Multimedia Enriched Practices

Teach Yourself Coding with the Micro:bit

Armin Ruch[1]

University of Education, Schwaebisch Gmuend, Germany & Istanbul Erkek Lisesi, Istanbul, Turkiye

Abstract: *Coding is a skill that is relevant for today's students. However, it can be a challenge to find material adequate for K9-students, as well as for educaters who did not have a formal coding-education themselves. Thus, this course aimes to provide a general background of skills that can be used for further learning. It is designed to lead students and educators through the first steps of programm-planning and programm-coding, on the basis of the famous Micro:bit environment.*

Keywords: *Stem education; coding; Micro:bit; basic skills*

Introduction

Germany has gone through and is still going through the process of finding an appropriate way to teach digital skills to students. Practical coding skills are one of these skills, even though there are several more. The following text addresses educators as well as students who like to gain or teach coding skills on a basic level. The text can be used in two different ways. First, it can be used as a guide to take students through corresponding classes. However, especially with motivated students, it can be given as an assignment as a do-it-yourself (diy) guide.

The overall aim is the introduction to important concepts of coding, as well as the introduction of important sensors and actors of the Micro:bit. These concepts will be introduced step by step and then integrated into the overall project. As the overall project, a game has been chosen. The project has been introduced to German students of different ages by the author and is most appropriate for students from age 14 on. The text, as well as the code will be shown here.

[1] Corresponding Author: Dr. Armin Ruch, University of Education, Schwaebisch Gmuend, Germany & Istanbul Erkek Lisesi, Istanbul, Turkiye, armin.ruch@iel1884.org

To cite this article: Ruch, A. (2023). Teach yourself coding with the Micro:bit. *Journal of Multimedia Enriched STEM Education*, *1*(1), 47-58.

This text is the first part of a series that will eventually cover the whole process of coding the aforementioned game.

Method

This text faces the challenge to put elements of graphic coding into writing. Thus, the author had to develop a form to write such elements. These rules are the following:

1. The name of a block is written between two slashes, such as /name of block/. Between the slashes the name will be spelled identical to the name that is spelled on the block itself. Within the text, the following text would read as /on start/ (see Appendix 3.1)
2. Some blocks offer the option to enter values. For example, the block /show number …/ has the option to enter a number. Other blocks offer the option to enter text or other blocks. To illustrate the fact that input can be made, the name is followed by three dots (…). Thus, the block below will be spelled as /show number …/ (see Appendix 3.2)
3. If a block is specified with a number, text or another block, the three dots will be exchanged against that specific input. In the following example, this is /show number 8/ (see Appendix 3.3).
4. If a block is specified with another block, the three dots will be exchanged with the block. If, as shown below, the /show number …/-block is specified with a mathematical operation such as "+", this will be spelled as /show number/… + …// (see Appendix 3.4). Note that the rules for "/" follow the rules for brackets in mathematics. For every opening "/" there has to be a closing "/".
5. To illustrate that something is a variable (this concepts will explained later in detail), the name of the variable is preceded and followed by an asterisk (*). Thus, the variable with the name "LED_X" will be spelled as *LED_X* in the text (see Appendix 3.5).
6. If a block is used that includes other blocks inside its "belly" (such as the /if … then/-block, the block will open with two slashes "//". This way, the condition on top, as well as everything inside the belly can be displayed. The following block //if button A is pressed then/ show number acceleration (mg) x// is the spelling for the block as displayed in Appendix 3.6.

For the purpose of better reading and the paper's use as a course for individual learning, the second person will be used to address the reader. This might be unconventional for journal articles; however it will increase the readability in this paper's specific context. In addition, the video (https://youtu.be/tR0kpqKXdxw) created to support understanding may be watched.

Phase 1

In this phase you will learn how to show the value of a sensor.

The following blocks will be used:

/forever/ ***/show number/*** ***/acceleration (mg) x/z/z/***

The following problem will be solved:

"Develop a code that will permanently show the tilt on the X-Axis."

This is what you should do:

The Micro:bit can be tilted into three directions. Scientifically this is called "acceleration" or "tilt", even though "acceleration" is usually used in another context in everyday language.

How far the Micro:bit is tilted to the left or to the right is called the X-direction.

How far a Micro:bit is tilted to the front and back is called the Y-direction. Turning the Micro:bit around itself, such as the hands of a clock around the center of the watch-face, will be displayed as Z-direction. These are also displayed in Appendix 3.14.

