CE

Alle Versuche im Überblick

Advent Calendar Raspberry Pi 2018

This advent calendar contains a hardware experiment with the Raspberry Pi for every day. Step by step, you will build a Christmas manger that will shine with all its glory on 24th December. The experiments are programmed with Scratch or Python. This programming language is pre-installed on the Raspberry Pi. All experiments work with Raspberry Pi 3 and Raspberry Pi 3 B+.

Controlling simple electronic systems – even just a few LEDs – with a normal PC or laptop is associated with almost unacceptable effort for the hobby programmer. The PC simply does not have the interfaces required for this. Also, the Windows operating system is pretty much unable to communicate with electronic parts.

The Raspberry Pi is a full-fledged computer – even if it does not look like it at first. Many things are a bit slower than what we are used to from modern PCs, but the Raspberry Pi is much smaller and, most importantly, much cheaper than a PC.

1. Day

Today in the advent calendar

- 1 patch panel
- 1 yellow LED with series resistor
- 2 GPIO connection cables (short)

Preparing the Raspberry Pi

To start the raspberry Pi, you will need:

- USB keyboard and mouse
- HDMI cable for the monitor
- Power cable
- MicroSD card with operating system Raspbian Stretch
- Micro USB mobile phone charger as power adapter (at least 1,500 mA)

The power adapter must be the last to be connected because the Raspberry Pi will switch on automatically. There is no extra on/off switch.

Installation of the operating system in brief

For all of you, who do not have their Raspberry Pi ready for operation with the current Raspbian version, follow the system installation in eleven steps:

Download NOOBS (at least version 2.4.5, for Raspberry Pi 3 B+ at least 2.7.0) from **www.raspberrypi.org/downloads** on your PC and unpack the zip archive on the hard drive.

Reformat the SD card using the SD formatter on the PC if it has been used before: **www.sdcard.org/downloads/formatter_4**. When doing this, switch on **Format Size Adjustment** (the SD card must have a memory of at least 8 GB).

Copy the NOOBS files and directories to the SD card.

Remove the SD card from the PC, insert it in the Raspberry Pi and boot. Select **English** as installation language at the very bottom. This will automatically select the English keyboard.

Check the box of the pre-selected Raspbian operating system and click **Install** in the top left. After confirming a safety message that the memory card will be overwritten, the installation starts automatically, which will take a few minutes.

The Raspberry Pi reboots when installation is complete.

Under **Settings** in the menu, start the tool **Raspberry Pi Configuration**.

On the tab **Localisation** in the field **Specify time zone**, select the options **Europe** and **Berlin**. **Language** and **keyboard** should be set to German automatically. If not, select German location and keyboard.

If you want to use WiFi on a Raspberry Pi 3 B+, use the button to select **Select WiFi country** and the option **DE – Germany**. The WiFi on Raspberry Pi 3 still works without this setting.

On the tab **Interfaces**, set the switch **SSH** to **Activated**, if you want to transfer data from the PC to the Raspberry Pi via the network.

Click **OK** and re-boot the Raspberry Pi via the menu item **Shutdown**. A warning may appear regarding an unsafe password, which can be ignored.

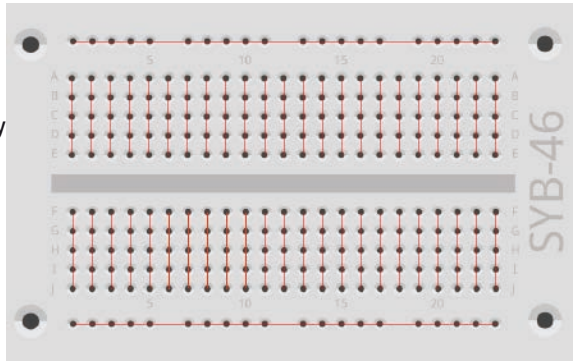
Num Lock key
Identical to almost all other Linux systems, the number pad is locked as default when starting Raspbian. Press the Num Lock key on the keyboard to unlock it.

The most important components in brief

Patch panel

The advent calendar comes with a patch panel so that electronic circuits can be built quickly without the need for soldering. Electronic components can be attached directly into a breadboard.

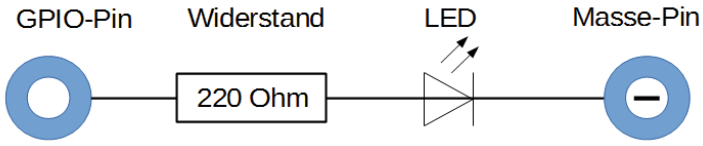
All outer longitudinal rows are connected with each other on this patch panel via contacts (X and Y). These contact rows are often used as plus and minus poles to supply the circuitry with power. In the other contact rows, five contacts (A to E and F to J) are connected laterally at a time with a gap in the middle of the board. This allows you to attach larger components and wire them with the surroundings.



The connections on the patch panel.

LEDs

LEDs (short for: light emitting diodes) when electrical current moves through them in the direction of the current. In circuit diagrammed, LEDs are indicated with a triangle symbol in the shape of an arrow, which shows the direction of the current from the plus poles to the minus pole or to the ground wire. One LED conducts almost any current in the direction of the current. It only has a very low resistance. To limit the current and, therefore, prevent the LED from blowing, a 220-ohm series resistor is typically installed between the used GPIO pin and the anode of the LED or between the cathode and the ground pin. This series resistor also protects the GPIO output of the Raspberry Pi from excessive currents. The LEDs in the advent calendar have a series resistor already installed and, therefore, can be connected directly to the GPIO pins.



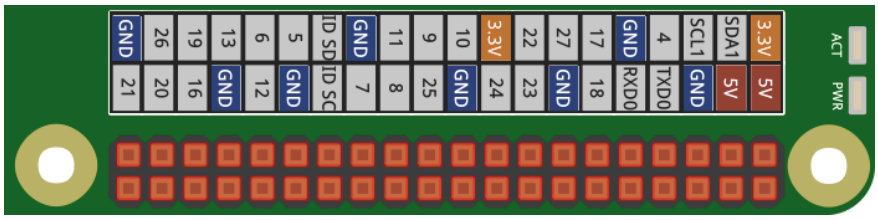
Circuit diagramme of an LED with series resistor.

Connecting the LED in which direction?

The two connecting wires of an LED are of different lengths. The longer wire is the plus pole or anode, and the shorter one is the cathode. Memory hook: The plus symbol has one more line than the minus sign so that the wire for the plus pole should be longer. Furthermore, most LEDs are flattened on the minus side, which is comparable to a minus sign. Another memory hook: cathode = short = edge.

GPIO connection cable

The coloured connection cables have a plug on one side and a socket on the other, which fits into a GPIO of the Raspberry Pi. The LEDs on the manger are also connected directly to these sockets. The plugs are plugged into the patch panel. The programmable GPIO pins on the Raspberry Pi are numbered; the ground pins are indicated with GND in the figure.



Allocation of the GPIO pins.

Precautions

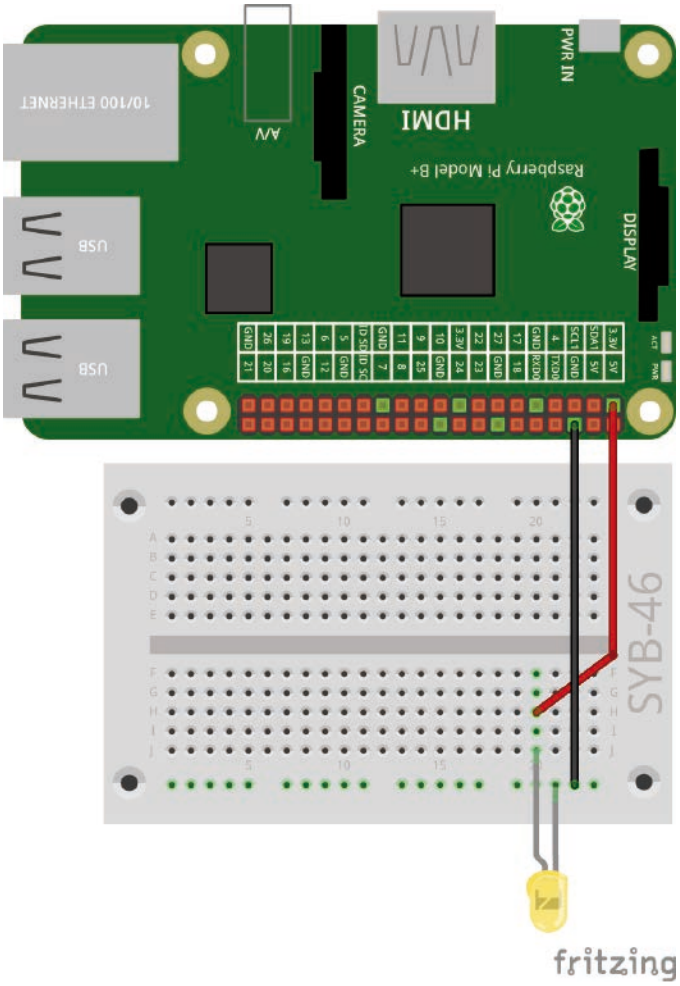
Never connect any GPIO pins with each other and wait to see what happens.

Not all GPIO pins are freely programmable. Some are firmly allocated for power supply and other purposes.

Some GPIO pins are connected directly to processor connections. A short-circuit can destroy the Raspberry Pi completely. However, when connecting two pins with each other via a switch or an LED, a protective resistor must always be installed in between. One exception are LEDs with installed series resistor.

Pin 1 supplies 3.3 V and can be loaded with up to 50 mA. It must always be used for logic signals. Pin 6 is the ground connector for logic signals.

Pins 2 and 4 provide +5 V for the power supply of external hardware. They can provide as much electricity as the USB power adapter of the Raspberry Pi can supply. However, these pins must not be connected with a GPIO input.



The first LED at the Raspberry Pi is illuminated.

One LED is illuminated

The first experiment does not require a programme. The Raspberry Pi is only used as power supply for the RGB LED. The experiment shows how RGB LEDs are connected. Pay attention to the correct installation orientation of the LED. The flat surface is on the right in the figure.

Most circuits use the contact strip on one long side of the patch panel as ground contact. The cathodes of all LEDs are inserted here and connected to a GND pin on the Raspberry Pi using a cable.

Components: 1 patch panel SYB-46; 1 yellow LED with series resistor; 2 short GPIO connection cables (Raspberry Pi – patch panel)

Programmes for download

The programmes used for the advent calendar can be downloaded here:
http://bit.ly/c-adventskalender-raspberry-pi-18

Open the webpage directly with the browser installed on the Raspberry Pi and download the zip-file into the home directory `/home/pi`.

Start the file manager on the Raspberry Pi. It shows the home directory automatically when started. Right-click on the downloaded zip-file and select **Unpack here** in the context menu.

The download archive for the advent calendar contains this manual as PDF in colour so that you can recognise the individual wires better in the circuit diagrams.

2. Day

Today in the advent calendar

· 3 GPIO connection cables (long)

The advent calendar contains GPIO connection cables in two different lengths. The short cables are used for a connection between the Raspberry Pi and the patch panel; the long cables are used to connect the LED on the manger with the patch panel.

The first LED on the manger is illuminated

The LED is inserted into the marked holes on the tail of the Christmas star over the manger. Cut it out of the cardboard on the back of the advent calendar along the lines.

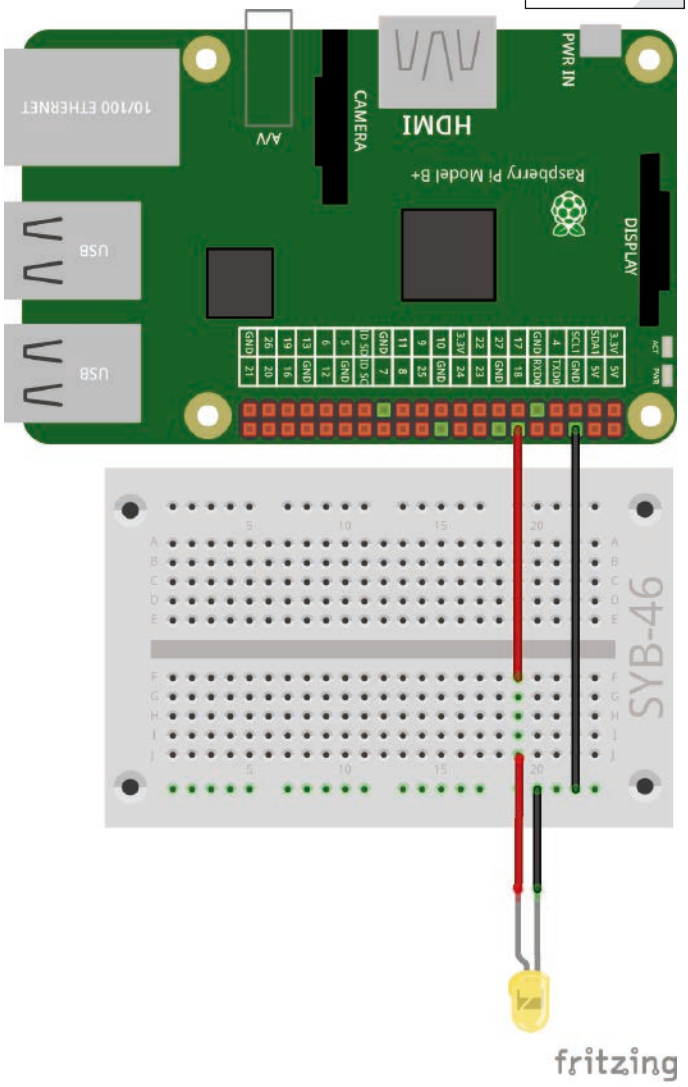
The sockets of the GPIO connection cables are connected to the connecting wires of the LED from the back. Insert the LED into the cable sockets as far as possible. If it still slides out, bend the last few millimetres of the ends of the LED connecting wires down a little.

Components: 1 patch panel SYB-46; 1 yellow LED with series resistor; 2 short GPIO connection cables (Raspberry Pi – patch panel); 2 long GPIO connection cables (patch panel – LEDs)

Attach the LED to the top of the star above the manger.

This time, the LED is not permanently illuminated, but it is switched on for half a second by a programme in the programming language Scratch.

Scratch is pre-installed on the Raspberry Pi under **Development** in the menu. It is one of the programming languages that is easier to learn.



The first LED on the manger is illuminated

The new Scratch 2

Since the first Raspbian Version, version 1.x of the programming language Scratch is pre-installed. The new version Scratch 2 with a lot more possibilities has been available for PCs for several years. Amongst others, individual function blocks can be created.

Scratch 2 runs online in the browser on a PC. However, this requires more computing power than the Raspberry Pi can currently provide. Since version NOOBS 2.4.0, a version of Scratch 2 is pre-installed in the Raspbian operating system. It runs offline, without the browser and therefore works with the power of a Raspberry Pi 3 without problems. Controlling the hardware via the GPIO interface has become considerably easier with Scratch 2. However, some important functions for GPIO control are not supported yet. Therefore, we will use the established Scratch 1.4 version for all projects in this advent calendar.

Basic settings for Scratch

Click on the globe in the top left next to the Scratch icon and select **German**. The selected language is saved so that it is not necessary to select it again every time.

Scratch 1.4 offers support for different hardware component at a GPIO port, which must be activated once in every programme using the menu item **Edit/start GPIO server**.

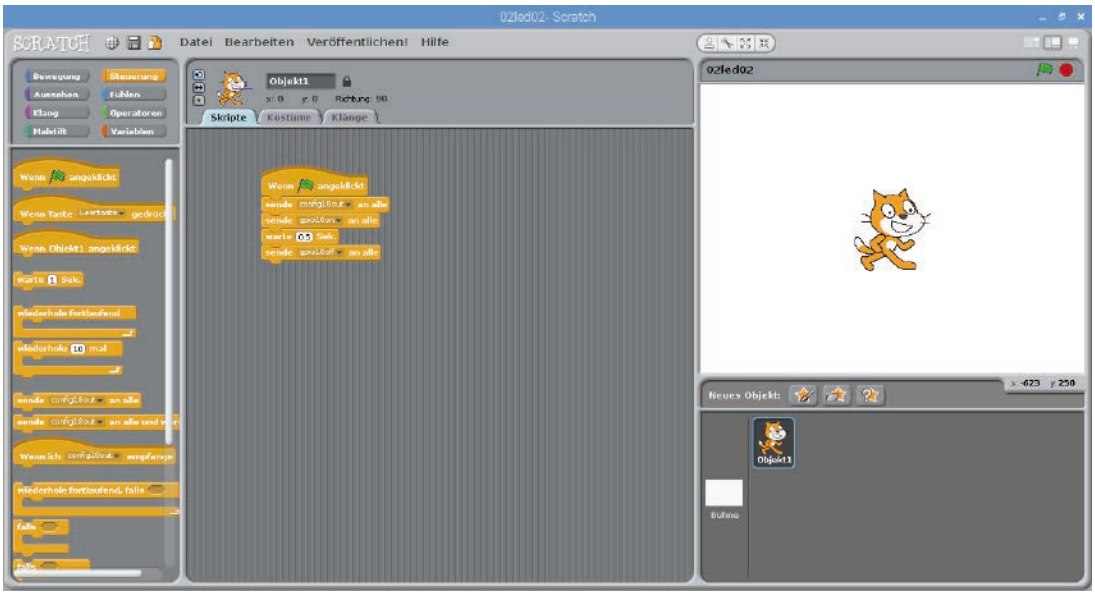
Changing Scratch 1.4 to German.

Starting the Scratch 1.4 GPIO server.

You can see whether the GPIO support is activated when this menu item changes to **Stop GPIO server**. Check this every time a Scratch programme is started.

The programme

You do not need to type code when programming in Scratch. The blocks are simply added to one another with drag-and-drop. The block pallet on the left side of the Scratch window shows the available blocks ordered by theme.



This Scratch programme `02led02` makes the LED light up for half a second.

You can create the programme on the screen yourself or you can use programme `02led02` from the download for the advent calendar. To do this, select **File/Open** and click on the button **pi** in the next dialogue to select the personal home directory where the downloaded programmes are saved.

In Scratch, click on the yellow icon **Control** at the top left so that the control block is shown in the block pallet. For the first programme, we only need the yellow blocks.

Drag the blocks you need simply from the block pallet into the script window in the middle of Scratch.

The block **If (green flag) clicked** is used to start a programme. The following script elements are run when clicking on the green flag in the top right in Scratch. The top of the block is round. Thus, it does not fit under another block. It must always be placed first.

The GPIO commands are made with the Scratch block **send... to all**. The respective pin names and key words are entered into the text field. To do this, click in the text field in the block, select **New/edit...** and enter the text.

At first, the GPIO pin 18 is defined as output with **config18out**. Every GPIO pin can be either input or output.

In the next step, the LED connected to GPIO pin 18 is switched on with another Scratch block **send... to all** with the text **gpio18on**.

Then the programme waits for half a second. Scratch has its own block **wait...sec** for this. Like many other American programmes, Scratch uses the dot as decimal separator and not the comma as is typical in Germany. Thus, to indicate half a second you need to enter `0.5` and not `0,5`.

In a last step, the LED connected to GPIO pin 18 is switched off again with another Scratch block **send... to all** with the text **gpio18off**.

The programme starts when clicking the green flag in the top right of the Scratch window.

3. Day

Today in the advent calendar

- 1 red LED with series resistor
- 1 GPIO connection cables (short)

Two LEDs flash alternately

The experiment of Day 3 makes two LEDs light up alternately in red and yellow. Everything is controlled with an infinite loop in Scratch.

Components: 1 patch panel; 1 red LED with series resistor; 1 yellow LED with series resistor; 3 short GPIO connection cables (Raspberry Pi – patch board)

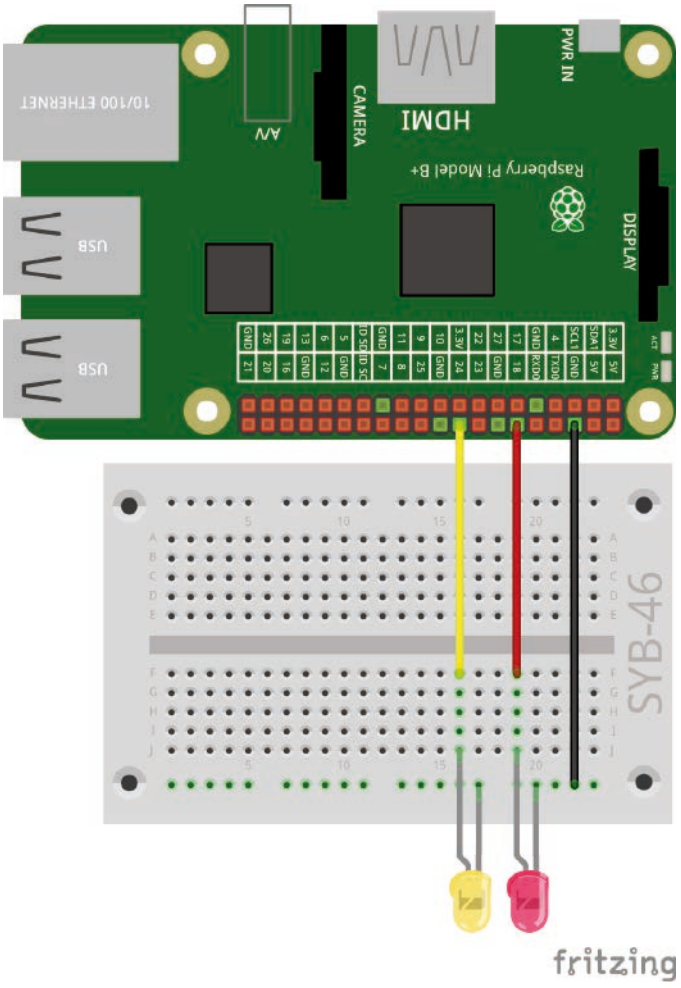
In this experiment, the LEDs are attached directly to the patch panel with the cathodes in the ground strip.

The programme

At first, the two GPIO pins 18 and 24 are defined as outputs with **config18out** and **config24out**.

A **repeat continuously** loop makes the two LEDs flash infinitely until the user clicks on the red stop symbol in the top right in Scratch.

When the red LED at pin 18 is switched on and the yellow LED at pin 24 is switched off, the programme waits for half a second. Afterwards, the yellow LED at pin 24 is switched on and the red LED at pin 18 is switched off in the same manner. After another half a second, the cycle starts from the beginning.



Two LEDs flash at the Raspberry Pi.

The programme `03led03` controls the two LEDs.

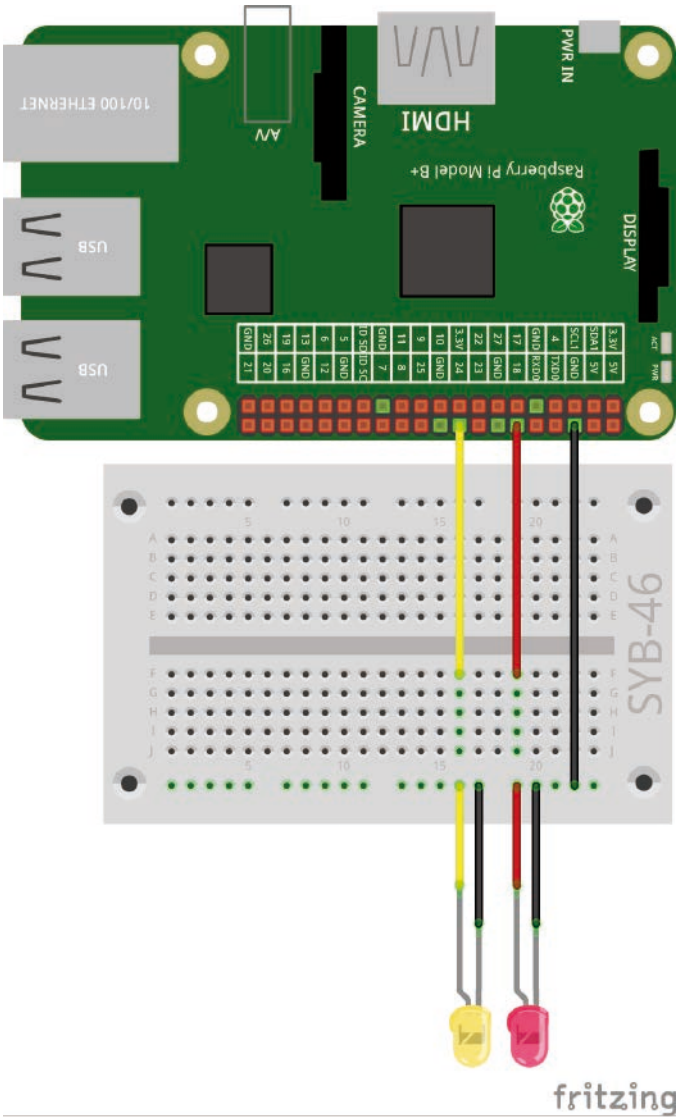
4. Day

4. Day

Today in the advent calendar

· 3 GPIO connection cables (long)

A star toggles two LEDs



Two LEDs are toggled with a Scratch programme.

