Teaching fundamental programming concepts with the BBC micro:bit

Nina Lobnig, Markus Wieser, Stefan Pasterk and Andreas Bollin

Department of Informatics Didactics, University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

Abstract

Learning to program is essential for computer science and computer science education. It helps in structured, at best computational thinking and in understanding how computers and algorithms work. We view this as a critically important skill these days. Programming can be taught in different ways using different tools and technology. In this contribution, we present materials for the BBC micro:bit specially designed for learning and practicing fundamental programming concepts such as sequential program flow, branching, variables, and loops, as well as event control. The learning materials consist of several parts and are structured step-by-step with detailed explanations and examples. Furthermore, they are supplemented by motivating, playful exercises. We describe the learning materials in detail, why they were structured in this way, and what teaching ideas and thoughts underlie their development.

Keywords

learn programming, micro:bit, block based programming, computer science education, programming concepts

1. Introduction

With the impact of computer science in society and education, also the interest of people to get in touch with this area's concepts rises. Besides related subjects at schools, other institutions offer workshops or laboratories to get in touch with topics of computer science. The University of Klagenfurt offers the *Informatics Lab* [11] (supported by the RFDZ, the regional center for subject-related didactics) since 2014, which develops teaching materials for computer science education. These materials are tested in repeatedly offered thematic workshops on a variety of computer science topics. On the website[1], there is also a platform where the developed materials are available for teachers and all other interested people. The lab does not only search for good examples, it also tests existing learning materials and continuously develops new ones.

Learning to program is part of most international curricula in different stages of education starting at early ages. Programming is often connected to computational thinking and to the understanding how computers and algorithms work. To support the process of learning, different programming environments and hardware kits have been developed and are used in

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

nina.lobnig@aau.at (N. Lobnig); markus.wieser@aau.at (M. Wieser); stefan.pasterk@aau.at (S. Pasterk); andreas.bollin@aau.at (A. Bollin)

[1] https://www.rfdz-informatik.at/ (note: site in German)

practice. One very popular tool is the *BBC micro:bit*, a small single board computer developed for the use in school classes [1].

In the *Informatics Lab*, workshops with the BBC micro:bit are offered as it is one of the most popular tools in many (local) schools. This, along with the fun and easy part of block-based programming, was one of the reasons for choosing the micro:bit. Another reason were the many possibilities in using the micro:bit later on and the available extensions, from working with the pins and breadboard to controlling robots (e.g. Bit:Bot).

In a first version, these workshops followed an exploratory approach. After a short explanation of the programming interface and the handling of the micro:bit, the students tried out their own ideas as well as given broad tasks (such as programming 'rock-paper-scissors'). These tasks were broad in the fact that their solution required a combination of several programming concepts (in the case of the example above: branching, variables and random numbers in combination with each other). However, these concepts were not explicitly taught, the intuitive and self-explanatory usage of block-based programming was relied upon. This often led to the situation that students who had a certain way of structured (computational) thinking managed the tasks quite well and others (often the majority) needed a lot of help and so solutions were often worked out together with the teacher.

This contribution presents revised and improved learning materials for the micro:bit which are currently used in the *Informatics Lab* and follow a more structured approach with focus on teaching fundamental programming concepts.

2. Related Work and Development

2.1. Related Work

A first step in the revision process was searching for comparable materials for teaching programming with the micro:bit. We could find some promising resources, but they were quite similar to the first version of materials in the *Informatics Lab* and inherited the same difficulties. There are a lot of learning resources that have the same or similar exploratory approach and focus on the fun part of working with the micro:bit and rely on the easy to use programming environment. One example is the open educational resource textbook from Bachinger and Teufel [1] (or the web version [8]), which is mentioned on many websites for using the micro:bit in school. They already write in the preface that the book is in fact not a pure computer science nor a programming textbook, but instead 'a collection of exciting, creative and practical tasks that help students to develop an awareness of everyday problems, for which they can use their microcomputer to solve'. They also state that the fun aspect is one of the core goals ('The joy of doing is always at the center!').

