OKIoT Open Knowledge IoT Project: Smart Home Case Studies of Short-term Course and Software Residency Capstone Project

Victor Takashi Hayashi, Vinicius Garcia, Renato Manzan de Andrade and Reginaldo Arakaki
Polytechnic School, University of São Paulo, Luciano Gualberto Venue, São Paulo, Brazil

Keywords: Internet of Things, Arduino, Education, Smart Speaker, Smart Home.

Abstract: The emergence of smart environments built on top of Internet of Things (IoT) solutions demand new skills and knowledge for developers. Dealing with inherent complexity of IoT architecture, constrained device limitations, communication faults and vendor lock-in can be drawbacks for successful deploy of IoT projects. With the objective of sharing knowledge between IoT developers, OKIoT project focuses on project-oriented education in Short-term and Long-term modes based on Software Engineering methodology. First qualitative results on a 6-week course on MBA subject offering and a case study of an open architecture for smart speaker executed with 1-year mentoring of a capstone project are summarized. Future steps based on case studies insights are presented.

1 INTRODUCTION

Cisco (2016) expects more than 25 billion devices to be connected to the internet by 2020, integrated in solutions like smart home, connected car and smart cities. Even though the opportunities are large, the question of whether there are enough expert developers to build these solutions still remains open.

Building reliable systems is one of incoming challenges with the Internet of Things (IoT) trend. Applicability of Software Engineering methods and knowledge such as quality analysis should be addressed. For example, trade-offs between simple devices and robust computational platforms are common studies found on literature (Baresi, Mottola, & Dustdar, 2015; Gubbi et al, 2013; Lu et al, 2015; Matharu, Upadhyay, & Chaudhary, 2014).

From a practical standpoint, developers could face challenges such as:

- Complexity: how to integrate heterogeneous interfaces, services and devices?
- How can one deal with faults like internet unavailability to maintain a desired level of service?
- How to develop considering constrained devices?
- When considering software developers, is it possible to overcome the hardware initial learning curve?

Open Knowledge Internet of Things Project (OKIoT) aims to be an accelerator for smart environment innovation. It is based on non-functional requirements of availability, usability and rastreability on top of a fault tolerant IoT architecture.

OKIoT is based on three pillars:

- **Conceptual Modelling:** developers can understand Software Engineering fundamentals, technologies used in existing solutions in order to understand constituent parts (hardware, software, communication) and its integration;
- **Hardware Accelerator:** encapsulates power control over smart plug appliance, rastreability components and serial communication used to integrate this solution with Arduino projects, as well as fault-tolerant architecture, provided with its documentation;
- **Software Solutions:** projects should be hosted on platforms like GitHub, together with their components, libraries, programs, apps, maintenance, build and deploy procedures, so future developers could understand, use, collaborate and advance existing solutions.

These IoT hardware, software and solutions combined build a growing knowledge project. This material may be widely used for teaching purposes in specialist training courses, higher education courses, technical training courses, and other types of special-purpose education related to digital transformation. Students, teachers, professionals can help the project

Hayashi, V., Garcia, V., Manzan de Andrade, R. and Arakaki, R.
OKIoT Open Knowledge IoT Project: Smart Home Case Studies of Short-term Course and Software Residency Capstone Project.
DOI: 10.5220/0009366002350242
In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security (IoTBDS 2020)*, pages 235-242
ISBN: 978-989-758-426-8

with positive feedback, improvement points, benchmark with market solutions and system specifications.

OKIoT was created based on:

Elderaid Project architecture had redundancies for critical communication paths. The system was intended for elderly care, but had few developments on fault tolerance (Hayashi, 2016);

Hedwig Capstone Project main deliverable was a proof of concept of a fault-tolerant smart home architecture, with three levels of functionality: direct to module, local and internet (Hayashi, 2017);

By 2018, Hedwig was put into practice by continuous development and deployment on a real smart home, which enabled it to start domus open data smart home repository (Hayashi, 2019);

A study on open architecture for smart speakers, further extended Hedwig Project, integrating it with conversational interface (Garcia et al, 2019). A short course was also supported (Hayashi, 2019b).

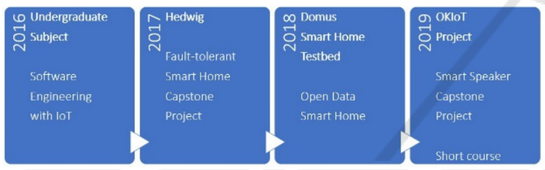

Figure 1: IoT Projects Timeline: past developments were the basis for OKIoT.