To show the corresponding value of the tilt in X-, Y-, or Z-direction, the following has to be programmed as followed. The example refers to the X-Axis and can be similarly used for Y and Z:

"Show the value (number) of tilt on the x-axis continuously."

Therefore, the blocks /show number/ and /acceleration (mg) X/ are necessary. They will be stacked into each other for /show number /acceleration (mg) X//.

As this has to be done continuously, the above block will be put into the /forever/-block. As a result,

/forever/show number/acceleration (mg) X///

has to be coded (see Appendix 3.7).

As the Micro:bit needs time to display the value, the numbers might appear to change slowly. That is due to an integrated pause in the background-programming of the Micro:bit. Without the display, the Micro:bit makes approximately 10.000 readings/second.

The program should be saved as "Phase 1".

Phase 2

In this phase you will learn how to display a sensor reading when pressing a button and how to use a if-then logic operation.

The following blocks will be used

/forever/ ***/show number/*** ***/acceleration (mg) X/Y/Z/*** ***/if-then/***
/button A is pressed/

The following problem will be solved:

Write code for the Micro:bit, so that:

1. You will show the X-acceleration on demand.
2. You will show the Y-acceleration on demand.

This is what you should do:

In phase 1 the Micro:bit displayed the value of x-acceleration permanently. This is not practical, especially, when other operations are needed, too. Therefore, the program should change in such a manner, that the acceleration is only displayed when wanted. As a signial for the demand, the buttons A and B will be used. Thus, the program should display:

When button A is pressed, display the x-acceleration.
When button B is pressed, display the y-acceleartaion.

You can see, that there are certain "conditions" that need to be fulfilled in order to make something else happen. Everytime that it is required to do something only if something else happened before, the /if-then/ Block can be used. This block is //if … then//. The condition can be entered inside the top of the block, while that what is supposed to happen afterwards, will be integrated below. The standard value for the condition is "True", which has to be replaced with the desired condition. This block can be found in the folder "Logic" (see Appendix 3.8).

As all of this is supposed to happen continuously, the /forever/-block will be used as the most outward block:

/forever//whenButton A ist pressed then/show number acceleration (mg) x///

The same can be done for the reading of the y-Axis-value. The whole code can be copied and pasted and the value afterwards changed from X to Y. Both codes will be combined in the /forever/-block. The text is now written in several lines to increase the readability. This writing almost looks like the block-coding (see Appendix 3.9).

/forever
//if button A is pressed then/
show number /acceleration (mg) x//
//if Button B is pressed then/
show number/acceleration (mg) x//
/

Again, a pause is included into the back-programming of the Micro:bit. Thus, the program might not react immediately after a button is pressed, when another button has been pressed very soon before. The program should be saved as "Phase 2"

Phase 3

In this phase you will learn how to display predefined icons on the display. Further, you will learn how to display individual icons on the display and how to turn them off again.

<u>The following blocks will be used:</u>

/forever/ ***/show number …/*** ***//if … then//*** ***/Button A/B/A+B is pressed/*** ***/show icon/***

<u>The following problem will be solved:</u>

1. When button A is pressed, a happy-smiley-icon is displayed.
2. When button B is pressed, a sad-smilley-icon is displayed.
3. When button A and B are pressed together, the display is cleared.

This is what you should do:

The Micro:bit has predefined functions for the use of the LED-matrix. These functions can be accessed easily. Later, for much more sophisticated projects, more complex code can be used, too. Looking at the problem, it becomes clear that a similar structure to phase 3 can be used. As it was the case in phase 3, here again certain actions are triggered by specific conditions – again, pressing a button. This times, however, three such conditions are needed. As a result, an icon has to be shown. These can be found in the menu "basic". There, the icons are displayed and can be chosen. However, one does not find an icon to turn of all LEDs. One has to use the /show leds/- block and turn off all the LEDs. With the /show leds/-block it is also possible to display own creations.

Therefore, to meet the demands of the problems, the following code is needed. As can be seen, the happy smiley is realized via the /show icon/-block, while the sad smiley is realized via /show leds/ and thus created "by hand" (see Appendix 3.10).

/forever
//if Button A is pressed/
/show Icon „happy Smiley"//
//if Button B is pressed/
/show led „sad Smiley"//
//if Button A+B is pressed/
/show leds „empty"//
/

Phase 4

In this phase you will learn how to turn on and off individual LEDs in the display without the /show leds …/-Block.

