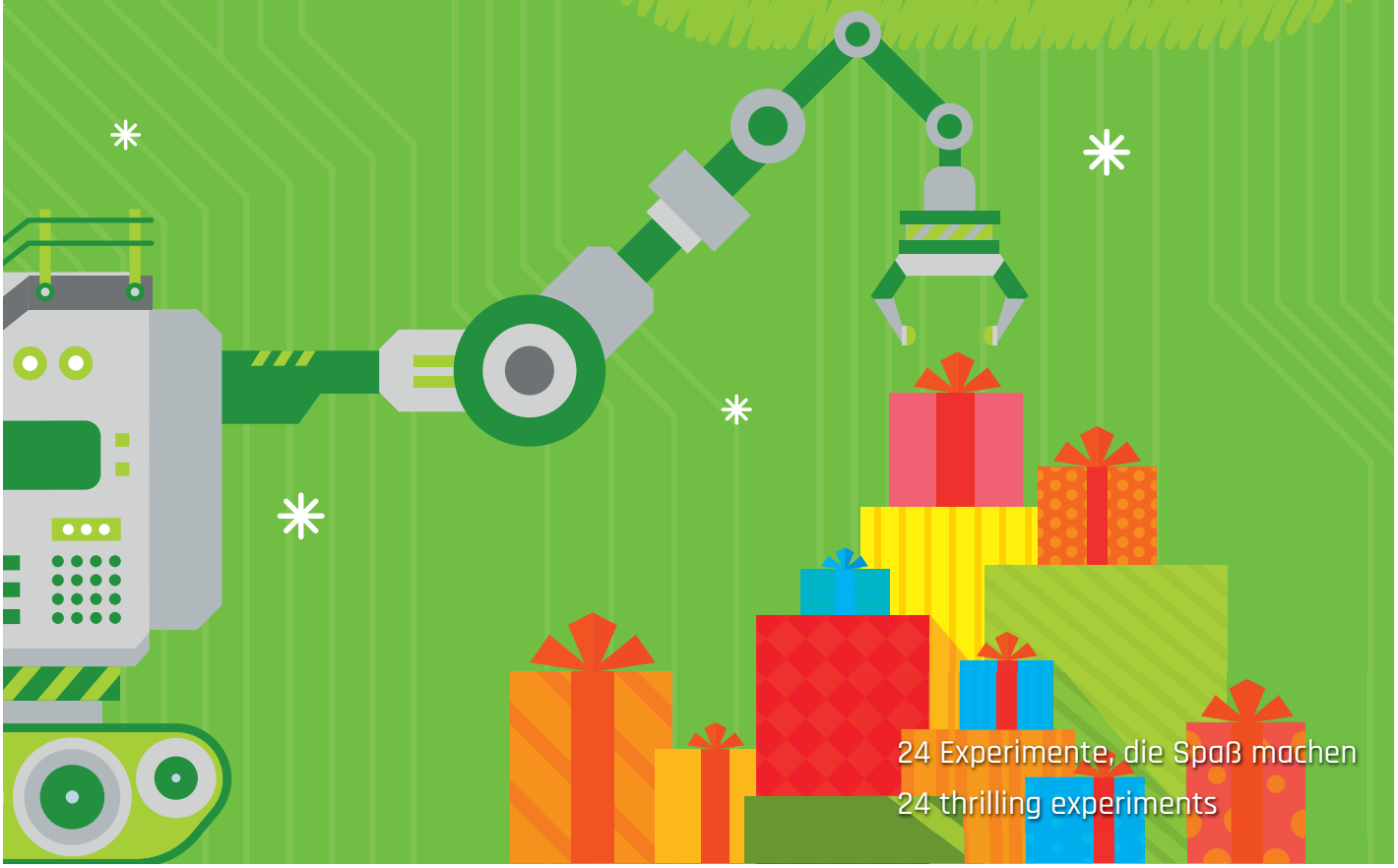
MAKER FACTORY

ADVENTSKALENDER FÜR MICRO:BIT

ADVENT CALENDAR FOR MICRO:BIT

COMPATIBLE WITH
BBC micro:bit



24 Experimente, die Spaß machen
24 thrilling experiments

Overview of all experiments

micro:bit-Advent Calendar 2019

The micro:bit is an experimental board and very easy to code. Though originally developed for use in schools, it can be utilized in many other experiments and games.

Obviously, the micro:bit is not the first programmable board. Just a few years ago, the microcontroller and single-board computer programming was a task for engineers. However, the well-known Raspberry Pi and Arduino boards made this technology accessible for everyone.

Unlike the Raspberry Pi, the micro:bit is not a real computer. It is a microcontroller board that simply processes a single program that was previously created on a computer and then uploaded via USB cable, and it is thus comparable to the Arduino and likewise very easy to code.

Precautions

Do not connect just any connectors and wait and see what happens.

Not all connectors can be freely configured. Some are pre-programmed for power and other purposes.

Some connectors are directly connected to microcontroller connector lines. A short circuit could totally ruin the micro:bit - at least in theory. The boards are amazingly stable against circuit errors. Thus when you connect two pins via an LED, you must always interpose a series resistor, if it is not already installed in the LED.

1. Day

Day 1

Today in the Advent Calendar

· Alligator clip cable

Alligator clip cable

You can employ these cables as sensor contact points using coins, spoons or other metal objects. Attach one alligator clip to the object and the other to one of the five contact surfaces with the holes on the micro:bit. But be careful not to cause a short circuit with one of the adjacent small contacts. Preferably, stick one side of the alligator clip through the hole in the board.

Preparing the micro:bit

For the micro:bit to work you will need:

· Computer with Windows 10. 8.1, 7 (or Mac OSX)

· Micro USB cable

The computer and the micro:bit are connected via a micro USB cable. You probably won't need to buy such a cable; most smartphones use this connector type. The cable is used as power supply and also for the transmission of data.

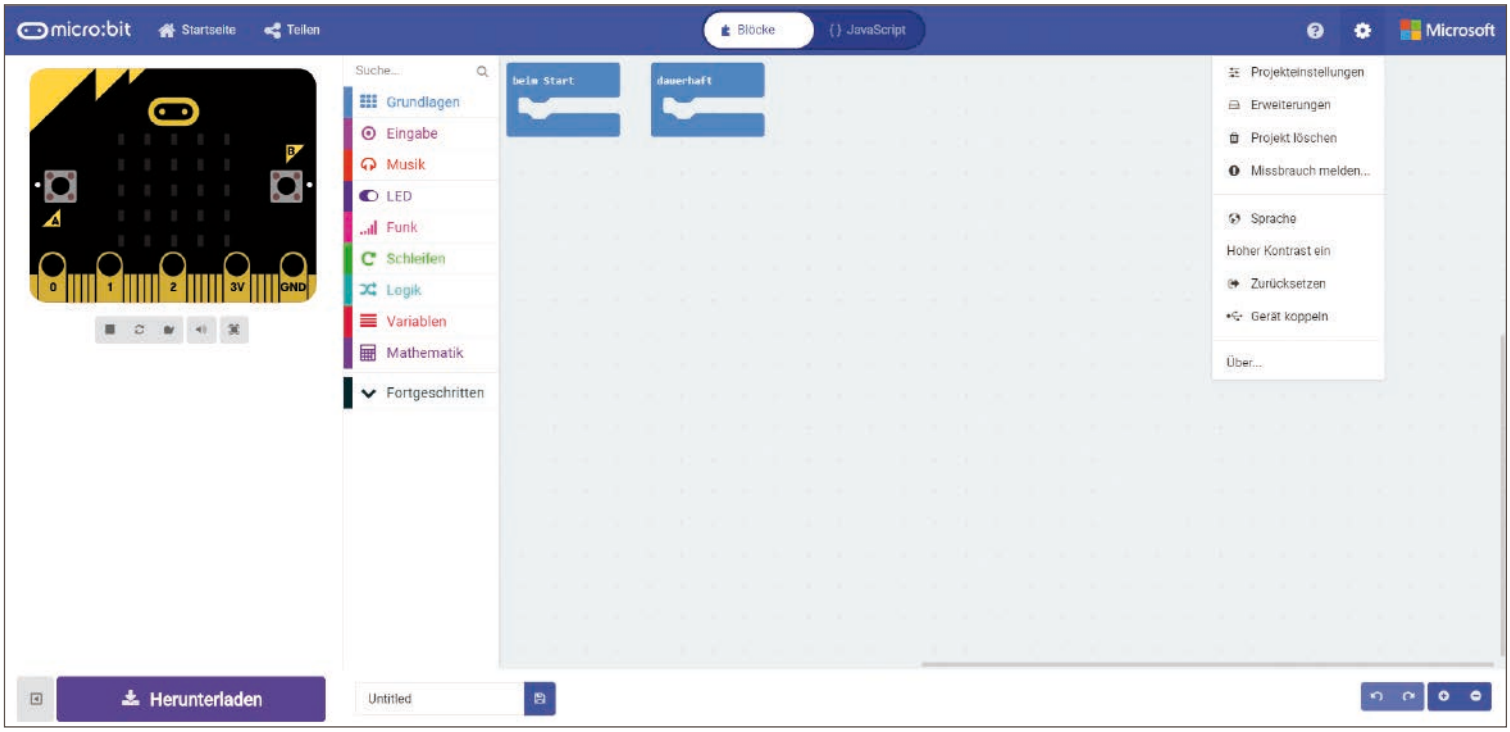
Connect the micro:bit via the USB cable to a free USB port on your computer. The driver is automatically installed on the first run and then the micro:bit will show up as a drive in Windows Explorer.

Once the driver is installed, you will find the micro:bit under the drive name MICROBIT and a previously unused drive letter in Windows Explorer view. However, the micro:bit is not a normal drive. It only stores one executable file at a time. You cannot see the file on the drive in Windows Explorer. It is also not possible to save other file types to it.

The Makecode editor

We use the Microsoft Makecode programming tool - **makecode.microbit.org** for our Advent Calendar programs. In Makecode you do not need to type any program code. You simply built it by drag-and-drop blocks. The Makecode editor runs in your browser. There is no need to install a program.

Click on the settings symbol in the upper right corner, select **Language** in the menu and then choose the language **English**.



Setting the language in Makecode Editor.

Top left you see the micro:bit that will play the program. This simulator will also show instantly any changes made to the program at any time, so you don't have to transfer the program to your "real" micro:bit.

The square symbol is used to stop the simulator and the triangle to start it again.

The available code blocks in the Makecode Editor are arranged in groups.

When you drag a block from the block palette into the workspace, it will first appear gray. The blocks will take on their original colour once the block arrangement forms an executable program. Delete a block by simply dragging it from the workspace back into block palette.

The blocks are arranged in a way that they only fit together, if the structure works for the program. Hence, you don't have to worry about the dreaded syntax errors that programmers keep struggling with day-to-day.

Every new program has two default blocks:

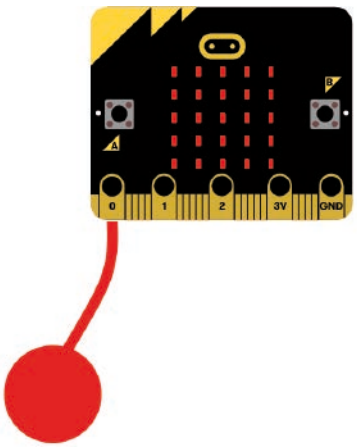
· **on start** – all blocks in this bracket are executed once at startup. Once transferred to the micro:bit, the program will start automatically. The program also restarts, if the button on the back is pressed or if power is reconnected.

· **forever** – After the blocks **on start** have been executed, the code in the bracket **forever** runs as infinite loop until the connector is unplugged.

The Advent Calendar programs
The programs we are using are available as a download at: **bit.ly/c-Advent Calendar-microbit-19.** Unzip the archive into a directory on your hard drive. The programs in the download are numbered by days. On the home page **makecode.microbit.org**, click **Import** to open a program. But you can also easily build the programs yourself with the help of the illustrations.

Merry Christmas

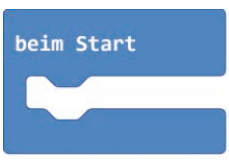
Components needed: 1x micro:bit, 1x alligator clip cable

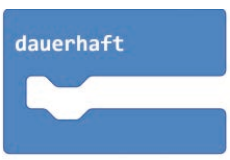


Connect the alligator clip cable to contact point 0 on the micro:bit. Connect the other end to a coin, metal spoon or any other conductive object.

The program

Program `microbit-01.hex` shows some text on an LED matrix, if the sensor is touched.





How does the program work

We won't need the **on start** and **forever** blocks for this program. The block i**f Pin P0 is pressed** in the **Input** group will execute all blocks within the bracket, if the 0 pin is touched on the micro:bit. That means, it is connected to ground, even if it's a 'weak' one.

Should the program not respond to the simple sensor touch, touch simultaneously the GND contact at the very right on the micro:bit.

Copying the program onto the micro:bit

In order to play the program not only on the simulator, but in real on the micro:bit, give it a name in the box at the bottom and then click on the large button **Download**. Save the downloaded file directly to the MICROBIT drive. The micro:bit should be connected to the computer via the USB cable, of course. The yellow LED starts flashing fast. Once copied there, the light of the LED will be steady on.

Shortly after the program has been copied, it will run automatically. The micro:bit can hold only one program at a time. Note that the *.hex file will no longer show up on the MICROBIT drive in Explorer view.

Browser settings for saving programs

Depending on your browser settings, you may have the option to select the directory, where the programs are saved to and more importantly, to choose a file name. It is essential that you can locate the file on your hard drive, because you will need to copy it from there onto the micro:bit.

Settings in Firefox

Click the menu symbol in the top right corner and select **Settings**. Go to the Settings page and in **Downloads** under **General** activate the check box **Always ask where to save a file**.

Settings in Google Chrome

Click the menu symbol in the top right corner and select **Settings**. Click **Advanced** at the bottom. Scroll down and under **Downloads** tick **Ask where to save each file before downloading**.

Settings in Microsoft Edge

Click the menu symbol in the top right corner and select **Settings**. Under **Downloads** tick **Ask where to save each file before downloading**.

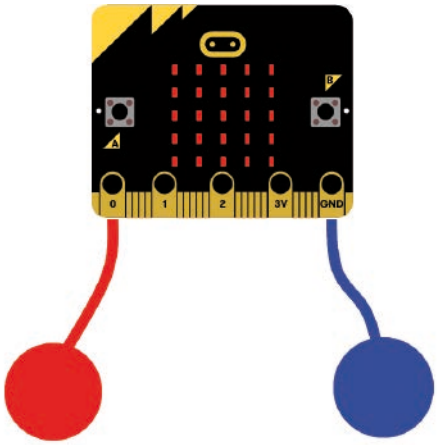
Day 2

Today in the Advent Calendar

· Alligator clip cable

Merry Christmas

Components needed: 1x micro:bit, 2x alligator clip cables

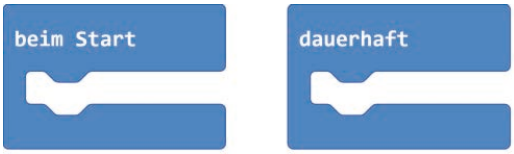


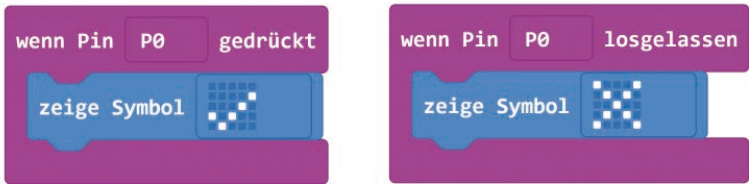
Attach one alligator clip cable to contact 0 on the micro:bit and the other to the GND contact. Attach coins, metal spoons or other conductive objects to the other ends.

The two sensor contacts will make sure that the program responds even though grounding is weak, if both sensors are touched simultaneously.

The program

Program `microbit-02.hex` will show a tick mark on the LED matrix when a sensor is touched and upon its release an X symbol.





How does the program work

The **show symbol** block in the Group **Basics** shows predefined symbols on the LED matrix. That works much faster than displaying your self-defined LED patterns. If you click on the symbol on the right side of the block, the selection palette with all the pre-defined symbols pops up.

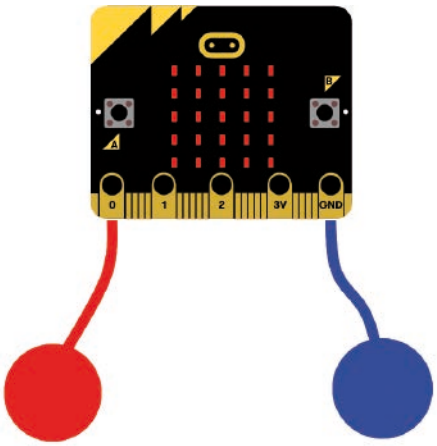
The block **if pin P0 is released** processes the blocks within the bracket whenever the contact 0 sensor is released, in other words, when there is no longer a connection to ground.

3. Day

Day 3

Today in the Advent Calendar

· 2x wire electrodes



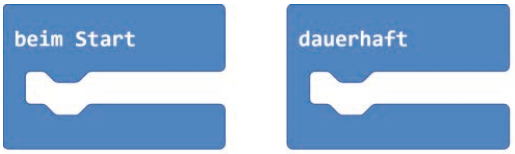
Numeral dice with bananas or apples

Components needed: 1x micro:bit, 2x alligator clip cables, 2x wire electrodes

With the wire electrodes you can use bananas, apples or similar objects as sensor contacts. Stick one electrode into a fruit and connect an alligator clip cable to it.

