

DigiMe Setup Workshop

Interreg



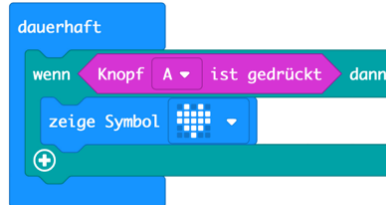
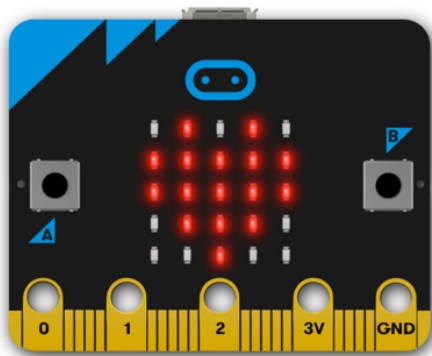
Österreich-Tschechische Republik

DigiMe

Europäischer Fonds für regionale Entwicklung

Project co-funded by the European Regional Development Fund (ERDF)

DigiMe Setup Workshop



Workshop Structure

► Contents

- ◆ Brief introduction (review) of micro:bit and Makecode platform
- ◆ Getting familiar with the radio module (sending messages)
- ◆ Making -> coding and technology (electronics & sensors)

► Examples and opportunities for use in class

► Design of the micro:bit (pins, power supply, etc.)

Practical Part

► Use of the micro:bit's pins:

- ◆ Output (LEDs, servo, motor, LC display)
- ◆ Input (joystick, buttons)
- ◆ Sensors (light, movement, distance, etc.)

► Basics of electronics (3 volts vs. 5 volts, motor drivers, pullups,

► Use of and options for the radio module

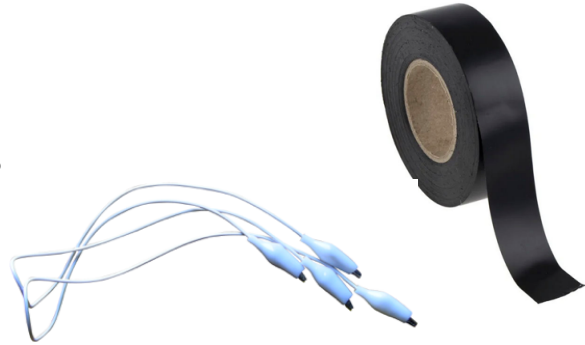
► Examples of possible projects

Hot Wire – Setup 1

You probably know the skill game “hot wire”. We now want recreate this game using the micro:bit, some electronics and handicraft. Depending how good you are with coding, you can incorporate your own features and/or extensions.

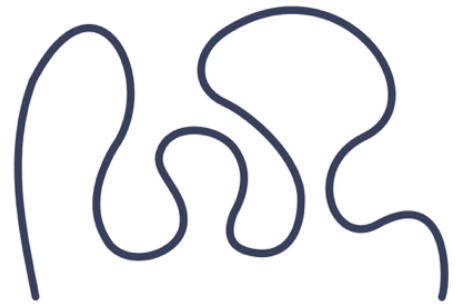
Things needed

- 1 length of wire
- 2 cables with crocodile clamps
- 1 wooden board or plasticine
- 1 micro:bit
- insulating tape



Structure of the wire

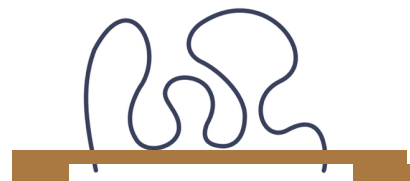
Bend the wire with pliers to your liking. The narrower the wire path, the more difficult it will get to cover the entire distance without any errors.



Draw the starting and end points of the wire path on the wooden board and drill a hole with the diameter of the wire.

Tipp: If you make the distance between the holes 10 mm smaller, the wire path will attach to the board better.

Now put the wire path into the holes you have just drilled. Then bend approx. 10 mm on the two ends of the wire on the underside of the wooden board and cut off the excess ends with a wire cutter. Mount 2 narrow wooden strips on the underside of the wooden board so that you can set down the game on the table.



Alternatively, you can use 2 lumps of plasticine, in which you push the 2 ends of the wire path, instead of the wooden rack.

Hot Wire – Setup 2

Structure of the wire loop

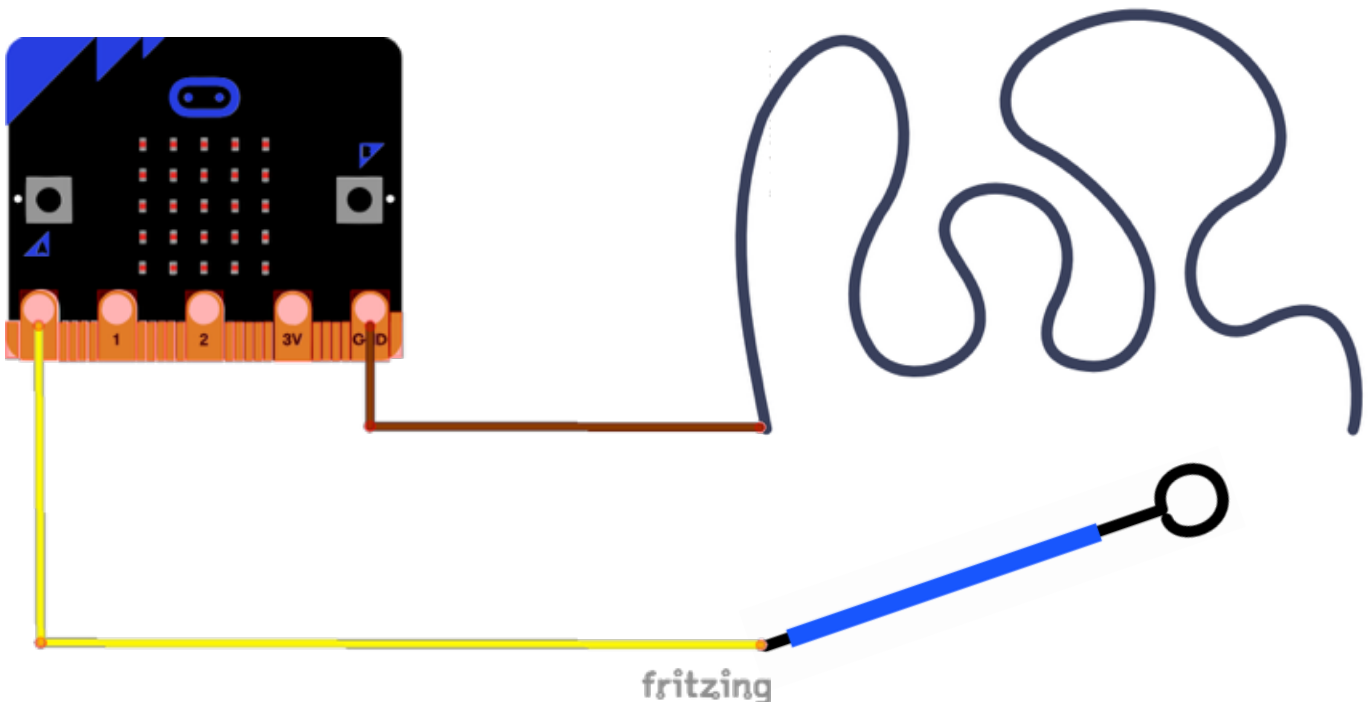
On the end of an approx. 15 cm long piece of wire, bend a loop with a diameter of approx. 15 mm. The smaller the diameter, the more difficult the game will be.

Then wrap the shaft with insulating tape, but leave the last 10 mm of the shaft uninsulated.

Interconnecti

Use a crocodile clamp to connect one end of the wire path with the negative pole (GND) of the micro:bit.

Then guide the other crocodile clamp cable from pin 0 of the micro:bit to the uninsulated shaft of the wire loop.



Hot Wire – Coding

Programming

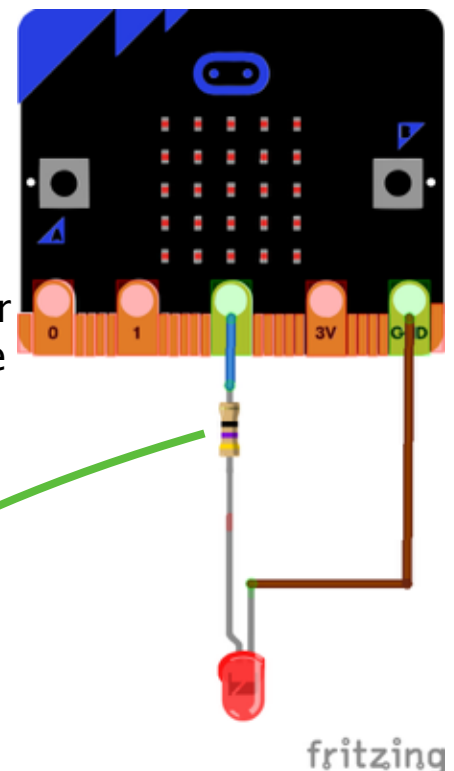
Since your wire loop is connected to pin 0, you need to query if a touch was detected. If this is the case, the sad smiley is displayed and then deleted again.



https://makecode.microbit.org/_a44akyhLFPyH

Error display by LED (optional)

In addition, you can have any error displayed by an LED (light-emitting diode) lighting up. For this, you only need a light diode in the colour of your choice and a series resistor (the costs are less than €1).



Note

A so-called series resistor is needed to prevent the light-emitting diode from getting destroyed. Despite its name, it does not matter whether it is installed before or after the light-emitting diode in the circuit. For “common” LEDs available on the market, you need a 47 ohm resistor in the colours black–purple–yellow for our 3 V power source of the micro:bit.