In the experiment of the 4th day, two LEDs are switched on or off. It is controlled by an object in Scratch that can be moved on the screen.

Components: 1 patch panel SYB-46; 1 yellow LED with series resistor; 1 red LED with series resistor; 3 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs)

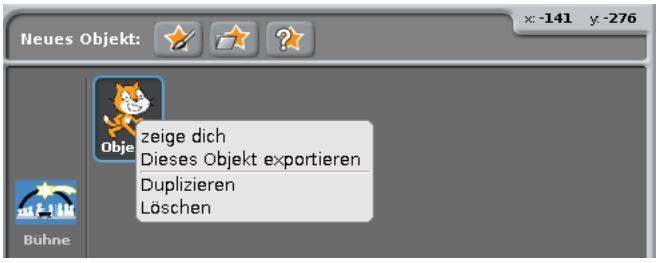
The programme

The programme uses the manger picture on the back of the advent calendar as a screen background.

Click on the stage in the object window (bottom right) and then on the tab **Backgrounds** in the script window.

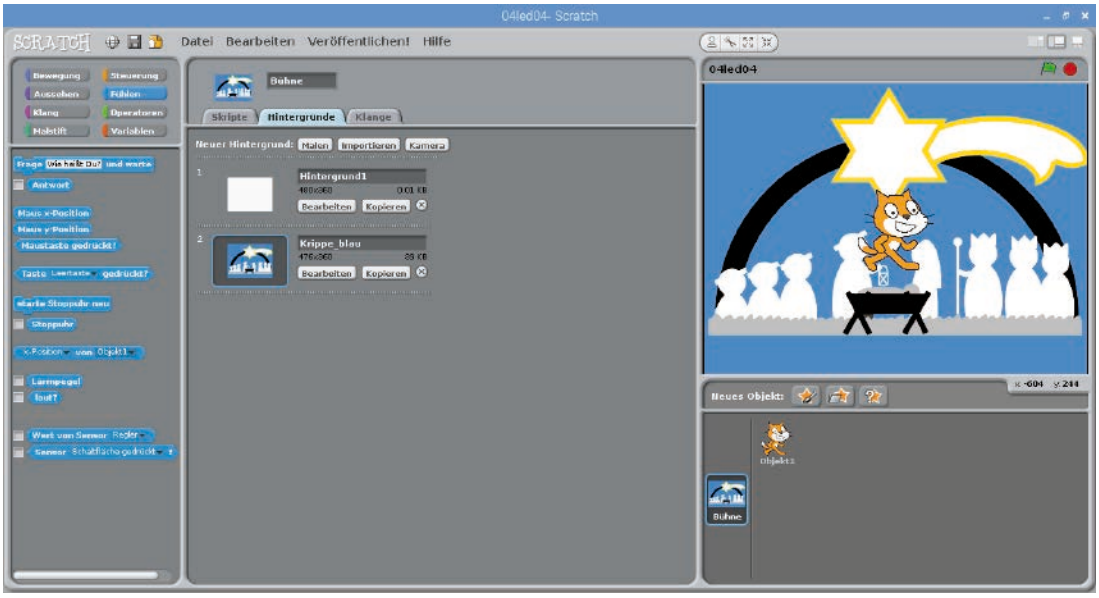
Click on **Import** and import the picture `Manger_blue.png` from the download. Afterwards, remove the default background, **Background 1** by clicking the X icon.

Then, delete the cat, which is not required for this programme. To do this, right-click on the cat in the object window and select **Delete** in the context menu.



Deleting the Scratch cat.

In the programme, a **star** object is to toggle the LEDs. Scratch includes a simple drawing programme you can use to draw this object directly in Scratch. To do this, click on the symbol **Draw new object** in the object window.

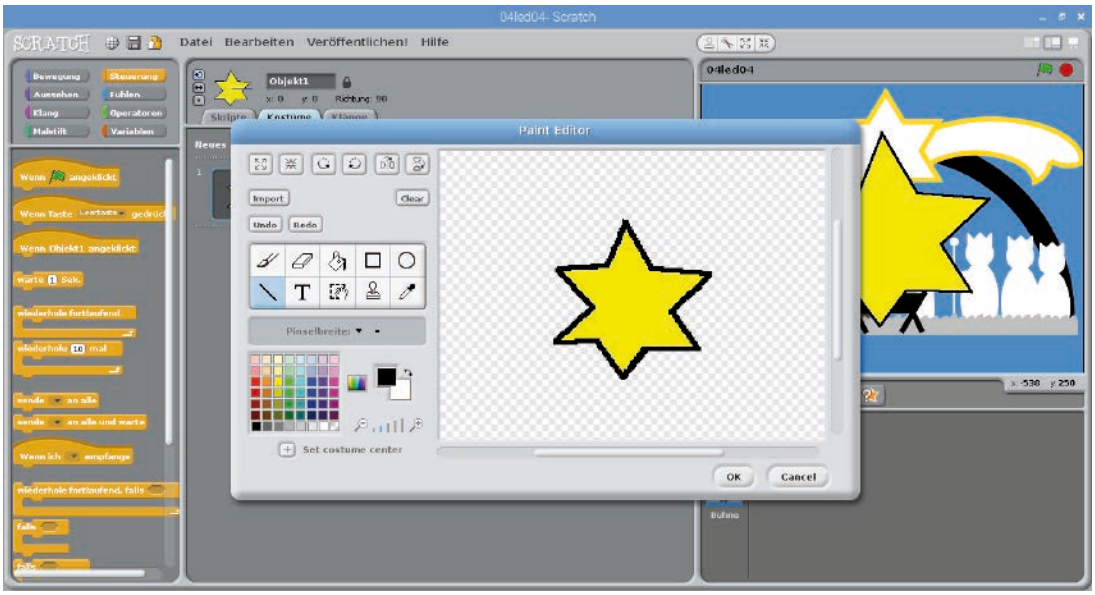


The new background picture for the programme.

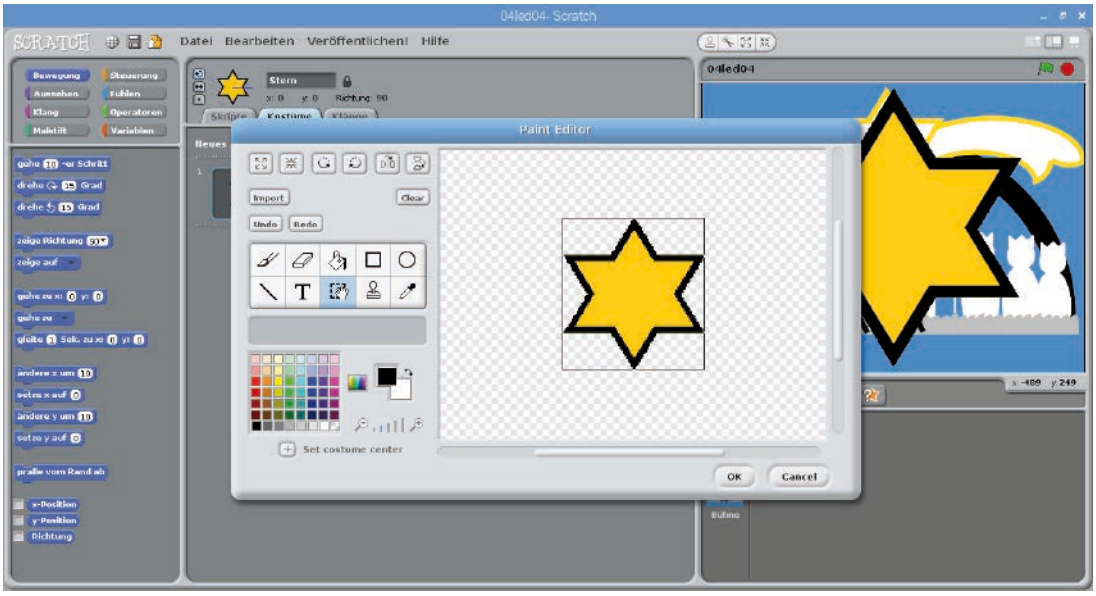
Alternatively, you can use the symbol **Load new object from file** to import the file `star.png` as new object from the download.

By default, the star is much too big for the scene. In the script window, click on the tab **Costumes** and then **Edit** to open the drawing programme.

Make the star smaller until it has the appropriate size using the symbol **Smaller** on the left.

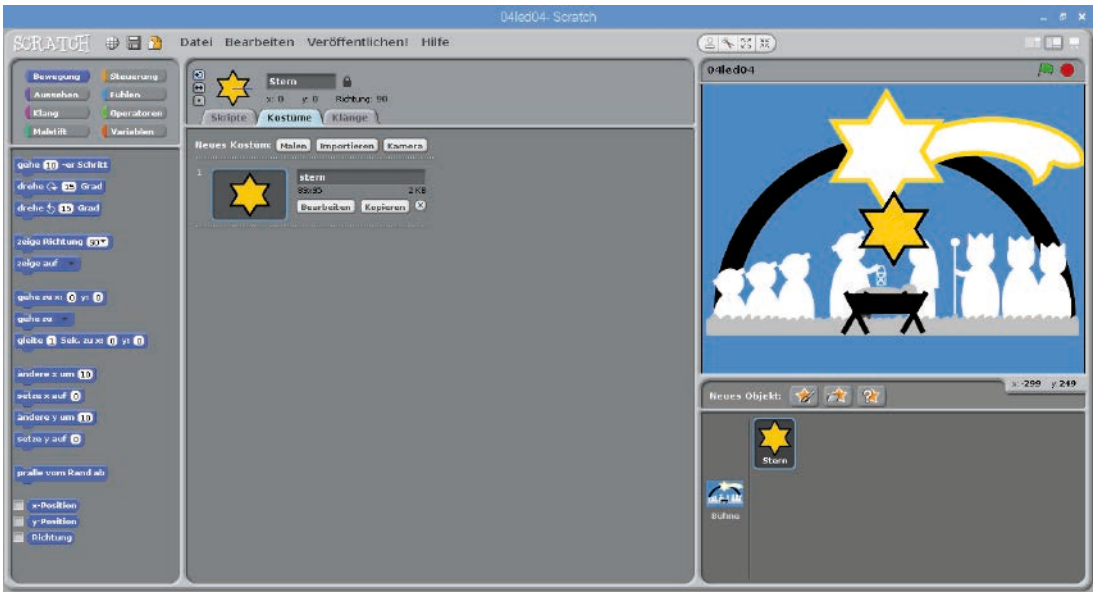


The drawing programme in scratch.



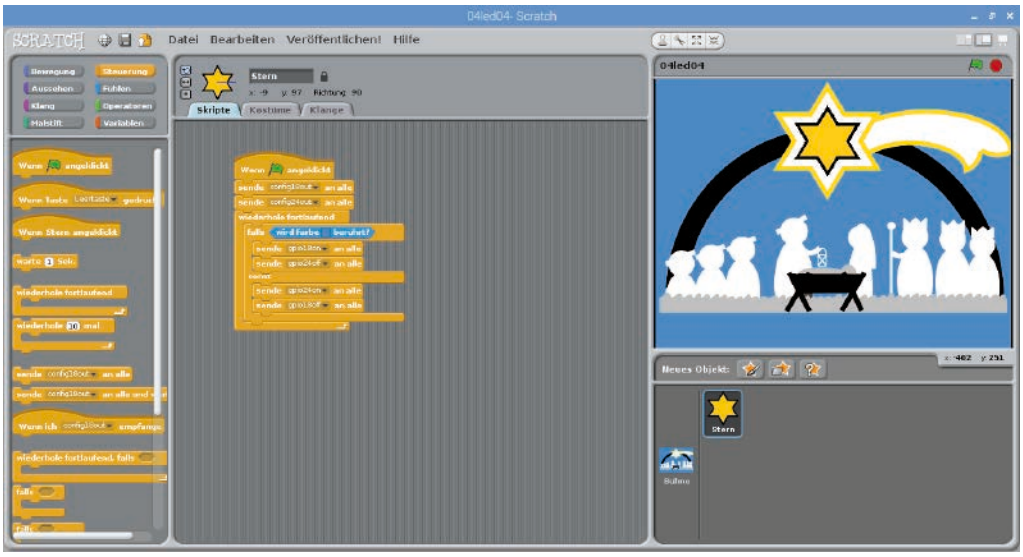
The imported star in the drawing programme in Scratch.

Name the new object into **Star** in the name field above the script window.

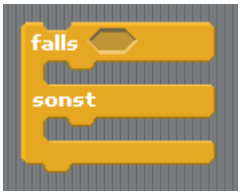


The smaller, renamed star.

Move the star on top of the large star of the background picture.



The programme `04led04` toggles two LEDs with a star.



After initialising the GPIO pins, the programme `04led04` starts an endless loop, which monitors if the star touches a blue surface. As long as this is not the case, it is located on the star in the background picture – one single white area that is large enough.

An **If ... else ...** block switches one LED on and the other one off depending on the location of the star.

Drag a block **colour ... is touched** from the block pallet **Sensing** into the query field of the **If ... else ...** block.

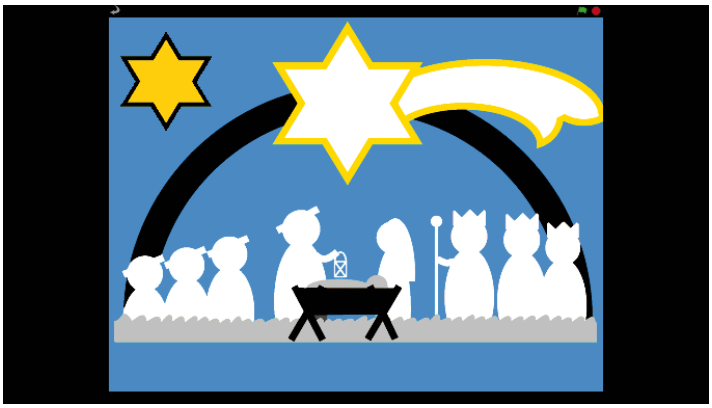
When tapping on the coloured square inside of this block, a pipette appears. Use to click on a blue area in the background picture. This colour is now selected in the query.

Now, add the **Send ... to all** block to the brackets of the **If ... else ...** block to toggle the LEDs, as shown in the picture.

Start the programme by clicking on the green flag and the LED at pin 24 will light up.

Drag the start towards one of the screen corners so that it touches the blue background. The programme automatically switches to the other LED.

The presentation icon in the top right corner of the Scratch window switches to the presentation mode, in which the Scratch stage appears in full screen mode. All operating controls disappear. You can return by clicking on the arrow in the top left.



The programme `04led04` in presentation mode.

5. Day

Today in the advent calendar

- 1 button
- 1 GPIO connection cables (short)

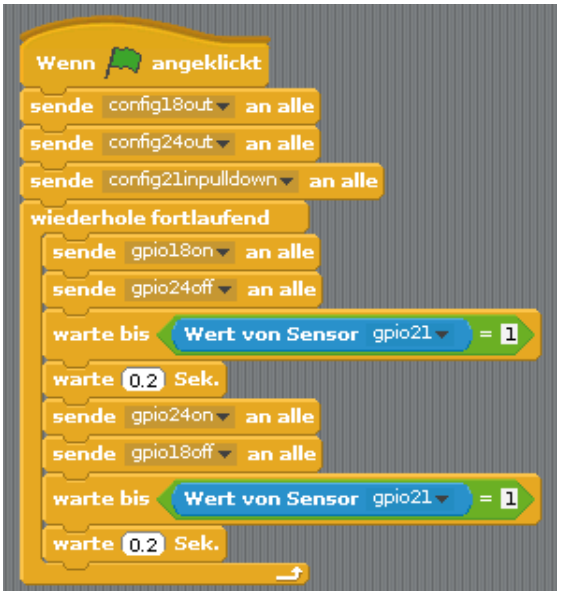
Toggling LEDs with a button

The experiment of Day 5 does not toggle the LEDs automatically, but only when the user presses a button.

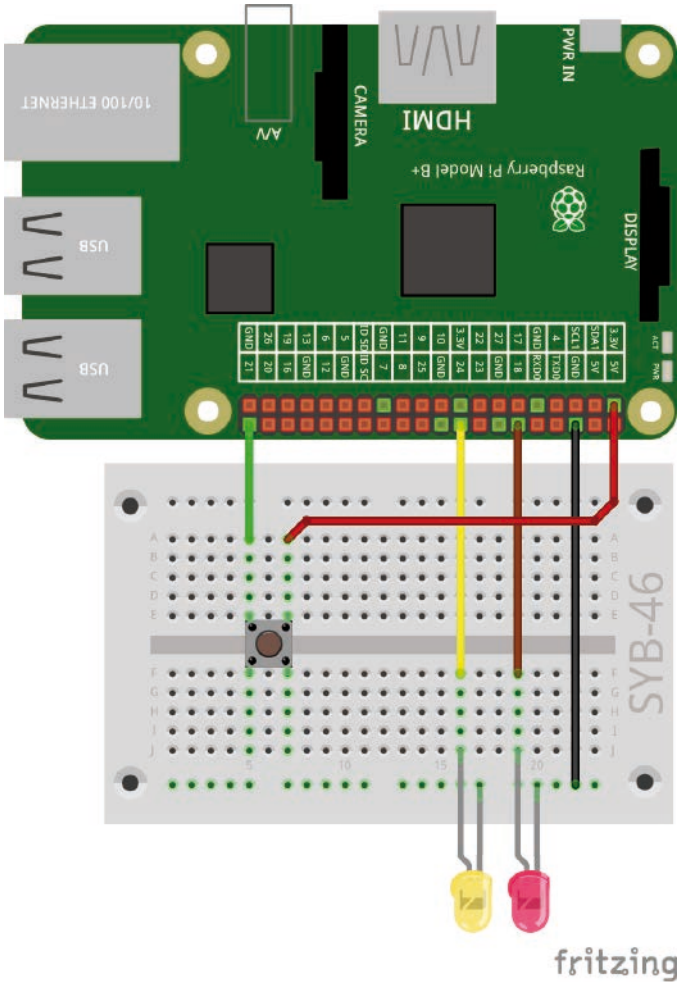
Components: 1 patch panel SYB-46; 1 yellow LED with series resistor; 1 red LED with series resistor; 1 button; 4 short GPIO connection cables (Raspberry Pi – patch panel); 1 long GPIO connection cable (up to this day, the number of short cables is insufficient)

The programme

The programme works in a similar way to the one on Day 3. Similarly, the two LEDs are switched on and off alternately in an infinite loop. In contrast to the previous programme, the LEDs are not toggled after a certain time, but only when the user has pressed the button. Each time the button is pressed the LEDs are toggled.



The programme `04led05` controls the two LEDs via a button.



Toggling LEDs with a button.

At first, GPIO pin 21 is defined as input in addition to the two outputs. Inputs at the Raspberry Pi process digital logic signals. If an input with +3.3 V is connected, it receives the logic signal **High**, which is seen as **1** in Scratch. If the input is connected to GND, it receives the logic signal **Low**, which is seen as **0** in Scratch.

Caution
Never use the +5 V pins of the Raspberry Pi for logic signals in circuits. 5 V would overload the GPIO inputs and damage the Raspberry Pi.

In our circuit, pressing the button connects GPIO pin 21 with +3.3 V. When the button is released, the input is undefined, which must not happen in digital electronics. In such cases, all GPIO pins have so-called pull-down resistors, which define an input without a signal automatically as **Low**.

Define the GPIO pin for the button with **config21inpulldown** in order to activate the inbuilt pull-down resistor at the input.

Now click the green flag in the top right once to start the incomplete programme. This will define the GPIO pins. They can be selected in the lists only in this way. The programme ends automatically.

The GPIO pins are initialised.

Subsequently, an endless loop will start, which first switches the LED at GPIO pin 18 on and the LED at pin 24 off.

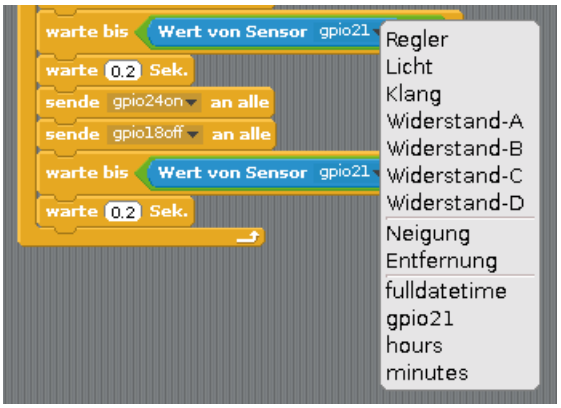
When an LED is lit, the programme no longer waits for a certain amount of time but uses a **wait until...** block to wait until a certain event occurs; in this case, until the GPIO pin 21 assumes the value **1**, i.e. the button is pressed.

For the query, a long field with pointed ends in the **wait until...** block is used. Here, a block from the green block pallet **Operators** must be inserted. Pull the block with the equal sign into the placeholder field in the **wait until...** block.

This operator is always true when both values on the left-hand side and the right-hand side of the equal sign are the same.

In our case, the value of GPIO pin 21 should be **1**. The value 1 means **High**. Thus, enter a `1` into the text field on the right in the green block.

To query GPIO inputs, the block **Value of sensor...** from the blue block pallet **Sensing** is used. Select the sensor **gpio21** in the list view of the blue block. In addition to some pre-defined sensors, all GPIO pins are shown that are defined as input. This is why the programme needs to be started briefly once.



Drag the blue block **Value of sensor...** into the left field of the green block inside the block **wait until...**.

The programme will then wait for 0.2 seconds using a block **wait...sec**. This will prevent the button from being recognised as pressed immediately when the programme continues to run. This is the time the use has to release the button.

The LEDs will be toggled only afterwards and the programme waits again until the user presses the button.

Duplicating blocks
When building programmes in Scratch, you do not need to make similar block combination again each time. Right-click on the first block you want to duplicate. Then, select **Duplicate** in the menu. All blocks below the selected block will then be duplicated automatically as well. The duplicated blocks can be inserted into the correct place within the programme.

6. Day

Today in the advent calendar

- 4 GPIO connection cables (long)

Toggling LEDs with the Scratch control panel

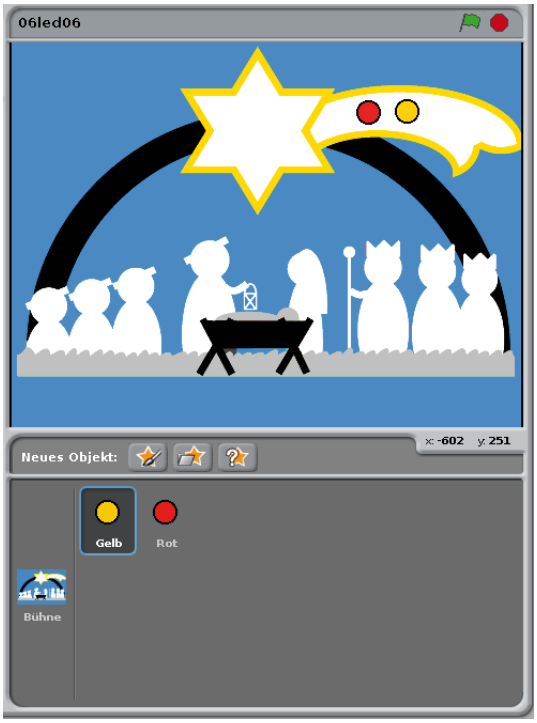
In the programme of Day 6, two LEDs are switched on and off by clicking in Scratch. The Scratch control panel will indicate the status of the LEDs.

Components: 1 patch panel SYB-46; 1 yellow LED with series resistor; 1 red LED with series resistor; 3 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs)

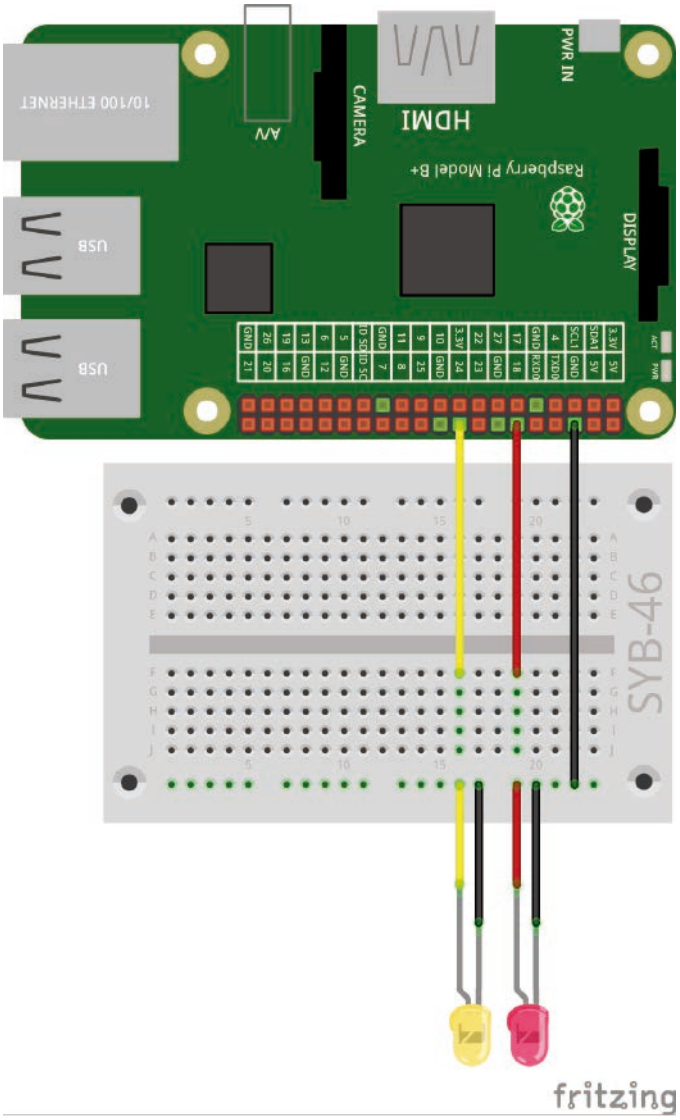
The programme

The programme `06led06` uses the same background picture as the programmes of the last days.