The fun and motivating part was one of our goals too, but we also wanted to teach the most important, fundamental ideas of computer science and programming. The exploratory approach did not fulfill this like we wanted to. These requirements were also not met by any found resources (e.g. [1, 9, 7, 6, 13, 5] and many others) and seemed to face the same difficulties like we had at first.

Therefore, more than four years ago, two of the authors started developing a new concept for the micro:bit workshops and for learning how to program with it. The goal was to focus on

programming concepts, combined with the playful approach offered by the micro:bit. Those are basic materials and provide a good basis for going further. It can be followed by working with the hardware components of the micro:bit or going deeper into the world of programming. Furthermore, with the acquired knowledge, it should also be more easy to learn textual programming later on, because the concepts are already known and have been practiced. This was the reason for developing the materials presented here and what goal they pursue.

2.2. Fundamental Programming Concepts

Probably every computer scientist can list the most important, fundamental programming concepts or program structures as variables, branches and loops (and maybe some more). A look into different programming books shows these as well, as in 'Sprechen Sie Java?' (translated to 'Do you speak Java?') [10], which is frequently used in German-speaking countries (and especially at universities), and many other: independently whether that is Java or Python as a programming language, or you program in VBA, and whether it is object-oriented or not (see [2, 4, 3]).

Now, we look at which fundamental programming concepts can be easily realized with the micro:bit, all object-oriented elements are discarded and what remains is: writing simple, sequential programs, working with variables (with the assignment of values to them), perhaps dealing with input and output, as well as the two fundamental control structures branch and loop.

In our material, the programming concepts are first learned and understood separately and used in combination in later tasks. In this way, we hope to improve the understanding of the concepts and that students do not create code on a guessing approach, but with specific considerations. For this reason, the materials deal with branches first, followed by variables, and then the combination of both. We hope to reduce the complexity by covering only one concept at a time. The loops then complete the material, although they could theoretically also be placed in between, but since the common textbooks (see references above) deal with them in this order (first branches, then loops), this was also chosen here. To make branching and arithmetic comparisons possible without the variable concept (in a meaningful way), random numbers are necessary. This is one of the reasons why these are taught quite early (in comparison to other approaches for learning to program). Another reason is that randomization is used quite often with the micro:bit as well as in programming - it occurs in dice or random games and similar programs (for example the popular oracle or rock-paper-scissors games), that are often implemented with the micro:bit.

The concept of subroutines, such as methods and functions can be implemented with the micro:bit as well, but due to the limited programming possibilities this does not actually make sense and rather seems to be a forced approach. Working with arrays is indeed possible and micro:bit offers suitable programming blocks, but this concept is somewhat more complex. In our opinion it is not one of the basic programming concepts and therefore is not covered in the (current) basic materials. For working with arrays in a meaningful way, prior knowledge about variables and especially about branches and loops is necessary. Hence, this can be used for an extension of the (basic) materials presented in this paper.

2.3. Didactical Considerations

The reason for the materials to be structured this way is based on educational and didactic considerations collected in the *COOL Informatics* approach [12]. It fosters a fun and motivating way of learning and follows the four principles *discovery*, *cooperation*, *individuality*, and *activity*. The fun factor was a basic consideration as the possibly first contact with programming should be exciting to increase the interest. But this factor is supported by the micro:bit as it is easy to use and makes a lot of fun.