For more details on the dimensioning of series resistors, see the worksheet on “Ohm’s law”.

Hot Wire – Buzzer

Buzzer extension



https://makecode.microbit.org/_LfhCLTHuXPY3

Tip

You can find the block in the category



in the panel

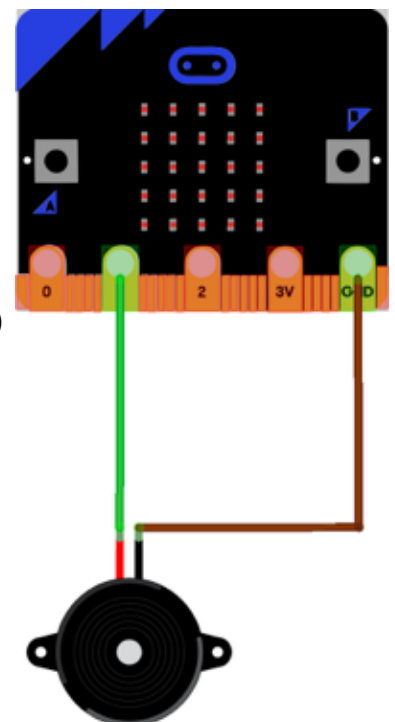


in

You can use a mini-buzzer for acoustic signalling.

Using a crocodile clamp cable, connect the negative pole (short pin) to the GND of the micro:bit or the wire path (since it is already connected to the GND).

Connect the positive pole of the buzzer (long pin) to pin 1 of the micro:bit using another crocodile clamp cable.



fritzing

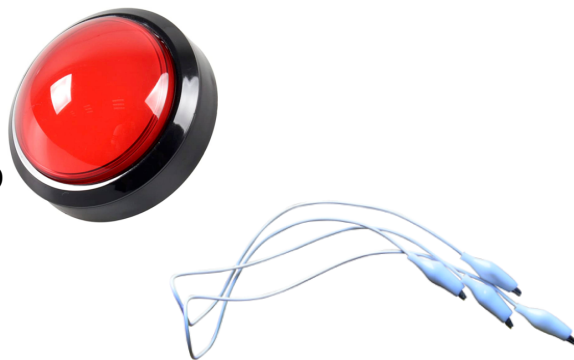
Game Buzzer – Setup

Who doesn't know this situation? Several players are asked a question and you have to decide who has answered it first.

This decision will from now on be made by a circuit you have programmed yourself. A light will go on for the person who has pushed the button first and all the other players can no longer push their buttons until the game master releases the buttons again.

Things needed

- Button
- 2 cables with crocodile clamp
- 1 micro:bit



 Funk

To allow wireless communication between all the players, you need to use the radio module that is incorporated in the micro:bit.

beim Start

setze Funkgruppe auf 7

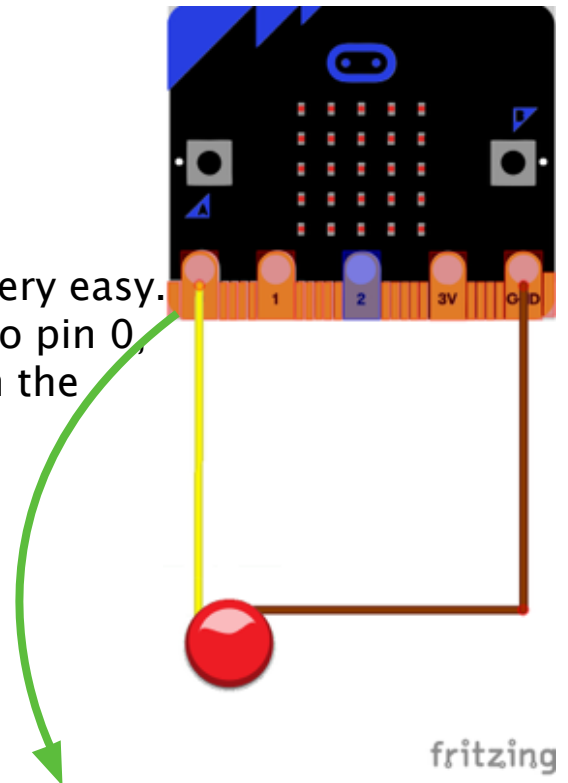
Prerequisite:

The micro:bits that communicate with each other need to share the same radio group (same channel).

Game Buzzer – Transmitter

Buzzer (transmitter)

Wiring the buzzer (radio transmitter) is very easy. One contact of the buzzer is connected to pin 0 while the other contact is connected with the negative pole (GND) of the micro:bit.



As long as the button is not pushed, 3 V (state 1) are applied to the micro:bit.

Once the buzzer is pushed, pin 0 is connected with the negative pole (GND) via the button and 0 V are applied.





In the coding part, the state of pin 0 is queried.

As soon as it is pushed, the text "Peter" is sent via radio. Send the name of the respective player for each of the buzzers used.

Game Buzzer – Receiver

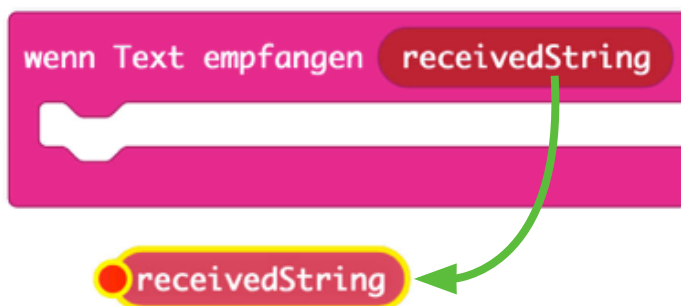
Buzzer (receiver)



To be able to enter a name here, you need this block from the categories “Fortgeschritten” [Advanced]  Text and .

When a text is received, it is automatically stored in the buffer “receivedString” and can be queried just like any variable.

Tip



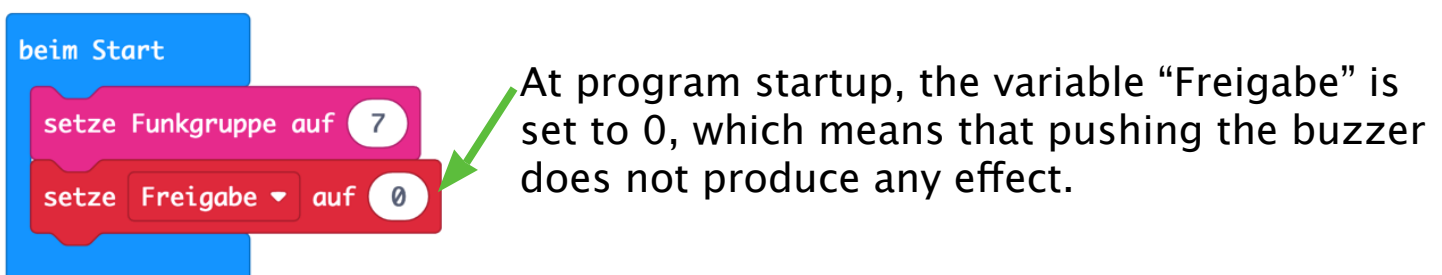
In order to be able to use the variable “receivedString”, you simply drag it from the block “wenn Text empfangen” [if text received].

Game Buzzer – Advanced

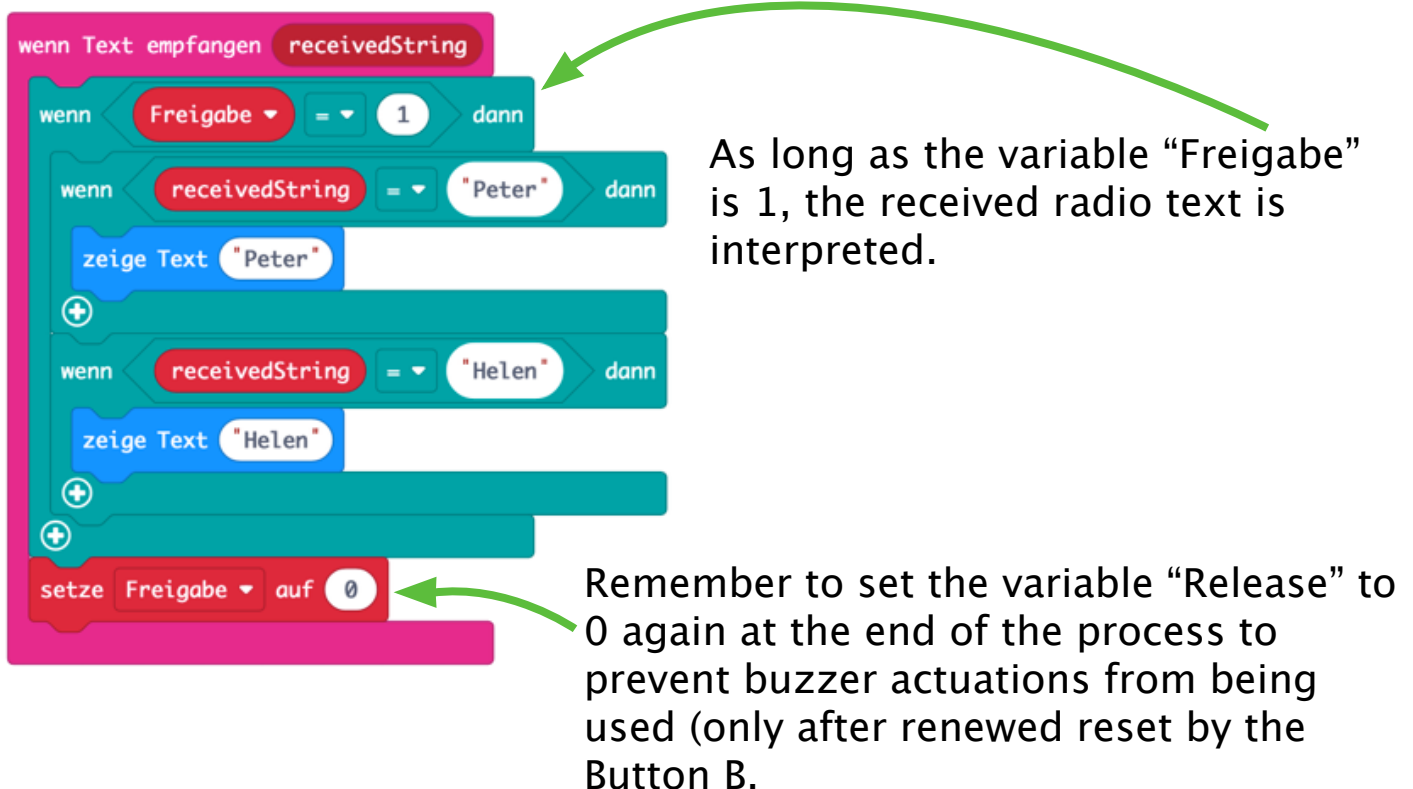
Problem

While the radio receiver displays the name of the person who has pushed the buzzer first, all other buzzer actuations are stored in a buffer. The consequence of this would be that all other names would be also shown one after another.