Two circular objects with the names **Yellow** and **Red** are created for both LEDs on the tail of the star. Delete the cat and draw one of these objects. You can also import the file `circle_yellow.png` from the download. To create the second object, you can duplicate the first object later, including all settings and script blocks, and make a few adjustments.



The two objects for the LEDs.



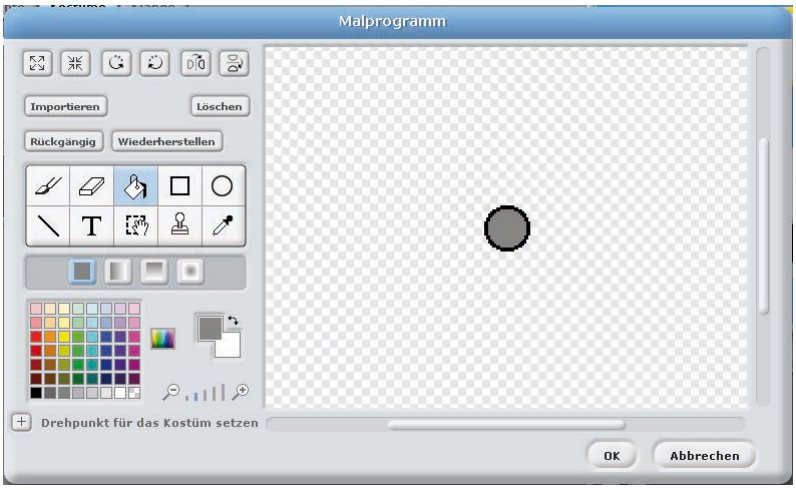
Three LEDs on the manger.

The appearance of each object in Scratch can be changed with so-called costumes. Select the LED object and click on the tab **Costumes** in the script window.

Copy the existing costume by clicking on **Copy**. Rename the two costumes: **one** for when the LED is switched on and **off** for when it is switched off.

Edit the costume **off** with the drawing programme. Fill the coloured area with grey to indicate the switched off LED.

The two costumes of the object Yellow.



Costume for a switched off LED in the drawing programme.

The programme consists of several parts – for the stage and the objects. First, select the stage in the object window. The script block of the stage initialises the GPIO pins when clicking the green flag.

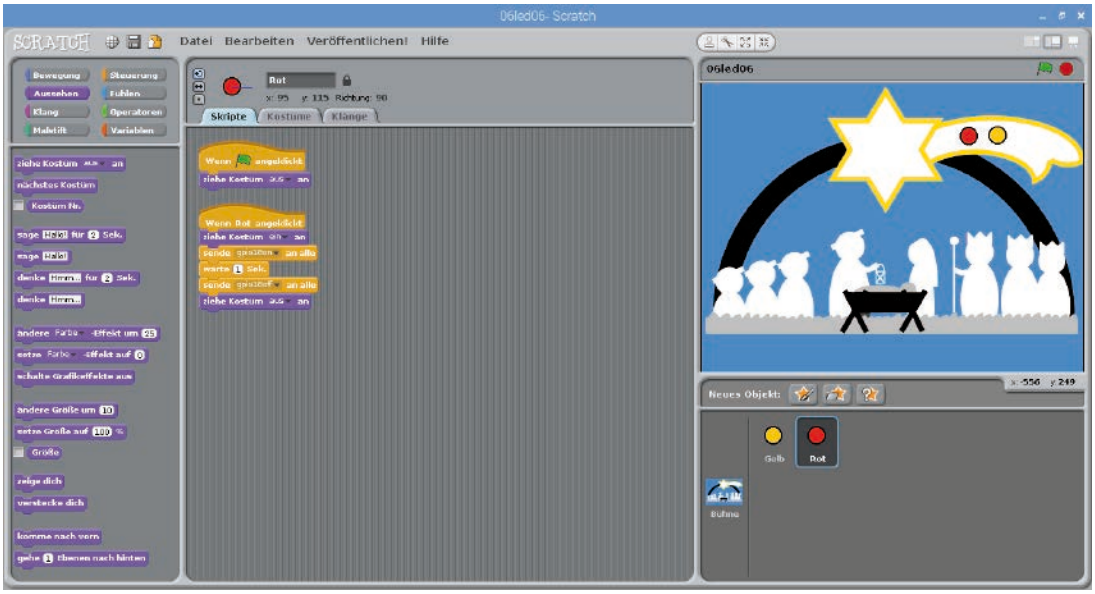
The object **Yellow** will have its own script blocks. When the green flag is clicked, the object should be switched off, i.e. should be grey. To do this, add a block **Put on costume ...** from the block pallet **Appearance** to the block **If green (green flag) clicked**. Select the costume **off** from the selection field of the block.

If the object is clicked, it should light up in colour on the screen for one second. At the same time, the corresponding LED should also light up.

To do this, use the block **If yellow clicked** from the tab **Control**.

In this case, the costume is changed to **on** and then the LED at GPIO pin 24 is switched off. After waiting for one second, this LED is switched off again and the costume is changed back to **off**.

Once all script blocks are completed, duplicate them with a right-click into the object window of the object **Yellow**. Rename the new object to **Red**.



The duplicated object for the red LED.

Edit the costume **on** with the drawing programme. Fill this circle with red. Subsequently, change the used GPIO pin in the script block to 18.

Start the programme by clicking on the green flag. Both LEDs are off, the objects appear yellow. When clicking on one of the objects, it will appear in colour and the corresponding LED lights up for one second.

7. Day

Today in the advent calendar

- 1 green LED with series resistor

Running light with three LEDs

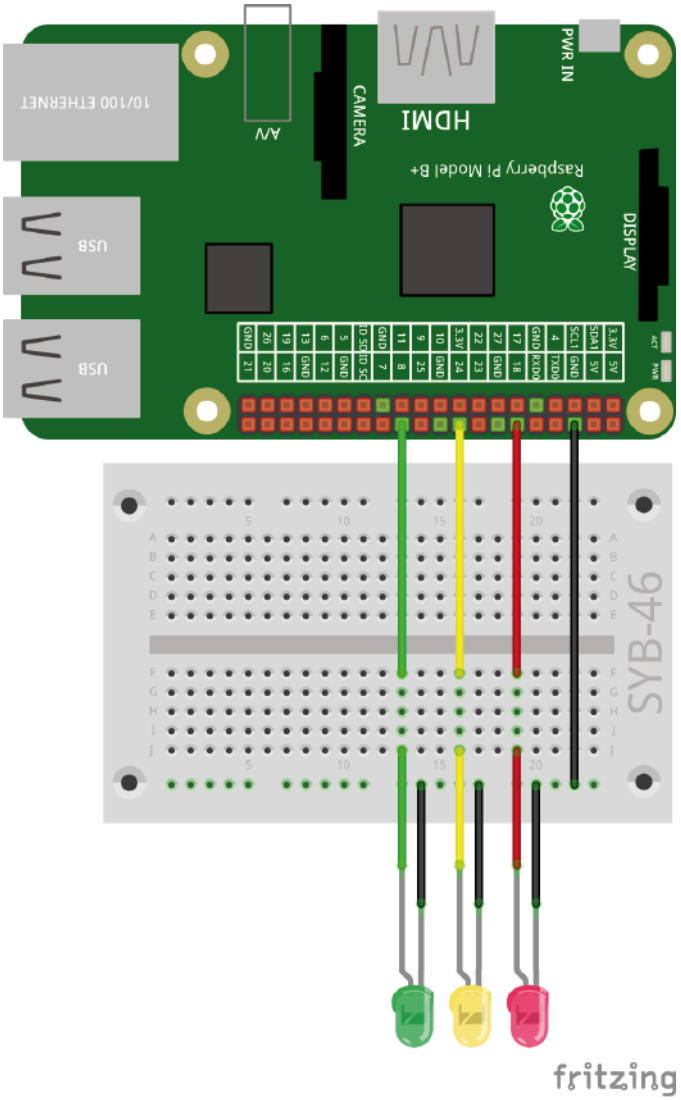
The programme of Day 7 makes three LEDs light up alternately. The running light effect is realised by one LED being lit up for a short time and the LED adjacent to it will be switched on immediately once the first LED is switched off. After the last LED in the row is switched off, the first LED will be switched on again.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 red LED with series resistor; 4 short GPIO connection cables (Raspberry Pi – patch panel); 6 long GPIO connection cables (patch panel – LEDs)

The programme

The programme `07led07` makes the three LEDs light up as running lights. The speed can be set interactively while the programme is running. To do this, the programme uses a variable with the name **pause**, which dictates the pause between two switches.

The programme `07led07` makes the three LEDs light up as running lights.



Running light with three LEDs.

Variables in Scratch

Variables are small memory spaces, in which a programme stores a number or something else. When the programme is closed, these variable memory spaces are erased automatically. In Scratch, variables first need to be defined in the block pallet **Variables** by clicking on **New Variable** before they can be used. Subsequently, you can drag the symbol of the new variable from the block pallet into a designated field of a block in the programme. There are also various blocks for reading and adjusting of variables on the block pallet.

The programme runs entirely on the stage; it does not contain objects. After the start, the three GPIO pins 18, 24 and 8 for the LEDs are first defined as outputs and all are switched off.

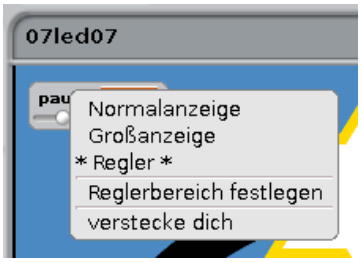
A block **set pause to 5** from the block pallet **Variables** sets the pause to 5, which corresponds to half a second in the programme.

Afterwards, an endless loop starts, which switches the LEDs on and off again in succession. Since only whole numbers can be set interactively as variable values in Scratch, the programme divides the value of the variable **pause** by 10. To do this, use the block **.../...** from the block pallet **Operators**. Drag the variable icon **pause** from the block pallet **Variables** into the left field of the operators and enter `10` on the right.

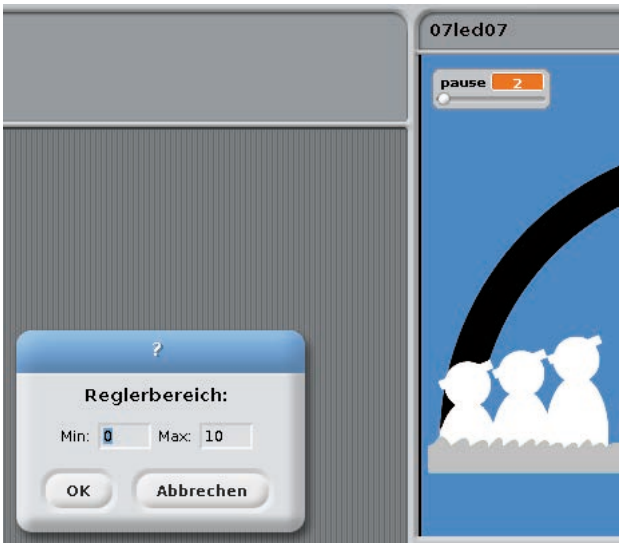
To be able to change the value of the variable interactively in the programme, check the box left of the variable symbol on the block pallet **Variables**. The variable will then appear on the stage and always display the current value.

Right-click on the variable on the stage and select **Controller** in the context menu. A controller will appear, with which the variable value can be set.

Then, also click on **Define controller range** and set the maximum value to **10**, which corresponds to a pause of one second when the value is later divided by 10.



Start the programme. The LEDs automatically flash one after another. You can now set the speed using the slider for the variable **pause** on the stage. Variables on the stage are also shown in presentation mode.



8. Day

Today in the advent calendar

· Switch wire

Switch wire

The advent calendar contains a switch wire today. You can use it to create short connections, with which contact strips on the patch panel are connected with each other. Cut the wire with a wire cutter to the appropriate length depending on the experiment. To be able to insert the wires better into the patch panel, it is recommended to cut them at a slight angle creating a small wedge.

Dimming LEDs

LEDs can assume two different conditions: on and off. The same is true for the GPIO ports defined as digital outputs. Thus, it would theoretically not be possible to dim an LED.

Left: key ratio 50 % – right: key ratio 20 %.

However, there is a trick with which the brightness of an LED connected to a digital GPIO port can be controlled. If you make an LED flash very fast, the human eye cannot perceive the flashing. The method is called pulse width modulation and generates a pulsing signal, which switches on and off in very short intervals. The signal voltage remains constant, only the ratio between the **False** level (0 V) and the **True** level (+3.3 V) is changed. The key ratio defines the ratio of the length of the switched-on condition and the total duration of the switching cycle. The smaller the ratio, the shorter is the duration when the LED is lit during a switching cycle. This makes the LED appear darker than a permanently lit LED.

Why 50 Hertz is the ideal frequency for PWM

The human eye does not detect changes in light that are faster than 20 Hertz. Since the DC grid in Europe uses a frequency of 50 Hertz, many bulbs flash with this frequency, which cannot be detected by the human eye. An LED flashing with more than 20 Hertz but less than 50 Hertz may cause interferences with other light sources so that the dimming effect no longer appears to be even.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 2 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs); one connecting wire

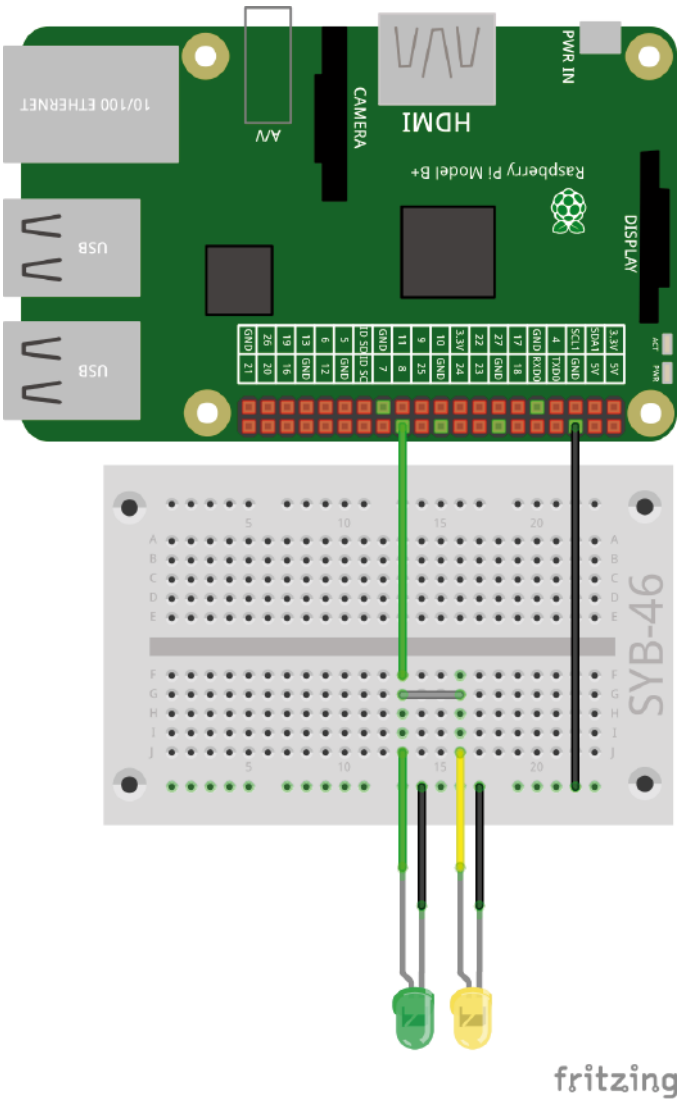
The PWM effect appears to be differently strong for different LED colours. In the circuit, the two LEDs are connected to the same GPIO pin in order to demonstrate this effect clearly. To do this, two strips of the patch panel are connected with a short switching wire.

The programme

The programme `08led08` dims the two LEDs with an interactively changeable value.

The programme `08led08` dims the two LEDs.

First, the programme defines the GPIO pin 8 as output with PWM function. For a pin defined in this way, Scratch offers special functions to control a PWM signal.



Two LEDs are dimmed with a GPIO pin.

Then the previously defined variable **PWM** is assigned the value **0**. In this state, the PWM pin is completely switched off. Afterwards, an endless loop queries the interactively changeable variable **PWM** and subsequently sets GPIO pin 8 to the new value.

Set the controller range of the variable to **Min.:0** and **Max:500**. Scratch will expect the values for PWM within this range.

In order to send the set value to the pin via **send ... to all**, we will use the block **connect ... ...**, which combines two arbitrary texts into one by adding them to each other. Here, numbers are treated like text, i.e. they are not added but also listed one after another.

Type the characters `gpio8PWM` into the first field of the block and drag the variable **PWM** into the second field. This will assign a certain value to a PWM pin. Then drag the block, **connect ...** in the text field of the block **send ... to all**.

The LEDs light up with the set brightness; the endless loop, then restarts and queries the new value of the variable **PWM**.

9. Day

Today in the advent calendar

- 4 GPIO connection cables (short)

Running light with button

The programme of Day 9 is another running light, but this time with special features. The LEDs only light up if a button is pressed, and the brightness of each LED can be set via PWM.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 red LED with series resistor; 6 short GPIO connection cables (Raspberry Pi – patch panel); 6 long GPIO connection cables (patch panel – LEDs)

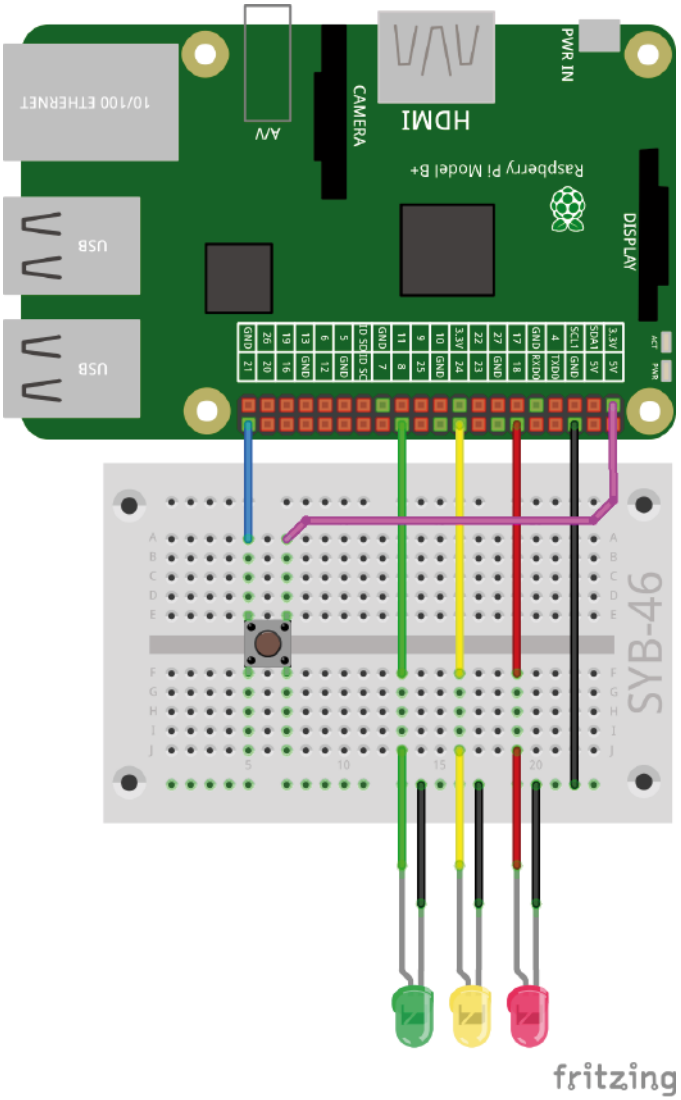
The programme

At the start, the programme `09led` initialises the GPIO pins 8, 24 and 18 as PWM outputs and pin 21 as input with pull-down resistor.

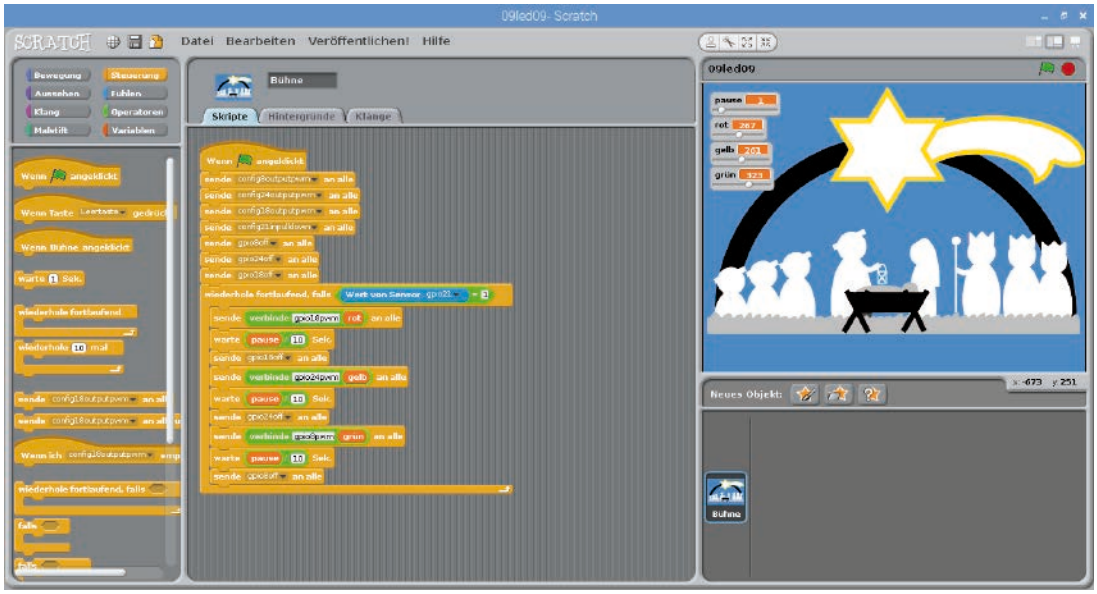
An endless loop continuously queries if signal one is present at pin 21, i.e. that the button is pressed.

The PWM values of the three LEDs are saved in the variables **red**, **yellow** and **green**, which can all be set with a controller within the range **0...500**. The pause can be set within the range **0...10** again and is converted into a tenth of a second like on Day 7.

Start the programme by clicking on the green flag. Set the PWM values and hold the button pressed to see the running light. The values can also be changed when the button is not pressed because the programme keeps running. As soon as you press the button again, the running light starts with the current PWM settings.



Three LEDs and one button.



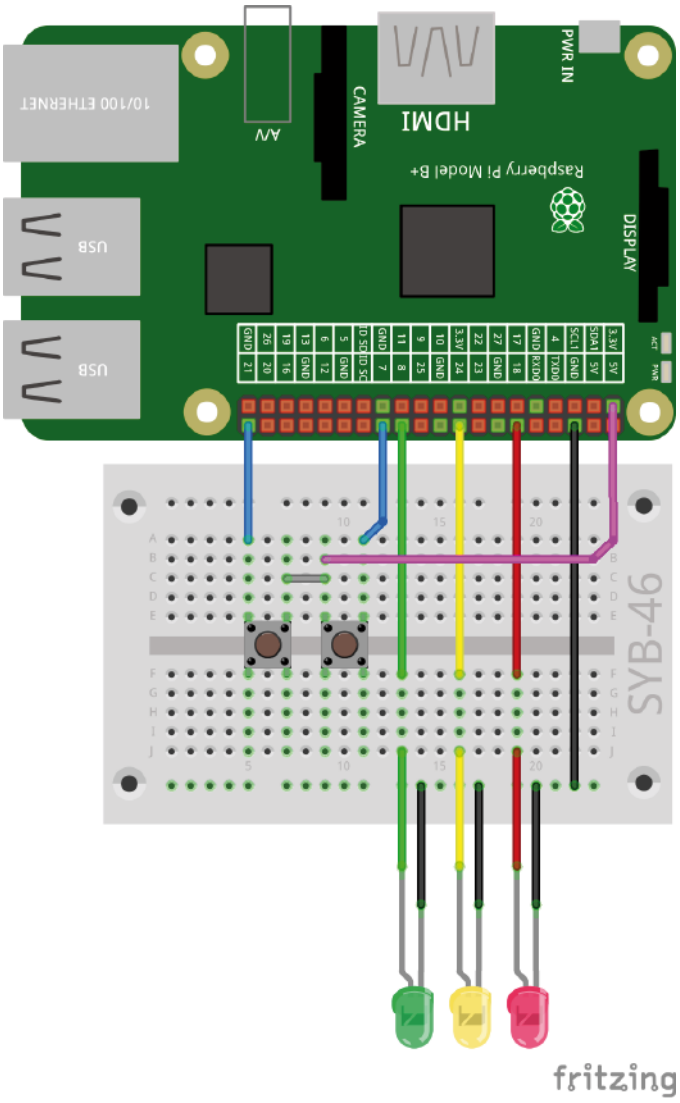
The programme `09led09` controls a running light with PWM LEDs.