The material is divided into several parts - each with an input part that explains the particular concept and its blocks and shows an example, as well as an exercise part with several different tasks. The explaining parts always contains a practical example, which describes and illustrates the introduced concept in detail, as it is promoted in the *COOL Informatics* principle *discovery*. Indeed, the materials are also designed for *self-study* and can be used and understood without a supervising teacher and can also serve as a reference guide for learners (this is done quite often, see later in chapter 4). As such, they also offer the possibility of being used in inverted or flipped classroom settings. Due to this, the materials can be seen as a textbook and allow flexibility, especially with the students' heterogeneous backgrounds and interests, following the principle *individuality*. The material has been designed to suit the abilities of all learners, which can vary widely. Furthermore, the tasks for the different topics are mixed in terms of difficulty and include so-called expert tasks (challenging tasks - for high-achieving students or those with previous experience or deeper interest). Additionally, according to the principle *activity*, learners should be encouraged and given the chance to implement their own ideas and have room to do so. The principle *cooperation* is considered within the workshops as students are encouraged to work together on tasks and to help each other if possible.

In addition, it was a matter of concern to make the materials appealing to all genders and to ensure that the tasks do not reflect any stereotypes or anything similar. Therefore, the tasks are kept very neutral (in terms of gender) and also the shown figures and graphics, as well as the used colors are not gender-specific. In addition, the examples and exercises are not overly technical or mathematical, as is often the case in programming books and courses. This should make programming more interesting for everyone, regardless of other (technical) interests and abilities.

Finally, the materials are *Open Educational Resources*, short OER, and thus available for use, free for modification by others (under the CC-BY-NC-SA license), and open for improvements and revisions.

3. Overview of the Learning Material

The material consists of six parts and every part is a combination of an explanatory and an exercise part. In total it has more than 30 pages and another nearly 30 pages for solutions to the exercises. Therefore, we will not present all the material here and instead only take a look what the material is about and give an overview of the content and the structure as well as the type of explanations and examples given. The documents are available on the website of the RFDZ[2].

[2] https://www.rfdz-informatik.at/grundlagenmicrobit/ (German only)

The materials are aimed for children from nine years and above. It was used in workshops with children from the upper elementary school and lower secondary school. There are no prerequisite skills, other than text comprehension and reading, the presented learning materials were designed for the first encounter to programming and programming environments. Although it is not a disadvantage if experience (with Scratch, code.org or other) is already brought along.

3.1. Introduction of the micro:bit and the Programming Environment

The first part starts with a general explanation of the micro:bit and answers the questions what the micro:bit is, what features it has and what you can do with it. Also, the advantages of block-based programming are briefly mentioned. It continues with the overview of the programming environment, where to find what and how to work with it. After all the introductory bits, the first blocks - the basic ones - are presented. It starts with the two blocks 'on start' and 'forever', which are already in the programming environment at the very beginning, and then the other basic blocks are listed with short explanations. These blocks are output blocks and display the given output, like text or symbols, with the 25 LEDs at the micro:bit's front side. It concludes with a first sample program (displaying a 'Hello' on start and then showing a smiley face forever), followed by a reference to the exercises.

The exercises start with the invitation to try implementing own ideas with the presented blocks and then give some example exercises for those, who have no idea or prefer predefined tasks.

3.2. Event control (Working with Input) and Random Numbers

This part consists of two concepts and the first proceeds with simple, easy to understand blocks. Those are the ones controlling the inputs recognized by the micro:bit. This is considered easy as the micro:bit was designed for this (in contrast to many other programming environments).

The learning material for this concept again starts with a short introduction and introduces the involved blocks. The given sample code for working with this type of blocks is an extension of the first sample program (the one from the part before) and adds the code for showing an arrow to the left, whenever button A is pressed. The provided exercises (apart from the suggestion to try out your own ideas) are producing output depending on certain movements or input via buttons. For example greet the persons next to you, when the button on their side is pressed and greeting yourself for the A+B input.

The second concept to be taught in this part of the learning material is the concept of *randomization* and therefore introduces the block (from the mathematics block category) for getting a random number within a specified range. This enables to write programs that represent a dice (sample program) and other fun possibilities, like an oracle that 'predicts' how many presents someone will get for their next birthday, or to challenge your seatmate who is better at guessing the micro:bits next shown number, or to develop an a instructions generator (with an extra description list, which activity to do when a certain number pops up). The given examples already indicate, why we decided to introduce the randomization quite early in the learning materials. It enables creating fun programs with just a few blocks and a bit of creativity.