The additional variable “Freigabe” [Release] is used to prevent this. Buzzer actuations are only used if the variable “Freigabe” is set to 1.



When the Button B is clicked, the variable “Freigabe” is set to 1, which means that the next buzzer actuation can be used.

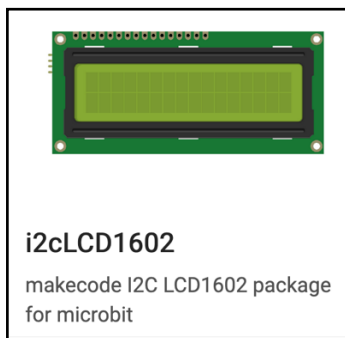


LCD Buzzer Display

We will expand the example above by using an LC display that shows the name of the player whose buzzer was pushed first to the game master.



A prerequisite for this is the implementation of the LCD on the Makecode platform. You can do this by searching for “LCD” in the category “Fortgeschritten” [Advanced] and “Erweiterungen” [Extensions].



From the proposed options, you select the following LC display: **i2cLCD1602**.

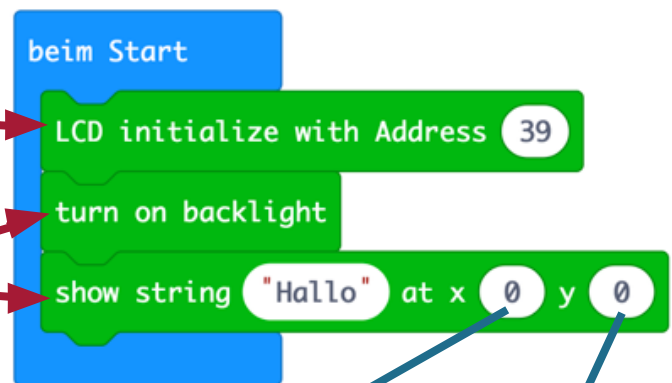
The category  **I2C_LCD1602** is then available.

The most important blocks for use of the LC display:

In order for the display to be used, it needs to be initialised using this line.

You use this to turn on the backlight.

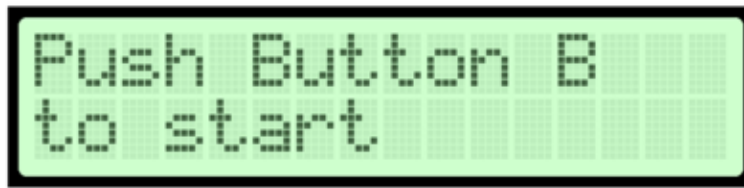
You use this line to indicate what is to be shown on the display.



1. Zeichen

1. Zeile

LCD Buzzer Display – Coding



After initialisation of the display and turning on the backlight, "Push Button B" is shown in line 1.

After a pause of half a second, "to start" is shown in line 2.

When button B is pushed, the display is cleared and you are ready for round 2.



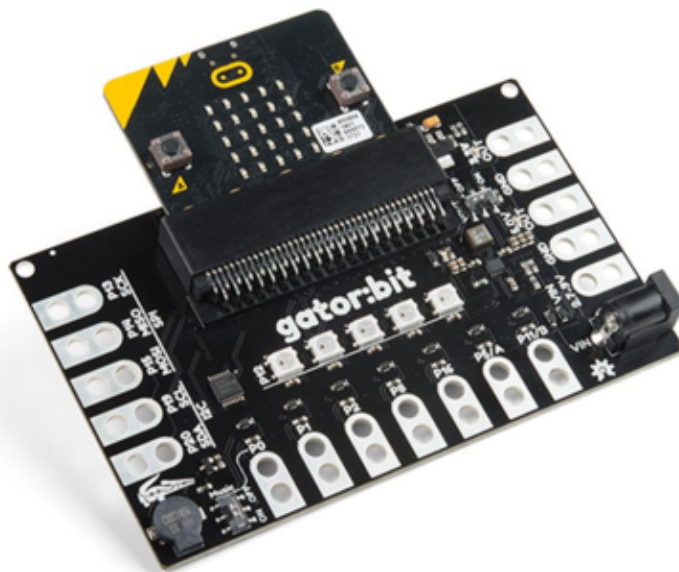
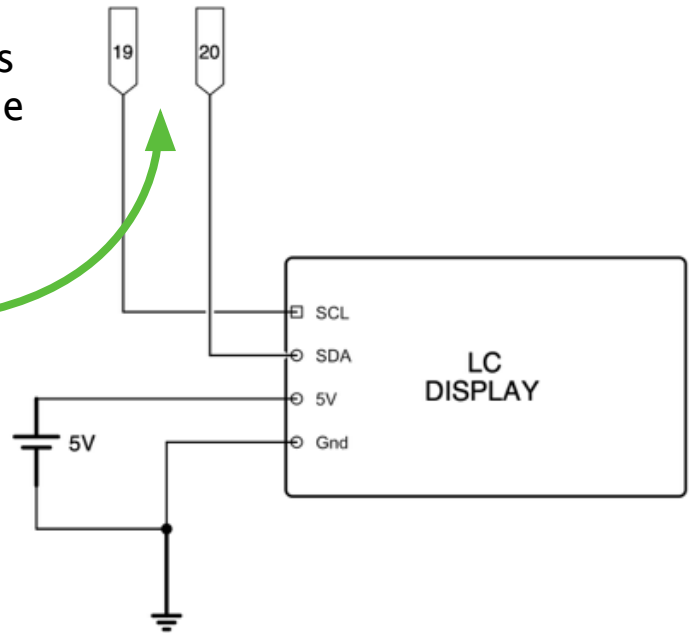
LCD Buzzer Display – Wiring

The wiring diagram on the right shows how the wiring of the LC display on the micro:bit should look like.

Connect SCL with pin 19 and SDA with pin 20.

Important!

You need an external 5 V power source for the LC display to work reliably. This can be a power bank, an external power supply or, as in our case, an add-on board for the micro:bit, which also provides the most important pins for pickup with a crocodile clamp in addition to a 5 V power source.



IR Distance Sensor – Theory



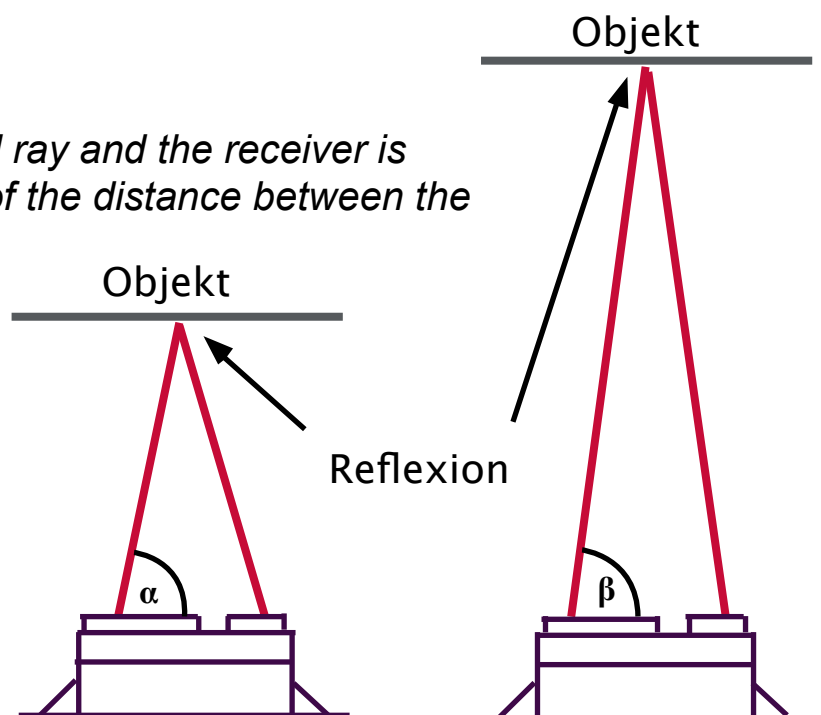
Theory

The Sharp infrared distance sensor emits an infrared ray on one side.

If the ray does not encounter an obstacle, it is not reflected and the receiver on the other side does not receive any information.

If the IR ray is reflected by an obstacle, it will return to the receiver at a specific angle (depending on the distance to the obstacle) and the distance is calculated internally.

The angle between the reflected ray and the receiver is used for internal determination of the distance between the object and the sensor.