10. Day

10. Day

Today in the advent calendar

- 1 button
- 1 GPIO connection cables (short)



Three LEDs and two buttons.

Toggling LEDs with two buttons

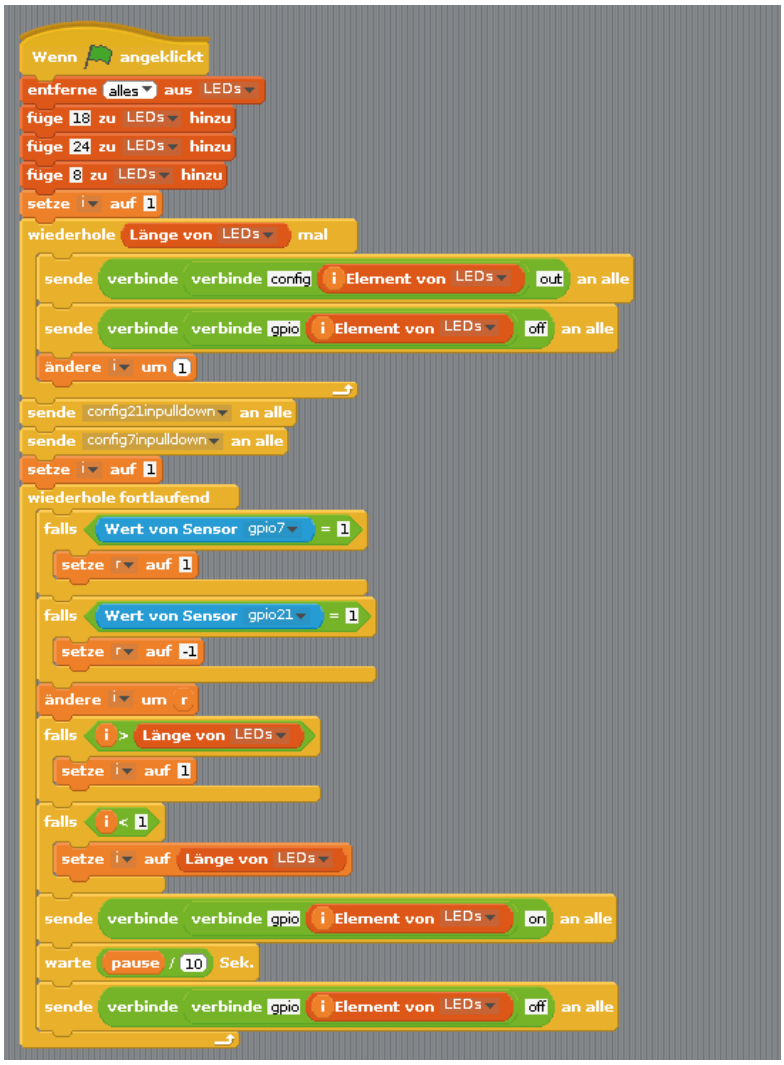
The direction of the running light can be selected with two buttons.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 red LED with series resistor; 2 buttons; 7 short GPIO connection cables (Raspberry Pi – patch panel); 6 long GPIO connection cables (patch panel – LEDs); one connecting wire

The connecting wire is used to connect two contact strips of the patch panel with which the two buttons are connected to the +3.3 V-Pn of the Raspberry Pi.

The programme

The programme `10led10` makes three LEDs light up and the direction is toggled with two buttons. To do this, the programme uses a list instead of individual variables for the three GPIO pins.



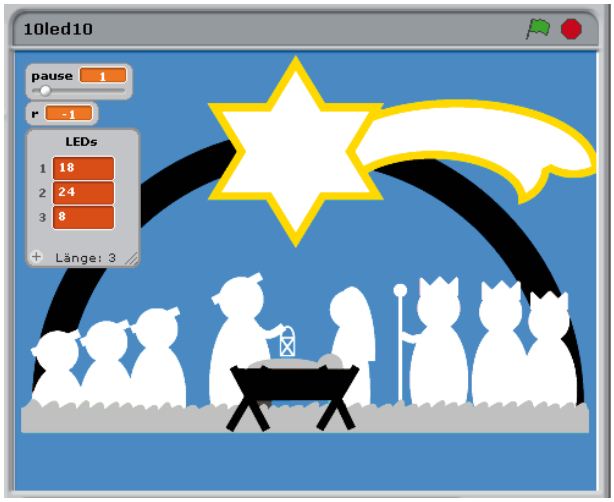
The programme `10led10` makes the three LEDs light up as running lights.

Create a list with the name **LEDs** for this programme. To do this, click on **New List** in the block pallet **Variables**. Multiple values can be saved in such lists and access via their position within the list.

Additionally, three normal variables are required:

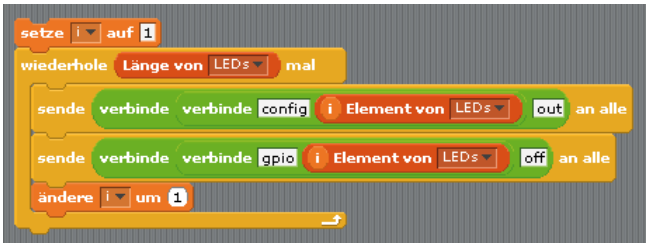
- **i** – a simple counter used in several locations in the programme.
- **r** – dictates the direction of the running light.
- **pause** – defines for how long an LED is lit as done is earlier programmes.

The variable **pause** can be changed interactively on the Scratch stage, the variable **r** is displayed but cannot be changed. The counter **i** is not of interest to the user and is not displayed. The list can be seen on the stage. However, the values never change so you can switch it off, as well.



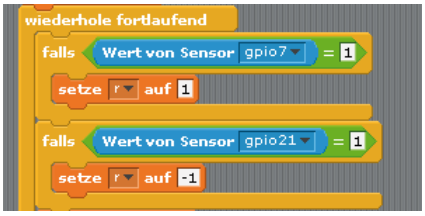
In a first step, all elements of the list are removed, if values are still present from running the programme previously. Afterwards, the numbers of the GPIO pins used for the three LEDs are saved in the first three positions in the list.

A loop runs three times accessing the three GPIO pins for the LEDs and switching them off. Using the block **Length of ...** instead of the number 3 the programme can easily be adjusted for longer lists.



The variable **i** represents a number in the range **1...**. Therefore, the block **i element of LEDs** indicates the number of the corresponding GPIO pin. Combinations of **connect ...** blocks will initialise the GPIO pins, e.g., **config18out** and **gpio18off**.

After initialising the two GPIO pins for the buttons as inputs, an endless loop starts running, which first queries the two buttons. Depending on which button is pressed, the direction of the running light is set to **1** or **-1** in the variable **r**.



Depending on the direction, the counter **i** naming the LED to light up first is increased or decreased.

This counter must not exceed the limit values 3 and 1. If the variable **i** is longer than the length of the list after increasing its value, it is reset to **1**. After the last LED lit up, the first one will light up again. In the same way, **i** is set to the number of the last LED if it was decreased to below 1.

After the programme has determined which LED is to light up, it will be switched on and off again after the defined pause using a combination of **Connect ...** blocks. The endless loop, then starts again.

11. Day

11. Day

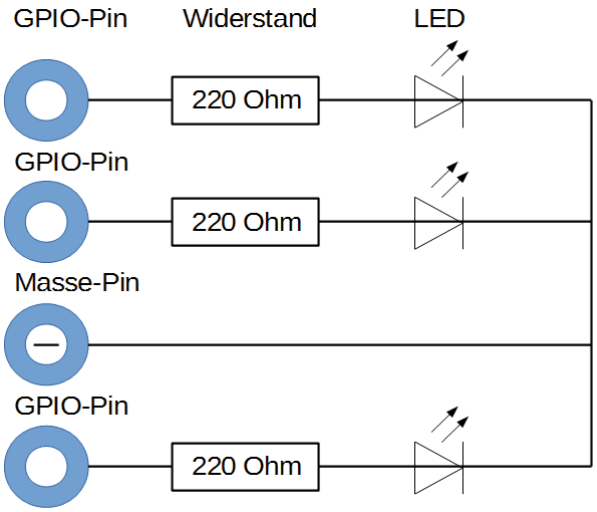
Today in the advent calendar

- 1 RGB LED with series resistor

Connection pins of an RGB LED.

RGB LEDs

A normal LED always only lights in one colour. The RGB LEDs in the advent calendar can glow in various colours as selected. In principle, RGB LEDs consist of three differently coloured LEDs installed in a transparent housing. Each of these three LEDs has its own anode, which is connected with one GPIO pin. There is only one cathode that is connected with the ground wire. That is why an RGB LED has four connection wires.



Internal circuit diagramme for an RGB LED with three series resistors.

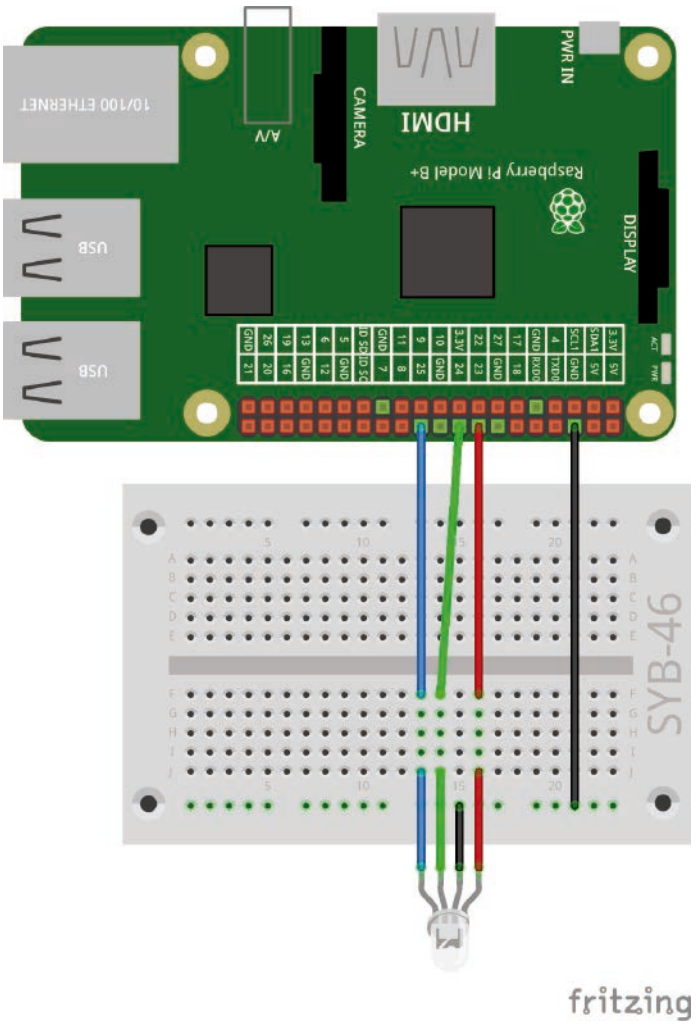
The connection wires of RGB LEDs have different lengths so that they can be distinguished clearly. In contrast to normal LEDs, the cathode wire is the longest.

RGB LEDs work like three individual LEDs and, therefore, require three series resistors. They are also installed in the RGB LEDs contained in the advent calendar.

Play of RGB colours

The experiment of Day 11 makes an RGB LED light up automatically in different colours in succession.

Components: 1 patch panel SYB-46; 1 RGB LED with series resistor; 4 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs)



RGB LED at the GPIO pins 23, 24, 25.

The programme

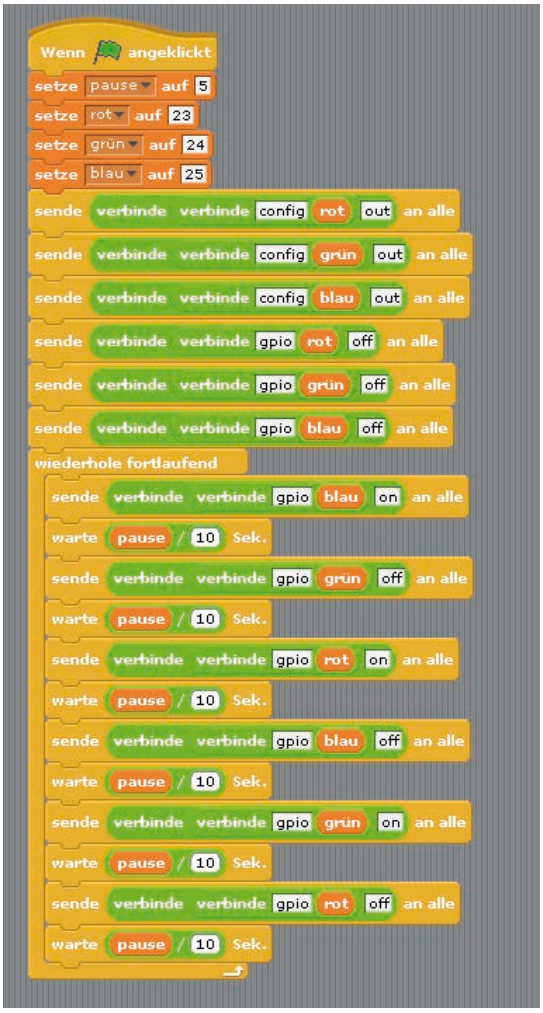
The programme `11rgbled01` switches different colours of the three colour components of the RGB LED on and off in succession. Since two colours are switched on for a duration, the RGB LED lights up the three basic colours alternately and in different mixed colours.

The time between the lighting of the individual colours and, therefore, the flashing frequency of the RGB LED is controlled with the variable **pause** again, which is shown with a controller on the Scratch stage.

After initialising the three used GPIO outputs, different pins are switched on and off in succession. The waiting time between switches can be changed interactively by the user using the controller on the Scratch stage.

Using GPIO numbers in the programme is quite confusing and increases the chance of errors when changing the programme. A second version `11rgbled02` of the programme works exactly in the same way but uses variable names for the GPIO pins. This makes the programme longer, but more structured.

The programme `11rgbled01` controls an RGB LED.



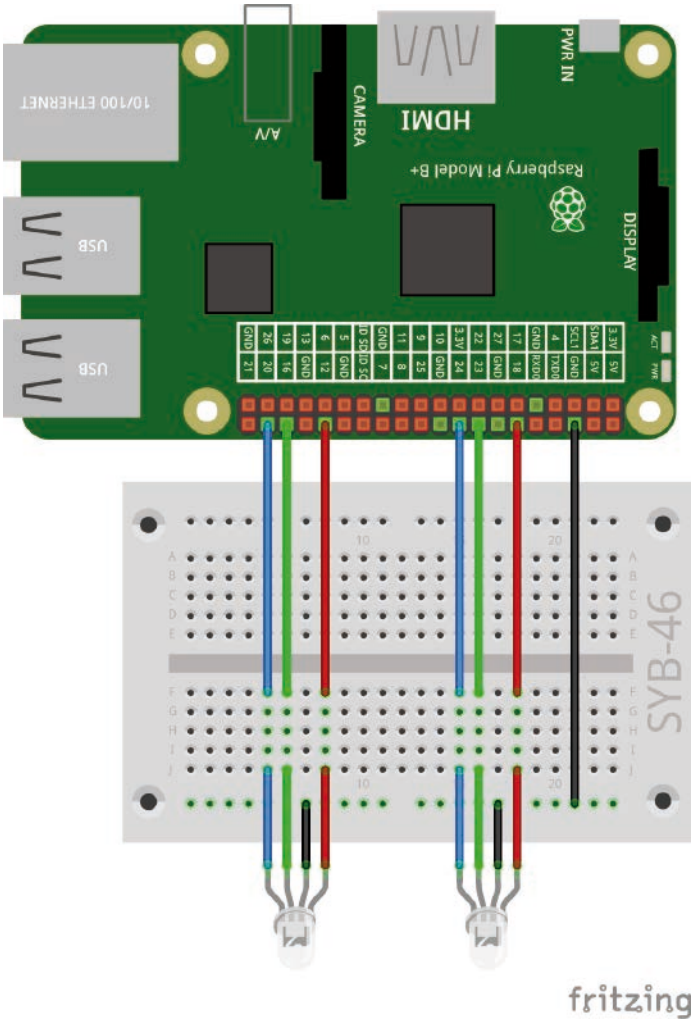
The programme `11rgbled02` uses variables for the GPIO pin numbers.

12. Day

12. Day

Today in the advent calendar

- 1 RGB LED with series resistor



Two RGB LEDs on the manger.

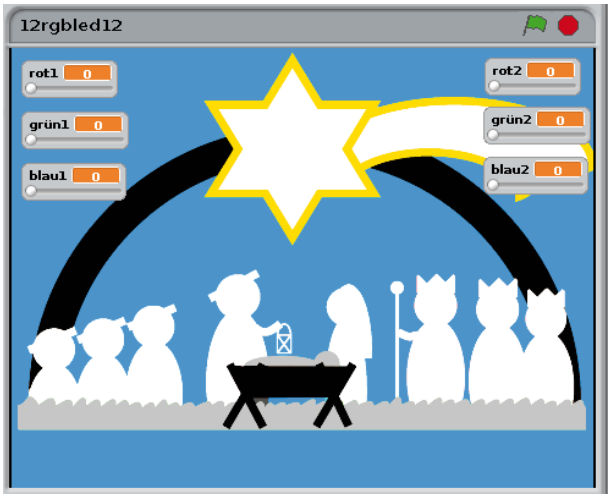
Mixing colours with RGB LEDs

The experiment of Day 12 mixes different colours of two RGB LEDs using PWM.

Components: 1 patch panel SYB-46; 2 RGB LEDs with series resistor; 7 short GPIO connection cables (Raspberry Pi – patch panel); 8 long GPIO connection cables (patch panel – LEDs)

The programme

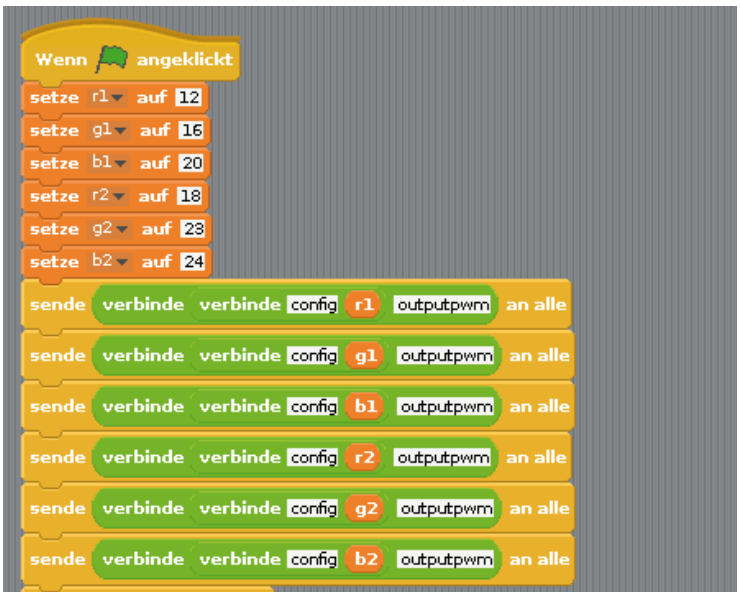
Each colour component of an RGB LED can be dimmed with PWM. This makes it possible to mix any colour. The programme `12rgbled12` contains six variables that can be set by the controller and control the three colour components of the RGB LEDs.

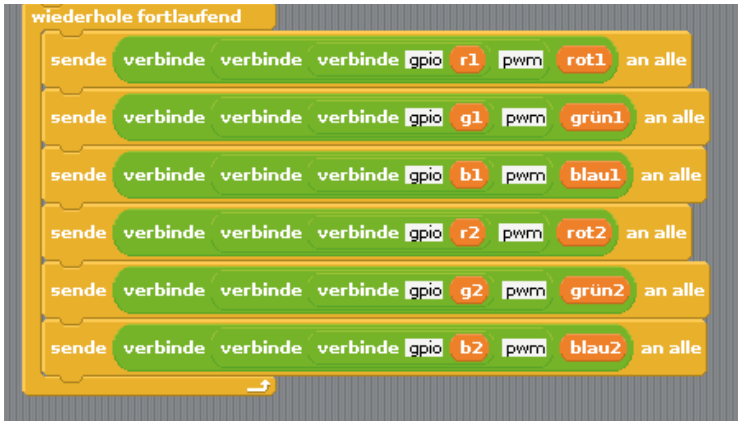


Six adjustable variables mix colours in the RGB LEDs.

Clicking on the green flag saves the pin numbers for the two RGB LEDs in the variables **r1**, **g1**, **b1** and **r2**, **g2**, **b2**; these pins are then set up as PWM outputs.

After initialising, an endless loop continuously queries the set values of the variables **red1**, **green1**, **blue1** and **red2**, **green2**, **blue2** and sets the PWM signals for the RGB LEDs correspondingly.





The programme `12rgbled12` mixes different colours in two RGB LEDs using PWM.

13. Day

Today in the advent calendar

- 1 portion modelling clay
- 1 20 Mohm resistor (red-black-blue)

Resistor

Resistors are used to limit the current in sensitive electronic components and as series resistors for LEDs. The unit for resistors is ohms. 1,000 ohms are one kilo-ohm, short kohm. 1,000 kohm are one mega-ohm, short Mohm. The omega character Ω is often used to denote the unit ohm.

The coloured rings on the resistors indicate the resistor value. With a bit of practice, they are easier to read than tiny numbers, which can only be found on very old resistors.

Most resistors have four of these coloured rings. The first two rings denote the numerical digits, the third ring a multiplier and the fourth the tolerance. This tolerance ring is mostly of gold or silver colour – colours that are not used for the first rings. This makes the reading direction clear. The tolerance value is hardly of importance in digital electronics. The table shows the meaning of the coloured ring on resistors.

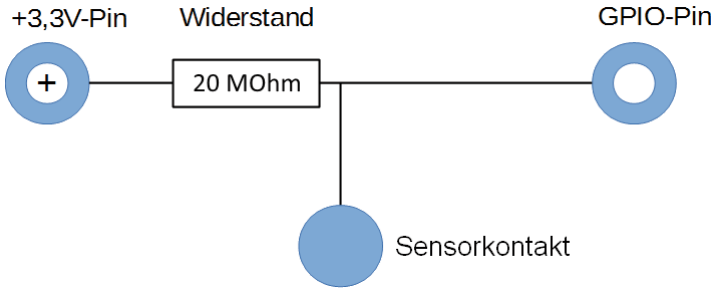
Colour	Resistor value in ohms			
	1st ring (tens)	2nd ring (ones)	3rd ring (multiplier)	4th ring (tolerance)
Silver			10^{-2} = 0,01	±10 %
Gold			10^{-1} = 0,1	±5 %
Black		0	10^{0} = 1	
Brown	1	1	10^{1} = 10	±1 %
Red	2	2	10^{2} = 100	±2 %
Orange	3	3	10^{3} = 1.000	
Yellow	4	4	10^{4} = 10.000	
Green	5	5	10^{5} = 100.000	±0,5 %
Blue	6	6	10^{6} = 1.000.000	±0,25 %
Purple	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	

The orientation in which a resistor is installed is not important. However, the orientation of LEDs is very important during installation.