The following problem will be solved:

1. When Button A is pressed, the LED x2|y2 is turned on.
2. When Button B is pressed, the LED x2|y2 is turned off.

This is what you should do:

From the previous programs you already know the //if … then//-Block. While before, LEDs were turned on and off with the /show leds/-block, now they will be addressed via their coordinates. The LEDs are arranged in a matrix, where every LED can be addressed by their x- and y-value. Unusual is

the fact, that the LED with the coordinates x0|y0 is located in the top left corner of the LED-matrix. The numbers increase to LED x4|y4 in the bottom right corner. The coordinates of all LEDs are provided in Appendix 3.11. In this program the task has been to turn on and off the LED in the middle of the matrix, which has the coordinates x2|y2.

The block that is used to turn on individual LEDs on and off is /plot x… y…/. It has two slots to enter information, as indicated by the three dots behind the x and behind the y. The opposite, turning off individual LEDs can be done with the /unplot x… y… /-block.

Therefore, the program needs to be written as followed (see Appendix 3.12)

/forever
//if button A is pressed/
/plot x2 y2//
//if button B is pressed/
/plot Pixel x2 y2 aus//
/

Phase 5

In this phase you will learn how to use a threshold-value from a sensor as an input instead of a button.

The following blocks will be used:

/forever/ ***//when … then//*** ***/show Icon …//acceleration (mg) X/Y/Z/***
/… < …/ ***/… ≥ …/***

The follwing problem will be solved:

1. When the Micro:bit is tilted to the left side, a sad smiley is displayed.
2. When the Micro:bit is tilted to the right side, a happy smiley is displayed.

This is what you should do

It is obvious in comparison with the phases before, that the pressing of the buttons has been changed to the tilt of the Micro:bit. The display of the two smileys is used as an example for any action that could later be started or ended by the reading of a sensor. To solve this problem, the programmer needs to know that the tilt sensor displays values ranging from -1024 to +1024. The smallest value is used to display that the Micro:bit is tilted all the way to left, standing upright on its left side. The highest value is used to display that the Micro:bit is standing upright on its right side. Lying flat corresponds with the value 0.

The necessary code translates to:

When the value of the sensor is smaller than 0, show a sad smiley.
When the value of the sensor is larger than 0, show a happy smiley.

On clear inspection, a minor flaw becomes obvious, as the value 0 is not assigned to any smiley. Therefore, on of the two operators for the comparison has to include 0. Thus the conditions will be changed to:

When the value of the sensor is smaller than 0, show a sad smiley.
When the value of the sensor is larger or equal 0, show a happy smiley.

The general code is already known from the previous phases. The only change is the introduction of the comparison blocks (see Appendix 3.13).

/forever
//when /acceleration (mg) X < 0//
/show Icon "happy smiley"//

//when /acceleration (mg) X ≥ 0//
/show Icon „sad smiley"//

Discussion

This is the first part of a course that is intended to introduce students and teachers to the concepts of coding. Despite looking rather simple on first glance, the Micro:bit proves to be a very complex system that enables the realization of simple tasks for beginners, as well as very sophisticated tasks for advanced users. This course has proven to be effective with K9 students in Germany.

Limitation and implications

Despite all, the course here has limits that need to be addressed. First, students will need access to a computer with an internet connection. That can be a challenge in some places. To truly unfold its power, at least one Micro:bit should be available as hardware in class. Again, that can be problem in some places, as prices and access to stores that sell the Micro:bit can be challenging. Lastly, this course has been written in English. Therefore, the ability to read and understand English as a foreign language, can be challenging in itself. Lastly, the continuation of this course has not been published, when this first part is. Thus, the initial readers will have to continue with the material on the makecode-page (www.makecode.microbit.org) or on the homepage of the Micro:bit (www.microbit.co.uk). In addition, publications about the Micro:bit are available now in several languages. The skills form this course can be a stepping stone for further learning.

