HelloArduBot: A DSL For Teaching Programming To Incoming Students With Open-source Robotic (OSR) Projects

Gustavo Slomski
gustavoslomski@gmail.com
Federal University of Technology, Paraná (UTFPR)
Francisco Beltrão, Brazil

Adair José Rohling
adairrohling@utfpr.edu.br
Federal University of Technology, Paraná (UTFPR)
Francisco Beltrão, Brazil

Paulo Varela
paulovarela@utfpr.edu.br
Federal University of Technology, Paraná (UTFPR)
Francisco Beltrão, Brazil

Michel Albonico
michelalbonico@utfpr.edu.br
Federal University of Technology, Paraná (UTFPR)
Francisco Beltrão, Brazil

ABSTRACT

Block-based languages have been used as a facilitator to teach programming to newcomer and end-user programming students. Another alternative is to abstract the programming domain by using educational robots. Such approaches face some challenges. Block-based languages are far different than conventional programming languages, resulting in an abrupt transition between the two paradigms. On the other hand, commercial educational robots are limited to predesigned projects, which bounds students' creativity. In this work, we propose an intermediate language (between blocks and traditional language that focuses on Arduino, allowing a wide range and student-designed projects. Preliminary results show that our language is simpler than the native Arduino language and that it would be a preferred alternative for beginner students of a computer science undergraduate course.

CCS CONCEPTS

• **Applied computing** → *Interactive learning environments*; • **Software and its engineering** → **Domain specific languages**; • **Computer systems organization** → *External interfaces for robotics.*

KEYWORDS

teaching programming, domain-specific language, open-source, robot-based.

ACM Reference Format:
Gustavo Slomski, Adair José Rohling, Paulo Varela, and Michel Albonico. 2022. HelloArduBot: A DSL For Teaching Programming To Incoming Students With Open-source Robotic (OSR) Projects. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2022), 5 pages. https://doi.org/10.1145/3555051.3555070

Authors' addresses: Gustavo Slomski, gustavoslomski@gmail.com, Federal University of Technology, Paraná (UTFPR), Francisco Beltrão, Brazil; Adair José Rohling, adairrohling@utfpr.edu.br, Federal University of Technology, Paraná (UTFPR), Francisco Beltrão, Brazil; Paulo Varela, paulovarela@utfpr.edu.br, Federal University of Technology, Paraná (UTFPR), Francisco Beltrão, Brazil; Michel Albonico, michelalbonico@utfpr.edu.br, Federal University of Technology, Paraná (UTFPR), Francisco Beltrão, Brazil.

2476-1249/2022/8-ART111 $15.00
https://doi.org/10.1145/3555051.3555070

1 INTRODUCTION

While learning programming, students usually struggle to understand **what** to program since they may not be connected to the topic [5]. Therefore, some professors rely on project-based learning methodologies, usually by using educational robots, which makes the programming topic more tangible [11, 18].

While robots help solve the problem of abstracting what to program, the students still need to learn **how** to program, which may be illogical at the beginning. Facing this, there has been a movement of visual programming tools, such as Scratch [13]. At some point after those, students need to move to real programming languages, which is an abrupt paradigm change. The process can be more didactic with a gradual learning approach, where single concepts being introduced at each step [7].

A the end, it is important **where** the students run their software. Commercial robotic kits, such as Turtlebot and Lego Mindstorms, are limited to specific types of robots, which restricts students' project scope. Such kits are also expensive, the Turtlebot 4, for instance, costs more than 1 thousand US dollars. As a solution, there are available open-source hardware platforms [2], especially the Arduino [3], which enables a broad range of low-cost robotic projects[1]. Besides the low-cost nature, we choose Arduino among other prototyping platforms[2] since it also provides great documentation and public projects.

Such contextualization leads us to *three main* **motivations**: i) the use of robots for teaching programming is an important factor on WHAT to program; ii) the programming language learning process must be the most gradual as possible, so students assimilate HOW to program; iii) WHERE to run robot-based software should be open-source, so it allows a more creative learning.

The main **goal** of this work is to provide an intermediary programming language specific for open-source robot-based projects. Our programming language aims at making the transition between visual and traditional programming languages gradual, besides being also an alternative for abstract programming of robotics projects. On the other hand, open-source prototyping platforms should enable more creative projects.

