Pain Points for Novice Programmers of Ambient Intelligence Systems: An Exploratory Study

Pain Points for Novice Programmers of Ambient Intelligence Systems: an Exploratory Study

Fulvio Corno
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
Email: fulvio.corno@polito.it

Luigi De Russis
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
Email: luigi.derussis@polito.it

Juan Pablo Sáenz
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
Email: juan.saenz@polito.it

***Abstract*—This paper presents an exploratory study aimed at identifying the pain points that novice programmers experience, from the software engineering perspective, when developing and deploying smart and distributed systems, that may be classified as Ambient Intelligence (AmI) systems. The exploratory study was conducted among undergraduate students, that worked in groups for developing AmI projects during a university course. Based on their own experiences, individually and as a group, the pain points were identified and prioritized over a common architecture and a set of software development activities. The quantification of the pain points was based on the difficulty level that the students perceived on the development activities and the time they spent completing them. Results represent a starting point for the design of tools and methodologies targeted at overcoming the complexity that novice programmers face when developing AmI systems.**

I. Introduction

The Internet of Things (IoT) is positioning itself as one of the most influential paradigms in the Information and Communications Technology (ICT) landscape. The idea of embedding computing and communication capabilities into objects of common use [1] leads to the development of all kinds of solutions that might be used in contexts like Ambient Intelligence (AmI), which focuses on the study of how IoT systems integrate and interact with users. The development of IoT systems, and specifically AmI systems, has aroused the interest of programmers, and has raised the need to introduce undergraduate IT students to their design and implementation. Nevertheless, several challenges emerge for novice programmers when attempting to develop AmI systems.

Implementation challenges are numerous, and mainly stem from the fact that AmI systems integrate several subsystems that are heterogeneous. Each subsystem has its own set of languages, protocols, devices, and communication mechanisms. Nevertheless, these subsystems are required to interact with each other in such a way that the whole system is able to cooperatively accomplish a certain goal.

The technical development of AmI solutions requires the understanding of several ICT areas such as database design, embedded systems development, mobile applications development, push notifications, and service oriented architectures, to name a few. Since undergraduates are not expected to have deep knowledge nor experience in these areas, the Ambient Intelligence course at the Politecnico di Torino uses an active learning approach to engage them with several concepts that are applied in the implementation of a course project.

Based on the students' experiences in the course, this paper aims at identifying which are the most painful points that novice programmers experience when developing AmI systems. For this purpose, first, we identified the common architectural decisions across the design and implementation of several course projects. Then, we represented these decisions into an overall architecture in which a set of subsystems were defined, along with their associated software development tasks. Lastly, upon the understanding of the AmI projects in terms of their architecture and software development tasks, an exploratory study involving students from the last version of the course was designed. This study was divided in three parts: in the first part, students were asked to grade the development tasks in which they participated according to their difficulty level and the time spent completing them. In the second part, students had to choose and prioritize the most complex tasks of each subsystem by ranking them. Finally, as a group, they discussed and reached a consensus about the overall most painful points.

Besides identifying the most painful issues experienced by the AmI course students, the outcomes of this study are intended to act as an objective basis for researching and proposing new tools and languages to support novice AmI developers, from the software engineering perspective.

II. Related Works

The literature presents approaches directed at facilitating novice programmers to learn, while developing IoT systems in realistic scenarios.

He *et al.* [2] describe the experience of using a single-board embedded computer (a BeagleBone Black) in teaching embedded systems courses. Particularly, with the purpose of improving students' awareness of IoT single-board hardware and software features, while teaching them to develop basic IoT applications. In the opinion of the authors the set of pre-installed software development tools (Node.js, Bonescript, and Cloud9) would make the implementation of simple IoT applications easier for novices. Kortuem *et al.* [3] present their experience on imparting a distance course called *My Digital Life* (Open University) using a set of tools, expressly designed