Appendix 1: Questions that can be used for measurement and evaluation activities

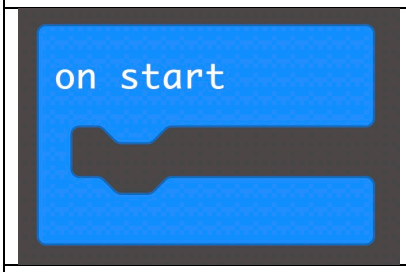
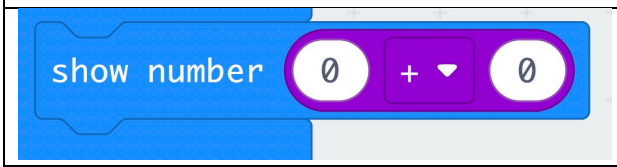
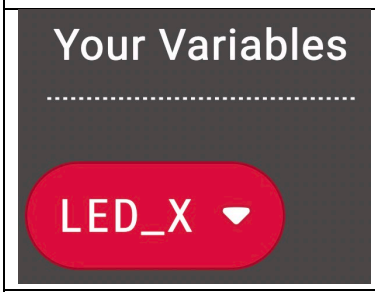
1. How can coding be used to improve everyday life?
2. Think of a tool that uses the gyroscope to improve the safety of workers. What other hardware would be needed?

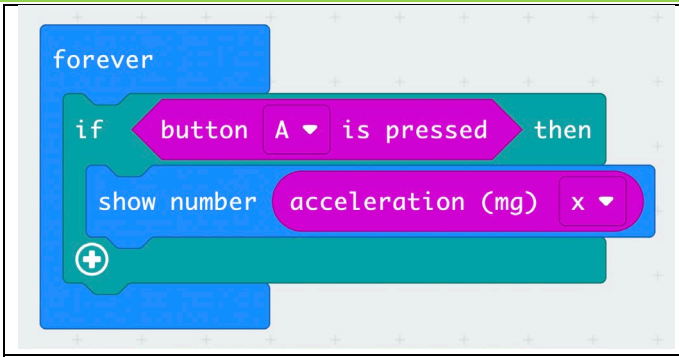
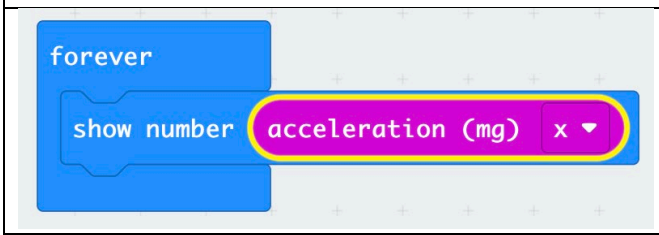
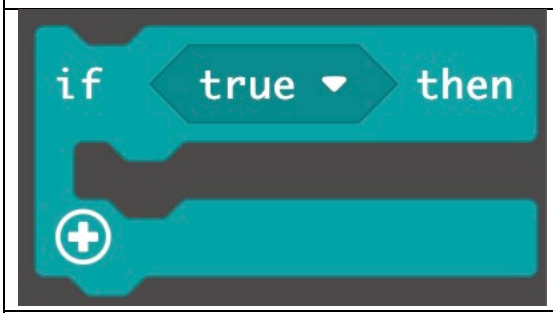
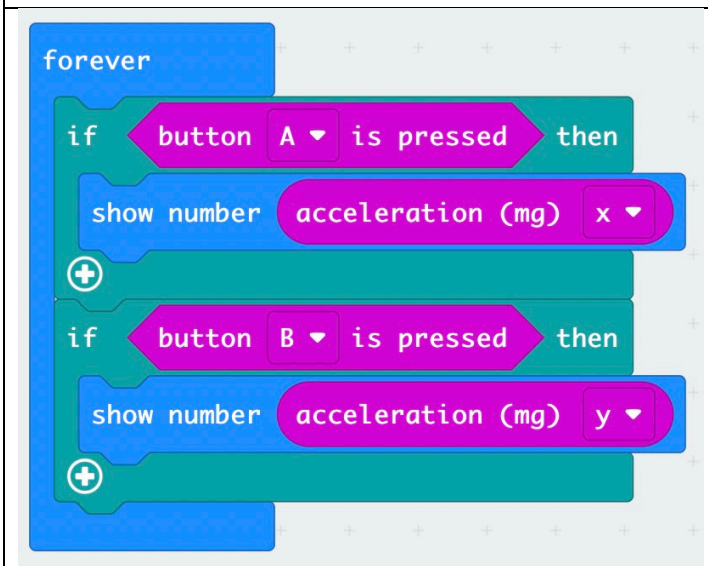
3. Think of an image that you would like to display on the Micro:bit display. Which pixels need to be turned on and off? Draw an image on checkered paper and add numbers accordingly.
4. Research how the light-sensor of the Micro:bit displays light-intensity. Write a code that shows different icons according to the level of light.

