Implementation of Tangible Programming for Thymio II

Stefan Teller, Alexander Tuma, Julian M. Angel-Fernandez, Markus Vincze

***Abstract*— It is difficult to introduce children to programming when they are still attending preschool or middle school. Tangible programming languages (TPL) offer a way to allow children to program robots without any prior technical knowledge. Our TPL focuses specifically on the above mentioned target group and uses mainly symbols and colours instead of text. A program is created by lining up tangible blocks in a row. This is photographed and converted to a program for the Thymio II. To verify functionality the TPL was implemented in MATLAB and it worked as expected.**

I. INTRODUCTION

Programming is nowadays an important part in the education of children. For example, the United Kingdom introduced a new curriculum to address this problem [1]. Usually children are older than 14 years before they are taught their first programming skills with text-based programming languages, for example C, Java and Python [2][3].

In order to improve the learning experience of programming, appropriate methods and technologies must be used. Moreover, exposing children earlier to programming could increase their interest in technology.

To facilitate the access to programming and get children interested in the field, TPLs can be used. The idea behind this is, that the tangible blocks have similarities with children's toys and so arouses their interest. M. Giannakos gives a good overview of some existing TPLs like Tern, Quetzal, T_ProRob and T-Maze in [4].

Another big advantage of TPLs is that they reduce the screen-time of kids in comparison to conventional programming languages. With our implementation the time in front of a screen is negligible small. Already screen-time of preschoolers is too high, as Tandon states in [5]. Kids do not have to sit in front of a PC to program a robot but instead are able to create their programs by hand, the same way they play with toys.

In this paper we describe the development of a TPL that can be used for children starting from preschool age. This is achieved by using text as well as symbols and colours, which would enable children without any reading knowledge to program Thymio II [6]. Our developed language makes it possible to create more complex programs whereby children up to middle school can benefit and learn from it. Furthermore, we have placed great importance on the fact that our TPL can be adapted quite easily to other hardware and robots.

II. DEVELOPING OUR TANGIBLE BLOCKS

A. Target group

The target group for the following TPL is in later preschool to primary school age. There is no previous knowledge necessary for kids as well teachers. When the abilities of this TPL are exhausted, one can easily change to more complex languages like Scratch.

B. First approaches

The first idea was to develop simple rectangular blocks for all statements in common text programming languages like conditionals, loops, jumps and subroutines. Since, for example, the IF statement consists of at least four blocks (IF, condition, action at fulfilment and action at not fulfilment), it needs a lot of space. This was the reason why complex blocks like "Drive forward until obstacle" were created.

C. Final block design

All the blocks and their individual functions should be self-explaining. Therefore, one block is divided into three areas like shown in Figure 1:

- At the top left is the description area. The block in this area is described with a few simple words like "turn left".
- The top right area is for the TopCode [7]. Each block has an individual identification number which is represented on it via a TopCode.
- At the bottom is the symbol area. An expressive symbol represents the purpose/action of the block. For "Set LED yellow" it is a lamp symbol with a yellow body.

The shape of the blocks should be chosen in a way that it is intuitive to connect them properly. For that reason the blocks have certain shapes to match up side by side. At the top and bottom of every block the colour is faded out in the meaning that there should not be placed any other blocks. This can be seen in Figure 2.

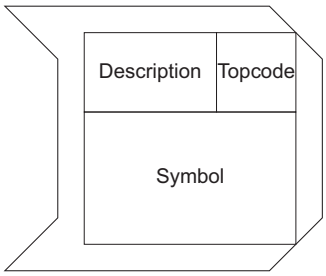


Fig. 1. Template for a block with the size of 60mm in length and 50mm in height. The notches have a length and height of 10mm.

D. Block repertoire

The blocks were designed to use as many hardware features as possible from the Thymio II robot. The following list gives an overview of the implemented blocks.

- Start (green): Program start.
- End (red): End of the program.
- Drive Blocks (purple): Drive forward until obstacle, drive backward until obstacle, turn left, turn right.
- Loop Blocks (green): Repeat five times, repeat forever, end of repeat.
- Colour block (rose): Set LED to red/green/yellow/blue, flash LED five times red/green/yellow/blue, turn off LED.
- Time blocks (grey): Pause 1s, Pause 0.5s.
- Sound blocks (yellow): Play system sound #1/#2/.../#7.