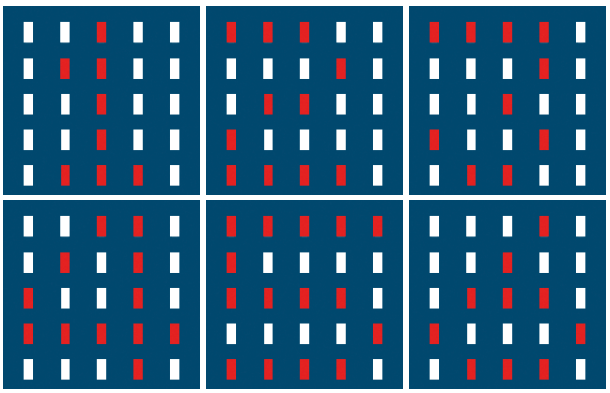
The program

There are many games that use dice but every so often there are no dice at hand. The program `microbit-03.hex` demonstrates how easy it is to program a die with the micro:bit. The first very simple version of the dice program will display a number between 1 and 6, when you touch the sensor contact – such as the banana.



The block **select a random number between ... and...** in the group **Mathematics** generates a random number within the given range of numbers. Simply drag this block from the Group **Basics** into the block's number field **show number ...**.

If you now touch the sensor, the micro:bit will show a random number between **1** and **6**.



How are random numbers generated?

We assume that nothing in a program happens accidently. After all, everything is planned. So, let's see how random numbers are generated by a program? Take a large prime number and divide it by any value. If you increase the divisor gradually, numbers from on an nth decimal digit will become barely predictable and seemingly erratic. Although this result appears to be at random, it can be reproduced at any time through an identical program or multiple calls to the same program. But, if you take a number composed of some of these digits and divide this number by a number that represents the second of the current time or the content of any memory location on your computer then you will get a result that cannot be reproduced. That is what we call a random number.

Day 4

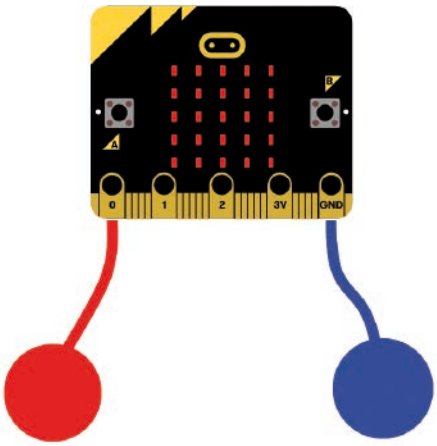
Today in the Advent Calendar

· Modelling clay

Modelling clay is perfect for forming sensor contacts because they are as conductive as human skin. However, a piece of modelling clay has a larger more handy contact surface than a wire. Consequently, you won't lose connection easily when you touch it. Insert the alligator clip of each of the two cables into one piece of dough and fix the other ends to the 0 and GND pins on the micro:bit.

Real dice with pips

Components needed: 1x micro:bit, 2x alligator clip cables, 2x modelling clay



The program

Yesterday's program suits our task at hand, but our dice didn't look like a real dice. The program `microbit-04.hex` uses code that is a bit more complex, but it produces a real dice on the graphic display. And every time you touch the modelling clay contact it shows a new die roll result.

How does the program work

The new dice will not display the result as a number, but depending on the count, it will display one of six images.

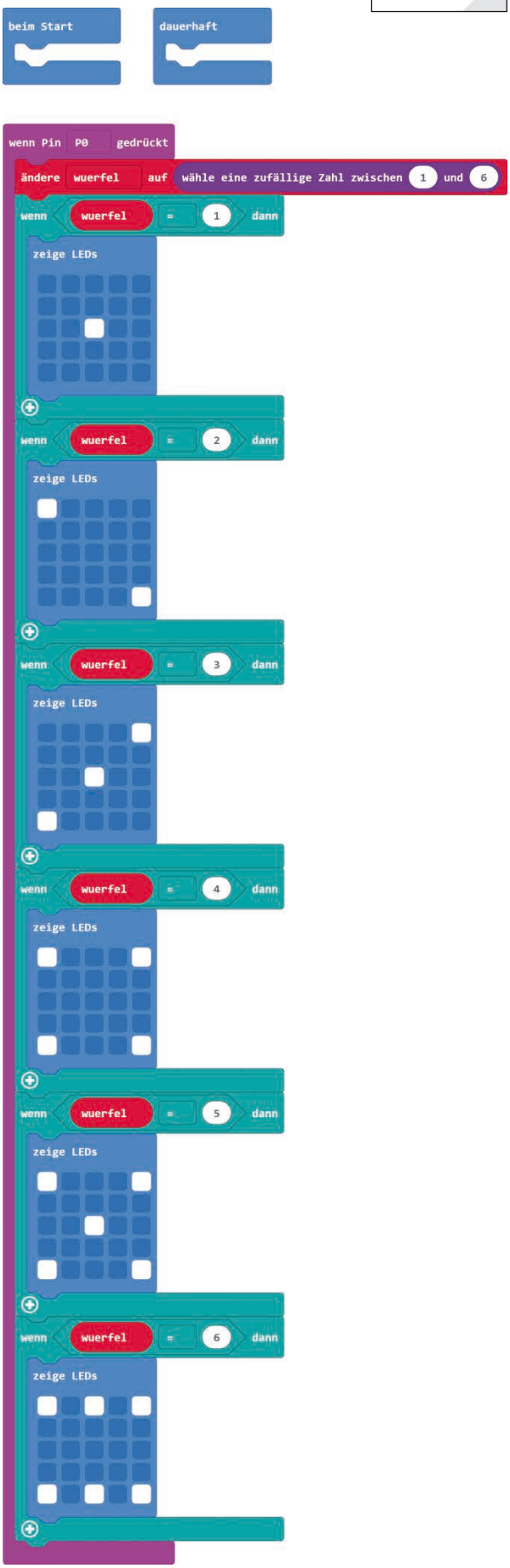
The randomly determined number will not be shown immediately, but is initially stored in the variable **dice**.

Variables in the Makecode editor

Variables are tiny memory locations to recall a number or something else during a program run. Upon program exit, these variable memories are automatically cleared. You will need to create these variables first in the block palette **Variables** in the Makecode editor with the button **Make a variable** before they can be used. Then you drag the symbol of the newly created variable into a designated field of a block in the program. The **Variables** group has two different blocks with different functions for modifying the variable, but the blocks are also easy to confuse.

change variable to ... assigns a specific value to the variable

change variable by ... increases the variable by a certain value.



The program has six different **if ... then ...**-queries in the **Logic** Group. Blocks within the brackets of this query are executed whenever the query condition is **true**.

All of these queries use the block **... = ...** from the **Logic** group to check whether the value of the variable **dice** matches one of the six possible numbers.

If this is the case, the corresponding die pattern is shown on the LED matrix. To do this we use the block **show LEDs** from the **Basics** group. You can set the desired LED pattern by clicking there.

Duplicate blocks

You do not necessarily need to build similar program blocks like the six **if ... then ...** queries from scratch. Just right-click on a block and select Duplicate in the context menu. A copy of the block and all nested blocks are created automatically, which you then drag to the desired position in the program and continue editing.

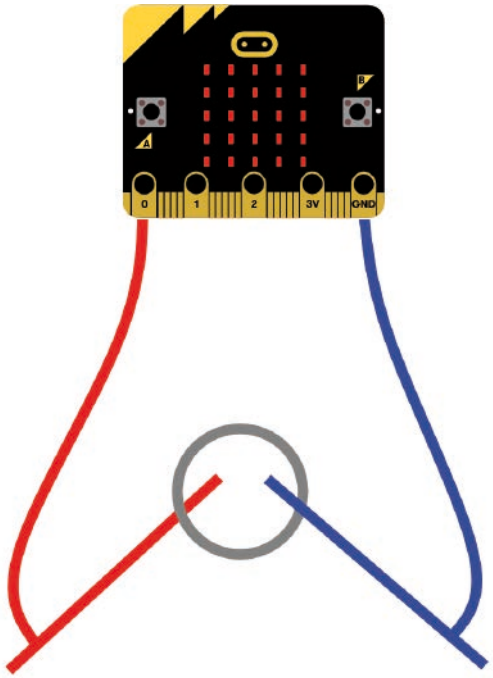
Day 5

5. Day

Today in the Advent Calendar

· Lead wire

Today our Advent Calendar presents us a lead wire. We are going to use it later on to build different wire connections between components.



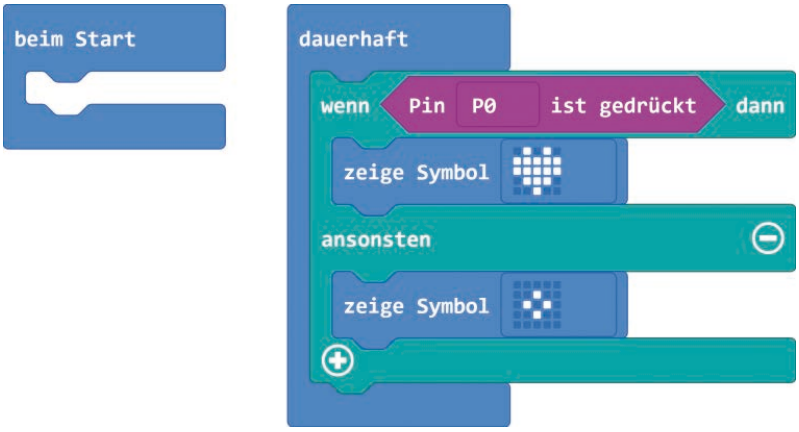
Game of skill

Bend the lead wire to form rings of different shapes. But first you need to remove the insulation with a sharp knife. Then attach a wire electrode to each of the two alligator clip cables. Now try transfer the wire rings from one electrode to the other so that you briefly touch both. At that moment, the micro:bit shows a heart on the LED matrix. Make sure to grab only the insulated alligator clips and do not touch the electrodes to prevent closing the connection with your hand.

Components needed: 1x micro:bit, 2x alligator clip cables, 2x wire electrodes

The program

The program `microbit-05.hex` shows us a different way how to query contact pins. This method often works faster, because at some point in the program, the contact is evaluated without a query that runs constantly in the background interrupting the program.



How does the program work

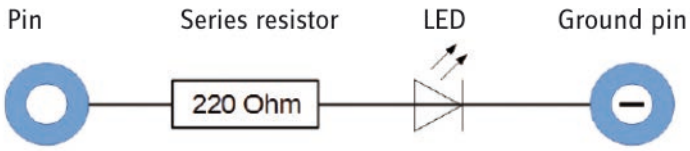
Within a **forever** loop the **if ... then ... else ...**block repeatedly queries Pin 0. The blocks in the first bracket are executed whenever the query returns **true** and the blocks in the second bracket if the query returns **false**. In this program, the heart symbol is displayed when the contact to ground is closed; the small diamond is displayed when pin 0 is not connected to ground.

6. Day

Day 6

Today in the Advent Calendar

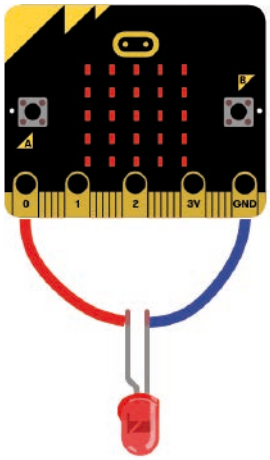
- red LED with series resistor



Circuit diagram of a LED with series resistor.

LEDs

LEDs (light-emitting diodes) glow, if current flows through in current flow direction. LEDs are illustrated in circuit diagrams with an arrow-shaped triangle symbol indicating the flow direction from the positive terminal to the negative terminal or to the ground wire. Because an LED has only a very low resistance, it permits almost any amount of current to flow through in flow direction. In order to limit the current flow and prevent the LED from burning out, it is usually necessary to install a 220 Ω resistor between the pin used and the LED anode, or between the cathode and the ground pin. But this series resistor also protects the output of the micro:bit from high currents. Our Advent Calendar LEDs come with the series resistor already installed and you can thus connect them directly to the pins.



Connecting an LED but what direction?

The two LED legs have different lengths. The longer leg is the positive terminal, the anode, the shorter leg is the cathode. Easy to remember: The plus sign has a dash more than the minus sign, so that would be the longer wire. Also, most LEDs are flat at the minus side, comparable to a minus sign. Easy to remember: cathode = cut short = flat edge.

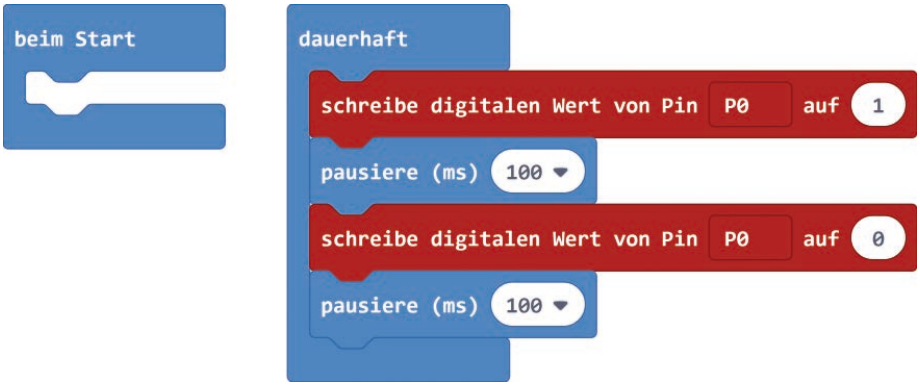
Blinking LED

Components needed: 1x micro:bit, 2x alligator clip cables, 1x red LED

Connect the LED with two alligator clip cables to the pins 0 and GND. The shorter leg must be connected to the GND pin.

The program

The program `microbit-06.hex` causes the LED to blink continuously.



In a **forever** loop, the LED is alternately turned on and off and in between, the program waits each time for 100 milliseconds.

How does the program work

Block **digital write pin P0 to...** switches pin **P0** on or off. If the numeric field shows **1**, the pin is switched off, if the value is **0,** the pin is switched on.

This block is a bit confusing. In essence, **digital write pin P0 to...** means, that this block assigns a logic value to a pin or sets a pin to a logic value. The logic values **true and false** which would have been more reasonable cannot be used, because the block uses the numerical values **1** and **0.**

Day 7

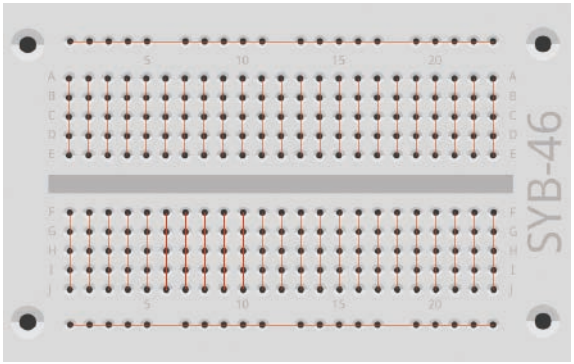
7. Day

Today in the Advent Calendar

· Breadboard (SYB 46)

Breadboard

The breadboard in the Advent calendar helps us to quickly build an electronic circuit without having to solder. Electronic components can be plugged directly into the holes of the breadboard.



The breadboard connections.

On our breadboard, the outer horizontal rows of contact points (X and Y) are all connected. These rows of contact points serve often as the plus and minus terminals to power the circuits. The other rows feature contact points, where each five contacts (A to E and F to J) are connected transversely. In the middle of the board is a gap, so that also larger components can be plugged in and wired to the outside.

Jumpers and connectors for alligator clips

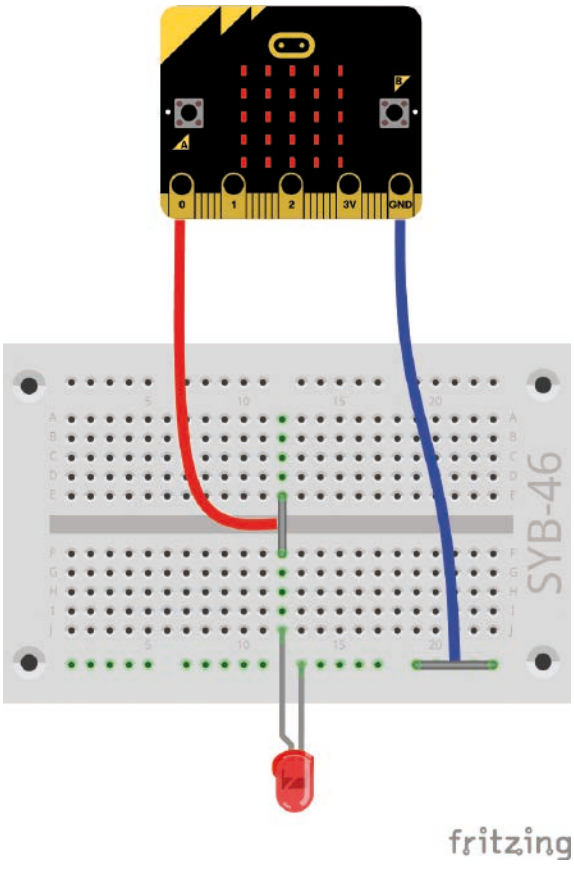
Lead wires are used to bridge closely spaced rows of contact points on the breadboard. These jumpers can also be used to connect alligator clip cables to the breadboard.

