FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR

LABORATOIRE DE BIOROBOTIQUE
CH - 1015 LAUSANNE

PROJET DE SEMESTRE – PRINTEMPS 2020

Titre:	Development of a mobile application for computer-vision based tangible programming languages		
Candidat(s):	Anthony Guinchard	**Section:**	Microtechnique
Professeur:	Prof. Francesco Mondada		
Assistant 1:	Christian Giang	**Assistant 2:**	Laila El Hamamsy

Donnée & travail demandé :

The goal of this project is to further develop the existing prototype of the PaPL platform, so that it can be evaluated in classroom user tests. The student will work on the improvement of the computer vision algorithm and develop a PaPL implementation for mobile devices (Android or iOS). This includes the creation of a mobile application as well as the fabrication of the tangible programming tiles. Finally, the student will develop the bridge for the communication with the new robot developed at the University of the Azores. In case of rapid progress, the student may be involved in the planning, execution and/or analysis of research studies. For the time of the project, the student will be provided, two tablets (nVidia Shield NV05 and Sony Xperia Z2), a BlueBot robot and a tangible programming set for the robot, which are to be returned after the completion of the semester project.

Remarques :

Un plan de travail sera établi et présenté aux assistants avant le vendredi 29 février 2020.

Une présentation intermédiaire (environ 10 minutes de présentation et 10 minutes de discussion) de votre travail aura lieu dans le courant du mois d'avril 2020. Elle a pour objectifs de donner un rapide résumé du travail déjà effectué, de proposer un plan précis pour la suite du projet et d'en discuter les options principales.

Un rapport, comprenant en son début un titre avec les données générales et une photo portrait de l'étudiant, suivi par l'énoncé du travail (présent document), puis d'un résumé d'une page (selon canevas), sera remis le lundi 1 juin 2020 avant 12 heures en format électronique à l'assistant responsable. L'accent sera mis sur les expériences et les résultats obtenus. Le public cible est de type ingénieur EPF sans connaissance pointue du domaine. Une version préliminaire du rapport sera remise à l'assistant le 15 mai 2020. Tous les documents en version informatique, y compris le rapport (aussi en version source), le document de la présentation orale, ainsi que les sources des différents programmes doivent rendu seon les consignes qui seront donnée avant la terminaison du projet.

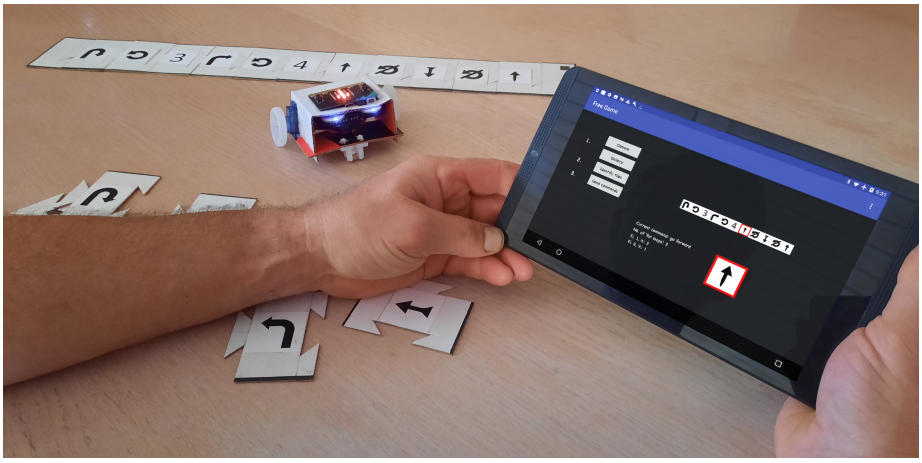
Une défense de 30 minutes (environ 20 minutes de présentation et démonstration, plus 10 minutes de réponses aux questions) aura lieu dans la période du 1 au 12 juin 2020.

Le professeur responsable:	L'assistant responsable:
Signature :	Signature :
Prof. Francesco Mondada	Christian Giang

Lausanne, le 26 janvier 2019

Semester project – Spring 2020

Development of a mobile application for computer-vision based tangible programming languages



Professor:
Francesco Mondada

Student:
Anthony Guinchard

Assistants:
Christian Giang
Laila El Hamamsy

June 16, 2020

Abstract

EPFL's Tangible Programming Language (TPL) [4] project was initiated in 2019 by Andrea Mussati, Christian Giang, Alberto Piatti and Francesco Mondada from the MOBOTS lab and SUPSI-DFA in Ticino. The first version of TPL at EPFL was based on the use of an external webcam continuously streaming an arrangement of physical tiles that were intended to program the Thymio robot. A few months later, Julien Dedelley, Noé Duruz and Aditya Mehrotra continued working on the project and moved to an alternative of TPL which was called Paper-based Programming Language (PaPL) [3]. As its name suggests, it proposes exclusively paper-based programming tiles. In addition, they made use of the built-in webcam of a computer and a mirror to analyze the program. Up until then, everything was aimed at using the Thymio robot [31]. For the current and third iteration of the project, a Maker's approach was considered. The Thymio was set aside and replaced by another two wheeled differential drive robot based on the micro:bit [35] and made mostly of cardboard. Moreover, the robot can be programmed using a set of tiles presenting simple commands that can be aligned to form sequences of instructions. The purpose was to elaborate a educational robotics system that anyone could build at home with very few and easily accessible resources. Both the robot and the tiles are made of cardboard. The project was thus named Cardboard-based programming Language (CaPL). An Android application allows to establish the link between the programming tangible tiles and the cardboard robot by recognizing sequences of commands. This report presents the design and implementation of different key elements of the project and, in the end, the emphasis is put on the elaboration and the functionalities of the activities constituting the dedicated tablet-based application.

Acknowledgements

I would like to thank all the people who helped me to realize this project and supported me during my second semester of Master in Robotics at EPFL. First of all, I would like to thank my assistants Christian Giang and Laila El Hamamsy for having taken care of the regular progress of the project and for having pointed me in the right direction and choices to take. I would also like to thank Professor Francesco Mondada for giving his approval to pursue a TPL project and for having had to take important decisions regarding the reorganisation of EPFL due to COVID-19. Thanks a lot also to Amaury Dame for his brilliant ideas and help regarding micro:bit and DIY considerations, Aditya Mehrotra for testing each new version of the Android application and finding a simple and nice design for the Cardbot, Melissa Skweres for giving her feedback and encouragements for pursuing the project, Lilia Da Costa for having kindly drawn the fancy world map of the "Geography Game" activity and my brother Tanguy Guinchard for having lent me his film equipment to prepare the cardboard construction tutorials. I would finally like to thank Christian Giang, Laila El Hamamsy and Aditya Mehrotra once again for proofreading and correcting this report.

Contents

List of Figures

List of Algorithms

1 Introduction

1.1 Tangible interfaces

The growing digitalization of our world has recently led society to realize the importance of learning robotics, programming and computational thinking for future generations. Several primary and secondary schools have already integrated these concepts into their learning program [21, 45, 43, 19]. To introduce children to fundamental computer science concepts, graphical programming languages such as Scratch [37], Blockly [38] or Alice [46] are often used. Research conducted in recent years showed however that the use of tangible platforms for learning the basics of programming can improve interaction, engagement and collaboration among students [20]. A tangible programming language (TPL) consists of physical blocks that can be grouped to compose a program to control virtual agents and/or real educational robots. In comparison with screen-based activities, it has been shown that a learning approach based on physical objects can in fact create more interest [16, 18] and a higher level of participation within workgroups [34, 4]. Moreover, TPLs provide, on the one hand, a reassuring framework for teachers and parents who are opposed to the use of screens in classrooms [3]. On the other hand, using these platforms can facilitate the understanding of the first milestone in basic robotics and programming by involving more playful ways of learning [15] and can lead to better learning gains [47]. For instance, using tangible programming tiles allows teachers to easily adapt constraints of exercises, such as limiting the set of available commands to reinforce reflection [3]. In addition, with this kind of systems, a group of several children can interact in the same digital environment. As opposed to computer activities where students usually work alone, tangible interfaces can hence impact their level of participation and motivation [28] by setting up workgroup activities. An important drawback of these robotics educational platforms, however, is that they might be expensive and a lot of schools are not able to afford them. Recent projects conducted at EPFL went over this issue by developing accessible tangible programming platforms.

1.2 TPL background at EPFL

In 2019, two projects have been proposed by EPFL researchers and students in order to develop low-cost TPL platforms and make them more accessible. At the beginning of 2019, Andrea Mussati set up the first version of a tangible programming language [4]. It was aimed at controlling the Thymio [31] educational robot thanks to an external webcam identifying VPL-like commands [24]. The programming tiles were either made of cardboard or 3D printed and could be customized in order to fit the needs of a class (e.g. simplified symbols for younger students, or more complex symbols to introduce the notions of variables and functions to more advanced students). The commands could be assembled in a puzzle-like fashion. The image of these commands was automatically processed by an external webcam and the corresponding instructions were then sent wirelessly to the robot. One of the purposes of this first project was to create an accessible transition between the TPL and the VPL platforms. An example of program in both versions (VPL and TPL) is depicted in figure 1. It was aimed at helping novices to get used to the robotics event-action principle. The study conducted during this project [4] wanted to assess the potential of a Thymio tangible programming language by evaluating the level of interaction of students with the VPL and TPL platforms. The results showed that TPL was more attractive than VPL and that students tended to participate more actively with the TPL than with the VPL platform. By interacting with the programming tiles and the robot, the students additionally showed more collaborative behaviour.

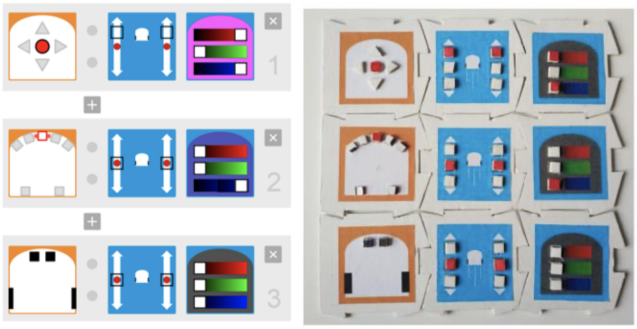


Figure 1: Example of program composed with the VPL interface and its corresponding TPL version

A few months after this first project, Julien Dedelley, Noé Duruz and Aditya Mehrotra developed a similar tangible programming language for the Thymio, but based exclusively on paper and using the built-in camera of a laptop [3]. Again, they integrated specific pattern recognition mechanisms to interpret sequences of commands and then send them to the Thymio. The researchers conducted a study with in total more than 100 senior year high school students and obtained promising results regarding the use of their PaPL (paper-based programming language) system in educational robotics. It was shown that PaPL may improve communication and collaboration in groups. In fact, with tangible programming, the students have to find arguments to defend their ideas in front of their peers before tackling programming. In addition, some teachers suggested that the introduction of PaPL in classrooms could be used as a concrete basis before learning how to use VPL. With their study, Julien, Noé and Aditya not only demonstrated that PaPL engaged the interest of children by involving tangible objects, but that it also allowed the teachers to be more available for helping the different groups due to the well-framed environment of the platform. One of the purposes of this project was to find a way to introduce STEM (science, technology, engineering and mathematics) with Thymio in an even more accessible and inexpensive manner than with the previous project. In fact, not only the educational robot used was again the widely spread Thymio, but the activity required only a computer and simple craft materials such as paper. The kind of PaPL command tiles presented in figure 2 needed hence only to be printed. Again, the study conducted in the frame of this project showed that a tangible platform is more inviting to the participants than its corresponding uniquely screen-based interface.

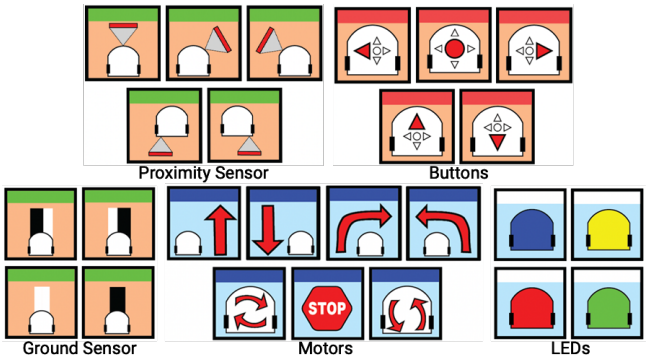


Figure 2: Example of blocks from of the PaPL platform

Both projects presented low-cost tangible programming platforms, but, up until then, everything was implemented with Thymio. Its advantage is that it is an educational robot already widely used in primary and secondary schools in Switzerland and other Central European countries to introduce STEM and CS (computer science) [4]. Thus, no additional expenses are required by most schools. But this only applies to institutions that already have a Thymio. The project described in this report proposes a new tangible programming language designed to work with a DIY (do-it-yourself) robot.