The ”Start” and ”End” blocks have their colours correspondent to those of traffic lights for go and stop. From the list above, it is evident that all blocks using the same robot's resource have the same background colour. Hence blocks with different colours could also be executed in a parallel manner. A selection of blocks can be seen in Figure 2.

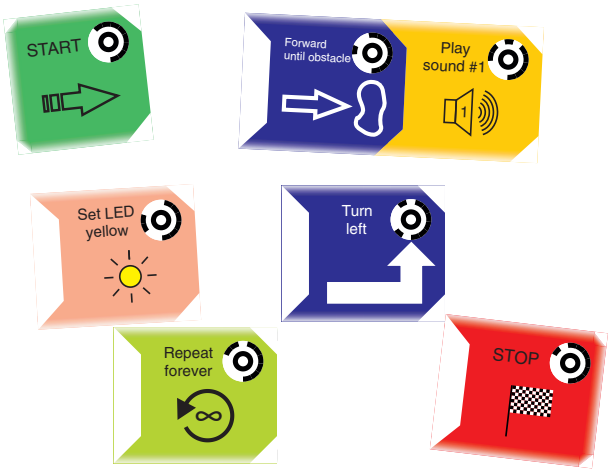


Fig. 2. Selection of blocks with all mentioned features.

E. Abstraction

Already simple tasks for a robot can require many lines of code in text programming languages, this is equal to blocks in a TPL. To keep track of the number of blocks for a program, we have introduced advanced ones. In this way, on one hand, the clarity can be increased, on the other hand it can be shown what functions are and how these work. Because advanced blocks are merely function calls, as Figure 3 illustrates vividly in the case of flashing five times with the red LED. This is an example of combining basic blocks to an advanced block. This can also be done the other way around by splitting an advanced block up into several basic blocks.

III. IMPLEMENTATION

A visual representation of the implementation can be seen in Fig. 4. The program is placed with its blocks on a flat surface. There has to be a start and an end block, all blocks between those two are considered to be part of the program. To convert the physical implemented program to digital

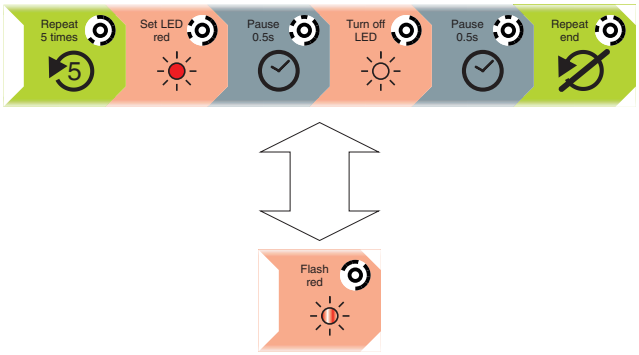


Fig. 3. Scalability: Six basic blocks combined to one advanced block to save space.

instructions for the Thymio II robot, an image of the arranged blocks has to be made first either by a webcam or still camera. This obtained image is processed using the TopCode library which returns the identity, position, diameter and rotation of all TopCodes in the image. Since every block (and so every TopCode) represents a specific instruction for the robot, it is now possible to translate these instructions into a text-based program written in Aseba [6], a text-based programming language especially developed for the Thymio II robot. This step is done by our developed compiler, which we call tangible compiler. The created file is then flashed to the robot. All calculations as well as the tangible compiler are implemented in a MATLAB script.

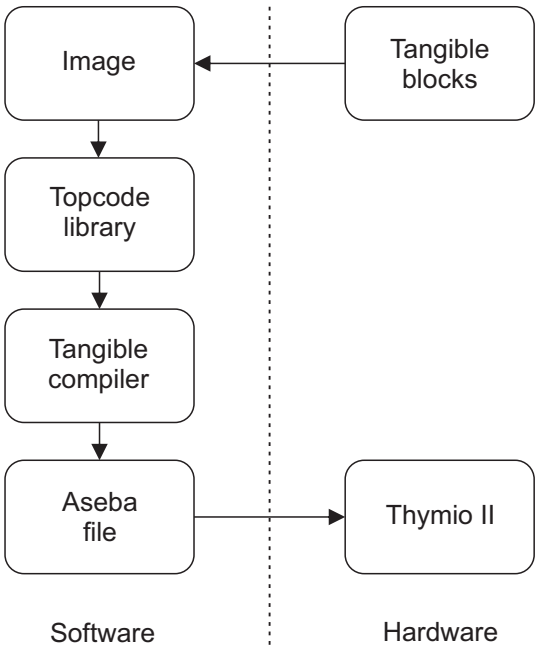


Fig. 4. Overview of the implemented system.