Additionally, it makes it possible to introduce the branching concept in the next step without the need for variables.

3.3. First Contact with Branching

In this part, a first contact with branching is provided for the students. It again starts with a short introduction and then goes over to the explanation of the corresponding blocks, which are the branching block itself as well as the block for conditions (like the comparison of two numbers). The concept of branching is explained as a distinction of cases and 'distinguishes between a special event or not'. We had the event handling before, but now we can define by ourselves, what a special event should be. And for each of the two cases, special event or no special event, programming blocks can be provided. The block for the (branching) condition is introduced as a so called 'question box', that can tell yes or no. With it, we can for example check if two given numbers are the same or one is smaller than the other and so on.

The two blocks are then shown in a sample program. This program is the improved version of the oracle from the exercises to *randomization* and displays a tick or cross depending on the random number given. This sample program is explained in great detail and supported by the illustration shown in Figure 1.

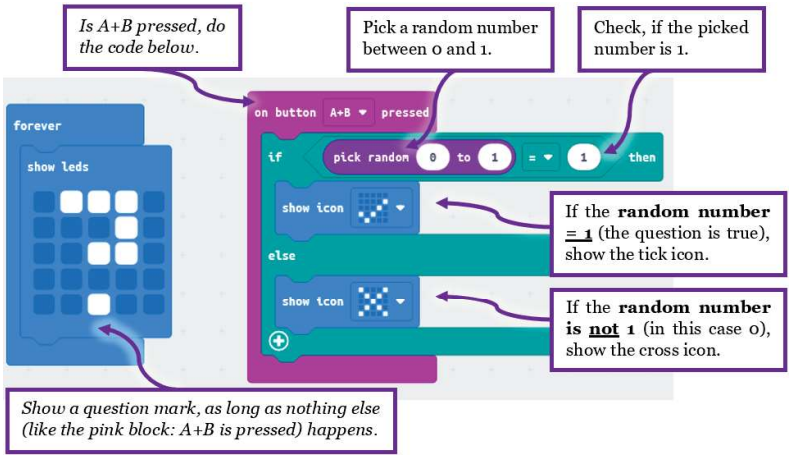

Figure 1: Explained sample code for branching

The associated exercises are comprehension questions about branching and conditions in combination with random numbers. It displays code snippets, but with a changed range for the random numbers and the students have to assign the possible random numbers to the corresponding code sections. This type of exercise is a pen and paper task, an excerpt is shown in Figure 2. But there are exercises for programming too, like implementing drawing lots with matching output text or randomly showing one of two possible icons when pressing a button or a rain and sunshine simulator (two sunny days for one rainy day) with matching icons.

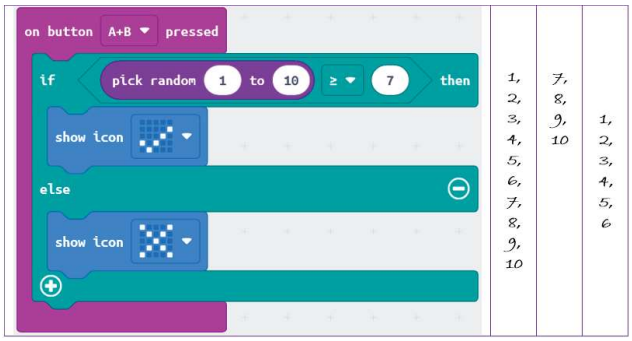

Figure 2: Excerpt of a exercise on branching and conditions with columns to write down: all possible numbers, the numbers for tick and those for cross

3.4. Start Working with Variables

After a first contact with branching and conditions and understanding and practicing this concept, the concept of variables is introduced meanwhile without combining it with branching. It is presented as the way to store values. It is explained as a kind of a cell, into which a single number can be written. This stored number can be retrieved and also be changed. The changing of a variable value is, by the way, explained like erase and newly write a value into the cell. Following this introduction, it is shown how to create a (new) variable and the related programming blocks - for retrieving the value, changing it *to* a certain number and changing it *by* a certain number.

