MAKER FACTORY

# ADVENTSKALENDER FÜR ARDUINO®

# ADVENT CALENDAR FOR ARDUINO®

24 Experimente, die Spaß machen
24 thrilling experiments

## Overview of all experiments

# Arduino® Advent calendar 2019

Programming microcontrollers used to be only for engineers and computer scientists. The Arduino platform offers a clear hardware and easy-to-understand software interface that allows anyone to get started with microcontroller technology.

**The name Arduino**
The Arduino originates from Italy and was named after the Italian King Arduino, who ruled Ivrea until 1005, headquarters of the Arduino manufacturer. The favourite bar thereat of Arduino developers Massimo Banzi and David Cuartielles is named after King Arduino.

**Precautions**
Under no circumstances should you randomly connect any Arduino pins for testing purposes.

Not all Arduino pins can be freely programmed. Some are permanently set up for power supply and other purposes.
Some Arduino pins are directly connected to microcontroller connectors. A short circuit can completely destroy the Arduino - at least theoretically. The Arduino boards are surprisingly stable against switching errors. If two pins are connected via an LED, a series resistor must always be connected in between.
For logic signals, some Arduino-compatible boards require 3.3 V, while others require 5 V. The Nano in this advent calendar uses a +5V signal as a logical HIGH or TRUE.

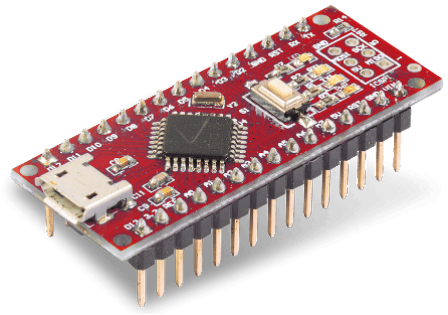
---

## Day 1

**1. Tag**

### Today in the Advent calendar
- Nano Board (Arduino compatible board)

### Nano Board - Arduino compatible board
The Arduino platform now offers a wide variety of boards for different applications. This Advent calendar contains an Arduino-Nano standard compatible board that can be plugged directly into a breadboard to connect additional electronics. Every day, a hardware experiment with its associated program is presented in the Advent calendar.



The Arduino-compatible Nano board.

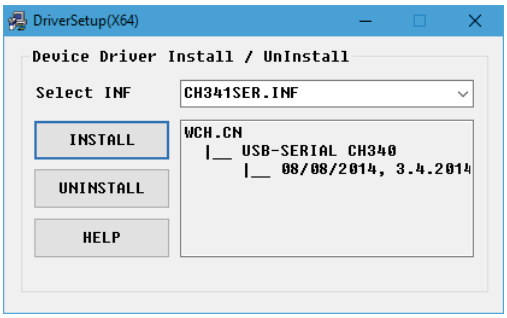
The experiments in these advent calendars are programmed with mBlock 3. This programming language is based on Scratch, one of the easiest programming languages to learn. The used programs can be downloaded here: **bit.ly/c-adventskalender-arduino-19.**
Extract the zip archive into a directory on your hard disk.

### Preparing the Nano
To put the Nano into operation, you need:

- PC with Windows 10, 8.1, 7
- Micro USB cable
- Drivers

The connection between PC and Nano is made via a Micro USB cable. You don't need to buy such a cable. Most smartphones use this type of connector. The cable is used simultaneously for power supply and data transmission.

If possible, connect the cable to a USB 2.0 port on your PC, as USB 3.0 ports are more likely to cause connection problems. For better differentiation USB 3.0 ports are mostly blue.




Installation of the device driver for the Nano board.

## Software installation in a nutshell

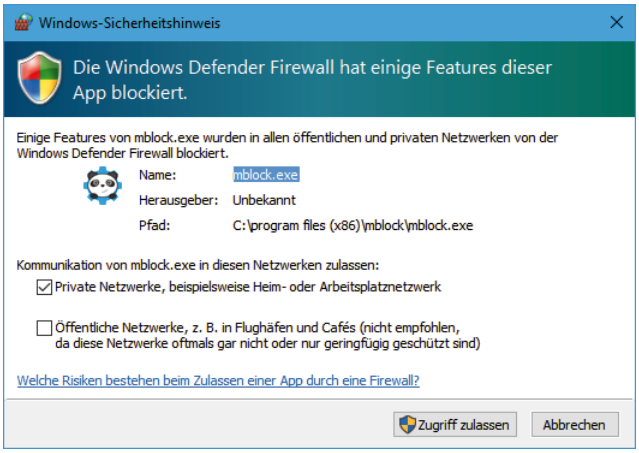
The driver installation in four steps:

1. Extract the ZIP archive with the downloaded software into any folder below your Windows user folder.

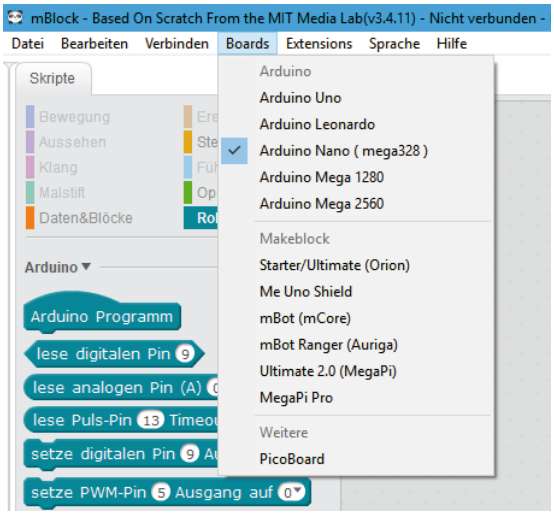
2. Extract the ZIP archive `67006-9-nano-board-Treiber.zip` into a separate folder.

3. Connect the Nano via the USB cable and then start the driver installation with the file `CH341SER.EXE` from the `Windows` subfolder of the driver folder. To install, you must confirm a Windows User Account Control request.

4. Click on **Install** in the installation dialog and wait until the driver installation confirmation appears.




Allow firewall access.



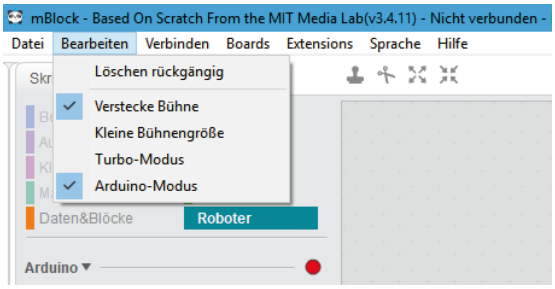

Select the Nano Board.

## mBlock 3

For the projects in the Advent calendar we use the easy-to-learn programming language **mBlock 3. Download** it from **www.mblock.cc/mblock-software** and install it.

The connection with the Nano is blocked by Windows Defender Firewall by default. mBlock requires a permission that you must grant by clicking **Allow access** in the dialog that appears automatically.

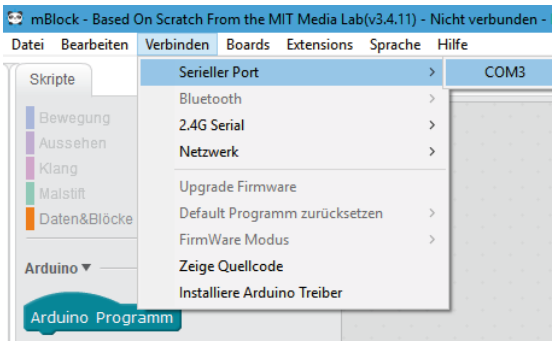
If mBlock 3 does not automatically start in English, select **English from the Language** menu of mBlock. Then select **Edit / Arduino Mode from** the menu.




Switch on Arduino mode.

All supported boards are listed in the **Boards** menu. Select here the **Arduino Nano**.

In order to transfer a program to the Nano Board, a connection must be established between the PC and the Nano. Select the appropriate port in the **Connect / Serial Port** menu. As long as only one COM port is displayed here, select it. If two interfaces are displayed, in most cases the one with the higher number is the correct one.




Select the interface for the connection.

The necessary firmware is automatically installed on the Nano Board during the transfer of programs.

## LED flashes

In mBlock 3 you do not need to type any program code when programming. The blocks are simply attached to each other using drag and drop. The block palette in the left part of the window contains the available blocks, sorted by topic.

**The programs for the Advent calendar**
The programs for the Advent calendar can be downloaded, numbered by days. Select **File / Load Project from** the menu to open a program. But you can also simply assemble them yourself every day using the illustrations.

The first program `01mblock` uses the most important blocks:

The Arduino program block from the Robots palette forms the start for programs which cannot run on the PC but only on an Arduino-compatible board.

**A sequence of 10 iterations is looped from the control palette, the loop repeats 10 times. In the number field - as is the case for some other blocks - you can enter any (meaningful) value.**

**Set digital pin xx to a value contained in the Robot palette and set one of the digital pins of the Nano board to a logic value HIGH or LOW. The values can be selected by clicking on the small arrow to the right of the field.**


`The 01mblock` program causes the LED on the Nano to flash briefly 10 times.

**The LED on pin 13**
For status displays without additional hardware, the Nano has its own built-in LED, which can be controlled via pin 13.

**wait x seconds as indicated in the control palette, the program will wait a certain time until the next step.**

**Switch off the Nano**
The Nano has no circuit breaker, all you have to do is disconnect the power supply and it switches itself off. The next time the unit is switched on, the last stored program starts automatically. The same thing happens when you press the button on the Nano.

## How the program works

The loop ensures that the blocks it contains are executed ten times in succession.

After the LED at pin 13 is switched on, a 0.1 seconds delay occurs until the LED lights up. The LED at pin 13 is then switched off again. Now the program waits 0.1 seconds again. The cycle then repeats itself from the beginning.

**Decimal point instead of comma**
mBlock 3, like many Chinese or American programs, uses the dot as decimal separator, not the comma as used in Germany.

Zurück | Upload zum Arduino | Mit der Arduino IDE editieren

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
    pinMode(13,OUTPUT);
    for(int __i__=0;__i__<10;++__i__)
    {
        digitalWrite(13,1);
        _delay(0.1);
        digitalWrite(13,0);
        _delay(0.1);
    }
}

void loop(){
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

```
avrdude: input file C:\Users\chris\AppData\Local\Temp\build1243307170804926706.tmp/project_01nano4
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.71s

avrdude: verifying ...
avrdude: 3116 bytes of flash verified

avrdude done.  Thank you.
```

send encode mode: ⊙ Binär-Modus ○ Zeichen-Modus
recv encode mode: ⊙ Binär-Modus ○ Zeichen-Modus
Senden

Program transfer to the Nano Board.


Message at the start of transfer.

### Transferring the program to the Nano

The right part of the mBlock screen shows the automatically generated Arduino program code. Click on Upload to Arduino in the upper left corner to automatically compile the program and transfer it to the Nano board.

Click close in the Start Upload message to follow the transfer.

In the lower part of the program window you can see the progress of the transfer. When the text **Thank you** appears, the transmission is complete.

The program starts automatically after the transfer to the Nano. You can also restart it at any time by pressing the button on the Nano.

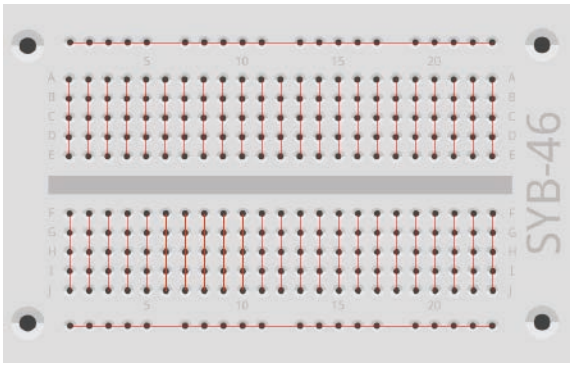
---

**2. Tag**

## Day 2

### Today in the Advent calendar

- 1x breadboard (SYB-46)
- 1x red LED with series resistor




The connections on the breadboard.

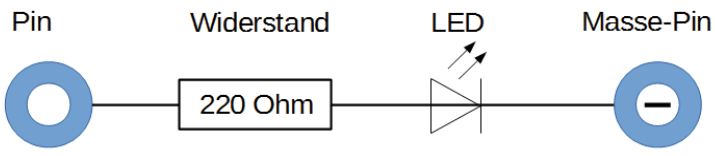
### Breadboard

For the quick assembly of electronic circuits without resolving to soldering, a breadboard is made available in the Advent calendar. Here, electronic components can be inserted directly into a hole grid.

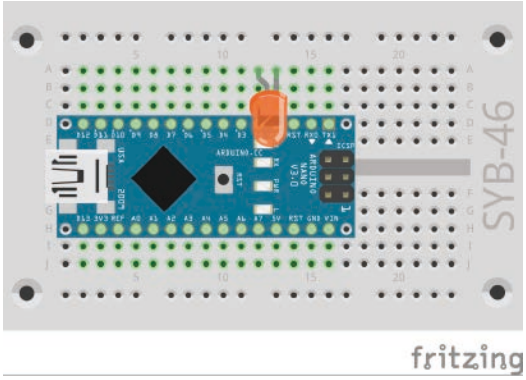
With this breadboard, the outer longitudinal rows are all connected with contacts (X and Y). These contact rows are often used as positive and negative terminals to power the circuits. In the other rows of contacts, five contacts (A to E and F to J) are connected transversely with a gap in the middle of the board. Thus, larger components can be inserted in the middle and wired to the outside.

### LEDs

LEDs light up when current flows through them in the forward direction. LEDs are shown in circuits with an arrow-shaped triangle symbol indicating the direction of flow from the positive pole to the negative pole or to the ground line. An LED allows almost any amount of current through in the forward direction, wherein it has a very low resistance. In order to limit the current intensity and thus prevent the LED from burning out, a 220 ohm series resistor must usually be installed between the connection pin used and the anode of the LED or between the cathode and the ground pin. This series resistor also protects the output of the Nano board from excessively high currents. The LEDs in the Advent calendar have the series resistor already installed and can therefore be connected directly to the pins.




Circuit diagram of an LED with series resistor.




LED flashlight on the Nano.

**How should I insert an LED?**
The two connecting wires of an LED are different in lengths. The longer one is the positive pole, the anode, the shorter one is the negative pole, the cathode. Easy to remember: The plus has an additional line compared minus, making the wire a little longer. In addition, most LEDs are flattened on the minus side, comparable to a minus sign. Easy to remember: cathode = short = edge.

### Alternating flashing light

A simple program `02mblock` lets two LEDs flash alternately: the built-in one on the Nano board and the external one on the breadboard. It is often difficult to plug the Nano Board into a new breadboard. Apply some even pressure, but DO NOT use tools such as a hammer or the like.

**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor

Make sure that the cathode (short wire) of the LED is connected to the **GND pin**, the anode (long wire) to the **D2 pin**.

**The pins on the Nano**
All pins marked D... are digital inputs or outputs, which can take the values True or False (On or Off). The pins marked A... are analogue inputs. GND pins are ground lines. Arduino-compatible boards work with different voltages and have two different plus pins as standard. Pin 3.3 has a +3.3 V voltage. Pin 5V has a +5 V voltage. The Nano board in the Advent calendar required a potential of +5 V for a logical true signal, some other boards require only +3.3 V.

### The program
The `02mblock` program causes the built-in LED on the Nano and the externally connected LED to flash alternately.

### How the program works
A continuous loop ensures that the two LEDs to flash endlessly alternately.

After the built-in LED on pin 13 is switched on, the LED on pin 2 is switched off. Now the program waits 0.1 seconds.

Then the LED at pin 2 is switched on in the same way and the LED at pin 13 is switched off. The cycle repeats itself after another 0.1 seconds.


The `02mblock` program lets two LEDs to flash alternately.

---

## Day 3

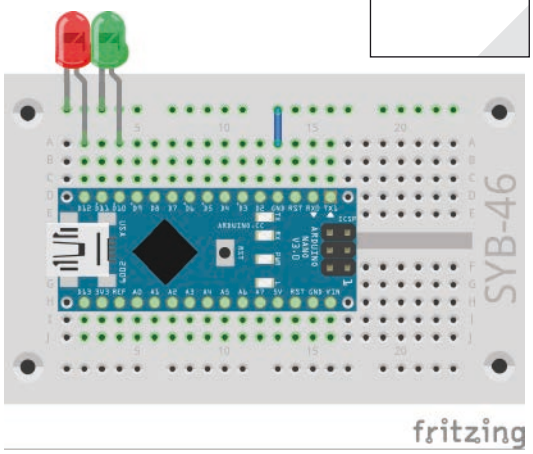
3. Tag

### Today in the Advent calendar
- 1x green LED with pre-resistor
- 1x connection cable

### connecting cable
Today a connecting cable is part of the Advent Calendar. These cables are used to connect contact strips to each other on the breadboard. The connecting cables have small wire plugs at both ends with which they can be easily plugged into the breadboard.

### LEDs flash with adjustable speed
The experiment of the third day causes two LEDs to flash alternately again. However, you can set the speed using a variable in the program.

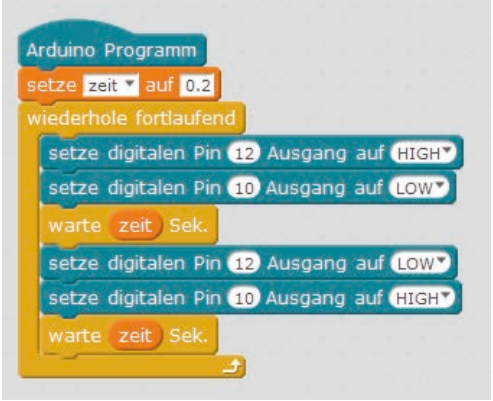
**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor, 1x green LED with series resistor, 1x connecting cable



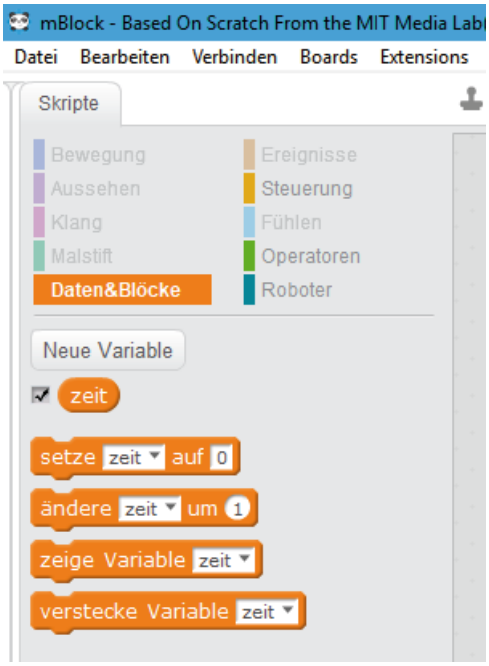

Alternating flashing LED light on the Nano.

Today's circuit shows the typical circuit design on the breadboard. One of the horizontal contact strips is used as a ground wire, which is connected to the GND pin on the Nano via a connecting cable. When setting up the circuit, make sure that the cathodes (short wire) of the LEDs are inserted into the ground strip. The anodes (long wire) in this circuit are connected to pins **D10** and **D12** on the Nano.

### The program
The `03mblock` program works similarly to the one of yesterday and lets the LEDs connected to pin **D10** and pin **D12** to flash alternately this time. The flashing frequency can be set via a **variable time.**




The program `03mblock` lets two LEDs to flash alternately with adjustable speed.




The data & blocks block palette with one variable.

### How the program works

The **repeat continuous loop** causes the two LEDs to endlessly flash alternately again. A variable is used instead of a fixed time between the switching operations.

**Variables in mBlock**
Variables are small memory locations where you can assign a number or anything else during a program. When the program ends, these variable memories are automatically cleared again. Variables must first be created in mBlock on the Block named Palette Data & Blocks using the New Variable button before they can be used. You can then drag the symbol of the newly created variable from the block palette into a corresponding field of the block in the program. Various blocks for reading and changing the variable are also available on the block palette.

Once created, a variable appears as an orange icon on the block palette.

In the program, the set value of the variable **time is** used for both waiting times. If you enter 0.2 as a different value in the set time block at the beginning of the program, the LEDs flash correspondingly faster or slower. The program must then be transferred to the Nano again.