Cut the wire to the appropriate length with a small side cutter depending on the experiment. In order to fit the wire ends better into the breadboard, we recommend cutting them at an angle, so that a kind of gore is created. Strip the insulation off from the entire length with a sharp knife. You can attach alligator clips to these jumpers. This is especially easy if a bridge needs to be created that jumps over the "ditch" in the middle of the breadboard.

Turn LED on and off with a switch

Components needed: 1x micro:bit, 1x breadboard, 2x alligator clip cables, 1x red LED, 2x bare jumper wires

If you push button A on the left on the micro:bit, the LED turns on; if you push button B to the right, the LED turns off.



The program

The program `microbit-07.hex` causes the LED to blink continuously.

How does the program work

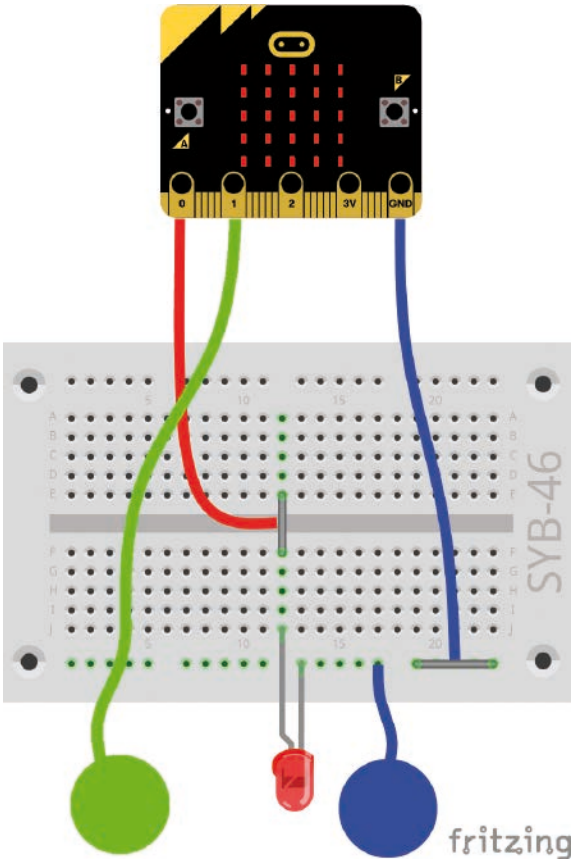
The program consists of two blocks querying the two buttons. If button A is pressed, digital value 1 is written to pin P0 and the LED is switched on. If button B is pressed, the value 0 is written to the pin.

8. Day

Day 8

Today in the Advent Calendar

· Alligator clip cable



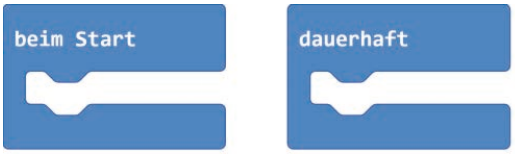
LED flashes on touch

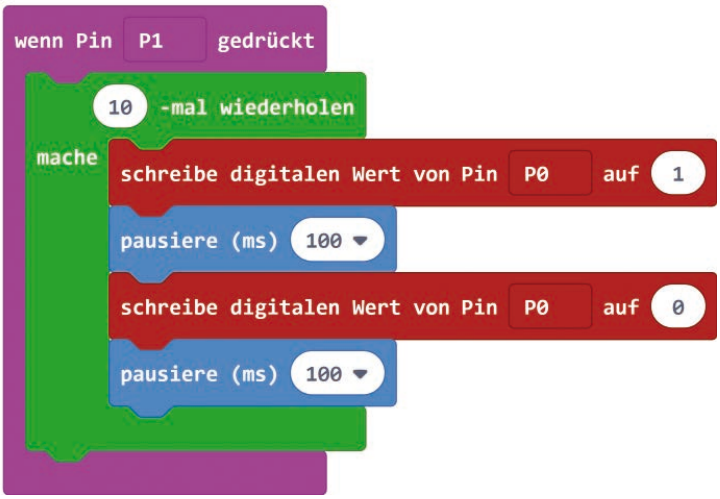
Components needed: 1x micro:bit, 1x breadboard, 3x alligator clip cables, 1x red red LED, 2x bare jumper wires, 1x lead wire, 2x modelling clay

In order to make contact to ground, the clay sensor needs a longer lead wire that connects to the ground strip on the breadboard. You only need to strip off 1 cm insulation on both ends.

The program

The program `microbit-08.hex` causes the LED to blink 10 times on contact.





How does the program work

The program waits until pin P1 is touched. In the selection field of the block **on pin ... pressed**, you can select the pin to be used.

The block ... **repeat times** ... from the **Loop** group will repeat the blocks within the bracket as many times as defined in the above condition.

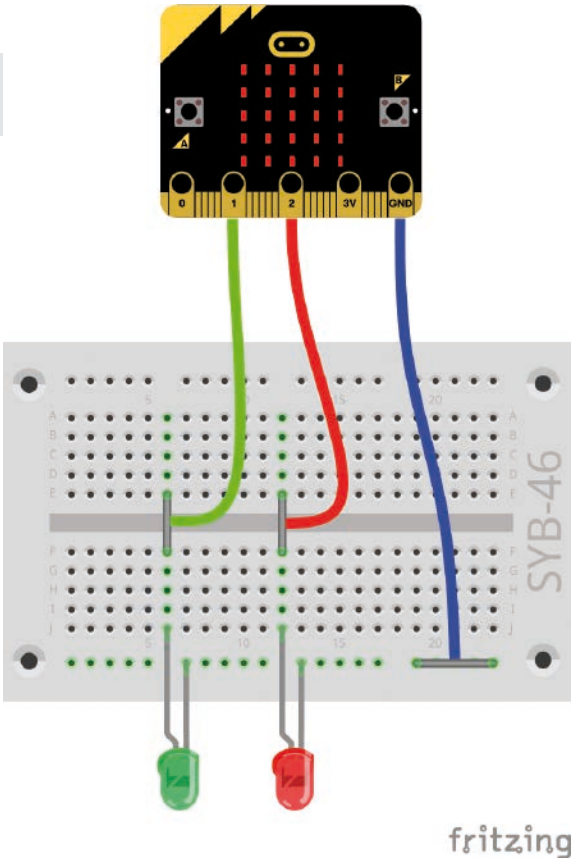
Day 9

Today in the Advent Calendar

· green LED with series resistor

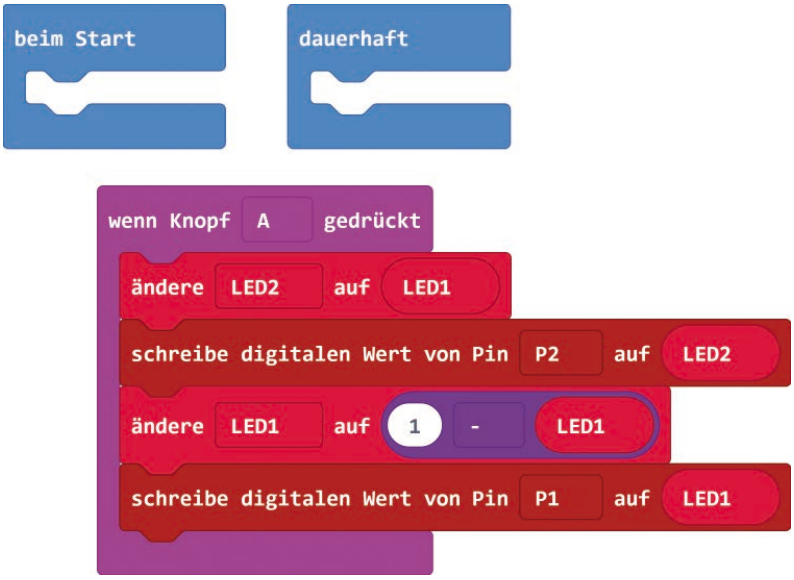
LEDs toggled with a button

Components needed: 1x micro:bit, 1x breadboard, 3x alligator clip cables, 1x red LED, 1x green LED, 3x blank jumper wires



The program

The program `microbit-09.hex` switches one of the two LEDs on and the other off and vice versa when button A is pressed.



How does the program work

The program will know whether the next button pressed will switch an LED on or off because the current switch states are stored in the memory in the two variables **LED1** and **LED2**.

So, if button A is pressed, first of all, the value of **LED2** is set to the current status of **LED1**. LED2 at pin P2 will then switch on or off according to variable **LED2**.

Then the variable **LED1** is set to the other switching state. This is simply done with block **1 – ...** from the **Mathematics** group. And so 1 changes to 0 and 0 changes to 1.

Value before conversion	Formula	Result
0	1 - 0	1
1	1 - 1	0

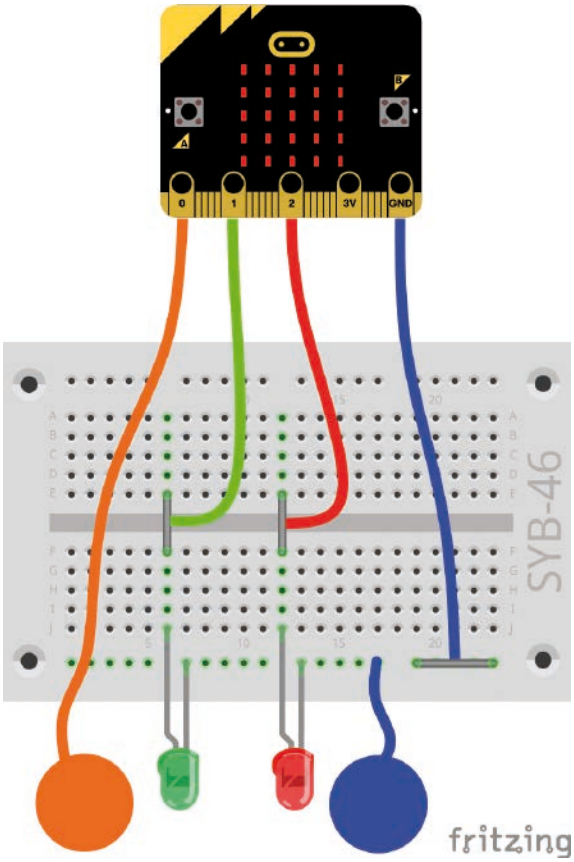
This new variable **LED1** value is then assigned to pin P1 in order to switch the LED accordingly.

10. Day

Day 10

Today in the Advent Calendar

· Alligator clip cable



Mobile earth ground resistance tester

Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 1x red LED, 1x green LED, 3x bare jumper wires, 1x lead wire, 2x modelling clay

Connection to the earth is not always good. The level of resistance between the hand that touches the sensor and the ground depends on many things, but is affected for all by the kind of shoes you are wearing or the type of flooring you are standing on. The earth ground connection is excellent when you are standing barefoot in wet grass, but it also works well on stone floors. A mobile earth ground tester will indicate if your ground condition is good enough for the sensor contact to function.

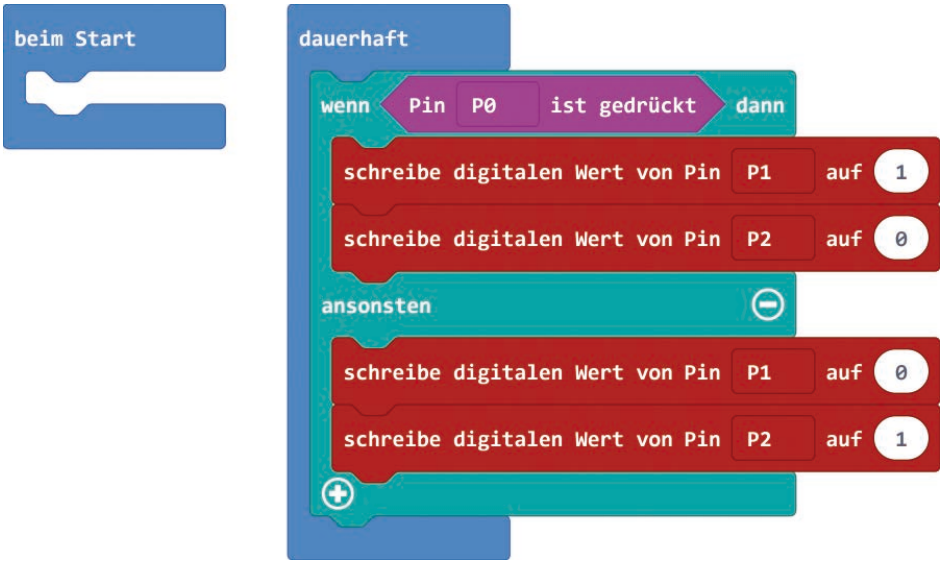
Place the clay sensor which is connected to ground by a long piece of lead wire on the floor or on appropriate pieces of furniture and hold the other clay sensor in your hand.

The program

The program `microbit-10.hex` causes the green LED at pin P1 to glow, if there is a connection between the sensor contact at pin P0 and to ground. Otherwise the red LED at pin P2 will glow.

How does the program work

Within the **forever** loop the **if ... then ... else**- queries whether contact point **P0** has been touched. To do this, we will use block **Pin P0 pressed** from the **Input** group. Depending on the result of the query, either the green or red LED will turn on. The other LED will turn off accordingly.



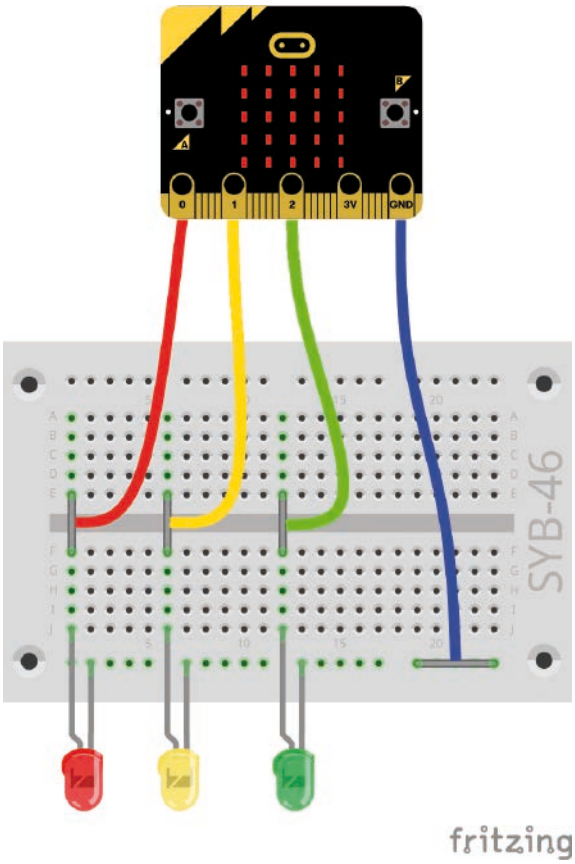
Day 11

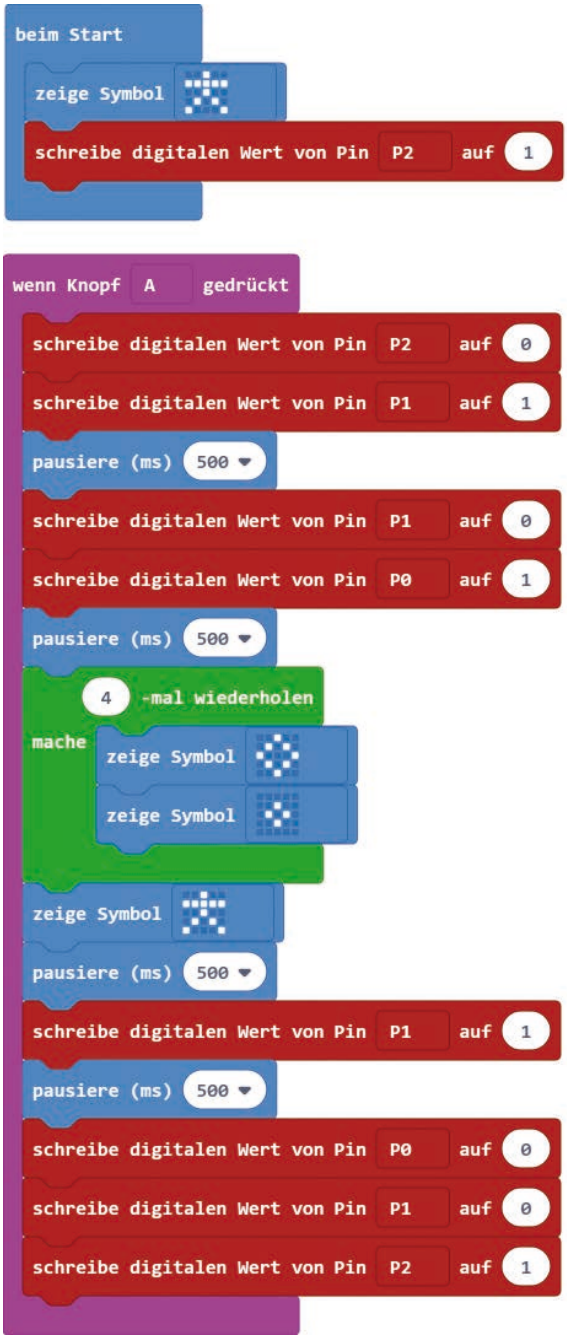
Today in the Advent Calendar

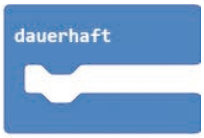
· Yellow LED with series resistor

Traffic signal with pedestrian lights switched by button

Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 1x red LED, 1x yellow LED , 1x green LED, 4x jumper wire blank







By means of three LEDs, a simple traffic signal with pedestrian lights will be shown on the LED matrix of the micro:bit. During the traffic signal's red phase, the pedestrian symbol will change on the LED matrix. Because the micro:bit has only three connection pins, the circuit cannot display a red/green pedestrian traffic light.

