notifications, QR Code and Auto-Lock features.

The solutions presented so far are representative of the majority of proposals for the mobile devices' integrations in IoT platforms, where they are regrettably considered as plain remote controls. The potential of those devices, with their many available sensors, in particular the camera, could be better explored.

The authors in [8] present an application that processes images in real-time for object detection, based on OpenCV libraries. The object detection system was developed and trained on a Windows machine and implemented on a Texas Instruments embedded platform. It adopted a cascade classifier based on Haar-like feature in order to reduce the computational time and to increase the speed of object detection. The system was trained with a dataset of around 4000 images from different angles and positions. This type of approach could be adopted, by using mobile devices' camera to collect images that could be sent to a back end central system to process them. However, the round-trip time penalty in a solution like this does not provide a good interactive experience when users have to interact with controlled devices in real-time.

Machine learning is another alternative approach for object detection. Software solutions, such as TensorFlow, provide a viable framework that can be used embedded in mobile devices [7] for object recognition [3]. Even though not related directly to IoT, the author in [2] detailed how to effectively train an object detector to accurately recognize Raccoons around his house. The system was developed using TensorFlow Object Detection API and was trained with a specific dataset of about 200 racoon images collected and labeled by the author. The training process was done with an object detection training pipeline based on a Single Shot MultiBox Detector [2] network with default settings and adapted to only one class. Since the image dataset was small and low training time was used, the detector does not recognize every single racoon, but it can deliver decent results with relatively good accuracy. The ability of performing object detection on the device, and not in distant cloud servers, suites better for interactive applications required for home automation.

3 Architecture of the Proposed Solution

The main goal of the proposed system is to promote the integration of the mobile devices into the home automation environment, providing the user the ability to interact intuitively with the system. This integration strives to go beyond the scope of the normal home automation mobile applications as far as the interaction with the user happens and as well as the concept of object recognition with real time detection is attached.

Thus, the proposed system consists in a sequence of modules which provide the foundation to the whole workflow that needs to occur since back from the user to the end devices. In Figure 1, a high-level system architecture is presented in which can be identified the user representation, the mobile cluster, the automation platform aggregate and in the opposite end, the smart devices connected inside home.

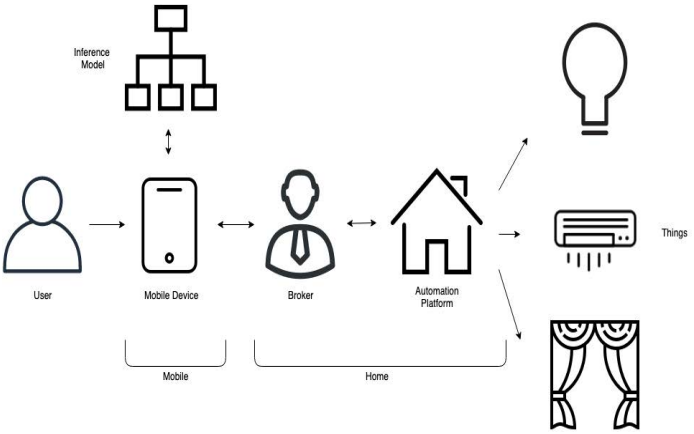

Figure 1: High-Level System Proposed Architecture

3.1 Mobile Module

This module serves has the entry point to the user in the whole workflow. All the interactions will happen within an application

installed in the mobile device and the actions taken inside it will trigger series of events throughout the system.
As illustrated in Figure 2, the mobile component is composed by a mobile application and a re-trained model integrated into the application. This integration occurs in real-time, providing the user instant feedback and therefore, according to the scenario, specific commands and actions will be sent through the channel of communication, existing between this module and the home automation one.