---

**4. Tag**

## Day 4

### Today in the Advent calendar

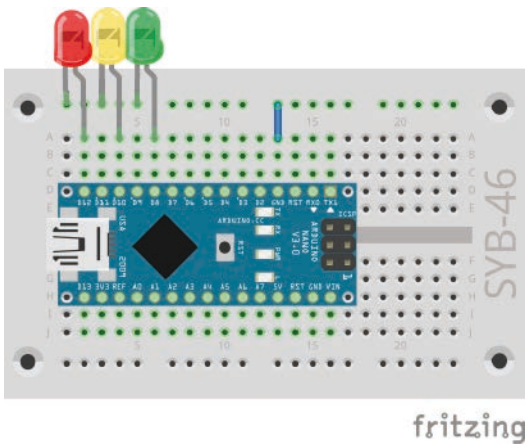
- 1x yellow LED with series resistor

### Traffic light circuit

Turning an LED on and off may be exciting at first, but you don't really need a microcontroller to do that. A traffic light with its typical light cycle from green via yellow to red and then via a light combination red-yellow and back to green demonstrates further programming techniques.


The traffic light program.

**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x connecting cable




Traffic light switch on the breadboard.

### The program

`The 04mblock` program works similar to yesterdays. Again different digital pins are alternately switched on and off in an endless loop. For the sake of clarity this time the pins of the LEDs are called variables, which also has the advantage to ease the circuit conversion into other pins.

### How the program works

Initially, the three variables are set to the pin numbers used for the LEDs. Afterwards the LEDs are set to a defined state before the start of the main loop. Only the green LED should light up.

When using variables, mBlock 3 cannot recognize at the beginning which pins are to be used as outputs and initializes them only at their first use. Therefore, it is not possible to assume a clearly defined state of the LEDs at program start. To avoid this, the script switches off all LEDs at the beginning and initializes them with it.

An endless loop starts whereby the typical traffic light cycle runs in each iteration. The yellow and red/yellow phases are each only 0.6 seconds long, whereas the traffic light is red and green 2 seconds each.

---

## Day 5

**5. Tag**

### Today in the Advent calendar

- Switching wire (insulated)

### Switching wire

Today the switching wire is part of the Advent Calendar. This allows you to create short connection bridges onto which contact rows on the breadboard are connected. Such wire bridges lie flat on the breadboard and are less disturbing than the long connecting cables when assembling the other components.

Cut the wire with a small side cutter to the appropriate length depending on the experiment. In order to be able to insert the wires better onto the breadboard, it is advisable to cut them slightly diagonally so that a kind of wedge is created. Remove the insulation at both ends over a length of about half a centimetre.

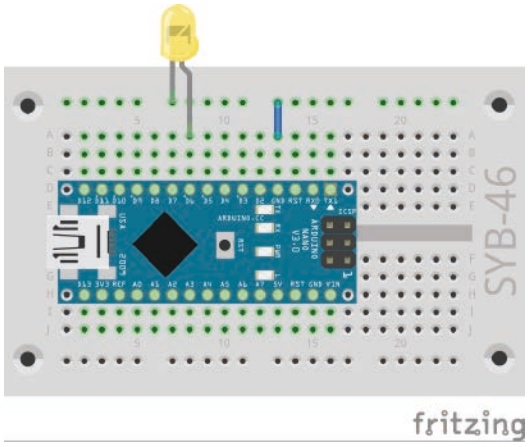
### Diming LEDs

LEDs are typical components for the output of signals in digital electronics. Two different states can be assumed: on and off, 0 and 1 or **LOW** and **HIGH**. The same applies to the digital pins defined as outputs. Therefore, it would theoretically not be possible to dim an LED.

Left: 50% Duty cycle - right: 20% Duty cycle.

With a trick it is still possible to control the brightness of an LED on a digital pin. If an LED flashes fast enough, the human eye no longer perceives it as flashing. The technique known as pulse width modulation (PWM) generates a pulsating signal that switches on and off at very short intervals. The voltage of the signal always remains the same, only the ratio between Level **LOW** (0 V) and Level **HIGH** (+3.3 V) is changed. The duty cycle indicates the ratio of the length of the switched-on state to the total duration of a switching cycle.

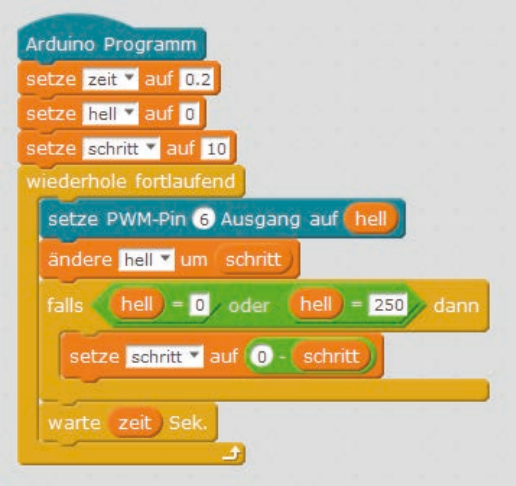
The smaller the duty cycle, the shorter the LED flash time within a switching cycle. This makes the LED appear darker than a permanently switched on LED.




An LED is dimmed.

**Components:** 1x Nano Board, 1x breadboard, 1x yellow LED with series resistor, 1x wire bridge

**Pins for PWM signals**
Pins 3, 5, 6, 9, 10, 11 are marked with a '-symbol on the circuit diagrams. These pins can be used for PWM.




The 05mblock program dims an LED at the PWM output.

### The program

The program 05mblock dims the LED at pin **D6** cyclically brighter and darker. Some new program blocks are used.

### How the program works

Three variables are defined at the beginning: **time determines** the speed for dimming, **light** indicates the PWM value for the brightness of the LED and **step indicates the step** size for dimming.

Now an endless loop begins. First, the current value of the variable bright is outputted to pin 6 as a PWM value for each iteration. On the **robot** block pallet there is a separate block for PWM pins. The value of the variable **is** then incremented by the value **step.**

In the next step, an "**if...then...**" block from the **control** block palette enables to check whether the value of **light** has reached the limits 0 or 250. In this case, an **or block** from the **Operators** block palette is used, which in turn contains space for two more queries. If at least one of these two is true, the **or block returns** the value **true**, and the content of the **if block** is executed. Blocks with queries are marked by pointed ends. They can be nested within each other and built into control blocks that contain fields in this form.

Two equality queries check whether the value of the variable **light** has reached the value 0 or 250. If this is the case, the variable **step** is set to a new value. Since mBlock 3 does not provide a way to reverse the sign of a variable, we use the '-' operator and subtract the value of the variable from 0, giving the same result.

Finally, the program waits for the 0.2 seconds stored in the variable **time in** each loop pass. The endless loop then restarts and supplies the LED with a new PWM value.

---

**6. Tag**

## Day 6

### Today in the Advent calendar

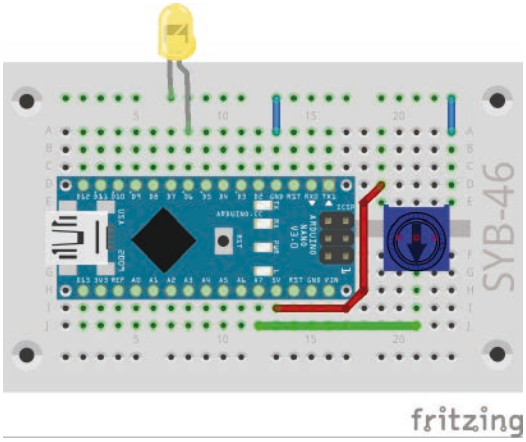
- 15-kOhm potentiometer-
1x connection cable

### Potentiometers

The potentiometer from today's Advent calendar is an adjustable resistor that can take values between 0 Ohm and 15 kOhm by turning the knob.

### Dimming an LED with a potentiometer

I order to actually dim an LED as with a living room lamp, we use the potentiometer from the Advent calendar. This adjustable resistor can be used to build a voltage divider that can supply any voltage between 0 V and +5 V. The voltage divider can also be used to adjust the voltage of the resistor. However, this analogue voltage cannot be transmitted directly to the LED, but must be converted into a digital PWM value.




LED dimmer with potentiometer.

**Components:** 1x Nano Board, 1x breadboard, 1x yellow LED with series resistor, 1x potentiometer, 2x connecting cable, 2x wire bridge

The two short wires are wire bridges. For the two longer cables you can use the two connecting cables or cut wire bridges to size.

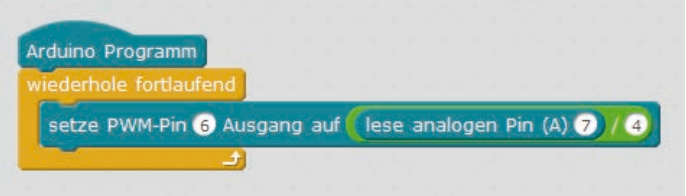
The Nano Board has eight analogue input pins **A0...A7, which** convert an applied voltage into a numerical value.

### The program

The program 06mblock dims the LED with the potentiometer. It is connected to analogue input **A7 for** this purpose.

The analogue inputs of the Nano evaluate an analogue voltage value and provide digital values between 0 and 1023. 0 stands for 0 V and 1023 for +5 V voltage at the respective pin. These values must be converted to values ranging between 0 and 255 in order to output them again as PWM values to an LED.

Analog inputs that are not connected do not have a defined state. The values read out there fluctuate due to the minimal electric fields on the Nano Board and nearby. In order to really evaluate an analogue value, a clearly defined voltage must be applied at all times. A variable resistor, a potentiometer or any sensor should always be connected as a voltage divider between 0 V and +5 V. The voltage of the sensor should be between 0 V and +5 V.




The `06mblock` program dims the LED with a potentiometer.

The mBlock-3 program `06mblock`, which evaluates the potentiometer, converts the value and sends it to the LED, is very simple.

### How the program works

In an endless loop, the value at analogue pin **A7 is** constantly read out, converted and output to PWM pin **D5** again.

The "**read analogue pin (A)...**" block from the **Robot** block palette reads the analogue voltage value at one of the analogue pins and returns a numerical value between 0 and 1023.

The value provided by the analogue pin is divided by 4 to provide a value between 0 and 255 that can be used for PWM.

The calculated value is then outputted via the "set **PWM-Pin...Output to...**" block **to** the PWM pin and causes the LED to light up accordingly.

---

## Day 7

**7. Tag**

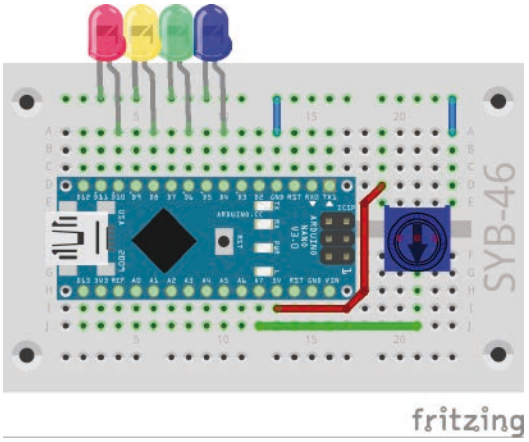
### Today in the Advent calendar

- 1x blue LED with series resistor

### Controlling chaser light with potentiometer

A potentiometer on an analogue input pin of the Nano board controls the speed of the chaser light.

**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x potentiometer, 2x connecting cable, 2x wire bridge




Controlling chaser light with 4 LEDs with potentiometer.

### The program

The `07mblock` program controls an LED bar according to the setting of the potentiometer.


The program for the chaser light.

The `07mblock` program lets the four LEDs light up cyclically as chaser lights. In each case, the pin numbers have a distance of 2 from each other. So they can simply be counted up in a loop.

### How the program works

At the beginning, the four pins used for the LEDs are switched off once, so they can be initialized, since they are only addressed later via a variable and would otherwise not be initialized correctly as outputs.

Then the endless loop starts, which sets the variable **n** to 4 in each pass, the number of the first LED. Each cycle of the endless loop causes a chaser light cycle of all four LEDs.

An inner loop lets the four LEDs light up shortly after each other. After each LED the variable **n** is increased by 2 and thus the next LED is activated. After all LEDs have illuminated once, the next cycle of the endless loop starts with the LED at pin 4.

The light duration of the individual LEDs is set interactively via the potentiometer at analogue input **A7.** Analog pin **A7** is queried in each iteration and its value is multiplied by 0.001 as to obtain useful values for the chaser. The value calculated in this way determines the time it takes for each individual LED to light up.

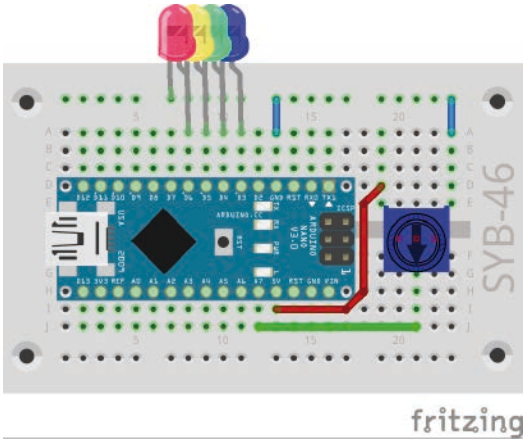
---

8. Tag

## Day 8

### Today in the Advent calendar

- 3x connecting cable




Four LEDs flash randomly.


The `08mblock` program lets four LEDs to flash randomly.

### LEDs flash randomly

The experiment of the eighth day has four LEDs flashing in random order. This time the LEDs are very close to each other on the breadboard, because the program needs consecutive pin numbers.

**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x potentiometer, 2x connecting cable, 2x wire bridge

### The program

The `08mblock` program lets four LEDs with consecutive pin numbers flash randomly alternately.

**How are random numbers created?**
It is generally thought that nothing can happen accidentally in a program - so how can a program generate random numbers? If you divide a large prime number by any value, starting at the umpteenth decimal place you get numbers that are hardly predictable. They also change without any regularity if the divisor is increased regularly. This result is apparently random, but it can be reproduced at any time by an identical program or by calling the same program several times. But if you take a number composed of some of these digits and divide it again by a number that results from the current computer time or the content of any memory location of the computer, you get a result that cannot be reproduced and is therefore called a random number.
In Arduino mode, mBlock 3 uses a random function that actually generates reproducible values, but this is not noticeable in this program. In later programs a self-programmed random number generator is therefore used.

### How the program works

At the beginning the four pins of the LEDs are switched off and initialized at the same time. Then an endless loop starts.

In the first step of each loop pass, the variable **n** is set to a random number between 3 and 6. This indicates the pin number of the LED to be switched on. Therefore the circuit requires consecutive pin numbers.

As in the last program, the speed of the colour change is controlled by the potentiometer connected to the analogue input **A7**, which is queried once in each loop pass.

The randomly selected LED is switched on for the set time and then switched off again. In the next loop pass a new LED is randomly selected. It can happen that the same LED lights up several times in a row.

# Day 9

9. Tag

## Today in the Advent calendar

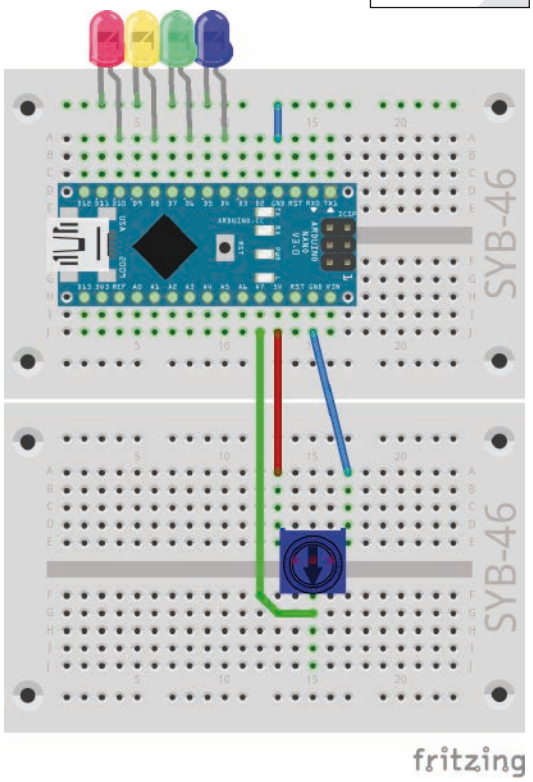
- 1x breadboard (SYB-46)

## Analog level display with LEDs

Analogue values can be read off at a glance on a level display. Such displays consisting of several LEDs are used, for example, in volume or temperature controllers. The experiment of the ninth day shows the value set on the potentiometer via a series of four LEDs.

**Components:** 1x Nano Board, 2x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x potentiometer, 3x connection cable, 1x wire bridge

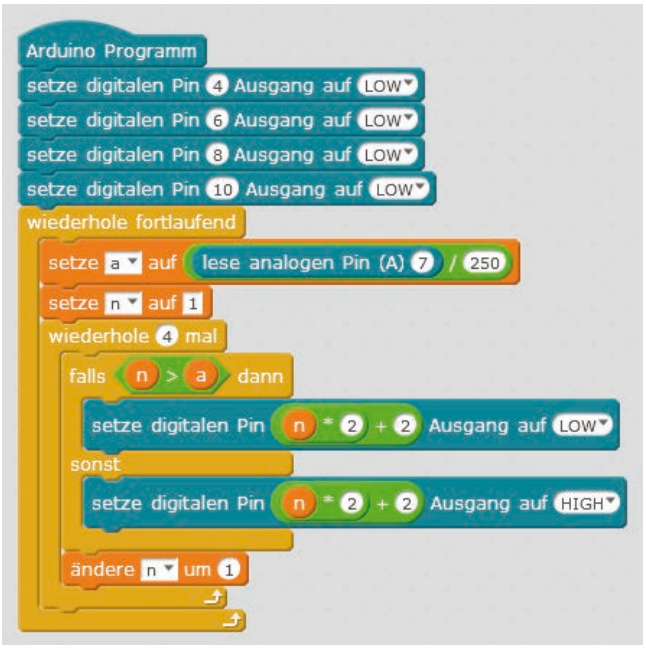
A second breadboard is used today in the Advent calendar. This allows the potentiometer to be set up in such a way as to ease its access.




Level display with 4 LEDs. and potentiometer.

## The program

The `09mblock` program reads the set value of the potentiometer at analogue input **A7** and displays it by means of a loop on four LEDs.




Analog level display on four LEDs.

## How the program works

In the main loop, the analogue value of the potentiometer is first read out, divided by 250 to obtain values between 0 and 4, and stored in variable **a. The value of the** potentiometer is then read out in the main loop. The analogue input provides values between 0 and 1023. If the value were divided exactly by 256, the value 4 would never be reached. The results would all be between 0 and 3.99. The program divides the analogue input value by 250, resulting in a value just above 4 at the very end.

The counter **n** starts at 1. Now a loop starts again, which runs four times for the four LEDs. For each LED, the system checks whether the number of LED **n** is greater than the level value stored in **a. If** this is the case, the corresponding LED is switched off. To do this, multiply the value **n** by 2 and then add 2 to get the pin number of the corresponding LED.

If the LED number is not greater, i.e. less than or equal to the level value, the LED is switched on. In this way, all LEDs are always checked, regardless of whether the potentiometer is turned up or down.