[1] https://create.arduino.cc/projecthub
[2] https://craft.co/arduino/competitors

In order to reach the goals, we design a **robot-based DSL language** and make it open-source. To model the language grammar, we choose the XText framework [9], which is open-source and part of Eclipse Modeling Framework [8] Our language abstracts a great part of the programming complexity for educational robotics by focusing only on robot's capacities and the main programming structures. As a consequence, the code should be cleaner and easier to understand. Finally, our language allows the students to program on Arduino micro-controllers, which enables a vast portfolio of robotic projects. This work only highlights, preliminary results, such as codes with the proposed language, and a brief survey in an undergraduate computing science course. We plan to do a more extensive evaluation of the tool artifacts and the language in future work.

2 RELATED WORK

Robotic has been present in educational projects since the late seventies. Papert's work on Logo Turtle [16] plays an important role in such popularization [21], which also influenced the design of educational products that are still on the market, such as Turtlebot [17] and Lego Mindstorms [15]. Despite Lego Mindstorms relies on a visual programming language (block-based), their prototyping platform is limited by only a few concepts, the same as for the Turtlebot. Furthermore, the Turtlebot does not provide a friendly programming interface/paradigm. Furthermore, both of them are expensive if compared to open-source hardware, such as Arduino and ESP32 [10], which would make their adoption restricted to environments with a significant budget.

In the literature, we also find robotic-specific languages. Baillie [4] proposes the URBI, a universal robotic body interface, a client/server system, where the robot works as a server, receiving commands remotely. Despite it being limited to only one type of robot, nowadays, such approach would be easily replaced by robotic frameworks, such as ROS [19]. Furthermore, the language, despite being domain-specific, is not designed to be end-user friendly. Roland and Gontean propose the AL5 language, which is restricted to a specific subdomain of robotics (i.e., robotic arms). Despite their language being designed for teaching, it aims at teaching the robot movements, instead of using the robot as a real-world abstraction for teaching programming. A work that goes towards our goal is the one by Kim and Jeon [12], which proposes a language for teaching programming with robotic projects. However, their work is based on Lego Mindstorms robots, which downsides have already been discussed before.

3 HELLOARDUBOT LANGUAGE

The proposed programming language is named *HelloArduBot*, a robot-based domain specific language (DSL). The language name is a reference to the common "hello world" first coding (Hello), Arduino used as the initial open-source prototyping platform (Ardu), and the robotic domain (Bot). For achieving the goal, HelloArduBot meets 4 main requirements:

- *Req1.* to address open-source robotic platforms, so the students can express their creativity with a broader range of possibilities, without the limited possibilities provided by commercial platforms already discussed;
- *Req2.* to be robot-specific, so projects are more appealing to the students, and they need to create their own projects;
- *Req3.* to introduce structured programming concepts, such as loops and procedures;
- *Req4.* to be textual, so students become aware of how conventional programming works;

We design our DSL grammar by using XText framework, wich is part of Eclipse Modeling Framework, popular in the Model-Driven Engineering (MDE) community. The language translation/building is performed with XTend, part of the Xtext framework.

Figure 1 shows the partial Ecore metamodel of our language. As a matter of presentation, we only present the parts related to the robot's interaction.

As illustrated in the meta-model, the robotic-related modeling is simplified, divided into *actuation* and *sensing*. The *sensing* element requires only two attributes: *port* to which the sensor is connected, and *type* of port, i.e., digital or analog. The *actuation* is a branch divided into two ways the robot interacts with the external world: movement and notifications. The meta-model foresees two common movements *run* and *turn*, for implementing wheeled robots and angular positioning, such as for a robotic arm. We consider three basic types of notification for User Interface (UI): flashing (with LED flashes), display (with Liquid Crystal Displays - LCD)[3], and alarm (with buzzers). We choose to keep the language as simple as possible. For further needs, the metamodel can be easily extended.

4 RESEARCH DESIGN/EVALUATION PLAN

In this work, we only present a new research idea and; therefore, we do not conduct a deep validation, but already provide preliminary results. For the purpose of the DSL empirical evaluation, we plan (in the future) to apply the new language in two scenarios: i) on programming language introductory classes, and ii) on a science outreach project at an elementary school (where we teach basic robotics). The goal is to assess the impact of the HelloArduBot language on programming understanding for both students, newcomers, and robotic end-users.