2 RELATED WORK

Smart speaker market leaders like Google and Amazon (Statista, 2017) make voice interfaces available on more than 200 million devices (Canalys, 2019). Though natural language commands become accessible through these commercial products, the imposition of a closed architecture limits user interaction. Compulsory use of pre-set wake word (e.g. “Alexa”) raises problems for users whose names resemble the wake word (e.g. “Alex”). On developer´s side, lock-in is clear: backend services and natural language understanding modules are attached to vendor platform. Portability becomes a challenge, as the assistants would be duplicated on different platforms and could have different responses for similar questions. with only cloud-based solutions available, other common concern regards privacy: an American Alexa user reported that the smart speaker recorded a conversation without consent and sent it to another user, present on contact´s list (The Guardian, 2018).

Some projects were created to address mentioned existing smart speaker limitations: SpeechRecognition (Zhang, 2019) library makes it easy to use different speech recognition services. It is possible to choose, by command-line interface, which service the developer would like to use for speech recognition. Snowboy (Chen, 2019) is an open alternative for wake word creation, and “An Open Voice Command Interface Kit” (Ansari, Sathyamurthy, & Balasubramanyam, 2016) developed a low cost hardware and used PocketSphinx for speech recognition process. Alias Project (Karmann & Knudsen, 2019) created a hardware that blocks what existing smart speakers listen to with a white noise. When a personalized wake-word is called, the additional hardware then stops the white noise, triggers the interaction with the commercial smart speaker by playing vendor´s wake word and letting the user complete the query. How to integrate these projects in order to create a smart speaker with an open architecture (and additionally perform a performance evaluation analysis) still remains an open question.

On a broad IoT perspective, there are several commercial platforms (Amazon, 2019; IBM, 2019; Oracle, 2019) with support to cloud computing, machine learning and big data services, easy integration with voice interfaces like chatbots on smartphones and smart speakers like Alexa. They share a common drawback: vendor lock-in. Developers (and systems, by consequence) tend to have low flexibility on the system architecture.

Some open IoT platforms have also been launched, such as DeviceHive, Mainflux, ThingsBoard, WSO2, Sitewhere, ThingSpeak, Zetta, Helix Sandbox and KnoT (DeviceHive, 2018; Helix Sandbox, 2018; KnoT, 2019; Mainflux, 2018; Thingsboard, 2018; ThingSpeak, 2010; Sitewhere, 2018; WSO2, 2017; Zetta, 2015), just to name a few. They provide integration based on Representational State Transfer API (REST API), websockets and Message Queuing Telemetry Transport (MQTT), also support Big Data and Machine Learning services; different hardware platforms (Arduino, Raspberry Pi, Intel Edison, ESP8266); programming languages like C/C++, JavaScript and python; security mechanisms and personalized dashboards. They focus entirely on development skills for system development. Besides the open aspect of these platforms, developers face the great challenge of discontinued support for some low-engagement platforms, as there is no unified platform to date.

Some institutions create their own tools for educational purposes, like IoTaaP and Copernicus (MVT Solutions, 2019; Szydlo & Brzoz-Woch, 2019). As there is limited communication across institutions, expertise become constrained as well.

The multiplicity of languages, communication protocols, hardware boards and interfaces make it a challenge to build a unified platform for IoT. Engaging and retaining a broad developer community is no easy task. As the applications have different system requirements, it is difficult to wonder about a platform that could be flexible enough to support all applications.

Our hypothesis is built on the premise that knowledge is an intangible asset easy to share, agnostic to technology choice and not subject to vendor lock-in. Our objective is to build an open knowledge sharing community that engages developers and enthusiasts, skilful on different technologies, taking smart home as the starting field. With the focus on knowledge, the key differential is the integration of Software Engineering methodology with IoT education.

An open architecture for smart speakers that allows developers to choose between different services on each step of the voice interaction process (speech-to-text, natural language understanding and text-to-speech), with the objective to optimize its end-to-end performance.

3 METHOD

We validate our approach by taking qualitative feedback on proposed project-oriented methods: short-term course and capstone project throughout one year.