10. Tag

# Day 10

## Today in the Advent calendar

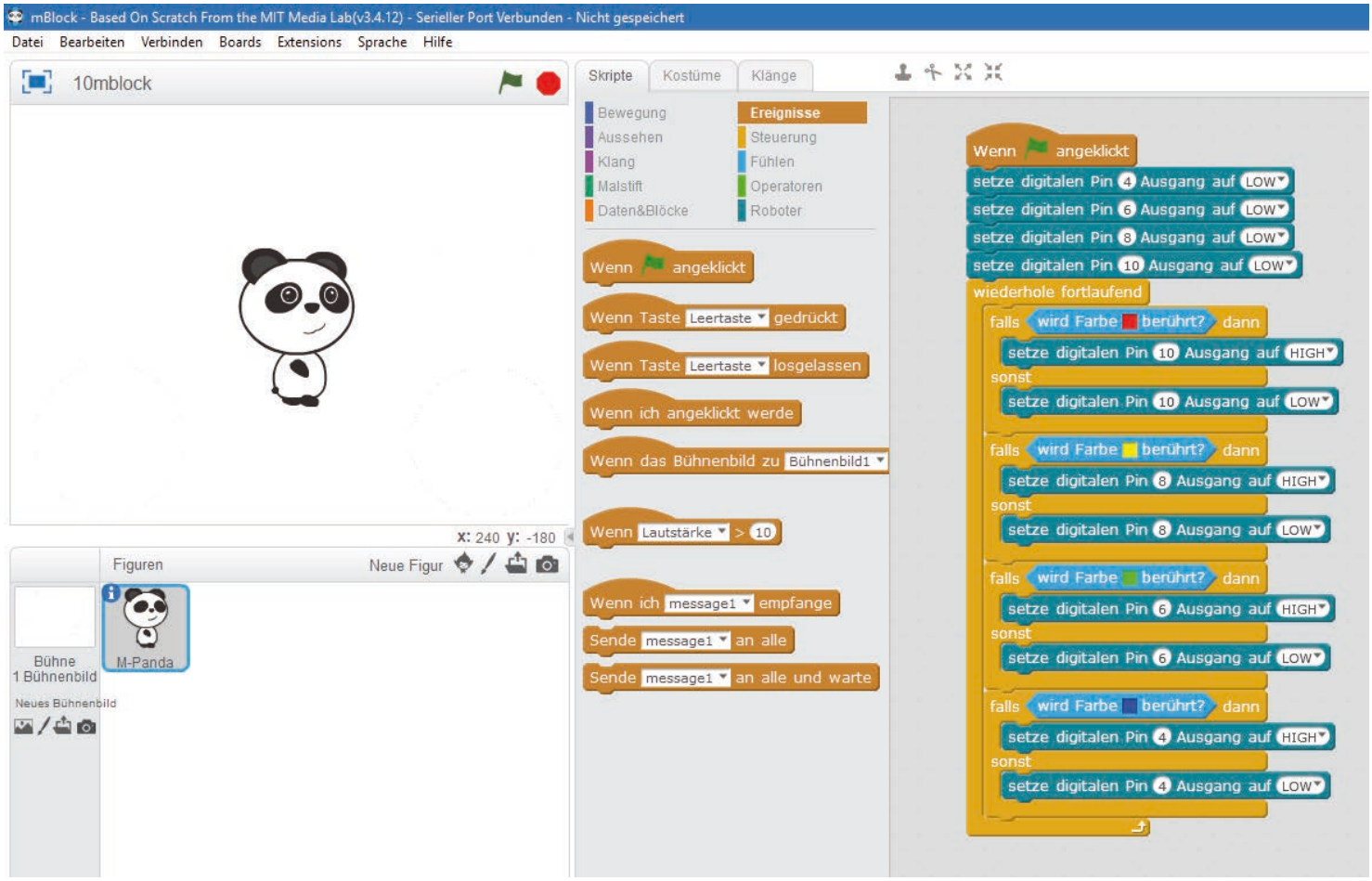
- 2x connecting cable

## Interactive control of Nano board via mBlock 3

mBlock 3 not only allows Arduino sketches to be generated and then loaded onto the Nano, but also provides an interactive mode in which the Nano is connected in real time to the program running on the PC.

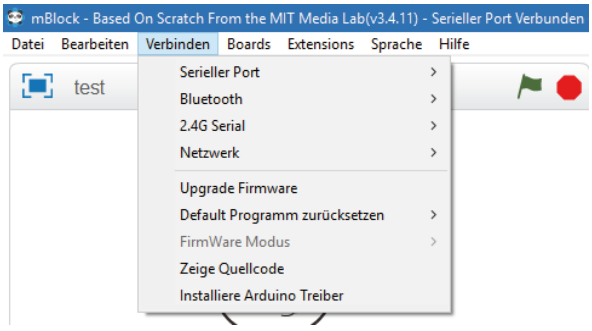
## Setting up a real-time connection with the Nano

Turn off the **Arduino mode** in the **Edit** menu. The window with the Arduino sketch disappears on the right. In the upper left corner the so-called stage for graphic programs with the M-Panda symbol figure appears.




mBlock 3 in standard mode.

Select the COM interface used in the **Connect / Serial Port** menu. The title bar of the window shows **Serial Port Connected**. If the connection is not established, select **Connect / Install Arduino Driver** from the menu.




Transferring mBlock 3 firmware to the Nano.

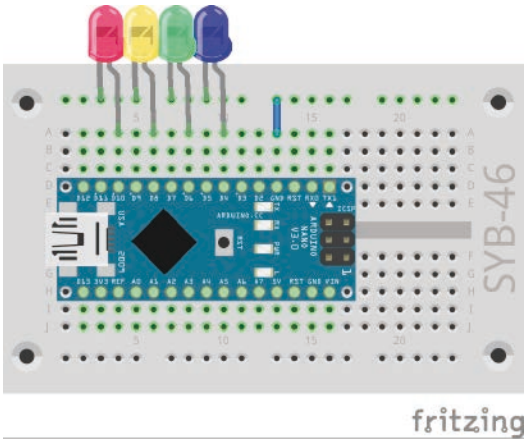
Once the connection is established, select **Connect / Upgrade Firmware from** the menu. This installs a special firmware for real-time communication to the Nano, which takes a few seconds, while the one LED on the Nano flashes rapidly. Don't be surprised that the percentage display on the screen stops at 0%.

**Components:** 1x Nano board, 1x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x jumper wire

## The program

The `10mblock` program lets one of the four LEDs light up when the M-Panda symbol figure touches the surface in the corresponding colour.


The stage with a stage design and the M-Panda character.




Four LEDs show the position of the M-Panda.


The `10mblock` program checks the position of the M-Panda and lets the LEDs light up.

The **RX** and **TX** LEDs on the Nano are flashing constantly, because in interactive mode data is permanently sent back and forth between the Nano Board and the PC.

### How the program works

mBlock 3 is based on the Scratch programming language and contains in interactive mode also its complete range of functions. In the Arduino mode only a part of the functions is available, which can be converted directly into Arduino sketches.

The "**when green flag is clicked**" block is one of the important standard blocks from Scratch. It starts a program when the user clicks on the green flag at the top of the stage.

Click on the green flag to start the program. All four LEDs are initialized and switched off.

Now you can move the figure M-Panda with the mouse on the stage. If it touches one of the coloured surfaces, the corresponding LED lights up.

An endless loop asks one after the other whether the M-Panda touches one of the four colours. "**If... then...else**" **blocks** are used. If the colour is touched, the appropriate LED is switched on, if it is not touched, the LED is switched off. Without switching off, all LEDs would eventually be switched on when the M-Panda has touched all colour areas once.

In order to select the colour in the "colour ... is touched" block, tap the colour field of the block. An eye-dropper appears, allowing you to select the desired colour on stage.

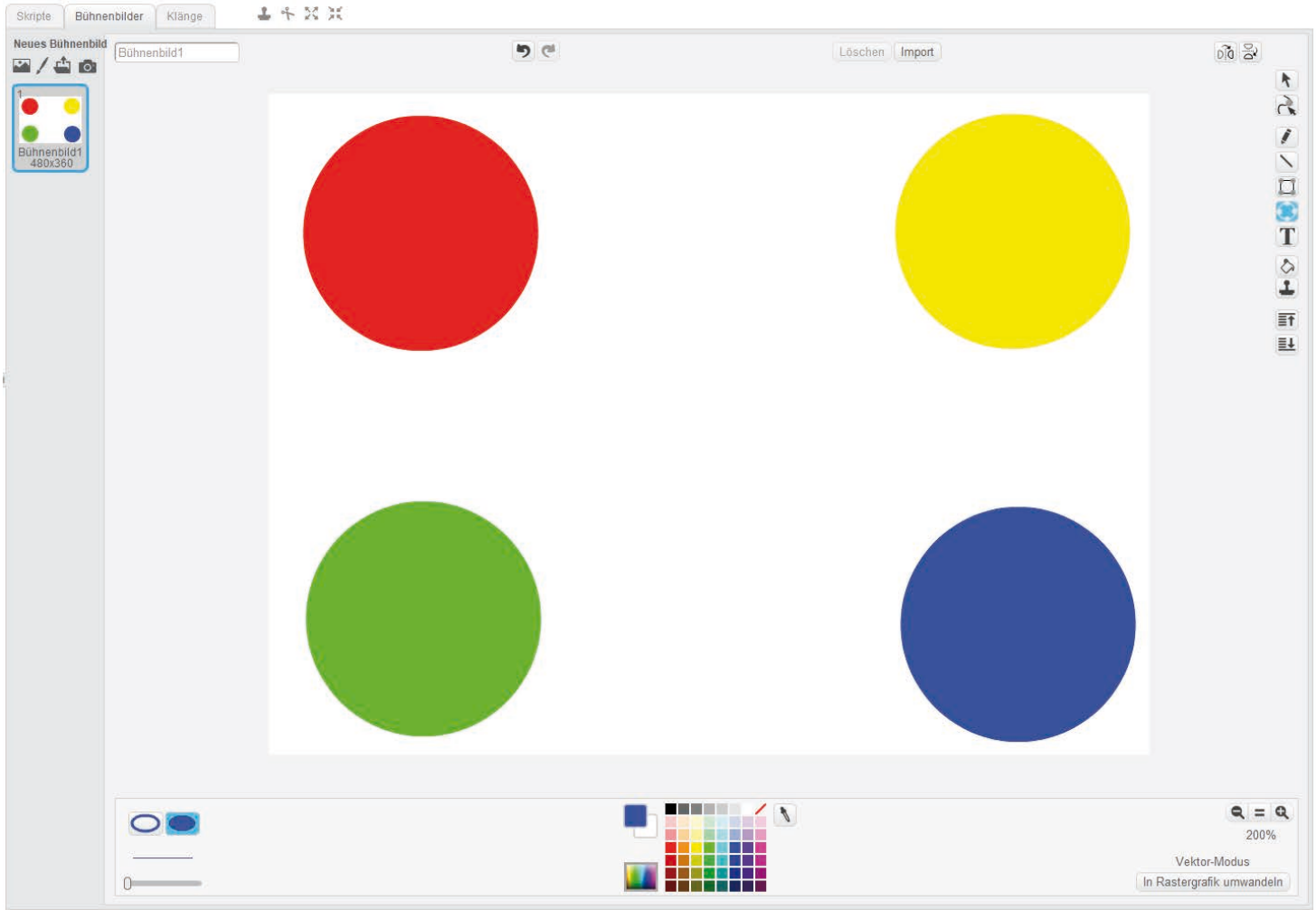
The loop is repeated endlessly until the user exits the program by clicking on the red stop icon in the upper right corner of the stage.

### The stage design

Each scratch program can contain different stage sets. Click on the stage design on the left of the figure palette to edit it or paint a new stage design.


The figure palette in mBlock 3.

If you switch to the **Scene** tab at the top, you can edit the set. mBlock 3 contains a graphics program for vector graphics and raster graphics, with which you can design stage sets and new figures.




The graphics program in mBlock 3

With the Circle tool you can create circles while holding down the [Shift] key. Switch to fill mode at the bottom left to draw filled circles. Select the colour on the palette at the bottom in the middle.

Use the selection tool at the top to select and copy the finished circle with [Ctrl]+[C] and [Ctrl]+[V]. In this way you get four circles of the same size, which you only have to copy to the appropriate places.

Finally, fill the circles with the colour bucket tool in the desired colours.

To edit the program again, click on the **M-Panda**. Then switch to the **Scripts** tab at the top. Here you can find the program blocks.

---

## Day 11

**11. Tag**

### Today in the Advent calendar

- 1x button
- 1x 10 kOhm resistor (brown - black - orange)

### Resistors and their colour codes

Resistors are used for current limiting on sensitive electronic components and as series resistors for LEDs. The unit of measurement for resistors is Ohm. 1,000 ohms correspond to one kiloohm, abbreviated kOhm. 1,000 kOhm corresponds to a megohm, abbreviated MOhm. The omega character `Ω` is also often used for the unit ohm.

The coloured rings on the resistors indicate the resistance value. With a little practice they are much easier to recognize than tiny numbers that can only be found on old resistors.

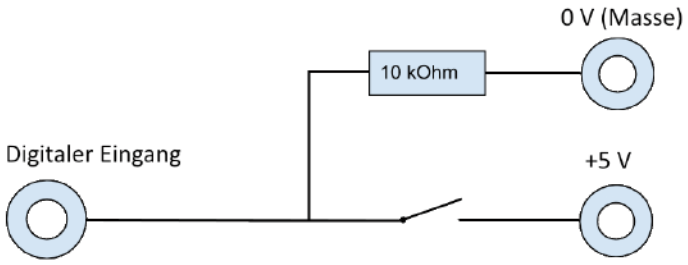
Most resistors have four such colour rings. The first two colour rings represent the digits, the third indicates a multiplier and the fourth the tolerance. This tolerance ring is mostly gold or silver coloured - colours that do not appear on the first rings. Thus the reading direction is always unambiguous. The tolerance value itself hardly plays a role in digital electronics. The table shows the meaning of the coloured rings on resistors.

It does not matter in which direction a resistor is installed. With LEDs, on the other hand, the mounting direction plays an important role.

| Colour | Resistance value in Ohm | | | |
|---|---|---|---|---|
| | 1st 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 % |
| Violet | 7 | 7 | $10^{7}$ = 10.000.000 | ±0,1 % |
| Grey | 8 | 8 | $10^{8}$ = 100.000.000 | ±0,05 % |
| White | 9 | 9 | $10^{9}$ = 1.000.000.000 | |

### Push-button

Digital pins can not only output data, for example via LEDs, but can also be used for data input. In today's project we use a push-button for input, which is plugged directly onto the breadboard. The push-button has four connection pins, whereby two opposite (large distance) pins are connected to each other. As long as the button is pressed, all four connectors are connected to each other. Unlike a switch, a push-button does not engage. The connection is immediately disconnected when released.




Circuit diagram of a push-button with pull-down resistor.

If a +5 V signal is present on a digital input, it is evaluated as logically true.

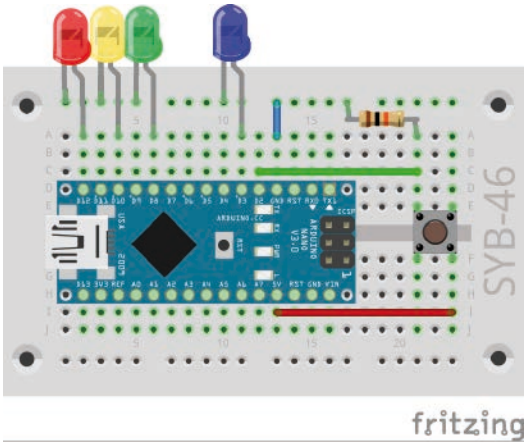
With the button open, the input would not have a clearly defined state. If a program queries this pin, random results may occur. To prevent this, a comparatively high resistance - usually 10 kOhm – is connected to the ground. This so-called pull-down resistor pulls the status of the input pin down to 0 V when the button is open. Since the resistance is very high, there is no danger of a short circuit as long as the button is pressed. When the button is depressed, the voltage of the pin becomes +5 V and the ground line is connected directly via this resistor.


The `11mblock` program `switches the` traffic light when the button is pressed.

### Switching traffic lights with pedestrian flashing light using push-button

The project of the eleventh day represents a simple traffic light switch with pedestrian flashing light. During the red phase of the traffic light, the blue LED flashes. Since there are only four LEDs in the Advent calendar, the circuit does not represent a red/green pedestrian light, as it is common in Central Europe, but a simple pedestrian flashing light, as it is used in some other countries. It shows pedestrians that the traffic lights for cars are red.

**Components:** 1x Nano Board, 1x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x push-button, 1x 10-kOhm resistor (brown - black - orange), 1x wire bridge, 2x connecting cable




Switching the pedestrian light with a push-button.

### The program

The `11mblock` program is based on the traffic light switch of the fourth day. An endless loop regularly checks whether the button is pressed. If this is the case, the traffic light cycle is started. As long as the button is not pressed, the traffic light remains green and the pedestrian flashing light is switched off.

### How the program works

At the beginning, four variables are created for the pin numbers of the four LEDs and another one for the pin number of the push-button. All four LEDs are switched off. The pin for the button does not need to be initialized.

Then an endless loop starts. In each iteration, the digital pin 2 to which the push-button is connected is queried using the "**if ... then ...**" block. If the button is pressed, the query returns the logic value **true**. In this case the already known traffic light cycle starts. During the red phase of the traffic light, a loop switches the blue LED on and off 20 times for 0.05 seconds each.

---

12. Tag

## Day 12

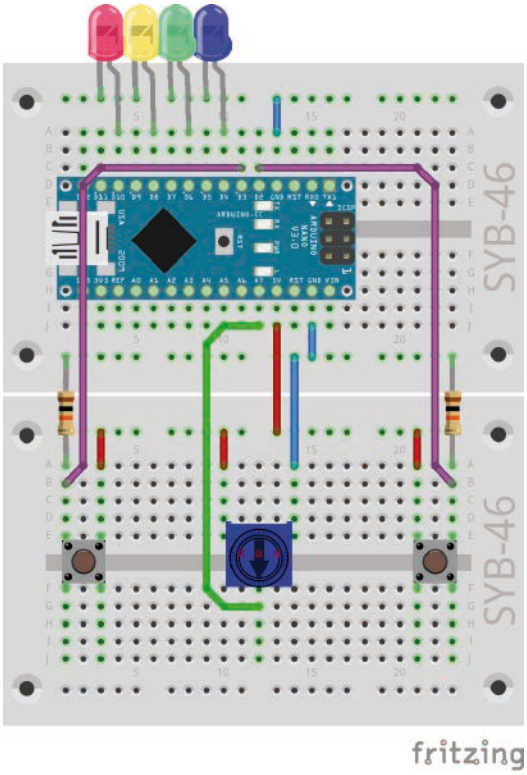
### Today in the Advent calendar

- 1x button -
1x resistor 10 kOhm (brown - black - orange)

### Chaser light in two directions with adjustable speed

Chaser lights are always popular effects, not only for advertising and party rooms. The experiment of the twelfth day lets four LEDs light up as chaser lights when a button is pressed. The other button lets the chaser light run in the opposite direction. A potentiometer controls the speed of the chaser light.

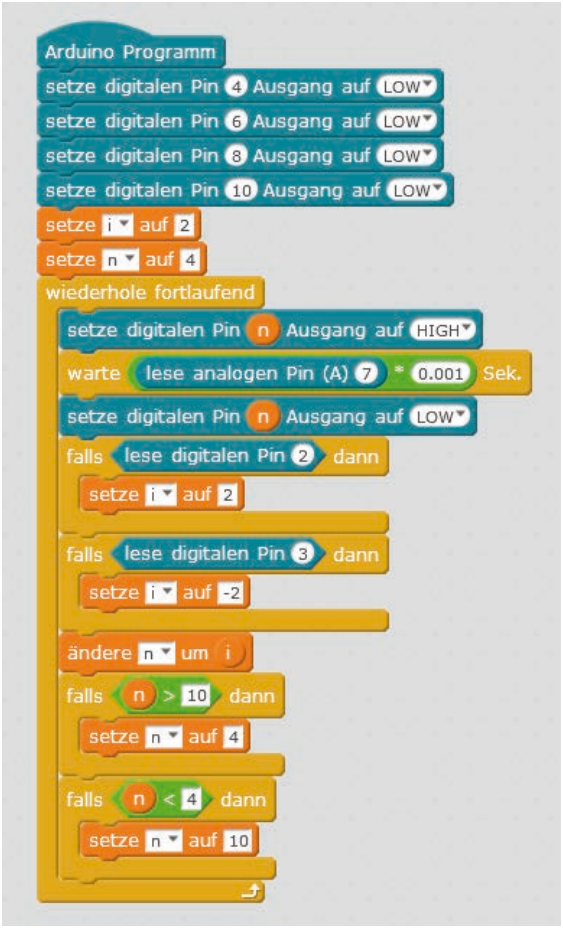
**Components:** 1x Nano Board, 2x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 1x potentiometer, 2x push-button, 2x 10-kOhm resistor (brown - black - orange), 5x connection cable, 5x wire bridge




Chaser light with 4 LEDs, potentiometer and two buttons with pull-down resistors.

## The program

The program `12mblock` works similar to the program of the seventh day, but additionally queries the two buttons to select the direction of the chaser using an additional variable.




Switching the chaser light direction with two buttons.

## How the program works

The variable **i** indicates the value by which the number of the pin for the LED to be lit is changed in each iteration. The value 2 lets the chaser light run from right to left: 4, 6, 8, 10, the value -2 lets it run from left to right: 10, 8, 6, 4.

The variable **n** contains the number of the pin that is currently lit or should be lit next.