Appendix 2: People From The History Of Science And Engineering

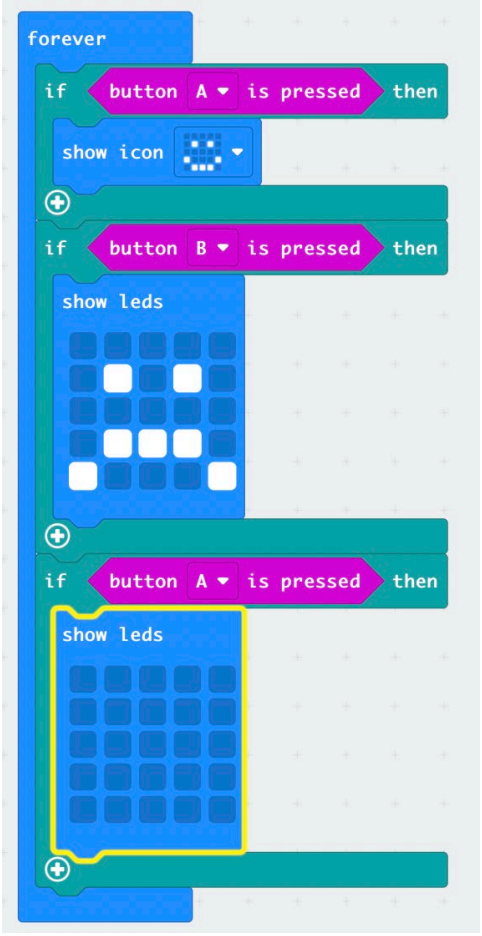
Alan Turing (1912 – 1954)	Alan Turing was a British mathematician, crypto-analytic, and information scientist. Today is viewed as one of the most influential computer-developers. Turing created much of the fundamental theories of modern computer-science. Amongst his inventions was the so-called "Turing-Machine", a theoretical model that defines an abstract machine. He is attributed with cracking the supposedly uncrackable "Enigma-Code" during the second world war.
John McCarthy (1861-1931)	John McCarthy is one of the founders of the field of artificial intelligence (AI). He was one of the co-authors of the document that invented the term AI. In 1971 he was awarded the prestigious Turing-Medal for his work on AI, as well as the United States Medal of Science.
Grace Hopper (1815 - 1852)	Grace Hopper was a computer scientist, mathematician and United States Navy rear admiral. She is attributed with the invention of the first "linker", something similar to today's links. Before she joined the Navy, she earned a Ph.D in mathematics from Yale University. She contributed greatly to the development of compilers that would translate English language into machine language. To honor her great achievements, she received 40 honorary degrees from Universities around the world. The navy named a destroyer after her. She received the United States Medal of Science and was posthumously awarded the Presidential Medal of Freedom.
Sam Altman (1985 – today)	Sam Altman is one of the minds behind the recently come-to-fame software ChatGPT. Altman started coding at a young age and became proficient in Macintosh programming first. He got admitted to Stanford University, but dropped out to persue the development of an App that he later sold for 43 Million USD. Together with the famous entrepreneur Elon Musk, Altman founded OpenAI in 2015. Besides ChatGTP, the company OpenAI developed several AI programmes, such as DALL.E., a program for the artificial creation of images.

Appendix 3: Micro:bit codes

1	
	*/on start/*
2	
	*/show number …/.*
3	
	*/show number 8/*
4	
	*/show number/… + …//*
5	
	**LED_X**
6	

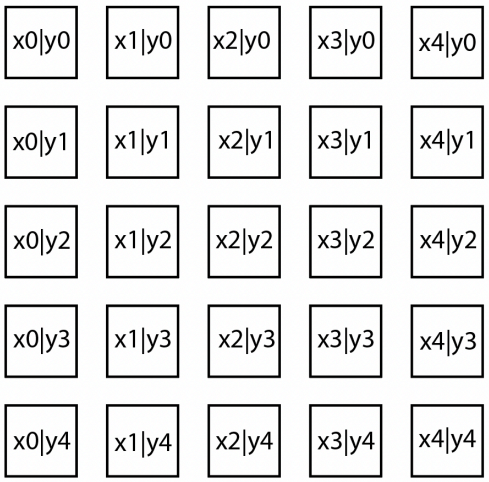
	***//if button A is pressed then/ show number acceleration (mg) x//***
7	
	***/forever/show number/acceleration (mg) X///***
8	
	***//if … then//***
9	
	***/forever*** ***//if button A is pressed then/*** ***show number /acceleration (mg) x//*** ***//if Button B is pressed then/*** ***show number/acceleration (mg) x//*** */*