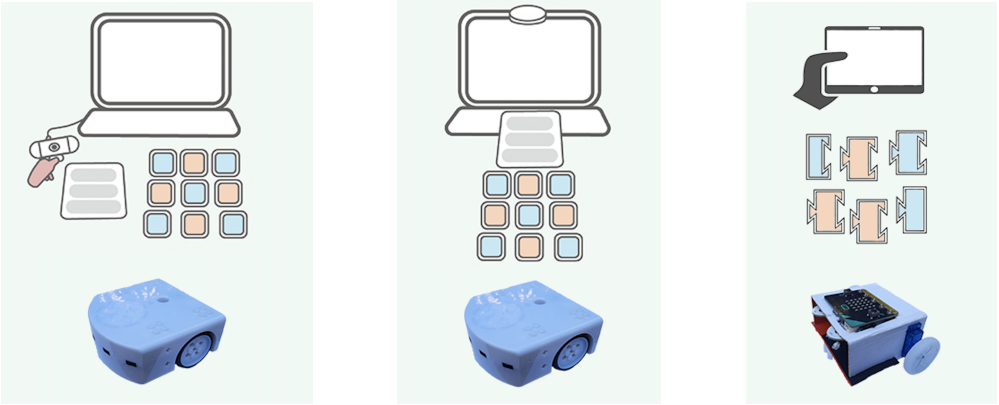
1.3 CaPL project

In the same perspective as the two previous projects, the Cardboard-based Programming Language (CaPL) project is a tangible platform aiming at setting up a smooth entry point to computing education. The interface stays simple and intuitive, but this time, paper is replaced by cardboard and a tablet is used instead of a computer. The idea is to create a do-it-yourself system that can be easily manufactured by teachers and students limiting additional costs. Cardboard is an inexpensive and accessible material that can be found in most schools. It is also an ideal material for do-it-yourself activities since it can easily be cut, folded and assembled. Moreover, cardboard is more robust than paper and can withstand a relatively longer-term use. This is an important feature of the programming tiles since they are here designed to be clipped together to form a sequence of commands for the robot. In addition to personal computers, most classrooms today have access to tablets. They are more handy and less expensive than computers. This also limits additional charges. In the CaPL project, a tablet is used to make the link between the programming tiles and the robot using a pattern recognition algorithm. The only requirement is hence to have installed the free CaPL Android app [5] on the device. Apart from these two differences in terms of equipment used, the CaPL project proposes to create a simple DIY robot. This robot is made of cardboard and only contains a few electronic components such as two servos and a micro:bit [35]. It was named “Cardbot“. This idea comes from several cardboard-based robotics projects and educative robots already existing such as for instance KamiBot [27], BYOR [14] or Smartibot [42].

While Thymio VPL, TPL and PaPL activities can be adapted to children of various ages, the CaPL activities presented in this report tend to focus on children of early primary school (between 5 and 7 years old). In fact, the basic version of the Cardbot does not embed any sensors and it can be programmed using action tiles only. With a future version of the robot integrating sensors, more advanced exercises could be created. However, the CaPL and Cardbot concepts are not limited to any specific age. DIY is an interesting way to integrate robotics and programming at all ages. The activities presented in this report are a “Free Game“ activity, where the students can get used to the CaPL platform and can freely program the robot, and a “Geography Game“ activity, bringing into play a world map where the Cardbot has to be programmed to go from a certain location to another. In comparison with having only simple blocks, the use of a tablet allows to introduce feedback within the activities. For instance, as the robot moves, the current executed tile is highlighted by a red square on the screen of the tablet. In addition, some information about the iterations in ’for loops’ is displayed. As seen with the previous projects, the tangible CaPL games and activities may as well encourage students to communicate, to share their opinion and to compare alternative solutions. In order to stimulate collaboration and reflection in computer education, the CaPL “Geography Game“ proposes a system where the programming game can be solved by a group of children of various size and in which each of them has to give his opinion. This is realized by a token system. Each member

of the group receives a token and the contribution of the whole group is validated only if all the tokens are presented to the system. This is an idea of Christian Giang that combines tangibility [13] and collaboration [41, 29]. This feature is interesting to involve group members that might be more introverted or less experienced. Exactly as TPL and PaPL, CaPL can create a activity framework where groups of children can progress relatively on their own. This is something that teachers cannot easily implement with purely screen-based solutions.

In summary, the work carried out during this semester was to develop an innovative concept that combines a tangible interface with custom DIY programming tiles, a DIY robot and an Android app to make the connection between the instructions and the robot. My main contribution to this project was the elaboration of the CaPL mobile application based on computer-vision and allowing to send the identified commands to the robot. In comparison with previous projects, CaPL not only supports a tangible programming language, but it also involves the creation of a DIY robot. This contributes to promoting both group collaboration and individual learning. Figure 3 illustrates the different projects that have been conducted at EPFL in the domain of tangible programming languages so far and the robot they involve.



(a) (b) (c)

Figure 3: EPFL tangible programming language projects: TPL (with Thymio), PaPL (with Thymio) and CaPL (with Cardbot)

In the following parts of this report, the section 2 presents first the methods used to develop the CaPL system from the puzzle-like programming tiles to the DIY robot and the Android application activities. Then the results regarding the overall CaPL project and especially the design of the app are discussed in section 3. Finally, the key points and the contribution of this project to the development of tangible programming platforms is developed.

2 Methods

Since this project proposes a robot, an interface and some tasks, it can be seen as a complete educational robotics system (ERS) [15]. This chapter presents the methods used to set up each of these ERS key elements. The elaboration of the tangible tiles is first tackled. Then, the transcription of the sequence of tiles to the tablet is depicted. After that, the elaboration of the DIY robot will be pointed out. Finally, the combination of all these aspects of the project are pooled in the section 2.5 where the use of the Android app is documented.

2.1 Tangible tiles

As mentioned previously, one of the goal of this project was to develop a cheaper activity than the one proposed by the Blue-Bot educative robot made by TTS, a British company that provides various types of contents for primary and secondary schools levels. The final activity is intended for children between the ages of 5 and 7. The programming tiles are hence deeply inspired from the tangible interface that comes along this robot. In fact, the Blue-Bot has no sensors and the principle of its activity is to align a sequence of plastic tiles on a TacTile Reader that detects the movement instructions and can then send them to the robot via Bluetooth. Figure 4 depicts the Blue-Bot, a TacTile reader and some programming tiles.

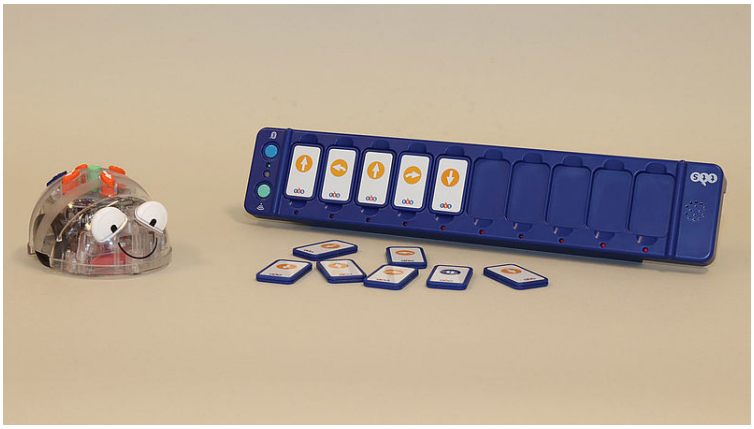


Figure 4: Blue-Bot, programming tiles and TacTile Code Reader [48]

The Tactile Reader of the Blue-Bot presents the limitation that we cannot program the robot to do more than 10 moves since it can contain 10 tiles at maximum. In order to deal with the limited number of commands the users can compose, the Blue-Bot also comes with an iOS [50] and Android [51] compatible app with which more than 10 commands can be sent. With this kind of setup, children can learn basics concepts of programming by creating and debugging simple programs. Based on this idea of the Blue-Bot activity, one of the goals of the current TPL project aims at providing a similar educational robotics system, but cheaper and accessible to any school. Moreover, the system will take advantage of the tablet to bring additional features such as real time informative feedback about the execution of the commands.

The combination of tiles had to be straightforward and to allow to put an indeterminate number of instructions in sequence. The material used to conceive the tiles had to be simple to cut, cheap, accessible but also robust. The advantage of the rigidity of the material should not only ensure the alignment of the tiles to allow the correct working of the computer vision algorithm proposed here (more information about this algorithm that has to detect the content of each tile is provided in section 2.2), but it also enables the tiles to be used several times and limit their degradation due to assembling/disassembling. Paper would have been interesting, but it is too weak to endure repeated long-term uses of the system. Instead of paper, the best solution was to consider cardboard. In fact, this material is stronger than simple paper, it is not expensive and it is present in most classrooms as well. In addition, considering the current confinement period, and since it does not require any specific tool, preparing prototypes of cardboard tiles was not an issue. It would have been much more complicated to print them using a 3D printer or to build them with wood using a laser cutter for example.

Having selected the materials, the design of the programming tiles was then addressed. In order to be able to detect the beginning and the end of a sequence of instructions of variable length using computer vision, the commands have to be organized between two delimiters: the first one indicating the beginning of the code and the other one pointing out its end. The code is situated in between these two extremities. Therefore, three types of tiles were defined and named “start tile“, “middle tile“ and “end tile“. As their name suggests, the “start tile“ defines the beginning of the sequence of instructions, the series of “middle tiles“ contains the representation of the code to be sent to the robot and the “end tile“ indicates the end of the sequence. Additionally, the two tiles at the extremities could be differentiated to attest the correct orientation of the photographed sequence. More precisely, the idea was to place a small black square on the top right-hand corner of the “end-tile“ in order to detect the correct orientation of the sequence of commands for the command extraction.

After having defined the types of tiles involved, a way to connect these parts of cardboard and to make sure that they stay aligned had to be found. The tiles could not have been left as simple rectangular parts and placed next to each other because potential misalignment could impact the correct working of the tiles recognition. On the contrary, the tiles have to fix to each other. In addition, the experience has to be reproducible. Hence it would not have been possible to tape the tiles nor to stick them together forever. The idea is to be able to clip them in the same way Andrea Mussati did in his project in 2019. The first version of the tiles is presented in figure 5.

Figure 5: Very first version of the programming tiles

Although the tiles could be assembled and clipped together, this design had to be simplified to be easily reproduced with a cutter. Iterative improvement of the design of the tiles finally replaced the curved elements by straight lines for easy cutting with a ruler and a Stanley knife. A black border following the contour of the sequence of commands was also added to facilitate the detection made by the computer vision algorithm. As discussed previously, a little black square was added on the "end tile" and serves as reference for the orientation of the sequence. The final proposed version of programming tiles looks as depicted in figure 6.

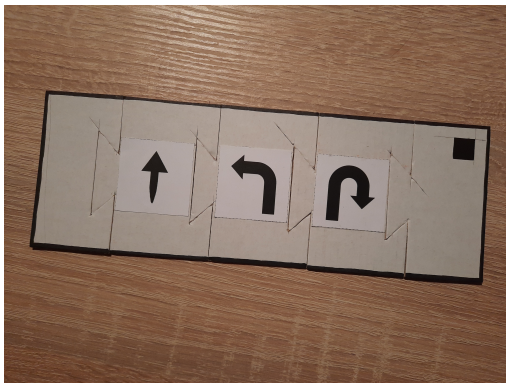
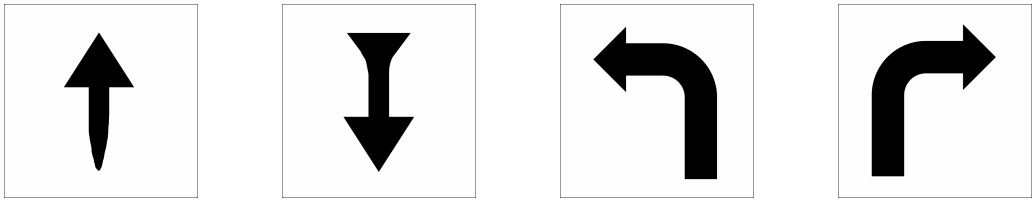


Figure 6: Final version of the programming tiles

Although some symbols can already be seen in figures 5 and 6, the final step in the conception of the tangible programming language was to define the different commands themselves. These are simple movements intended to be executed by the robot that are encoded by the template shapes of figure 7. These shapes are intended to be printed and then stuck on the "middle tile supports":



(a) (b) (c) (d)

Figure 7: “Go forward“, “go backward“, “turn left“ and “turn right“ shapes

Note that the “go backward“ shape is not exactly the flipped version of the “go forward“ shape. Since their contour would be quite similar if they had both a straight tail, these arrows present a different extremity. This was done on purpose to allow the computer vision algorithm to discriminate between the “go forward“ and “go backward“ instructions. In addition to the standard tiles of the Blue-Bot, a “turn back“ tile was added. This allows the robot to change direction using a single instruction that makes it rotate on itself by 180°. This shape can be seen in figure 8.

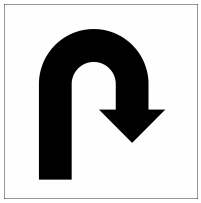


Figure 8: “Turn back“ shape

While it is true that the planned educational robot has no sensors and will only be able to execute action commands, ’for loops’ have been implemented in order to make programming more attractive. The beginning of a ’for loop’ is depicted with the symbol in figure 9.

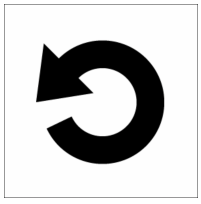


Figure 9: “Start repeat“ shape

This symbol is directly followed by a digit which indicates the number of times the ’for loop’ has to be repeated. Figure 10 represents the digits currently implemented in the CaPL system.

(a) (b) (c)

Figure 10: Digits shapes

Currently only the digits 2, 3 and 4 have been considered. By modifying the code at the locations where the digits are handled, it is possible to add additional digits. However, since the computer vision algorithm is conceived to detect shapes with strictly continuous contour (cf. section 2.2), it would not be possible to integrate double-digit numbers such as 10 and above. The core of the 'for loop' begins right after the digit. Finally, the shape indicating the end of a 'for loop' is based on the same kind of symbol as the one used to declare the beginning of a 'for loop'. It presents simply a diagonal bar crossing the symbol as depicted in figure 11.

Figure 11: "End repeat" shape

An interesting feature of this project is that the 'for loops' can be nested and the app indicates in which iteration of 'for loop' the robot is currently situated. This implementation can be useful for children to understand the mechanism of the 'for loops'. If a future version of the Cardbot will embed sensors, it would be possible to integrate additional tiles implementing event actions using for instance "if/else statements" and "while loops". For example, the robot could be asked what it is sensing at a specific point in the program and according to its measurement, a decision is drawn and the robot will either move one way or another. It would then exit the "if/else statement" and continue to run the rest of its linear program. In the same vein, "while loops" could be implemented. As long as the robot would measure a certain value, it would repeat some instructions. Reference [7] might be used by interested readers to find precise instructions about the construction of the tiles. As a complement to this manual, reference [11] proposes a video tutorial summarizing the main stages of the programming tiles construction.