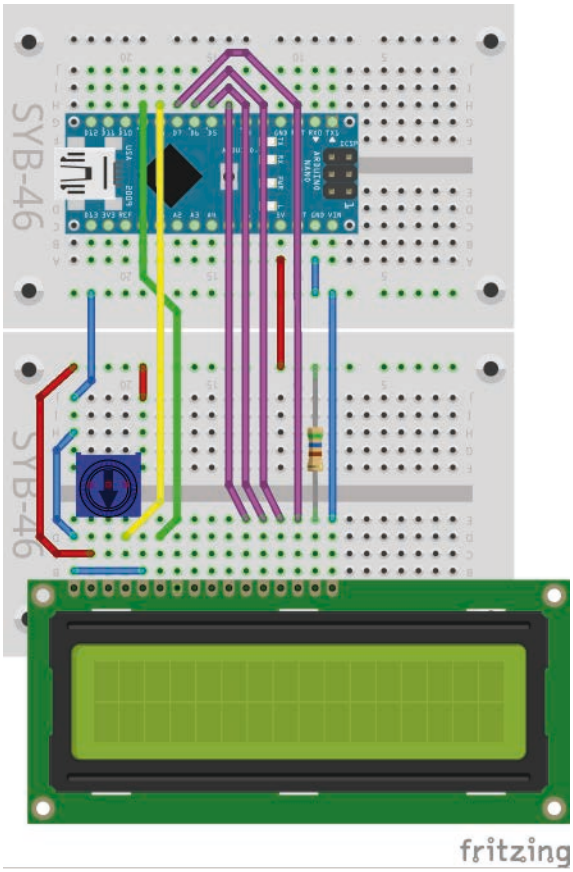
The LED on pin **n** lights up each time the loop is run. The time is determined by querying the potentiometer. The two buttons are then queried. If the button on pin 2 is pressed, the variable **i is** set to 2 so that the chaser light runs to the left. If the button on pin 3 is pressed, the variable **i is** set to -2 so that the chaser light runs to the right. The pin number **n** is set to the new value. Then it is checked whether **n** has exceeded the range of possible values for the LED pins. If **n** has a value greater than 10, due to the change, the chaser light on the right starts again at pin 4. If the new value of **n is** less than 4, the chaser light on the left starts again at pin 10.

13. Tag

# Day 13

## Today in the Advent calendar

- 1x LCD module-
1x 560 Ohm resistor (green - blue - brown)




Yellow, green: control lines, violet: data lines, red: +5 V, blue: ground.

Today an LCD module is part of the Advent calendar. Such display modules are installed in many electronic devices. The LCD module has a 16-pin terminal strip with which it is inserted on the breadboard.

The LCD module in the Advent calendar works character-oriented. This means that you do not have to worry about controlling individual pixels. A separate controller on the LCD module processes characters that are sent as whole or half bytes on the data lines to the LCD module. Almost all such display modules, like the one contained in the Advent calendar, are compatible with the HD44780 quasi-standard. This refers to the Hitachi chip that controls such LCD modules.

**Components:** 1x Nano board, 2x breadboard, 1x LCD module, 1x 560 ohm resistor (green - blue - brown), 1x potentiometer, 7x connection cable, 7x wire bridge (different lengths)

The construction of the circuit may look a bit confusing at first glance. mBlock 3 provides this pin assignment. If you use the Arduino IDE you can change it freely. Use the potentiometer to adjust the contrast of the LCD module so that the font is easy to recognize.

### Pin Assignment of an HD44780 Compatible LCD Module

| Pin | Function | Description | Nano Pin |
|---|---|---|---|
| 1 | VSS | Ground line power supply 0 V | 0 V |
| 2 | VDD | Power supply +5 V | +5 V |
| 3 | V0 | Contrast adjustment, 0 V ... 5 V | Potentiometers |
| 4 | RS | Select tab | Pin D8 |
| 5 | RW | Read/Write, if nothing is read from the LCD module, connection to 0 V | 0 V |
| 6 | E | Enable (changeover signal) | Pin D9 |
| 7 | D0 | Data bit 0 (not required in 4-bit mode) | not required in 4-bit mode |
| 8 | D1 | Data bit 1 (not required in 4-bit mode) | not required in 4-bit mode |
| 9 | D2 | Data bit 2 (not required in 4-bit mode) | not required in 4-bit mode |
| 10 | D3 | Data bit 3 (not required in 4-bit mode) | not required in 4-bit mode |
| 11 | D4 | Data bit 4 | Pin D4 |
| 12 | D5 | Data bit 5 | Pin D5 |
| 13 | D6 | Data bit 6 | Pin D6 |
| 14 | D7 | Data bit 7 | Pin D7 |
| 15 | A | Backlight, 560 Ohm series resistor required | Series resistor 560 Ohm |
| 16 | K | Backlight ground line | 0 V |

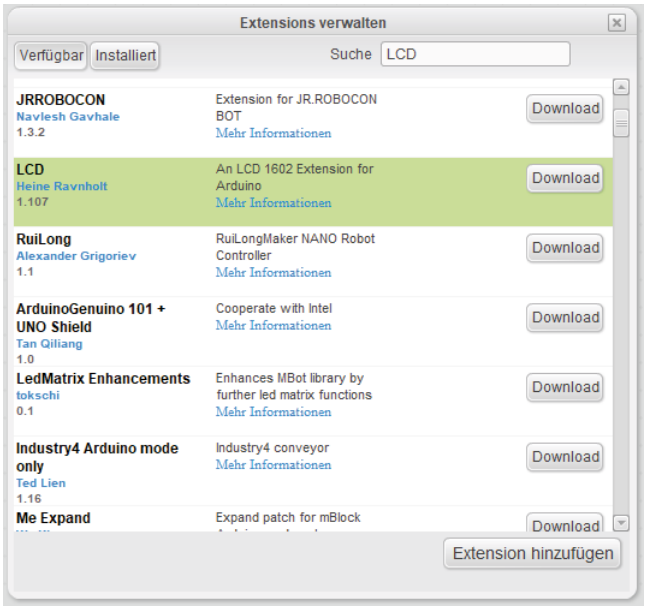
**Check the circuit carefully**
Before you supply power to the Nano and thus to the circuit, check all connecting wires again carefully. An incorrectly connected +5 V line can damage the LCD module. The maximum voltage at the contrast pin must be less than +5V. Never open the potentiometer completely.

The LCD module supports two modes for data transfer. In 8-bit mode, a complete character can be transmitted at once. However, the 4-bit mode is mostly used because it saves four ports on the sending device. Here the data of a character is transferred in two blocks one after the other.

## Controlling the LCD module with mBlock 3

The functional range of mBlock 3 can be extended by various so-called extensions. Among them there is also an extension for LCD modules.

Select **Extensions / Manage Extensions from** the menu. On the **Available** page, search for **LCD at** the top of the search box. Install the **LCD** extension **by** clicking on **Download**.



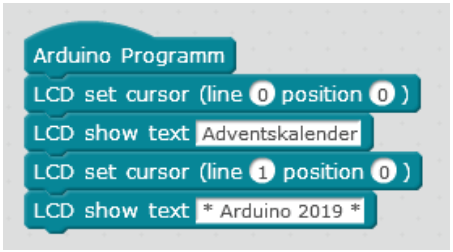

Download and installation of the LCD Extension for mBlock 3.

After installation, the new blocks of the LCD extension automatically appear on the **Robots** block palette.

**Important**
The LCD extension only works in Arduino mode, not in interactive mode of mBlock 3.

## The program

The `13mblock` program shows a text on the LCD module.




A simple program for a two-line text on the LCD module.

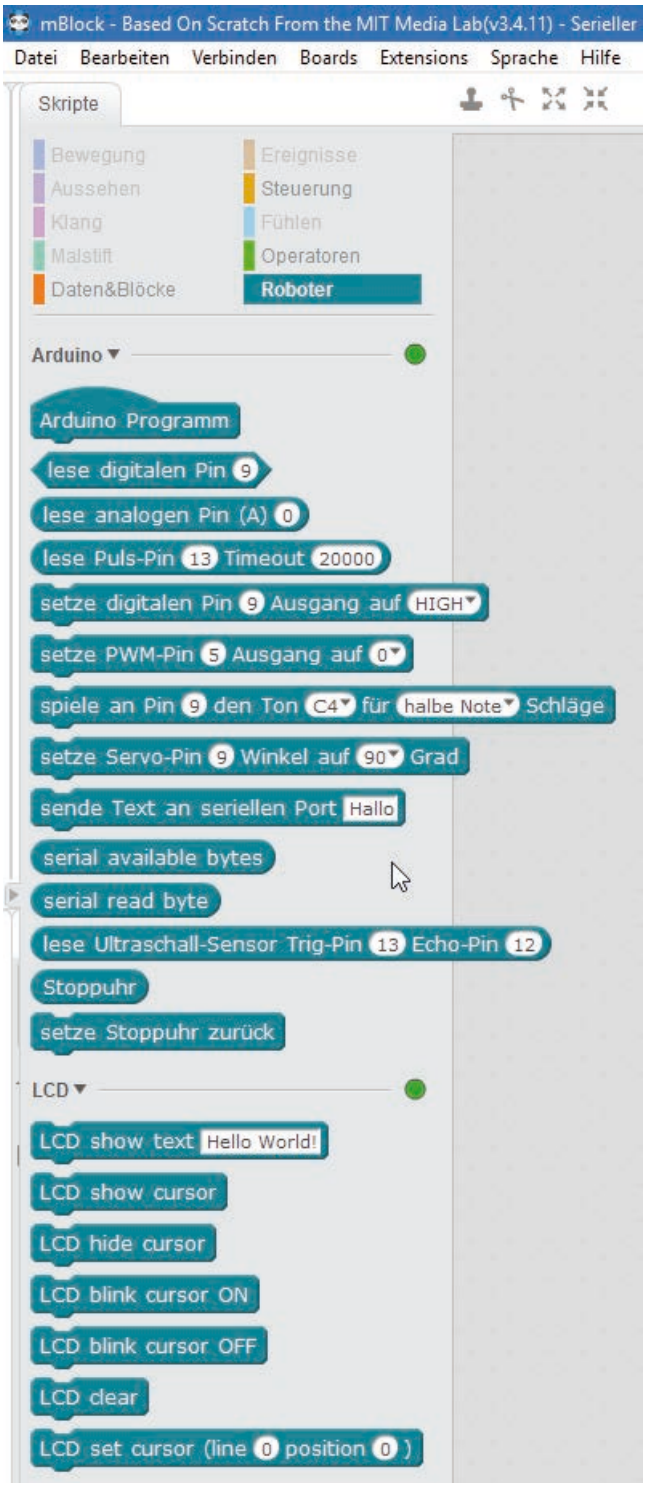
## How the program works

The program needs no loop and no variables. After it has run once, the text stops on the LCD module.

The "**LCD set cursor (line ... position ...)**" block from the LCD extension places the cursor on the line **(0** or **1)** specified in the first field and the column **(0-15)** specified in the second field, in this case on the beginning of the upper line 0.

The "**LCD show text ...**" **block** from the LCD extension displays a text on the LCD module. It starts at the current cursor position. The first block writes the words "`Advent calendar`" at the beginning of the upper line.

The "**LCD set cursor (line ... position ...)**" block from the LCD extension sets the cursor to the beginning of the lower line 1. Here the text `* Arduino 2019 *` is displayed.




The blocks of the LCD extension.

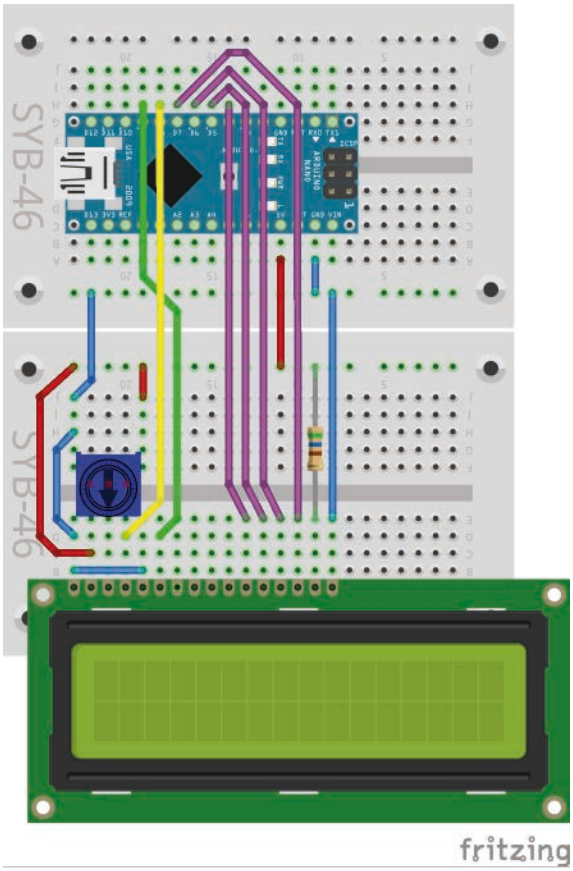
14. Tag

## Day 14

### Today in the Advent calendar

- 2x connecting cable

The cables are not used until a later day. The circuit design is unchanged from the 13th day.



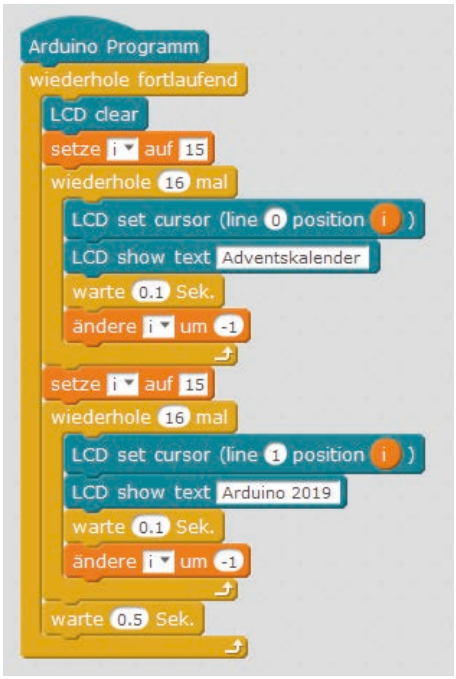

The unchanged circuit design.

### Moving text on the LCD module

If you let a text run into the LCD module as a ticker, it is much more noticeable than just a static text.

### The program

The `14mblock` program shows a text that runs animated from the right into the LCD module.




Extended program for animated two-line text on the LCD module.

### How the program works

At the beginning of each cycle of the infinite loop, an "**LCD clear**" block from the LCD extension deletes the display on the LCD module. After that the display is empty and the new text can run in.

The variable **i** indicates the position at which the text is to begin. It is set to 15 at the beginning and thus designates the last character of a line.

Now a loop runs through 16 times. In each iteration, the LCD cursor is moved one character to the left and then the entire text line is displayed at this position. Letters that no longer fit into the display are simply truncated. The text to be displayed still has a space at the end, which overwrites the last character from the previous pass, so that no letters remain in the display.

After the first text in the upper line, a second text in the lower line is displayed in the same way.

---

15. Tag

## Day 15

### Today in the Advent calendar

- 1x battery box

### Battery box

The Nano Board can also be used without a PC and execute a stored program. This requires an external power supply. This can be a USB mobile phone charger, a power bank or a battery. Today, the Advent

calendar contains a battery box that delivers a voltage of 6 V with four AAA batteries and 4.8 V with rechargeable batteries, which is sufficient to power the Nano board. The batteries are not included.

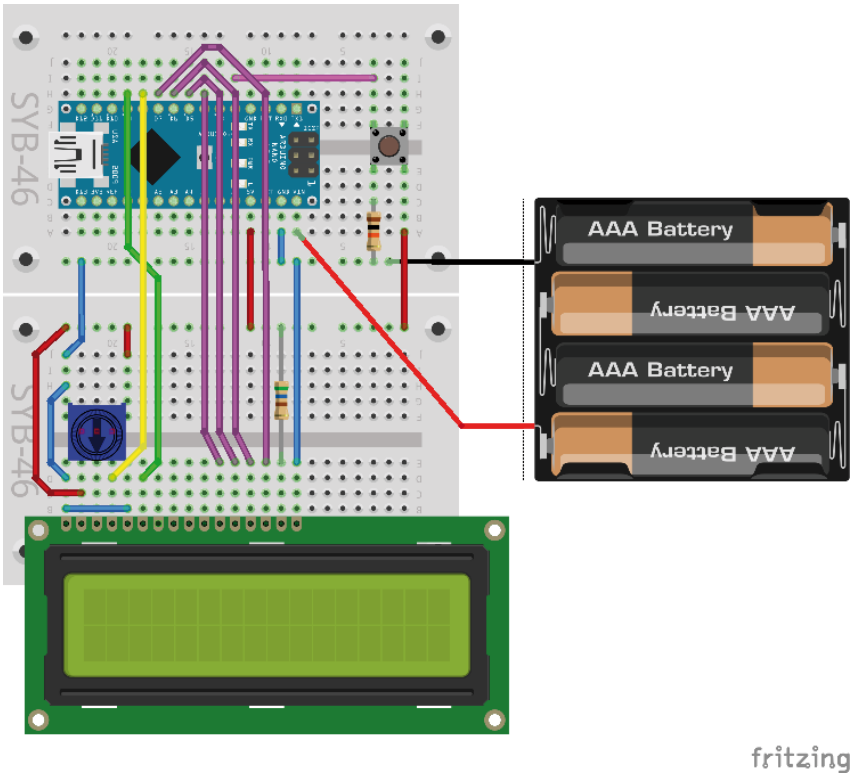
Do not connect the battery box yet, as you are still supplying power to the Nano via the PC until the new program has been transferred. The battery box is connected to the pins **VIN** and **GND of** the Nano board after disconnecting the USB cable.

### Stopwatch

Arduino compatible boards do not have a real time clock, but a timer that runs continuously and can therefore be used as a stopwatch.

The circuit uses a single button to start, stop and reset the stopwatch. The bottom line of the LCD module shows which function is triggered by the next keystroke.

**Components:** 1x Nano Board, 2x breadboard, 1x LCD module, 1x 560-Ohm resistor (green-blue-brown), 1x 10-kOhm resistor (brown-black-orange), 1x push-button, 1x potentiometer, 1x battery box, 9x connection cable, 7x wire bridge (different lengths)




Stopwatch with LCD module and battery box.

### The program

The `15mblock` program runs a stopwatch on the LCD module. Transfer the program to the Nano in Arduino mode.

### How the program works

The program consists of a large endless loop in which the stopwatch and the display on the LCD module are reset at the beginning. After the user presses the button, the stopwatch runs until the user presses it again. If the key is pressed a third time, the infinite loop starts anew.

The "**reset the stopwatch**" block from the Arduino extension resets the internal stopwatch on the Nano. None of it is directly visible yet.

The "**LCD clear**" block from the LCD extension deletes the LCD module and sets the cursor position for the next text in the upper left corner.

The "**LCD show text ...**" **block** from the LCD extension displays a text on the LCD module. This starts at the current cursor position. The first block writes the word `time:` at the beginning of the upper line.


The program for the stopwatch.


This is what the display looks like when you start the program.

The "**LCD set cursor (line ... position ...)**" block from the LCD extension sets the cursor to the line (**0** or **1**) specified in the first field and the column (**0-15**) specified in the second field, in this case to the beginning of the lower line. Here the word `key:` is written.

The text `0.00` is displayed at position 7 in the upper line. Later here, when the stopwatch is running, the time appears. The word `Start` appears at the same position in the bottom line. The next keystroke is always displayed here.

Now the program waits until the user presses the key.

The "**wait until ...**" **block** from the **Control** block palette waits until the logic value in the field is true. At this point the value of digital pin **D2 to** which the push-button is connected is read out. The program waits until the button is released to prevent misinterpretations if the button is pressed for a longer period of time. Only then should the stopwatch start. With the "**not ...**" **block** from the block palette **operators** a logic value can be inverted. So the second "wait **until ...**" Block waits until the button is released again.

Then the word `Stop` appears in the bottom line instead of the word `start` displayed there, as the next keystroke stops the stopwatch.

Now a loop starts in which the stopwatch runs until the button at pin **D2** is pressed again. In each pass, the value of the **stopwatch** block from the Arduino extension is displayed at position 7 in the upper line of the LCD module. Like a variable, this block can also be used for calculations.

If the user presses the key, the loop is terminated. Afterwards the key is waited again until it is released. The word `Reset` is then displayed in the bottom line. The next key press should reset the stopwatch.

If the user now presses the key again, the current cycle of the main loop is terminated. The loop jumps back to the beginning, where the stopwatch is reset.