On September-October 2019, a short-term course on an MBA course offering was supported. An accelerator kit and its manual were given to each of the 5 groups.

One training session of 4 hours was implemented on each of the 6 weeks. As the opening class was based only on introduction of the accelerator kit, and the last had the final presentations, 4 hands on experiments were performed on the remaining 4 weeks.

Project proposals had a common theme: smart home. Groups of 2-3 people each had to choose an environment (e.g. kitchen, living room, bathroom), a value proposition and a persona. Initial brainstorming was based on smart home major areas of interest found on literature (Batalla, Vasilakos, & Gajewski, 2017).

A key aspect of project proposals was the user-centric approach. IoT was presented as a supporting technology that must be centred on user interactions that create value. As smart homes are shared environments with different user profiles and goals, designing IoT systems based on value proposition was a must.

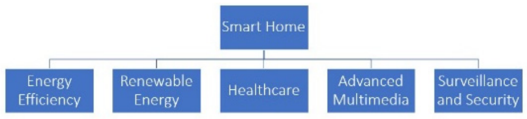

Figure 2: Smart Home major interest areas.

On first class, Hedwig Project IoT modules were presented in order to inspire the students (see Figure 3). The presentation was focused on accessibility and fault tolerance over the different interfaces of a smart switch for smart lightning system (namely, presented interfaces were voice, switches, sensors and radio frequency controls).

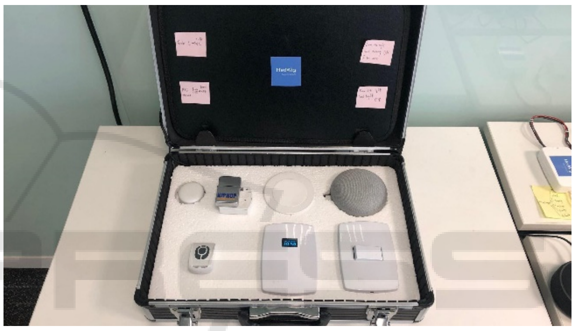

Figure 3: Hedwig Project demonstration case.

The accelerator kit (as depicted in Figure 4) was handcrafted on this initial offering. It was developed based on past Hedwig capstone project infrastructure, and had the objective of easy transformation of Arduino projects into IoT projects, with a black box module that is integrated with Arduino modules through a serial protocol, has a smart plug for power appliance control and has a fault-tolerant architecture (e.g. tolerates internet unavailability). For cloud, Blynk Cloud IoT Platform was used (Blynk, 2019).

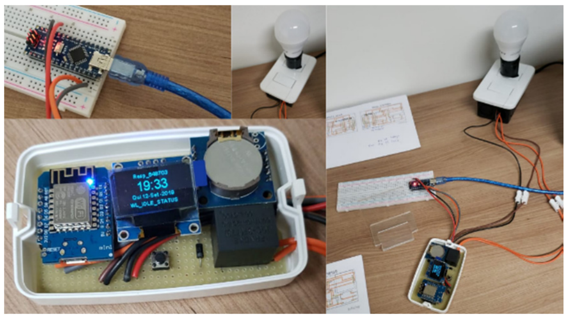

Figure 4: Hedwig IoT Kit.

The kit is composed by:

- Relay: physical actuator;
- Switch: local control of the device;
- OLED Display: Used to view module's address after connecting to Wi-Fi network;
- RTC: date/time synchronization, if offline operation is required;
- Wemos D1 R2: ESP8266 based development board with built-in Wi-Fi (hotspot and client connection), responsible for direct communication, local network and internet connectivity. Performs serial communication with Arduino Nano, and it is used as a black box by developers;
- Arduino Nano with protoboard: Development board to be programmed through Arduino IDE, with USB connection to a computer. Responsible for domain-specific rules, sensors and actuators.

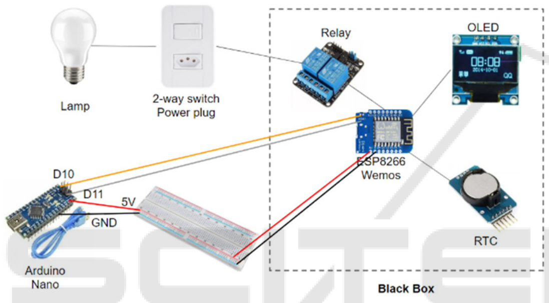

Figure 5: Hedwig IoT Kit Components.