for the course. They developed these tools to enable students learning and experimenting with sensing, actuation, and networking. The set consisted of: an embedded network sensor device; a visual programming language and environment (built upon the Scratch); and a cloud platform that connects all the networked devices together. According to the authors, this approach was effective in helping the students quickly develop an understanding of the principles of programming simple IoT applications. Dobrilovic *et al.* [4] describe the design of a low-cost, easy to deploy platform to be used in university courses for teaching IoT. This platform relies on Arduino Uno on the hardware side, Python as the programming language, and ThingSpeak as the platform to store and visualize in a web layout, all the data generated by the sensors. Authors also identified a set of low-cost hardware components to be used in the platform. However, the platform did not include Wi-Fi nor Bluetooth modules, and the integration with SQL or NoSQL databases was not achieved. Despite the good comments, students' feedback was not collected nor formally analyzed. Hahm *et al.* [5] present an open source platform to facilitate the development of IoT applications. It is targeted to bachelor and master students that have only basic knowledge in systems programming and little time to get deeply involved in learning specific programming environments. It enables the developers to code in standard programming languages (C and C++) using standard debugging tools, supports basic networking protocols, and allows the setup of arbitrary virtual networks so multiple native instances can run on a single computer.

The presented works aim at identifying a safe and comprehensive development environment for enabling student development. In this paper, however, we examine the difficulties encountered when novice developershave to develop a complete system with the variety of technologies and languages that are nowadays required in the IoT.

III. The Ambient Intelligence Course

Ambient Intelligence is defined as "a digital environment that proactively, but sensibly, supports people in their daily lives" [6]. As a field, it is at the intersection of artificial intelligence, human-computer interaction, ubiquitous computing, and sensor networks. Four cyclical steps comprise the behavior of an AmI system: *sensing* from the environment, from the users, and/or from the Internet; *reasoning* about sensed data; *acting* upon the environment and/or towards the users; and *interacting* with the users, to keep them always in the loop.

To provide its students with the systems-level skills needed to understand and develop complete IoT systems, in 2014 Politecnico di Torino initiated a course in "Ambient Intelligence". In this project-based course, a teamwork and design-driven paradigm is applied to teaching AmI system design [7]; core student skills acquired in previous courses are exploited in a multidisciplinary project work.

The main topic of the course is the design and the realization of AmI systems. This entails a strong focus on the application and on user needs. From the beginning of the course, students form three- to four-people teams, and are guided to define the requirements for an AmI system, and then to design and implement it. This system and the "deliverables" produced throughout the semester are the focus of the course exam, which also includes a presentation of the team projects and an oral discussion on. Every year, a theme is chosen for the projects; in 2016 it was the "Health and Well-being" (in a broad sense). Target environments could be homes, offices, vehicles, open spaces, gyms, etc.

From the university point of view, a course is over once the exam is passed. However, students' hard work and the quality of many of their projects, allow course results to be transformed into a public technology event. All students who passed the exam were invited (on a voluntary basis) to present their project to the public, during a dedicated open event, the "AmI Student Showcase". During the Showcase, projects are voted by the public, and the three most voted project are awarded some high-tech gadgets. In 2016, the three winning projects were: "Safety Mama" (a system for helping pregnant women with their anxiety), "Emergency Quest" (for elderly persons with mild cognitive impairments), and "Study Station" (a wearable device for maintaining the right posture while studying).

IV. Exploratory Study

This section describes the exploratory study that we carried on to gain knowledge about the difficulties and issues that AmI students encountered during the development of their projects. We first present a generalized structure of AmI projects, so that we may map and compare specific projects. Starting from the general project architecture, we designed a study based on a questionnaire, and we conducted the study on a subset of the AmI course students.

A. General AmI project structure

Despite the specific goals of each AmI project, common architectural decisions and development tasks can be recognized. To gain a common understanding of the most painful issues that students across different projects experienced, it was necessary to represent those common architectural decisions, and their associated development tasks, into a general interpretation framework.

Figure 1 represents an overall architecture of the AmI projects. Five interconnected subsystems are identified: Sensors, Gateways, Back-end, Actuators, and End-user. Different projects need to implement different parts of the subsystems illustrated in the diagram; therefore, the proposed abstraction represents more a reference model in which all the projects' architectures could fit, than a unique common architecture that all of them had to implement. The following is a description of the identified subsystems.

Sensors generate data resulting from continuously monitoring the *End-user* behavior and activities, as well as environmental variables. In the context of the considered projects, they generally consist of wearable devices and wireless transmitters.