Possible uses

- Parking assist system
- Robot
- Hand dryer
- Alarm system
- Toilet flush
- Door opener

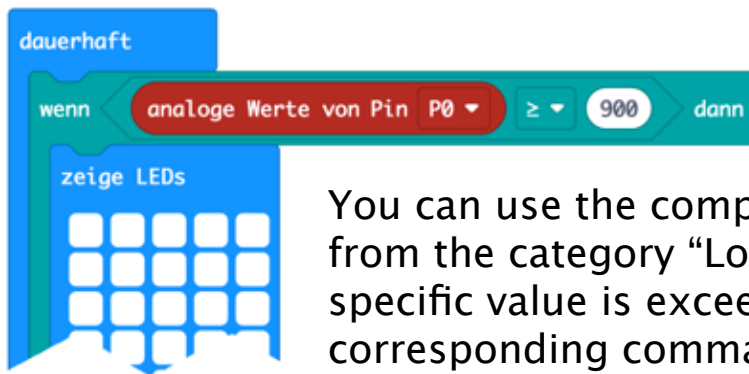
IR Distance Sensor – Coding

Wiring

You can connect the two power supply cables of the IR sensor directly with the micro:bit. [red -> 3 V / black -> GND]

Connect the yellow cable with pin 0, 1, or 2.

In the category **Pins**, you will find the block **analoge Werte von Pin P0**. Depending on the distance, this value will be between 0 and 1023. The higher the value, the shorter the distance to the obstacle.



You can use the comparative operator from the category “Logik” [“Logic”] to check if a specific value is exceeded and then execute the corresponding commands.

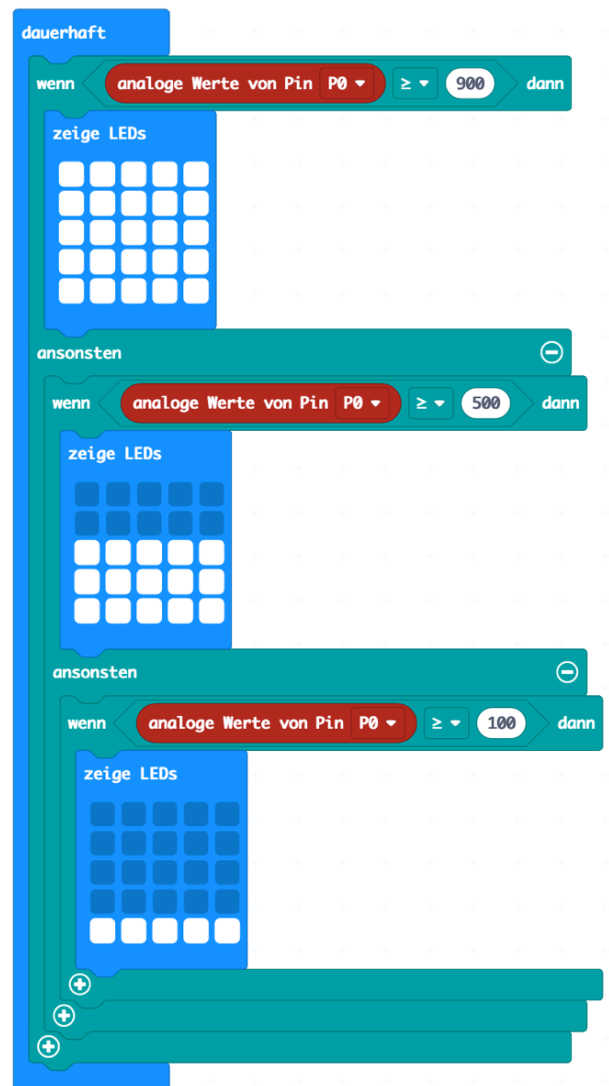


Parking assist system

For inch-perfect parking, it would be helpful to have visual feedback on a display.

For this, you simply need to query pin 0, to which the IR sensor is connected, for its value.

The higher the value, the more bars are displayed (smaller distance).



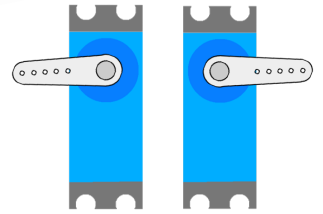
Servo – General

Functionality

Contrary to a conventional motor, a servo motor does not rotate 360°, but assumes a specific position in the range between 0° and 180°.

A conventional servo can only mechanically rotate in the range between 0° and 180° because it is attached at one end (exception: 360° continuous rotation servo).

Besides the power supply (brown – 0 V, red – 5 V), a control line (yellow) is also available. The duration of the pulse applied to this line determines the angle of the servo. Every 20 ms (0.02 s), the servo expects a pulse that determines the angle between 1 ms [0°] and 2 ms [180°].



Info

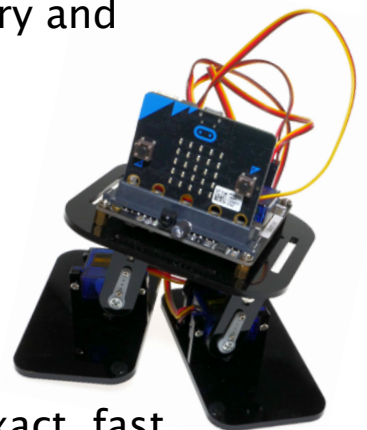
A SG90 can be operated directly on the micro:bit with reduced power. For several servos, the easiest option is to use a servo board designed for this purpose or an external power source.

Possible uses

Servo motors have many uses. They are used in industry and mechanical engineering, but also by hobbyists:

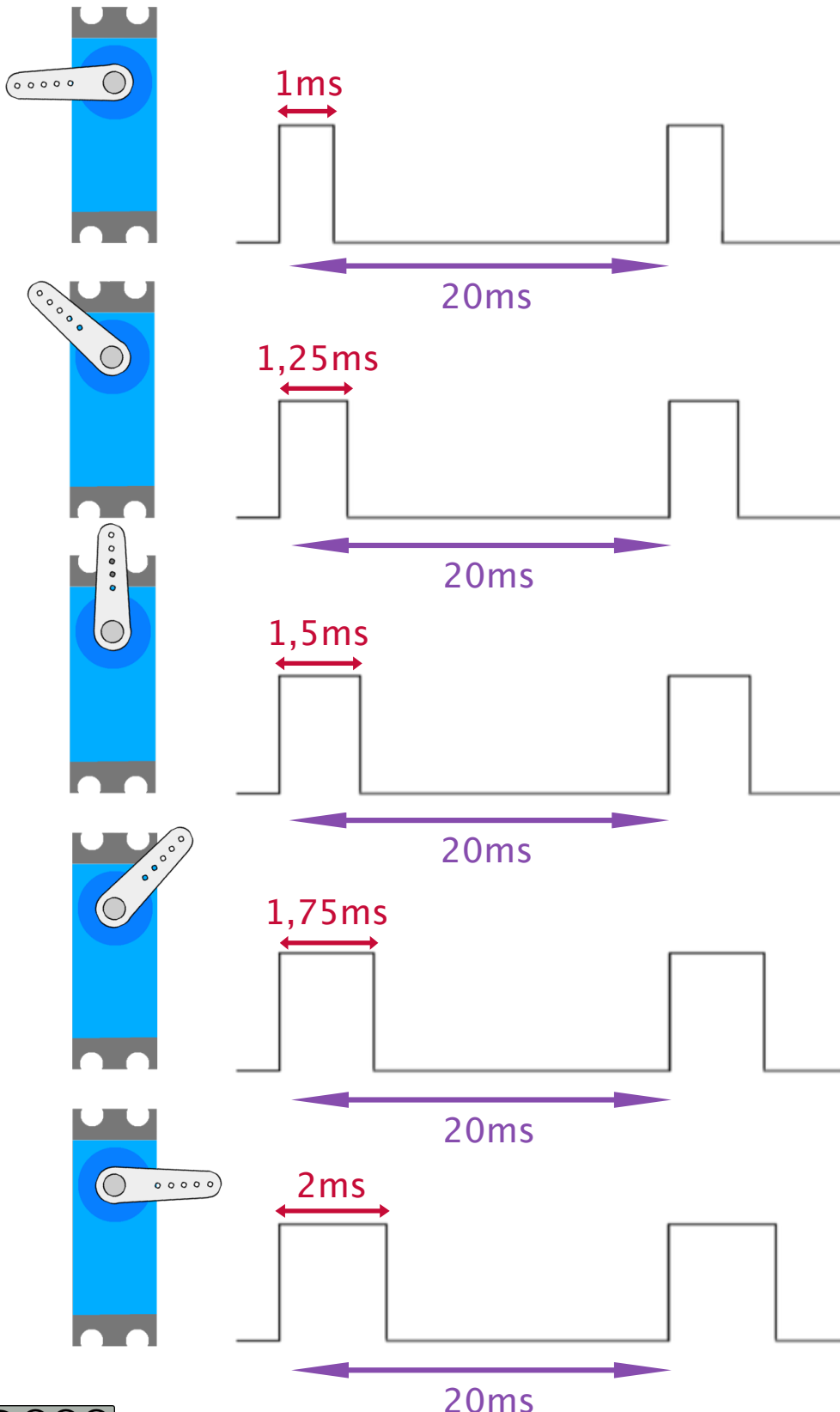
<u>Industry:</u>	Robot arm
<u>Leisure:</u>	Scale modelling
<u>Motor vehicles:</u>	Automatic seat adjustment
<u>Sensors:</u>	Positioning of sensors

Servo motors are often used when high torques and exact, fast movements are important.