The accelerator has three-level fault tolerant architecture (module, local area network, and internet). Its communication channel redundancies allow for greater usability and accessibility (interfaces for different personas, at different times), as well as fault tolerance itself (local interfaces).

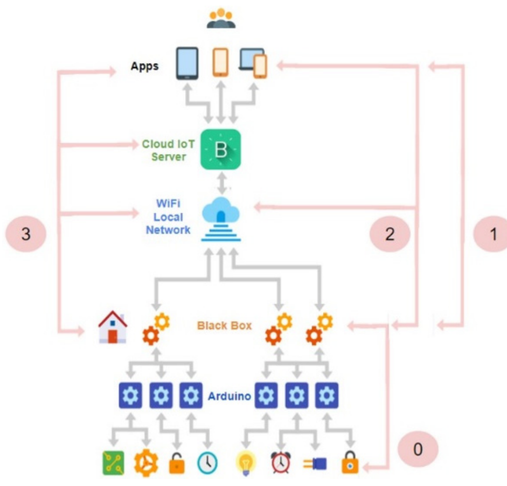

Figure 6: Hedwig IoT Kit Architecture.

Each hands-on procedure performed on class was matched to a communication level:

0. User controls smart plug module on offline mode through physical switch, which is interpreted by black box;

1. Direct communication between smartphone and black box allows local monitoring and control on a web interface, with support to HTTP requests to monitorate and control the smart plug. With this setup, black box operates as a Wi-Fi hotspot and the smartphone connects to it, with an integrated web server deployed entirely on the IoT module itself;

2. Through the interface of procedure 1, the black box is connected to a local Wi-Fi network. The same requests can now be made to multiple modules, and the mobile phone becomes an integrated interface for all modules in the same Wi-Fi network;

3. After account and project creation on Blynk platform, the match between the black box power module will occur through project auth token and V20 virtual pin mapping. With this setup, the module will be connected to the internet via local Wi-Fi network, and therefore connected to the Blynk server. Through mobile network, your mobile phone can access Blynk's server and control / monitor the black box. Blynk also offers a REST API for integrating other applications, like Google Assistant if you use it with IFTTT (IFTTT, 2019).

All procedures are described step-by-step on specific document and videos (Hayashi, 2019b).

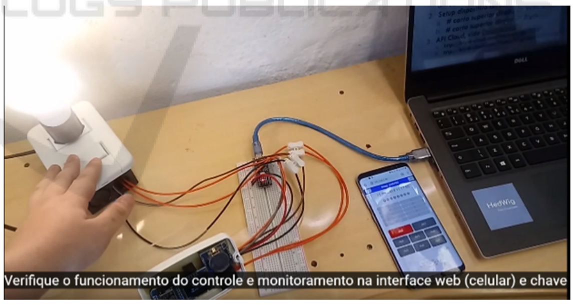
Verifique o funcionamento do controle e monitoramento na interface web (celular) e chave

Figure 7: Video tutorial example with subtitles. Local web interface is accessed on smartphone.

For final evaluation, a qualitative questionnaire was given to each student at the end of the 6-week subject. This quick survey had 6 questions:

1. What is your overall course evaluation?
2. In your opinion, what were the main concepts learned?
3. Based on your perspective, how important are the architectural concepts explored in the course?
4. What did you think about the pedagogical approach of the course (learning software engineering concepts through IoT)?

5. Suggestions for course improvements
6. Free comments:

A Software Residency program was deployed over February-December 2019 for a long-term approach for OKIoT. It was composed of the following steps:

1. User-centric Value Proposition: it was proposed a smart home for elderly care, but with a focus on user´s independence and ease of use;
2. Software Engineering Modelling Method: user journeys, prioritization of critical aspects with system modelling and specification by description of functional and non-functional requirements, along with architecture and sequence diagrams (for static and dynamic modelling);
3. Iteration: with chosen qualitative and quantitative programmed tests, iterative development was performed, with some relevant changes on initial scope value proposition and system specification;
4. Results and Analysis: documentation of results and analysis. For OKIoT, this is the most important step, as it is how the knowledge will be passed on.

A main contribution of the software residency for OKIoT Project was the study of an open architecture for a smart speaker, whose development and evaluation method are described below.