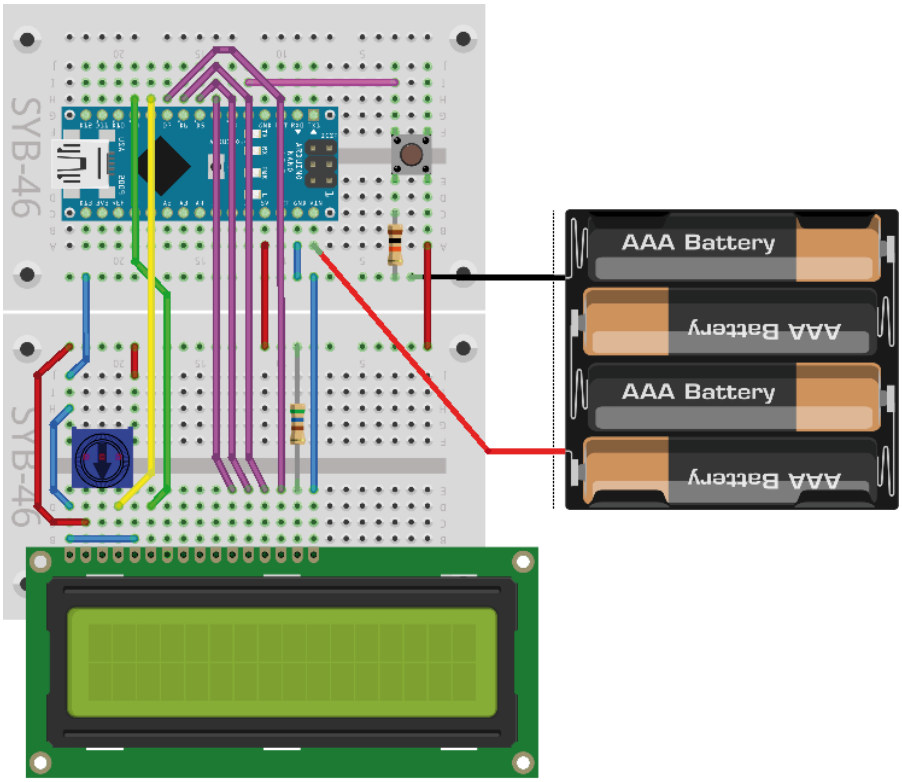
---

16. Tag

## Day 16

### Today in the Advent calendar

- 1x connecting cable




The unchanged circuit design.

The cable will not be used until a later day. The circuit design is unchanged from the 15th day.

### Christmas message with typewriter effect

A Christmas message appears on the LCD module. It does not light up all at once, but is typed letter for letter, just like a typewriter.

### The program

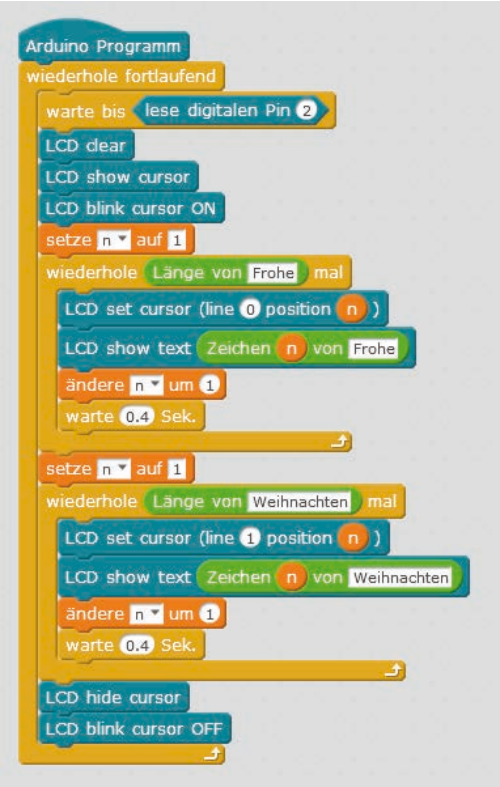
The `16mblock` program shows the running text on the LCD module. Transfer the program to the Nano in Arduino mode.

### How the program works

An endless loop first waits until the button on pin **D2** is pressed. Then the display on the LCD module will be cleared, the cursor will be displayed and set to flashing. There are separate blocks for this. You do not need to program the cursor flashing yourself.

The variable **n** is set to 1. It specifies the cursor position when writing. Another loop writes up to the character length of the word `happy`, the individual letters one after the other into the upper line of the LCD module. For this purpose a "**character ... from**" **block is used** which picks individual characters from a string.

The word `Christmas` is then written in the same way in the bottom line of the LCD module. The cursor is then hidden again. The text stops until the user presses the key again.




The program for the Christmas message.

---

## Day 17

17. Tag

### Today in the Advent calendar

- 1x connecting cable

The cable will not be used until a later day.

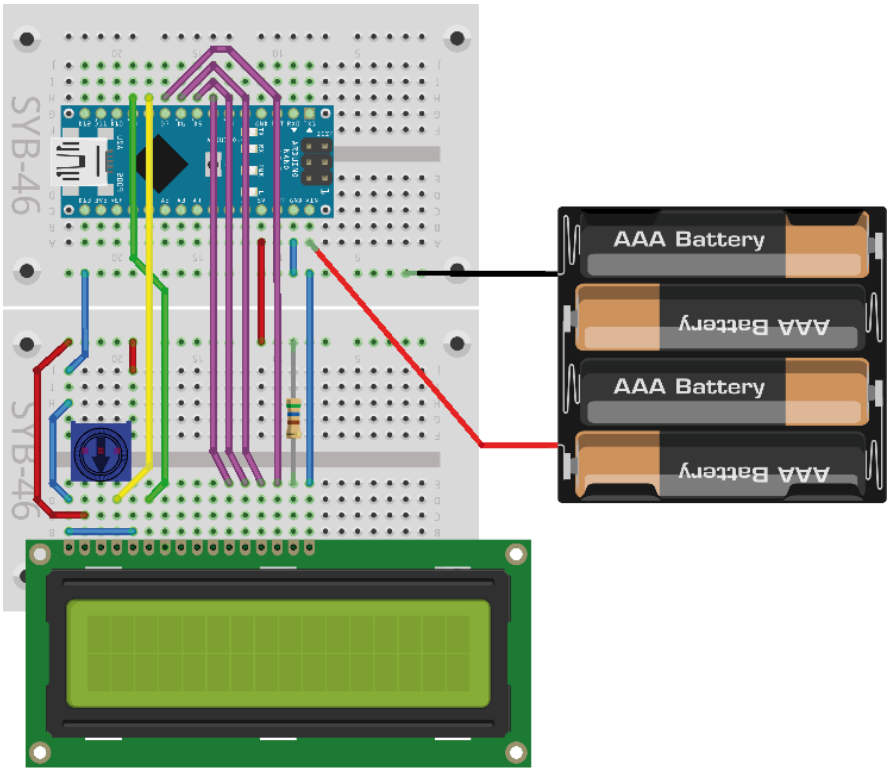
### Reading analogue inputs

The experiment of the 17th day shows the values of four analogue inputs. As long as no defined signal is present at an analogue input of the Nano, these inputs deliver relatively random values, which are caused by potentials on the board. If you connect a remaining connection cable or wire to an unconnected input, you can even use it to collect the electrosmog from the environment, as shown by the measured value of the input.

**Components:** 1x Nano board, 2x breadboard, 1x LCD module, 1x 560 ohm resistor (green-blue-brown), 1x potentiometer, 1x battery box, 7x connection cable, 7x wire bridge (different lengths)

### The program

The `17mblock` program displays the states of four analogue inputs on the LCD module. Transfer the program to the Nano in Arduino mode.




This circuit displays the values of four analogue inputs on the LCD module.


The program reads four analogue inputs and displays their values on the LCD module.

### How the program works

At the beginning the pin numbers of the used analogue inputs are displayed on the LCD module as legend of the values. An endless loop then reads out four analogue inputs one after the other and writes the values to the corresponding positions after the displayed pin numbers. After 0.5 seconds the loop runs again.

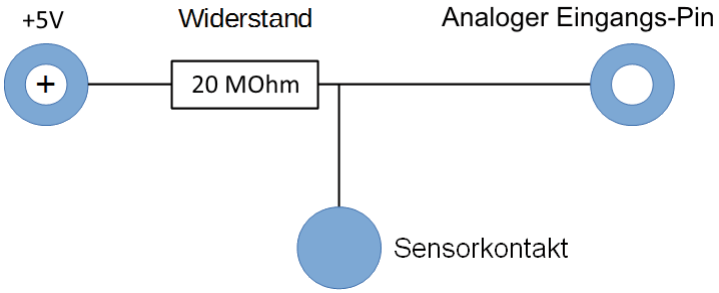
---

**18. Tag**

## Day 18

### Today in the Advent calendar

- 1x plasticine -
- 1x 20 MOhm resistor (red-black-blue)




Circuit diagram for a sensor contact.

### Sensor contact made of plasticine

Today traffic lights, door openers, light switches and automats are often controlled with sensor contacts that you only have to touch. Push-buttons that really need to be pressed are becoming increasingly rare.

The digital pin switched as input is connected to +5 V via an extremely high-impedance resistor (20 MOhm), so that a weak but clearly defined signal is present. A person who is not floating freely in the air is always grounded and delivers a low level via the electrically conductive skin. If this person touches a sensor contact, the weak high signal is superimposed by the significantly stronger low level of the fingertip and pulls the pin to low level.

How high the resistance between hand and mass really is depends however on many things, including shoes and flooring. Barefoot on wet grass is the best connection to the earth's mass, stone floors also usually works well. Wooden floors insulate more strongly, plastic floor coverings are often even positively charged. To ensure that the circuit always works, an additional ground contact is provided for each circuit, similar to sensor keys on lifts and doors. If this and the actual sensor are touched at the same time, the ground connection is necessarily established.

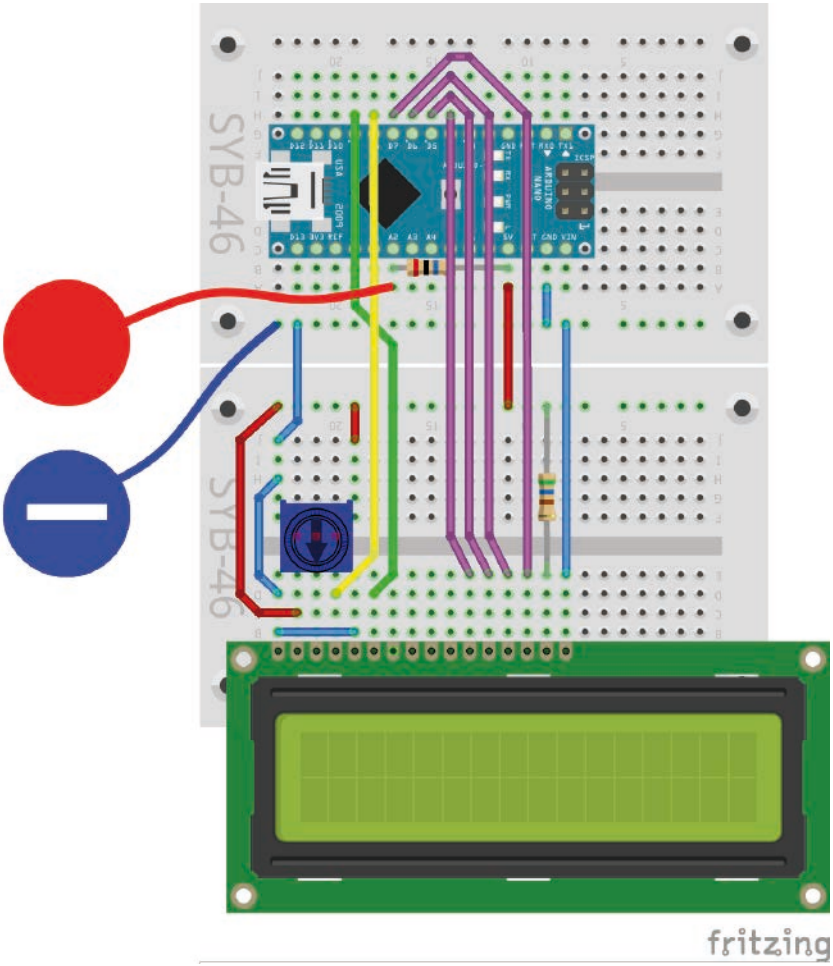
Plasticine conducts electricity about as well as human skin. It can easily be brought into any shape, and a plasticine contact is much easier to handle than a simple piece of wire. The surface with which the hand touches the contact is much larger. So it does not come so easily to a "loose contact". Plug one end

of a connecting cable into a piece of plasticine and the other end into the breadboard as shown in the illustration.

Since mBlock always switches on the pulldown resistors built into most Arduino-compatible boards, digital inputs are automatically pulled to 0 V and have a low level even without touching them. Arduino-compatible boards have additional analogue inputs, which are very suitable for sensor contacts. Analog inputs provide values between 0 (low level) and 1023 (high level). Depending on the board type, values around 200 are good limit values to distinguish between touched and non-touched sensor contact.

The experiment of the 18th day toggles a text on the LCD module via a simple sensor contact.

**Components:** 1x Nano board, 2x breadboard, 1x LCD module, 1x 560 ohm resistor (green-blue-brown), 1x 20 ohm resistor (red-black-blue), 1x potentiometer, 2x plasticine, 10x connection cable, 6x wire bridge (different lengths)




Sensor contact, and ground contact made of plasticine.

## The program

The `18mblock` program shows the state of the sensor contact on the LCD module.


The `18mblock`. program reads the value of an analogue input.

## How the program works

The analogue pin A2 is read out in each pass of the endless loop. As long as the sensor contact is not touched, it has a very high value because it is connected to +5 V via the 20 MOhm resistor.

If you touch the sensor contact and preferably the ground contact simultaneously, the value of the analogue input pin drops significantly to almost 0. The program checks whether the value falls below the 200 limit. In this case, the word `on` is displayed on the bottom line of the LCD mode. As long as the input value is higher than 200, the word `out` is displayed the other way around. For both character strings to be displayed, four blanks are appended at the back and four at the front. The word `one` appears further to the left than the word `out`. This effect makes the switching even more noticeable than if both words were displayed at the same position on the LCD module.

19. Tag

# Day 19

## Today in the Advent calendar

- 1x button -
1x 10 kOhm resistor (brown - black - orange)

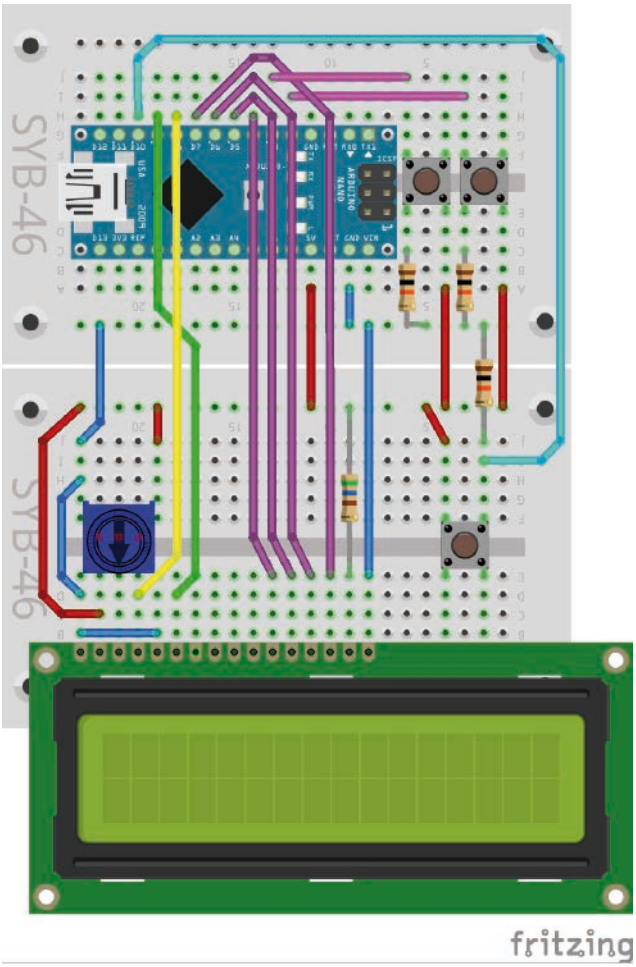

The number guessing game.

## Guess numbers

The project of the 19th day is a simple guessing game in which a player has to guess a number randomly chosen by the Nano in as few steps as possible.

The numbers are displayed on the LCD module and entered via buttons. The lower line of the display shows symbols to explain the buttons:

**Components:** 1x Nano Board, 2x breadboard, 1x LCD module, 1x 560-Ohm resistor (green-blue-brown), 3x 10-kOhm resistor (brown-black-orange), 3x push buttons, 1x potentiometer, 9x connection cable, 10x wire bridge (different lengths)




Guessing game with LCD module and buttons.

## The program

The `19mblock` program generates a secret number smaller than 100. With the two upper buttons you set a tip and enter it with the lower button. The program displays on the LCD module whether the secret number is smaller or larger than the last tip. If the number was typed correctly, the number of attempts is displayed.

## How the program works

The static text elements of the game display are displayed in the upper line of the LCD module. Instead of the three question marks, the game later shows the last tip. The bottom line indicates whether the solution number is greater or less than this tip.

The "**random number from ... to ...**" block generates random values on the PC in interactive mode, but in Arduino mode the randomizer is not initialized correctly, as the same sequence of random numbers is generated every time the program is started. Therefore, we use our own procedure to determine a random number.

Two unconnected analogue pins are read out. The "**... mod ...**" block determines the indivisible remainder of a division that is even more unpredictable than the value of the analogue input itself. The secret number is formed from the two digits thus determined and stored in the variable **x.**


Calculation of the number to be guessed.

The **tip** is pre-set to 50, exactly in the middle of the possible value range of the random number. The counter **n** for the number of attempts starts with 0. The variable **end** is set to 0. If the number is correctly guessed, it is set to 1, ending the infinite loop. This uses a block of type "**repeat until ....**". The loop is repeated until the number is guessed and the end of the game is reached.

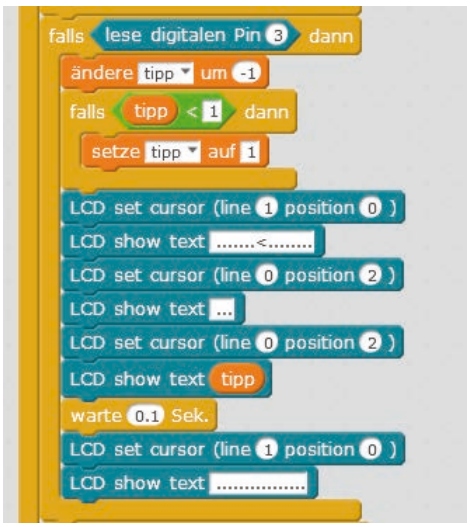
As long as the lower button on pin D10 is not pressed, the user can use the other buttons to set the next typed number. For this purpose, the buttons are constantly interrogated and evaluated one after the other. The user can set the desired number by quickly pressing the buttons or keep a button pressed for a longer time and thus quickly count up or down.

If the button at pin **D2** was pressed, the typed number is increased by 1. If the number becomes larger than 100, the current tip is automatically set to 100.

In the bottom line of the LCD module, an arrow to the right indicates that the number has been increased.

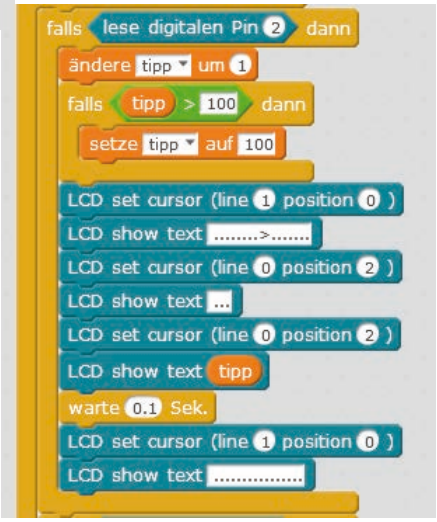
In the upper left line, the last selected number is overwritten with three dots so that no digit remains if the newly selected number has one digit less. Overwriting the area with spaces does not work because mBlock 3 does not correctly evaluate a string of characters consisting only of spaces in Arduino mode. The new number will then be displayed at this point.

After a waiting time of 0.1 seconds, the arrow symbol disappears from the bottom line of the LCD module. Now the next keystroke can be evaluated.

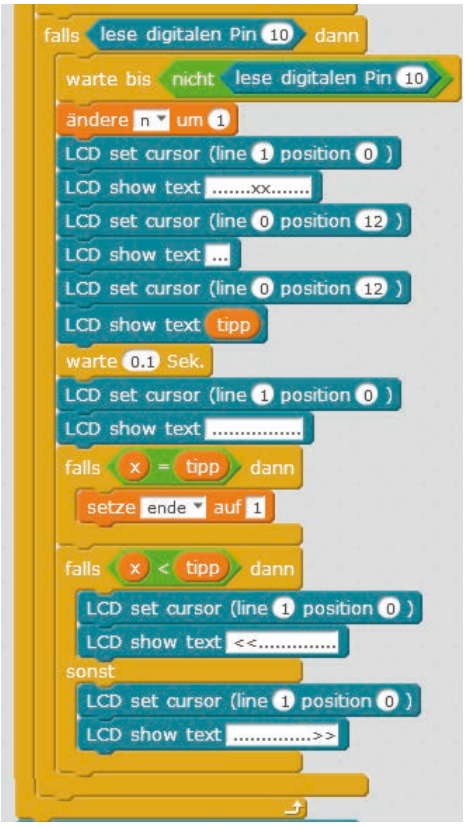



The left button on pin D3 reduces the number by 1.