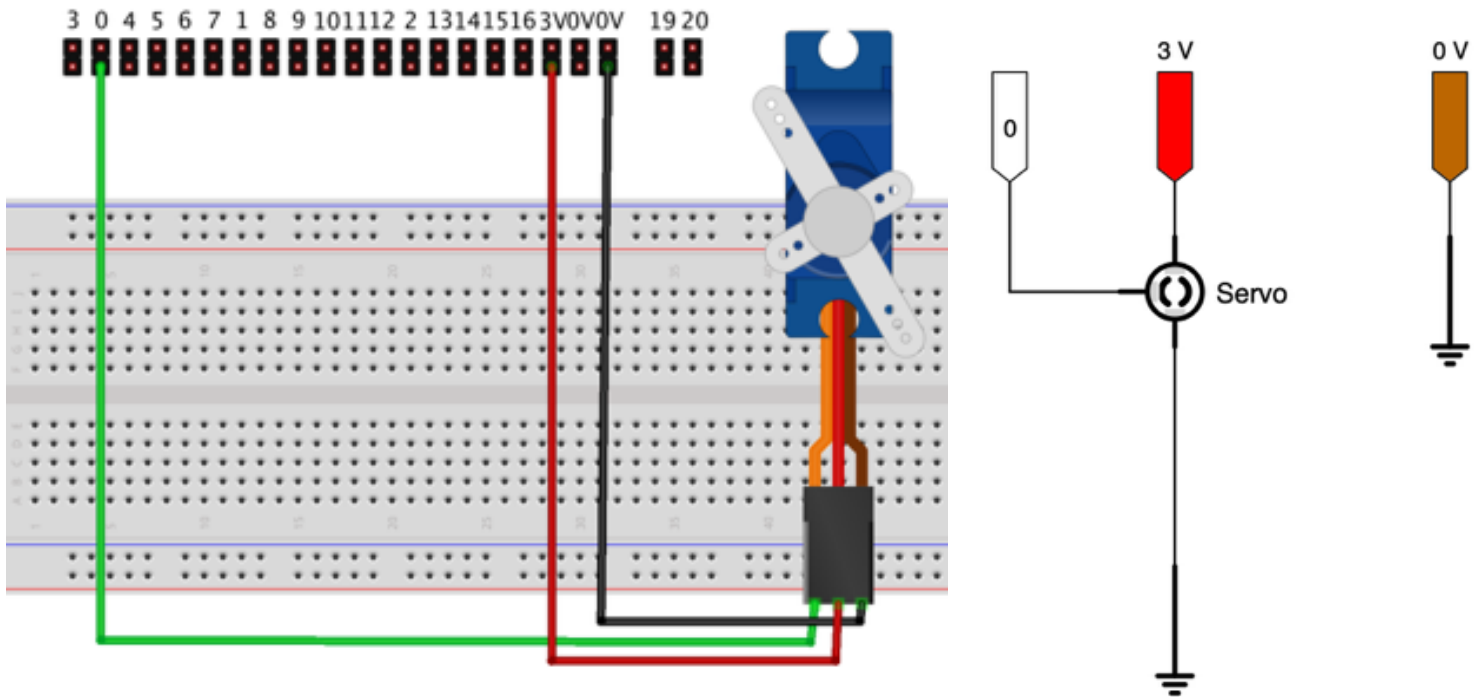


Servo – Theory

Within a period duration of 20 ms, the pulse width of the control signal indicates the angular position of the servo. In case of a pulse width of 1 ms, the servo is in the position 0° (leftmost) and, with the increase to 2 ms, it moves to 180° (rightmost position). These values are for orientation only and must be taken from the data sheet. The timing is the task of the microcontroller (micro:bit) and is implemented in programming by means of the so-called pulse width modulation.



Servo – Wiring



- Wiring of the servo: route the leftmost terminal (control line) via the **green cable** to **pin 0** of the micro:bit.
- Use the central terminal to connect it via the **red cable** with the **3 V pin**.
- Finally connect the GND of the servo using the **black cable** with the **0 V pin** of the micro:bit.

Info

In most cases, a servo will require more power than the micro:bit can provide. If this is the case (servo does not move or “jitters”), you need to provide the servo with its own power source.

Remember to connect the external mass (negative pole) with the 0 V pin of the micro:bit.

Servo – Coding

A prerequisite for this is the implementation of the servo on the Makecode platform. You can do this by clicking “Servo” in the categories “Fortgeschritten” [Advanced] and “Erweiterungen” [Extensions].



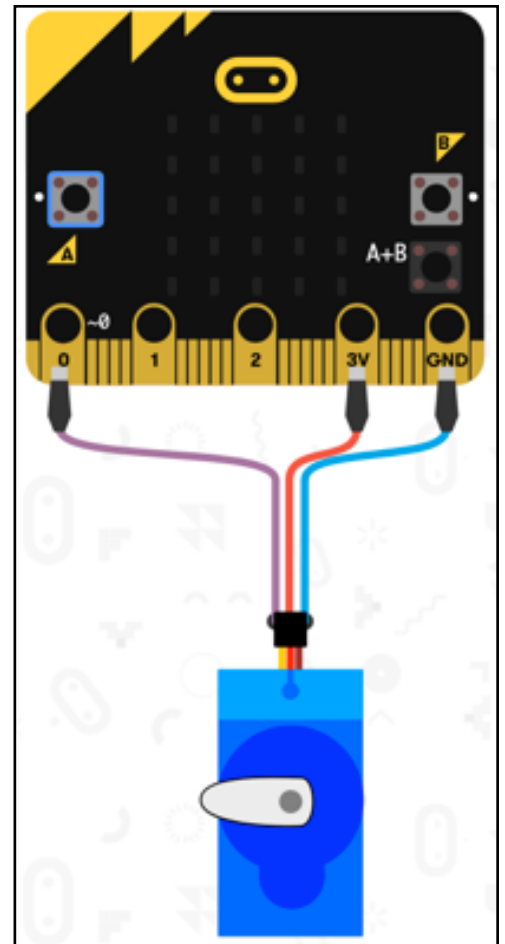
You will then have the category  available.

Test the function of the servo by assigning 3 angular positions to the buttons.



Info

You can also test this function easily without hardware with the online simulator on the Makecode platform.



Servo – Brightness Display

In this example, we want to indicate the ambient brightness with a on the servo.

To do this, you need the block **Lichtstärke** and the block that you use to indicate the servo angle.

setze Winkel von Servo an Servo P0 auf 0°

Problem

dauerhaft
setze Winkel von Servo an Servo P0 auf Lichtstärke°

If you want to output the luminous intensity as a servo angle, like in the example above, the values will not match.

The value range of the servo is 0–180 (degrees), whereas the value range of the brightness sensor is 0–255 in the simulator and 0–1023 in the micro:bit itself.

Solution

To map a numerical value from one range onto another range, you can use the function “verteile” [distribute] from the category “Mathematik” [Mathematics].

In our case, the possible brightness values 0–255 of the simulator are to be assigned to the possible servo values 0–180.




The complete code looks like this.

dauerhaft
setze Winkel von Servo an Servo P0 auf verteile Lichtstärke von niedrig 0 von hoch 255 zu niedrig 0 zu hoch 180°

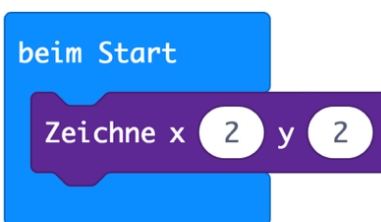
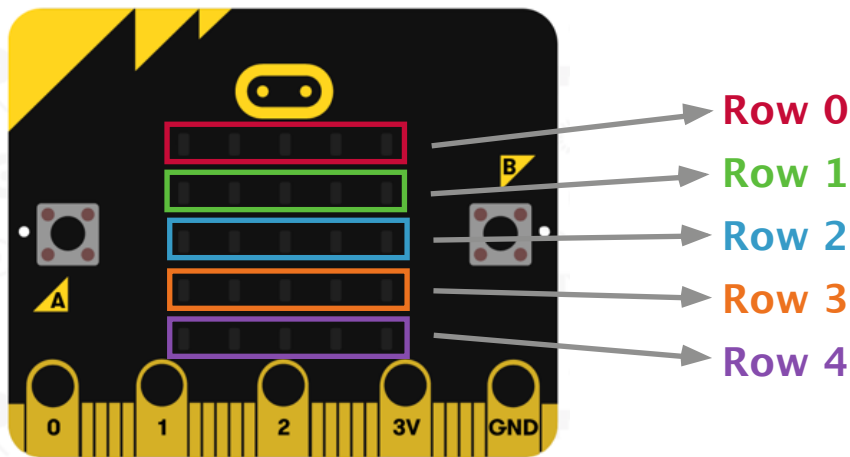
If you use the “real” micro:bit instead of the online simulator, you need to change the max. brightness value from 255 to 1023.

Drawing with the Joystick – 1

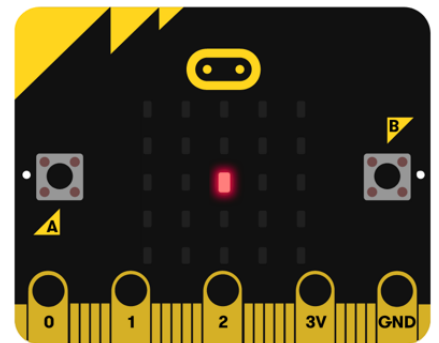
Move the LED pixel on the micro:bit using the joystick. You can save the current position of the pixel by pushing button A. Pushing button B shows you the overall image of all stored pixels. They can be icons, emojis, letters or simple images.

For this purpose you need the block.  from the category “LED”.