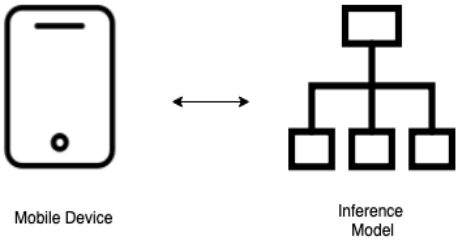

Figure 2: Mobile Module Architecture

3.1.1 Mobile Application. The mobile application will be installed in the user device and its main role is to collect user interactions, interpret the data collected and establish the communication with the automation platform. This application must furthermore allow the communication to be established between the mobile device and the home automation platform. To do so, it will need to play the client role in a communication protocol between both, sending information to the platform.
3.1.2 Inference Model. The main purpose of the proposed integration is the ability to use the mobile device sensors to interact intuitively with the smart devices. One of the major information collectors which can be found in every mobile device is the camera. So, to take advantage of all its potentiality, a way of interpreting the data collected by the camera sensor needed to be found.
A possible solution is to implement traditional computer vision algorithms, using feature extraction and pattern recognition processes to manually identify objects. Notwithstanding the fact of these techniques have proven themselves to be reliable and effective, a more promising approach was taken in consideration. Here is where the Machine Learning component comes into play. The use of a machine learning library compatible with the architecture proposed, able to load and process data, build, train and re-use models with easy deployments is a valid solution.
In addition to provide the ability to re-train a neural network without the need of sophisticated and powerful hardware, the biggest overall advantage off this approach is the possibility to run machine learning models on mobile devices within a lightweight and mobile solution. Using on-device machine learning, the system architecture stays simpler, without the need to make consecutive server calls to evaluate information. That would require constant data streaming resulting in more energy consumption, higher latency and extra processing with back and forth communication also having to consider the possibility of data loss in it. This also prevents that personal data is sent to third-party providers.

3.2 Communication

The connecting point between the main module described before and the later one described ahead is the communication established between both. The communication protocol to use between the device and the platform depended heavily on the choice of the automation platform and the application ecosystem since each platform and operating system supports a different set of protocols and integrations. Despite the choice, it has to assure that the communication between the two has low bandwidth, low latency and good performance.
Within the system architecture context, one component will act as a publisher, one as a broker and the other as a subscriber (Figure 3).

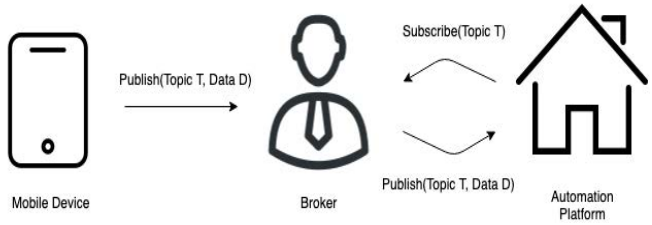

Figure 3: Communication Architecture Workflow

The most logical way of implementing the protocol architecture in this scenario is being the mobile device the subscriber responsible for publishing data to a certain topic which is subscribed at the automation platform end. By doing so, the mobile device can constantly push updates on state changes and user interactions knowing that these will be received on the automation platform, that is listening for data in the specific subscribed topics.

3.3 Home Module

The last, but not the least important element in the proposed system architecture, is the home module. This one houses two main components, the broker, responsible for the communication, and the home automation instance, responsible for the integration of the smart devices and sensors. As displayed in the Figure 4, both are housed inside the same installation platform.

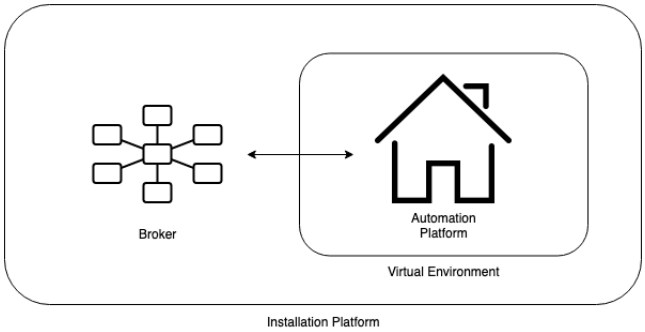

Figure 4: Home Module Architecture

One major concern was to have the broker and the automation platform installed in the same hardware so the need of extra hardware could be avoided. Consequently, the choice of all three

components was made having in mind the need of each one being compatible between them.

3.3.1 Broker. The broker will be responsible to manage all the publish and subscribe calls made and therefore maintain the communication between the involved elements.

Since in a public broker, any device or entity can publish and subscribe to any topic on it, and that most home automation platforms analyzed did not have a broker by default, the securest and easiest way to integrate this node into the architecture was relying on a private broker where only the devices with given permission can publish and subscribe to the topics managed by that broker.

Figure 5 represents a Broker placement within the system architecture, receiving subscribing requests from the Automation Platform and therefore delivering the messages there upon a data publish receipt from the mobile device, consequently establishing the connection for data transfer between both.