The program

The program `microbit-11.hex` starts the traffic signal sequence, when the button is pressed. The traffic light stays on green and the pedestrian symbol shows a standing traffic light man, if the button is not pressed.

How does the program work

On start the standing traffic light man is displayed on the LED matrix, the green LED on pin P2 is switched on. The other two LEDs are all switched off, as all pins are set to 0 by default upon program start. These settings stay as they are until someone activates the A button on the micro.bit.

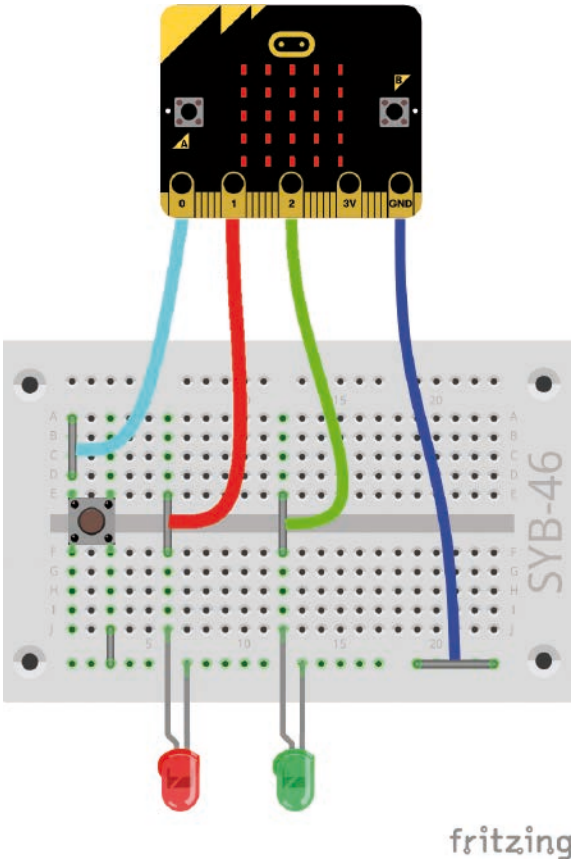
Now the traffic light signal turns from yellow to red. During the traffic light's red phase, a loop changes the symbol on the LED matrix 40 times. There are no waiting times between switching operations. Because it takes a few milliseconds to display the symbol, it is easy to see the flashing. After that the standing traffic light man reappears, and the traffic light cycle continues and moves from red//yellow to green. In this state, the traffic signal pauses again until someone presses the button.

12. Day

Day 12

Today in the Advent Calendar

· Button



Button

You can also connect external buttons to pins P0, P1 and P2 and use them in addition to the buttons permanently installed on the micro: bit. The Advent Calendar today has a button, which you can put straight onto the breadboard. The button has four connection pins. The two legs facing each other at the larger distance are interconnected. All four legs are connected as long as the button is pressed. Upon release, the connection immediately breaks.

External button to toggle LEDs

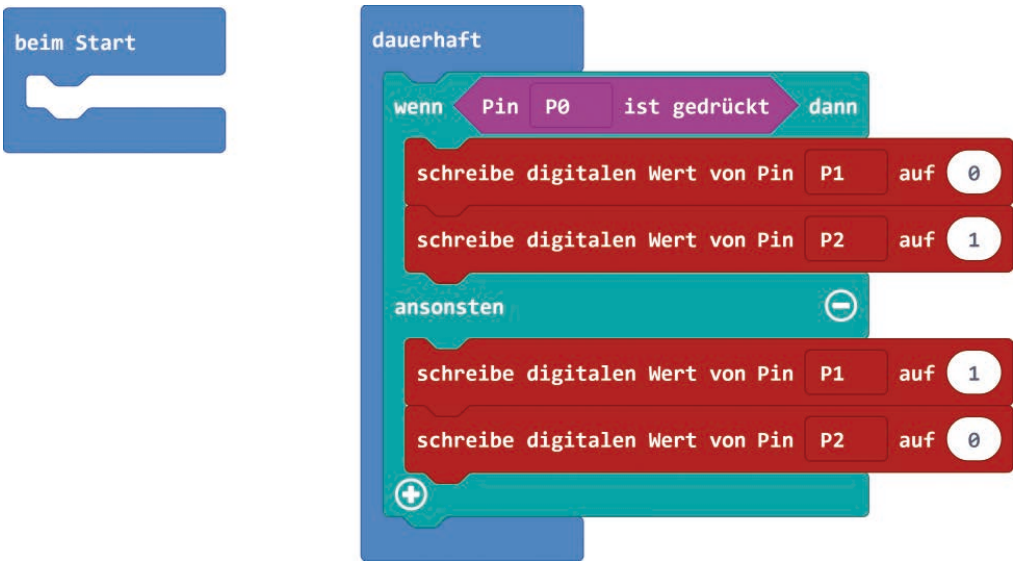
Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 1x red LED, 1x green LED, 1x button, 5x bare jumper wires

The program

The program `microbit-12.hex` switches LEDs by means of a button. If the button is pressed, the green LED at pin P2 turns on. The red LED at pin P1 glows, if the button is not pressed.

How does the program work

The program follows the known pattern. The query **if ... then ... else ...** checks whether the button is pressed, in other words, whether P0 pin is connected to ground. If this is the case, the LED at pin P1 is turned off and the LED at pin P2 is turned on and vice versa, if this is not case.



Day 13

Today in the Advent Calendar

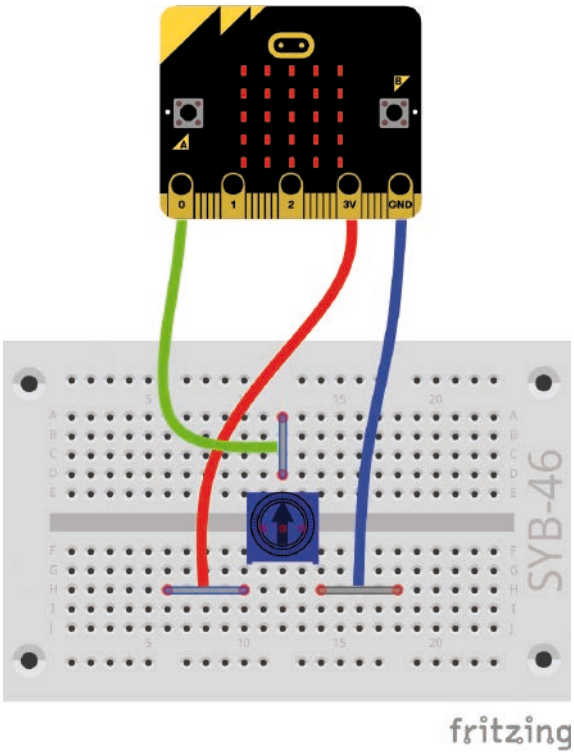
- 15 kΩ potentiometer

Potentiometer

Today's Advent Calendar potentiometer is a resistor that you can set to values between 0 Ω and 15 kΩ by turning the knob.

Analogue level indicator on LED matrix

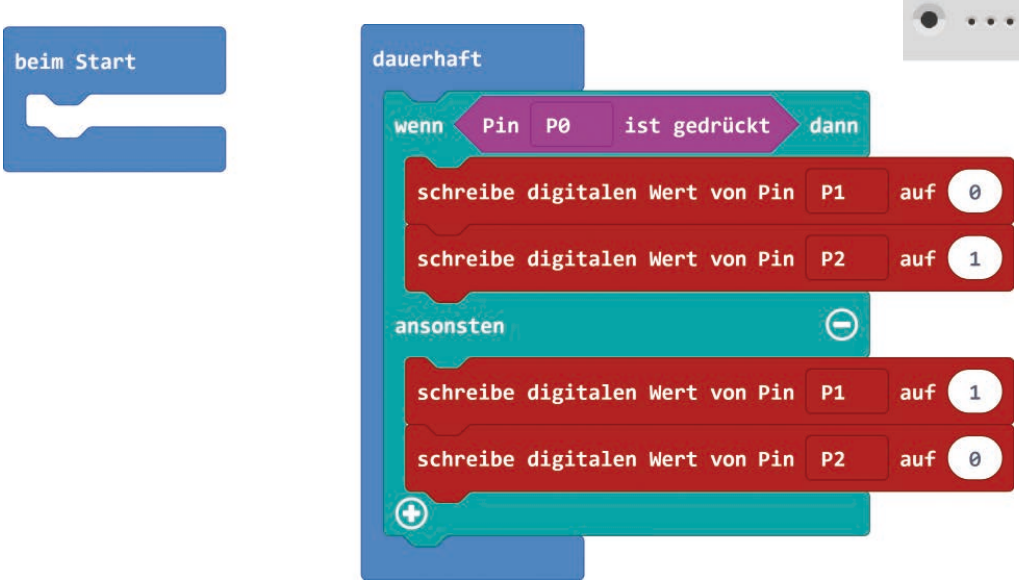
Components needed: 1x micro:bit, 1x breadboard, 3x alligator clips wire, 1x potentiometer, 3x bare jumper wires



The three pins P0, P1 and P2 will also read analogue input values. These are represented as numeric values, depending on the applied voltage, ranging between 0 and 1023.

The program

Analogue values can be roughly measured at a glance on the level indicator. Such indicators, which consist of a number of LEDs are used in volume or temperature controls, for example. The `microbit-13.hex` program will display the value set on the potentiometer as a bar graph on the LED matrix..



How does the program work

The program uses the block **plot bar graph of ... up to ...** from the **LED** group. This block will represent the analogue value, which is read in with the block **analogue read pin ...** from the **Pins** group, as a bar graph on the LED matrix. The second value in this block indicates the maximum possible bar graph value that can be used to scale the values to the LED matrix. One row of LEDs thus equals 20% of the maximum possible value. Interim values are indicated by filling the next row above starting in the middle.

14. Day

Day 14

Today in the Advent Calendar

· Resistor 10 kΩ (brown - black - orange)

Resistors and their colour coding

Resistors are used for example as current limiter for sensitive electronic components or as series resistors for LEDs. Resistance is measured in ohm. 1,000 ohms are one kilo ohm, in short kΩ. 1,000 kilo ohms are one mega ohm, or MΩ. Most often the omega symbol Ω is used to express ohm.

The colour rings on the resistor indicate the resistor value. They are much easier to detect than the tiny numbers which you can still find on very old resistors.

Colour	Resistor value in ohm			
	1. Ring (tens)	2. Ring (ones)	3. Ring (Multiplier)	4. Ring (Tolerance)
Silver			10^{-2} = 0,01	±10 %
Gold			10^{-1} = 0,1	±5 %
Black		0	10^{0} = 1	
Brown	1	1	10^{1} = 10	±1 %
Red	2	2	10^{2} = 100	±2 %
Orange	3	3	10^{3} = 1.000	
Yellow	4	4	10^{4} = 10.000	
Green	5	5	10^{5} = 100.000	±0,5 %
Blue	6	6	10^{6} = 1.000.000	±0,25 %
Violett	7	7	10^{7} = 10.000.000	±0,1 %
Gray	8	8	10^{8} = 100.000.000	±0,05 %
White	9	9	10^{9} = 1.000.000.000	

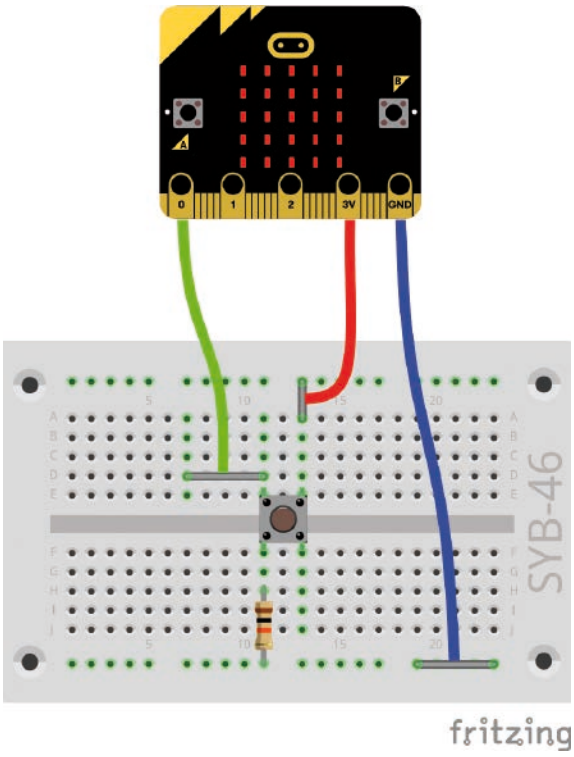
Most resistors have four such colour rings. The first two colour rings stand for the digits, the third indicates a multiplier and the fourth the tolerance. This tolerance ring is usually either gold or silver -those colours are not used for the other rings. Thus the reading direction is always unambiguous. The tolerance value itself is not really important in digital electronics. The table explains what these colour rings on resistors stand for.

The direction how a resistor is installed, does not really matter. On the other side, the direction an LED is installed is very important.

Switch with pull-down resistor

If the switch is in open state, the input to which the switch is connected is theoretically in an undefined state. A program that queries this pin may receive random results. In the case of digital input signals this is compensated by electronics integrated in the micro:bit.

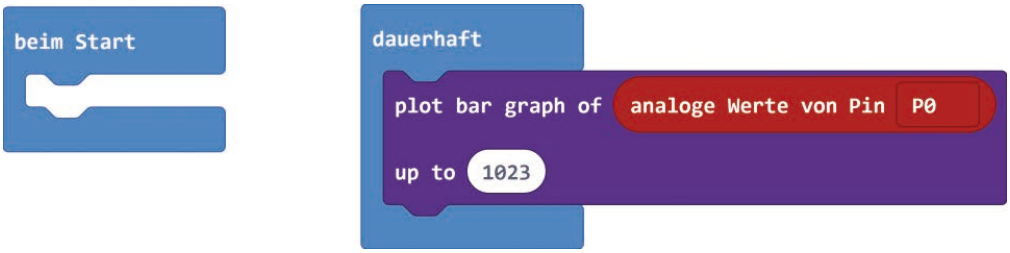
However, if you connect a switch to an input pin and then evaluate the analogue signal, the undefined state is clearly evident. To pre-empt this, you can connect a resistor of relatively high resistance - usually 10 kΩ - to ground. This so-called pull-down resistor drives the status of the input pin back to 0 V, if the switch is open. Since the resistance is relatively high, you don't need to worry about a short circuit while the button is pressed. This resistor connects +3 V and the ground wire when the button is in the closed state.



Components needed: 1x micro:bit, 1x breadboard, 3x alligator clip cables, 1x button, 1x resistor 10 kΩ (brown - black - orange), 3x bare jumper wires

The program

The program `microbit-14.hex` shows the analogue input value at pin P0 as bar graph on the LED matrix. The program is similar to the program of yesterday.



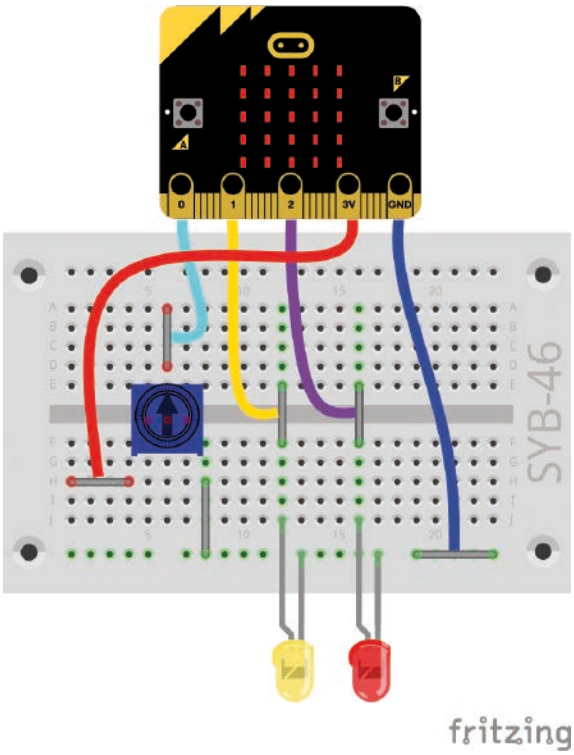
If you press the button, the displayed value changes from 0 to full scale, which is approximately 1023. When you remove the resistor, the bar graph won't even reach the 0 value on the graph, if the button is pressed. The now open analogue input will always have an indefinable value applied which is shown in approximately 20% of the bar height.