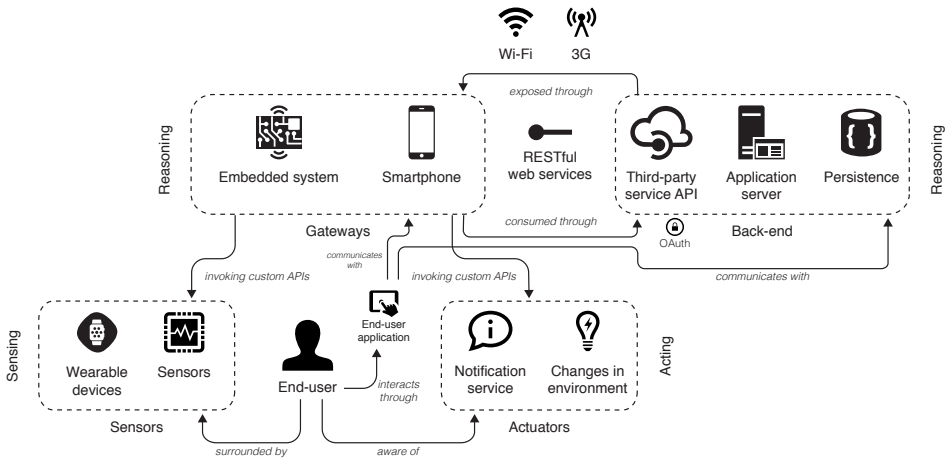

Fig. 1. Overall architecture of a typical Ambient Intelligence course project

Gateways gather the data produced by the *Sensors* and process this data, do some computation or reasoning over it, establish communication and transmit it to the main application server in the *Back-end*, and finally control the *Actuators* and send them commands and notifications. In the scenario of the course projects it encompasses embedded systems such as Raspberry Pi, and smartphones with their corresponding mobile applications.

In some cases, it is possible to gather the data directly from the *Sensors* via Bluetooth or Wi-Fi, but in some other it is necessary to gather the data through the consumption of these devices' specific cloud APIs, which usually requires OAuth authentication.

The **Back-end** encompasses the third-party service APIs, the main application server and the persistence component. Third-party service APIs, as just said, are commonly used to interact with wearable devices, such as Fitbit. The main application server and the persistence component, in the context of the course projects, consists of a set of RESTful web services and a relational database, correspondingly.

In some projects the functions of the *Back-end* components are delegated to the *Gateways*. However, no matter where the back-end services are deployed, they can be clearly distinguished as another subsystem.

Actuators span actuating devices, such as smart lighting systems, that may trigger changes in the physical environment, and remote push notifications on the end-user smartphone, through which he is notified about the occurrence of a given event. In the course projects the acting devices were generally controlled by the gateway devices via Bluetooth or Wi-Fi, while the remote push notifications are commonly generated by the *Back-end* server and delivered by the Android and iOS push notifications platforms.

End-user subsystem groups the interaction systems by which the end-user is able to interact with the *Sensors* and *Actuators* that surround him. A mobile or web application in the end-user smartphone is the most feasible alternative to enable this interaction. However, this application does not typically communicate with the *Sensors* and *Actuators*, instead, it communicates with gateway devices and the back-end components, which are able to control the behavior of the actuators and generate remote push notifications.

B. Method

The study was conducted by inviting the students to interview sessions, with one group invited per each session. Each session consisted on three phases: an introduction, the questionnaire, and a discussion. The sessions were conducted by two researchers, and were held in English.

1) Introduction: One researcher briefly explained the objective of the study, the structure of the questionnaire and in general the organization of the session. It was clarified that the questionnaire had to be filled from the personal point of view (each student should respond to those activities in which he was directly involved, only), while the following discussion would involve their evaluation as a group.

2) Questionnaire: A questionnaire was designed with the purpose of identifying, based on the personal experience of each member participating in the chosen projects, the most painful issues when developing an AmI system. Starting from the overall architecture, the structure of the questionnaire was defined focusing on three subsystems, only: the ones whose implementation and integration with other subsystems, relied mainly on software development activities. These subsystems were: End-user, Gateways, and Back-end. In the considered projects, in fact, sensors and actuators were off-the-shelf devices and didn't require software development activities (but only integration).

As *Gateways* and *Back-end* subsystems resulted quite large, in terms of number of tasks, their corresponding sections were divided in two: the first one for the subsystem *development* tasks, and the second one for the subsystem *integration* tasks.