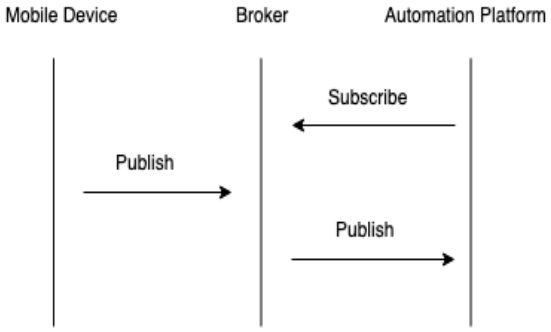

Figure 5: Broker Communication Role in System Architecture

3.3.2 Automation Platform. Being a core element of the system, the automation platform is a key element in the system potentiality. Architecture wise, as referred in the beginning of this chapter, the home automation platform is installed alongside the broker, also already mentioned.

Bearing in mind the intended integration, a good implementation choice needs to check all the boxes: being open-source, developed in a well-known easy to learn language, having a strong base community of users and the possibility of being installed on a low processing power and costless device.

As illustrated in Figure 6, the platform will also have the smart home devices connected to itself, being able to process rules, send commands, perform operations and state changes according to the information sent by the user from the mobile device and transmitted through the broker.

4 Implementation and Validation

In order to demonstrate the proposed integration of mobile devices in home automation, the system architecture had to be implemented in a prototype that allowed for the validation of the solution here presented.

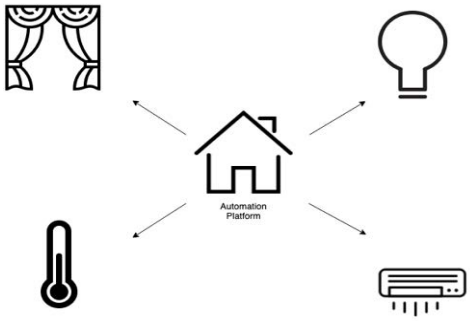

Figure 6: Smart Devices Connected to Home Platform

4.1 Implementation

At first, the machine learning component was developed and converted into the desired format. After that, the mobile application responsible for integrating the model in real time was built. Thereafter, the automation platform was installed and configured and consequently the communication processes were implemented. These three major stages of system development and implementation are the keys to a successful operation of the proposed solution.

4.1.1 Inference Model. To implement the architecture above described and having in mind the necessary requirements, a machine learning model had to be trained, converted and later deployed on the mobile device.

Regardless of the final solution, a custom Machine Learning model training process requires an image dataset as an input. As stated before, for the model to perform in a certain scenario, it will have to be trained with enough data to learn the patterns and desired features. In order to start the initial approach considering object detection, a dataset of images, divided in 3 groups, was gathered and the choice of 3 objects that can be found in most of so-called smart homes was made: Bulbs (608 images), Air Conditioners (508 images) and Window Shutters (808 images).

To begin the training process, the following elements were needed: TFRecord Files, Label Map, Model Config File, Pre-Trained Model. An already trained model can be used as a checkpoint to transfer learning and to retrain the final layers providing the data wanted.

After setting up the environment, the next step was the re-training process. This one was done using MobileNets [1] which are a set of computer vision models optimized to TensorFlow, designed to obtain high accuracy using limited computation power and restricted resources, building in this way a lightweight convolutional neural networks. For this reason, it isn't expected a significant increase on energy consumption of the mobile device.

To begin the training, a Python Script obtained from TensorFlow repository was used which is responsible for the download of the pre-trained model and consequently add the new layer to be trained on the given dataset. The default number of iterations (4000) was used and the script was executed afterwards. The Script output reported a Final test accuracy of 91.9%.

As illustrated in Figure 7 and Figure 8, when consulting TensorBoard, a monitoring tool included in TensorFlow, during and after the training, a set of outputs could be evaluated:

1. Accuracy – Divided in training accuracy and validation accuracy, these values represent, respectively, the percentage of images labeled correctly and the validation precision on a set of images chosen. In Figure 7, the accuracy, represented in the y-axis, is a function of the training progress, represented in the x-axis. The orange line represents the training accuracy of the model while the blue line exhibits the validation accuracy. As the validation accuracy remains constant as the training accuracy increases, we can say that the model did not entered in overfitting which is a scenario when the model is learning more of the training data proprieties than the data patterns itself.
2. Cross Entropy – In short, cross entropy is a positive loss function which tends to zero as the neuron improves computation of the desired output, y, for all training inputs, x, as represented in Figure 8.