15. Day

Day 15

Today in the Advent Calendar

· Alligator clip cable



Alternating flashing light with adjustable speed

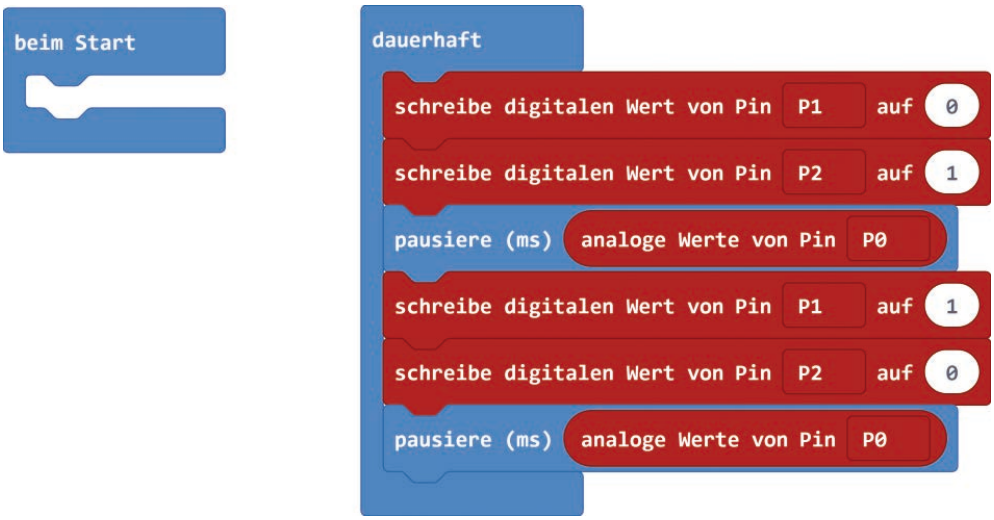
Components needed: 1x micro:bit, 1x breadboard, 5x alligator clip cables, 1x p, 1x red LED, 1x yellow LED, 6x bare jumper wires

The program

The program `microbit-14.hex` causes two LEDs to flash alternately. The flashing rate is set with the potentiometer.

How does the program work

The two LEDs at pins P1 and P2 are switched on and off alternately in a **forever** loop, causing one LED to turn on and the other off. Waiting intervals between switching operations are not fixed; at these points the analogue value of pin P0 is read in each time. The value is set with the potentiometer and can range anywhere between 0 and 1023. This will produce waiting intervals from 0 up to about one second.



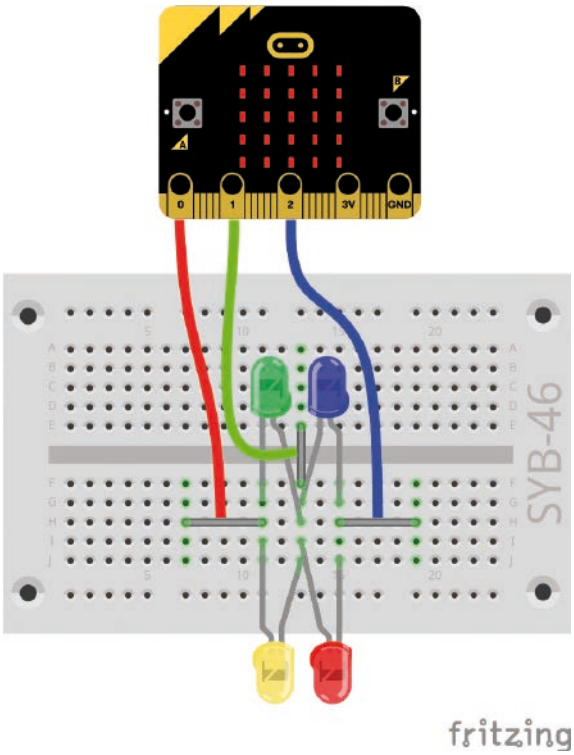
Day 16

Today in the Advent Calendar

· Blue LED with series resistor

Flashing colour pattern with four LEDs

Components needed: 1x micro:bit, 1x breadboard, 3x alligator clip cables, 1x potentiometer, 1x red LED, 1x yellow LED, 1x green LED, 1x blue LED , 3x bare jumper wires



Theoretically, you would only be able to connect three LEDs to the connection pins P0, P1 and P2 and have them flash alternating. However, when you switch the connections in turns as +3 V and 0 V, you will get several possible combinations.

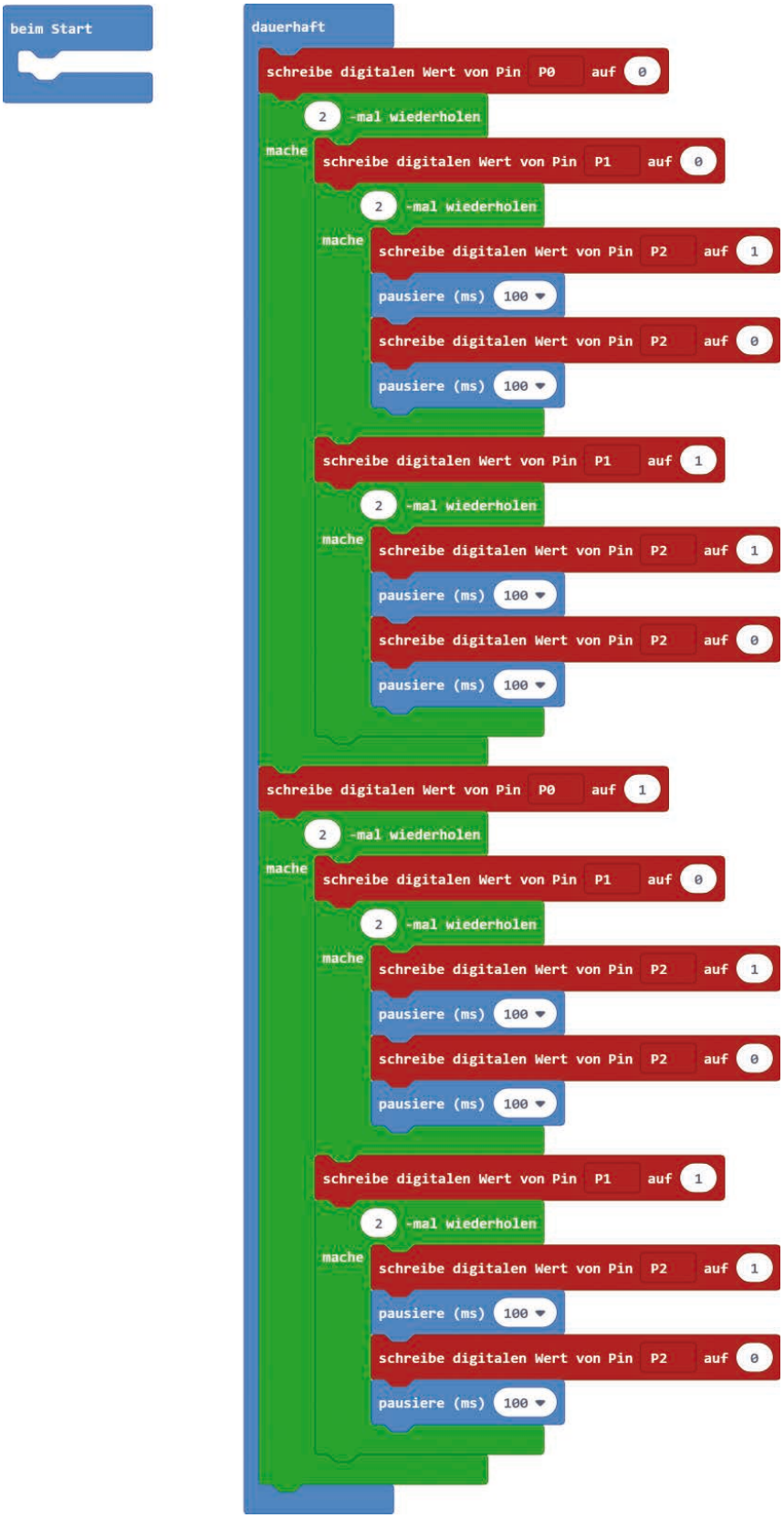
This time, the LEDs are not connected to the ground wire on the micro: bit, but between the pins P0, P1 and P2. The cathodes of the LEDs in the upper part of the figure are located on the left-hand side and the cathodes of the LEDs in the lower part are located on the right.

The program

The program `microbit-16.hex` causes four LEDs to flash in turns.

How does the program work

The **forever** loop has nested loops, which will set the three connection pins to 0 and 1 in turns. One LED will turn on if the pin at the anode is switched to 1 and the pin at the cathode is switched to 0.

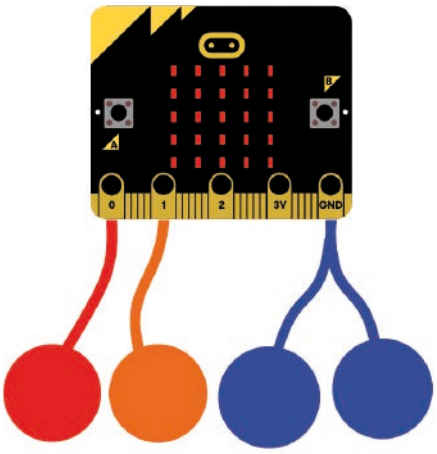


17. Day

Day 17

Today in the Advent Calendar

· Blue modelling clay



Quick response game

Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 4x modelling clay

The program

The program `microbit-17.hex` is a game that requires a quick response from the two players. The one, who touches the sensor contact first, when the heart lights up, scores a point. The one who scores five points has won the game. Each player has a clay contact as sensor. Also each player will have a ground contact, which can be held in one hand or you could place one on a conductive surface, such as a metal plate that could be touched by both players.

The best approach would be that both players always hold the ground contact in their hands and touch the sensor contact with the other hand at the right moment. But careful here; place the sensor contacts always on a nonconductive surface when not touching them, to prevent the sensors from coming into contact with the ground contact.

The rules of the game

· At the start of the game the symbols flash randomly.

· As soon as the heart appears, the players touch their sensor contact as fast as they. The one, who touches the sensor first, scores a point.

· If a player makes contact prematurely, the other player will get the point. Only one point is awarded in each round.

· The game is over, once a player has accumulated five points.

· The points are shown at the end of the game by means of the LEDs.

How does the program work

The game variables:

· **A** – points scored by player A.

· **B** – points scored by player B.

· **End –** Logic value that indicates that the end of the game has been reached. If this is the case, the sensor contacts are not queried any longer to prevent that a player who touches the senor will still score points.

· **index –** loop count.

· **Point –** Logic value that indicates whether in this round a point still needs to be awarded.

· **Game –** Logic value that indicates whether the heart currently glows. In that moment the player will receive a point when touching the sensor. On the other side, if the sensor is touched too early before the heart appears, the opponent will get that point.

The game runs in a bigger **on start** block. Initially, the scores of both players are set to 0 and the variables **end** and **game** are set to **false**. In other words: Now is not the right time to touch your sensor contact. If you do so, your opponent gets the point.

The game continues until one of the players scores five points. The main loop of the game is repeated as long as both players have less than five points. The condition in the ... **while... do**... loop consists of a block **... and ...** two blocks **... <5**.

At the start of each round, the variable **point** is first set to **true**, meaning, now you can again score a point. Once a player is awarded a point for that round, this variable is set to **false** preventing that a second point is awarded in this round.

In each round, three different symbols will flash in turn until the heart appears. This is randomly repeated up to four times and thus makes the game unpredictable.

After the symbols are flashing, we need to create the moment where points can be made through touching the sensor contact. To do this, the **game** variable is set to **true** and the heart symbol is shown. After a one second waiting interval, the **game** variable is reset to **false**. If during this second of time neither of the two players has touched the sensor contact, no point will be awarded in this round and the next round begins.

Points are always awarded whenever a player touches a sensor contact. Say, if player A (left) touches the sensor contact P0, and the end of the game has not yet been reached, **if ... then ... else** will query the **game** variable. If **true**, player A receives the point, if a point must be given due because contact to the sensor was made at the right moment of time. But if **false**, player B gets the point, if a point must still be awarded. In that case, Player A has touched the sensor outside the timeframe during which the heart is flashing.

In this case the **then** query checks whether the **point** variable is still **true**, in other words, it checks if it is indeed true that a point should still be awarded in this round. If true, score A is increased by 1 and the **point** variable is set to **false**. This will ensure that only the first contact with the sensor is scored as a point. If the other player touches the sensor a split second later, he will not receive a point anymore.

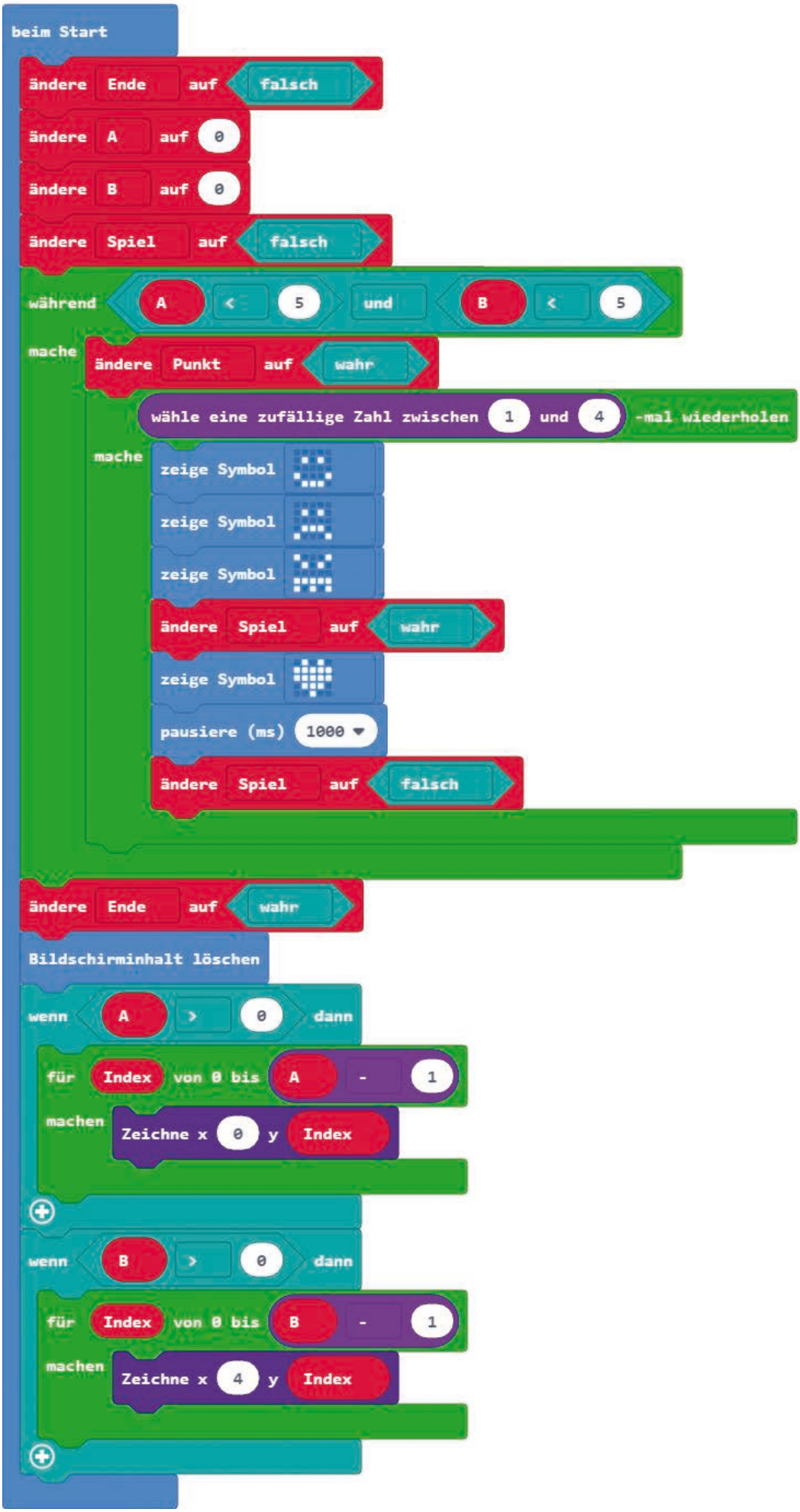
In the case **else** ..., when the **game** variable has the value **false,** the variable **point** is used to check whether a point should still be allocated in this round. But this point will go to the opponent. Player B's score is thus increased by 1.

Basically, the same happens, when Player B touches the sensor. If Player B touches the sensor contact at the right moment, score **B** increases by 1. Yet, if contact with the sensor is made prematurely, score **A** would increase by 1.

The main loop of the game runs until a player has five points. At this stage, the game is over, and the points will be indicated by two bars on the LED matrix. To do this, the last displayed heart symbol is deleted behind the loop by means of a **Delete screen content block**.

Say, Player A has achieved at least one point. In this case, one to five LEDs would glow in the left column of the LED matrix at x-coordinate 0. So, to illuminate the one to five LEDs, which have the y-coordinates 0 to 4, the block counter has a block **... – 1** for the **index from 0 to ...** loop. The block **x 0 y index** from the **LED** group will illuminate the number of LEDs in column 0 within the loop indicated by placeholder **A**. If player B has scored more than 0 points, the LEDs in column 4 of the LED matrix will display the number of points in the same way.