Sensor contact made with modelling clay

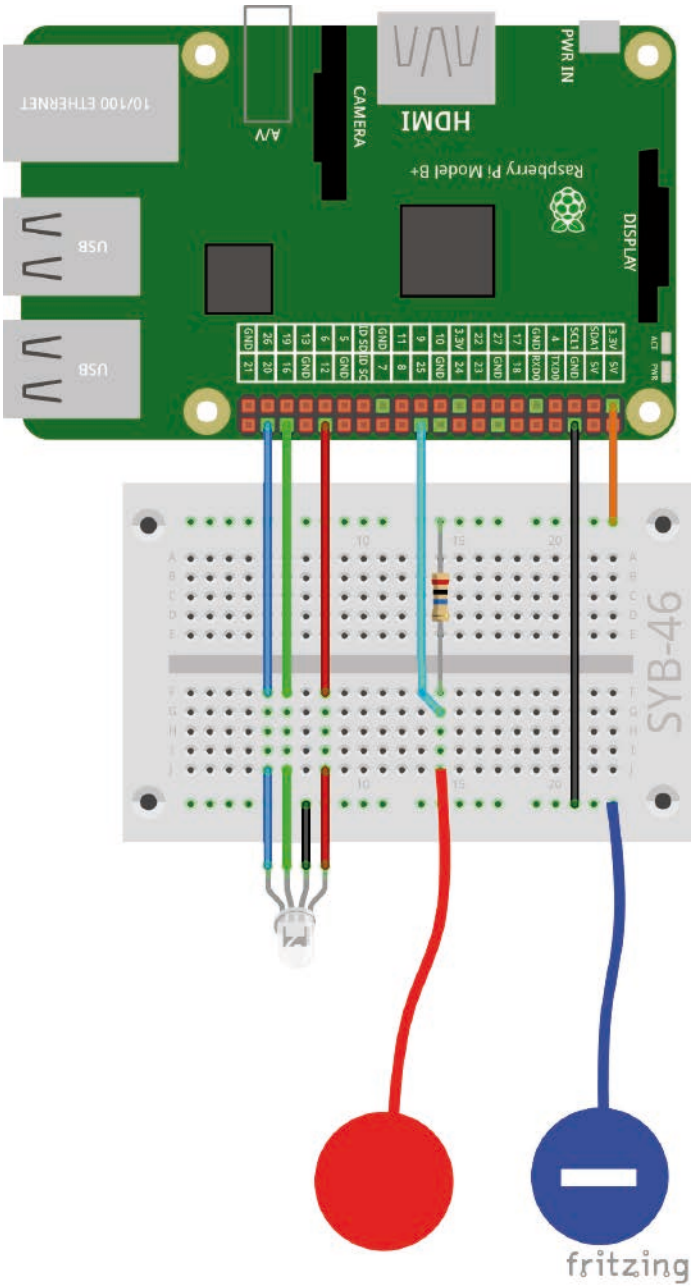
Traffic lights, door openers, light switches and automats these days are often controlled with sensor contacts, which only need to be touched. Buttons that actually need to be pressed are becoming rarer. The experiment of Day 13 controls an RGB LED with a simple sensor contact.

The GPIO pin defined as input is connected to a high-impedance resistor (20 Mohms) with +3.3 V so that a weak but clearly a **High** defined signal is present. Unless a person levitates above the ground, they are always grounded and provide a **Low**-level signal via the electrically conducting skin. If a person touches a sensor contact, the weak **High** signal is superimposed by the clearly stronger **Low** signal of the Hand and provides a **Low** signal at the GPIO pin. The pull-down resistor at the GPIO input must be switched off.



The actual magnitude of impedance between the hand and the ground is dependent on many things such as shoes or the type of floor. The connection to the ground is best when as a person stands barefoot on grass, but a stone floor also works well. Wooden floors are stronger insulators and synthetic carpets are often even charged positively. If a sensor contact does not work, the circuit has a second modelling clay contact, which is connected to the ground strip of the patch panel. Touch this contact and the actual sensor at the same time. This will definitely create a ground connection.

Modelling clay conducts electricity, approximately as well as human skin. It can be shaped into any form and a modelling clay contact is more pleasant to touch than a simple piece of wire. The area between the hand and the contact is considerably larger. This prevents loose contacts fairly well. Insert a piece of stripped connecting wire into a bit of clay. Insert the other end of the wire into the patch panel.



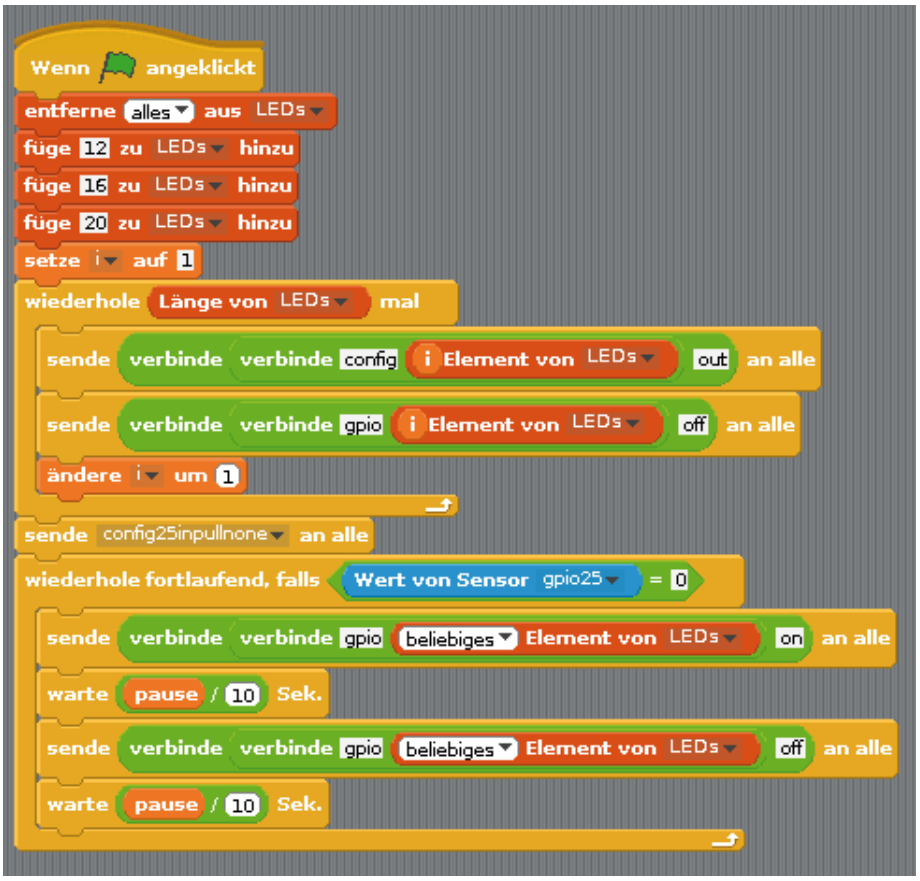
The modelling clay contact controls the RGB LED.

The RGB LED flashes randomly.
In the experiment of Day 13, an RGB LED flashes in random colours as long as the sensor contact is touched.

Components: 1 patch panel SYB-46; 1 RGB LED with series resistor; 1 20-Mohm resistor; 6 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs); 2 modelling clay contacts

The programme

The programme `13rgbled` uses a list for the three GPIO pins of the RGB LED. A loop initialises the pins as input and switches them all off. The block **send config25inpullnone** initialises GPIO pin 25 for the sensor contact as input without pull-down resistor.



The programme `13rgbled13` controls an RGB LED server via a sensor contact.

The RGB LED should only flash if the sensor contact is touched. This is done by a loop of type **repeat continuously if ...** If the sensor contact is touched, it will be connected to the ground and the signal at the GPIO input is set to 0. The condition inside the loop again uses the block **Value of sensor ...** Start the incomplete programme once by clicking on the green flag in order to initialise the GPIO input so that it appears in the selection list of the sensors.

With each iteration of the loop, a randomly selected colour component of the RGB LED is switched on. To do this, a block **any element of LEDs** is used.

How are random numbers generated?
The general opinion is that nothing can happen randomly in a programme. So how can a programme be able to generate random numbers? When a large prime number is divided by an arbitrary number, the xth decimal place of the resulting number will be almost impossible to predict. They also change without much regularity when the divisor is increased frequently. This result seems random, but it can always be reproduced by an identical programme or by executing the same programme several times. However, if you take a number created by these digits and divide it again by another number corresponding to the current seconds of the time or the content of an arbitrary memory space of the computer, you will get a result that cannot be reproduced and, therefore, is called a random number.

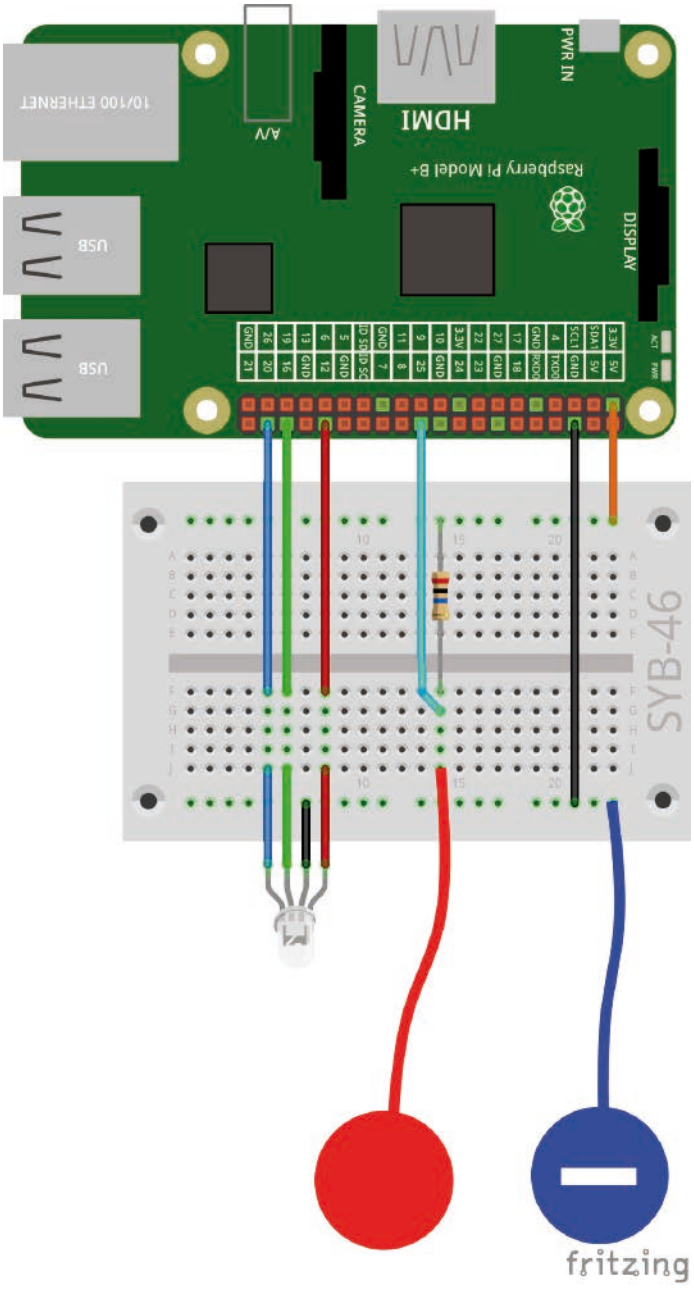
After a pause set again with a controller, a randomly selected colour component of the RGB LED is switched off. Due to the used algorithm, it is possible that the RGB LED switches off entirely for a short period of time or that the same colour is shown twice in succession.

14. Day

14. Day

Today in the advent calendar

- 1 portion modelling clay



The modelling clay contact controls the RGB LED.

With the second portion of clay, you can build the two sensor contacts at the GPIO pin and the ground pin in different colours.

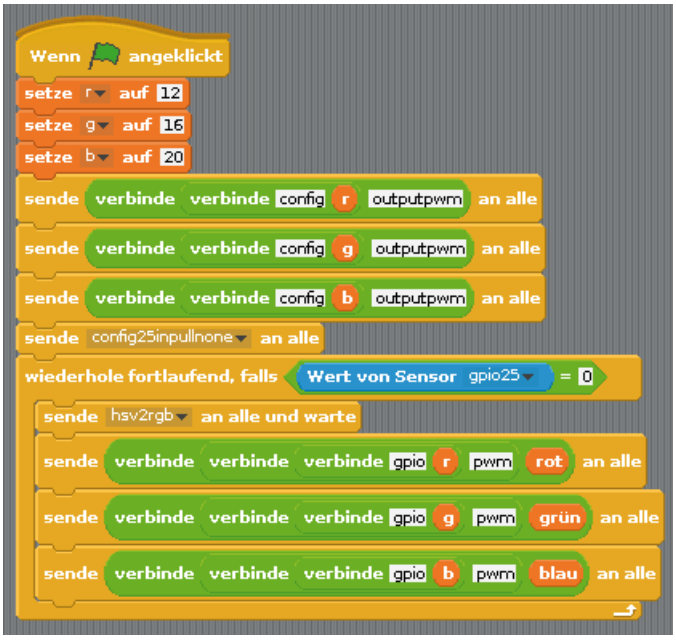
RGB LED light effects

The programme of Day 14 makes the RGB LED light up in any colour of the rainbow if the sensor contact is touched. The colour is selected with on single controller instead of three RGB values. If the sensor is released, the colour shown last will stay on.

Components: 1 patch panel SYB-46; 1 RGB LED with series resistor; 1 220-Mohm resistor; 6 short GPIO connection cables (Raspberry Pi – patch panel); 4 long GPIO connection cables (patch panel – LEDs); 2 modelling clay contacts

The programme

The programme `14rgbled14` consists of two independent programme blocks. The block started by clicking on the green flag is largely similar to the programmes known up to now. The three pins of the RGB LED are initialised as PWM outputs, pin 25 as input without pull-down resistor for the sensor contact.

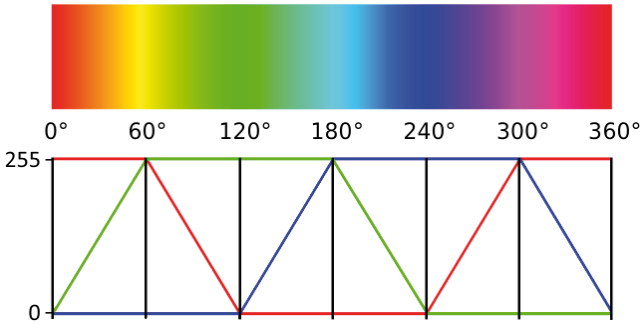


The programme `14rgbled14` makes the RGB LED flash in all colours of the rainbow.

An endless loop checks if the sensor contact is touched. If this is the case, a block **send ... to all and wait** sends the message **hsv2rgb**. In this way, sub-programmes can be built in Scratch, which are called from the main programme. The main programme waited until the sub-programme has run and then sets the GPIO pins of the RGB LED to the computed PWM values.

HSV and RGB colour systems
The RGB colour system has been used in all programmes so far and describes colours in three components – red, green and blue – that are mixed together. For humans, it is relatively difficult to imagine a mixed colour. In contrast, the HSV colour system describes colours with the values H = Hue, S = Saturation and V = Value (brightness).
By simply changing the H-value, all colours of the colour spectrum can be described in full intensity, if the other two values are set to maximum.

The message **hsv2rgb** starts a programme block, which computes the three colour components R, G and B from the set H-value. The H-value can assume values between 0 and 360 corresponding to the degree values of a colour circle. The values S = Saturation and V = Value (brightness) are automatically set to maximum.



The diagramme shows how the H-value of an HSC colour is converted into RGB values.

Sub-programme to convert HSV to RGB values.

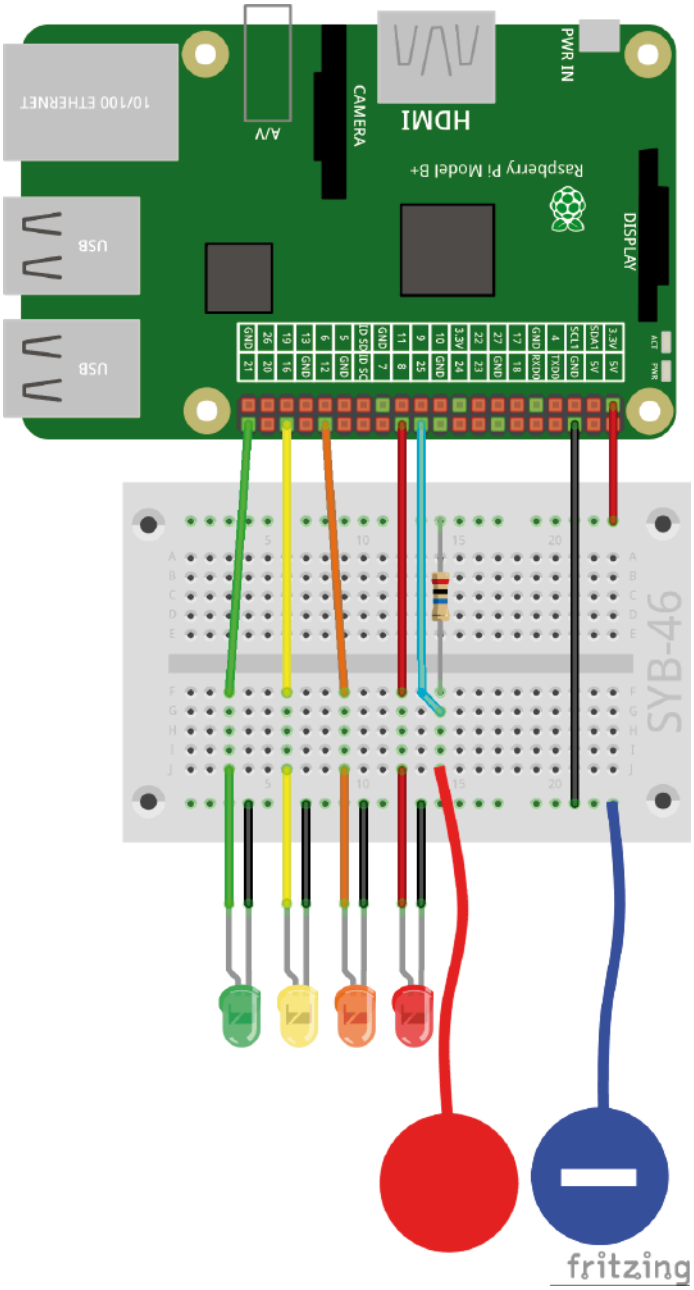
The block **if I receive ...** starts a sub-programme, if a certain message was received. This message can be sent by a block **send ... to all and** or **send ... to all and wait**.

15. Day

15. Day

Today in the advent calendar

- 1 orange LED with series resistor



Modelling clay contact controls running light.

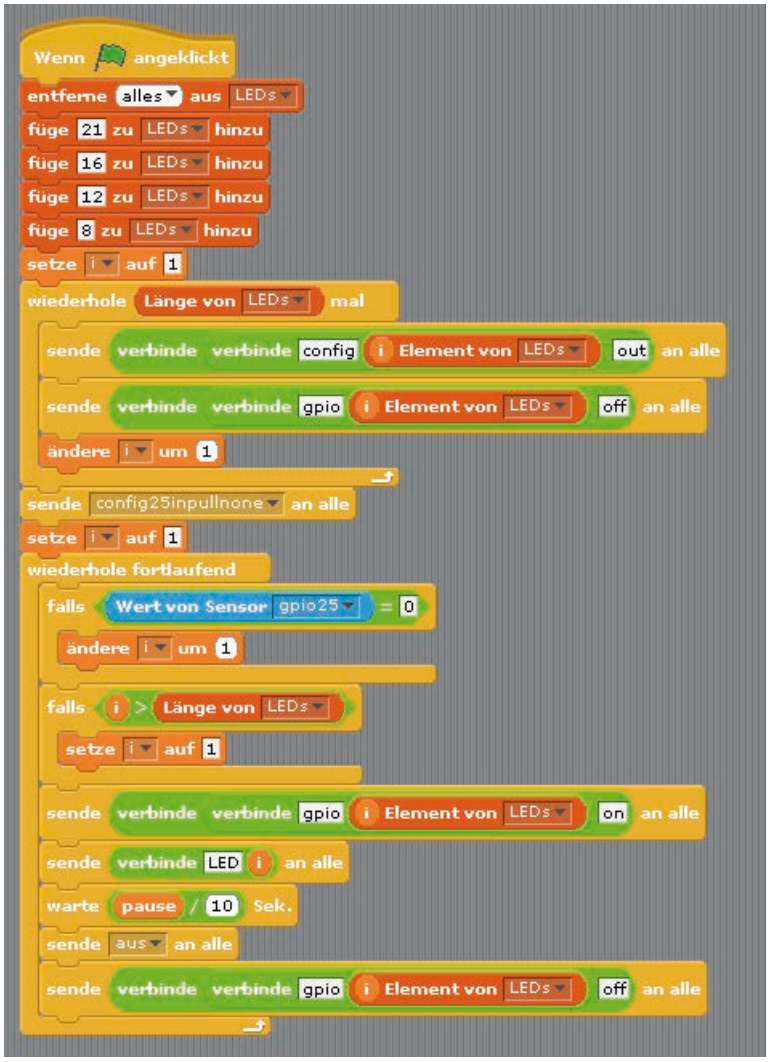
Controlling a running light with modelling clay contact

The experiment of Day 15 is another running light, which this time is controlled with a sensor contact made with clay. In addition to the real LEDs on the Christmas manger, colourful symbols will light up on the Scratch stage. As long as the sensor contact is touching, the LEDs flash alternately as running light. If the sensor contact is released, the LED switched on last, remains on.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 20-Mohm resistor; 7 short GPIO connection cables (Raspberry Pi – patch panel); 8 long GPIO connection cables (patch panel – LEDs); 2 modelling clay contacts

The programme

The programme `15rgbled15` consists of several independent programme blocks. The programme block of the stage is started by clicking the green flag, it creates a list with the GPIO pins of the four LEDs, initialises them as outputs and pin 25 as an input without a pull-down resistor.



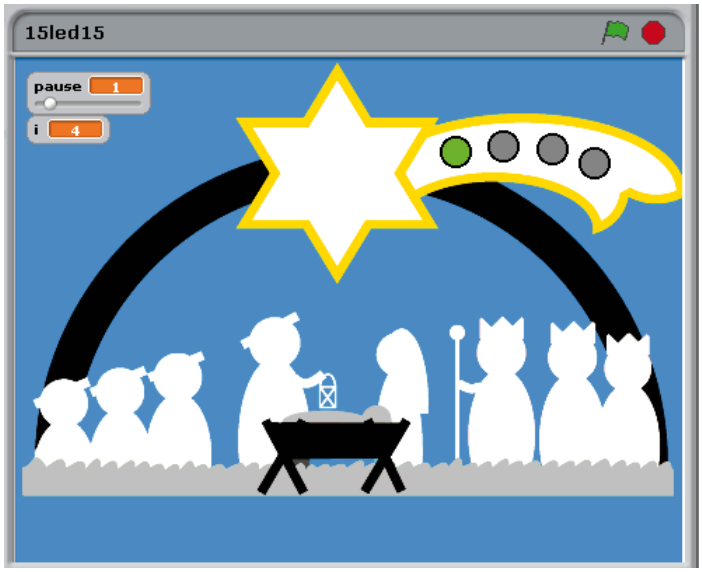
The programme block of the stage contains the main loop of the programme.

The endless loop first queries if the sensor contact was touched. If this is the case, the counter **i** is incremented by 1 to make the next LED of the list flash.

If **i** becomes larger than the length of the list, **i** is reset to **1** and the running light starts again with the first LED.

The GPIO pin corresponding to the variable **i** in the list is now switched on. After that, a block **connect ... ...** creates a message containing the word **LED** and the current number of the variable **i**: **LED1**, **LED2**, **LED3** or **LED4**. This message is sent in order to switch on the LED symbol on the Scratch stage.

After an adjustable pause, the message **off** is sent, which switches all LED symbols on the Scratch stage off. In contrast to switching on, it is not necessary to specify an LED number here. The programme simply switches all LED symbols off, independent of whether they were switched on before or not.



The four LEDs on the Scratch stage.

After that, the currently lit LED is switched off via its GPIO pin before the next iteration of the loop begins.

The four LED symbols on the tail of the star are the four individual Scratch objects. Load the picture `circle_green.png` into the scene as new object. As soon as the object is completed, the other objects will be duplicated from it.

Highlight the new object in the object list and copy a second costume on the tab **Costumes**. Fill this circle with grey in the drawing programme.

The costumes of the green LED symbol.

The programme block of the green LED symbol.

Rename the object to **green** and the two costumes to **on** and **off**. Subsequently, create two programme blocks for this object on the tab **Scripts**.

If the object receives the message **LED1**, the costume **on** is activated. The symbol lights up in green.

If the object receives the message **off**, the costume **off** is activated. The symbol appears in grey.

Duplicate the object three times by right-clicking on the object pallet. Rename the new objects to **yellow**, **orange** and **red** and move them to the corresponding positions on the tail of the star.

For each object, click the button **Edit** in the costume **on** and fill the circle with yellow, orange and red. Then change the message in the first block to **LED2**, **LED3** and **LED4** in the tab **Scripts**.