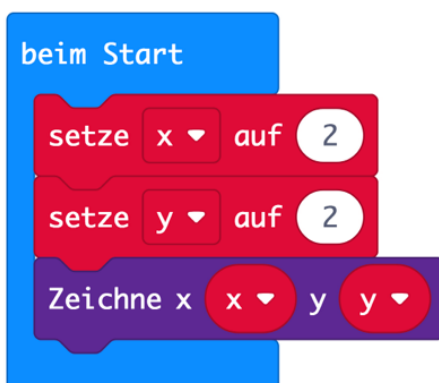
As in a co-ordinate system, x indicates columns and y indicates rows. For programming languages, it is common usage to start the count at 0. Accordingly, the first row is row 0 and the last one is row 4.



If you want the pixel to appear in the centre at startup, you can achieve this with the coordinates $x=2$ and $y=2$.



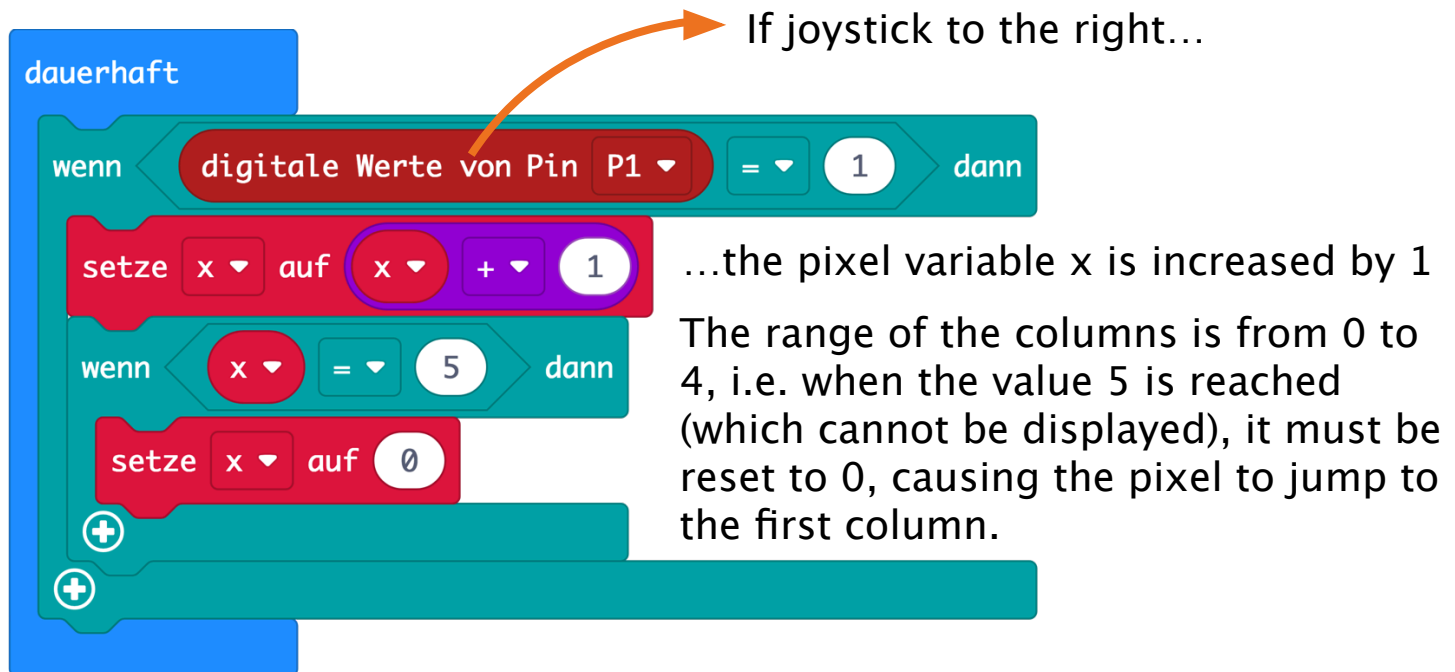
Since the lit pixel is to be “movable” using the joystick, it must not be static, but dynamic, i.e. it needs to be indicated with variables.



The output on the display is the same – from now on, however, the joystick can be used to increase or decrease the LED value, so that the LED pixel moves.

Drawing with the Joystick – 2

The following program block causes the LED pixel to wander to the right as soon as the joystick is pushed to the right (pin 1; provided that it has been wired accordingly).



Wiring:

A joystick is nothing else than 4 buttons that each close a circuit if operated accordingly.

Since the micro:bit detects a voltage of 0 V as “pushed” on the digital input, a cable with GND (0 V) is connected on the other one with the respective pin of the micro:bit, e.g. pin 1. Just as the button is actuated (e.g. joystick to the right), the circuit is closed and 0 V are applied to pin 1, which is detected as an actuation of the button by the micro:bit.

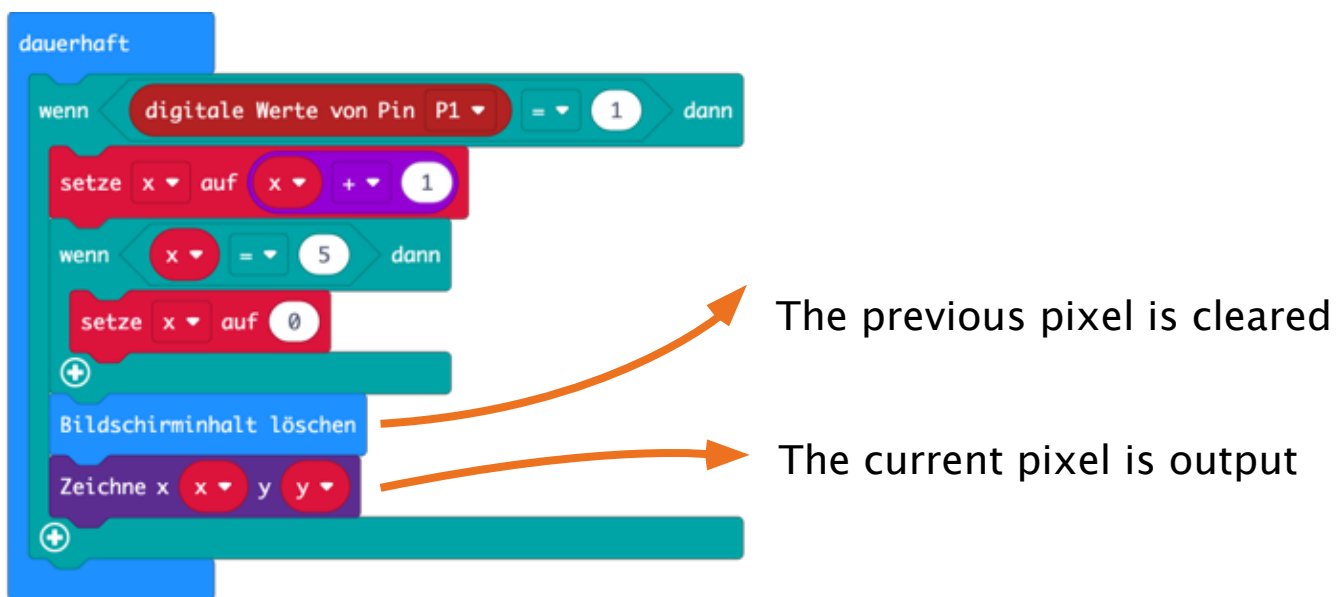
Drawing with the Joystick – 3

To also display the pixel (currently only the corresponding variable has been changed), the following block must be added.

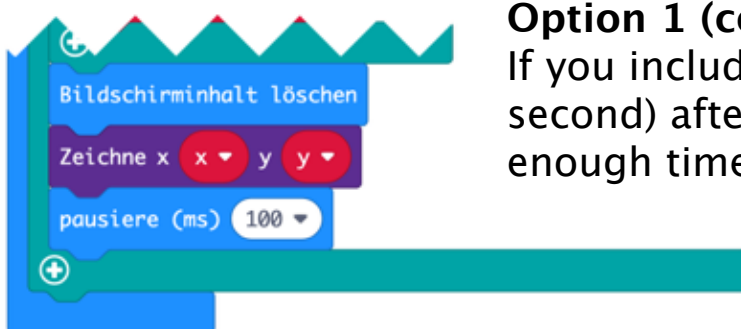


Since the already existing pixel is not cleared when the joystick is pushed to the right, each actuation would result in another pixel being added.

In order to clear the previous pixel in each case, the display must be cleared before the current pixel is output.



As the micro:bit executes instructions very fast, even a short push of the button would cause the pixel to be moved forward many positions. There are 2 options to prevent this:



Option 1 (compromise):

If you include a pause of 100 ms (1/10th of a second) after the output of the pixel, you have enough time to release the button again.

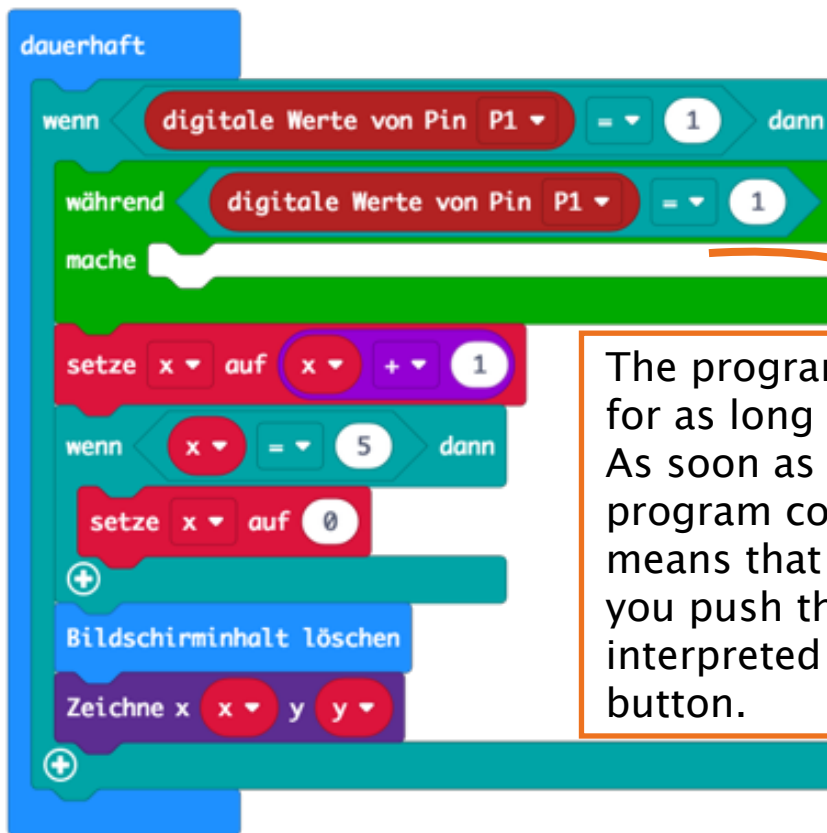
Note

If you push the button or joystick too long (for more than 100 ms), the pixel will still continue to move. If you define a pause that is too long, the pixel will no longer be able to move forward fast enough.