The player's scored points are displayed for 500 milliseconds at the end of the game, and also every time a button is pressed.


beim Start
ändere Ende auf falsch
ändere A auf 0
ändere B auf 0
ändere Spiel auf falsch
während A < 5 und B < 5
mache
ändere Punkt auf wahr
wähle eine zufällige Zahl zwischen 1 und 4 -mal wiederholen
mache
zeige Symbol
zeige Symbol
zeige Symbol
ändere Spiel auf wahr
zeige Symbol
pausiere (ms) 1000
ändere Spiel auf falsch
ändere Ende auf wahr
Bildschirminhalt löschen
wenn A > 0 dann
für Index von 0 bis A - 1
machen
Zeichne x 0 y Index
wenn B > 0 dann
für Index von 0 bis B - 1
machen
Zeichne x 4 y Index


wenn Pin P0 gedrückt
wenn nicht Ende
wenn Spiel dann
wenn Punkt dann
ändere A um 1
ändere Punkt auf
pausiere (ms) 100
ansonsten
wenn Punkt dann
ändere B um 1
ändere Punkt auf
pausiere (ms) 100
Bildschirminhalt löschen
wenn A >
für Index von 0 bis
machen
Zeichne x 0
wenn B >
für Index von 0 bis
machen
Zeichne x 4
pausiere (ms) 500
Bildschirminhalt löschen

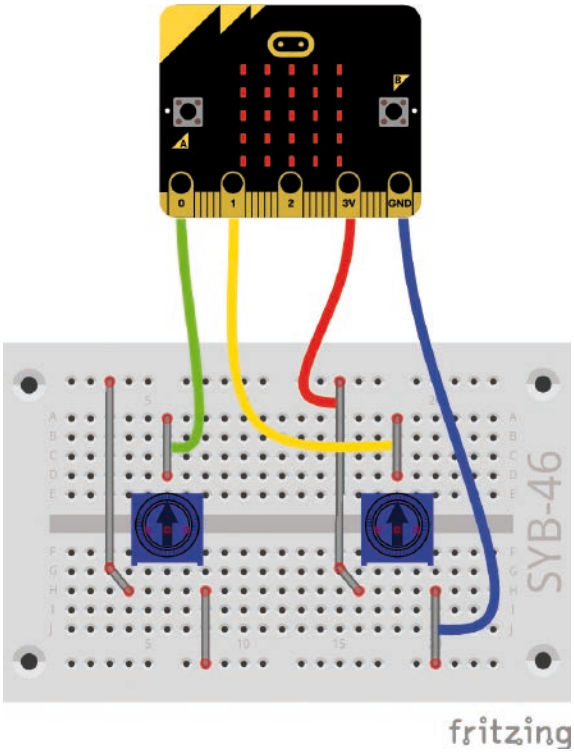

dann
falsch
falsch
dann
A - 1
y Index
dann
B - 1
y Index
wenn Pin P1 gedrückt
wenn nicht Ende dann
wenn Spiel dann
wenn Punkt dann
ändere B um 1
ändere Punkt auf falsch
pausiere (ms) 100
ansonsten
wenn Punkt dann
ändere A um 1
ändere Punkt auf falsch
pausiere (ms) 100
Bildschirminhalt löschen
wenn A > 0 dann
für Index von 0 bis A - 1
machen
Zeichne x 0 y Index
wenn B > 0 dann
für Index von 0 bis B - 1
machen
Zeichne x 4 y Index
pausiere (ms) 500
Bildschirminhalt löschen
dauerhaft

18. Day

Day 18

Today in the Advent Calendar

· 15-k× potentiometer



Game programming with sprites

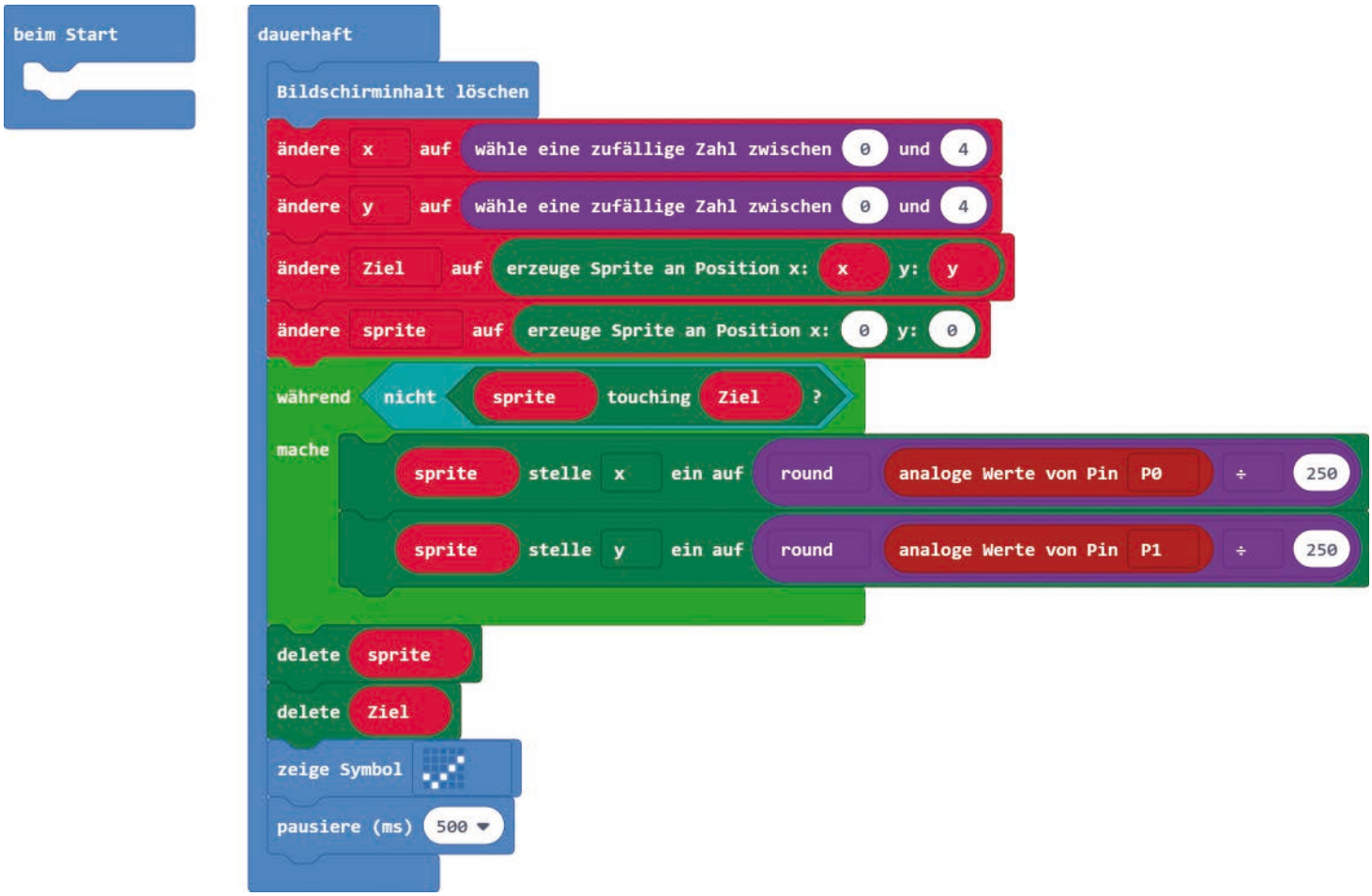
Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 2x potentiometers, 6x bare jumper wires

The breadboard contact strip in the upper part of the figure is used to connect the two potentiometers to +3 V, the contact strip at the bottom is the GND connection.

The program

The program `microbit-18.hex` is a simple game. It is about controlling a sprite with a simple game controller built with the two potentiometers and trying to catch a glowing LED. This program takes a fresh approach at programming the LED matrix by using sprites, just like in professional computer games.

A sprite is a moving object in a computer game, which is controlled by the graphics processor of the computer or game console. It can move across the playing field or a background image. This technology made the first fast computer games possible in the first place. Older computer hardware was not fast enough to animate a moving full image in real time. In the Makecode editor on the micro:bit, our sprite is a single LED that seems to move across the LED matrix with the help of different blocks. This technique makes it far more easier to control animations than the blocks of the **LED** group, where for every movement you have to switch off the LED at the previous position, calculate the new position and then switch the LED back on. The word sprite means something like troll or genie.



How does the program work

In each of the loops of the infinite loop, two random **x** and **y** coordinates are generated, and there the **target** sprite that the player is supposed to catch. To do this, we use **Block create sprite at position x ... y ...** from the **game** group. You find this group in the block list groups under **Advanced**.

A sprite is stored in a variable, which is used to move it around later on. A **sprite** variable is automatically created when the first sprite block is used. The **target** variable for the second sprite in the game is created manually like any other variable.

The sprite **sprite**, which the player moves around by way of the potentiometers, is generated at the position **x:0 y:0**.

The sprite **sprite** can move as long as the sprites do not make contact. The block ... **touching** ... will check whether two sprites are touching each other. Together with a block **not ...** from the **Logic** group it forms the condition for a **while ... do ...** loop. Within this loop, the two analogue values of the potentiometers are read, divided by 250 and rounded to get integer values between 0 and 4.

With the blocks **sprite set x/y to ...** the sprite is moved to the position set with the potentiometers.

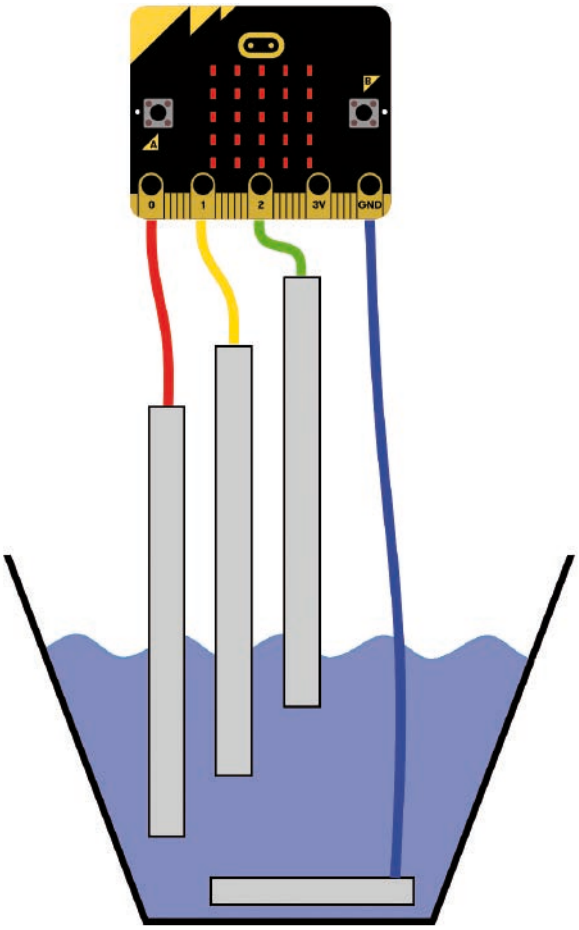
When both sprites make contact, the loop ends, the sprites are cleared, and a tick mark appears on the LED matrix. After a 500 milliseconds waiting interval, the **forever** loop starts the next round of the game.

19. Day

Day 19

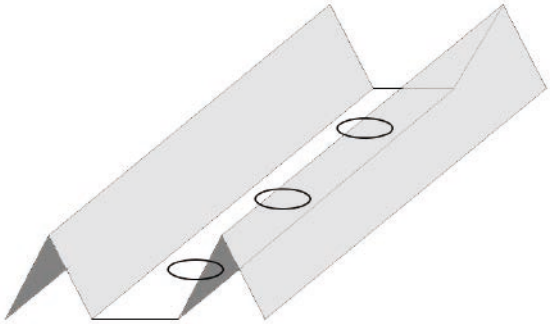
Today in the Advent Calendar

- 2x wire electrodes



Water level sensor

In this experiment we are going to use the wire electrodes to build a sensor that will indicate the level of liquid in a glass. The electrodes are suspended from a bridge over the glass. You can fold the bridge out of the cut out stencil on the back of the Advent Calendar. Attach three wire electrodes with the alligator clip cables in such a way that the wires are suspended in the glass at different depths.

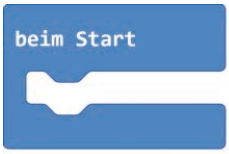


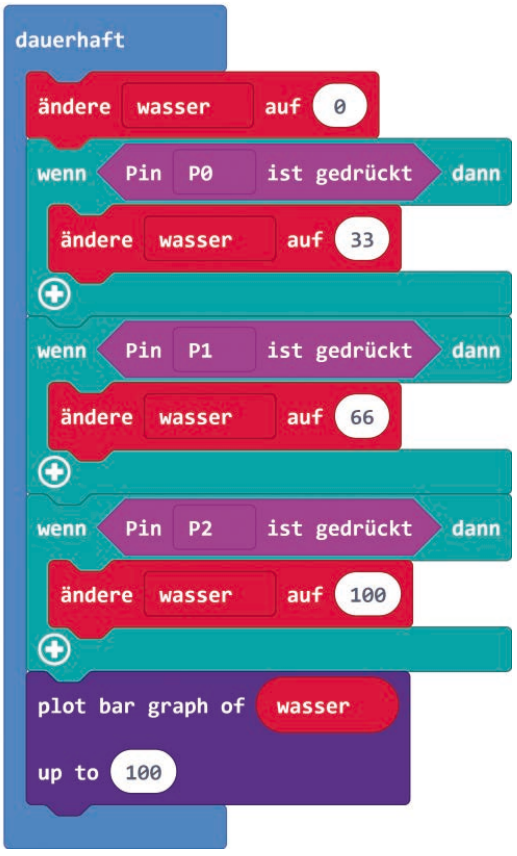
Bend the fourth wire electrode to a circle to form the ground contact and place it down in the water glass, then put weight on the electrode by using a coin or any other piece of metal so that it stays on the ground and won't float up, keeping contact to ground even when very little water remains in the glass.

Components needed: 1x micro:bit, 4x alligator clip cables, 4x wire electrodes

The program

The program `microbit-19.hex` will keep checking the three pins P0, P1 und P2 one after the other to assess the water level in the glass.





How does the program work

The **water** variable is initially set to 0 at the beginning of each run in the **forever** loop and then revaluated and adjusted according to the electrodes' different immersion depths in the glass.

The three electrodes are queried one after the other at the pins P0, P1 and P2. If the electrode is immersed in water at pin P0, it has ground connection. The pin is recognized as being **pressed**. At this time, the glass with water is at least one third full. The **water** variable is set to **33**. This number represents the water level in the glass in percent. Depending on the immersion depth of the electrode, you may also change this value.

Likewise, the other two electrodes, which are not as deep immersed in the glass, are queried. We assume a level of 66% for the electrode at pin P1 and the electrode with the shortest depth at pin P2 would then represent a full water glass.

At the end of each loop run, the program will display a bar graph on the LED matrix.
The value shown corresponds to the **water** variable, and the scale range should be up to 100. All LEDs on the matrix will glow, if all three electrodes are immersed in water.

20. Day

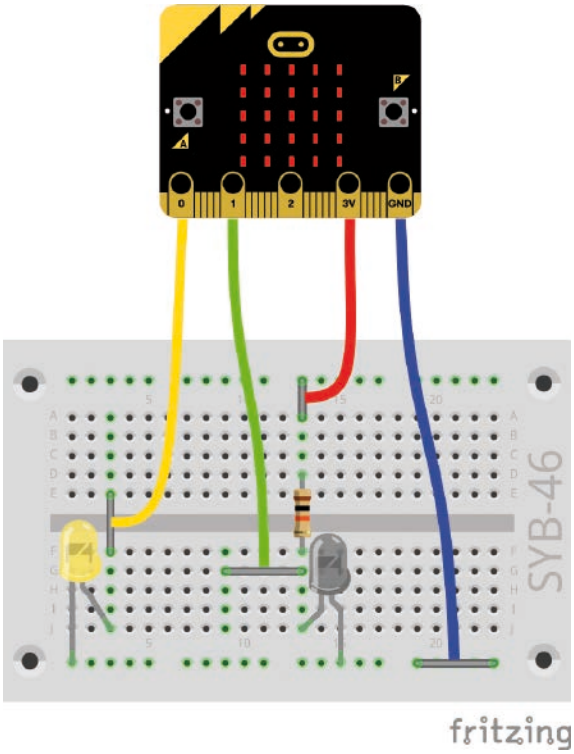
Day 20

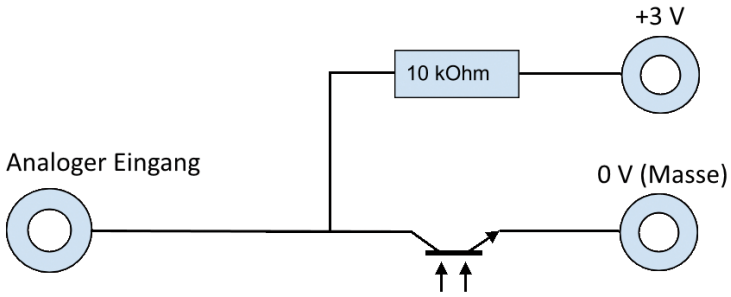
Today in the Advent Calendar

· Phototransistor

Phototransistor

A phototransistor is a device sensitive to light and looks at first glance like a transparent LED. Depending on the light leaks, the illustrated circuit can achieve different values at an analogue input. The brighter the light that falls on the phototransistor, the lower the value at the analogue input. Unlike LEDs, a phototransistor's long leg must be connected to ground and not the short leg.





Schematic diagram for a phototransistor

The micro:bit has such a brightness sensor installed, but it is however not really responsive. Today in the Advent Calendar we have a phototransistor which we will use to build a brightness sensor, and which also has the advantage that you can place it at a distance to the micro:bit to transmit brightness values.

Nightlight in the dark

Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cable, 1x yellow LED, 1x phototransistor, 1x resistor 10 kΩ (brown - black - orange), 4x bare jumper wires

Waking up at night in a pitch dark room can leave you feeling disorientated. A nightlight with a single LED is often enough to show you the way. Such night lights turn on automatically in the dark and off when there is light. The light of these night lights glows so weak that they do not affect the brightness sensor.