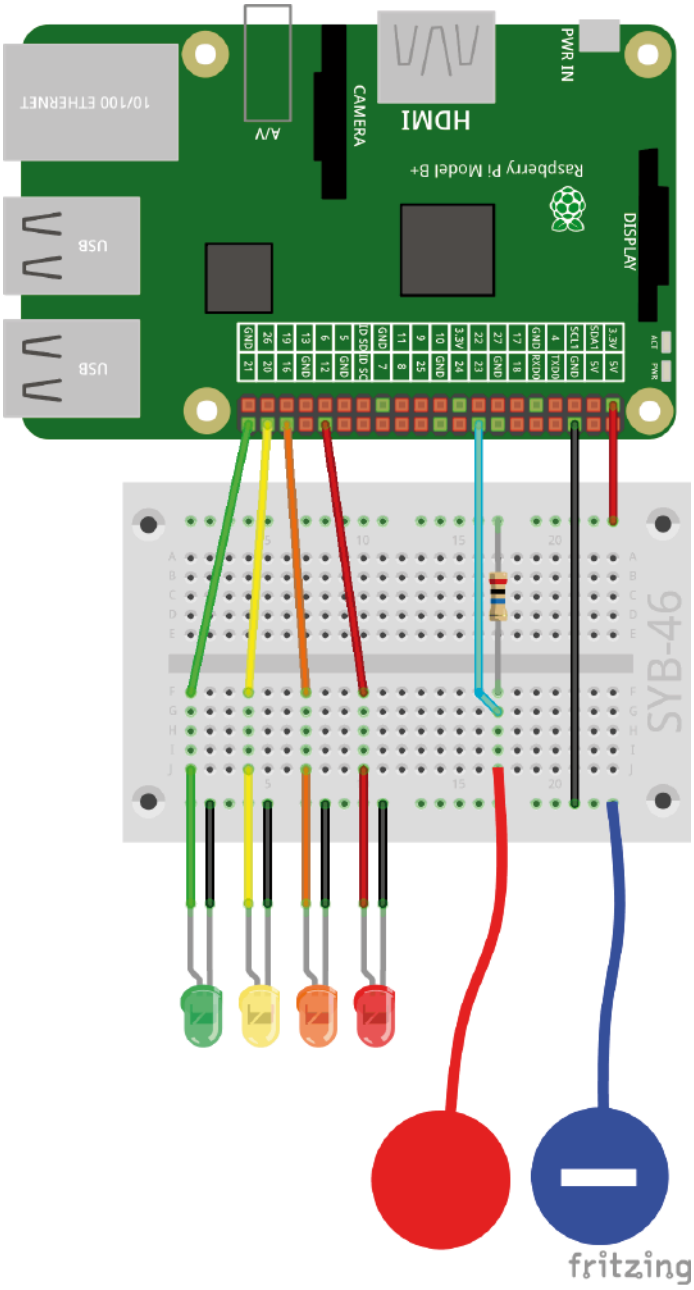
Start the programme by clicking on the green flag and touch the sensor contact. The running light starts.

16. Day

16. Day

Today in the advent calendar

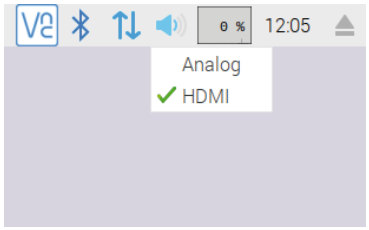
- 4 GPIO connection cables (long)



The modelling clay contact controls the LEDs.

The Raspberry Pi makes sounds

The Raspberry Pi can play music via an HDMI monitor, an external loudspeaker or headphones at the analogue 3.5 mm radio socket. For computer monitors with DVI connection connected to the HDMI output, a loudspeaker must be connected to the analogue **output** because the audio signal cannot be transferred via the DVI cable. Right-click the speaker icon in the top right to select the audio output.



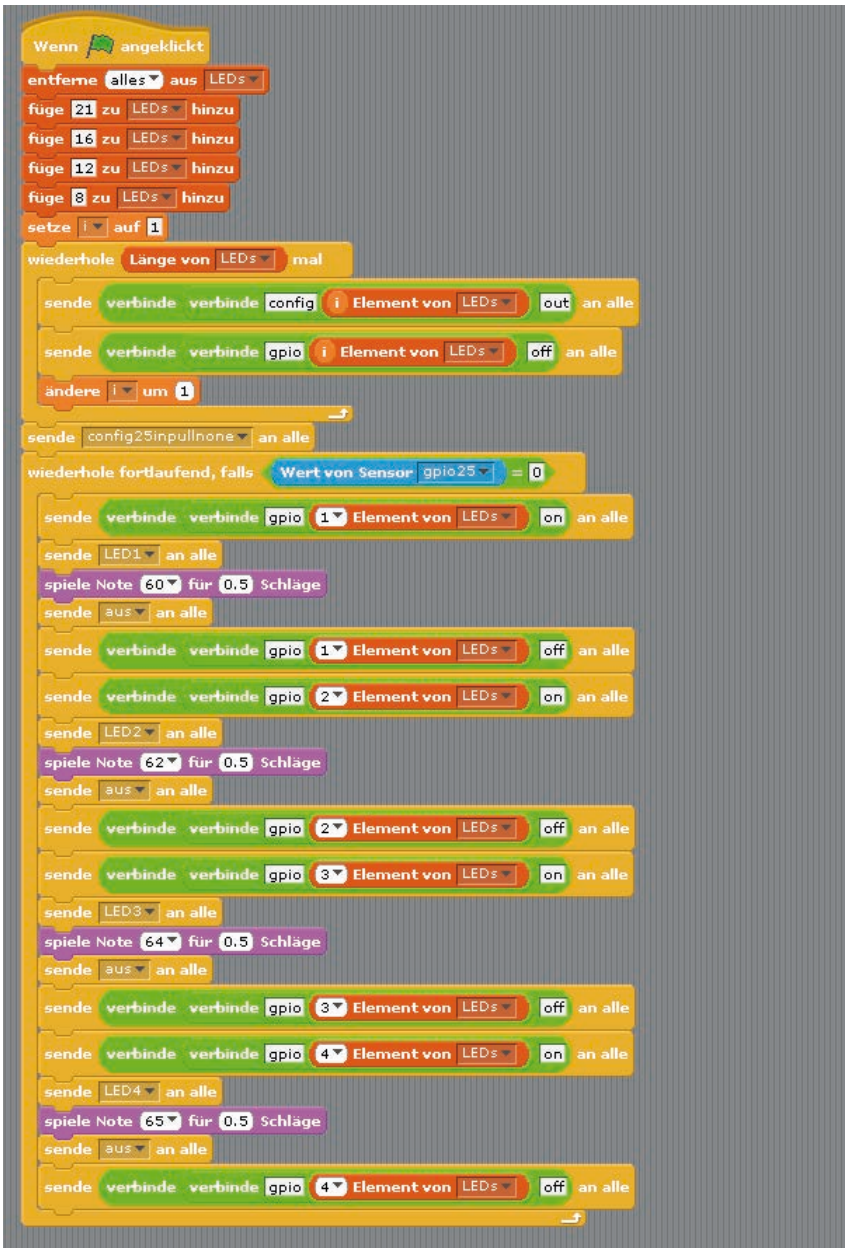
Selecting the audio output.

Every time you touch the modelling clay contact, the Raspberry Pi plays a series of notes and makes the LEDs light up successively.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 20-Mohm resistor; 7 short GPIO connection cables (Raspberry Pi – patch panel); 8 long GPIO connection cables (patch panel – LEDs); 2 modelling clay contacts

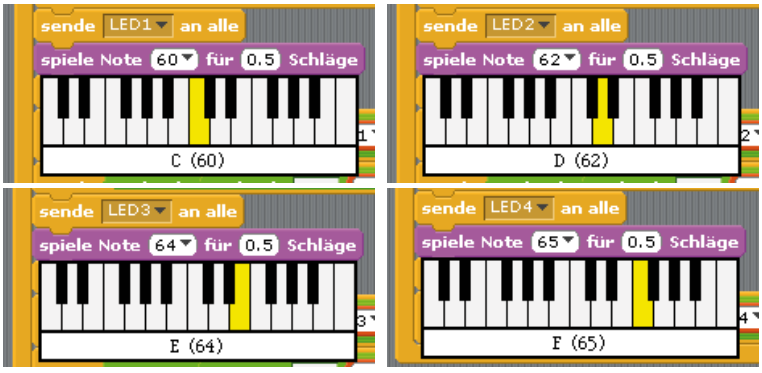
The programme

The programme `16led16` works in a similar way as the programme of the previous day and makes four LEDs flash successively with a short pause, if the modelling clay contact is touched. A note is played for each LED.



The programme `16led16` makes four LEDs flash with short pauses and plays notes alongside.

The numbers of the LEDs are again saved in a list. However, the main loop of the of the programme contains separate programme blocks for each LED that you can interactively specify the note for each individual LED via a keyboard in the block **play note ... for ... beats**. The programme uses the notes C, D, E and F.



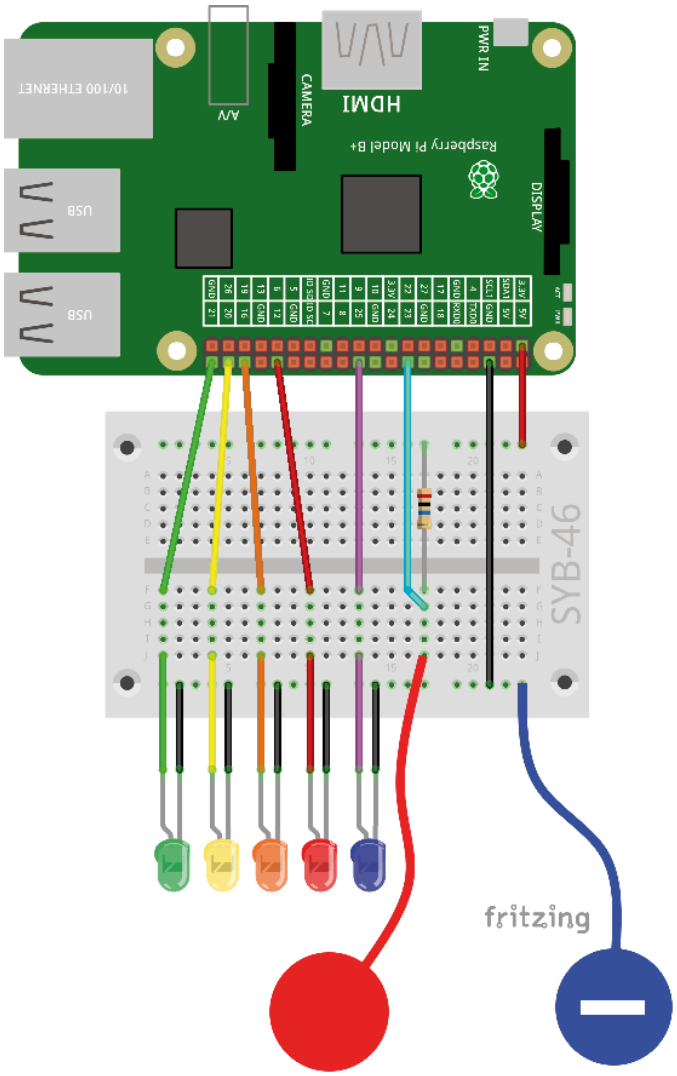
The four used notes.

17. Day

17. Day

Today in the advent calendar

- 1 purple LED with series resistor



Running light with five LEDs.

The GPIO pins are initialised in the same way as the previous programme. In every iteration, the main loop of the programme sets the counter **i** to **1** when the sensor contact is touched. A nested loop runs five times and switches on one of the LEDs and the corresponding LED symbol on the stage with every iteration. After that, a note corresponding to the number in the list **Sound** is played. This note also defines for how long the LED is lit. After that, the LED symbol and the LED are switched off, the loop counter is incremented by 1 and the next iteration is started with the next LED.

Expanded running light

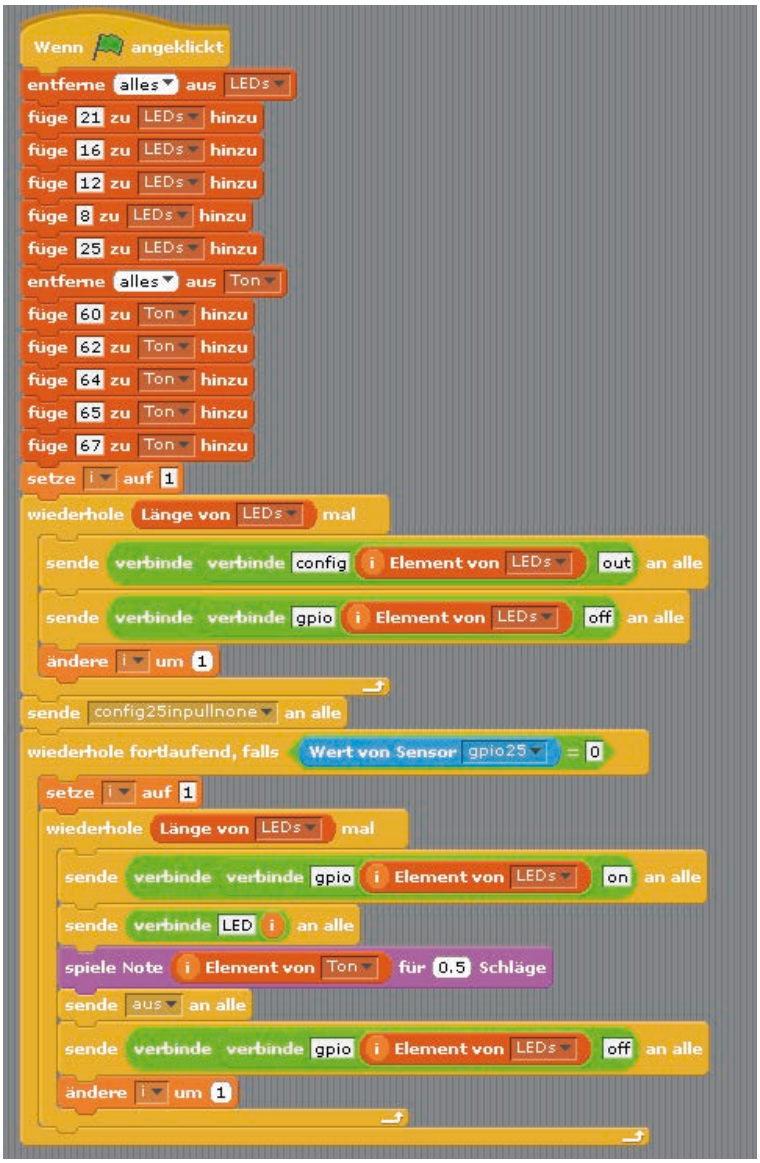
The experiment of Day 17 expands the running light with a fifth LED. A series of notes is played again.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 1 20-Mohm resistor; 8 short GPIO connection cables (Raspberry Pi – patch panel); 10 long GPIO connection cables (patch panel – LEDs); 2 modelling clay contacts

The programme

In the programme `17led17`, the notes are also played via a list, which makes the main loop of the programme considerably shorter.

In addition to the list **LEDs**, a list **Note** is created at first, which are played along the individual LEDs. This method also provides a keyboard. It is best to try the notes with a block **Play note ... for ... beats**. The programme uses the notes **C(60)**, **D(62)**, **E(64)**, **F(65)**, **G(67)**.



Running light with five LEDs and note sequence.

18. Day

Today in the advent calendar

- 3 GPIO connection cables (short)

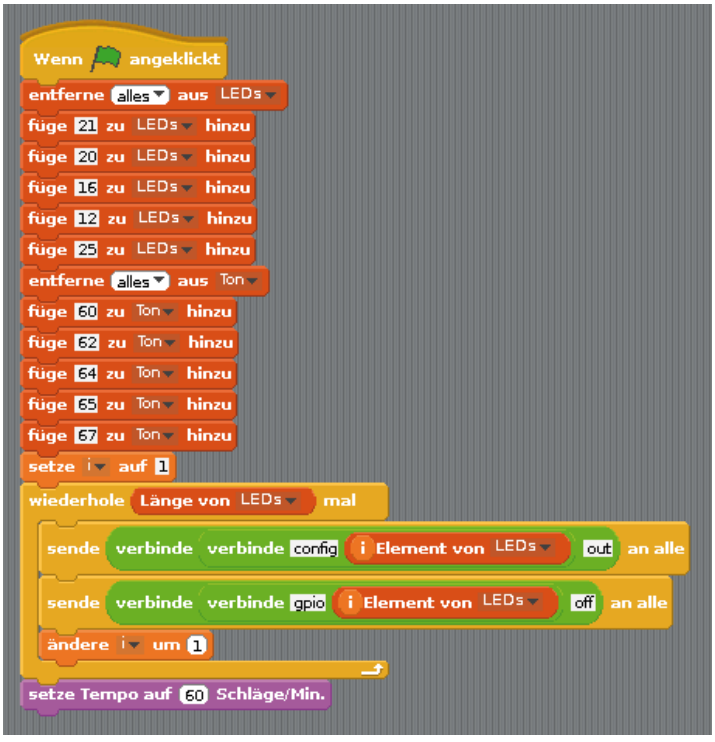
Three different running light patterns.

The programme of Day 18 shows two other light patterns in addition to the running light. To do this, the programme `18led18` contains three buttons on the Scratch stage, with which these patterns can be switched on.

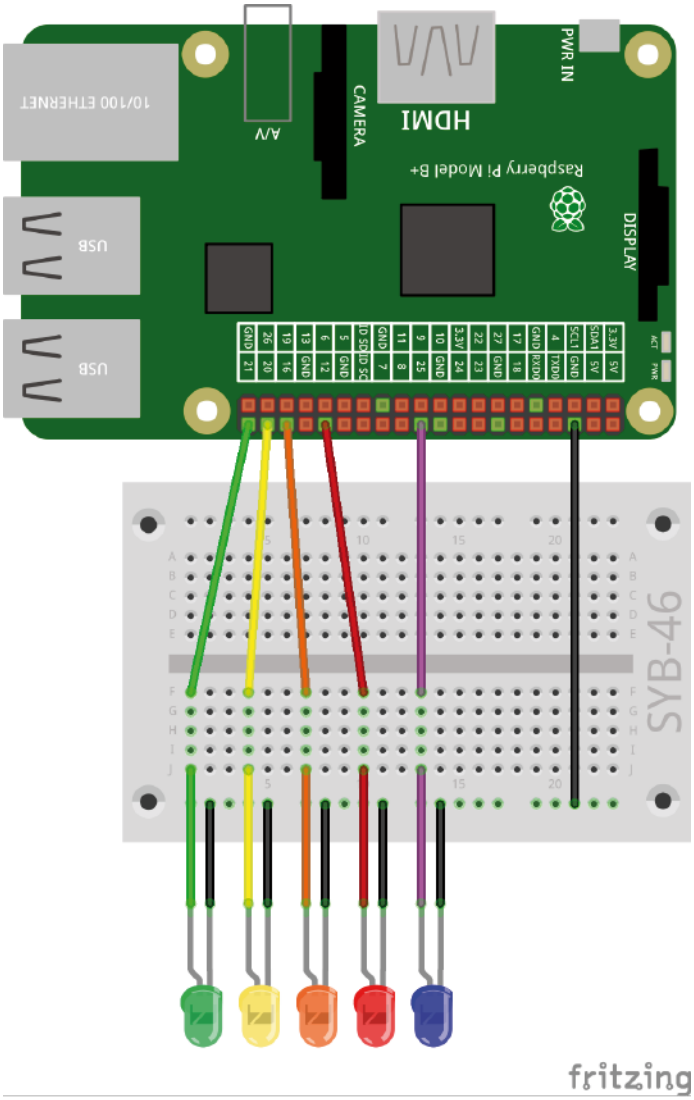
Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 6 short GPIO connection cables (Raspberry Pi – patch panel); 10 long GPIO connection cables (patch panel – LEDs)

The programme

The programme `18led18` only initialises the two lists for LEDs and sounds as well as the GPIO pins on the stage. At the end, a speed is specified for the music. Music is played with a tempo, which is characterised with so-called beats, while all other time units are given in seconds in Scratch. The block **set tempo to ... beats/min** specifies the tempo for the playback of music.

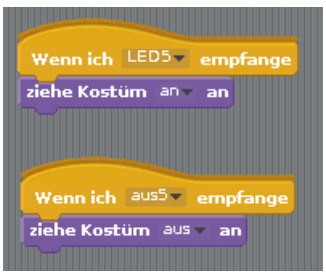


The programme blocks of the stage.



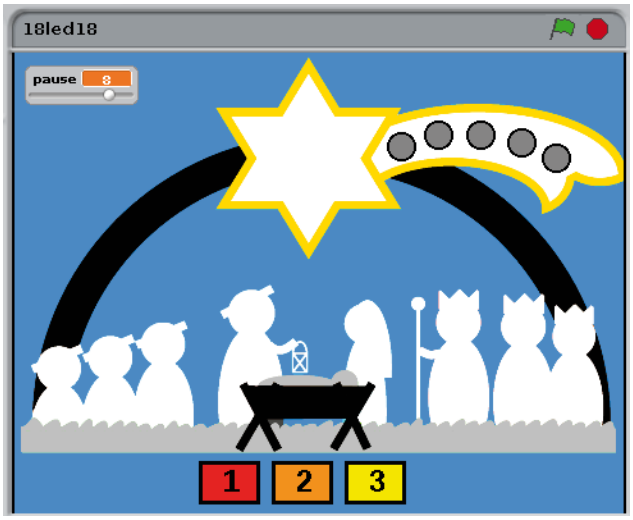
The running light is played with different patterns.

Like previous programmes, the programme contains five objects for the five LEDs. Each object reacts to its own switch-off signal (e.g. `off5` in the figure) so that the LEDs can be controlled with different patterns. However, the LEDs are not all switched off at the same time.



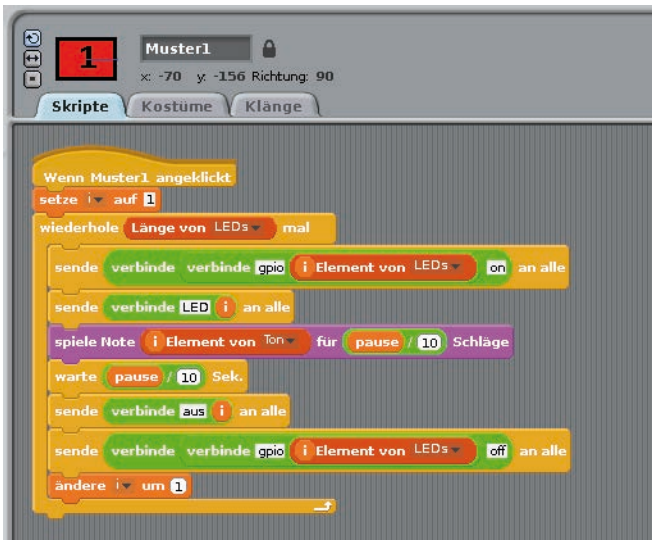
Programme blocks of the object **purple**.

In addition to the objects for the LEDs, the programme contains three buttons. They are separate object, which play one of the three pre-defined LED patterns when clicked. The time between each flash is again controlled with the interactively adjustable variable **pause**.



The three buttons toggle the flashing patterns of the LEDs.

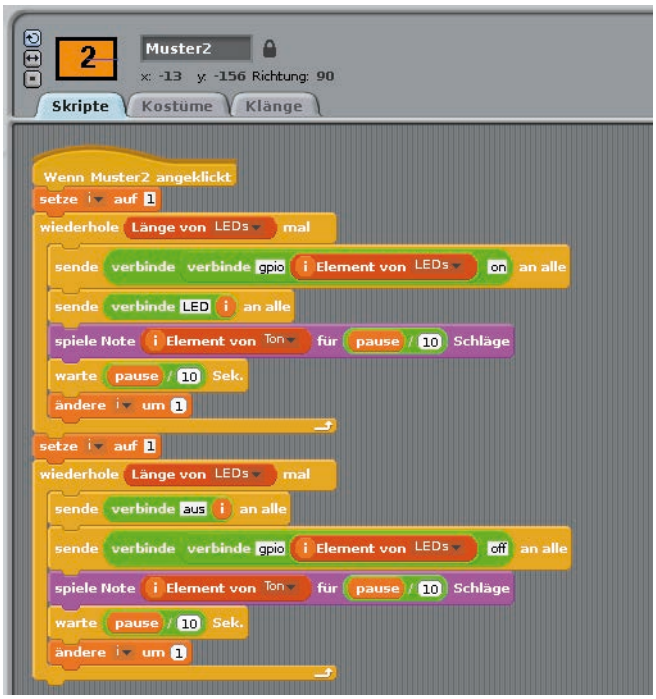
The button with number 1 starts the known running light sequence when clicked. One LED after another is switched on and off again after the specified pause.



The programme blocks for button 1.

After clicking the button, a loop runs through all elements of the list **LEDs**. First, one LED is switched on via its GPIO pin. A block **send ... to all** sends a message containing the LED number in order to switch on the corresponding LED object on the stage.