If the button at pin **D3** was pressed, the typed number is reduced by 1. If the number becomes smaller than 1, the current tip is automatically set to 1. The other blocks are also comparable to those for the first button.




The right button on pin D2 increases the number by 1.




The lower button at pin D10 gives the tip.

If the button on pin **D10** was pressed, the set number is given as a tip. The number of bets placed in variable **n** is increased by 1 and the number is displayed as a new bet in the top right corner of the LCD module.

If the searched number **x** is equal to the given tip, the variable **end is** set to 1. After this run, the loop is not repeated again. The game is over.

If the number **x you are** looking for is smaller than the tip you have entered, an arrow to the left is displayed at the bottom left; if it is larger, an arrow to the right is displayed at the bottom right. This tells the player in which direction to continue betting.

When the game is over, the number of attempts required to guess the number appears in the bottom line of the LCD module.

---

**20. Tag**

## Day 20

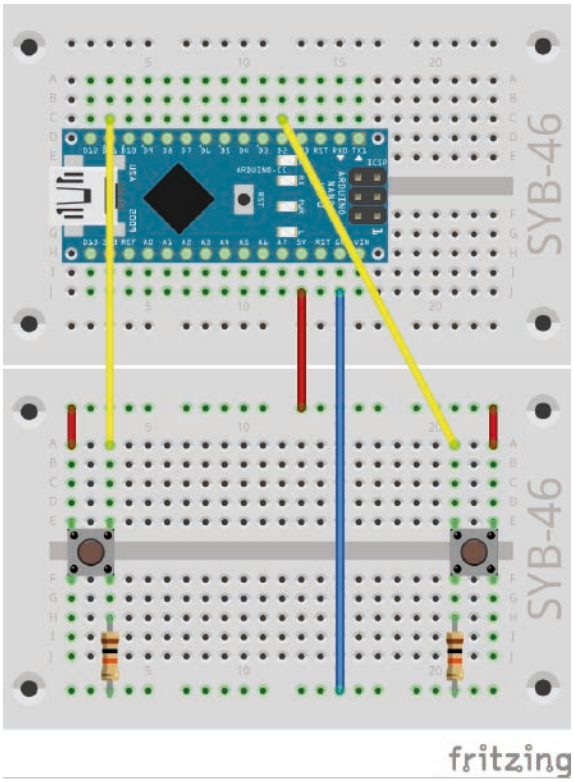
### Today in the Advent calendar

- 1x connecting cable

The cable will not be used until a later day.

### Controlling the Pong game on the PC with keys

Games and animations in Scratch are usually controlled by mouse and keyboard, since Scratch only supports these input methods by default. mBlock 3 makes it possible to use keys and other input devices via Arduino to control games in Scratch. The next project shows how to replace classic controls with Arduino keys in an existing scratch game.




Two buttons control the Pong game.

**Components:** 1x Nano Board, 2x breadboard, 2x 10-kOhm resistor (brown-black-orange), 2x button, 4x connection cable, 2x wire bridge

The buttons are arranged in such a way that they can be conveniently operated with two fingers.

### The program

The two keys are intended to move the panel of a classic Pong game in both directions. The original scratch does provide such a game with its examples, but mBlock 3 unfortunately doesn't. You can find the program as `20pong01mblock` in the download archive.

**Interactive mode**
The game runs in the interactive mode of mBlock 3. Switch mBlock 3 and the Nano to the interactive mode as described on day 10.

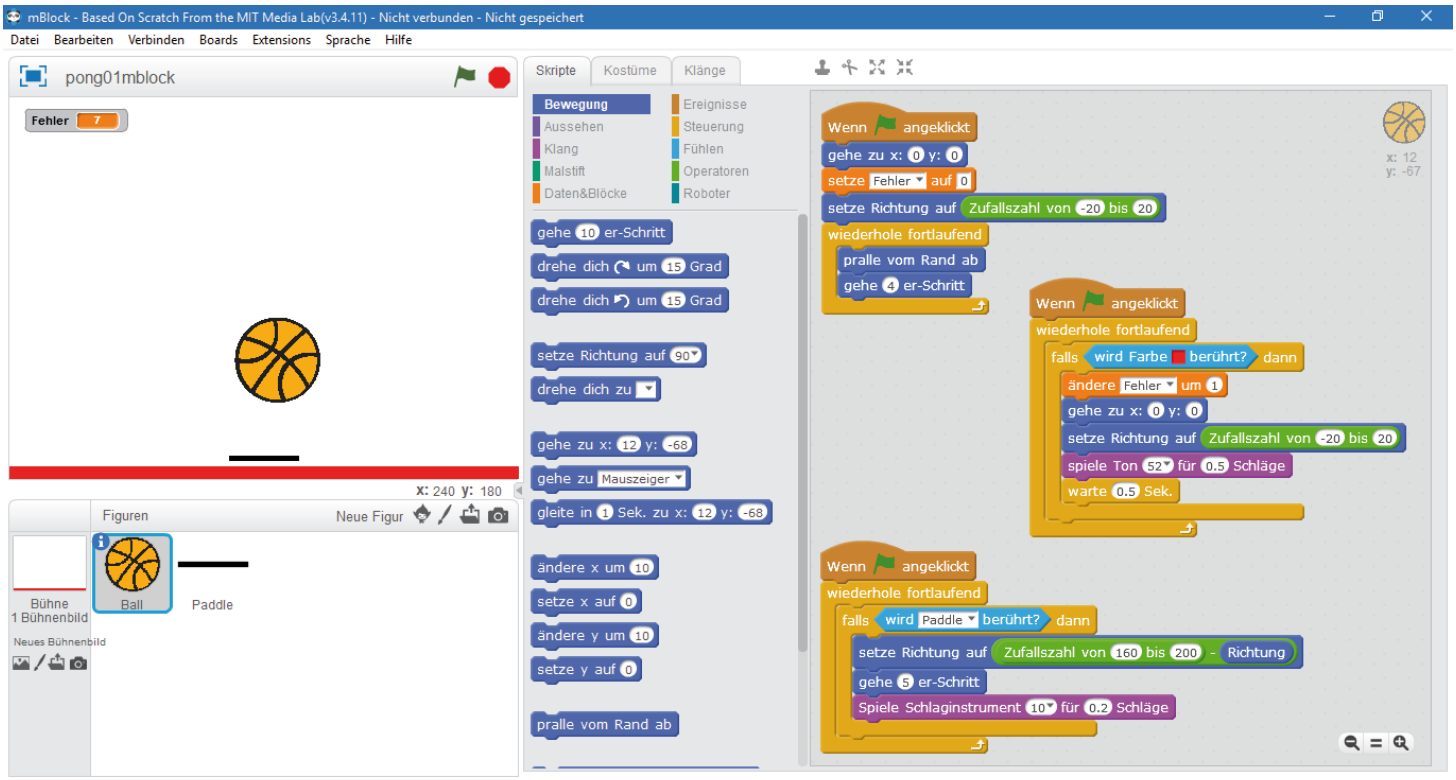
### How the game works

Before we start to change the control of the game, first a few basics about how the game works: This classic game, which was already played in the 70s on early computer systems, is about keeping a ball in the air with a simple bat for as long as possible. The so-called paddle can only be moved horizontally. When the ball comes, the paddle must be in the right place.

The game consists of two objects, ball and paddle, each of which - as usual in Scratch - uses its own program blocks that run independently of each other.

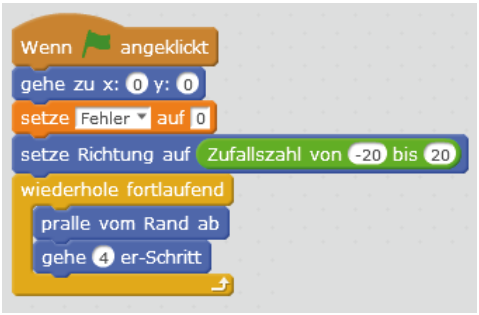
### The Ball

The ball is controlled by three scripts that all run simultaneously and are started when the user clicks on the green flag.




The classic Pong with the script for the ball still without Nano.

The normal ball movement is controlled by the script shown above. It first brings the ball to its starting position, sets the error counter to 0 and sets a random direction for the first move. Scratch provides its own block that ensures that an object that touches the edge of the stage bounces off and flies on in the right direction. If the object is not at the edge of the stage, this block is ignored and a simple movement step of length 4 is performed in the pre-set direction.




The blocks for normal ball movement.

The second script block waits until the ball touches the red colour at the bottom of the stage. In this case it has fallen off and the player gets one mistake point. The ball is then returned to the starting position, a new random direction of movement is determined, and a short sound signal is played.


The blocks in case the ball touches the red line.

The third script block checks in an endless loop whether the ball touches the paddle. In this case, the direction of movement is randomly changed so that the ball moves away from the panel and is rotated at a random angle so that it continues to fly in an unpredictable direction. Otherwise it could happen that the ball flies a perpetual square course by bouncing off the three sides of the stage and the panel. At the end a short sound is played.

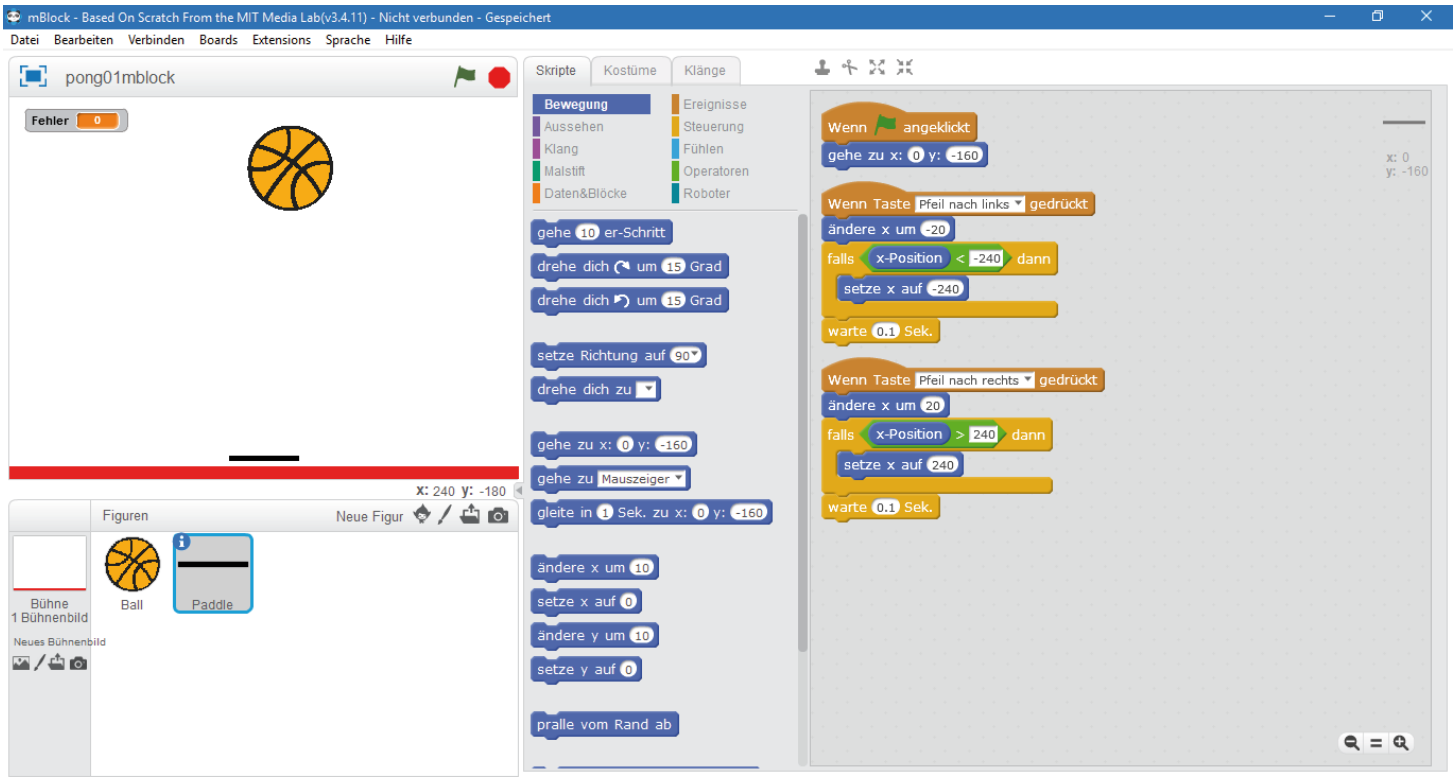

The blocks in case the ball is hit back by the paddle.

## The Paddle

After the green flag has been clicked, the paddle is set to its start position. Two further script blocks query the keys [left arrow] and [right arrow]. If the user presses one of these keys, the paddle is moved by 20 units in the

corresponding direction. The coordinates are used to check whether it would run out over the left or right edges, and in this case it is set directly to the edge.

Click the paddle at the bottom left of the object palette to view and edit the script blocks for the paddle.



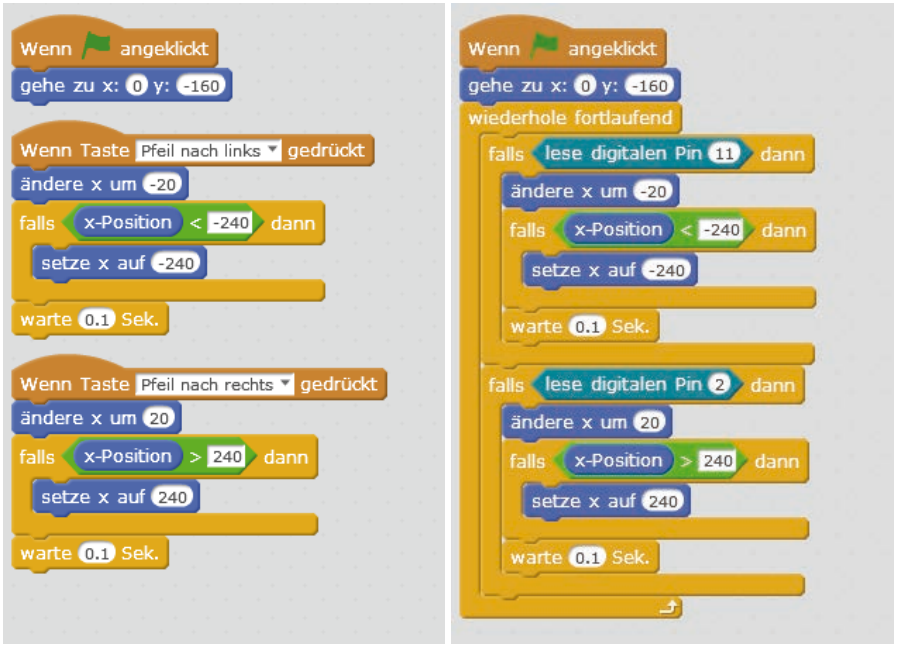

The script for the paddle - still without Arduino.

You can play the game as it is. Use the arrow keys on the keyboard to control it. The Nano is not needed.

### Controlling Pong with buttons on Nano

In the next version the paddle will be controlled by the two buttons on the breadboard, which are connected to pins **D11** and **D2** on the Nano.

mBlock 3 does not offer on the "events" block palette any comparable block to "**If key ... pressed**" for the case of which a high signal is applied to a digital pin. The program cannot directly evaluate the pressing of one of the buttons as an event. So we rebuild the program so that an endless loop alternately requests and evaluates the two buttons.




Program variants for keyboard keys (left) and Arduino keys (right).

Pull two "**if... then**" **blocks** into the loop. Each of these blocks inquires through a "**read digital pin ...** " block from the **robot** block pallet for a digital pin on the Nano. If the respective pin delivers a high signal, the button is pressed. In this case, the same blocks for moving the paddle are processed as in the previous program version when an arrow key is pressed on the keyboard.

Now click on the green flag to start the game. You can now control the paddle with the two buttons. You can find the finished program as `20pong02mblock` in the downloads.

---

## Day 21

21. Tag

### Today in the Advent calendar

- 1x 20 MOhm resistor (red-black-blue)

### Controlling tennis with sensor contacts

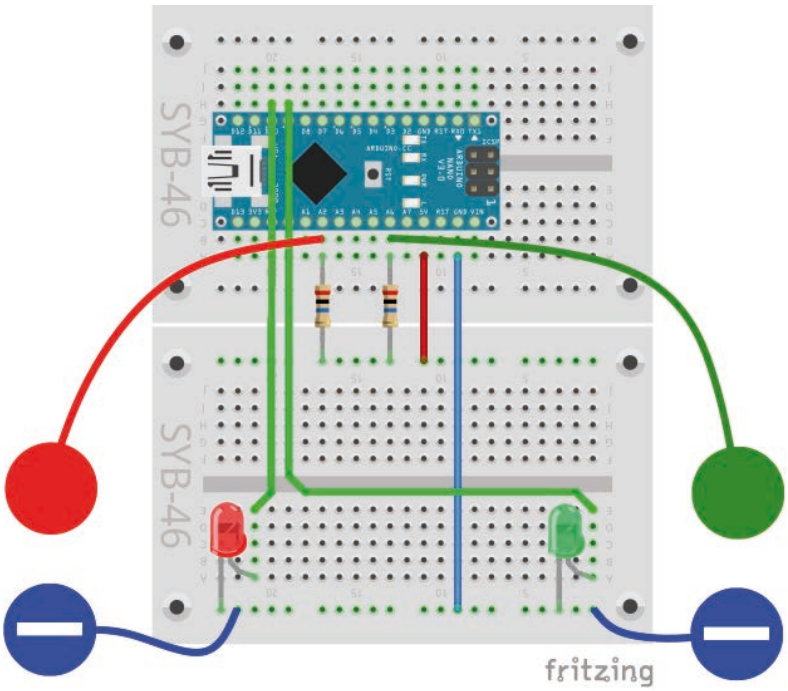
Today's game is a tennis game for two players, as we know it from 70s video games. Each player gets two sensor contacts in own hand, one of which is a ground contact, plus an LED that indicates the touch of the sensor.

**Components:** 1x Nano board, 2x breadboard, 1x LCD module, 1x 560 ohm resistor (green-blue-brown), 1x 20 ohm resistor (red-black-blue), 1x potentiometer, 2x plasticine, 10x connection cable, 6x wire bridge (different lengths)

### The program

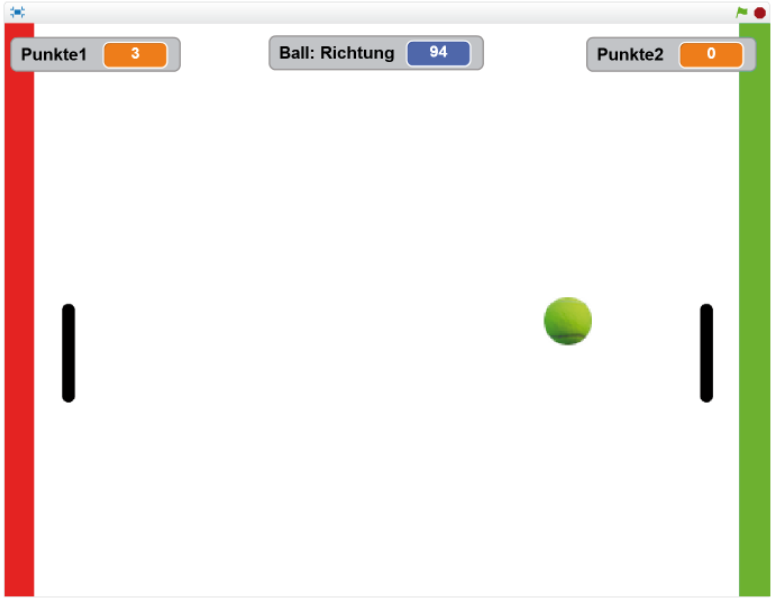
To start playing immediately, open in mBlock 3 the program `21mblock` from the downloads. Of course, you can also create the game yourself using the pictures to follow each step.

**Interactive mode**
The game runs in the interactive mode of mBlock 3. Switch mBlock 3 and the Nano to the interactive mode as described on the 10th day.




Sensor contact, and ground contact made of plasticine.

A click on the blue rectangle symbol at the top left of the program name switches the stage to full screen mode so that you can play on the whole screen.




The tennis game in full screen mode.