The main points of the HelloArduBot language are to make the code easier to understand, and as a consequence, to smooth the transition between block-based and commercial programming languages. Our evaluation plan goes toward such features, which relies on two empirical user studies [20]: i) a survey approach, and ii) a subject-based experiment. The first study investigates the syntax comprehension of the HelloArduBot language, while the second one investigates the speed and accuracy of reading the code.

In the first study, we plan to survey the students before they start working on their projects. We will use a survey form that addresses two main factors: i) the student's current background in a programming language, and ii) their comprehension of simple code snippets in two languages (native Arduino and HelloArduBot). It is expected that undergraduate students will already have some background in programming language syntax, while elementary school students will not (based on previous classes over the years). Therefore, we can evaluate whether the HelloArduBot is easier to comprehend, and which is its acceptance by those who have already programmed in other introductory programming languages.

[3]Note that the display requires multiple pins, which are reserved (3-5 and 11-12).

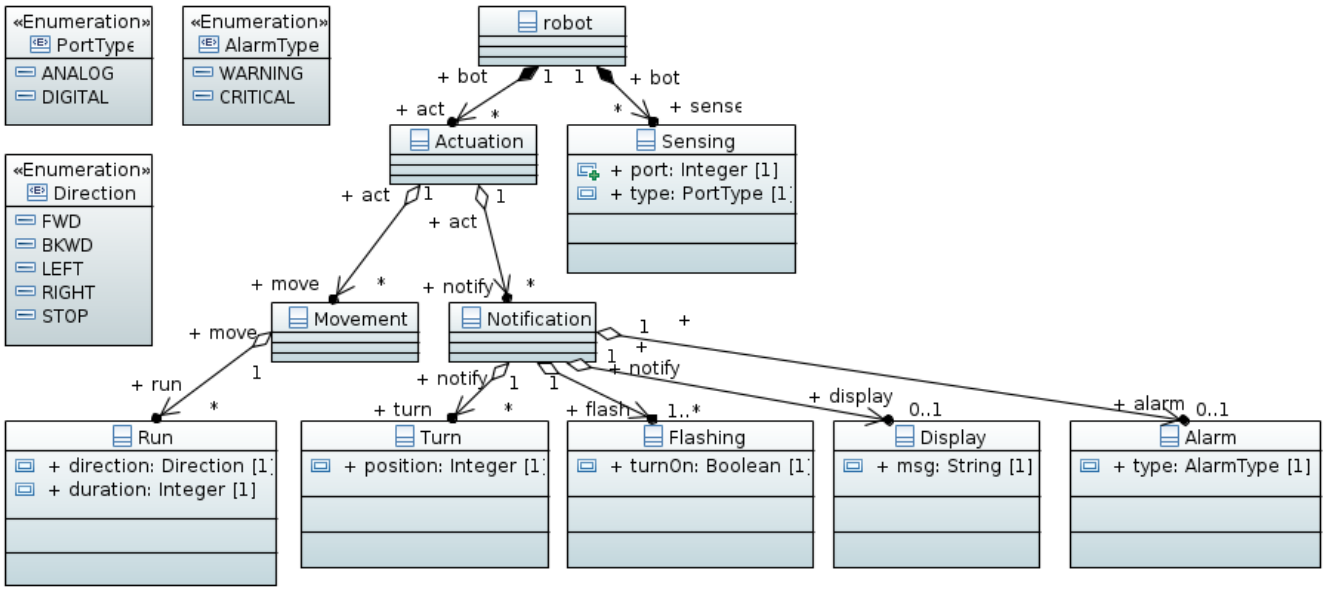


Figure 1: Partial metamodel of the language grammar (model).

The second study is a controlled subject experiment, used to evaluate the code written in the HelloArduBot language. The study must be conducted only with undergraduate students, who already studied basic programming languages (except the HelloArduBot). We plan to write codes in native Arduino and HelloArduBot for two different robotic projects. Then, we split the classes into groups, where each group will receive an Arduino and a HelloArduBot code, both from different projects. Without knowing what are the projects, they must describe what those codes are supposed to do. This will give us insights into how quick and accurate they can read others' code, and whether HelloArduBot can ease such a process and improve readability [6].

After the evaluation experiments, we also plan to validate the HelloArduBot language as an alternative to smooth the transition between block-based and commercial programming languages. We plan to conduct a qualitative analysis of the project completeness when using the native Arduino and HelloArduBot languages. This validation can also be applied to elementary school and undergraduate students. Basically, we split the students into groups with similar knowledge of programming languages, and challenge them to program a robotic project. Part of the groups delivers only block-based and native Arduino code, while others also deliver the HelloArduBot language code. All the groups will work within a time-constrained interval, and only at the robotic lab. Then, we qualitatively analyze the features delivered by the groups.