2.2 Computer vision

The computer vision portion of this project was inspired from the Python code done by Julien Dedelley, Noé Duruz and Aditya Mehrotra in the context of the PaPL project [3] which was based on the computer vision library *OpenCV* [40]. The same approach was used and adapted to the new tiles. One major difference is that the previous project extracted the commands from a continuous stream of images, whereas this project interprets the commands from a single image. Since Android apps are based on Java, the Python code was translated to Java which also has an OpenCV library. However, since Python is a good prototyping language, a first version of the computer vision algorithm was developed using the intelligent IDE *PyCharm* [26] from *JetBrains*. Coding this algorithm from scratch was in fact more handy to fully understand, to debug and to optimize the computer vision steps. Once the computer vision algorithm was working in Python, the code was translated into Java using the IDE *IntelliJ IDEA* [25]. After that, once everything was running smoothly in Java, the code was slightly adapted so that it can run within the software *Android Studio* [22] and be installed on the CaPL app. Figure 12 summarizes the development steps of the computer vision algorithm.

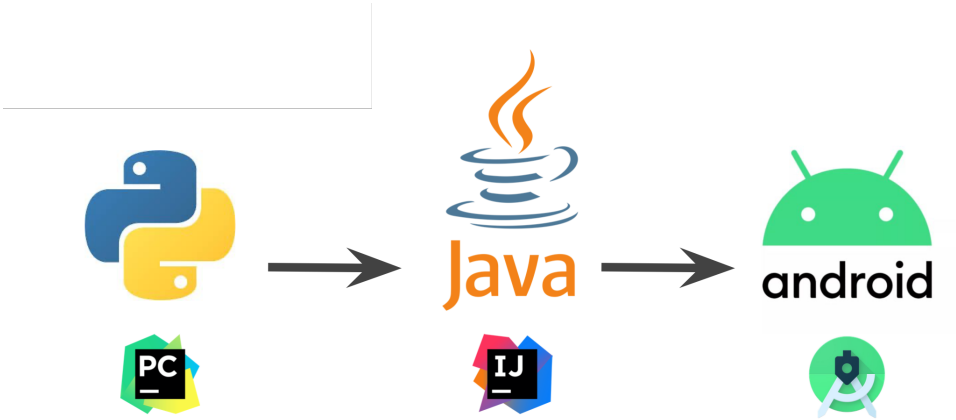


Figure 12: Development workflow of the computer vision algorithm

2.2.1 Algorithm

Algorithm 1 presents the main steps of the computer vision method used to identify the programming tiles:

Algorithm 1: Tiles recognition

Result: `commands`
Input: `photographed picture`

1. Pre-processing
   (a) Grayscale conversion
   (b) Median filtering
   (c) Binarization
   (d) Contours finding
   (e) Canny filtering the convex hull
   (f) Extracting the four corners
2. Rectification
3. Re-orientation
4. Template matching

The following lines of this section 2.2 illustrate the operations of the tiles recognition algorithm. Prior to analyzing the sequence of tiles, one first arranges the sequence of tiles as desired and then takes a picture of the setup. In order for the detection to work properly, it is recommended to have an empty space around the sequence of tiles (or at least no object presenting an area greater than the one of the whole sequence of tiles itself). Figure 13 depicts a good example of picture of programming tiles that can be treated by the computer vision algorithm.

Figure 13: Original image

The computer vision algorithm then processes the image when one presses on the *Identify Tiles* button of the "Free Game" or the "Geography Game" activity. This executes sequentially following steps on the input picture. These steps are illustrated in figure 14.

1. **Pre-processing**:
   First, various pre-processing steps are performed on the input image. This helps to find the four corners of the sequence of tiles using the *OpenCV* function `goodFeaturesToTrack`.

   (a) Grayscale conversion:
   The original image is converted to grayscale.

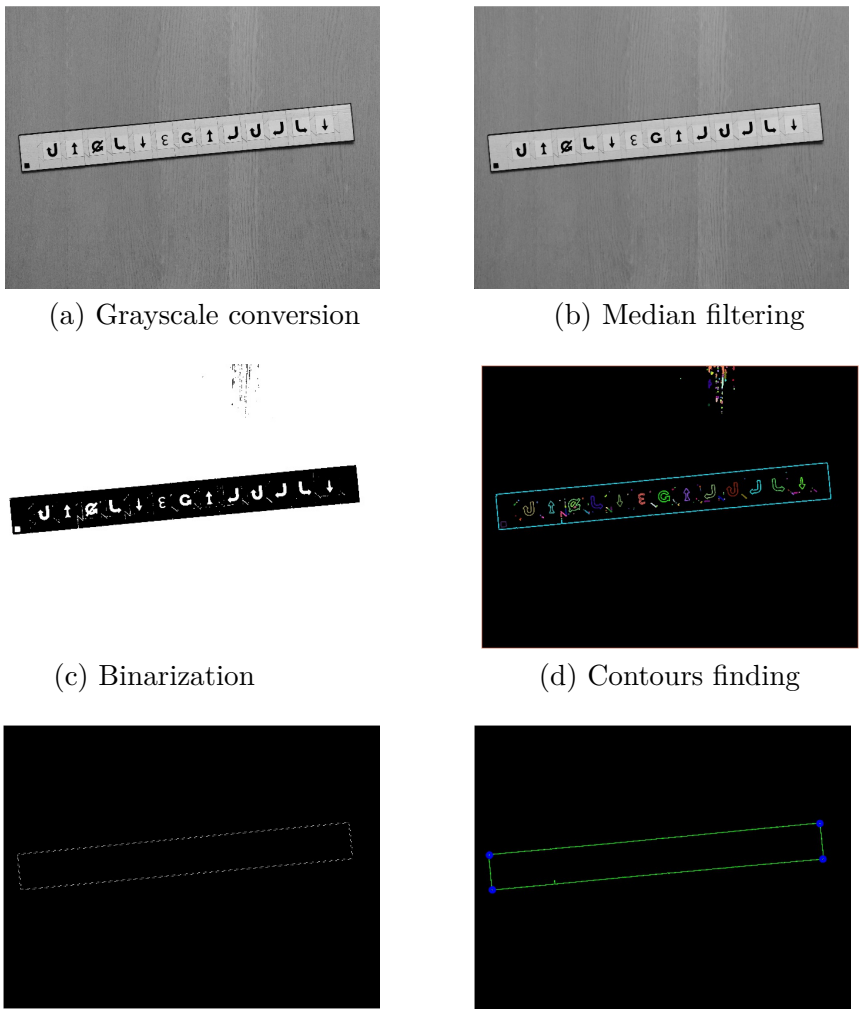
   (b) Median filtering:
   A median filter is applied on the grayscale image in order to remove noise and facilitate the next steps.

   (c) Binarization:
   The blurred image is then binarized using Otsu's thresholding method. This method automatically calculates a threshold value from the image histogram and finds the best location to binarize. The black border, which forms a frame around the entire sequence of tiles, facilitates the binarization by creating a clear demarcation between the center of the tiles and the background.

   (d) Contours finding:
   This step is about finding the contour of interest. The contour of the sequence of tiles is extracted from all the detected contours using the function `findContours`. This task can be solved by knowing that the contour of the sequence of tiles is the one with the largest surface area.

   (e) Canny filtering the convex hull:
   The solution found to facilitate the extraction of the four corners is to apply a Canny filter on the convex hull of the contour of the sequence of tiles. In fact, the convex hull allows to smoothen the contour and ensures good performance of the Canny filter which is important for the feature extraction in the next step.

   (f) Extracting the four corners:
   Applying the function `goodFeaturesToTrack` extracts finally the four corners of the rectangle automatically.



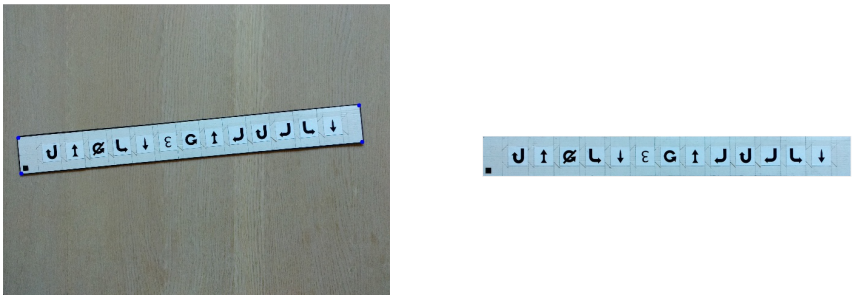
(a) Grayscale conversion (b) Median filtering

(c) Binarization (d) Contours finding

(e) Canny filtering the convex hull (f) Extracting the four corners

Figure 14: Pre-processing steps

2. **Rectification**:
   Now that the locations of the four corners have been detected, this information can be used to extract the picture of the sequence of tiles. From these four corners, it is in fact possible to rectify the sequence of tiles. Using the functions `getPerspectiveTransform` and `warpPerspective`, this step makes the sequence fit a horizontal rectangle shape and crop the borders. Figure 17 illustrates the rectification step of the computer vision procedure.



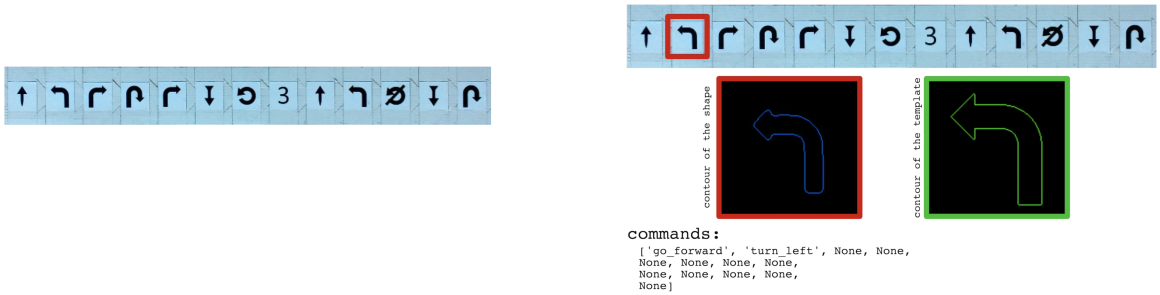
(a) (b)

Figure 15: Rectification

3. **Re-orienting the sequence**:
   To execute the program in the correct sequence, it is important to verify the presence of the black square on the end tile support. If this is not the case, the image has to be rotated by 180°. Figure 16 depicts the re-oriented sequence of commands.

Figure 16: Flipped

4. **Template matching**:
   The last step is to identify the shape on each tile. First of all, the "start tile" and the "end tile" are removed (see figure 17a). Then, each tile of the sequence of commands is identified one after the other, from left to right, and its contour is compared to the contour of the template shapes (see figure 17b). With reference to section 2.1, this is where the proper alignment of the shapes is necessary. In fact, the principle of this part of the algorithm is to move a sliding window horizontally to capture each shape individually. This is the reason why the symbols have to be centered in their respective "middle tile" supports and put in the middle of the height vertically. The contour which appears to be the closest to the contour of the current shape is considered to be the identified shape.



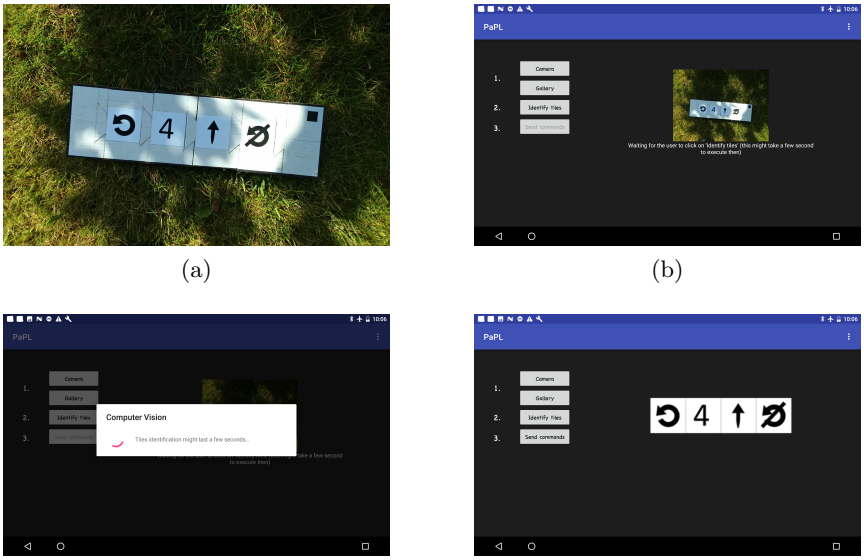
(a) (b)

Figure 17: Cropping and template matching

Finally, as an additional explanatory material, reference [10] presents a video that summarizes the above-mentioned steps of the computer vision approach. It is not a simulation, but a reconstruction of the process.

2.2.2 Advantages

In this subsection, positive points of the computer vision algorithm are described. Using the NVIDIA Shield tablets, it is possible to identify sequences of up to 19 tiles. Note that the maximum number of identified tiles depends not only on the resolution of the camera of the Android device , but also on the drift effect. This effect comes from the fact that a sliding window (i.e. the red square in figure 17b) is used to analyze each shape and implies that the more tiles are considered, the more the shift of the moving window across tiles can get important. This drift can lead to situations where the moving window gets centered between two shapes at some iteration (considering the last half of the preceding one and the first half of the next one). This makes the recognition of a shape impossible. One way to deal with this issue would be to have sequences of tiles that extend over several rows. This would allow to take a picture from closer and hence limit resolution drift issues. Figure 18 shows that the program also seems to be quite robust to changes in illumination as it has performed well with tiles laying in grass between shadow and sun.



(a) (b)

(c) (d)

Figure 18: Identification of a sequence laying on grass between shadow and sun

2.2.3 Limitations

Even if this algorithm can recognize the tiles from a good picture, like in figure 13 for instance (i.e. presenting the whole sequence of tiles from "start tile" to "end tile", with no surrounding patterns with a greater surface than the sequence of tiles, with sufficient sharpness, not too much shaded, etc.), it is not perfect and has certain disadvantages. First of all, the pre-processing is very time-consuming in comparison to the rest of the program. In fact, even if the remainder of the algorithm requires looping through the different commands, it executes rapidly in comparison to the pre-processing. This latter part of the program takes around half of the total time required to run the tiles identification. But this is a step we cannot leave out. Runtime tests have been conducted with sequences containing up to 19 "middle tiles". Interested readers can download these pictures from this link [9]. As shown in figure 19, the number of tiles in the sequence does not seem to influence the overall execution time. It can be observed however that executing the algorithm in Java is slower than in Python. Moreover, considering only Java programming language, the execution time on the tablet (NVIDIA Shield K1 2014, 2GB RAM, 2.2[GHz], Quad-core Cortex-A15) is almost twice as long as the time on a computer (MacBook Pro 2018, 16 GB RAM, processor 2.7[GHz], Quad-Core Intel i7).