In this manner, the questionnaire was structured in five sections, 20 tasks, and 62 sub-tasks. The resulting structure of the questionnaire in terms of subsystems and their tasks

is shown in Table I, that also report the number of sub-tasks defined per each task.

The questionnaire asked students, for each section:

- to identify the complexity of each of the sub-task in which they were involved. To quantify the complexity of each sub-task, they had to be **graded** according to their difficulty level and the time spent completing them. Two ordinal scales ranging from 1 to 5 were included in the questionnaire for each sub-task, as shown in Figure 2.
- to identify the three sub-tasks that were most difficult to complete. In this case, such sub-tasks had to be **ranked** as the first, second and third most difficult task in the section.
- to justify their ranking choice with an open question, where they could also mention any other tasks that was not listed but resulted complex to achieve.

Rank	Section A: End-user	Difficulty					Time spent				
Develop a native end-user mobile application											
	Become familiar with the mobile application platform-specific programming language	1	2	3	4	5	1	2	3	4	5
	Configure the development environment	1	2	3	4	5	1	2	3	4	5
	Develop the models' classes	1	2	3	4	5	1	2	3	4	5
	Develop the controllers' classes	1	2	3	4	5	1	2	3	4	5
	Develop the user interface (views)	1	2	3	4	5	1	2	3	4	5
	Connect the push notification module with the platform notification service	1	2	3	4	5	1	2	3	4	5
	Handle the notifications received in the end-user's smartphone	1	2	3	4	5	1	2	3	4	5

Fig. 2. Example of a task decomposed in sub-tasks

3) Discussion: After all students completed the questionnaire, a final discussion was held to identify, as a group, the most complex and painful tasks. The discussion was prompted by a single question: "As as group, which are the most difficult tasks you encountered in developing your system?" The students were free to discuss among themselves, and with the researchers.

C. Participants selection

In the last version of the AmI course 18 projects related to Health and Well-Being were developed. To identify the most painful issues that the students faced, from the software development perspective, two of these projects were chosen to be deeply analyzed in this preliminary study. The selection criteria for the two projects were based on their good results at the course showcase (they were among the winners) and on the software intensive nature of their implementation.

The first project, called *Safety Mama*, was targeted at pregnant women helping them to keep their stress and agitation levels low during their pregnancy. The system detects when the pregnant woman becomes stressed or agitated based on her heart rate and physical activity (number of steps), in which case, a relaxing setup is deployed until the moment in which the system detects again that the stress-levels are down.

From the implementation perspective, the system employs a wearable activity tracker that senses and monitors the heart rate and the number of steps of the pregnant woman. The data generated by the bracelet is transferred via Bluetooth to a mobile application that forwards it to the Raspberry Pi. This controller is responsible for reasoning over the data to generate remote push notifications to warn the pregnant women about the increasing detected stress levels and suggests her to move to a relax area, where the smart lighting system, the music player and the smart fragrances diffuser, are deployed.

The second project, called *Emergency Quest*, was aimed at improving the quality of life for people suffering from mild dementia or Alzheimer, while granting them greater autonomy and allowing their caregivers to be aware of their situation. By means of a wearable device, the system could constantly monitor the stress levels, location and activities performed by the patient. If the system detects, by means of the acceleration sensors of the bracelet, that the patient becomes agitated, it will react trying to calm him down through the deployment of a relaxing setup that includes music and lighting. Moreover, if the patient moves towards an artifact that might represent some sort of danger for him, such as a stove, the system would react by playing pre-recorded messages on the patient smartphone warning him about the dangerous situation. Additionally, if the patient abandons a certain area, the system would react warning his caregivers by generating and delivering notifications on their smartphones.

Technically speaking, the system uses a wearable activity tracker that monitors the patient location and sends the corresponding data, via Wi-Fi, to a micro-controller. This micro-controller communicates and transmits this data to the main application server, whose function is to generate and deliver push notifications into the end-user mobile application, and delegate the micro-controller to activate the relaxing system. To do the latter, the micro-controller communicates via Wi-Fi with the smart lighting system and the music player.

The *Safety Mama* group was composed by 4 students and *Emergency Quest* group was composed by 3 students. However, one of the members of the second group was an international student who returned to her home university, therefore a total of 6 students were involved in the study.