After that, the sound corresponding to the LED is played from the list **Note**. The programme waits until the sounds and flashes run synchronously. Then a block **send ... to all** sends a message containing the LED number in order to switch off the corresponding LED object again. This pattern may switch off all LED objects. This does not work for the other pattern and the switch-off message also contains the number of the LED. Finally, the "real" LED is switched off via its GPIO pin and the loop counter is incremented by 1.

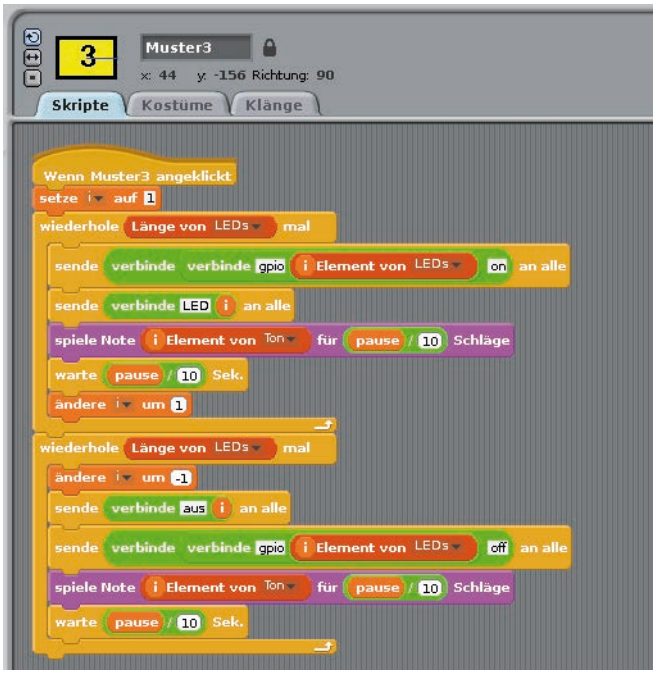


The programme blocks for button 2.

The button with the number 2 starts one LED after another when clicked and leaves them switched on. Once all five LEDs are lit, they are switched off again in the same order.

When clicking the button with the number 2, two loops run in succession. The first loop switches on an LED via its GPIO port, sends the message to switch on the LED object on the stage, plays the sound and waits.

Once the loop has run five times, the loop counter is reset to **1**. After that, the LED objects and LEDs are switched off again one after another and the sounds are played again.



The programme blocks for button 3.

The button with the number 3 starts one LED after another when clicked and leaves them switched on. Once all five LEDs are lit, they are switched off again in reverse order.

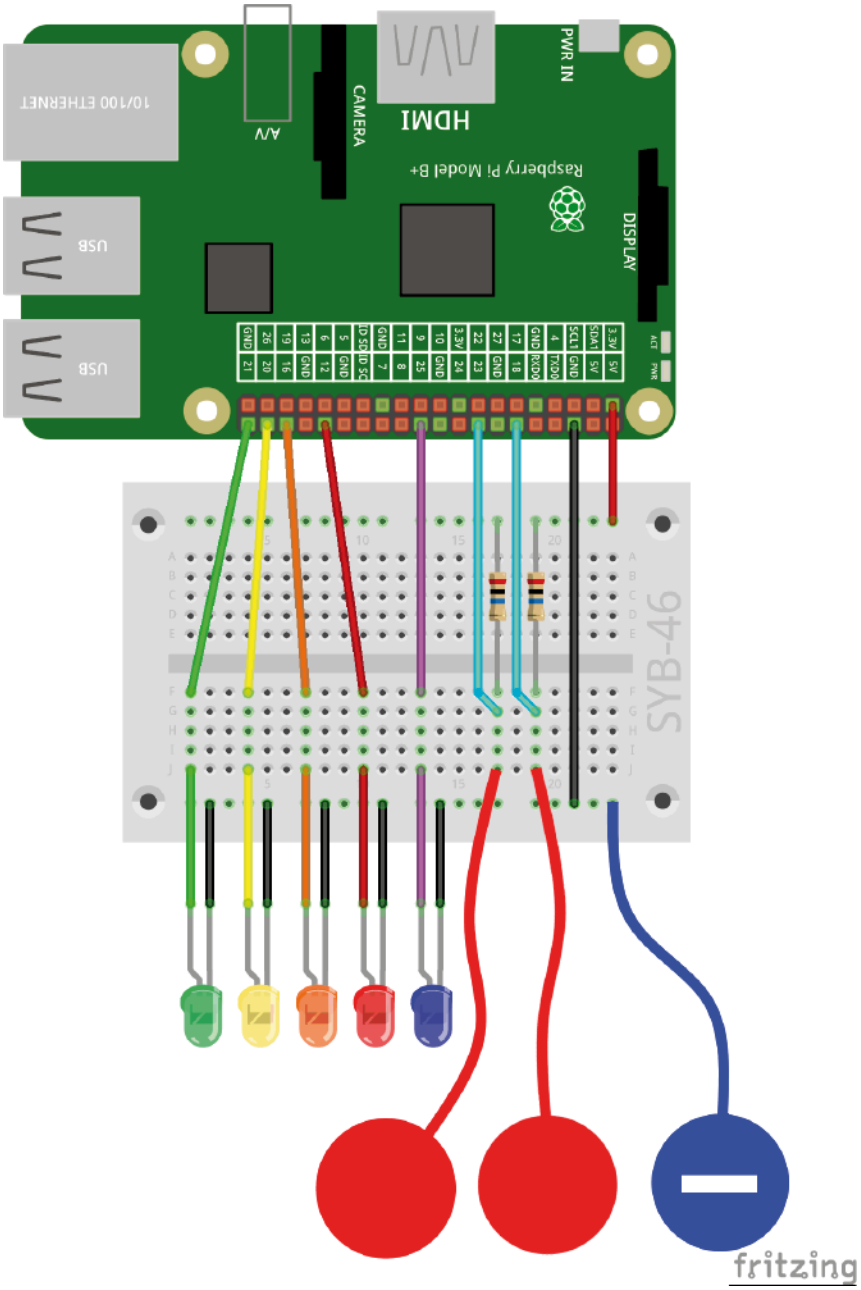
The first loop is the same as the one for the button with number 2. After five iterations, the loop counters stays at **5** and starts counting down 1 with each iteration of the loop. The switches the LEDs off in reverse order.

19. Day

19. Day

Today in the advent calendar

- 1 20 Mohm resistor (red-black-blue)



Two modelling clay contacts control the LEDs.

Running light or flashing

The experiment of Day 19 contains two modelling clay contacts and one ground contact. When touching the modelling clay contact at GPIO pin 23, a running light starts with the selected pattern. When touching the modelling clay contact at GPIO pin 18, all LEDs light up briefly once.

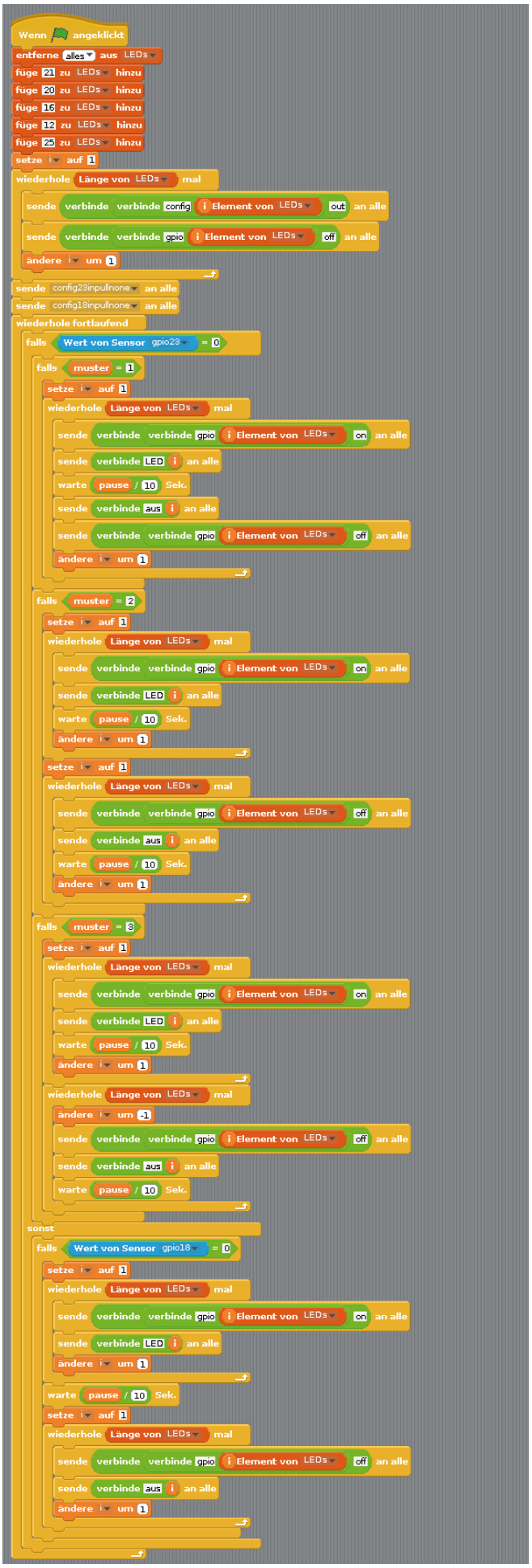
Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 2 20-Mohm resistors; 9 short GPIO connection cables (Raspberry Pi – patch panel); 10 long GPIO connection cables (patch panel – LEDs); 3 modelling clay contacts

The programme

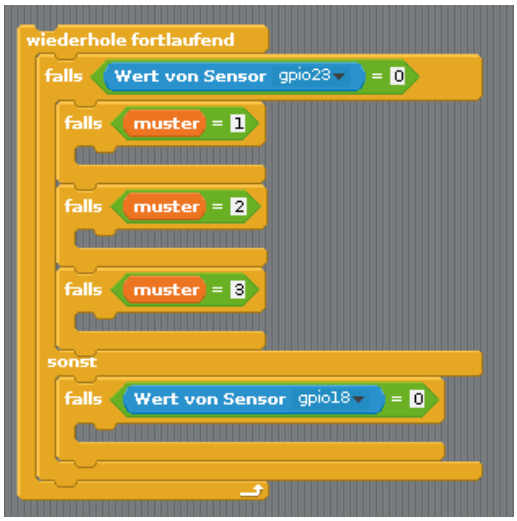
The running light in programme `19led19` is not started by clicking a button as was the case in the previous programme, but it waits until the user touches a modelling clay contact. It therefore follows a different programme logic. When clicking a button, it sends a message with the name of the pattern.

Programme block of one of the three buttons.

The main programme on the stage waits until the modelling clay contact is touched and then evaluates the selected pattern. Another GPIO pin is initialised as input without pull-down resistor for the second modelling clay contact.



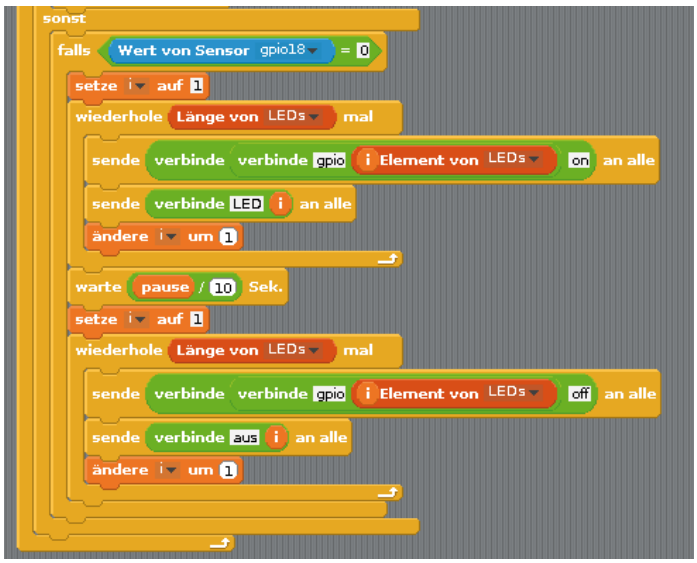
The main programme on the stage.



Overview of the structure of the main loop.

If the modelling clay contact at GPIO pin 23 is touched a running light pattern is played depending on the button clicked last. As in the previous programme, the LED objects on the stage light up as well.

In case of **else** – if the modelling clay contact was not touched, another query checks if the modelling clay contact at GPIO pin 18 was touched. In this case, all LEDs are switched on and off after a short break.



If the modelling clay contact at GPIO pin 18 is touched, all LEDs are switched on and off again after a short pause.

20. Day

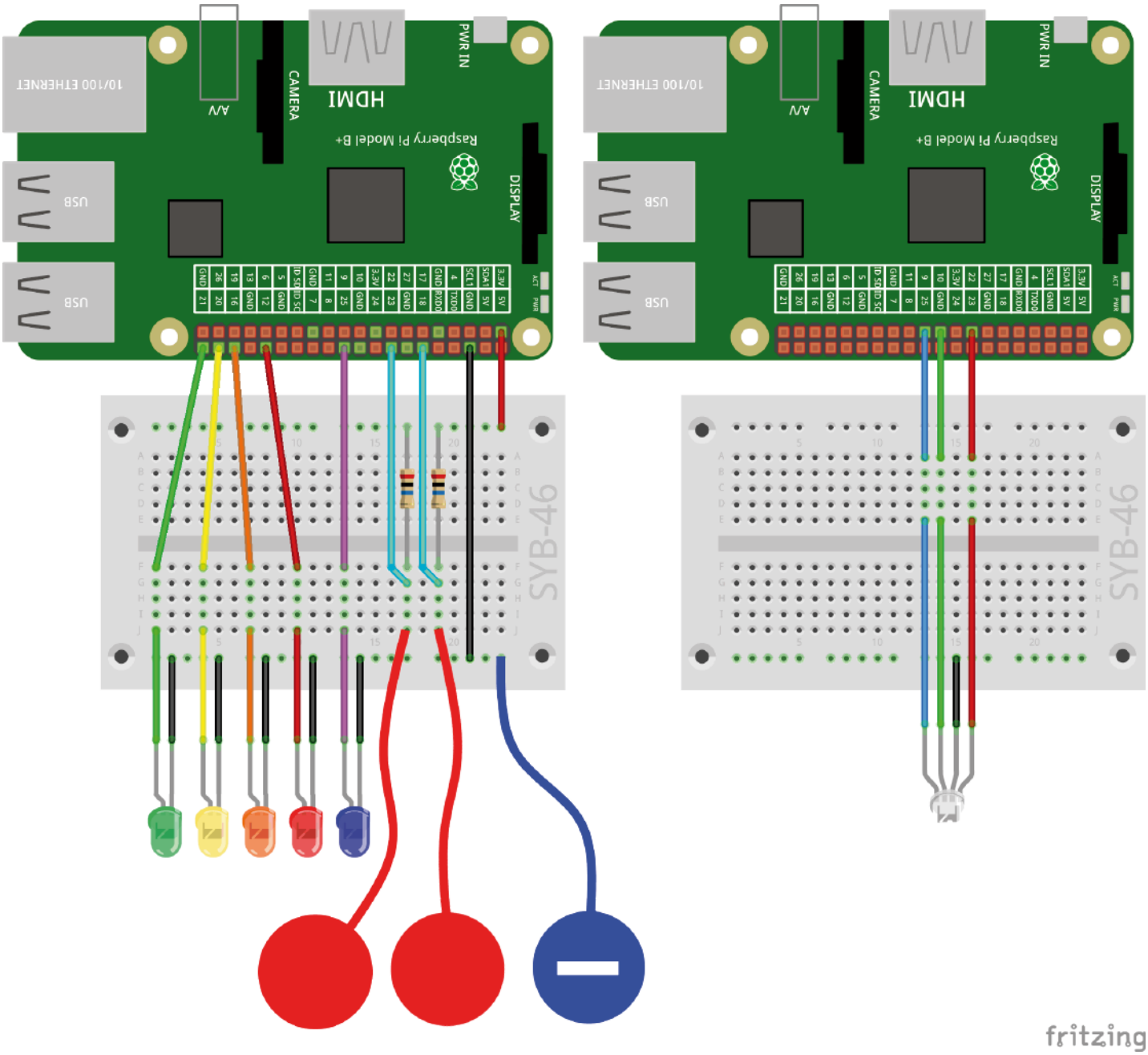
Today in the advent calendar

· 4 GPIO connection cables (long)

Controlling RGB colour plays with the modelling clay sensor

Similar to the previous programme, the programme of Day 20 shows different running light patterns when the modelling clay contact at GPIO pin 23 is touched. If the modelling clay contact at GPIO pin 18 is touched, the LED object on the star of the Scratch stage light up. The RGB LED also lights up in changing colours.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 1RGB LED with series resistor; 2 20-Mohm resistors; 12 short GPIO connection cables (Raspberry Pi - patch panel); 14 long GPIO connection cables (patch panel - LEDs); 3 modelling clay contacts



Two modelling clay contacts control the running light and on contact controls the RGB LED.

There is not much space left on the patch panel so that the circuit diagramme is shown in two parts for today and all following days. The cables of the RGB LED that are inserted into the hole in the star over the manger would cover other connection cables in the diagramme.

The programme

The programme `20rgbled20` first initialises an additional three GPIO pins as PWM outputs for the RGB LED.

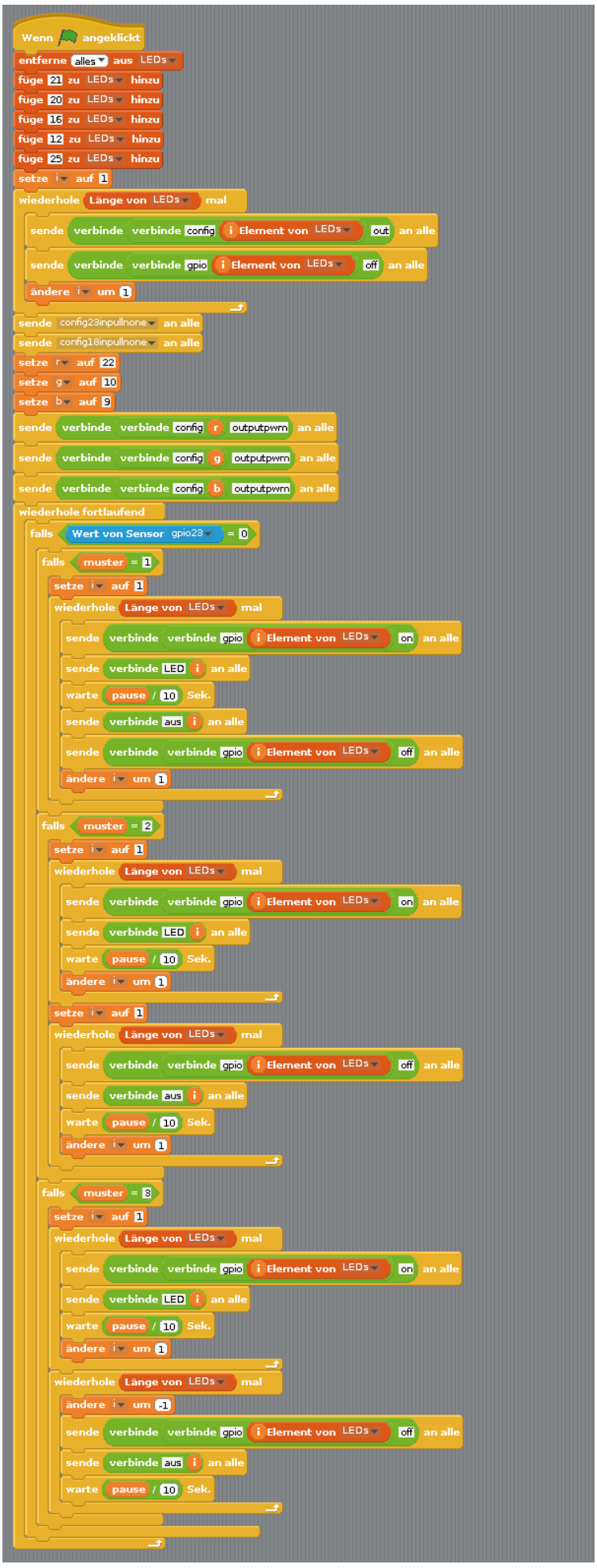
Furthermore, the familiar sub-programme to convert H-values into RGB colours can also be found on the stage. If you build the Scratch programme yourself, you can import this programme block from a previous programme.

The programme blocks for the RGB LED are allocated to the object **star** because it is to change as well when the modelling clay contact at GPIO pin 18 is touched.

A loop increases the H-value in steps from 0 to 360. At the end of the loop, the endless loop starts from the beginning and starts the nested loop with 0 resulting in a continuous change of colour.

Only when the modelling clay contact at GPIO pin 18 is touched, the current H-value at this time is converted into RGB. The star lights up the bright costume, which represents a switched on LED, and the calculated RGB value is represented via PWM signals on the RGB LED.

As long as the modelling clay contact is not touched, the object **Star** uses the grey costume and the RGB LED remains switched off.



The programme blocks on the stage.

21. Day

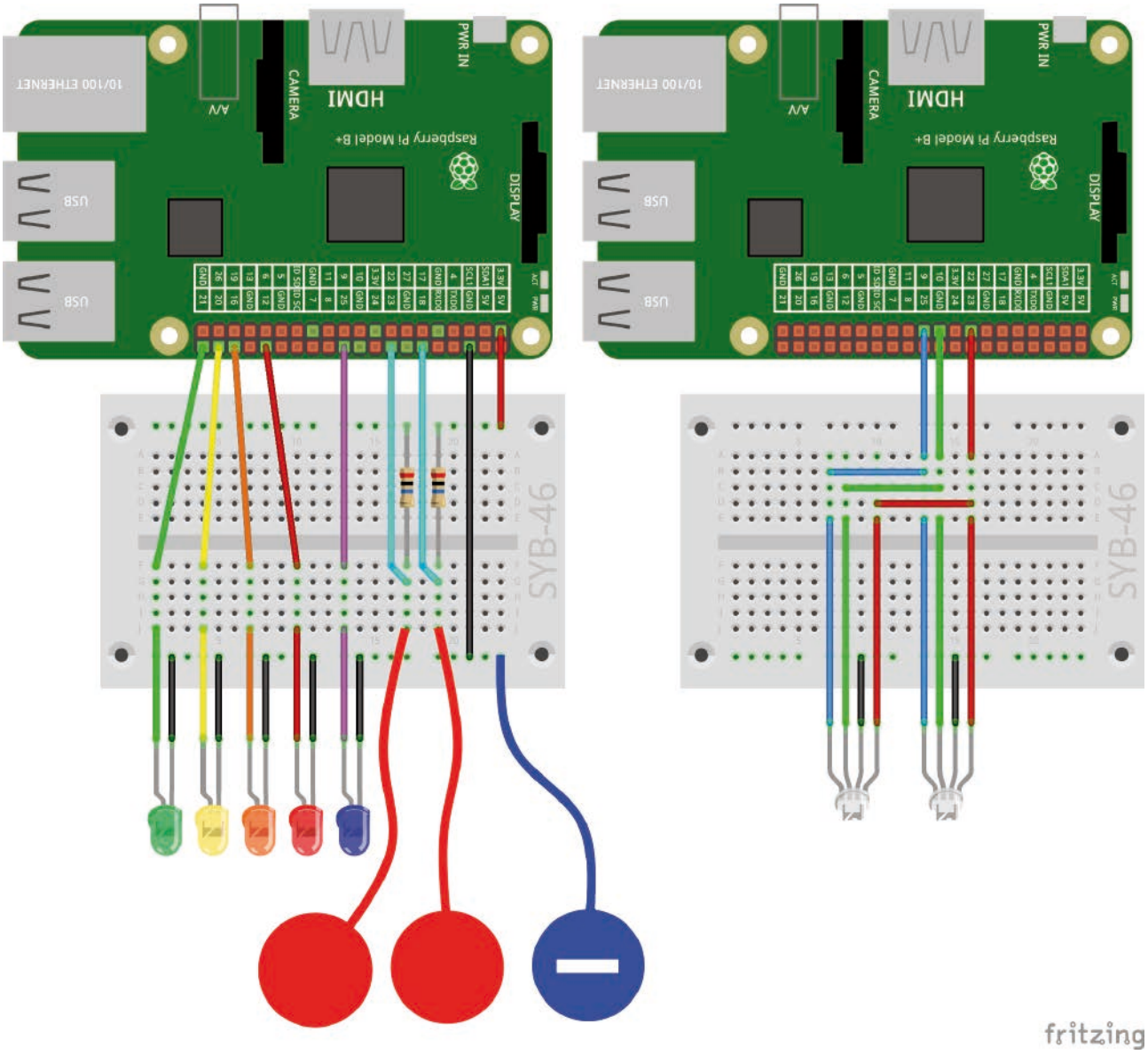
Today in the advent calendar

· 4 GPIO connection cables (long)

Colour plays on the screen

The experiment of Day 21 is similar to the one on the previous day. However, two RGB LEDs show the same colour change and a coloured point on the star shows the same colours as the RGB LEDs. The running light patterns are controlled as was done in the previous programmes.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 2 RGB LEDs with series resistor; 2 20-Mohm resistors; 12 short GPIO connection cables (Raspberry Pi – patch panel); 18 long GPIO connection cables (patch panel – LEDs); 3 connecting wires; 3 modelling clay contacts



Two modelling clay contacts control the running light and two synchronised RGB LEDs.