A. Syntax

The grammatical syntax description in Backus-Naur-Form is shown in Figure 5. To minimize syntax errors, all blocks

```
<Program> ::= <Start>
    <Centre Blocks>* <End>
<Centre Blocks> ::= <Blocks> | <Loops>
<Loop> ::= <Start of repeat>
    <Centre Blocks>* "End of repeat"
<Start of repeat> ::=
    "Repeat five times" |
    "Repeat forever"
<Blocks> ::=
    <Drive Blocks> | <Colour Blocks> |
    <Time Blocks> | <Sound Blocks>
<Drive Blocks> ::=
    "Drive forward until Obstacle" |
    "Drive backward until obstacle" |
    "Turn left" |
    "Turn right"
<Colour Blocks> ::= "Set LED to red" |
    "Set LED to green" |
    "Set LED to yellow" |
    "Set LED to blue" |
    "Flash LED  five  times red" |
    "Flash LED  five  times green" |
    "Flash LED  five  times yellow" |
    "Flash LED  five  times blue" |
    "Turn off LED"
<Time Blocks> ::= "Pause 1s" |
    "Pause 0.5s"
<Sound Blocks> ::=
    "Play system sound #1" |
    "Play system sound #2" |
    "Play system sound #3" |
    "Play system sound #4" |
    "Play system sound #5" |
    "Play system sound #6" |
    "Play system sound #7"
```

Fig. 5. Backus-Naur-Form (BNF) syntax [8].

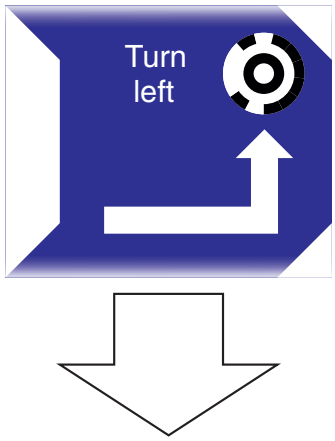
have pre written code snippets like shown in Figure 6. Blocks can be placed in an arbitrary order with a few exceptions:

- There has to be a start block and an end block.
- With every start of a loop there has to be an end of the loop.

Following these simple rules, no syntax errors will occur and the program will be executed from the start block to the end block.

B. Tangible compiler

The tangible compiler creates a Aseba file of the blocks which can be flashed on the Thymio. With the position of the start and end block in the image, a straight line through these two blocks is calculated as well as the normal distance of all other blocks to this line. Now all blocks are recognized as part of the program when they are between the start and end




```
timer.period[0] = TIME_TO_TURN
motor.left.target=-MOTOR_SPEED_TURN
motor.right.target=MOTOR_SPEED_TURN

if timer_fin == 1 then
    timer_fin = 0
    motor.left.target=0
    motor.right.target=0
    state = state + 1
    timer.period[0] = 0
end
```

Fig. 6. Translation from a block to a code snippet for turn left.

blocks and the normal distance is below a defined threshold. For the correct order, the remaining blocks are sorted from start to finish. With this approach it is achieved, that blocks which appear unintentionally in the picture or are obviously not part of the program, are not taken into account. Figure 7 shows an example setup including the calculated line (red) as well as the threshold (yellow) without considering the blocks in the upper half.

The next step is to investigate whether an advanced block is included in the program. If so, its identity is replaced by those it represents (Figure 3). Then the syntax of the program is checked.

Next, the file is created and flashed on the Thymio. For this purpose, a text file containing the code for each identity must exist. The header, constants and definitions are in the code of the start block. The actual program is built as a state machine, which dynamically builds up to the size of the program with the code snippets of the individual blocks.