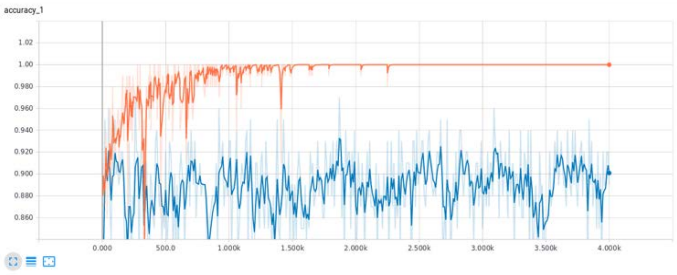

Figure 7: Training Accuracy (TensorBoard)

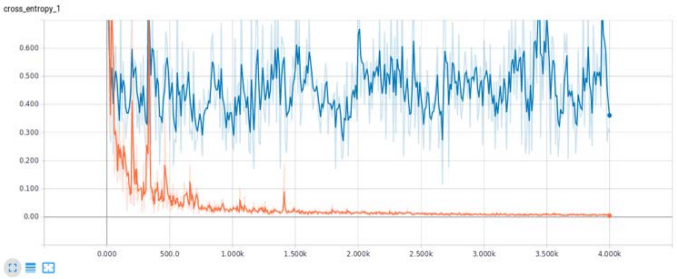

Figure 8: Training Cross Entropy (TensorBoard)

After obtaining the final re-trained graph from the previous training output, the model needed to be converted and optimized to run on the mobile device. The process uses TensorFlow Lite and its tools, namely, a TFLite Converter and a TFLite interpreter. After executing the python scripts, the optimized_graph.lite file was generated under the output path defined.

4.1.2 Mobile Application. After generating the .lite file, the customized model was then prepared to be integrated inside the Application. As referred above, the application is an Android Application and it was developed using Android Android 5.0, granting minimum compatibility with approximately 88.2% of devices [9]. The application has to handle two main tasks: the interpretation of the model in real-time and the communication with the forward elements of the architecture.

The first step was to integrate the model already trained, optimized and converted into the TFLite interpreter. After successfully integrating the inference model in real-time with data collected from the camera main sensor, the device needs to send commands to the platform in order to trigger the user desired actions. This communication process was implemented using Message Queuing Telemetry Transport (MQTT). The solution was to implement the Paho Android Service [5], which provides an interface to the Original Paho Java MQTT Client. This allows to encapsulate the connection inside a service and to run it in background along the Android Activities providing reliability in MQTT connections and message receiving and sending.

4.1.3 Home Automation Platform. Having the inference model trained and integrated inside the android application, the third and last step to complete the system implementation was the installation of the automation platform on a small form factor processing board, including the MQTT broker to allow the platform to receive the messages originated in the mobile device and consequently trigger automation rules and perform actions. The components were installed in a Raspberry Pi 3B+ inside a python virtual environment, providing the flexibility needed to the system implementation.

For the prototype the used automation platform was HomeAssistant [6]. Despite the fact that it is the possible to define an in-platform broker and that there are public brokers available, a private MQTT broker was used to implement the MQTT communication on the server side, to have more control on the integration and security management. The choice relied on Mosquitto [4] to implement the machine-to-machine messaging protocol. After the installation, the system can be started and is able to receive publish and subscribe requests. The broker runs as a service and has to be started whenever the system boots.

One big advantage of Home Assistant is the ability to define automation rules. An automation rule contains 3 main blocks: Trigger; Condition and Action. One of the trigger types is the MQTT trigger and is fired when a specific message is received in a specific topic. This way, automation rules could be created to make the desired system actions according to the MQTT messages received.

Figure 9 shows an example of an automation rule where an action is triggered when a message with the payload “switch” is received on topic “room/switch/bulb” with no conditions implicit. The action will take place on the bulb described in the entity tag with the service light.toggle, which is responsible for toggling the bulb state. Additionally, the system Web interface has a tool to create the automations, displaying the list of triggers, entities available, actions and the respective services.

4.2 Validation Results

Having completed the implementation of the previous steps, a prototype of the proposed system was obtained. Then, a comparison between the manual inference at server side and the on-

device machine learning final solution was possible, considering the obtained processing times. It was also possible to perform accuracy tests using real application scenarios. In the end a discussion over validation results provided by the implementation prototype is followed.

```
- id: '1560184994896'
  alias: Ligar-Desligar Luz
  trigger:
  - payload: switch
    platform: mqtt
    topic: room/switch/bulb
  condition: []
  action:
  - data:
      entity_id: light.yeelight_color1_7c49eb138f90
    service: light.toggle
```

Figure 9: Bulb Automation Rule

4.2.1 Prototype Implemented. The developments made aim at a hypothetical scenario where a user can control the devices existent in his/her home, through the camera user interface having visual representation of the actions, while providing intuitive interactions. Looking at the implementation obtained, the major visual results come from the mobile application and the device connected to the platform.