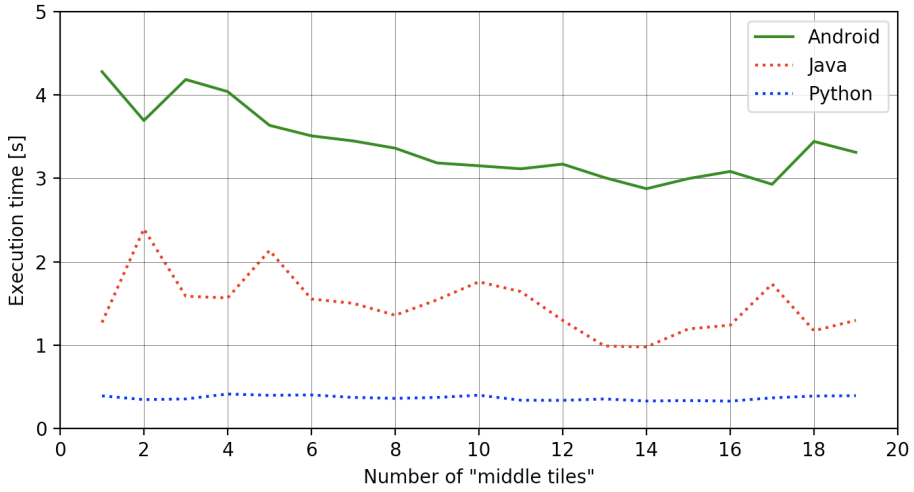


Figure 19: Runtime comparison of tiles identification for sequences of variable length

On average and independently of the number of tiles, the algorithm runs in approximately 0.3[s] in Python, 1.5[s] in Java on the computer and in around 3.5[s] in Java on the tablet. Since each tile needs to be analyzed individually, another limitation could have been that the implementation of the necessary 'for loop' slows down the overall process. But as observed in figure 19, the number of tiles does not seem however to affect the speed of the algorithm. Finally, another drawback is that the shapes need to be continuous. In fact, if they present a discontinuous contour (like it was the case with the first version of the "start repeat" shape, cf. figure 20), the program crashes. The reason is that the algorithm is made to compare one single shape contour with one single reference contour.

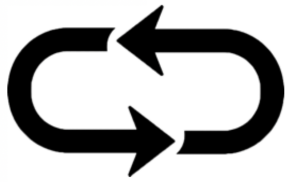
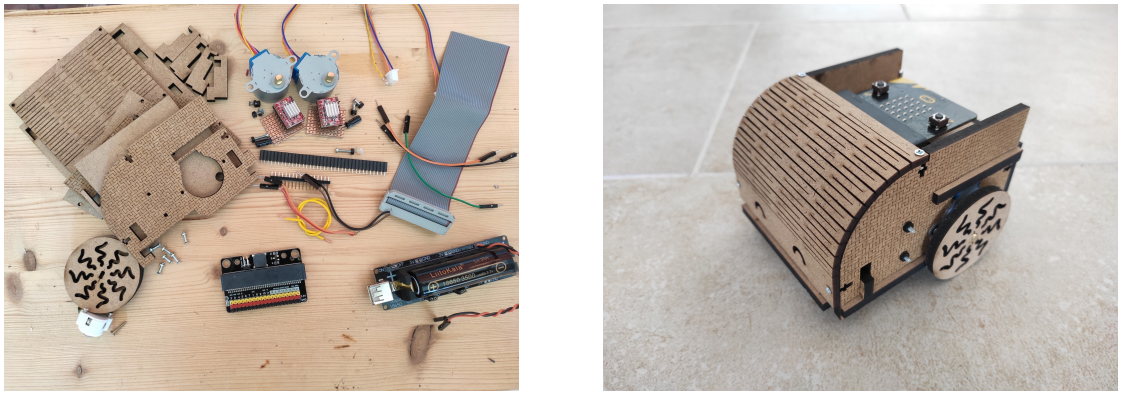


Figure 20: First version of the "start repeat" shape

2.3 Cardbot

Even if this part does not represent my main contribution to the CaPL project, I think it is an important milestone and I wanted to develop shortly the history of the development of this robot. At the beginning of this project, it was planned to use the robot developed by a group of researchers from the Azores Islands who were also interested in creating a cheaper and more accessible version of a Blue-Bot-like activity. However, due to the COVID-19 pandemic, an alternative had to be found. Still in contact with these motivated Portuguese researchers, the first idea was to build our own robot and discuss some design ideas with them. Similar to the Blue-Bot, the robot had to stay very simple. It is a two wheeled differential drive robot containing a basic microcontroller and two servos that actuate the wheels. As a base for the embedded technology allowing the control, a micro:bit [35] was chosen. A micro:bit card is a tiny computer made by the BBC that makes coding accessible

and promotes digital creativity. The advantage of the micro:bit is that it is a compact chip that can receive specific events from an Android device and execute the desired pre-programmed behaviours. In the same vein, an Arduino or a Raspberry Pi could have been used, but this is idea comes from our colleague Amaury Dame who already had some experience with micro:bits and recently developed a robot integrating such a device. We decided to continue using the micro:bit because it is an ideal learning tool since it comes with a few built-in sensors, it can be programmed using either Python, JavaScript or a block-based interface and it offers plenty of activities that do not require welding. Figure 21 represents mechanical and electronic parts and the mounted robot.



(a) (b)

Figure 21: Amaury's robot and some pieces he needed

The goal was to develop something similar but even simpler. A future version of this project could enhance the features of the robot with the addition of sensors. But at this early stage of the project, it had just to be able to be driven by the commands received from an Android device. As a base for the robot, a Bit:Buggy Car [17] (see figure 22) was hence bought at a Swiss retailer.

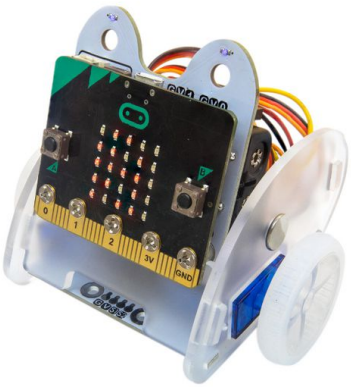


Figure 22: Bit:Buggy Car for micro:bit from Pi Supply

The drawback of the Bit:Buggy robot, however, is that it is made of plastic. It is thus difficult to reproduce a robot based on the pieces of the Bit:Buggy at home without any 3D printer. Since one of

goals of this project was to create a DIY system, it was decided that any plastic parts are to be avoided completely. That's the reason why our colleagues Aditya Mehrotra and Christian Giang modified the original plastic-based structure of the Bit:Buggy so that it works the same way but is mainly made of cardboard. In fact it is composed of just two folded pieces of cardboard. In order to limit the number of pieces (e.g. screws), and facilitate the construction of the robot, the servos are held in place by rubbing against the cardboard, plugged in two holes. As for the tiles, interested readers are invited to have a look at reference [2] for more information about the building instructions of this robot. In addition, reference [12] leads to a video tutorial presenting the stages of the Cardbot construction. Finally Aditya and Christian developed the "Cardbot" that looks as depicted in figure 23.

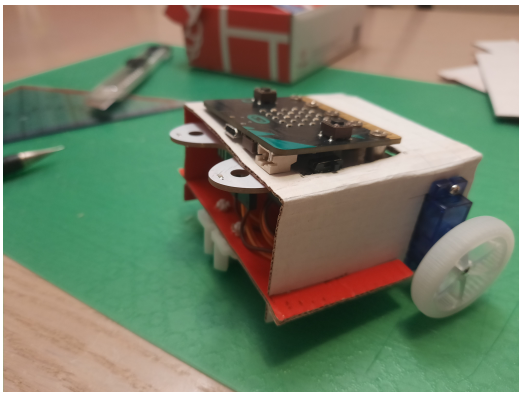


Figure 23: Cardbot elaborated by Aditya Mehrotra and Christian Giang based on the Bit:Buggy Car

2.4 Communication

Having created the tangible interface and designed the computer vision algorithm able to recognize the commands on the cardboard tiles, the next step was to set up the communication between the robot and the tablet to send the commands. To this end, the built-in Bluetooth module of the micro:bit was used to establish the communication with the tablet.

Bluetooth is a wireless communication technology that allows devices to communicate with each other without the help of a third-party access point. There exists two types of Bluetooth: Bluetooth Classic and Bluetooth Smart. The latter is a more recent version of Bluetooth from the end of 2010 that is also called Bluetooth Low Energy (BLE). With the arrival of connected objects and the Internet of Things, the purpose of BLE was above all to considerably reduce the power consumption of both the emitter and the receiver in comparison with the original version of the Bluetooth. Deep understanding of the working of Bluetooth won't be further developed in this report. Concisely, a device such as a smartphone or a tablet can interact with a micro:bit via a client/server architecture. An Android app that uses the bluetooth service on the phone is the client and the micro:bit is the server. They communicate thanks to a communication protocol called Attribute Protocol (ATT). Figure 24 shows the principle of the client/server architecture between an Android device and a micro:bit. As the goal of this project is to implement an Android app that is compatible with micro:bits, the built-in Android APIs handle the ATT protocol.

Figure 24: Client/server architecture between an Android device and a micro:bit

Several Android apps exist that make use of the API and integrate micro:bit-compatible activities like sending string through UART protocol, commanding LEDs, controlling the buttons, etc. To integrate the communication with the micro:bit to the CaPL app, a solution was to merge one of these apps with the CaPL app itself to make use of the portion of code implementing the BLE connection. Here are a few solutions that were considered:

- Configuring a bluetooth option with a simple app created with the AppInventor online tool of the MIT [33]. This alternative was rejected as it did not offer compatibility between AppInventor and Android Studio.
- UART communication with the *Bluefruit Connect* app from Adafruit [1] (source files available on GitHub). Although this source code allows to establish a connection between an iOS device and a micro:bit, this option has not been further developed since it was not up to date for Android devices.
- Official *micro:bit* app from Samsung [44] (source code is available on GitHub). Since this app can only send hex files through BLE, this would not have helped to do something interesting. In fact, the *micro:bit* Samsung app can only be used to send a whole program (under the form of a hex file) to a micro:bit over BLE. The main drawback of this method is that sending a complete code through BLE takes a long time. It takes about 20[s] to load a single custom program onto the micro:bit. This is definitely too slow. An acceptable time would be maximum 2 to 3 seconds to send a sequence of commands. A much quicker solution was needed, something responsive that can be directly executed by the robot as soon as the signal is received.

All this research has finally led to the discovery of a possibility to control the LED matrix of a micro:bit using the *micro:bit Blue* Android app [32] (see figure 25) made by Martin Woolley, a senior British developer and Bluetooth expert.

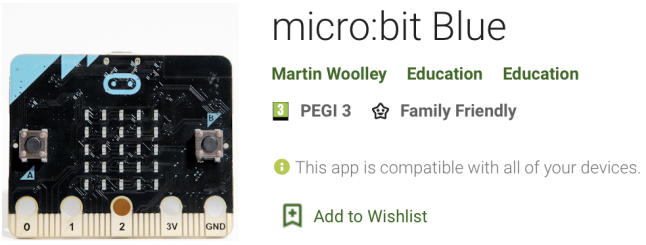
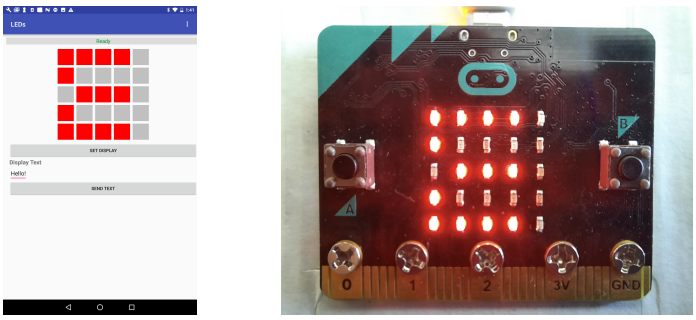


Figure 25: *micro:bit Blue* Android app created by Martin Woolley

The compatible open source code of the *micro:bit Blue* app was then integrated to the CaPL app. Therefore, after standard pairing with the micro:bit (see "Help" menu under "Device List" in the CaPL app), basic instructions could be sent to the micro:bit to display specific shapes on its LED matrix as depicted in figure 26.



(a) (b)

Figure 26: Controlling the LED display of a micro:bit from an Android tablet

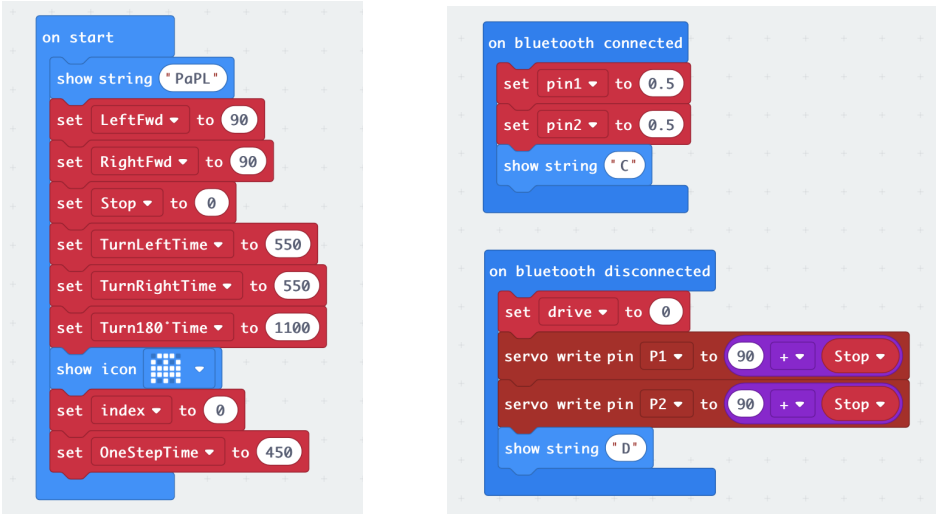
This was the most promising solution to get the bot to move according to the photographed commands. On the other hand, the required program to execute the movements was here flashed directly on the micro:bit. This way no new hex file had to be loaded on the micro:bit each time a sequence of commands had to be executed. But this implies that the hex file of the micro:bit should contain all the possible movements the robot might be asked to do beforehand. With the micro:bit online MakeCode editor [36], a block-based code containing the 5 basic motions of the robot was created. Each of these moves is linked to a specific event. Once the micro:bit receives a particular event from the tablet, the execution of the corresponding movement is triggered. In fact, the *micro:bit Blue* app was initially programmed to send such kind of events to a micro:bit. The CaPL app was then coded in such a way that the robot behaves as expected when a specific event is received from the Android device.