The given sample code is a implemented counter, that goes up by one or down by two depending on the pressed button. This program is expanded with a starting value of 5. Then the question is raised, how to make the program start with a random number and how to expand the code to show a heart for reaching the target value of ten (instead of the shown number in any other case). The corresponding solution with explanations is also part of the learning material.

There are corresponding exercises for programming with variables too. From tasks for exploring the function and effect of the blocks, to various simple counter programs as well as a bit more complex tasks with negative variable values and events for reaching the target value. The task for 'experts' is one with counting how many times the micro:bit has been shook, but with a randomly chosen target value (that can be displayed). For this, two variables are needed.

3.5. Combination of Branching and Variables

In the next-to-last part of the learning material, the concepts of branching and variables are combined and we show the possibility of having more than two branching cases and how to handle that. What follows is the usual introduction and short explanation to expanding the branching block and where which code belongs. In this part, there are no new blocks. The given sample code together with detailed explanations is from the familiar oracle program, which is expanded by one more case. The added case represents the neutral oracle answer 'I don't know' and is displayed with a certain smiley icon. The corresponding code is shown in Figure 3.

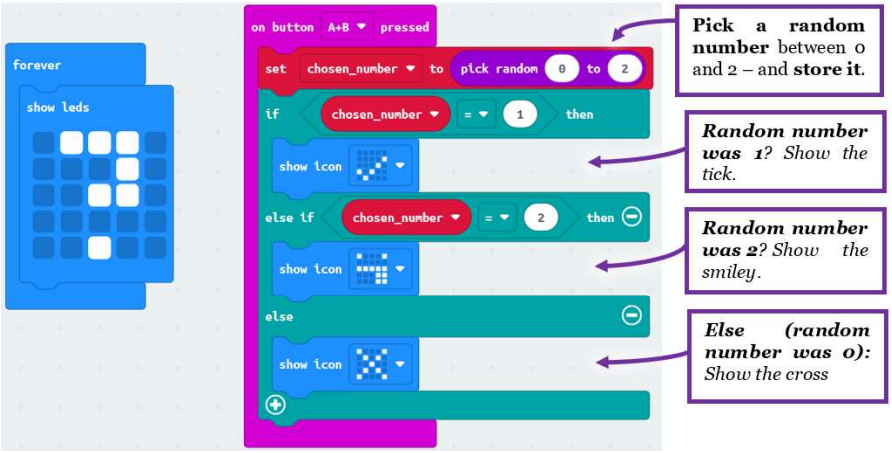

Figure 3: Explained sample code for a oracle with three cases (branching in combination with variables)

With this knowledge, it is now possible to have even more fun exercises, starting with expanded oracle games to the implementation of rock-paper-scissors. As noted before, this is part of many micro:bit materials and is also suggested as one of the most prominent games on microbit.makecode.org. Here it might become clear, why we take a rather structured approach with our materials, rather than a solely experimental approach (the problems that arise in doing so have already been briefly addressed in the mentioned section).

Other exercises in our learning material are the slightly modified game 'Rock-Paper-Scissors-Lizard-Spock' (known from a popular tv series), as well as a version of using the micro:bit as a dice with the related dice-icons. There is also a task for 'experts', which is is about implementing an instructions generator that even checks if the instructed activity was performed correctly (like pressing a certain button or moving the micro:bit in a certain way).

3.6. Repeating Code with Loops

The last part is about repeating code with loops (going beyond the already known forever loop). The three main types of loops - counting, for and while - are introduced. Each of these is explained with an example, such as repeating code exactly nine times with the counting loop or using the for loop for the additional output of the current number of repeats. The while loop is demonstrated with the example of a startup screen, which is displayed till shaking the micro:bit.