V. Results and Discussion

The study was conducted over two separate sessions, one for *Safety Mama* (with 4 students), the second for *Emergency Quest* (2 students). The completion of the questionnaire took each student, in average, approximately 50 minutes. The later discussion about the most painful issues as a group took around 20 minutes. From this study, three types of outcomes emerged: 1) the *grading* of each sub-task in terms of their associated difficulty level and completion time; 2) the set of tasks that were *ranked* as the most difficult and time-spending on each subsystem; 3) and the perception of the *group* about the most complex tasks.

The first kind of sub-tasks, in which the time-spent was high and the difficulty was low, corresponded mainly to *configuration* tasks. The second kind of sub-tasks, in which the task was perceived as difficult but not time-consuming, barely occurred. The third type of tasks, perceived as the most difficult and time spending, generally involve software

TABLE I
SUBSYSTEMS AND TASKS IN THE QUESTIONNAIRE

Section A: End-user subsystem	**# Sub-Tasks**
Develop a native end-user mobile application	7
Develop a hybrid end-user mobile application	8
Develop a web responsive end-user mobile application	6
Develop the integration between the end-user application and the gateways [computation node, smartphone]	2
Deploy the end-user mobile application into the smartphone	1
Section B: Gateways subsystem (Part I: Development)	
Configure the development environment	2
Develop the business logic of the gateway device [computation node, smartphone] application	2
Configure the OAuth authentication between the gateway device [computation node, smartphone] and third-party services APIs	3
Develop the module for generating notifications to be displayed on the end-user application	2
Deploy the software into the gateway devices [computation node, smartphone]	1
Section C: Gateways subsystem (Part II: Integration)	
Develop the integration between the gateway device [computation node, smartphone] and the sensors [wearable devices, static sensors]	4
Develop the integration between the gateway device [computation node, smartphone] and the back-end [third-party service API, application server, persistence] by consuming these last ones' custom APIs	4
Develop the integration between the gateway device and the actuators responsible for changes in environment	2
Section D: Back-end subsystem (Part I: Development)	
Configure the development environment	2
Design and develop the persistence component	3
Develop the business logic on the application server	2
Develop the RESTful web services	3
Section E: Back-end subsystem (Part II: Integration)	
Develop the integration between the application server and third-party services	2
Configure OAuth between the application server and third-party services	3
Develop the integration between the application server and the persistence component	3

configuration tasks for mobile applications, the integration with notification services and their invocation, and receiving and handling sensor data in real time.

From the second outcome, the most difficult sub-tasks could be distinguished from other sub-tasks and prioritized according to the *ranking*. Table II presents the list of highest-ranked sub-tasks that were chosen by the students in each section. The numbers in the right column represent the number of times in which the given sub-task was ranked as the first, second and third most difficult sub-task of the section, respectively.

From the third outcome, the members of the *Safety Mama* project identified as the most painful tasks, the following ones:

- The **integration** between the wearable activity tracker API (Fitbit bracelet), and the presentation layer of the End-user mobile application. Particularly when updating or changing the views, and the application was forced to re-fetch remote data.
- The poor or absent **documentation** regarding the integration between the Fitbit bracelet and the Android mobile application, especially in what concerns to available tools or libraries that would facilitate the OAuth authentication.
- The **communication** between the End-user mobile application and the Gateway reasoning devices in connection with the proper way to query the RESTful web services exposed by the latter and the data synchronization with the mobile application background services.

The members of the *Emergency Quest* project identified as the most painful tasks, the following ones:

- Regarding the **communication** between the mobile application and the main server, they experienced unpredictable problems, since the request that was outgoing from the hybrid mobile application towards the main server sometimes did not received back any response. Moreover, they did not have any tool to debug the communication.
- Dealing with the **remote push notifications** in the End-user hybrid mobile application, since the integration with the Android and iOS specific notification services had to be addressed. Therefore, the parametrization, and the inability to test the proper functioning of the notifications, represented to the group a major issue both in the *End-user* and the *Gateways* development.