To sum up, the commands received under the form of strings from the computer vision will be converted one after the other to simple movements to be executed by the robot. Practically, the block-based code is as depicted in figure 27. This code is then automatically translated into JavaScript and the online Microsoft MakeCode compiler produces a hex file that can then be loaded on the micro:bit. This hex file can be downloaded from this link [6].

Figure 27: Block-based code installed on the micro:bit of the Cardbot

The basic functionalities of the CaPL activity (which are to make the robot move either forward, backward, turn left, turn right or turn back) appear on the left-hand side of figure 27. The code composing the Easter Egg (see section 2.5.2.3) is depicted on the right-hand side of figure 27. The implementation is straightforward and is explained in the next few lines. Figure 28a shows that as soon as the micro:bit is turned on, powered by the battery shield of the robot, the variables used to calibrate the movements of the Cardbot are initialized. In fact, since the servos are unique and do not execute exactly the same kind of movements while receiving the same command, they have to be calibrated. This is done by testing different values for the `LeftFwd` and `RightFwd` variables. The

purpose is to make the wheels move as synchronized as possible while the robot is moving forward and backward. The values of the variables `LeftFwd` and `RightFwd` vary from 0 (min) to 90 (max), but they should normally not be far from 0. On figure 28b, basic actions regarding the moment the micro:bit gets connected and disconnected via Bluetooth are represented. It can be seen that when the micro:bit is connected via Bluetooth to a portable device, the pins to which the servos are connected are set to a value making the robot stay in place. Similarly, as soon as the connection with the client is lost, the servos are stopped.



(a) (b)

Figure 28: Core blocks of the micro:bit block-based code

The rest of the code is composed of five blocks similar to the one presented in figure 29. This figure describes the fact that as soon as the micro:bit receives the event “MES_DPAD_CONTROLLER_ID“ carrying the value “MES_DPAD_BUTTON_1_DOWN“, it has to display a forward arrow on its LED matrix and then set the value of its pins 1 and 2 (the two pins to which are connected the servos) to a specific calibrated value making it move straight forward for `OneStepTime` [ms]. This latter value can also be adjusted and allows the Cardbot to stop after a specific time. This time defines the displacement unit of the robot.

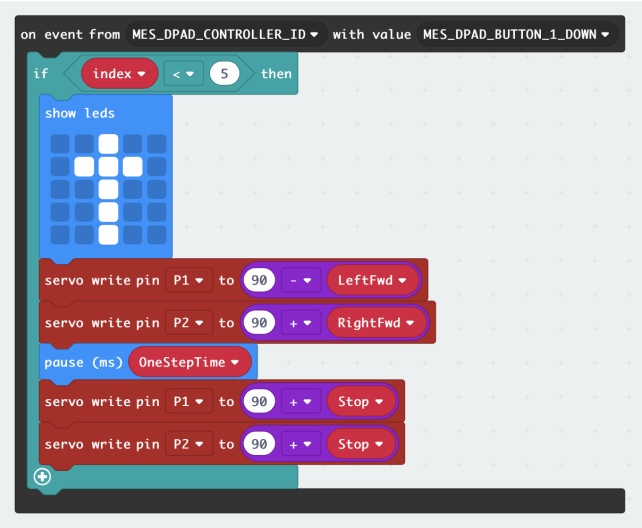


Figure 29: Block-based code illustrating the actions driven by the micro:bit once a specific event is received

2.5 Android app

As stated in the introduction, the aim of this project is finally to unify all different aspects that have been described so far. This is done by setting up different activities on an Android platform. This makes the CaPL Android app the core of this project. It is here that each part meet and work together. The app is designed to work on an NVIDIA Shield Tablet K1 running Android version 7.0 and allows the tablet to make the link between the arranged cardboard tiles and the robot. It integrates the shapes of the tiles, sends the corresponding code to the Cardbot via Bluetooth and provides some real time feedback about the movement of the robot.

2.5.1 App Workflow

Before going into the details of the working of the app, it is important to give a brief overview of its architecture. In the language of Android development, an activity corresponds basically to a specific screen displayed by the device. The CaPL app is composed of 8 activities:

- `SplashScreenActivity`
- `MainActivity`
- `DeviceListActivity`
- `MenuActivity`
- `FreeGameActivity`
- `GeographyActivity`
- `DeviceInformationActivity`
- `GamePadControllerActivity`

Note that the `SplashScreenActivity` serves simply as an introduction to the app and provides only a transition screen as the CaPL app is launched. It presents an `ImageView` with the Cardbot. Figure 30 illustrates the overall workflow of the CaPL app.

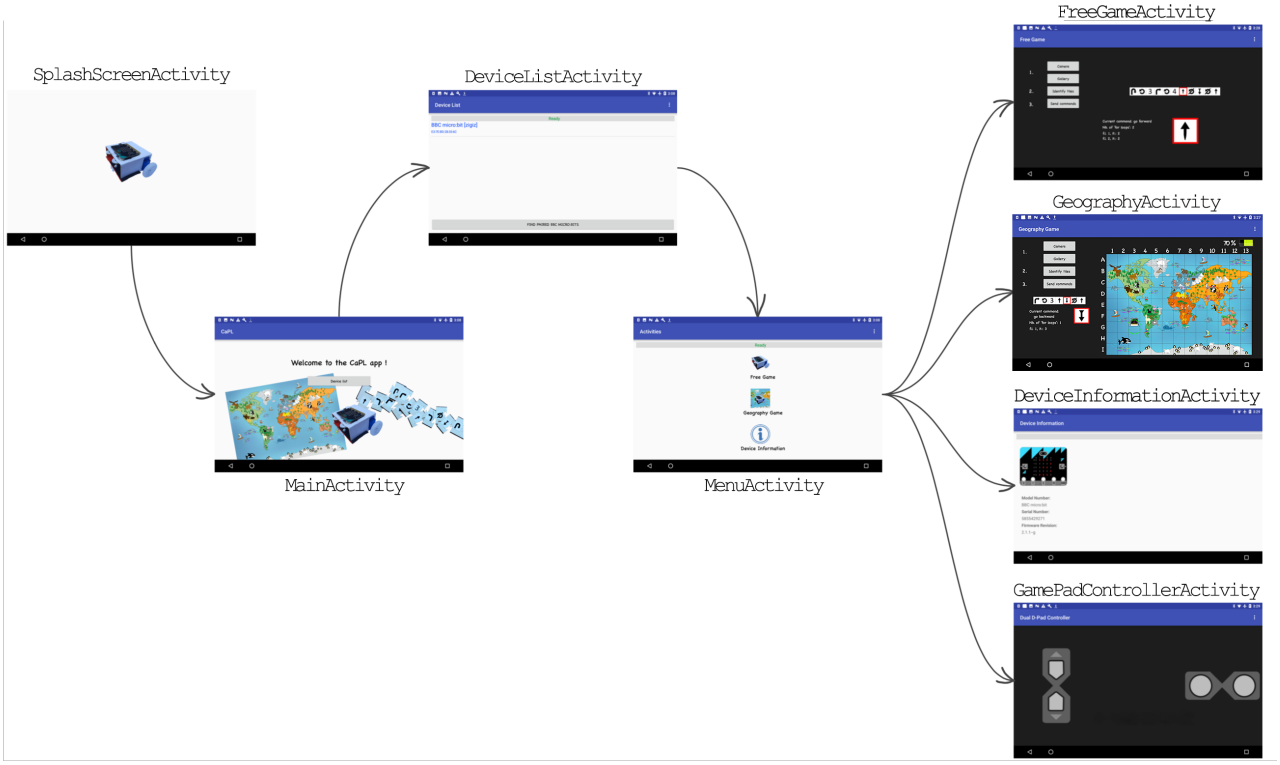


Figure 30: CaPL app workflow

As it can be seen on figure 30, `MainActivity` is the first fixed screen the users see right after launching the app. It presents some pictures illustrating the CaPL environment and allows to lead to the `DeviceListActivity`. Once the users press on the button "FIND PAIRED MICRO:BIT(S)", the latter activity sets the Android in scan mode in order to scan for in range advertising micro:bits. Once the micro:bit of the Cardbot is detected, the users select its name on the list and the connection between the tablet and the micro:bit is established. The next activity to show up is finally the `MenuActivity`. This is in fact the menu from which the actual Cardbot activities (and not only the "activities" in the Android development sense) can be selected. The proposed options are the `FreeGameActivity`, the `GeographyActivity`, the `DeviceInformationActivity` and the hidden activity `GamePadControllerActivity`. The `DeviceInformationActivity` allows the app to obtain information regarding the current connected micro:bit such as firmware version details. In the next sections, the activities of interest, which are the `FreeGameActivity`, the `GeographyActivity` and the hidden `GamePadControllerActivity`, will be developed and explained in more detail.

2.5.2 Activities

The first of the activities of interest is called “Free Game“. This program provides the features to load a tangible sequence of tiles captured by the camera of the Android device, to interpret these commands and to finally send them to the Cardbot. It allows to freely discover the working of the cardboard-based tangible programming of a Cardbot. This serves as a base framework for the second, more elaborated activity called “Geography Game“. This activity sets up a world map environment where the users are asked to make the robot evolve from a start point to a goal. In addition to simply compose a program of movements, the robot might stop on its journey to the destination. Here a MCQ screen pops up and asks a question regarding the specific location at which the robot currently lays. This activity is intended for advanced users that have already discovered the basic functionalities of the CaPL app and the “Free Game“ activity. Additionally, it is programmed in such a way that several users can participate in the game. The last activity presented in this report is a hidden activity that allows the users to freely control the Cardbot with a gamepad. The next few subsections present the different aspects of each of these activities, how they are conceived and what they could bring for educational purposes.

2.5.2.1 Free Game

As explained in the introduction of this subsection, the `FreeGameActivity` implements the basic control of the Cardbot using the tangible programming tiles. It is divided in 3 steps. Figure 31 shows the startup screen of the `FreeGameActivity`.

Figure 31: Initial screen of the `FreeGameActivity`

First, one has to provide a picture of a composition of tangible commands to the activity. This is done either by taking a picture of the arrangement of tiles (respecting the fact that the sequence of tiles appears entirely and that the background should not contain any other big object, like in figure 13 for instance) or by loading a picture of a sequence of tiles from the gallery. Sample images can be downloaded by following this link [8]. Figure 32 presents the screen of the `FreeGameActivity` once a picture has been loaded.

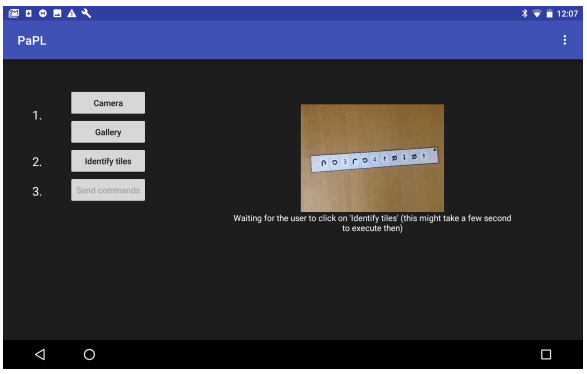
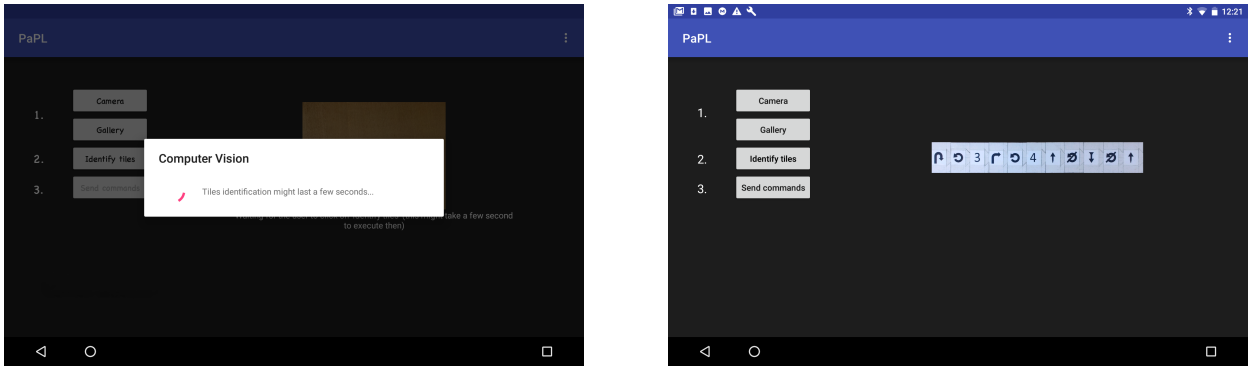


Figure 32: Step 1 of the `FreeGameActivity` - loading a picture

As depicted in figure 33, the computer vision algorithm can be then launched to identify the commands by pressing the "Identify tiles" button. The tiles identification procedure then lasts a few seconds.