An API was created in order to provide flexibility for the developer to choose between 2 services on each of the basic services of a smart speaker. With the setup shown in Figure 8, the objective was to propose a method to evaluate possible combinations of these basic services (Amazon, 2019b; Dialogflow, 2019; Google, 2019a; Google, 2019b; IBM, 2019b; Rasa, 2019).

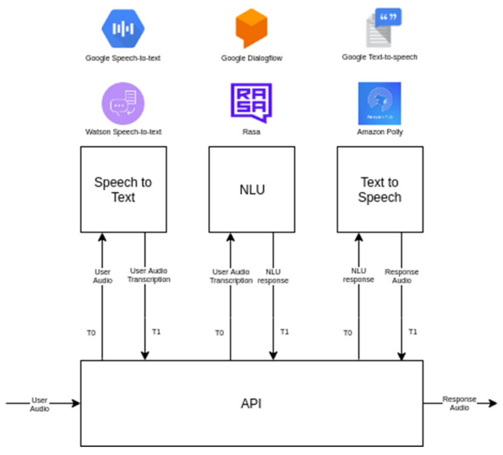

Figure 8: Smart Speaker Basic Services API.

The API was run on a Dell Inspiron 15 7000 series notebook, 16GB RAM, 7th generation Core i7 processor running Windows 10 operating system and connecting via ethernet cable to a 60mb internet. Connection quality was monitored by using SpeedTest (Speedtest, 2019).

Quantitative results for each scenario were obtained by using the average of 50 tests. Four different audios were analysed, for a total of 200 tests. The audios were chosen in order to considerate the clarity of speech and the noise of the environment:

- On first audio, the speech is loud and clear, without external interference and with very little ambient noise;
- The second recording contains a song that is played along with the speech;
- An old recording is accompanied by a certain echo and an ambient noise;
- On fourth recording, it is possible to understand the speech, but some effort is necessary due to the low volume of the speech and the environment noise.

Speech-to-text assertiveness was evaluated qualitatively in relation to the expected transcription performed by a human.

Time markers were inserted in the API code for the quantitative evaluation. It was possible to measure the interval between the activation of a service and its response. Results were recorded in a log file in CSV format. These times are represented by “t0” and “t1” present in the interaction arrows between the API and the services in Figure 8.

4 RESULTS

Short term course projects are described in brief:

1. Bathroom: people with visual impairments have difficulties finding the towel after taking a shower. Sound-based offline feedback was designed;
2. Kitchen: gas leakage, temperature, humidity and presence sensors were used together in order to detect safety threatening situations;
3. Home Office: redundancy of temperature monitoring with external data source for thermal comfort;
4. Room: automatic light bulb for children going to the bathroom at night, parent monitoring and control with smartphone;
5. Living room: accessibility for the physically handicapped through curtain automation, through voice and smartphone app.

Feedbacks obtained by quick surveys run at the end of the short-term course were:

- More classes must be integrated, as the content and hands on procedures were passed very quickly;
- Projects could be more related to everyday life;
- Smart home area generated interest of some students to continue with some home automation projects;
- Hands on approach was approved by most students, but some of them had some difficulties (e.g. students without technical background);
- Explain some practical cases concerning the Software Engineering methodology in a dynamic and critical way.

Smart speaker results supported initial proposition that combining different vendor's basic services could be the best option for an efficient smart speaker, with minimized response time.

Google's speech-to-text and IBM Watson qualitative results are given below:

- First Audio (loud and clear speech): no significant difference;
- Second Audio (speech with song): IBM Watson delivered an incomplete transcription. Google performed a complete transcription, with some mistakes
- Third Audio (echo and ambient noise): Google could understand some words, while IBM Watson could not deliver any word;
- Fourth Audio (low volume and ambient noise): similar result to third audio.

As for the quantitative evaluation, average response time difference was ~7,021ms, which represents an approximately 68% greater response from Google speech-to-text than IBM Watson. Higher response times were observed when the audio had a lot of noise (third and fourth audios). Regarding Natural Language Understanding (NLU) modules, Dialog Flow's performance was over 7000% higher than rasa's performance, with a difference of approximately 1103ms between them. It was expected, as Rasa is a local service.

For instance, a developer building a smart speaker could choose to combine three basic services from different vendors: Google Speech-to-text, RASA and Amazon Polly. Proposed evaluation method provides some possible reasons: RASA and Amazon Polly showed better response times and no clear disadvantage.