Drawing with the Joystick – 4

Option 2 (more elegant method):

After the button has been pushed → an empty loop is used to wait until the button (pin 1) is released again.



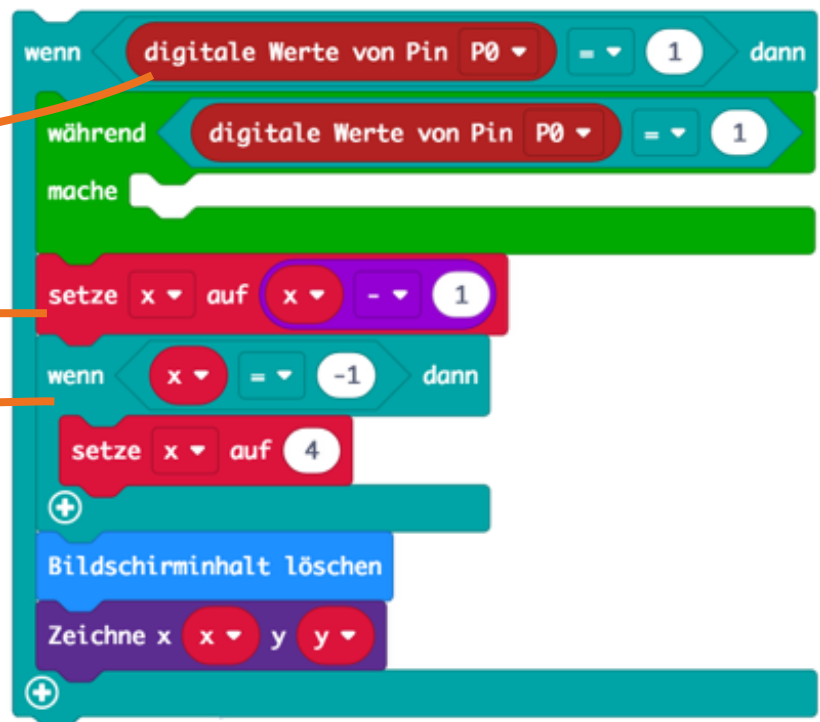
The program remains in this “waiting loop” for as long as the button is being pushed. As soon as the button is released, the program code below is continued. This means that it does not matter how long you push the button – it is always interpreted as a single actuation of the button.

For the movement of the joystick to the left, you need to change the following:

Connection of joystick to the left to pin 0.

The pixel value is reduced by 1.

If the leftmost column ($x=0$) is reduced by 1 ($x=-1$), it must be set rightmost, to $x=4$.



Overall pixel control with the joystick:

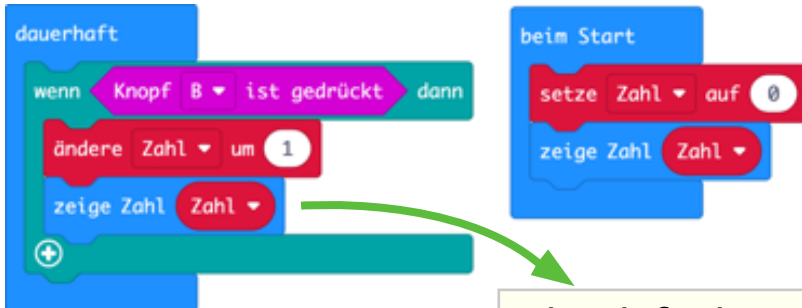
https://makecode.microbit.org/_0VxWRVh5rhLe

Advanced Button Actuation – Problem

Task:

At each startup, the display shows the number 0.

Each time button B is actuated, the output is to be increased by 1.



The default setting for the display output is 0.6 seconds (600 ms).

Note

You can change the duration of the display by placing a comma after the output value in JavaScript or Python and indicating the duration in milliseconds (1 second = 1000 ms).

```
Python Code:
1 Zahl = 0
2 basic.show_number(Zahl)
3
4 def on_forever():
5     global Zahl
6     if input.button_is_pressed(Button.B):
7         Zahl += 1
8         basic.show_number(Zahl, 200)
9 basic.forever(on_forever)
10
```

Problem

If the output duration is too short, the display will jump upwards two or more times when the button is pushed once.

If the output duration is too long, you need to wait for a long time until the next actuation of the button is detected, or it is not detected at all.

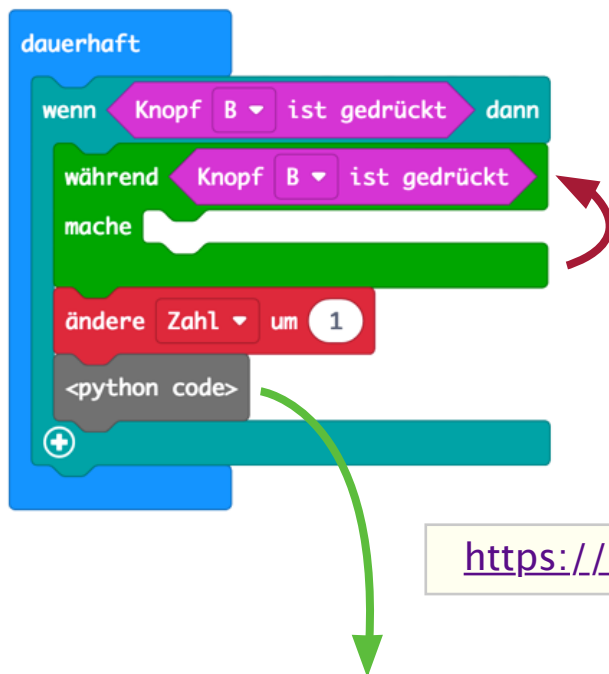
Advanced Button Actuation – Solution

Solution to the problem:

An elegant solution would be to value the actuation of the button only after it has been released again. In this case, it would be up to the user for how short or long the button is actuated – it would only be detected

Implementation

After the button has been pushed, the system waits until the button is released again. Only after that, the command is executed.



This can be done using a loop:

The loop is repeated for as long as the button is pushed. The program is continued only after it is released. Since nothing happens during waiting, the “make” [make] block within the loop remains empty.

https://makecode.microbit.org/_9bpi07HUJasw

For a short push of the button to also be detected, reduce the output duration on the display to approx. 10 ms

```
basic.show_number(Zahl, 10)
```

Note

The output of multi-digit numbers would have to be adjusted by a separate query to a longer output duration on the display to prevent the output from becoming too fast.

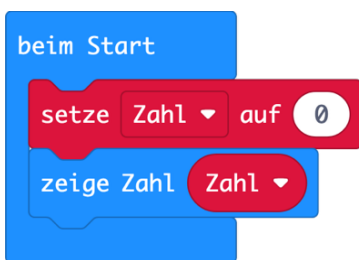
https://makecode.microbit.org/_Fsr7p915mh7c

Differentiation – Short and Long Button

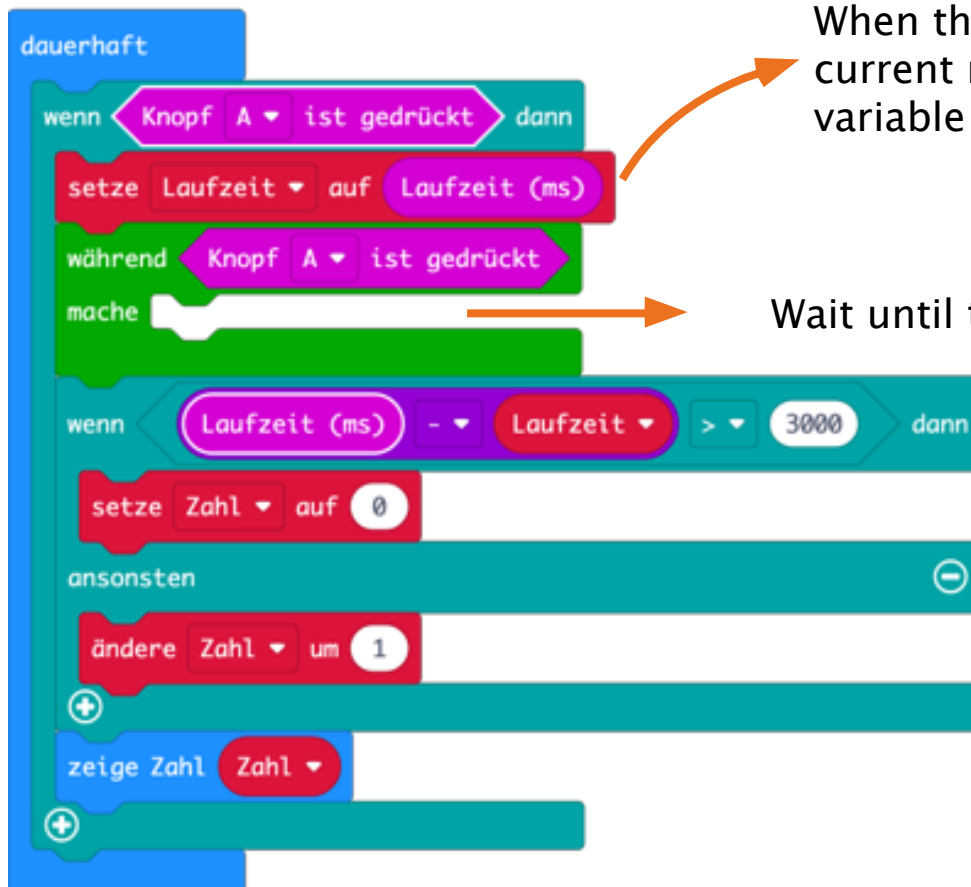
Task description:

A single button is to be used to display the next number after pushing the button for a short time. When the button is pushed longer, the display is to be reset to 0.