(a) (b)

Figure 33: Step 2 of the `FreeGameActivity` - performing tiles recognition

So far, the commands are identified but not yet interpreted. First of all, the code is analyzed and the activity checks if it is coherent. In case a mistake is detected, the app stops the process and pops up a message to the users telling them what is wrong in their code or at least provides them some indications about where the error could come from. For instance, if a 'for loop' is not correctly built (for example with the digit for the repetition appearing at another place than directly after the command "start repeat"), a specific error message will be displayed and the commands cannot be sent to the robot. This is the kind of feedback that can be provided with a tablet in comparison with having simply blocks. Once the code is corrected and composed correctly, there are then two possible cases. On the one hand, where the sequence of commands does not include any 'for loops', the series of movements to send to the robot does not need any specific processing and can be directly sent. But on the other hand, if the sequence includes 'for loops', it has to be interpreted to deploy the whole final series of movements to be executed by the Cardbot. This is done by replicating the commands situated inside 'for loops' the correct number of times (specified by the digit at the beginning of the 'for loop'). The location, in the sequence, of the end and beginning of of each 'for loop' is hence needed. The lists `end_repeat_idx_list` and `numbers_idx_list` are used to temporarily store these

key locations in the sequence. This process is illustrated with algorithm 2:

Algorithm 2: Interpret commands

Result: `interpreted_commands`
`interpreted_commands` = `commands`;
while *`end_repeat_idx_list` not empty* **do**
  `end_for_loop_temp` = min(`end_repeat_idx_list`);
  `begin_for_loop_temp` = max(`numbers_idx_list` smaller than `end_for_loop_temp`) + 1;
  `current_for_loop_commands` = `interpreted_commands`[`begin_for_loop_temp`, `end_for_loop_temp`];
  replicate `current_for_loop_commands` `repetition` times;
  concatenate `current_for_loop_commands` inside `interpreted_commands`;
  remove last element of `end_repeat_idx_list`;
  remove element with value `begin_for_loop_temp` from `numbers_idx_list`;
end

According to algorithm 2, the 'for loops' are considered one after the other automatically beginning from the ones ending the first (i.e. presenting a "stop repeat" symbol situated the most on the left). Figure 34 illustrates on which basis the two key lists `end_repeat_idx_list` and `numbers_idx_list` are organized according to following sequence of commands:

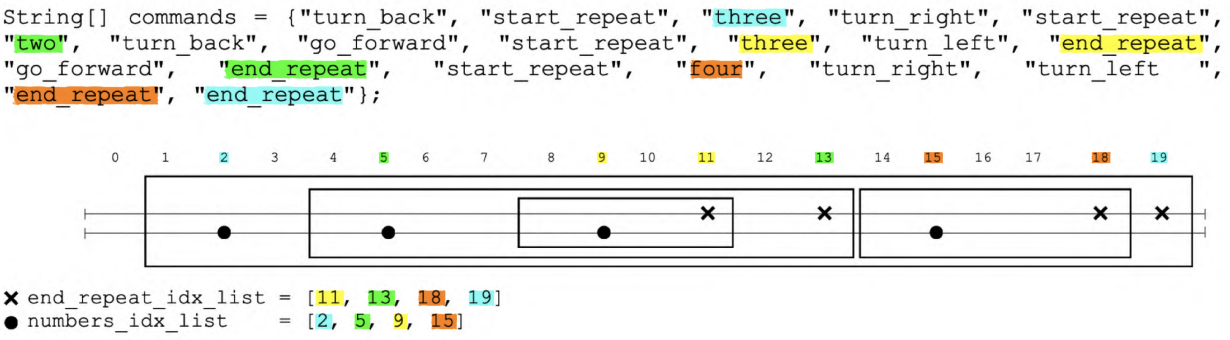


Figure 34: Illustration of `end_repeat_idx_list` and `numbers_idx_list` for the case of a sequence including 4 nested 'for loops'. An identical color is used for the digit and "end repeat" index of each loop

Once the tangible code is interpreted, the third and last step is to send the final instructions to the Cardbot via Bluetooth. Here, the commands are sent one by one under the form of an event carrying a specific value that corresponds to the movement to execute. As the robot moves, the current executed tile is highlighted by a red square and is even printed in large format below the sequence. In addition, the activity provides some information regarding the number of 'for loops' in the code and which iteration of each 'for loop' the robot is currently executing. As explained above, the 'for loops' are ordered according to the location of their "end repeat" tile in `end_repeat_idx_list`. Hence, the first 'for loop', "f.l. 1", is for instance the smallest 'for loop' in the middle of the sequence in figure 34. In figure 35 ("f.l. 1 it: 3, f.l. 2 it.: 1"), it can be deduced that the robot is currently running the 3rd

iteration of the first 'for loop' (i.e. the nested one, "f.l. 1 it: 3") and the 1st iteration of the second 'for loop' (i.e. the big outer one, "f.l. 2 it.: 1"). There is no limit to the number of 'for loops' and the program supports sequences containing 'for loops' nested at will. The feature of the nested 'for loops' definitely brings an additional dimension in comparison with what is currently available with other educational robotics system like the Blue-Bot and its activities for instance.

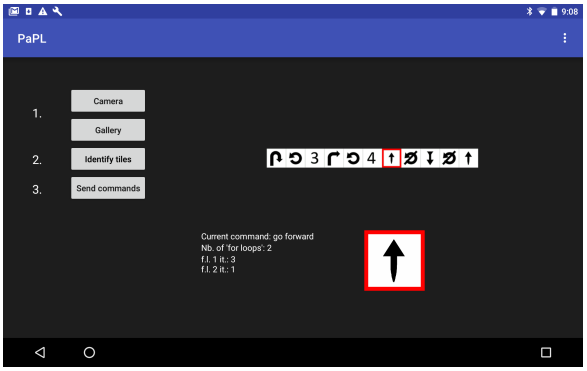


Figure 35: Step 3 of the `FreeGameActivity` - sending the commands to the robot

By going through the whole process of this activity, it allows the users to understand each of the steps required to finally make the robot move. Furthermore, it is not possible to miss a stage because the buttons are disabled as long as the previous required steps are not completed. The overall workflow of the `FreeGameActivity` is summarized in figure 36.

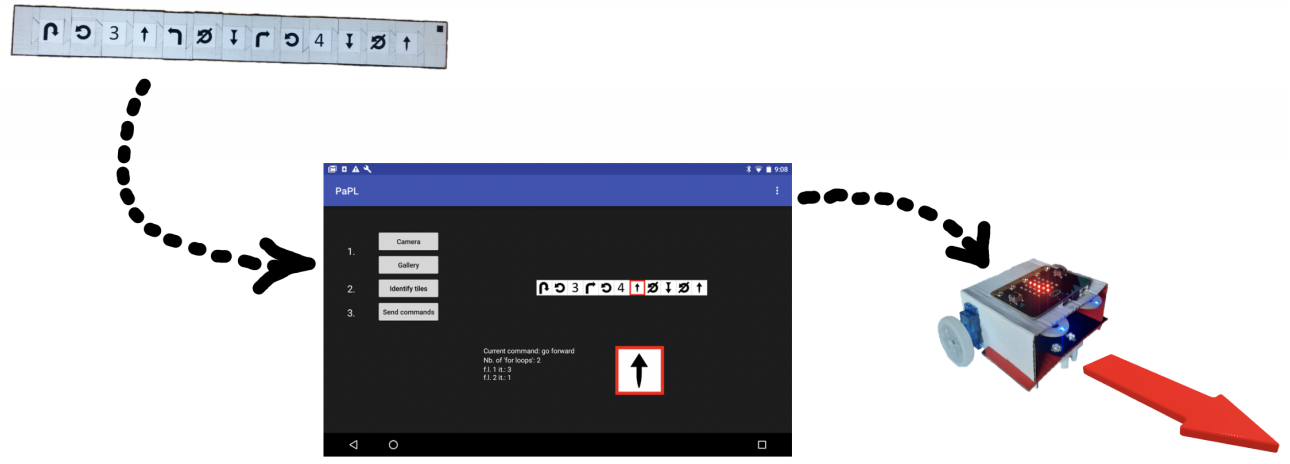
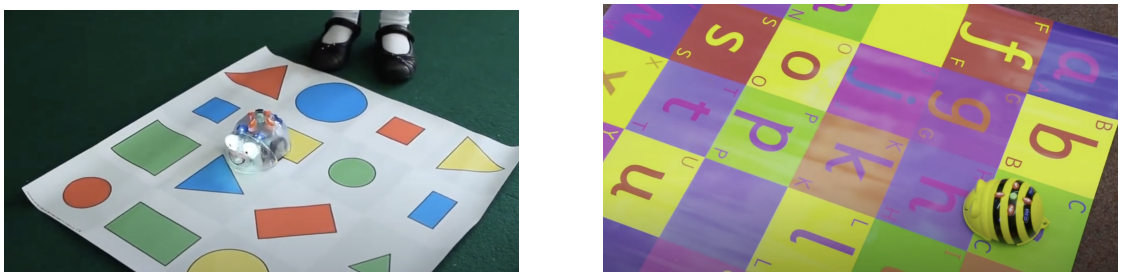


Figure 36: Workflow implemented in the `FreeGameActivity`

Some ideas of improvements to bring to this activity could be for instance adding some additional features such as a debugger option allowing the users to run the sequence of commands step by step. It would also be an interesting add-on to integrate the possibility to program the robot within the app itself with a drag-and-drop system. This would allow the robot to be programmed using virtual tiles directly on the screen of the tablet as it is done with VPL [24] to program a Thymio.

2.5.2.2 Geography Game

Once the basic framework described in previous section 2.5.2.1 is built, this CaPL educational robotics system can extend to a game-like learning activity. There are many activities that can be imagined. One of them could be to define a grid of basic shapes of various colours and ask the children to compose a program that makes the robot move for instance from a red square to a blue triangle [52] as it is represented in figure 37a. This kind of exercise can help the children train their object recognition and classification skills. Another activity could be a game involving letters recognition [49]. Here the children would have to program the robot to move between letters and form words. This kind of activities is depicted in figure 37b.



(a) (b)

Figure 37: Shapes classification [52] and letters recognition [49] activities using Blue-Bot

As briefly described at the beginning of this chapter 2.5.2, the activity proposed here is a "Geography Game" activity. Given some instructions, the main task of this activity would be to go from a start to a destination place (identified by animals for instance) on a world map with monuments, trees and typical animals. To provide feedback, this printed world map is also presented digitally on the app and a fictive representation of the robot reproduces the movement of the Cardbot in real time. The Cardbot has to be programmed in such a way that it should move on the map in the most efficient way making use of a fictive amount of energy. Knowing that each step of the robot costs some energy, the users would have to compose a trajectory avoiding water (e.g. which costs twice as much as terrain). This notion of cost and the idea of path planning comes from a paper written by scientists from the CHILI lab of EPFL [23]. Additionally, the users will then come across a series of multiple choice questions that pop up at random along the way and which are relative to the current position of the robot on the map. These questions may appear multiple times until the destination is reached. For example, the following question could appear when passing by location C4 on the map presented in figure 38:

Figure 38: World map [30] of the `GeographyActivity`

What is the name of the country situated at C4?

- ☐ USA.
- ☐ Italy.
- ☐ China.

The fictive battery level of the robot is limited to a certain amount of energy and each positive response drives it up while each incorrect response drives it down. Here are some examples of topics related to countries for the MCQ that could be addressed to younger children:

- Flag of country
- Typical animal
- Language
- Etc.

And for older children, this kind of questions could be asked:

- Populations
- Currency
- Mountain chains
- Rivers
- Typical monument
- Typical meal
- Land area
- Famous monuments

- Seas
- Etc.

The “Geography Game“ aims not only at teaching children basics concept of robotics while learning aspects of geography, but it is designed to be a collaborative activity. In fact, at the beginning of the game, it will be asked to enter the number of players and each of them will receive a tangible black token. This is an idea of Christian Giang that combines tangibility [13] and collaboration [41, 29]. The token will serve as an identification badge at the moment a question has to be answered. In fact, each member will have to approve the response. This is assessed by taking a picture of the present tokens after a response to the MCQ has been clicked. The number of tokens is identified by means of watershed segmentation [39]. Figure 39 depicts typical tokens used in the “Geography Game“.

Figure 39: Example of tokens used in the “Geography Game“

If a token is missing while the picture is taken, the question is assumed to be responded by only some members of the group and a penalty is applied (by decreasing the battery level). If however the amount of tokens corresponds to the amount of players in the group, the question can be normally evaluated (either it is correct and the battery level increases or it is evaluated as incorrect and the battery level decreases). In the end the group of players can simply evaluate his result by taking note of his residual battery level. The performance of each group could finally be compared across the whole class. The use of an Android tablet for this activity is convenient since it allows to integrate some ideas of feedback (like the current command executed by the robot and its location on the map), interaction (the children will interact with the printed map, the tangible coding tiles and the CaPL robot) and collaboration (considering the fact that each children has to participate and give its opinion in order to hope for an answer to be evaluated correctly and to earn some points for the group). The idea of the tokens force collaboration. Since children usually don’t naturally collaborate, here, in order to win the game, they will have no other choice than discussing about their choice and verbalize their ideas. In this sense, the “Geography Game“ imposes the children to start sharing and communicating their ideas. Figure 40 shows the layout of the “Geography Game“. On this picture the virtual Cardbot is executing a “go backward“ movement. The starting point is depicted in green and the destination in red.