The first program

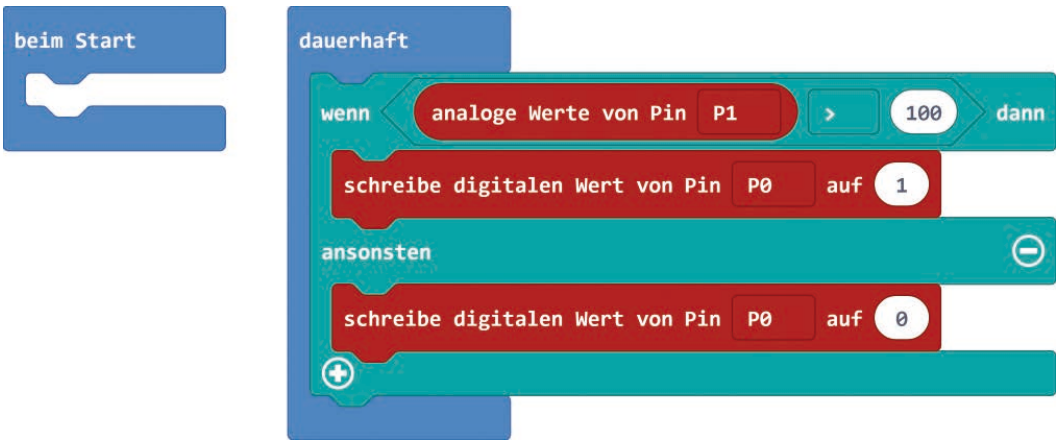
The program `microbit-20-01.hex` uses the built-in brightness sensor on the micro:bit. Thus, we won't need the additional phototransistor yet.

How does the first program work

The built-in brightness sensor delivers an analogue value, which depending on the brightness level, is higher when more light falls on it. If this value is below 100, the LED on pin P0 should glow. If the ambient light is brighter, it will turn off. The current value of the brightness sensor can be read at any time via the **Light intensity** block of the **Input** group.

The second program

The program `microbit-20-02.hex` uses an externally connected phototransistor.



Depending on the level of brightness, the phototransistor delivers an analogue value between 0 and 1023, which in this case is higher the less light falls on the phototransistor.

How does the second program work

The program reads the analog value at pin P1 and turns on the LED at pin P0, if the value is greater than 100. Depending on the ambient brightness level, you may want to change the reference value in the **if ... then ... else...** query, if the nightlight turns on too early or too late.

21. Day

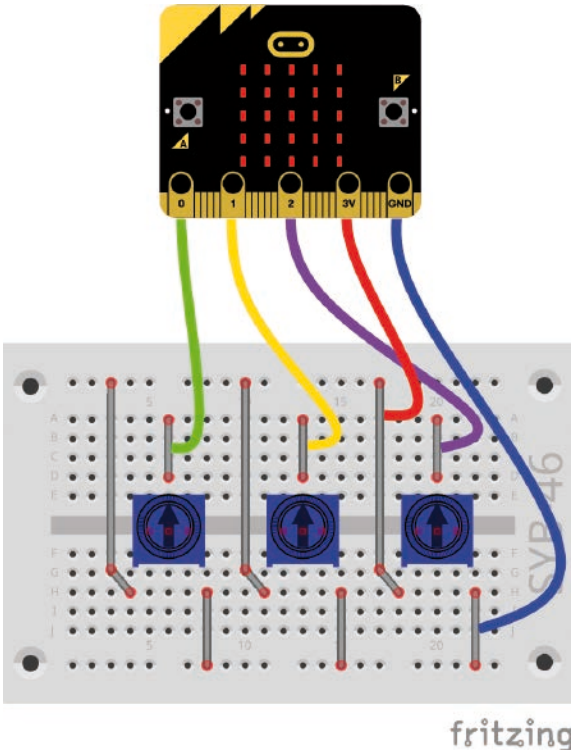
Day 21

Today in the Advent Calendar

- 15-kΩ potentiometer

Running light on the LED matrix

Components needed: 1x micro:bit, 1x breadboard, 5x alligator clip cable, 3x potentiometer, 9x jumper wire (5 of them bare)

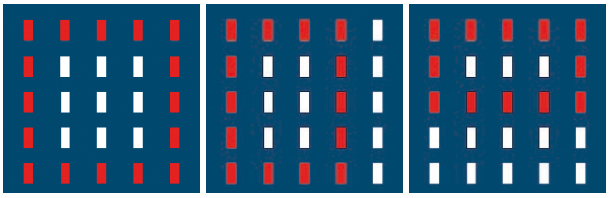


The program

The program `microbit-21.hex` generates different running light effects on the LED matrix depending on the potentiometer settings.

How does the program work

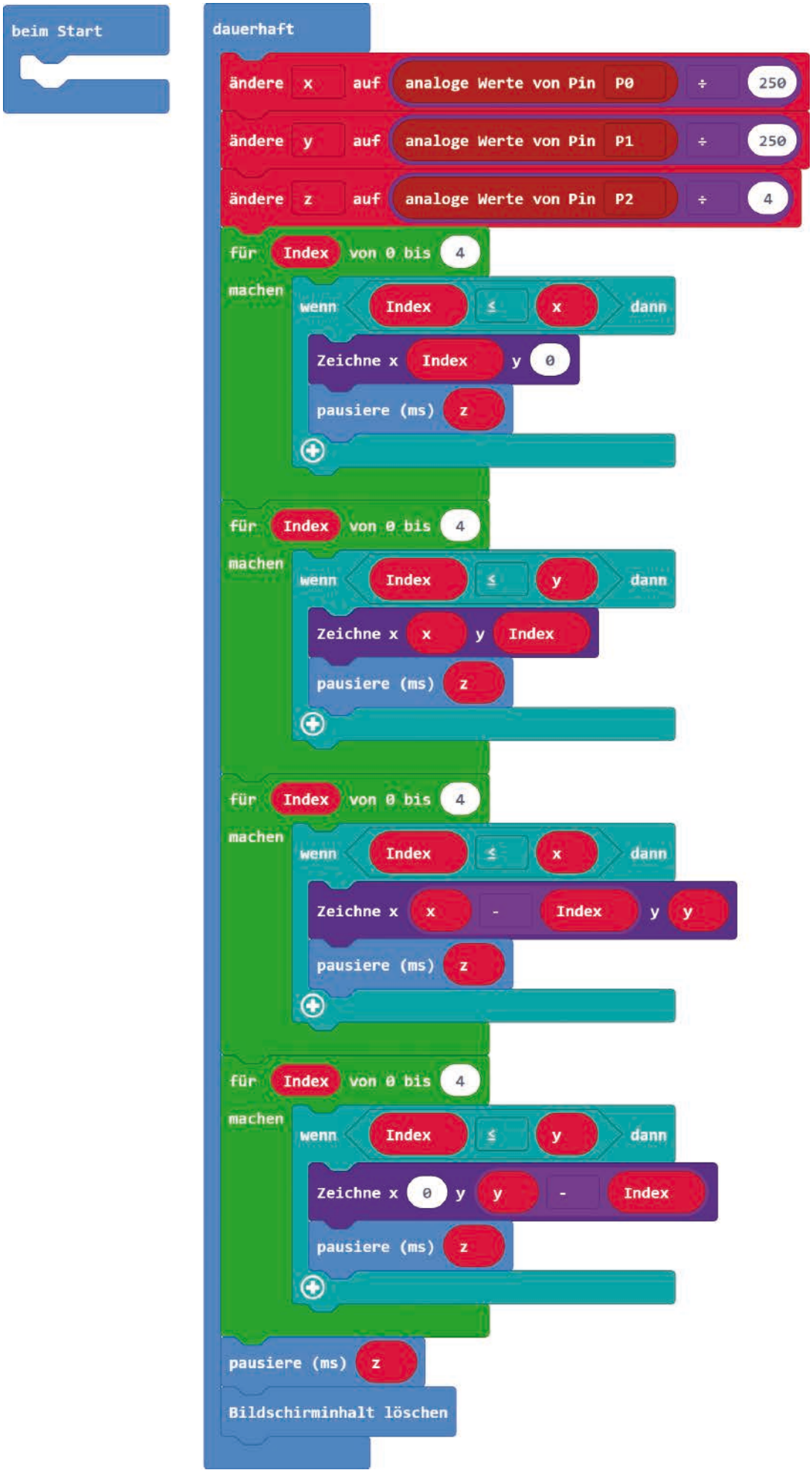
The analogue values of the three potentiometers are read at the start of the **forever** loop run. The first two potentiometers indicate the width (x) and height (y) of the rectangle in which the running light is running. The analogue values can range between 0 and 1023 and are divided by 250, thereby delivering values between 0 and 4 for the five columns and rows of the LED matrix. These values are stored in the **x** and **y** variables.



The value of the third potentiometer indicates the speed of the running light. This value is then divided by 4 which will give us the value for the waiting time between the individual switching operations. It can be anything between 0 and 255 milliseconds,. This value is stored in the **z** variable.

Subsequently, four loops will run, which will represent the top, the right, the bottom and the left side of the rectangle on the LED matrix. Each loop contains an **if ... then ...** query that checks whether the current **Index** loop counter is still less than or equal to the defined rectangle size. The corresponding LED turns on and then the program pauses during the set waiting interval. The calculation formulas in the last two loops are necessary for the decrementing counts.

After the last LED at the end of the run we have again a short waiting interval. Then the screen contents are cleared, and the next run starts.

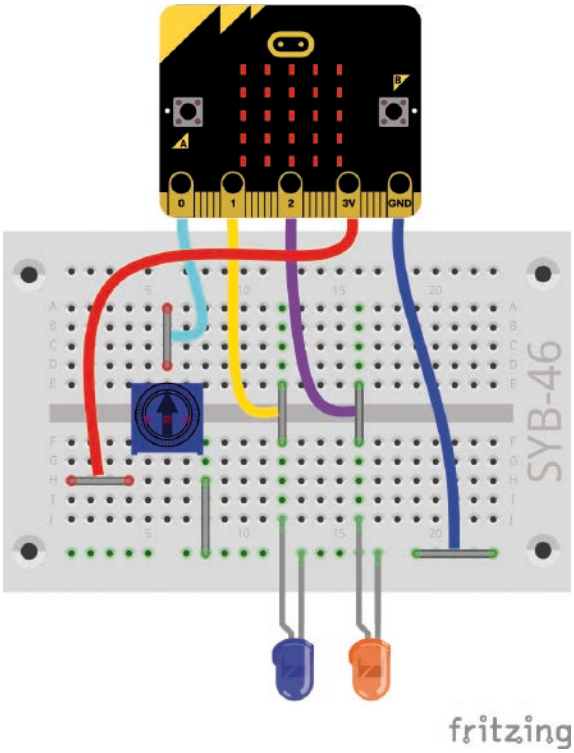


22. Day

Day 22

Today in the Advent Calendar

· Orange LED with series resistor



Guess the number

Components needed: 1x micro:bit, 1x breadboard, 5x alligator clip cables, 1x potentiometer, 1x blue LED, 1x orange LED, 6x bare jumper wires

The program

The program `microbit-22.hex` is a simple guessing game. The player has to guess a randomly generated number in as few steps as possible. The numbers are displayed on the LED matrix whereby a varied number of dots are used and entered via the potentiometer. You can take your guess by pressing the A key. Two LEDs indicate whether your last guess is lower or greater the number you are looking for. At the end, the LED matrix shows how many guesses it took until you guessed that number right.

How does the program work

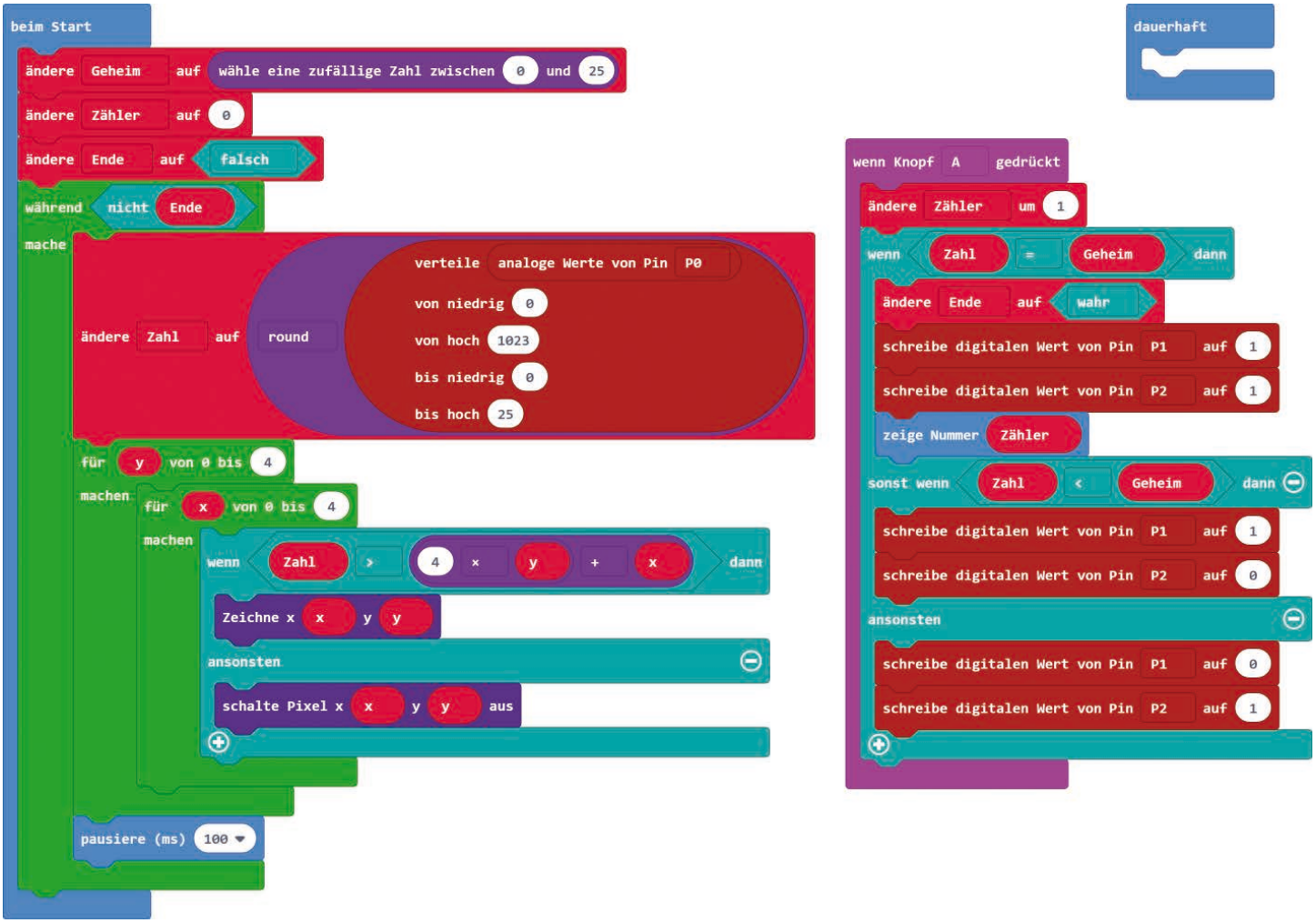
The game variables:

· **End –** logic value indicating the end of the game.

· **Secret** stores the secret number

· **Index –** loop counter

· **x –** x-value for the coordinates on the LED matrix to display the set number



· **y –** y- value for the coordinates on the LED matrix to display the set number

· **number** stores the number, which the player sets and guesses

· **counter** stores the number of guesses

At program start , a secret number between 0 and 25 is randomly selected and stored in the **Secret** variable. The **counter** is set to 0 and the **end** variable is set to **false**.

As long as the **end** variable is not true yet, the value of the potentiometer is queried whether a guess was placed. The **Distribute** ... block of the **Pins** group automatically converts the input values - any number between 0 and 1023 - into the appropriate output values 0 up to 25 for all the 25 LEDs on the matrix. If the potentiometer's maximum value of 1023 is present at the input P0, all 25 LEDs should be illuminated and the number 25 should be keyed in.

Then the set number is displayed on the LED matrix by two nested loops. The outer loop controls the rows with five LEDs each, the inner loop controls the five LEDs of a row.

Following a brief wait interval of 100 milliseconds the value of the potentiometers will be re-assessed.

If the player presses the button, the set number will be used as a guess. The number of guesses taken is increased by 1 in the **counter** variable.

If the entered guess **number** is equal the **secret** number looked for, the **end** variable **is set to true**. The loop will not repeat its run after that. The game is over. Both LEDs are on, and the number of guesses stored in the **counter** variable will be displayed as number on the LED matrix.