5 PRELIMINARY RESULTS

This work results in i) an extensible metamodel that basis the HelloArduBot grammar; ii) the implementation of an intermediary robot-specific language for teaching incoming programming students; iii) an editor for writing the code with an auto-completion method; iv) a language translator that can be extended to other languages besides C++ (used by Arduino); v) a replication package with all the software licensed under the MIT license[4].

Our Ecore metamodel can work as a reference for further Eclipse plugins or even standalone MDE applications. Our language abbreviates the complexity of robotics programming to simple tasks, which still allows a comprehensible number of projects to be implemented. Its editor is available online, and can also run in standalone mode. The editor comprehends a language translator, where the Arduino code can be downloaded, and then uploaded straight to the robot. Since all the software artifacts are publicly available under an open-source license, they can also be adapted to further needs.

5.1 HelloArduBot Coding

Figure 2, shows a simple public robotic project[5] that we use as study case to exemplify the difference between coding in native Arduino and HelloArduBot languages. It refers to a simple public robotic project coding that rotates a DC motor left and right. Listings 1 and 2 show the code for the case study in both, Arduino native and HelloArduBot language, where the simplicity of the HelloArduBot is evident.

[4] https://anonymous.4open.science/r/HelloArduBot-672D/
[5] https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor

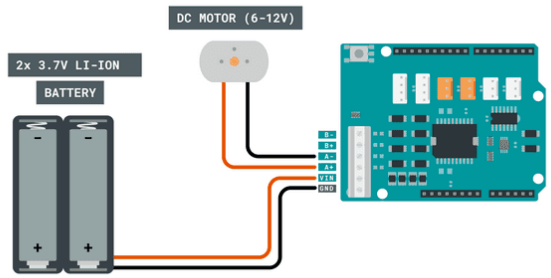


Figure 2: Sketch of the case study robotic project.

```
int directionPin = 12;
int pwmPin = 3;
int brakePin = 9;
bool directionState;

void setup() {
  pinMode(directionPin, OUTPUT);
  pinMode(pwmPin, OUTPUT);
  pinMode(brakePin, OUTPUT);
}

void loop() {
 directionState = !directionState;
 if(directionState == false){
   digitalWrite(directionPin, LOW);
 } else{
   digitalWrite(directionPin, HIGH);
 }
 digitalWrite(brakePin, LOW);
 analogWrite(pwmPin, 30);
 delay(2000);
 digitalWrite(brakePin, HIGH);
 analogWrite(pwmPin, 0);
 delays(2000);
}
```

Listing 1: Arduino native code.

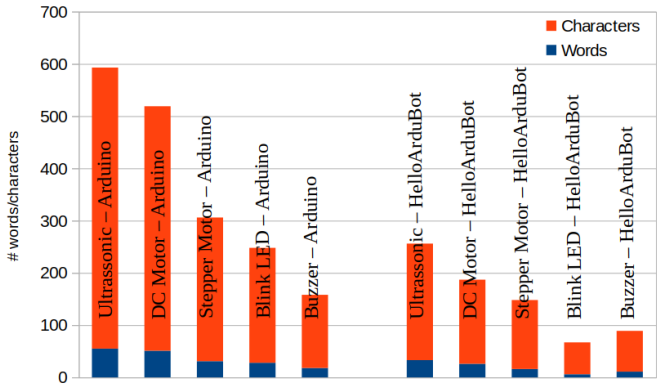
```
Vars
  Number duration = 5000
  Number direction = FWD
Begin
  IF direction == FWD THEN
    direction = BKWD
  ELSE
    direction = FWD
  FI
  run(direction, duration)
End
```

Listing 2: HelloArduBot code.