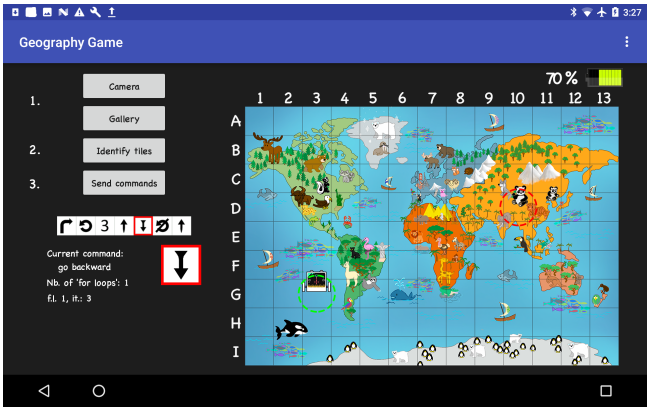


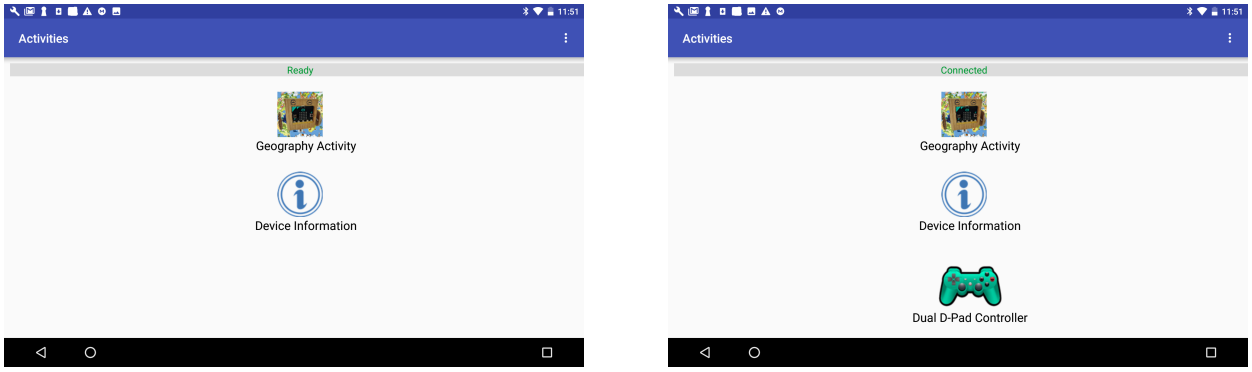
Figure 40: Layout of the "Geography Game"

An improvement that can be made to this game would be to visualize the remaining fictive battery level of the robot on the LED matrix of the real Cardbot. This idea comes from a paper written by some researchers from the CHILI lab of EPFL [23]. In their experiment with the Cellulo City activity, they used the 6 RGB LEDs on top of the robot to physically show the level of the fictive battery. Regarding the set of MCQ, an additional interesting feature would be that the teachers could customize the database of questions by attributing some specific questions to specific locations on the map. This can be done by manually changing the questions in the source code of the `GeographyActivity`, but the idea would be to create an interface allowing the teachers to easily change the questions. By reviewing the task and the questions, this "Geography Game" activity could be adapted to both young and less young children. Regarding for instance the task, younger children could be asked to make the robot travel from one specific animal to another, whereas older children could be asked to make it move from a certain country to another.

2.5.2.3 Easter Egg

It would not be fun to have done all this work without finally being able to freely control the micro:bit-based robot. This is the reason why the app hides a secret activity permitting the users to take control of the Cardbot and pilot it as a regular remote-controlled toy car. In order to access the hidden `GamepadControllerActivity`, there are two steps to follow:

1. The users have simply to click five times on the white space right under the icon leading to "Device Information". This will enable the access to the hidden gamepad controller as shown in figure 41.



(a) (b)

Figure 41: Access to the hidden Activity `GamepadControllerActivity` after having pressed 5 times under "Device Information"

2. On the side of the Cardbot, one has to press five times on the "A" button. Consecutive digits will appear on the LED matrix of the micro:bit and once the skull symbol is displayed (as depicted in figure 42), this indicates that the micro:bit has switched to another interpretation of the received events from the tablet.

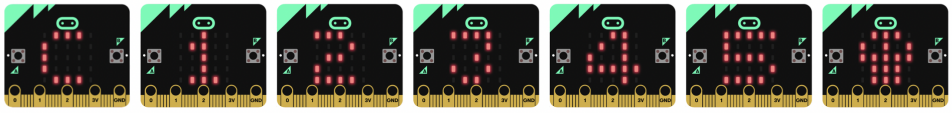


Figure 42: Successive symbols displayed on the LED matrix of the micro:bit in order to set it up to use the hidden control mode

With this new control mode, one can make use of the gamepad illustrated in figure 43 to drive the robot as he wants! The control works as follows: the up and down buttons of the left pad make the robot move forward and backward while the right and left buttons of the right pad make the robot turn left and right.

Figure 43: Screen view of the `GamePadControllerActivity`

In order to go back to the original mode making the robot move unit by unit like it is implemented in the “Free Game“ and the “Geography Game“, only the micro:bit has to be reset to its original behaviour. One can simply do that by clicking simultaneously on the “A“ and “B“ buttons of the micro:bit. In fact, this will reset to ’0’ the `index` that allows to switch the control mode.

3 Results

Originally, it was planned to test this educational robotics system with EPFL students during the demo sessions of the Robotics Practicals MICRO-453 course around mid-May. The collected data would have helped to get some feedback about the overall project, i.e. from the construction of the Cardbot to the use of the application and the the programming with tangible tiles.

Unfortunately, as the school closed in mid-March due to the COVID-19 pandemic, it would have been difficult to conduct an actual study about this project in classrooms like it has been done with the previous TPL projects. Thankfully, it was possible to organize a study with five teachers from Ticino and one from Italy. In the context of an educative robotics course, they were invited to test the Cardbot project and various related activities. The plan was to prepare six kits containing all the required equipment and information needed to test the project and additional activities around the micro:bit device. These kits have been then sent via post directly at their home. The participants are primary and lower secondary teachers that already have some experience with educational robotics systems (such as Thymio [31], Scratch [37] and Blockly [38]). They discovered the CaPL project and its related activities as a mini project of a few hours that they did autonomously at home with the help of manuals and video tutorials. In addition, they were invited to answer some research questions during an interview by teleconference. They were also asked if the making activity was new to them and if they would consider continuing to use Cardbot and micro:bit in their class.

Meanwhile, an internal review of the whole educational robotics system was conducted with members of the MOBOTS group. This was an important step that allowed to get additional feedback on the project before finalizing the protocol of the activities planned to be tested by the Italian-speaking teachers. The final procedure they had to follow was to first create their own micro:bit-based Cardbot, then to build its cardboard programming tiles and finally to use them for solving different easy tasks by mean of the "Free Game" Activity. In spite of the period of confinement it was hence possible to test the system. But, in comparison with what has been done during the previous semesters, this was a restricted study. Hopefully this project will continue and with the resumption of normal public life, it will certainly be soon possible to meet even small groups of schoolchildren to conduct some experiments.

Unfortunately the results of the study with the teachers from Ticino and Italy were not available before the submission of this report and it was not possible to analyze their feedback. However a self-reflecting heuristic evaluation [15] of the system has been performed. Using this kind of heuristic is a good way for developers to identify sometimes unobvious limitations and weaknesses of their educational robotics system. Moreover, it represents a resource-efficient alternative to costly and time-consuming user studies. It can either be conducted by external parties who have tested the system or by the developer itself. I am going to present here the evaluation of the CaPL educational robotics system (ERS). As a remainder, ERS is "a model that describes the combination of the educational robot, the programming/interaction interface and the tasks presented for classroom activities" [15]. The schematic representation of the CaPL educational robotics system is depicted in figure 44.

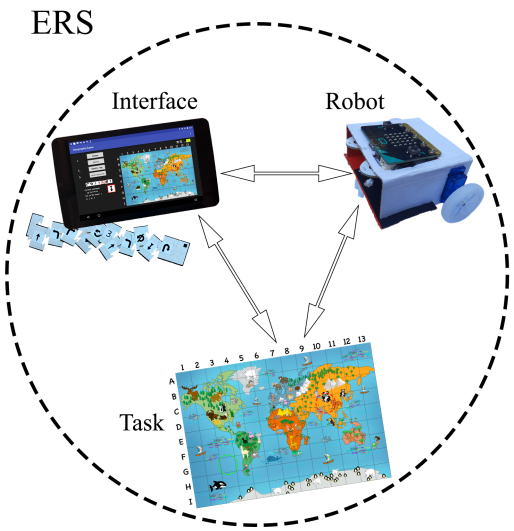


Figure 44: Schematic representation of the CaPL educational robotics system

The usability issues of an ERS are defined according to 7 design principles [15]. These evaluation criteria state that the ERS should be educationally relevant, inviting and engaging, supporting reflection, facilitating user interaction, versatile, promoting collaboration and communication and compatible with classrooms. As in the study conducted with the first version of the heuristics for ERS by Christian Giang et al. [15], the usability issues are ranked by importance:

- 0 - Not a usability problem at all.
- 1 - Cosmetic problem only: Has no profound impact on the activity.
- 2 - Minor problem: Has a slight impact on the activity and influences the experience a bit.
- 3 - Major problem: Has a severe impact on the activity and negatively influences the user experience.
- 4 - Usability catastrophe: This problem has to be fixed in order to allow for a decent user experience.

Figure 46 presents the results of my heuristic evaluation in a chart based on shades of red. The darker a box, the greater the problem. Based on the ranks from the paper written by Christian et al. [15], the issues scale used is represented in figure 45.

Relevance	Not a usability issue at all	Cosmetic problem	Minor problem	Major problem	Usability catastrophe
Color level					

Figure 45: Scale used to rank the usability issues

The chart depicted in figure 46 hence visually represents the importance of the usability issues.

Heuristic	Issues	Cardbot	Interface		Geography activity
			Tiles	Android app	
1. Educational Relevance					
2. Attractiveness	Improving aesthetics				
3. Support for reflection					
4. User Interaction	Crashes				
	BLE disconnection				
	CV algorithm				
	Language				
5. Versatility	Difficulty levels				
6. Collaboration and communication					
7. Classroom compatibility	Expenses for micro:bit				
	Structure robustness				
	Construction time				
	Tools				
	Lack of tablets				

Figure 46: Self-reflecting heuristic evaluation

The elements of the ERS are evaluated together according to the design principles listed on the left-hand side of figure 46. The colours allow to quickly spot what has to be improved. First, the aesthetics of the programming tiles and the app are cosmetic problems. It might be improved but it does not prevent the proper use of the system.

The most important problems appear in the center of the chart of figure 46. They concern the CaPL Android app. The version of the app sent to the Ticino teachers embeds a computer vision algorithm that often crashed. This happens due to "Out of Memory" issues of the Android system while taking a picture with the camera. A solution to this might be to take the picture of the sequence of tiles with the built-in camera app of the tablet and then load it inside the app using the "Gallery" button. Overall, the computer vision algorithm presents even additional issues. First, the picture of the sequence of programming tiles needs to be taken under specific restrictive conditions. In fact, surrounding objects in the field of view are to be avoided, the picture should not be too blurred and the background should not present light reflection. As a reference, figure 13 presents a good example on which the computer vision algorithm performs without any issues. In addition to the fact that the computer vision algorithm is not really robust, it presents the limitation that the icons on the cardboard tile supports must present a continuous contour. If we make the app open source so that other people can use it and personalise the programming tiles with the addition of other tile icons, this problem will have to be fixed. Finally, the execution time of the identification algorithm is still pretty slow. As seen in section 2.2.3, it runs in approximately 3.5[s] in Java on the tablet. All This comes in the category "user interaction".

Another limitation of the app is that it is currently only available in English. This also concerns the "Geography Activity". It could be a good idea to translate the app in different languages (at least French, German and Italian) to make it accessible to as many schools as possible.

Regarding the “Geography Activity“, the current app does not integrate the possibility to easily customise the MCQ. This can be done by manually modifying the code of the app, but there is so far no interface that facilitates this process. This would be an important add-on that would allow the teachers to easily adapt the game specifically to the target children regarding age group and level of education. Moreover, the “Geography Game“ is pretty basic and presents only the simple feature of moving from a starting point to a destination. There are no levels like in a standard video game. There is also no feedback regarding the personal progression of the users. The addition of different levels could be used to track the progression of the users across time.

On the bottom left-hand corner of the chart of figure 13, important issues regarding the classroom compatibility of the Cardbot and the tiles were identified. First, schools have to spend money to obtain micro:bits and other basic electronic parts. But next to the expenses for acquiring the material, it should be noted that the construction of the cardboard elements might be problematic for schools. In fact, the cardboard structures should be more robust to be used by children and to ensure accurate movements of the robot. On the other hand, regarding the time allotted for a standard school activity, we have to face the fact that the preparation of the cardboard-based elements takes too much time. Especially the programming tiles that need to be made accurately to allow the correct working of the recognition algorithm. Indeed, a lack of accuracy in the conception of the tiles could lead the tiles recognition to fail. In the elaboration of the video tutorial for building the tiles [11], it took me more than one hour to produce 16 tiles. On top of that, as presented in the video tutorials [11] [12], it is easier to cut cardboard with a Stanley knife than with scissors. But this is not appropriate for all ages and the use of cutters should be avoided in classrooms.

Finally, an important issue is that not all schools have access to tablets. These schools have hence no way to use the CaPL system. However, today most people have cell phones. It would be hence useful to adapt the layout of the CaPL app and make it compatible with cell phones. This would make the CaPL platform accessible to more people.

4 Conclusion

CaPL is a complete ERS that includes tangible programming tiles, a robot and an Android app presenting some tasks. Its cardboard-based system can raise the interest of participants. Cardboard is an inexpensive material that is simple to cut, fold and assemble. It is a simple way to create the structure of the robot and the tangible tiles that can be assembled in a puzzle-like fashion to form the program instructions. However, especially considering the programming tiles, the experience can be frustrating if their design is not reproduced with a minimum amount of accuracy. In fact, the tiles identification algorithm is conceived to work with tiles of specific dimensions and proportions. In consequence, it may behave in unexpected ways or even crash the application if a bad picture of tiles is provided. In the case of use in schools, it might be a good idea either to already provide the tiles or to provide adequate assistance to students during the construction of the do-it-yourself. This depends on the age of the group of children and the total time allotted for the activity. On the other hand, the cardboard robot can be built much quicker than the tiles and will certainly give more personal satisfaction to the participants when they will see it moving and will be able to say they did it themselves. In certain cases it might hence be interesting to keep at least the construction of the robot in the activity.