The implementation described resulted in a mobile application which, after installed in and android device, is where the user will interact. Figure 10 represents the application main screen, containing the camera viewfinder, a label containing the object identified and the accuracy of the evaluation alongside the buttons to take actions on that device when evaluated with accuracy over a pre-defined threshold.

Figure 10: Application Main Screen Example

In the screenshot presented in Figure 10, it is visible that the user was pointing the device at a light bulb and since the evaluation result is constantly returning values close to 100% the button to switch on the device is presented, which represents the action passible to be taken at that moment for that device. When the button is pressed, the whole process described before takes place and the state of bulb will then toggled. Additionally, to the evident visual feedback on the bulb, looking at HomeAssistant interface it is possible to see the event occurrence.

4.2.2 Server vs On-Device Machine Learning Performance Tests. Even though traditionally, machine learning and neural networks are concepts associated with increased computation power and robust hardware, the scope of this work addresses the on-device artificial intelligence arising ubiquitousness and its major potential. Therefore, an interesting result to analyze is the accuracy obtained with lighter models and the level of latency or performance decreasing when running on less powerful devices.

This way, a metric that can give indicators of both benefits and drawbacks of this approach is the time spent during the execution of the model trained to produce an output and effectively label the input image. A possible way of analyzing this is by running the inference model at the machine where it was trained with a python script, label_image.py and, after that, running the same model already integrated in the mobile application, measuring the times before and after the run.

This analysis was performed using three different images and ran three times in each of them to guarantee a minimum coherence in the results. In the case of the mobile device, since the input comes as a video stream, the three images were represented by three different scenarios with the three different bulbs. The results are presented bellow in Table 1.

Table 1: Comparison between Manual Inference versus On-Device Run

	Server Run			On-Device		
Time (ms)	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Image 1	12.8	12.7	12.9	16.0	15.0	16.0
Image 2	13.5	13.7	13.6	20.0	19.0	17.0
Image 3	12.4	12.4	12.5	30.0	30.0	30.0

4.2.2 Accuracy Tests. The implemented prototype is based on a neural network retrained to identify representative smart household devices, particularly bulbs, air conditioners and electrical window shutters. Therefore, the scenarios used to validate the accuracy of the solution matched these items.

The first test exemplifies a use case scenario where the user points the device at a light bulb. In this case, when the evaluation output is higher than the 95% accuracy threshold, the switch option appears on the screen and the user can interact with the device. To obtain maximum results and since the image dataset used for testing was simple light bulbs, the 3 tests performed were on simple lightbulbs connected to power (Figure 11).

In a similar way, test cases were made for window shutters and air conditioners and the accuracy results obtained are presented in Table 2.

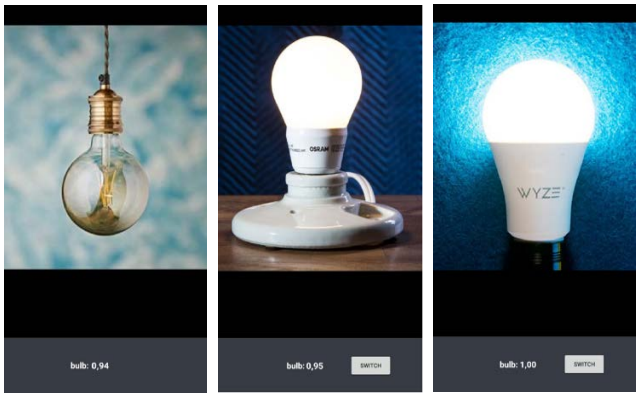

Figure 11: Bulb Tested Scenarios

Analyzing the results obtained, concerning the comparison made between manually running the inference model on the training environment, at the server side, and the inference happening on the mobile device, we can see that the results were not that distant and, therefore, the system can produce fast results in real-time without compromising the performance obtained. In terms of accuracy obtained, every test provided result higher than 90%, which is positive and allowed to control the devices in almost every situation with confidence.