For practising the loop concept and the use of the three different loop types, there are associated exercises too. They are about reproducing a very similar code to the ones explained before (like repeating code for a specified number of times or adding a startup screen to an already existing program). Another exercise is about counting backwards and another one about implementing a counter (like in the exercises related to variables), that flashes an icon depending on the set number. The next-to-last exercise is implementing an adjustable countdown and has a follow-up question regarding the use of negative numbers. Expanding the program to work with these too is then the goal of the last, the 'expert' task (requires branching).

4. Experience with the Material

The presented learning material for the micro:bit has been around since the end of 2018 (first concept) or mid-2019 (first elaborated version) and therefore, we already worked with it multiple times as well as teachers did. We are now interested in experiences with the materials and therefore, we started a first pre-study asking three teachers about their opinions (through free text answers to questions regarding their perception and experience with the learning material).

Moreover, we wanted to learn more from the students' point of view, so we did a second pre-study (n=18) and asked those of our last micro:bit workshop to fill out a survey on their perception. The findings are very promising and show a trend (students' interest in computer science as well as in programming increased quite well), but further research is needed.

4.1. Experiences from the Teacher's Point of View

In the *Informatics Lab*, we use the materials in larger workshops when the learners are with us for several days (e.g. in summer) or the workshop consists of several units (e.g. once a week over a few weeks). Typically, we introduce and explain the concepts and corresponding blocks in front of the class and afterwards it is the students' turn for practicing the concepts and writing programs by themselves. This teaching setting is also used by the three teachers - out of one told us, that she does not use the explanatory parts often with students, but for preparation purposes or for looking things up. This teacher also wrote, she uses the exercise solutions quite often. She for example cuts them into pieces and has the students reassemble the correct code or places the solutions on the side of the classroom and the students can go and take a look when they need help or need a hint how the code should be structured.

There are quite big differences in the knowledge and experience as well as in the performance of the students, but the materials were designed for handling this heterogeneity. The materials are often printed for all the students and given to them step-by-step, so low performing students can reread things and following parts can be given ahead of time - for everyone to do challenging things and practice newly known concepts. In addition, it is possible for everyone to work at their own pace. All this (well working with heterogeneity) was also broadly mentioned by all of the teachers. The large amount of tasks (and its clear separation) was also noted very positively.

Moreover, the asked teachers view the developed materials as a good approach for learning fundamental programming concepts (using the micro:bit). And students are seemingly programming code more purposeful and less chaotic (mentioned by one teacher), but this has to be investigated further to put this as a general statement. Besides this, it is observed that the associated exercises encourage students to work with the micro:bit and practice the concepts (in a playful way), as mentioned by the teachers. Later on, it is also possible to do group projects with the micro:bit and even interdisciplinary approaches like the multiple examples found on the internet (for example the cited references from section 2.1).

4.2. Survey Results on the Student Perception

We now know quite well what the experience and opinions look like from the teachers point of view. But what about the student side? How do they perceive the lessons and the materials?

And do they - positively or negatively - influence their perception of computer science and programming? These were our central questions in a short survey the students from our last workshop filled out. The lessons (referred as 'workshop' later on) were held in April/May 2022 as part of the computer science education in a local school and 22 students (lower secondary school) took part, of which 18 completed the survey (two weeks after the last lesson). The majority of the students had no prior experience with programming and very few with computer science. The existing knowledge was only about the basic handling of the computer (turning it on, navigating, opening programs) and in software for text-editing and creating slideshow presentations.

The first four questions aimed at the perception of computer science and programming, the students had before the workshop and now have since the workshop. The precise questions were Q1: 'If you can remember, what did you think of computer science *before* the workshop (with the micro:bit)?' and Q2: 'What do you think of computer science *since* the workshop with the micro:bit?' (both translated from German). And the same for programming too (Q3 and Q4). The results are promising, although more (also comparative) data has to be collected on this in the future. But it shows a strong trend, as we can see in Figures 4 and 5. The interest in both has increased quite a lot, which are very good news (regarding our goals and the promises of the micro:bit).