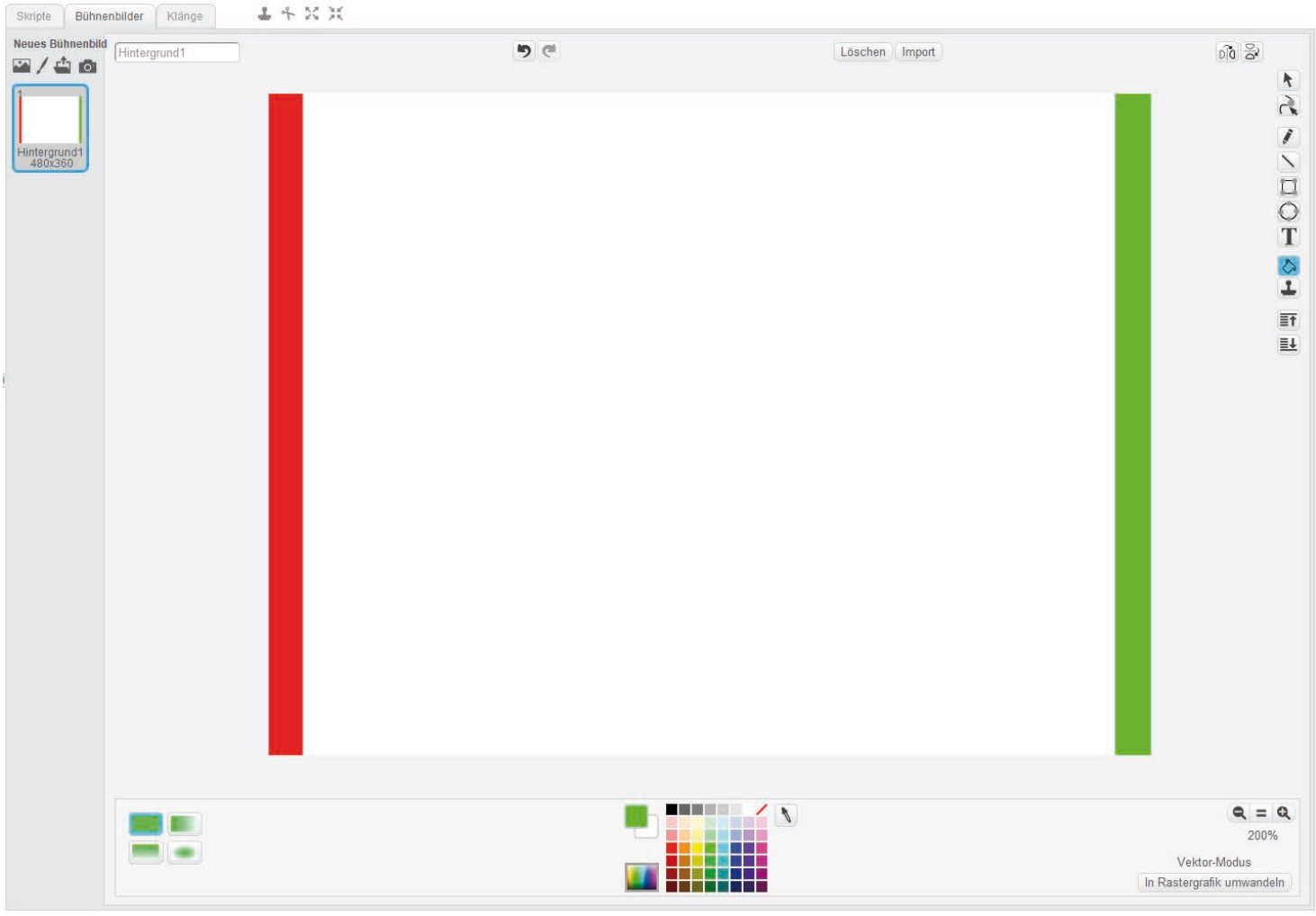
Click on the green flag, the game starts and the ball flies through the room. Every player can move his paddle upwards by tapping the sensor keys. If the sensor contact is not touched for a while, the paddle slowly falls down again. With some skill the players must try to hit the ball back again and again with their paddle so that it does not fly against the red or green wall.

If the ball touches your own wall because you didn't put the paddle in its way in time, you get a minus point. The ball then starts again at the starting position. Click on the red stop icon to stop the game at any time. Whoever has fewer minus points wins.

### How the program works

There are three objects in the game: the ball, which controls the actual game and also gives the minus points when it hits the coloured edges and the paddles for the two players left and right. The paddles also evaluate the contact sensors and switch the LEDs.

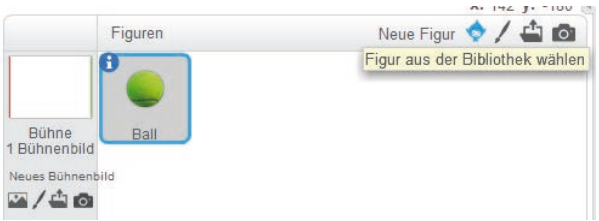
The two coloured stripes at the edges are painted directly onto the stage. If you click on them in the object palette on the left, you can edit the background with the painting program integrated in mBlock 3 on the **Stage settings** tab (at the top).



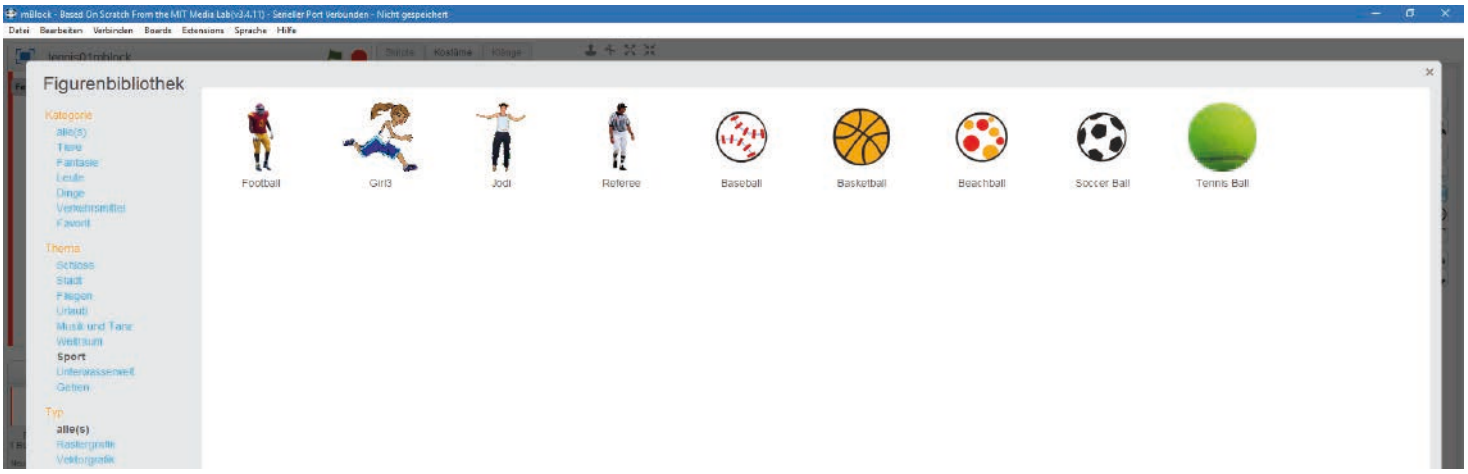

The background for the tennis game in the painting program of mBlock 3.

### The Ball

The ball is a tennis ball from the figure library that comes with mBlock 3. On the Object Palette, click the **Select Figure from Library** icon. Here you can find the **tennis ball at the** themes under **sport**.



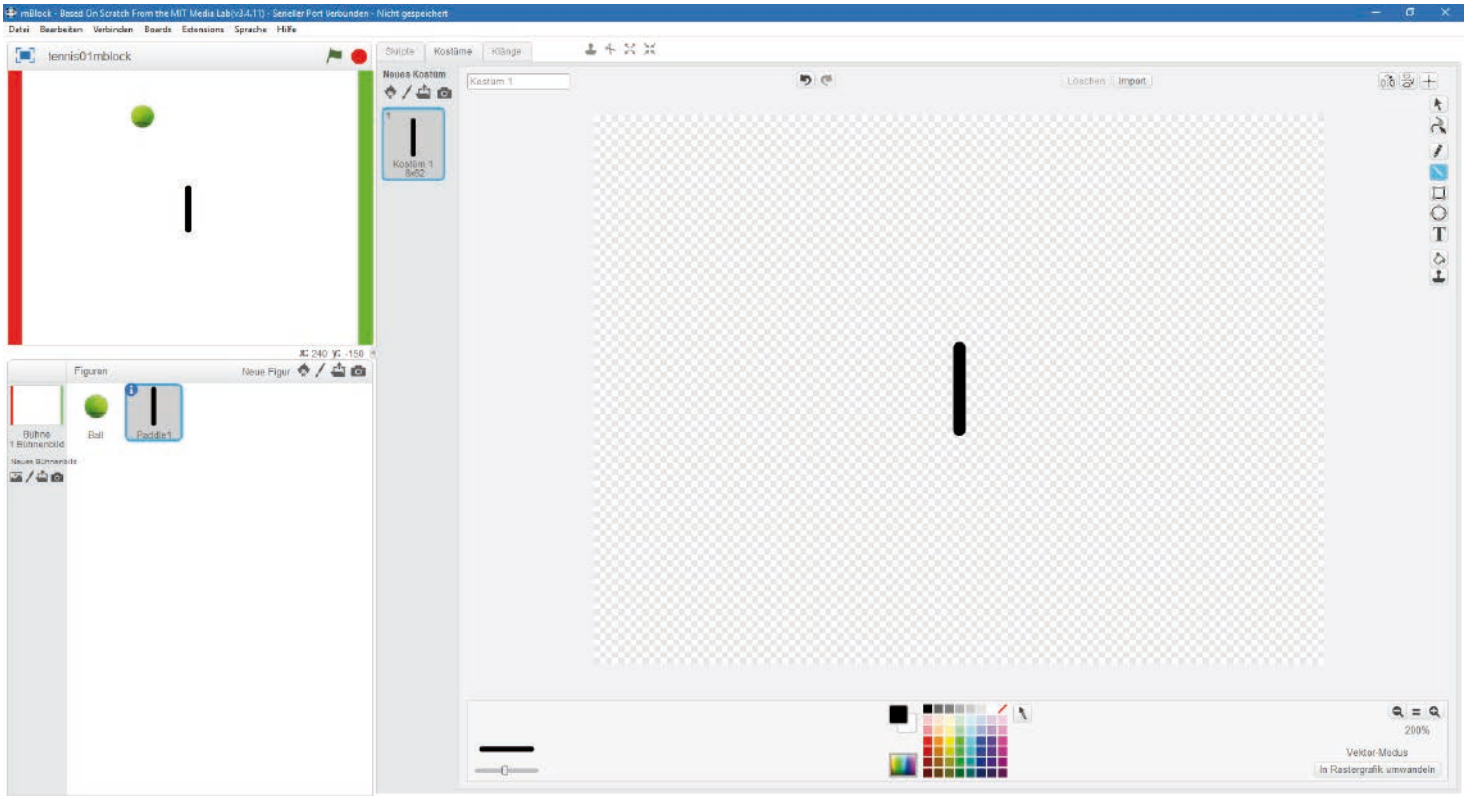

Selecting a new figure from the library on the Object palette.




The sport theme in the figure library.

## The Paddles

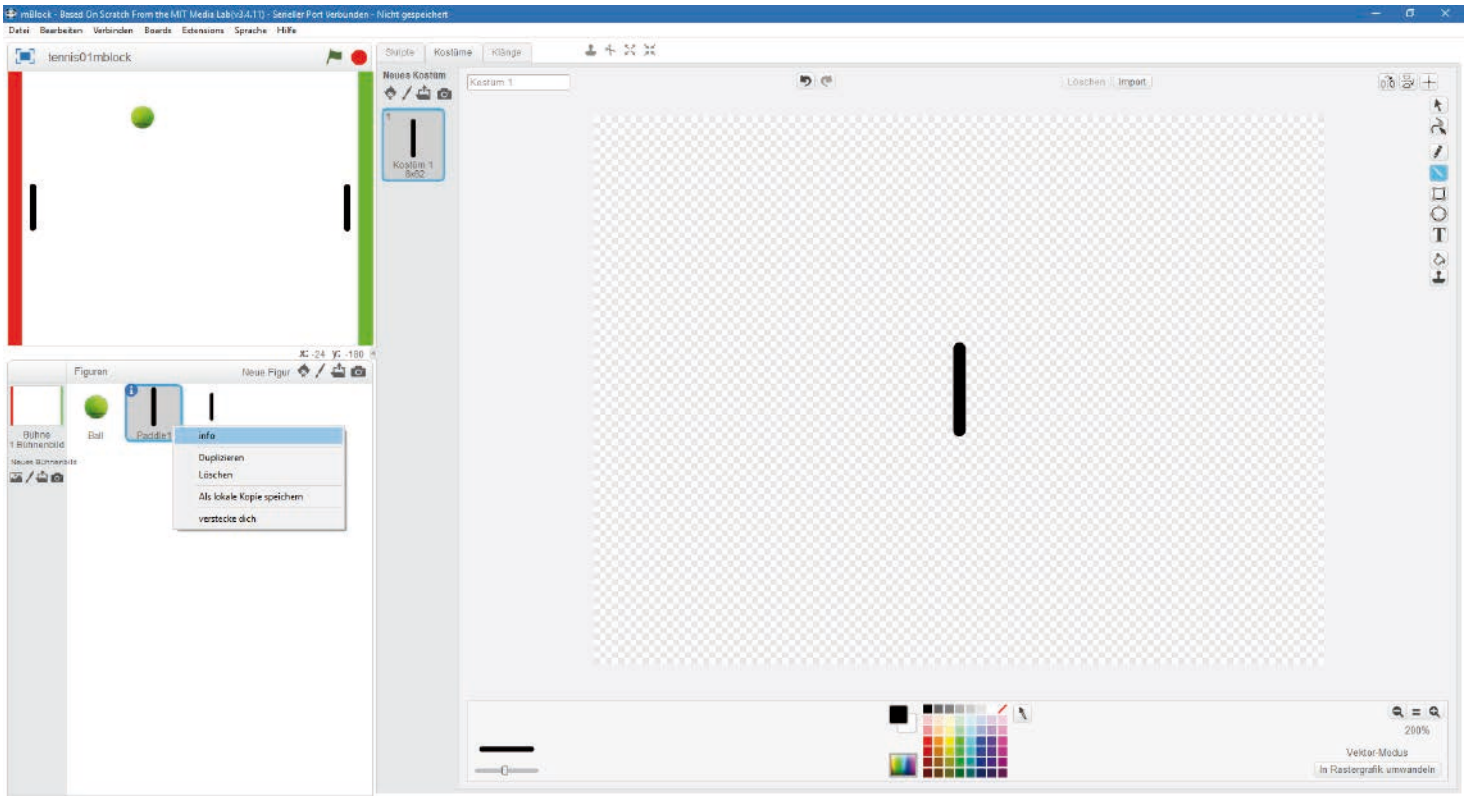
Click the **Draw New Figure** icon on the Object palette to draw the first paddle. In the drawing program, use the **Line** tool, select the black colour and set a mean line width at the bottom left. Draw the line vertically in the middle over the small cross in the middle so that the centre of the paddle is at the coordinate zero point. When the [Shift] key is pressed, the line becomes exactly vertical.



The first paddle.

Then click on the blue info symbol for the new figure on the object palette and give the paddle the name **Paddle1**.

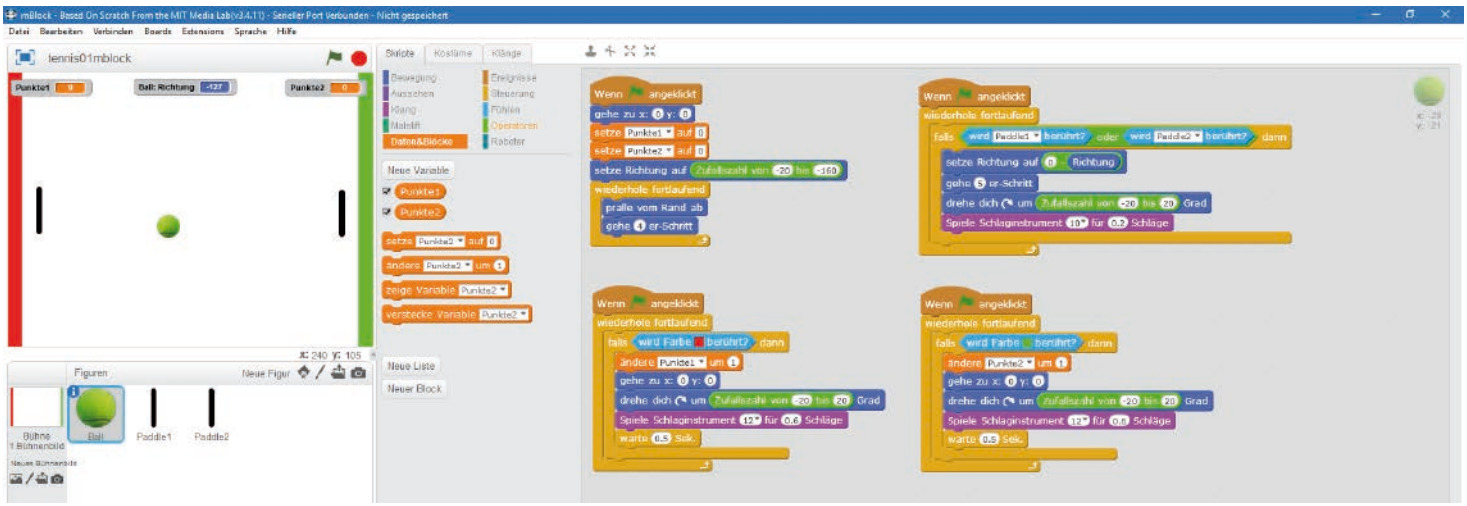
Right-click the **Paddle1** on the Object Palette and select **Duplicate from** the menu. Then slide the **paddle1** to the middle just before the red edge on the left and the **paddle2** to the middle just before the green edge on the right. The exact positions are set later by the program.




The two paddles.

## Program for the ball

The ball is controlled by four scripts that all run simultaneously and are started when the user clicks on the green flag.




The scripts and variables for the ball

First, two variables are defined that are displayed on the stage in real time. Move out the display for the points of the right player on the stage to the right.

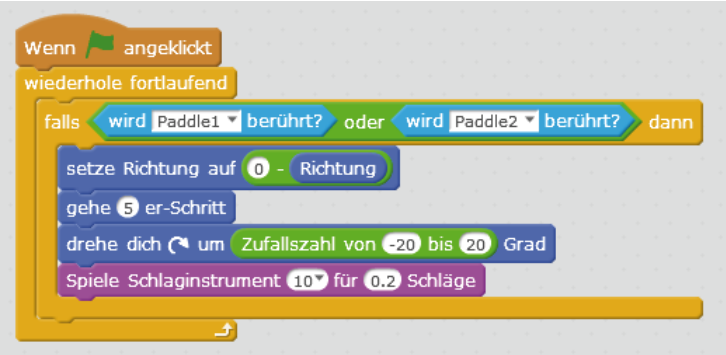
| | |
|---|---|
| **Points1** | Minus points for player1 (left) |
| **Points2** | Minus points for player2 (right) |

The main script creates the basic requirements for the game by clicking on the green flag. First the ball is brought to its starting position at **x:0 y:0.** The two scorers are then set to their initial values. Both players have **0** points.

The ball should fly off at a random angle. The direction is set to a random value between **-20** and **-160.**

The movement of the ball is then repeated continuously. It bounces off the edge if it touches it. Otherwise it makes four-step fly in the set direction. This movement theoretically repeats itself endlessly.

However, since clicking on the green flag starts three more script blocks for the ball, other movements may occur as well.




The ball changes direction when it touches one of the paddles.

If the ball touches one of the two paddles, the direction of movement is reversed. The ball flies down at the same angle from which it came from above. The current direction can be used as a variable via the **Direction** block from the **Movement** block palette. In order to make the movement a little more unpredictable, the ball is first moved a small five-step, so that it does not touch the paddle any more afterwards if possible. The flight direction is then changed by a random value between **-20** and **+20** degrees compared to the previous direction. Finally, a short tone is emitted from the PC loudspeaker.

If the ball does not touch the paddle but the red bar at the left edge, the left player gets a minus point. In order to select the colour in the "colour ... **is touched**" block, tap the colour field of the block. An eyedropper appears, allowing you to select the desired colour on stage.

The ball is returned to its starting position and rotated by a random value between **-20** and **+20** degrees so that it does not take exactly the same flight path again, but still flies approximately in the direction in which it last flew. Again another tone is emitted – this time, a more piercing one.