Table 2: Tested Scenarios Accuracy

	Test 1	Test 2	Test 3
Bulb	94%	95%	100%
Air Conditioner	95%	90%	93%
Window Shutter	99%	91%	98%

5 CONCLUSIONS

As an overall solution, the system developed worked as a proof of concept of the integration of mobile devices in home automation with use of machine learning for object recognition. Even though machine learning and deep learning-based systems often need massive hardware to be trained and developed, the work developed for this paper proves that lightweight alternatives can be found and adapted to meet the desired goals.

Looking at the obtained results, an important conclusion to bear in mind is that on-device machine learning proved to be surprisingly accurate and able of handling the tasks defined, confirming in this way the already arising paradigm of artificial intelligence increasingly moving to edge devices, without compromising functionality. The usage of this technology doesn't impact the performance of the home system, because the processing is mainly made on the mobile device, and since no new hardware is required, there isn't also a cost increase.

The developed solution also proves that an inference machine learning model can be used in real-time, integrated inside a mobile device application, providing instantaneous visual feedback. This leads to also real-time action triggering in the automation platform making use of the integration and consequently built a seamless and uninterrupted information workflow. This is key for achieving an appropriate interactive experience for end users.

6 FUTURE WORK

For future work, many sensors in the mobile device, which were not used in the course of this work, can be further explored which enables the possibility to control a range of smart devices, from air humidifiers to door locks, in new and innovative ways.

Concerning the computer vision topic, the evolution from image classification to object detection would also provide the ability to track multiple objects at the same time and allow better precision and control of the devices. Even though implying major developments in the application, this improvement would unlock more possibilities of user interaction and functionalities to the application.

Additionally, a contribution than can be remarking, in the potentiality of the concept introduced here, is the re-training of the inference model based on user feedback. Even though a solution can be prepared for a general use case scenario, each case is unique, and each user will have different needs. If the mobile application provides a way of collecting user feedback according to his respective scenario and reality, this input could be used to re-train the model with improved accuracy and consequently to obtain flawless and consistent results.

ACKNOWLEDGMENTS

This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020

REFERENCES

[1] Andrew Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [Online] Available: https://arxiv.org/abs/1704.04861v1

[2] Dat Tran. 2017. How to train your own Object Detector with TensorFlow's Object Detector API. [Online] Available: https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9, (visited 22/12/2018).

[3] Davide Mulfari, Antonino Minnolo and Antonio Puliafito. 2017. Building TensorFlow Applications in Smart City Scenarios. In *Proceedings of IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, pp. 1-5*. https://doi.org/10.1109/SMARTCOMP.2017.7946991

[4] Eclipse Foundation. 2019. Eclipse Mosquitto, an open source MQTT broker. [Online] Available: https://mosquitto.org, (visited 24/10/2019)

[5] Eclipse Foundation. 2019. Eclipse Paho Android Service. [Online] Available: https://github.com/eclipse/paho.mqtt.android, (visited 10/04/2019)

[6] Home Assistant. 2019. Home Assistant. [Online] Available: https://www.home-assistant.io, (visited 24/10/2019)

[7] Jeff Tang. 2018. Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence Apps Using TensorFlow Mobile and Lite for iOS, Android, and Raspberry Pi. *Packt Publishing*. ISBN: 1788834542, 9781788834544

[8] Souhail Guennouni, Ali Ahaitouf and Anass Mansouri. 2014. Multiple object detection using OpenCV on an embedded platform. In *Proceedings of IEEE*

International Colloquium in Information Science and Technology (CIST). Tetouan, pp. 374-377. https://doi.org/10.1109/CIST.2014.7016649

[9] StatCounter. 2019. Mobile & Tablet Android Version Market Share Worldwide. [Online] Available: https://gs.statcounter.com

[10] Trio Adiono, Suksmandhira Harimurti, Billy Austen Manangkalangi and Waskita Adijarto. 2018. Design of smart home mobile application with high security and automatic features. In *Proceedings of International Conference on Intelligent Green Building and Smart Grid (IGBSG). Yi-Lan, pp. 1-4.* https://doi.org/10.1109/IGBSG.2018.8393574

[11] Vignesh Govindraj, Mithileysh Sathiyanarayanan and Babangida Abubakar. 2017. Customary homes to smart homes using Internet of Things (IoT) and mobile application. In *Proceedings of International Conference On Smart Technologies For Smart Nation (SmartTechCon). Bangalore, pp. 1059-1063.* https://doi.org/10.1109/SmartTechCon.2017.8358532