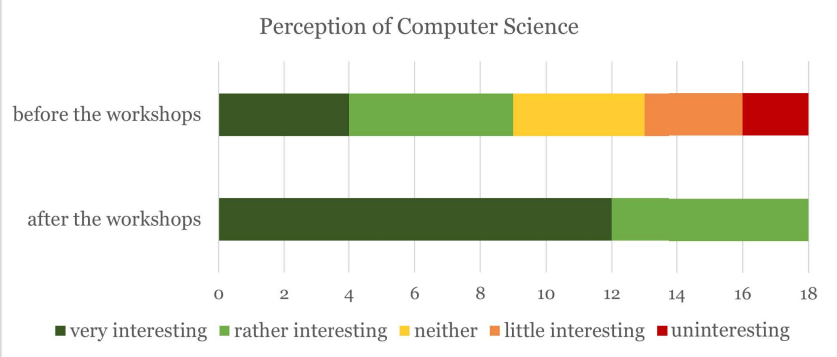

Figure 4: Results on the change of students perception to computer science (n=18)

Another part of the survey (Q5) was to tick all the words the students would describe the workshop with. There were eight predefined words ('was fun', interesting, time-consuming, exciting, motivation, confusing, difficult and boring) as well as room for other words. This was used a few times, but this can be omitted (as synonym words were used and ticked additionally to the original ones). All 18 students described the workshop to be fun and nearly all of them (16) ticked 'interesting' too. But learning to program (and programming itself) is also time-consuming, which was expressed by seven students. 'Exciting' and 'motivating' were ticked six times each, confusing four and difficult three times and boring never.

Q6 and Q7 focused on how interesting the workshop was perceived as well as the given tasks/exercises (overall). We are happy with the result, displayed in Figure 6, as it gives a good feedback to the workshop. However, the first may include side parameters (e.g. sympathy).

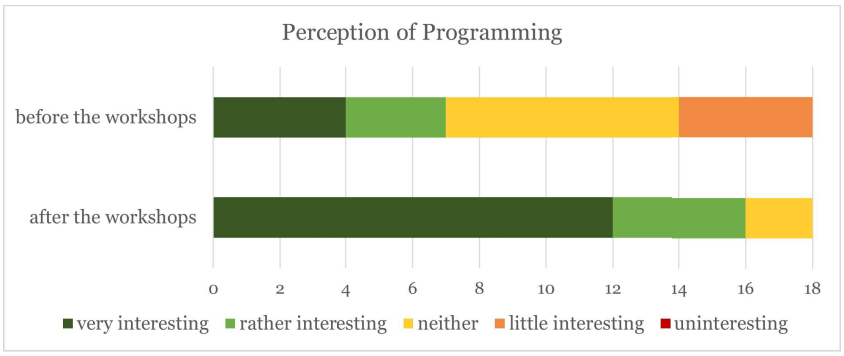

Figure 5: Results on the change of students perception to programming (n=18)

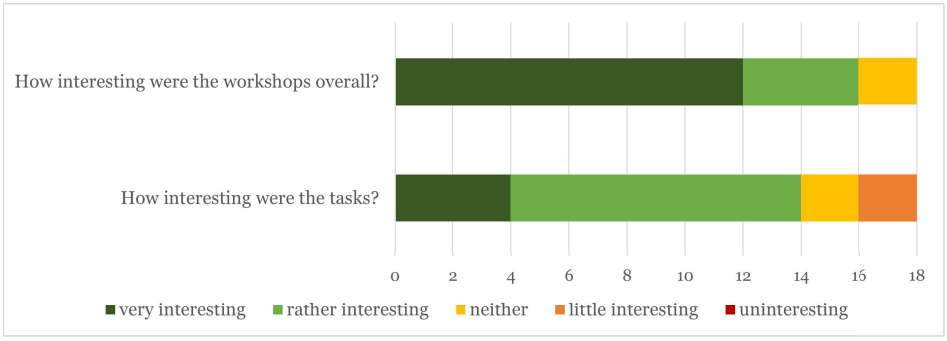

Figure 6: Interest in the programming course and the provided exercises (n=18)