10



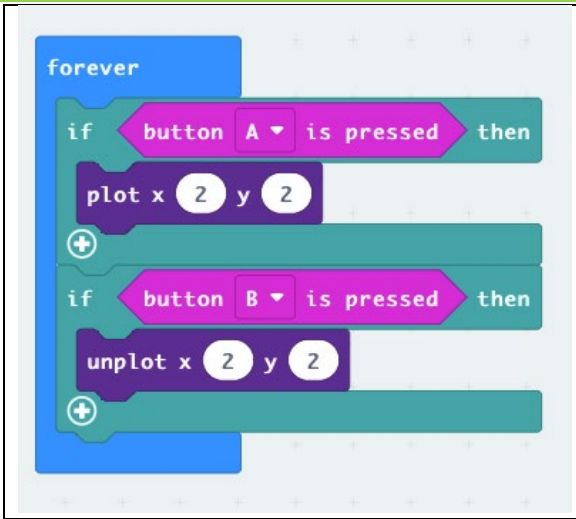
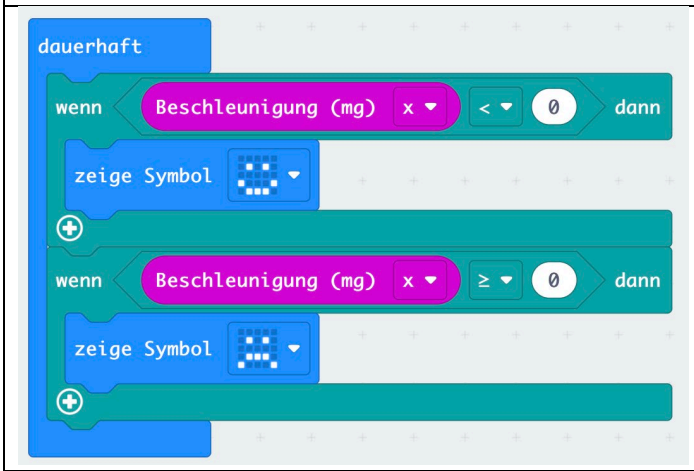
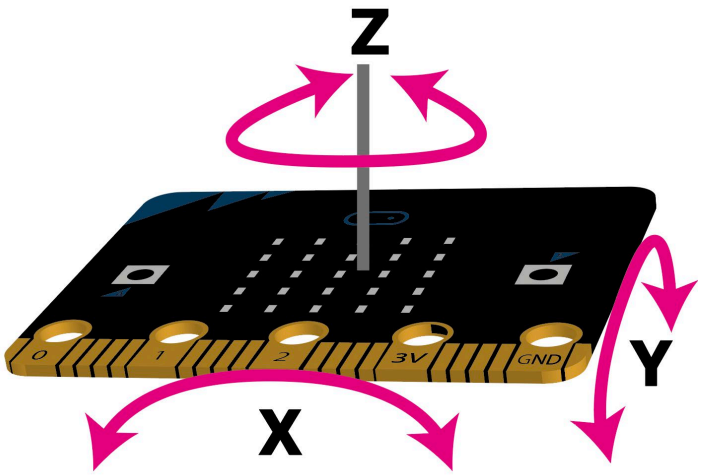
/forever
//if Button A is pressed/
/show Icon „happy Smiley“//
//if Button B is pressed/
/show led „sad Smiley“//
//if Button A+B is pressed/
/show leds „empty“//
/

11



cooridnates for the LED-display on the Micro:bit

12

	*/forever* *//if button A is pressed/* */plot x2 y2//* *//if button B is pressed/* */plot Pixel x2 y2 aus//* */*
13	
	*/forever* *//wenn /acceleration (mg) X < 0//* */zeige Icon "happy smiley"//* *//wenn /acceleration (mg) X ≥ 0//* */zeige Icon „sad smiley"//*
14	
	***Axis directions on the Micro:bit***