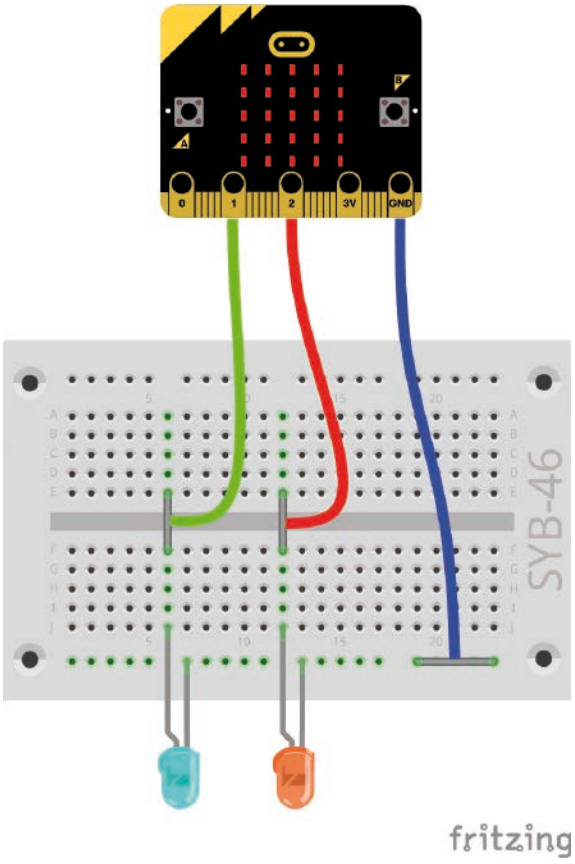
If the guess is less than the target number, LED at pin P1 is switched on; if the number is greater the LED at pin P2 turns on. This gives the player a hint, which direction the guesses should take.

23. Day

Day 23

Today in the Advent Calendar

purple LED with series resistor



Space Invaders

Components needed: 1x micro:bit, 1x breadboard, 3x alligator clip cables, 1x purple LED, 1x orange LED, 3x bare jumper wires

The program

The program `microbit-23.hex` is a simplified variant of the computer game classics Space Invaders on our LED matrix of the micro:bit.

Rules of the game:

- Two enemy ships each the size of two LEDs will drop down from above at random positions.
- The player's ship has the size of one LED and travels along the bottom row on the LED matrix.
- By pressing the (A) button the player's ship moves one grid to the left.
- By pressing the (B) button the player's ship moves one grid to the right.
- Upon pressing both buttons simultaneously the ship fires. The shot should be very fast from bottom to the very top and hit and destroy an enemy ship that is moving in the same column. If hit, the LED should flicker orange.
- If an enemy ship reaches the bottom, the LED should flicker purple.

How does the program work

A few variables are initially set upon start. Then the loop runs forever. Unlike previous games, this game has no end. It could go on forever and ever.

The game variables:

- **Fire –** Loop count for the shots
- **Index –** loop counter
- **Invader –** Position of the invader ship in x-direction
- **Position –** Position of the player shit the moment the shot is triggered
- **Player** – Position of the player ship in x-direction
- **Goal** – Logic value, if the invader is hit

The actual game runs in a forever loop. Once the x-value for the Invader start has been defined, a loop starts that would have the Invader move downwards to the bottom row of the LED matrix in a total of five steps. The **Invader** variable is the position of the invader in the x-direction. The invader ship is two LEDs wide, the position of the left LED therefore can only travel on the first four LEDs of a row.

Here each time is checked, whether the invader was hit by fire. If this is the case, the **Index** loop counter is reset to 0 and the invader will start again in the top row at a random position.

If the invader was not hit, the two LEDs are turned on and will turn off after 200 milliseconds. The next run will then start on the row below. Once the Invader has reached the bottom, the purple LED at pin P1 flickers briefly 4 times in a row. The flickering effect is achieved by the switch-on time, which will be longer than the switch off-time in this sequence.

If you press the A button, the player ship will move one grid to the left. To achieve this, the value of the **Player** variable is reduced by 1. But this should happen only, if this placeholder is still greater than 0. At 0, the player ship is located in the first column of the LED-Matrix to the very left and cannot move further left. The **if...then** block queries then block **if button A pressed** to get the current value of the **player** placeholder.

Likewise, the player ship moves to the right, if button B is pressed.

The player ship should fire, if the two buttons A and B are pressed simultaneously. The fire trail should travel very fast from the bottom to the top and then glow in its entire length for a short time.

First, the current position of the player ship is stored in the **fire** variable to prevent any error in the case the player ship moves during the launch. Then a loop counts up and quickly switches on the LEDs in the column specified by the **fire** variable starting at the bottom moving to the top, which will be perceived as a brief blaze skywards.

While the fire trail glows, we must check if the invader was hit. The invader took a hit, if the player ship and the fire tail are both at the same x-coordinate as one of the two invader LEDs. This will be done by an **if... then ...** query.

If the Invader is hit, the orange LED on pin P2 flickers four times. The **goal** variable is set to **true**, signaling to the main loop that a new invader must start in the top row.

During a significantly longer pause of 100 ms, all four LEDs of the fire tail are glowing and then switched off by a different loop. This time we don't have any waiting interval and the LEDs are all switched off together. In reality of course one after the other is switched off but so fast that we will not notice.


beim Start
ändere Spieler auf 2
Zeichne x Spieler y 4
ändere Treffer auf falsch
wenn Knopf A gedrückt
schalte Pixel x Spieler y 4 aus
wenn Spieler > 0 dann
ändere Spieler um -1
Zeichne x Spieler y 4
wenn Knopf B gedrückt
schalte Pixel x Spieler y 4 aus
wenn Spieler > 0 dann
ändere Spieler um 1
Zeichne x Spieler y 4
wenn Knopf A+B gedrückt
ändere Position auf Spieler
für Feuer von 0 bis 3
machen
Zeichne x Position y 3 - Feuer
pausiere (ms) 10
wenn Position = Invader oder Position = Invader + 1 dann
ändere Treffer auf wahr
für Index von 0 bis 4
machen
schreibe digitalen Wert von Pin P2 auf 1
pausiere (ms) 50
schreibe digitalen Wert von Pin P2 auf 0
pausiere (ms) 100
pausiere (ms) 100
für Feuer von 0 bis 3
machen
schalte Pixel x Position y Feuer aus
Zeichne x Spieler y 4
dauerhaft
ändere Invader auf wähle eine zufällige Zahl zwischen 0 und 3
für Index von 0 bis 4
machen
wenn Treffer dann
ändere Index auf 0
ändere Invader auf wähle eine zufällige Zahl zwischen 0 und 3
ändere Treffer auf falsch
ansonsten
Zeichne x Invader y Index
Zeichne x Invader + 1 y Index
pausiere (ms) 200
schalte Pixel x Invader y Index aus
schalte Pixel x Invader + 1 y Index aus
4 -mal wiederholen
mache
schreibe digitalen Wert von Pin P1 auf 1
pausiere (ms) 50
schreibe digitalen Wert von Pin P1 auf 0
pausiere (ms) 100

Day 24

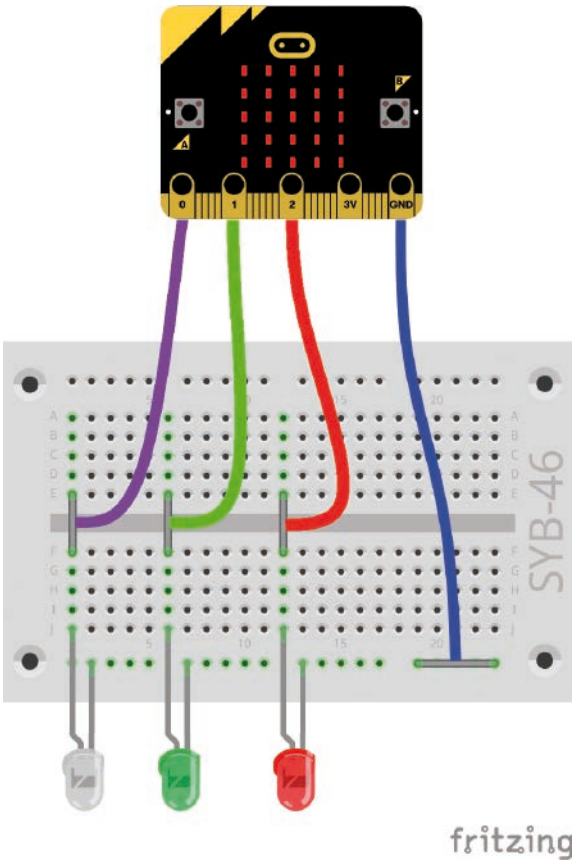
Today in the Advent Calendar

· Blink LED with series resistor

As a special Christmas treat the Advent Calendar today surprises us with an LED that flashes automatically without a program needed. This kind of LED is used for optical signals in situations where it is important to attract special attention.

Christmas quiz

Components needed: 1x micro:bit, 1x breadboard, 4x alligator clip cables, 1x red LED, 1x green LED, 1x blink LED, 4x bare jumper wires



The program

The program `microbit-24.hex` is a quiz with ten Christmas questions, which are displayed on the LED matrix. With the two buttons you can answer each question with yes (A button) or No (B button).

How does the program work

The program uses lists, also called arrays. It is a special form of variable that contains multiple values in a certain order. Such lists are created with the block **change list to ...** for lists with numbers or logic values and for lists with texts the block **change text list to** ... of the **Arrays** group is used.

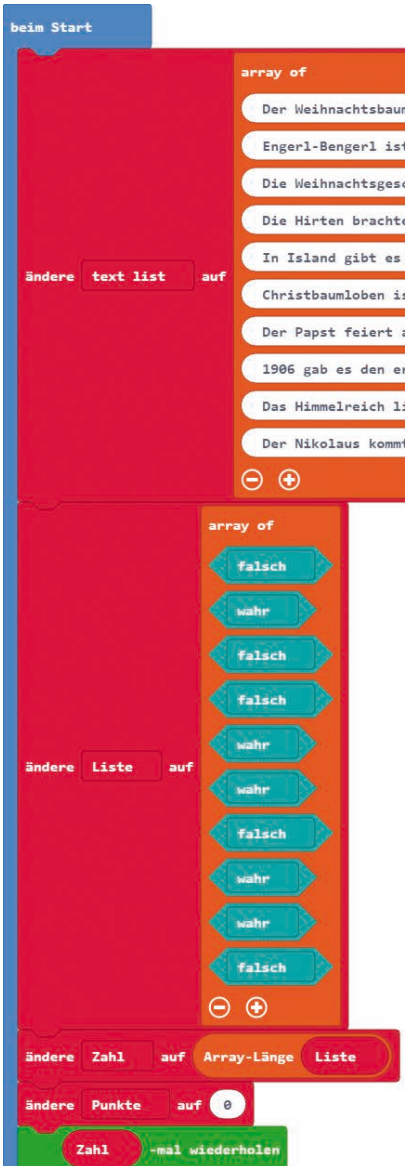
The game variables:

· **Answer –** Players' answer

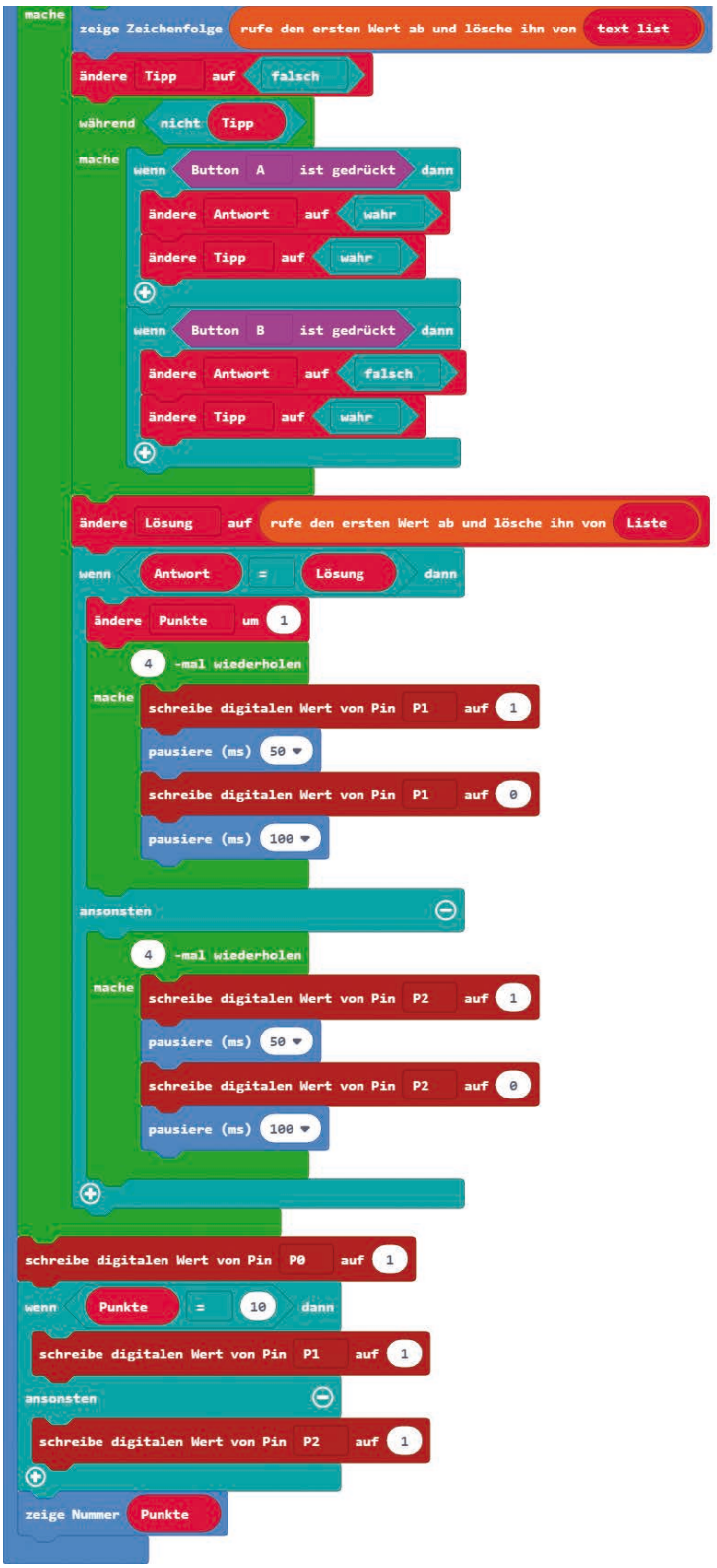
· **List –** Answers to questions

· **Solution –** Correct answer to the question shown

· **Points –** Points scored for correct answers


beim Start
ändere text list auf
array of
Der Weihnachtsbaum kommt aus Schweden
Engerl-Bengerl ist das gleiche wie Julklapp
Die Weihnachtsgeschichte steht bei Markus
Die Hirten brachten Gold und Weihrauch
In Island gibt es 13 Weihnachtszwerge
Christbaumloben ist ein Brauch aus Bayern
Der Papst feiert am 24.12. Geburtstag
1906 gab es den ersten Adventskalender
Das Himmelreich liegt in Niedersachsen
Der Nikolaus kommt aus Bethlehem
ändere Liste auf
array of
falsch
wahr
falsch
falsch
wahr
wahr
falsch
wahr
wahr
falsch
ändere Zahl auf Array-Länge Liste
ändere Punkte auf 0
Zahl -mal wiederholen


dauerhaft


mache
zeige Zeichenfolge rufe den ersten Wert ab und lösche ihn von text list
ändere Tipp auf falsch
während nicht Tipp
mache
wenn Button A ist gedrückt dann
ändere Antwort auf wahr
ändere Tipp auf wahr
wenn Button B ist gedrückt dann
ändere Antwort auf falsch
ändere Tipp auf wahr
ändere Lösung auf rufe den ersten Wert ab und lösche ihn von Liste
wenn Antwort = Lösung dann
ändere Punkte um 1
4 -mal wiederholen
mache
schreibe digitalen Wert von Pin P1 auf 1
pausiere (ms) 50
schreibe digitalen Wert von Pin P1 auf 0
pausiere (ms) 100
ansonsten
4 -mal wiederholen
mache
schreibe digitalen Wert von Pin P2 auf 1
pausiere (ms) 50
schreibe digitalen Wert von Pin P2 auf 0
pausiere (ms) 100
schreibe digitalen Wert von Pin P0 auf 1
wenn Punkte = 10 dann
schreibe digitalen Wert von Pin P1 auf 1
ansonsten
schreibe digitalen Wert von Pin P2 auf 1
zeige Nummer Punkte

· **test list** – Text used for the questions

· **Guess** – Logic value, used for the player's answer

· **Number** – Number of questions

Upon start, a text list with the ten questions and a list with the ten logic values used for the answers are created following the same order. The block **array length** ... from the **arrays** group calculates the length of the list with the answers and stores them in the **number** variable, which is used as a loop counter. You can thus easily add additional questions to the game.

The main program loop runs as many times as there are elements in the list. Our program is set to ten times.

First of all the first text is taken from the list and displayed on the LED matrix. After that, it will be deleted from the list automatically.

The **guess** variable is set to **false**. It specifies whether the player has already entered a guess for the current question. The two buttons are queried for as long as this is not the case. Button A sets the answer to **true**, button B to **false**. The **guess** variable is set to **true** for both buttons when the player has provided an answer.

Now the first value is taken from the **List** list and stored in the **Solution** variable. The value contains the correct answer to the current question. If the player's answer is equal the solution, the points counter will increase by 1 and the green LED on pin P1 will flash four times. If the answer is not equal the solution, the red LED will flash. In this case no point is awarded.

Once all of the ten questions have been answered, the game is over. The blinking LED light on pin P0 is switched on. If the player has correctly answered all the questions, the green LED will also glow. But if the player on the other hand failed to score ten points, then only the red LED will glow. In both cases, the score achieved is ultimately displayed on the LED matrix.

Of course, you can always expand the game and add more questions. Important here is that both lists must have the same number of elements, meaning, there is an answer to every question. There is nothing else in the program that needs to be changed.

Merry Christmas!