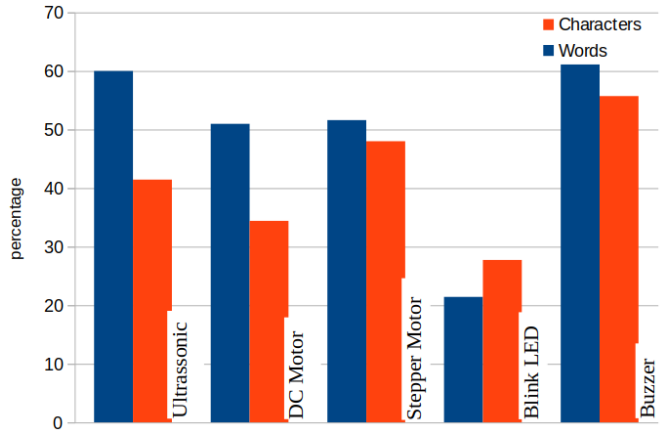
As a matter of further comparison, Figure 3 illustrates the number of words and characters for the case study and four other public Arduino projects: i) blink a LED[6], ii) buzzer[7], iii) ultrassonic sensor[8], iv) stepper motor[9]. Reviewers can assess such codes that are provided in the replication package. Our *hypothesis* here is that:

[6]https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
[7]https://create.arduino.cc/projecthub/SURYATEJA/use-a-buzzer-module-piezo-speaker-using-arduino-uno-89df45
[8]https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6
[9]https://www.arduino.cc/en/Tutorial/LibraryExamples/StepperSpeedControl

less words and, as consequence, less characters may be more appealing for the students, and cause less confusion.



(a) Number of words and characters.



(b) Code reduction with HelloArduBot.

Figure 3: Comparison between native Arduino and HelloArduBot code.

In the figure, we see that despite HelloArduBot results in fewer words/characters, at least ≈ 20% smaller than native Arduino (higher in most of the cases), there is no proportionality between native Arduino and HelloArduBot codes (see Blink LED results).

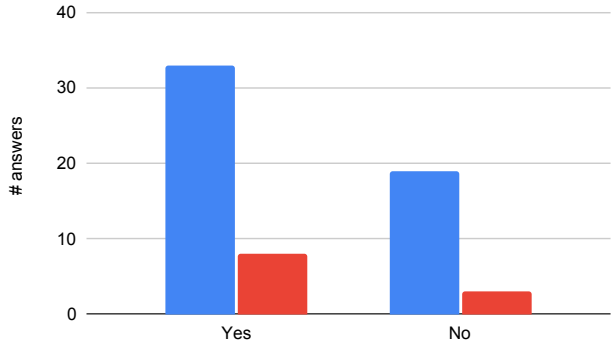
We see that HelloArduBot code is simpler than the native Arduino one, with more abstract concepts. For instance, instead of setting up all the environment, including the DC motors for the robot's wheel, it is only necessary to program the *run* method call. This also results in fewer words/characters (in the example, 26/161 vs 51/468). This is confirmed in Figure 3a, where for all the examples, the HelloArduBot resulted in fewer words and characters. Figure 3b depicts that difference, which goes from ≈ 20 – ≈ 60%.

5.2 Survey – Computer Science Students

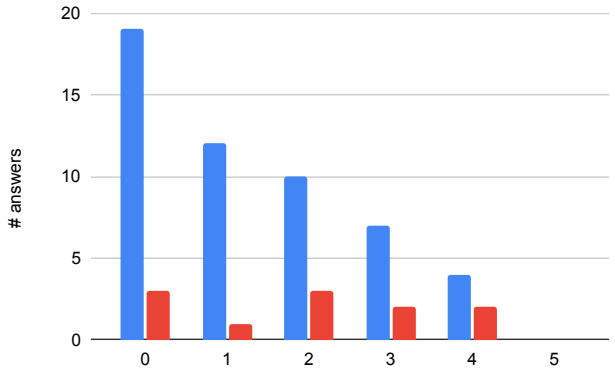
We also survey 63 computer science students about the HelloArduBot idea. The survey questions try to investigate two main correlations: i) previous knowledge in programming languages (0 for *none* or 1 for *some*) and their preference for an intermediary language such as HelloArduBot (we provide code snippets), ii) how

they classify their own programming skills (0–5) and their preference for a intermediary language such as HelloArduBot.

Figure 4 depicts the result of the preliminary survey with the students. We see that most of the students would prefer an intermediary language (Figure 4a), even if they already have some knowledge in programming languages. As depicted in Figure 4b, none student auto-classified (her)himself as expert. Such figure also shows a pre-ference for HelloArduBot for beginners (levels 0-2 mainly), which indeed is the its main purpose.



(a) Previous knowledge in programming.



(b) Programming skill.

Figure 4: Computer science students' interest in the HelloArduBot language.