In summary, by combining the information coming from the *grading*, from the *ranking* and from the *discussion*, some specific programming areas were recurrently mentioned as particularly hard and painful. In particular, the integration of different subsystems, that require over-the-network communication protocols, and their debugging was extremely difficult,

TABLE II
THE HIGHEST RANKED SUB-TASKS ON EACH SECTION OF THE QUESTIONNAIRE

Section A: End-user subsystem		**Ranks**
Develop a native end-user mobile application	Become familiar with the mobile platform-specific language	2–0–0
	Connect the push notification module with the platform service	1–0–0
Integrate the end-user application and the gateways	Parse and handle the JSON- or XML-formatted response	1–0–0
Section B: Gateways subsystem (Part I: Development)		
Configure the OAuth authentication between the gateway device and third-party services APIs	Develop the methods or functions required to establish the connection	3–1–0
	Set up the parameters needed to establish the connection	1–2–1
Develop the module for generating notifications on the end-user application	Generate the notifications by invoking the platform notification APIs	1–0–0
Section C: Gateways subsystem (Part II: Integration)		
Develop the integration between the gateway device and the sensors	Develop a component for receiving real-time data from the sensors	2–0–0
Develop the integration between the gateway device and the actuators	Develop the methods required to handle the actuators behavior	1–1–0
	Develop the methods required to establish the connection	1–0–1
Section D: Back-end subsystem (Part I: Development)		
Develop the RESTful web services	Define the HTTP methods along with their URI and operation	1–0–0
Section E: Back-end subsystem (Part II: Integration)		
Develop the integration between the application server and third-party services	Implement the HTTP asynchronous requests through the RESTful web services exposed by third-party service APIs	1–0–0
Develop the integration between the application server and the persistence component	Set up the connection between the application server and the database	1–0–0
Configure OAuth between the server and third-party services	Develop the methods or functions required to establish the connection	1–0–0

due to the diversity of client and server environments and the difficulty of tracing the remote calls. This was worsened by the fact that some third-party services are proprietary, give little visibility over their behavior, and each of them requires to follow different approaches and programming patterns. Also, the configuration of mobile, web or hybrid applications was perceived as a pain point, since all the plugins, libraries and dependencies had to be satisfied, and the rules are different in each environment.

VI. CONCLUSIONS

This paper presents an exploratory study to identify the most painful points that novice programmers experience when developing AmI systems. Undergraduate students who developed an AmI project in the context of a university course were involved to share their experience. An AmI architecture was elaborated to abstract the common architectural decisions taken across the projects. Over this architecture, five interconnected subsystems were identified along with their associated software development tasks and sub-tasks, and a questionnaire was developed based on the overall architecture and the development tasks. The study was conducted in three phases that enabled identification of the pain points from a quantitative and qualitative point of view. From the study outcomes, students perceived as extremely difficult the sub-tasks regarding: the integration of different subsystems, interaction with proprietary third-party services, and the configuration of mobile, web or hybrid applications. Other sub-tasks regarding configuration of the subsystems were perceived as simple but time-demanding. By structuring the study around the overall architecture, a system-level view of the main pain points was achieved. Future work will consist in designing and developing mechanisms (tools and methodologies) targeted at supporting novice programmers in developing AmI systems, thanks to the analysis of the collected data. In parallel, we are extending the application of the questionnaire to a larger number of students, from the last 3 years of the course.

REFERENCES

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497 – 1516, 2012.

[2] N. He, Y. Qian, and H. w. Huang, "Experience of teaching embedded systems design with beaglebone black board," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, May 2016, pp. 0217–0220.

[3] G. Kortuem, A. K. Bandara, N. Smith, M. Richards, and M. Petre, "Educating the internet-of-things generation," *Computer*, vol. 46, no. 2, pp. 53–61, Feb 2013.

[4] D. Dobrilovic and S. Zeljko, "Design of open-source platform for introducing internet of things in university curricula," in *2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2016, pp. 273–276.

[5] O. Hahm, E. Baccelli, H. Petersen, M. Wählisch, and T. C. Schmidt, "Demonstration abstract: Simply riot - teaching and experimental research in the internet of things," in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, April 2014, pp. 329–330.

[6] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, Aug. 2009.

[7] F. Corno and L. De Russis, "Training engineers for the ambient intelligence challenge," *IEEE Transactions on Education*, vol. 60, no. 1, pp. 40–49, Feb 2017.