One of the purposes of this project was to elaborate an accessible tangible programming platform to be used in schools that do not necessarily own a Thymio. It also presents a simple interface to introduce STEM and CS in their classrooms. However, they still need basic components such as two servos and a micro:bit. This has to be purchased in comparison with previous project that used the Thymio that is already widely spread among schools in Switzerland and other Central European countries. But the cost of accessing this platform remains reasonably low since the mainly used material is cardboard. In comparison with the two previous tangible programming projects, the activities designed in the Android application of this project are aimed at a younger audience between 5 and 7 years old. However, similarly to the other platforms, the design of the programming tiles can be adapted to specific level of students (e.g. by adding other digits for the 'for loops', making the robot move over greater distances, allowing 45° rotations, etc.) and to the case where a future version of the Cardbot would include some sensors (e.g. "if else statements" and "while loops" could be introduced as briefly explained at the end of section 2.1). In addition, the current programming games and activities of the app can be easily adapted to various age groups by, for instance, changing the displacement costs of the robot in the "Geography Game" or by implementing more advanced MCQ. In other words, the CaPL concept can be adapted to appeal to novices of all ages. Based on the experience and the motivation of the students, the adaptability of the tangible programming platforms can allow teachers to adjust the difficulty of the activities to their need [4]. I personally think that the most interesting point of the CaPL system is that participants have the possibility to build something themselves. It will certainly be something very rewarding and motivating for them. With 4 to 5 tablets per class, we believe that a study around this activity could certainly lead to promising results in the introduction of STEM and CS using a tangible approach.

The add-on of this CaPL project in comparison with the Thymio TPL and Thymio PaPL projects is really the use of a tablet. The last two projects aim already at better utilizing existing technological resources such as computers. But the use of a tablet in the current project was ideal for a tangible programming platform. It plays a vital role in the tangible experience of the activities by allowing spatial freedom. From a development point of view, the tablet also allows freedom to test new ideas

and to customize the activities. Moreover, activities like the “Geography Game“ presented here are holistic since they provide real time feedback, introduce both real and virtual agents and promote collaboration within workgroups. In the end, the activities of the CaPL project foster both group teamwork (thanks to the tokens system of the “Geography Activity“) and individual achievement (for instance, the elaboration of a Cardbot). Thus, this system gives a first attractive overview of STEM and facilitates the transition to more advanced programming and educational robotics activities such as Thymio TPL, PaPL and finally VPL.

There are of course always adjustments to be made and bugs to be solved. For instance, the execution time of the computer vision algorithm should be reduced, the design of the programming tiles could be made more attractive and it would be useful to translate the application into several languages. But all in all, this project provides a basic framework for implementing activities similar to the “Geography Game“ activity bringing into play a micro:bit-based Cardbot and an Android device. Finally, In spite of the confinement period and that it was not possible to have in-person meetings as of mid-March, the project was carried out in good conditions and was completed successfully. In fact, between mid-March and until the end of the semester, Christian, Aditya and I met only once in person. This was to prepare the kits for the study with the Ticino teachers. Unfortunately, a classroom study could not be carried out during this semester to assess the learning gains achieved by the system. The project presented in this report provides a framework for future studies when it will again be possible to physically go to schools to experiment with students. For the time being, the first steps have been initiated with the study with the Ticino teachers. We are currently waiting on their feedback and the results of the research. This project certainly has great potential and deserves to be pursued. From a personal point of view, I am very grateful to have been able to participate in the realization of this project. Technically, it allowed me to improve not only my knowledge in Python, Java and Android (with the coding of the app), but also in video editing and Photoshop (with the video tutorials and the figures of this report). Finally it was instructive to make use of the heuristics for the development and evaluation of ERS [15] to identify among others the limitations and weaknesses of my application. This is definitely the kind of tools I will continue to use in the development iterations of my future projects.

References

[1] Adafruit Industries. *Bluefruit Connect* Android application code. https://github.com/adafruit/Bluefruit_LE_Connect_Android, 2018. [Consulted on the 17th of May 2020].

[2] Aditya Mehrotra. *Instructions for building the Cardbot*. https://drive.google.com/open?id=1NmH7YyDJSo8QTSbmjwIc3QOasEoFy-_y, 2020. CaPL project.

[3] Aditya Mehrotra, Julien Dedelley, Christian Giang, Andrea Mussati, Francesco Mondada, Noé Duruz, Melissa Skweres. *Introducing a Paper-Based Programming Language for Computing Education in Classrooms*. https://infoscience.epfl.ch/record/276956, 2020. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education.

[4] Andrea Mussati, Christian Giang, Alberto Piatti, and Francesco Mondada. *A Tangible Programming Language for the Educational Robot Thymio*, 2019. In 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA). IEEE, pages 1–4.

[5] Anthony Guinchard. *CaPL* Android application code. https://github.com/Antho1426/capl-app, 2020. CaPL project.

[6] Anthony Guinchard. *CaPL-CardbotControl* hex file. https://makecode.microbit.org/_E78XRj12tMRL, 2020. CaPL project.

[7] Anthony Guinchard. *Instructions for building CaPL tiles*. https://drive.google.com/open?id=151KBXWqqCvBmEavRthDpzZYDldhrMoSD, 2020. CaPL project.

[8] Anthony Guinchard. *NVIDIA tablet pictures, different 'for loops' sequences*. https://drive.google.com/open?id=1rvpu6Q5EzOvNbjnQQleC_i9uNjz88B9E, 2020. CaPL project.

[9] Anthony Guinchard. *NVIDIA tablet pictures, sequences of various lengths up to 19*. https://drive.google.com/open?id=1e2DqL6UY9JCQ2Jet1UNcf2jIzN833B7G, 2020. CaPL project.

[10] Anthony Guinchard. *Video presenting the tiles identification procedure*. https://drive.google.com/open?id=1ioa26-rr3prDeVsS4RCqXhdKN-yAtl9U, 2020. CaPL project.

[11] Anthony Guinchard. *Video tutorial for building CaPL tiles*. https://drive.google.com/open?id=1W9eaOxSYfFDcqskoeT3yRGKapwvFj1Xd, 2020. CaPL project.

[12] Anthony Guinchard. *Video tutorial for building the Cardbot*. https://drive.google.com/open?id=1CyGw2uAddxRAWLTwhuk6XsiXkBVfigLi, 2020. CaPL project.

[13] Bertrand Schneider, Patrick Jermann, Guillaume Zufferey, Pierre Dillenbourg. *Benefits of a Tangible Interface for Collaborative Learning and Interaction*. https://ieeexplore.ieee.org/abstract/document/5654494, 2010. In IEEE Transactions on Learning Technologies, vol. 4, July-Sept. 2011, pages 222-232.

[14] BYOR / Solly Systems. *Build Your Own Robot*. https://www.byor.nl/de/, 2017.

[15] Christian Giang, Alberto Piatti, and Francesco Mondada. *Heuristics for the Development and Evaluation of Educational Robotics Systems*, 2019. In IEEE Transactions on Education.

[16] D. Alimisis. *Educational robotics: Open questions and new challenges*, 2013. In Themes Sci. Technol. Educ., vol. 6, no. 1, pages 63–71, 2013.

[17] DISTRELEC. *PIS-1585 - Bit:Buggy Car Kit for micro:bit, Pi Supply*. https://www.distrelec.ch/en/bit-buggy-car-kit-for-micro-bit-pi-supply-pis-1585/p/30163427, 2020. [Consulted on the 17th of May 2020].

[18] Edward F Melcer and Katherine Isbister. *Bots (Main) Frames: exploring the impact of tangible blocks and collaborative play in an educational programming game*, 2018. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM, page 266.

[19] European Commission. *A new skills agenda for Europe: Working together to strengthen human capital, employability and competitiveness*, 2016.

[20] F. B. V. Benitti. *Exploring the educational potential of robotics in schools: A systematic review*, 2012. In Comput. Educ., vol. 58, no. 3, pages 978–988, 2012.

[21] G Berry, G Dowek, S Abiteboul, JP Archambault, C Balagué, GL Baron, C de la Higuera, M Nivat, F Tort, and T Viéville. *L'enseignement de l'informatique en France Il est urgent de ne plus attendre*, 2013. In Rapport de l'Académie des sciences.

[22] Google Developers. *Android Studio*. https://developer.android.com/studio/?gclid=CjwKCAjwztL2BRATEiwAvnALcgdeZjovD8se4BOy0Imu7958FlAswY_bDMsfER_RI5n9B0qGmDcqShoCmokQAvD_BwE&gclsrc=aw.ds, 2020. The Official Android IDE.

[23] J. Nasir, U. Norman, W. Johal, J. K. Olsen, S. Shahmoradi, P. Dillenbourg. *Robot Analytics: What Do Human-Robot Interaction Traces Tell Us About Learning? 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019. Computer-Human Interaction in Learning and Instruction (CHILI) Lab Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland.

[24] J. Shin, R. Siegwart, and S. Magnenat. *Visual Programming Language for Thymio II Robot*, 2014. Interact. Des. Child.

[25] JetBrains. *IntelliJ IDEA*. https://www.jetbrains.com/idea/, 2020. Capable and Ergonomic IDE for JVM.

[26] JetBrains. *PyCharm*. https://www.jetbrains.com/pycharm/, 2020. The Python IDE for Professional Developers.

[27] Kamibot. *Kamibot Interactive Papercraft*. http://www.kamibot.com/default.php, 2020. Introducing the world's first programmable paper robot.

[28] Kori M Inkpen, Wai-ling Ho-Ching, Oliver Kuederle, Stacey D Scott, and Garth BD Shoemaker. *This is fun! we're all best friends and we're all playing: supporting children's synchronous collaboration*, 1999. In Proceedings of the 1999 conference on Computer support for collaborative learning. International Society of the Learning Sciences, page 31.

[29] Laila El Hamamsy. *Multi-Party Collaborative CoWriter: Activity With Engagement Strategies and Modelling*, 2019. Master Project in Micro-Engineering, CHILI-Lab, EPFL.

[30] Lilia da Costa. World map picture. lilianyumew@icloud.com, 2020. Made with iPad Procreate app.

[31] M. Chevalier, F. Riedo, and F. Mondada. *Pedagogical Uses of Thymio II*, 2016. In IEEE Robot. Autom. Mag., vol. 23, no. 2, pages 16–23.

[32] Martin Woolley. *micro:bit Blue* Android application code. https://github.com/microbit-foundation/microbit-blue, 2017. [Consulted on the 17th of May 2020].

[33] Massachusetts Institute of Technology. *MIT App Inventor*. https://appinventor.mit.edu, 2020. [Consulted on the 17th of May 2020].

[34] Michael S. Horn, Erin Treacy Solovey, and Robert J.K. Jacob. *Tangible programming and informal science learning: making TUIs work for museums*, 2008. In Proceedings of the 7th international conference on Interaction design and children. ACM, pages 194–201.

[35] Micro:bit Educational Foundation. *BBC micro:bit*. https://microbit.org, 2020. [Consulted on the 17th of May 2020].

[36] Microsoft. *Microsoft MakeCode for micro:bit*. https://makecode.microbit.org, 2020. [Consulted on the 18th of May 2020].

[37] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. *Scratch: programming for all*, 2009. In Commun. ACM 52, 11 (2009), pages 60–67.

[38] Neil Fraser. *Ten things we've learned from Blockly*, 2015. In 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond). IEEE, pages 49–50.

[39] OpenCV. *Image Segmentation with Watershed Algorithm*. https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html, 2020.

[40] OpenCV. *OpenCV*. https://opencv.org, 2020.

[41] Pierre Dillenbourg. *Design for classroom orchestration*. https://www.sciencedirect.com/science/article/abs/pii/S0360131513001061?via%3Dihub, 2013. In Computers Education, vol. 69, November 2013, pages 485-492.

[42] Ross Atkin. *Smartibot: The world's first A.I. enabled cardboard robot*. https://www.kickstarter.com/projects/460355237/smartibot-the-worlds-first-ai-enabled-cardboard-ro?lang=de, 2020.

[43] Royal Society (Great Britain). *Shut down or restart?: The way forward for computing in UK schools*, 2012. Royal Society.

[44] Samsung. *micro:bit* official Android application code. https://github.com/Samsung/microbit, 2019. [Consulted on the 17th of May 2020].

[45] Stefania Bocconi, Augusto Chioccariello, and Jeffrey Earp. *The Nordic approach to introducing Computational Thinking and programming in compulsory education*, 2018. In Report prepared for the Nordic@ BETT2018 Steering Group. doi: https://doi. org/10.17471/54007.

[46] Stephen Cooper, Wanda Dann, and Randy Pausch. *Teaching objects-first in introductory computer science*, 2003. In ACM SIGCSE Bulletin, vol. 35. ACM, pages 191–195.

[47] T. Sapounidis, S. Demetriadis, and I. Stamelos. *Evaluating children performance with graphical and tangible robot programming tools*, 2015. In Pers. Ubiquitous Comput., vol. 19, no. 1, pages 225–237, 2015.

[48] Tanja Waculik. *Der große Lernroboter-Test – Teil 11: Der Blue-Bot*. https://lehrerweb.wien/aktuell/single/news/der-grosse-lernroboter-test-teil-11-der-blue-bot/, 2018. [Consulted on the 14th of May 2020].

[49] TTS Group. *Using Bee-Bot for Literacy*. https://www.youtube.com/watch?v=ZJaSQgsDQ1w&list=PLpRLICEWJ7WYbI3skS7EOshfS2I7_2ZqL&index=8, 2011. [Consulted on the 17th of May 2020].

[50] TTS Group. *Blue-Bot* iOS app. https://apps.apple.com/us/app/blue-bot/id957753068, 2017. [Consulted on the 1st of June 2020].

[51] TTS Group. *Blue-Bot Remote* Android app. https://play.google.com/store/apps/details?id=uk.co.tts_group.BlueBotRemote&hl=en, 2017. [Consulted on the 1st of June 2020].

[52] TTS Group. *Bee-Bot Lesson Idea 3 from TTS Group*. https://www.youtube.com/watch?v=re2tlLmHuf8&list=PLpRLICEWJ7WYbI3skS7EOshfS2I7_2ZqL&index=6, 2018. [Consulted on the 17th of May 2020].