This scenario can be found in hardware, for example, that does not have sufficient space for several buttons, or you try to avoid an unintended reset by requiring long actuation of the button.



For this scenario you need a kind of “stopwatch” that indicates whether the button has been pushed long enough. This task is assumed by a so-called **timer**. The time will start automatically when the micro:bit is supplied with power and counts upwards in microseconds.



When the button is pushed, the current runtime is saved in the variable “Laufzeit” [stopwatch].

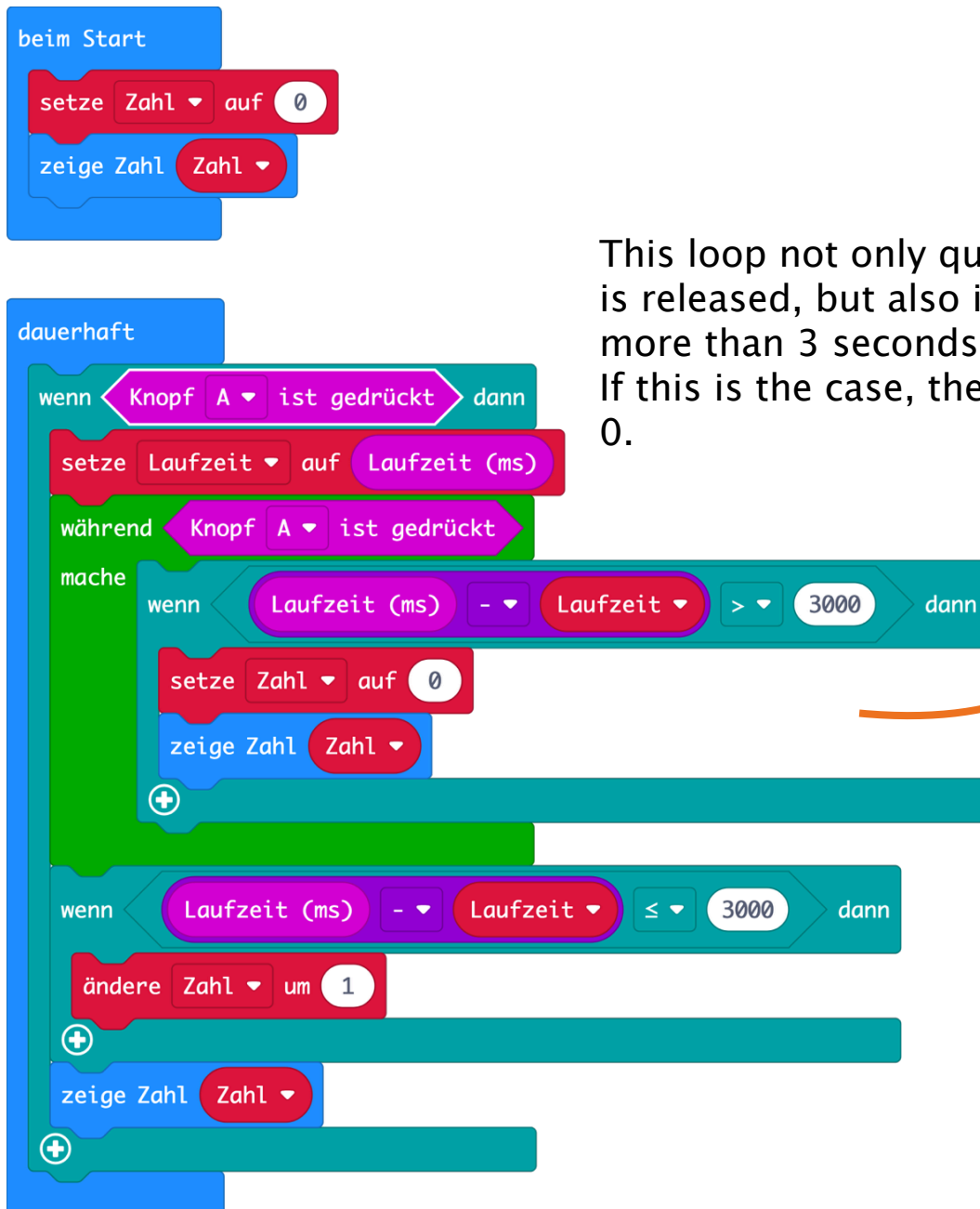
Wait until the button is released.

If the button was pushed for more than 3 seconds, the number is reset to 0.

https://makecode.microbit.org/_8MWLTCPR8fDx

Improvement – Short and Long Button

In the above example, the display was only reset in case of longer button actuation after the button was released. We now want this reset to be shown as soon as the time has elapsed.



This loop not only queries if the button is released, but also if it was pushed for more than 3 seconds. If this is the case, the display is reset to 0.

https://makecode.microbit.org/_autMpUT7gJE7



micro:bit – Pin Assignment

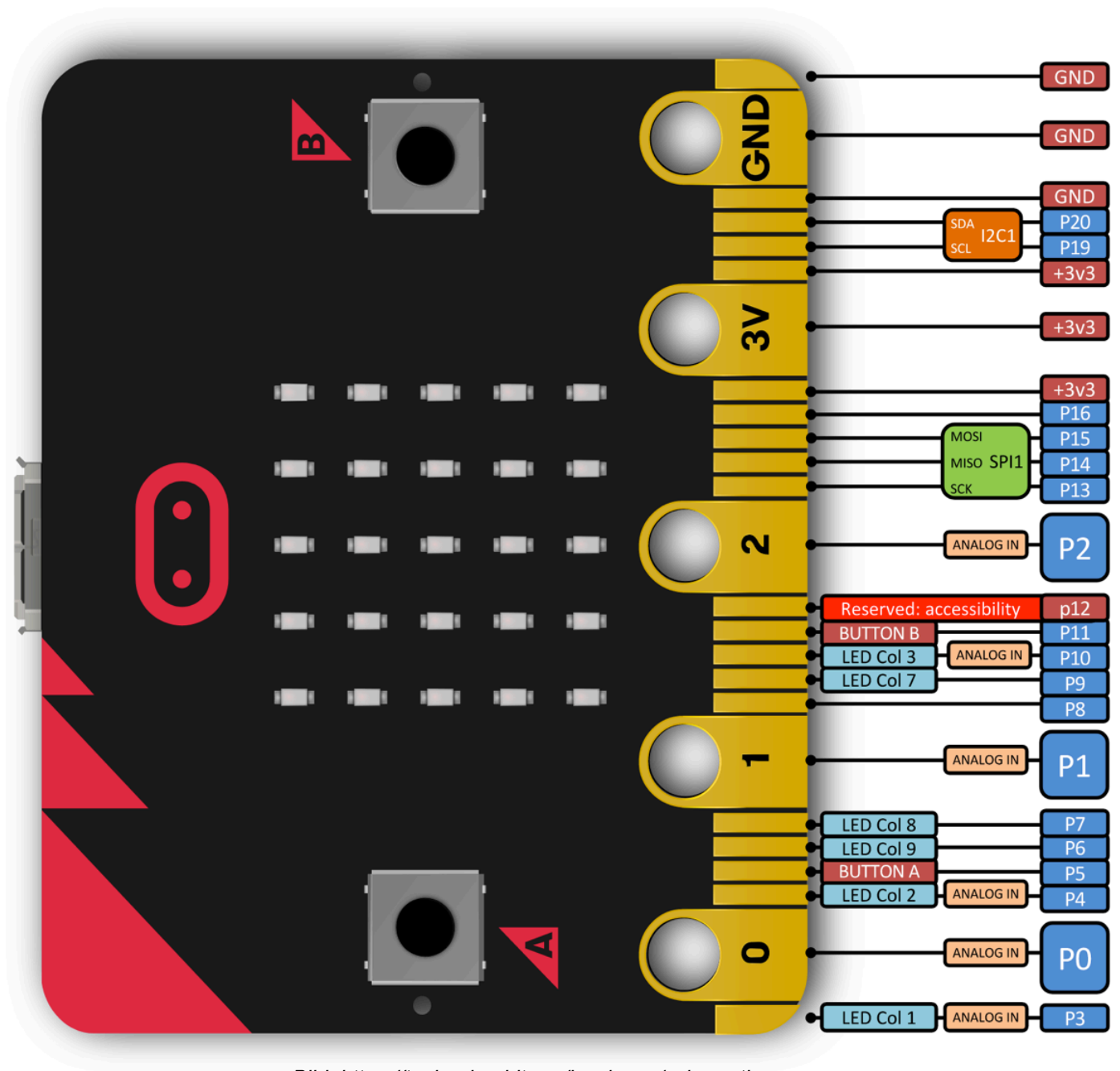


Bild: <https://tech.microbit.org/hardware/schematic>