Q8 was on the difficulty of the exercises. Exactly half of the students rated them as appropriate (9), a few as rather easy (3), a few more as somewhat difficult (5) and one as very difficult. Because of this, we will consider improving the tasks a little bit (for example with reassembling code snippets and other task types) in the future. The last question (Q10) was an open question (regarding further feedback), where two students reported they would have enjoyed more time and lessons with the micro:bit.

So overall, the workshop and the developed learning materials are not only perceived well by teachers, but also by the students, who seem to have a lot of fun using (and learning with) the micro:bit. The survey results are promising and we are curious on more data and research regarding the materials.

5. Conclusion and Future Work

As shown above, the developed learning materials are suitable for learning programming (using the BBC micro:bit) and at the same time increase the students' perception on computer science and programming. Even out of a teachers point of view, the materials are very practical, especially in heterogeneous classes (regarding performance and pre-knowledge in programming).

It is planned to produce videos and interactive (H5P) elements for this learning material and do further research - with more students and teachers and also on other aspects of the material. In the future, we also want to look at whether the concepts taught are also remembered and mastered in the long term. Another issue is translating the learning materials into English, but there is currently limited time and capacity for all of this. We also wish to improve the materials further in expanding the materials to more concepts (e.g. arrays) as well as the given exercises.

References

[1] Bachinger, A., Teufel M.: Digitale Bildung in der Sekundarstufe - Computational Thinking mit BBC micro:bit. 1st edn. Austro.tec, Grieskirchen (2018). Available under: https://microbit.eeducation.at/images/f/f2/Buch-microbit _20180729.pdf. Last accessed 19 May 2022

[2] Inden, M.: Einfach Java: gleich richtig programmieren lernen. 1st edn. dpunkt.verlag, Heidelberg (2021).

[3] Kellner, F., Brabänder, C.: Programmieren in VBA. In: VBA mit Excel. Springer, Berlin, Heidelberg (2020).

[4] Klein, B.: Einführung in Python 3: in einer Woche programmieren lernen. 2nd edn. Hanser, München (2014).

[5] LAG Informatik: micro:bit - Beispiele für den Unterricht. 2018. https://docplayer.org/140654151-Micro-bit-beispiele-fuer-den-unterricht.html. Last accessed: 20 May 2022

[6] Learning Lab (TU Graz): BBC micro:bit Werkstattbeispiele. https://learninglab.tugraz.at/informatischegrundbildung/bbc-microbit-werkstattbeispiele/. Last accessed 20 May 2022

[7] makecode.org: Tutorials, Games. https://makecode.microbit.org/. Last accessed 19 August 2022

[8] microbit eEducation: micro:bit - Das Schulbuch. https://microbit.eeducation.at/wiki/Hauptseite. Last accessed 20 May 2022

[9] microbit.org: Projects. https://microbit.org/projects/. Last accessed 19 August 2022

[10] Mössenböck, H.: Sprechen Sie Java?: eine Einführung in das systematische Programmieren.3rd edn. dpunkt.lehrbuch, Heidelberg (2005).

[11] Pasterk, S., Sabitzer, B., Demarle-Meusel, H., and Bollin, A.: Informatics-Lab: Attracting Primary School Pupils for Computer Science. In proceeding of the 14th LACCEI International Multi-Conference for Engineering, Education, and Technology, San Jose, Costa Rica (2016).

[12] Sabitzer, B.: A Neurodidactical Approach to Cooperative and Crosscurricular Open Learning: COOL Informatics, Habilitation thesis, University of Klagenfurt, (2014).

[13] SchulArena.com: micro:bit. https://www.schularena.com/ict/informatik/make-it/micro-bit. Last accessed: 20 May 2022