Finally, the created file is downloaded to the robot. In our case the software provided with Thymio is used. Depending on the settings, the program starts automatically or by pushing a button on the robot.

C. Adaptation

The described procedure can easily be implemented for other robots and in other programming languages. The

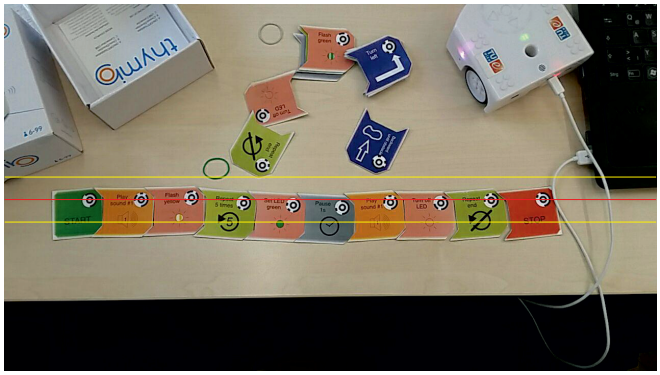


Fig. 7. An example of a created program.

used TopCode library is open source available in several programming languages and the tangible compiler which was implemented as a MATLAB script can also be easily ported, since no extraordinary functions were used.

The code in the target language (which was Aseba in this case) can be changed easily to any other text based language by changing the code snippets from the external files and the syntax of the state machine.

D. Results

The described implementation of the TPL works as expected. The processing time lasts only a few seconds, with the download of the program to the robot taking most of the time.

IV. FURTHER IMPROVEMENTS

A. Conditional statements

Since conditional statements would consist of several blocks and it is difficult to display them with symbols, explicit conditional executions were not implemented. We have put more emphasis on simple syntax than on complex programming. However, implicit conditional statements occur in some blocks, for example in the ”Forward until obstacle” block.

B. Parallelism

Currently only single-line programming is implemented. However, the TPL could be extended to multi-line programming, for example in combination with the aforementioned conditional statements. By parallel or two-line programming after an IF query, the upper line would represent the IF branch and the lower one the ELSE. Another possibility would be that parallel branches are executed simultaneously. This could be combined well with our colour concept. Since different colours are linked to different resources of the robot, they can be executed in parallel, which means that each colour can only appear in one line.

C. Subroutines

To create small complex programs, self-defined subroutines could be implemented. For this, new blocks would have to be introduced, one block for defining the subroutine and several blocks for its execution. The subroutine would be programmed like a normal program, except that it is started with its definition block instead of the start block. The subroutine has to be visible in the picture as well.

V. CONCLUSIONS

An attempt has been made to develop a programming language for children from preschool age, which does not require prior technical knowledge. Because of the user’s young age, the programming language should be kept simple using colours and symbols. This has been achieved by the development of tangible blocks. Each block displays a symbol, a TopCode and a description. Lining up the blocks represents the steps of creating a program. Using a picture of the lined-up blocks, the program is read from it using the TopCodes library. A specially developed compiler creates a text-based code which is loaded onto the robot using its provided software. The developed programming language still has to be experimented with children further and adapted appropriately. As a next step, the number of available blocks and the complexity of the language can be increased.

REFERENCES

[1] The national curriculum in England https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/425601/PRIMARY_national_curriculum.pdf, 2013.

[2] Lehrpläne - Neue Mittelschule https://www.ris.bka.gv.at/Dokumente/Bundesnormen/NOR40181121/NOR40181121.pdf, 2016.

[3] Kölling, Educational Programming on the Raspberry Pi. 2016.

[4] M. Giannakos, Reviewing the Affordances of Tangible Programming Languages: Implications for Design and Practice. 2015

[5] Pooja S. Tandon, Chuan Zhou, Paula Lozano, Dimitri A. Christakis, Preschoolers’ Total Daily Screen Time at Home and by Type of Child Care. 2011.

[6] Thymio II and the Aseba language, https://www.thymio.org/ Programming Languages for Classroom Use. 2007.

[7] TopCodes - Tangible Object Placement Codes, http://users.eecs.northwestern.edu/~mhorn/topcodes/

[8] Peter NAUR (ed.), Revised report on the algorithmic language ALGOL 60, 1960.