The same happens when the ball touches the green bar at the right edge, with the difference that this time the right player gets the minus point.

## Program for the paddles

The paddles are moved upwards step by step when the sensor contact is touched. If you release the sensor contact, the paddle drops down again in smaller steps.

The sensor contacts are again connected to two analogue inputs. If the analogue input supplies a value smaller than 200, the program assumes that the sensor contact has been touched. If there are more false reactions, try out smaller or larger limit values.



Script blocks for the paddles.


Main program block for initialization and movement of the ball.


This script block is executed when the ball touches the left red border.


This script block is executed when the ball touches the right green border.

Clicking on the green flag starts a similar script block for each of the two paddles. They first set the paddle to the start position. Then endless loops run. In the infinite loop an **if/else construct** triggers one of two possible actions each time:

- If the player touches the sensor contact, the pin is set to LOW, i.e. it behaves like an unpressed button. The y-coordinate of the paddle is increased by 40 so that it moves up by 40 units. The LED lights up for 0.1 seconds.

- As long as the player does not touch the sensor contact, the query condition is not fulfilled, and the instruction **otherwise** listed is executed. The y-coordinate of the paddle is reduced by 20 units, the paddle slides down by 20 units. This value is deliberately chosen smaller, so that the paddle does not fall so quickly, but can be moved quickly upwards again in large steps.

The left **paddle1** uses the sensor contact at pin A2 and the LED at pin D10. The right **paddle2** uses the sensor contact at pin A6 and the LED at pin D9.

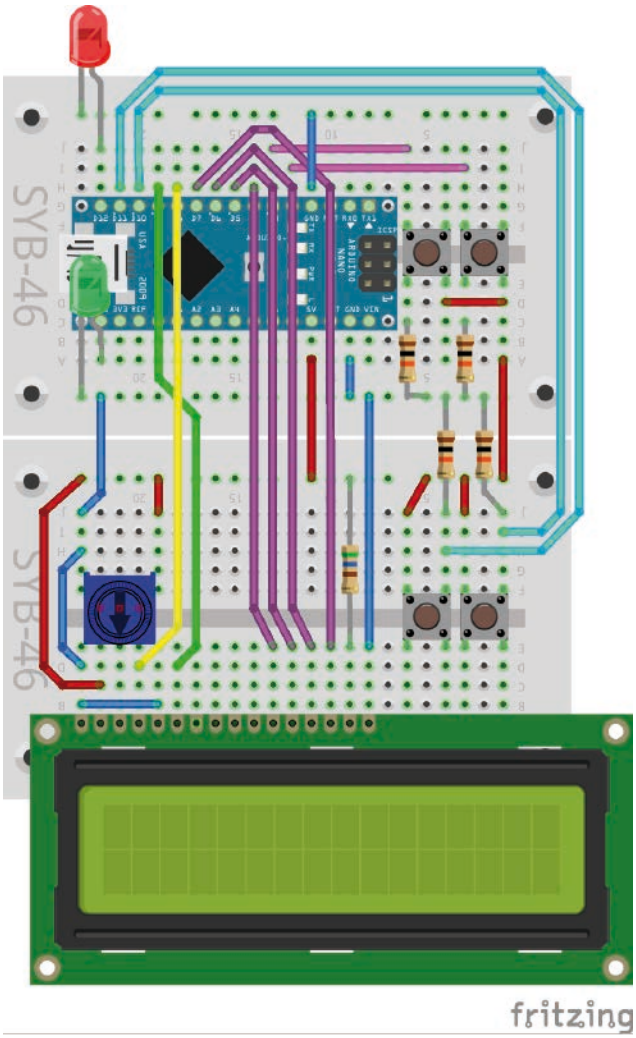
---

22. Tag

# Day 22

### Today in the Advent calendar

- 1x push-button
 1x resistor 10 kOhm (brown-black-orange)




Reaction game with LCD module, two LEDs and four buttons.

### Reaction game

In this reaction game, when the red LED is lit, you must press the correct key as quickly as possible.

**Components:** 1x Nano Board, 2x breadboard, 1x red LED with series resistor, 1x green LED with series resistor, 1x LCD module, 1x 560-Ohm resistor (green-blue-brown), 4x 10-kOhm resistor (brown-black-orange), 4x push-button, 1x potentiometer, 12x connection cable, 11x wire bridge (different lengths)

The game runs independently in the Arduino mode of mBlock 3, so the battery box can also be used to power the game independently.

### The program

The `22mblock` program uses the stopwatch on the Nano to measure the player's reaction time.

### How the program works

At the beginning of each cycle of the infinite loop, a random value between 1 and 4 is again generated with the aid of an unconnected analogue input. This number is displayed in the upper line of the LCD module. Then the red LED at pin 12 is switched on and the stopwatch is reset.

**The stopwatch on the Nano**

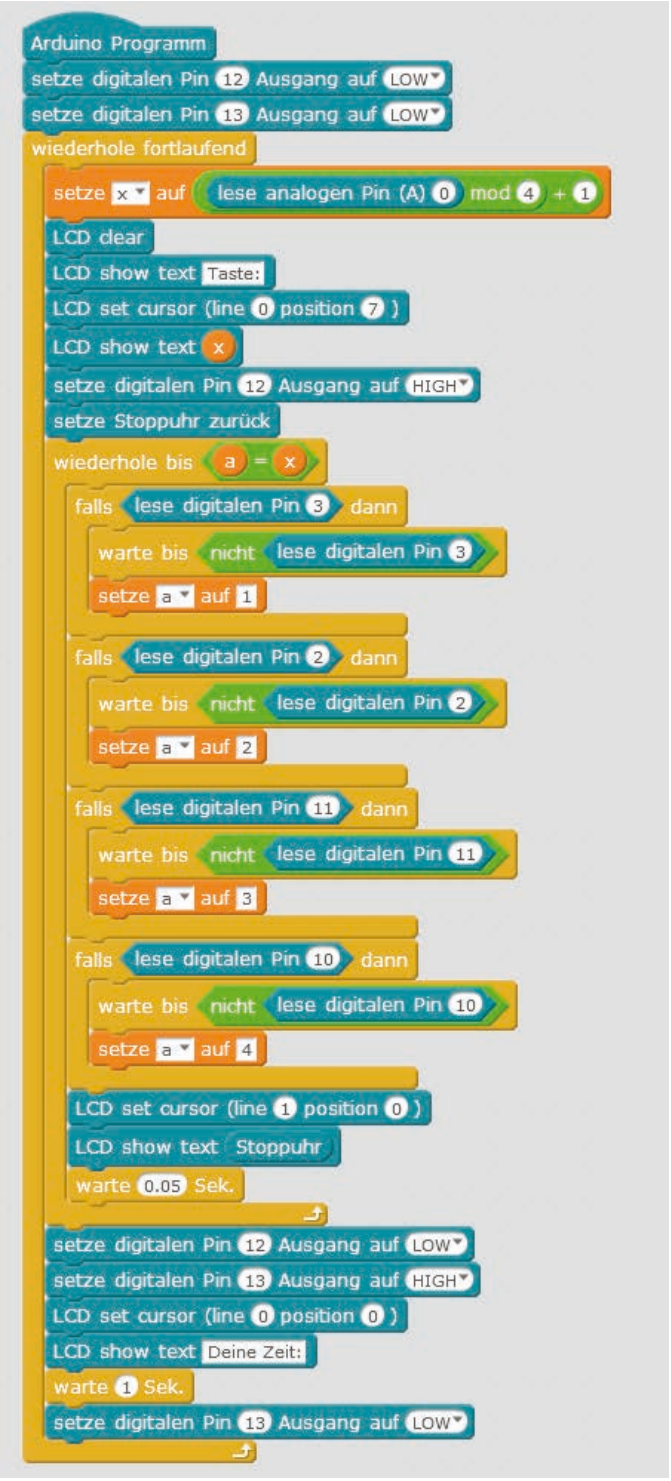
All Arduino compatible boards have an internal clock that can be used for tasks that require specific periods of time. This clock does not show a real time, but simply the elapsed time since the program start. A program can reset the clock at any time. It then restarts automatically. In this way, time periods can be measured.

A loop starts that runs until the key corresponding to the displayed number is pressed.

To do this, the four keys are queried one after the other. To prevent the game from responding when the user presses all the keys simultaneously, the variable **a,** which contains the number of the key pressed, is not set until the key is released.

In each loop pass, the current value of the stop timer is displayed in the bottom line of the LCD module.

If the player has pressed the correct button, the red LED is switched off and the green LED is switched on. The bottom line of the LCD module shows the stopped time. The program restarts after one second.



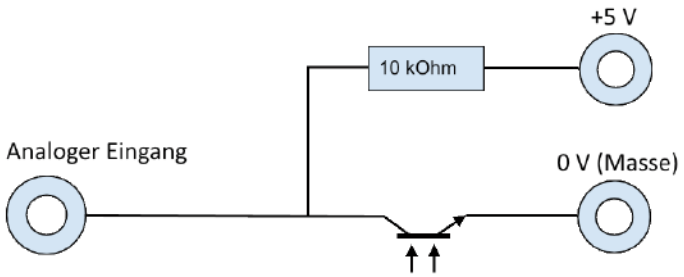

The `22mblock` program controls the reaction game.

23. Tag

# Day 23

## Today in the Advent calendar

- 1x phototransistor




Circuit diagram for a phototransistor

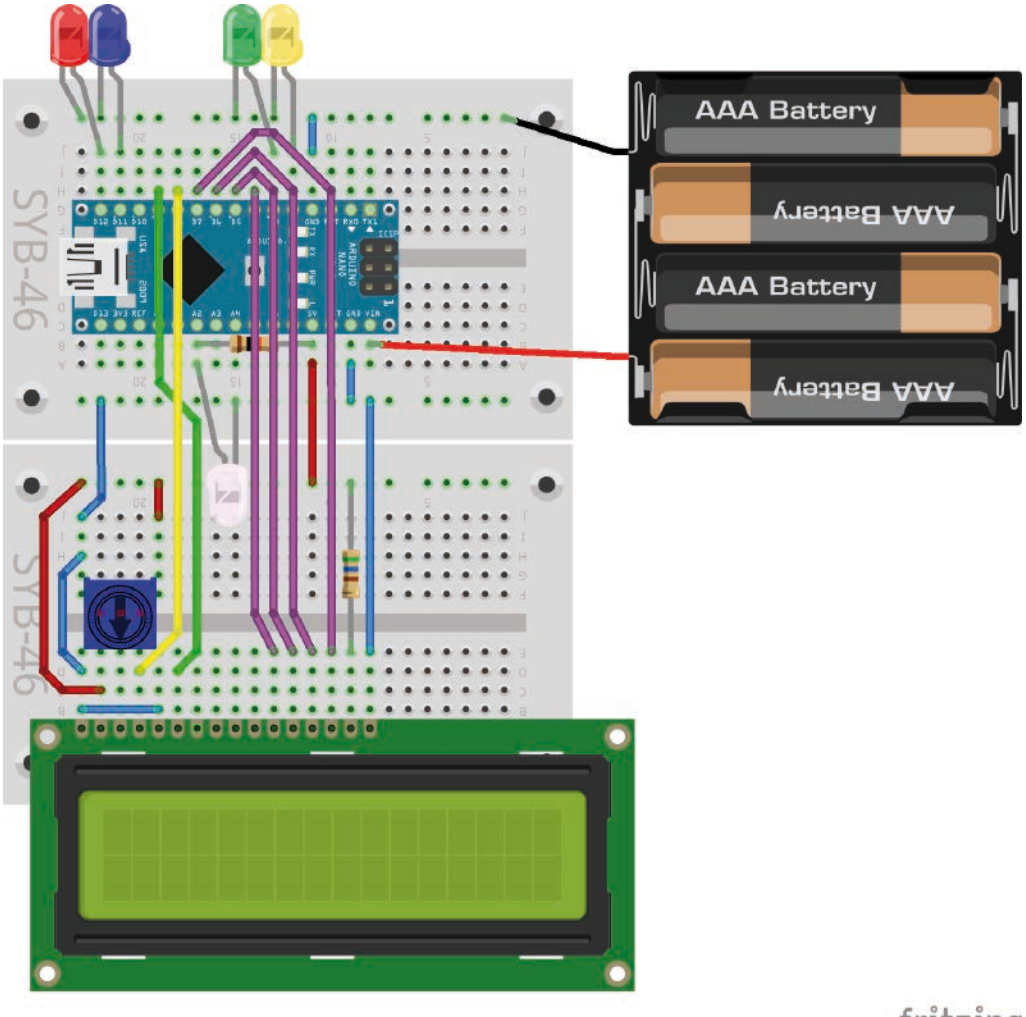
### Phototransistor

A phototransistor is a light-sensitive component that looks like a transparent LED at first glance. Depending on the intensity of the light incidence, different values can be achieved with the circuit shown on an analogue input of the Nano board. The brighter the light on the phototransistor, the lower the value at the analogue input. Unlike LEDs, a phototransistor connects the long terminal to ground, not the short one.

### Christmas light effects in the dark

The experiment of the 23rd day causes LEDs to flash brightly when it is dark enough. After programming, connect the battery box and carry the circuit into a dark corner. The LEDs will then start flashing. A level indicator on the LCD module shows the current illumination level.

**Components:** 1x Nano Board, 2x breadboard, 1x red LED with series resistor, 1x green LED with series resistor, 1x yellow LED with series resistor, 1x blue LED with series resistor, 1x LCD module, 1x 560-Ohm resistor (green-blue-brown), 1x 10-kOhm resistor (brown-black-orange), 1x phototransistor, 1x potentiometer, 1x battery box, 8x connection cable, 7x wire bridge (different lengths)




LCD module, 4 LEDs and photo transistor.

### The program

The `23mblock` program displays the brightness falling on the photo transistor as a bar on the LCD module. Transfer the program from mBlock 3 to the Nano in Arduino mode. It can then also be supplied independently by the battery box.

### How the program works

Depending on the brightness, the phototransistor delivers an analogue value between 0 and 1023, which is inversely proportional to the value-amount of light that falls on the phototransistor. This value is divided by 64 to obtain 16 possible levels for the bar graph. In order for brighter light to produce a longer bar, a higher value is needed in this case than for dark light. Therefore the calculated value is subtracted from 15 and stored in the variable **a.**

The counter **n** starts at 0. Now a loop starts again, which runs 16 times for the 16 digits of the LCD module. For each digit, the system checks whether the number **n** is greater than the level value stored in **a. If** this is the case, the corresponding position on both lines is switched off. For this purpose, a sequence of blanks is displayed at this point, which also overwrites asterisks further to the right that are still there from earlier. Since character strings consisting only of spaces are not correctly transferred to the Nano in mBlock 3, the character string contains a dot at the last position, which is never displayed due to the length of the LCD module.

If the number **n is** not greater, i.e. less than or equal to the level value, an asterisk is displayed at the position on the LCD module. In this way, all the places are always checked, regardless of whether the brightness value is higher or lower than the last measurement.

If the value **a is** less than 1, i.e. only minimal light falls on the phototransistor, the four LEDs flash alternately briefly in pairs. Since the next iteration begins immediately afterwards, an apparent even flashing appears in darkness, although each LED is switched on and off only once after the measurement.


The `23mblock` program shows light effects in darkness.

---

## Day 24

**24. Tag**

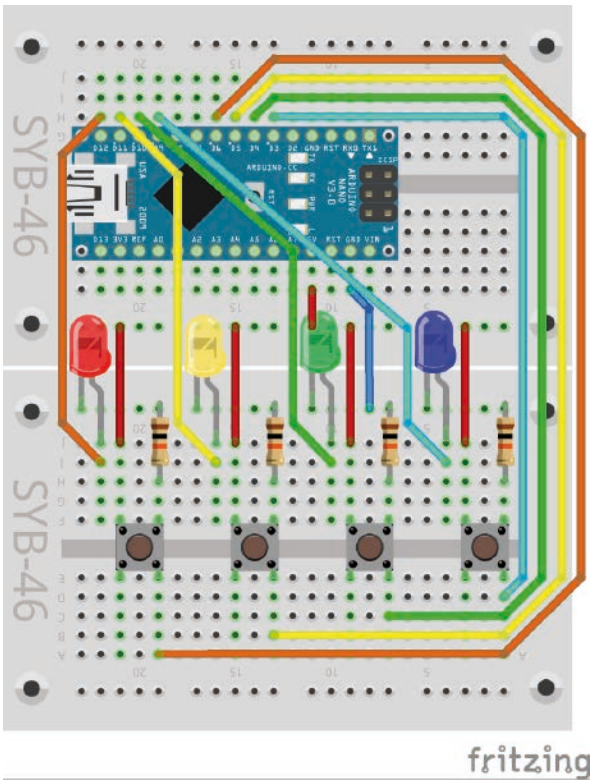
### Today in the Advent calendar

- 1x download code

### Christmas jukebox

The program of the 24th day plays a Christmas song when a button is pressed. There are four different songs to choose from. You can download Christmas carols in mp3 format using the download code in the Advent calendar today.

**Components:** 1x Nano Board, 2x breadboard, 1x red LED with series resistor, 1x yellow LED with series resistor, 1x green LED with series resistor, 1x blue LED with series resistor, 4x 10-kOhm resistor (brown-black-orange), 4x button, 8x connecting cable, 6x wire bridge (different lengths)




Music machine with Christmas carols.

## The program

The `24mblock` program plays four different Christmas carols depending on which button the user presses. The matching LED also lights up.


The program for the Christmas carols.

## How the program works

At the beginning variables for the pin numbers of the four LEDs and buttons are created and all LEDs are switched off.

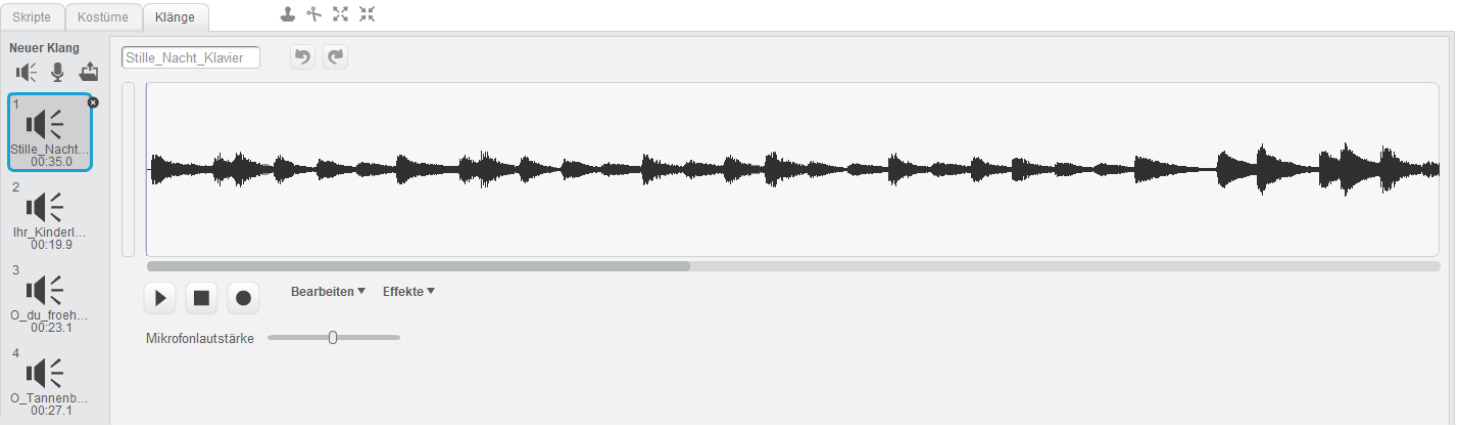
An infinite loop then waits for the user to press one of the keys. For this purpose, four identically structured queries run one after the other, one for each key.


Button 1 plays Silent Night.

If the program detects that a key has been pressed, it waits until this key is released again. This prevents malfunctions caused by prolonged pressing.

The LED is switched on and one of the songs is played. The LED is then switched off again.

The Christmas carols are stored in the program. On the **Sounds** tab, you can import MP3 files using the "**Load sound from a file**" icon. Each imported sound automatically gets a name that can be changed at any time. Under these names the sounds appear in the selection list in the "**play sound ... in full**" block.




The Christmas carols are stored in the program.

The download code today in the Advent calendar provides some more Christmas songs in MP3 format. With it you can exchange the songs pre-installed in the program according to your personal taste.

**We wish you a merry Christmas**
Those who hear this melody in the pre-Christmas period almost daily in shopping centres and at Christmas markets suspect a modern American Christmas song behind it. In fact, however, it is a song from the English Christmas tradition that can be traced back to the year 1500 and can therefore be used today without copyright.

Merry Christmas!