Even though the method provides a base to evaluate smart speaker basic services, the results are not absolute. When analysing speech-to-text services, there was a better performance of Google's service in the qualitative approach and better performance of IBM's service in the quantitative approach. The developer could choose the best speech-to-text that better fits its application environment: he could choose IBM because of lower response time if the ambient noise is not relevant; or choose Google because of its higher accuracy on environments with high noise.

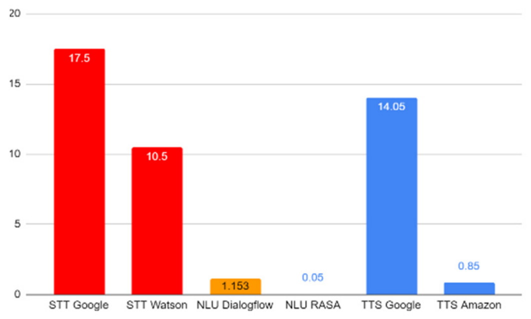

Figure 9: Response times (seconds) for smart speaker basic services.

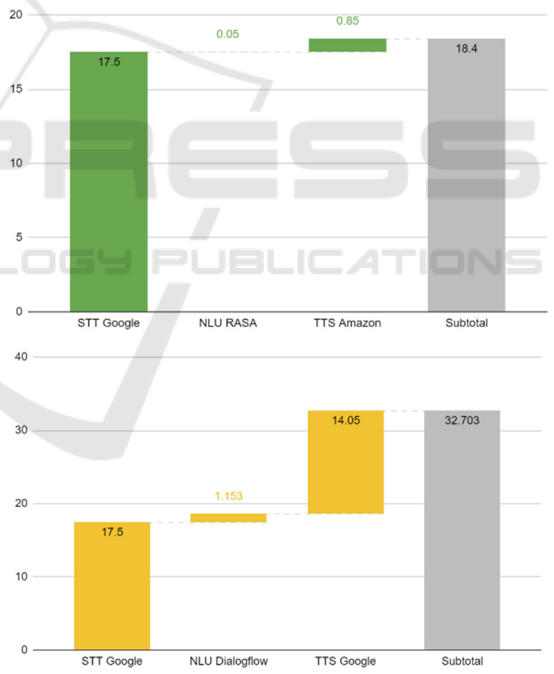

Figure 10: Response time (seconds) comparison for open architecture (up) and closed architecture (bottom).

5 CONCLUSIONS

OKIoT Project initial results were presented. Proposed IoT education method, project-oriented with Software Engineering methodology was evaluated on short term course and long-term capstone project mentoring. A study on the open architecture for smart speakers and associated evaluation method were also presented as one initial contribution to OKIoT.

As future steps, OKIoT roadmap include:

1. Support to developers: creation of forum and blog, aimed to complement existing GitHub and videos;
2. Online course: future creation of specific courses on online platforms after some presential course iterations;
3. Undergraduate course: future support for undergraduate course, project-oriented, with more classes than the MBA;
4. Hackathon: future sessions of hackathons in order to test IoT education with Software Engineering concepts;

Additionally, Problem Based Learning (PBL) as an alternative to black-box approach is being considered for future OKIoT deployments. Regarding smart speaker study, local alternatives to basic services could be a way to preserve user privacy and reduce response time.

REFERENCES

Amazon (2019). *AWS IoT*. Retrieved from https://aws.amazon.com/iot/

Amazon Polly (2019) *Amazon Polly - Text to Speech in 47 Voices and 24 Languages*. Retrieved from https://aws.amazon.com/polly/

Ansari, J. A., Sathyamurthy, A., Balasubramanyam, R. (2016). An Open Voice Command Interface Kit. *IEEE Transactions on Human-Machine Systems 46 (3*), 467-473.

Baresi, L., Mottola, L., & Dustdar, S. (2015). Building Software for the Internet of Things. *IEEE Internet Computing*, 6-8.