6 CONCLUSION

The HelloArduBot language impacts at least two main scenarios: i) teaching end-user programmers, such as school students with robotics in their curriculum; and ii) teaching introductory programming classes for computer science. Given its higher abstraction level, it is expected to smoother programming paradigms transition. It is also expected that the language is used directly for abstract programming in robotic classes.

In future work, as stated in Section 4 we must evaluate our approach with elementary school and undergraduate students, comparing it to conventional programming languages. We plan to convert graphical programming languages (such as Ardublock [1], which makes the process even more facilitated. So far, the students must write the HelloArduBot code from scratch. Finally, the language should also be extended to support open-source robot-specific frameworks, such as ROS [19] and micro-ROS [14], so robot prototypes can be used in real-world robotic environments.

REFERENCES

[1] A Graphical Programming Language for Arduino. 2022. Ardublock [Online]. Retrieved May 20, 2022 from http://blog.ardublock.com/
[2] Michel Albonico, Adair Rohling, Juliano Santos, and Paulo Varela. 2021. Mining Evidences of Internet of Robotic Things (IoRT) Software from Open Source Projects. In *15th Brazilian Symposium on Software Components, Architectures, and Reuse* (Joinville, Brazil) *(SBCARS '21)*. Association for Computing Machinery, New York, NY, USA, 71–79. https://doi.org/10.1145/3483899.3483900
[3] Arduino. 2022. Arduino[Online]. Retrieved May 20, 2022 from https://www.arduino.cc/
[4] J.-C. Baillie. 2005. URBI: towards a universal robotic low-level programming language. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 820–825. https://doi.org/10.1109/IROS.2005.1545467
[5] Gillian Bain and Ian Barnes. 2014. Why is programming so hard to learn?. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*.
[6] Dustin Boswell and Trevor Foucher. 2011. *The Art of Readable Code: Simple and Practical Techniques for Writing Better Code.* " O'Reilly Media, Inc.". https://www.oreilly.com/library/view/the-art-of/9781449318482/ch01.html
[7] Walter Cazzola and Diego Mathias Olivares. 2015. Gradually learning programming supported by a growable programming language. *IEEE Transactions on Emerging Topics in Computing* 4, 3 (2015), 404–415.
[8] Eclipse Foundation. 2022. Eclipse Modeling Project [Online]. Retrieved May 20, 2022 from https://projects.eclipse.org/projects/modeling
[9] Eclipse Foundation. 2022. Xtext - Language Engineering Made Easy [Online]. Retrieved May 20, 2022 from https://www.eclipse.org/Xtext
[10] Expressif Systems. 2022. ESP32 [Online]. Retrieved May 20, 2022 from https://www.espressif.com/en/products/socs/esp32
[11] Bassey Isong. 2014. A Methodology for Teaching Computer Programming: first year students' perspective. *IJMECS* 6, 9 (2014), 15.
[12] Seung Han Kim and Jae Wook Jeon. 2006. Educating C language using LEGO Mindstorms robotic invention system 2.0. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 715–720.
[13] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 1–15.
[14] MicroROS. 2022. ROS 2 Onto Microcontrollers [Online]. Retrieved May 20, 2022 from https://micro.ros.org/
[15] Mindstorms. 2022. Lego [Online]. Retrieved May 20, 2022 from https://www.lego.com/en-gb/themes/mindstorms
[16] MIT. 2022. Logo History [Online]. Retrieved May 20, 2022 from https://el.media.mit.edu/logo-foundation/what_is_logo/history.html
[17] Open Source Robotics Foundation. 2022. TurtleBot [Online]. Retrieved May 20, 2022 from https://www.turtlebot.com/
[18] Martinha Piteira and Carlos Costa. 2013. Learning computer programming: study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*. 75–80.
[19] ROS. 2022. Robot Operating System [Online]. Retrieved May 20, 2022 from https://www.ros.org/
[20] Mazin Saeed, Faisal Saleh, Sadiq Al-Insaif, and Mohamed El-Attar. 2016. Empirical validating the cognitive effectiveness of a new feature diagrams visual syntax. *Information and Software Technology* 71 (2016), 1–26.
[21] Cynthia J Solomon and Seymour Papert. 1976. A case study of a young child doing Turtle Graphics in LOGO. In *Proceedings of the June 7-10, 1976, national computer conference and exposition*. 1049–1056.