Batalla, J. M., Vasilakos, A., & Gajewski, M. (2017). Secure Smart Homes: Opportunities and Challenges. *ACM Comput. Surv. 50, 5*, Article 75. https://doi.org/10.1145/3122816

Blynk (2019). *Blynk IoT Platform*. Retrieved from https://blynk.io/

Canalys. (2019, April 15). *Canalys: Global smart speaker installed base to top 200 million by end of 2019*. Retrieved from https://www.canalys.com/newsroom/canalys-global-smart-speaker-installed-base-to-top-200-million-by-end-of-2019

Chen, G., et al. (2019, November 5). *Snowboy Hotword Detection*. Retrieved from https://snowboy.kitt.ai/

Cisco. (2016). *Global - 2020 Forecast Highlights - Cisco*. Retrieved from https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf

DeviceHive (2018). *DeviceHive*. Retrieved from https://devicehive.com/

Dialogflow (2019) *Dialogflow*. Retrieved from https://dialogflow.com/

Garcia, V., et al. (2019, December 8). *Smart Home for Elderly*. Retrieved from https://github.com/ViniciusGarciaSilva/ speaker-API

Google (2019). *Google Speech-to-text*. Retrieved from https://cloud.google.com/speech-to-text/

Google (2019) *Google Text-to-speech*. Retrieved from https://cloud.google.com/text-to-speech/

Gubbi, J., et al. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29.7*, 1645-1660.

Hayashi, V. T. (2019, December 8). *OKIoT Open Knowledge Internet of Things*. Retrieved from https://github.com/vthayashi/OKIoT

Hayashi, V. T., et al. (2016, September 28). *Elderaid*. Retrieved from https://github.com/usp-labsoft/Home-Elderaid/

Hayashi, V. T., et al. (2017, December 21). *Hedwig Connected Home*. Retrieved from https://github.com/hedwig-project

Hayashi, V. T., et al. (2019, June 23). *Domus Smart Home Testbed*. Retrieved from https://github.com/vthayashi/domus

Helix Sandbox (2018). *Helix Sandbox*. Retrieved from https://github.com/helix-iot/helix-sandbox

IBM (2019). *Watson IoT*. Retrieved from https://www.ibm.com/internet-of-things

IBM (2019). *IBM Watson Speech-to-text*. Retrieved from https://www.ibm.com/watson/services/speech-to-text/

IFTTT (2019). *IFTTT*. Retrieved from https://ifttt.com/

Karmann, B., Knudsen, T. (2019, January 14). *Project Alias*. Retrieved from https://bjoernkarmann.dk/project_alias

KnoT (2019). *KNoT Network of Things*. Retrieved from https://knot.cesar.org.br/

Lu, J., et al. (2015) Roundtable: The Future of Software Engineering for Internet Computing. *Software, IEEE 32.1*, 91-97.

Mainflux (2011). *Mainflux Open Source IoT Platform*. Retrieved from https://www.mainflux.com/

Matharu, G. S., Upadhyay, P., & Chaudhary, L. (2014). The Internet of Things: Challenges & Security Issues. *Emerging Technologies (ICET), 2014 International Conference*.

MVT Solutions (2019). *IoTaap Connectivity platform*. Retrieved from https://iotaap.mvt-solutions.com/

Oracle (2019). *Internet of Things (IoT) | Oracle*. Retrieved from https://www.oracle.com/internet-of-things/

Rasa (2019). *RASA: Open source conversational AI*. Retrieved from https://rasa.com/

Sitewhere (2018). *SiteWhere Open Source Internet of Things Platform*. Retrieved from https://sitewhere.io/

Speedtest (2019) *Speedtest by Ookla*. Retrieved from https://www.speedtest.net/

Statista. (2017) *Global intelligent assistant market share 2017-2020 | Statista*. Retrieved from https://www.statista.com/statistics/789633/worldwide-digital-assistant-market-share/

Szydło, T., Brzoza-Woch, R. (2019) *AGH Copernicus*. Retrieved from http://galaxy.agh.edu.pl/~tszydlo/copernicus/

The Guardian (2018, May 24). *Amazon's Alexa recorded private conversation and sent it to random contact*. Retrieved from https://www.theguardian.com/technology/2018/may/24/amazon-alexa-recorded-conversation

Thingsboard (2018). *Thingsboard*. Retrieved from https://thingsboard.io/

ThingSpeak (2010). *ThingSpeak*. Retrieved from https://thingspeak.com/

WSO2 (2017). *IoT - WSO2*. Retrieved from https://wso2.com/iot

Zetta (2015). *Zetta*. Retrieved from http://www.zettajs.org/

Zhang, A. (2019, July 2). *Uberi/speech_recognition - GitHub*. Retrieved from https://github.com/Uberi/speech_recognition