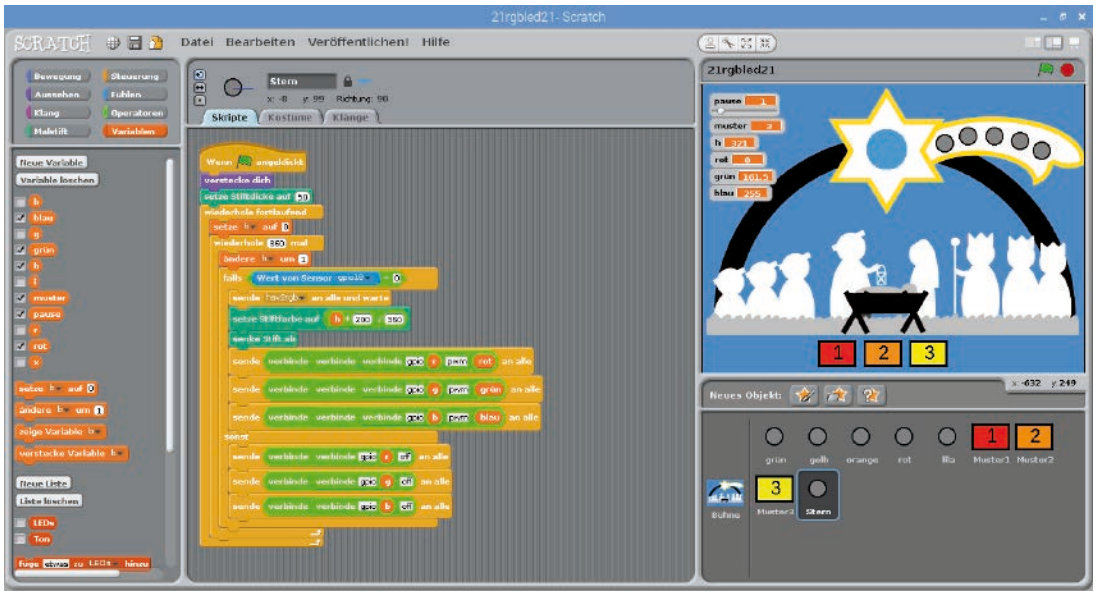
The two RGB LEDs are connected to the same GPIO pins using connecting wires and therefore always show the same colour.

The programme

Due to the large number of colours, the colour change on the star is not represented with costumes, but with the so-called drawing pen in Scratch, with which objects can leave traces of their current position on the stage.

The programme blocks of the object **Star** again cyclically change colour and query the modelling clay contact at GPIO pin 18.

First, the object **Star** is hidden so that it will not cover its own drawing traces. An object that is hidden on the stage with the block **hide** from the block pallet **Appearance** can no longer be seen, but can continue to move, run programme blocks and leave drawing traces.



The programme blocks for the star.

A block **set pen thickness to 50** from the block pallet **drawing pen** sets the thickness of the drawing pen so that it leaves a thick dot.

Every time the modelling clay contact at GPIO pin 18 is touched, a coloured point is painted in addition to the RGB LED lighting up. The point is located exactly where the hidden object **Star** is positioned.

To do this, the new pen colour is computed. Scratch uses 200 different pen colours along the colour spectrum, similar to H-values in the HSV system but not within the range **0...360** but only within the range **0...200**. For conversion, the current H-value is multiplied by 200 and then divided by 360.

The drawing pen is then lowered. Similar to a pen plotter, the object leaves traces during each movement with a lowered drawing pen, but not with a raised drawing pen. The drawing pen also makes a dot in the current pen colour even when there is not movement.

22. Day

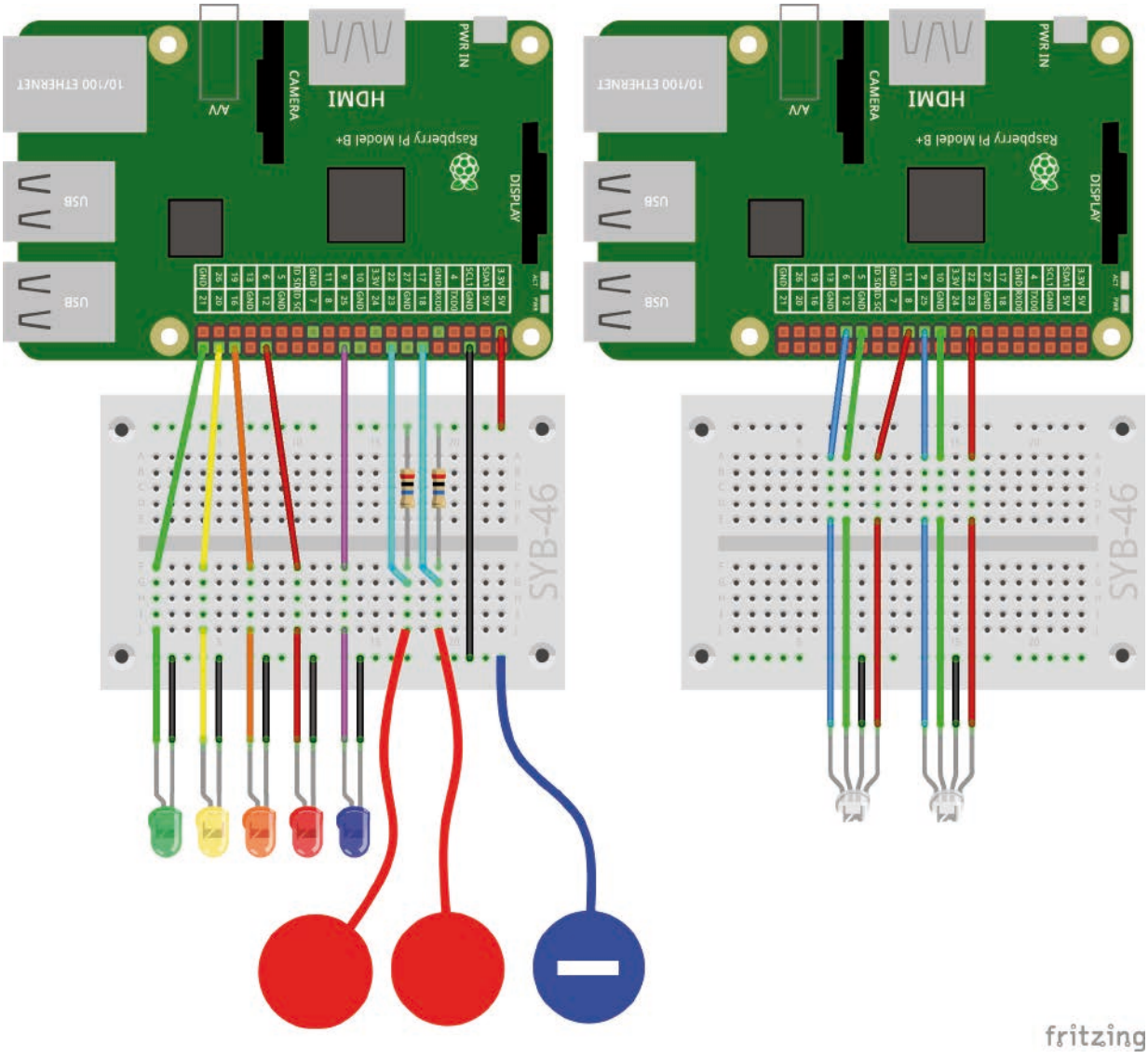
Today in the advent calendar

· 6 GPIO connection cables (short)

Controlling two RGB LEDs independently

The experiment of Day 22 controls the two RGB LEDs independently from each other using the two modelling clay contacts. Three additional GPIO pins are required for this. The running light patterns are only controlled via the buttons. This programme has two additional buttons. Pattern 4 makes all LEDs light up briefly; the button **off** switches all LEDs off.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 2 RGB LEDs with series resistor; 2 20-Mohm resistors; 15 short GPIO connection cables (Raspberry Pi – patch panel); 18 long GPIO connection cables (patch panel – LEDs); 3 modelling clay contacts



Two modelling clay contacts control the running light and two independently switchable RGB LEDs.

The programme

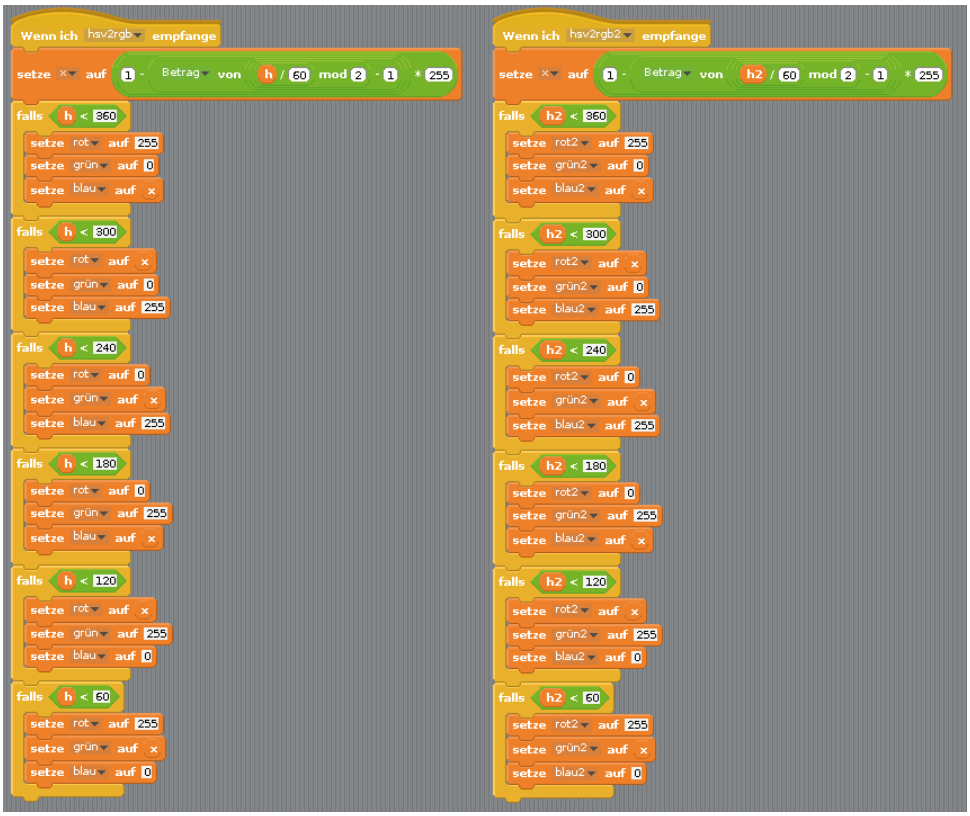
As is already familiar, the main programme on the stage initialises a list of five LEDs for the running light and the three PWM outputs **r**, **g** and **b** for the first RGB LED. An addition is the three other PWM outputs **r2**, **g2** and **b2** for the second RGB LED.

An endless loop represents one of five flashing patterns of the five LEDs using the variable **pattern**, which is set by the buttons. The patterns 0 and 4 are new:

Pattern 0 switches all five LEDs off without delays.

Pattern 4 makes all LEDS flash almost at the same time. Because the GPIO pins are switched on in succession and other Scratch blocks are run at the same time, short delays between the flashing of the individual LEDs may be noticeable.

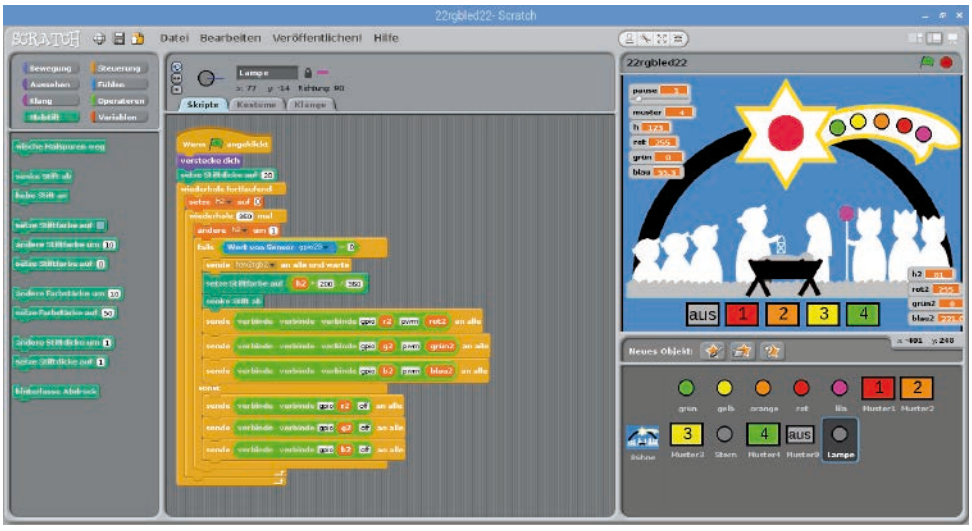
Furthermore, there are two programme blocks on the stage, which are run if the signal **hsv2rgb** or **hsv2rgb2** is received. The principle is already known. It is possible to allocate the second programme block directly to the corresponding LED object, but it can be created easily by duplicating and replacing the variable so that it is also located on the stage.



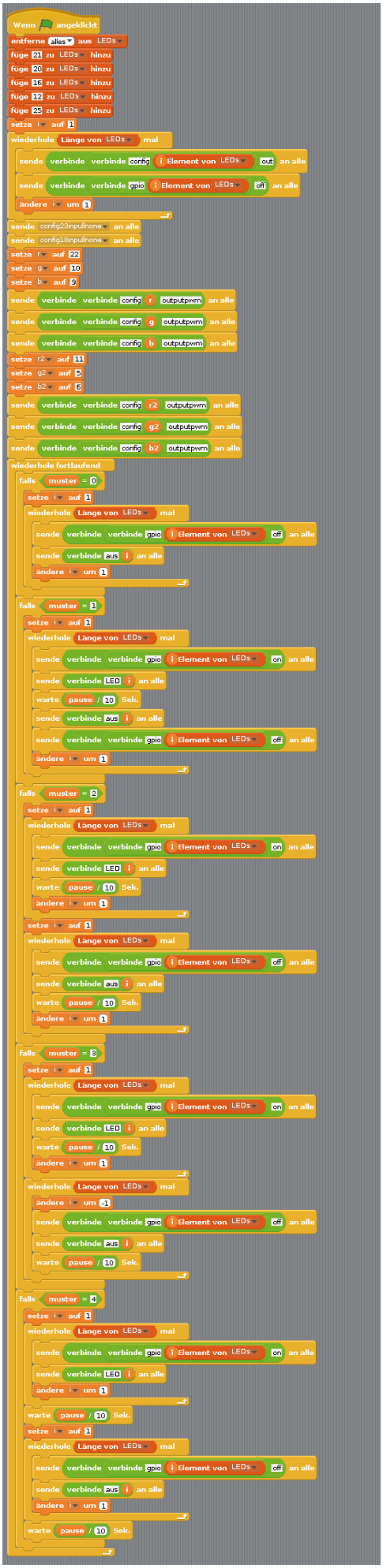
These blocks determine the RGB colours for the two RGB LEDs from H-values.

The new object **Lamp** at the staff of the first king shows the colour of the second RGB-LED. Following the familiar principle, the colour value **h2** is increased in steps from 0 to 360. If the modelling clay contact at GPIO pin is touched, the message **hsv2rgb2** is send and the RGB colour values **red2**, **green2** and **blue2** are calculated. After that, a pen colour is determined, a coloured point is drawn and the GPIO pins **r2**, **g2** and **b2** are set to the calculated PWM values.

The RGB LED remains off as long as the modelling clay contact is not touched. The object **Lamp** shows the colour set last.



The programme blocks for the new object **Lamp**.



The main programme on the stage.

23. Day

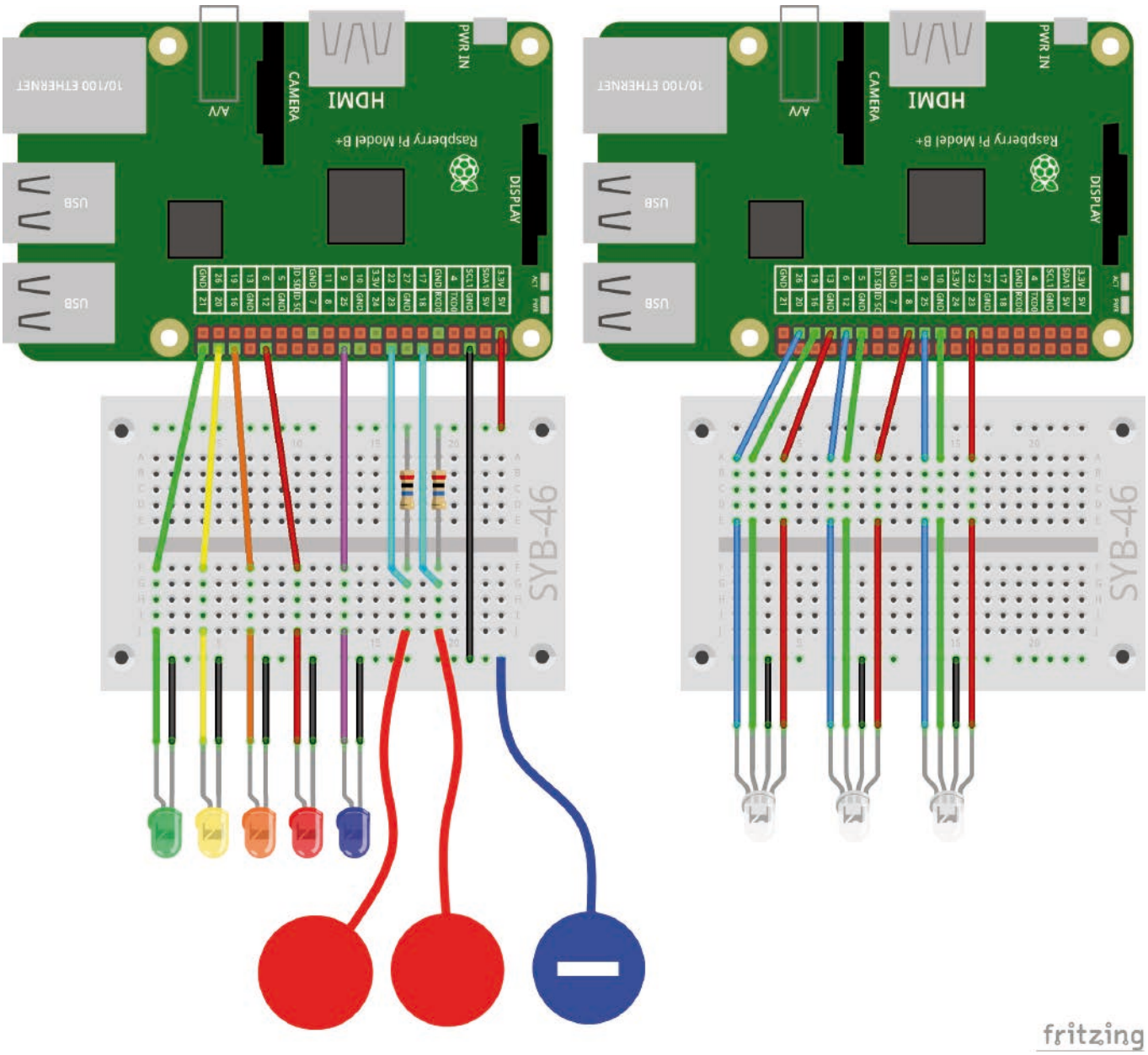
Today in the advent calendar

· 1 RGB LED with series resistor

Colourful illuminated Christmas manger

A third RGB LED in the lantern Joseph is holding on the manger lights up in different basic colours depending on the modelling clay contact being touched.

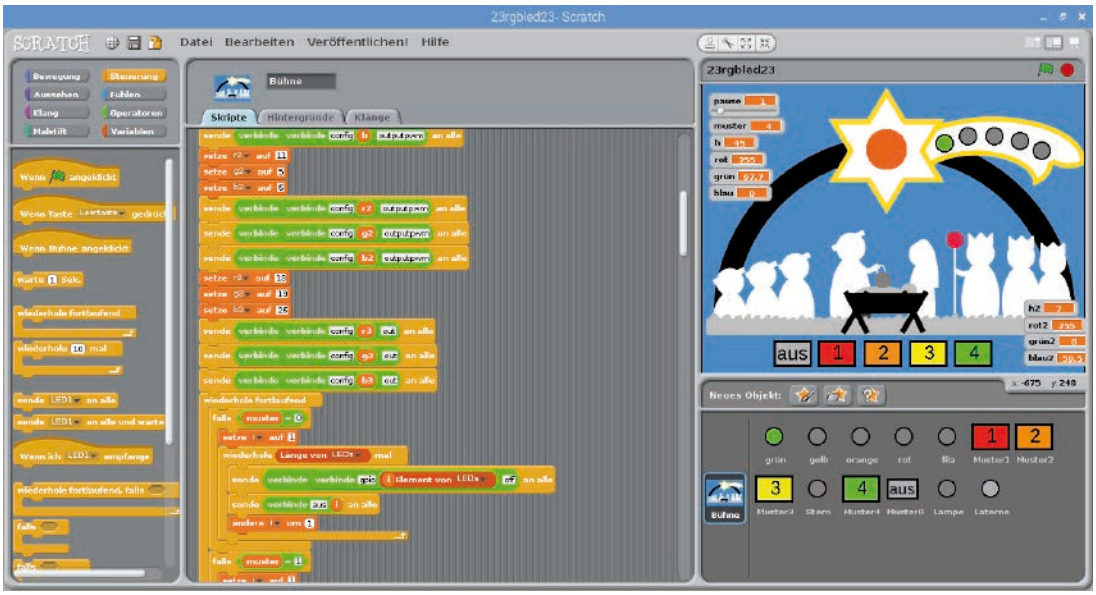
Components:1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 3 RGB LEDs with series resistor; 2 20-Mohm resistors; 18 short GPIO connection cables (Raspberry Pi – patch panel); 22 long GPIO connection cables (patch panel – LEDs); 3 modelling clay contacts



Two modelling clay contacts control the running light and three independently switchable RGB LEDs.

The programme

The main programme on the stage again initialises the GPIO pins for the LEDs and RGB LEDs. The GPIO pins named **r3**, **g3** and **b3** are new today. They are initialised as simple outputs and not as PWM outputs.



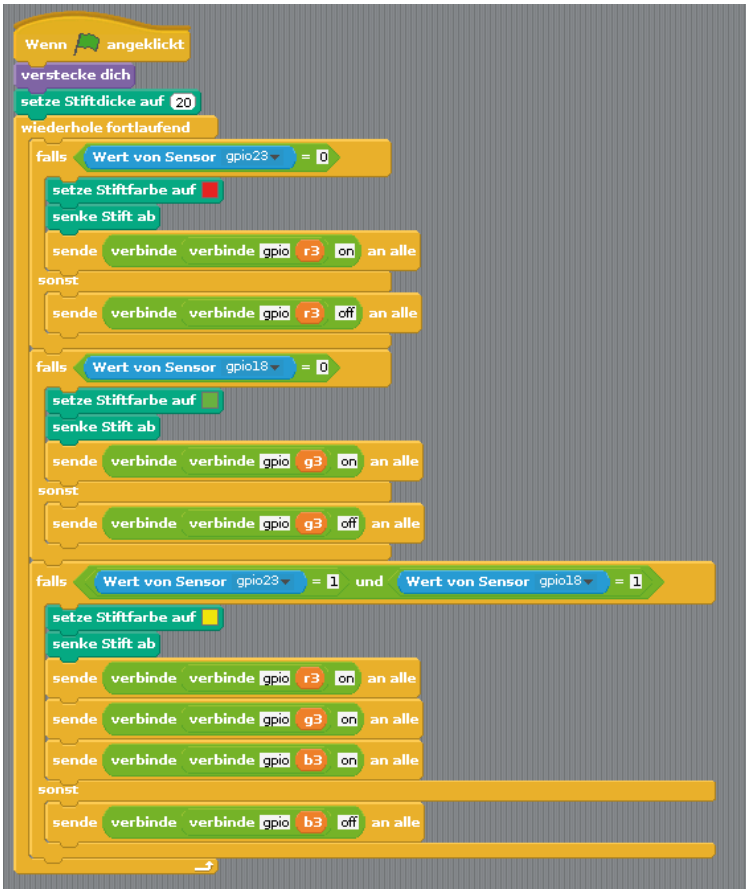
Additional GPIO outputs and a new object on the stage.

The new object **Lantern** is hidden from the beginning and only leaves a pen trace when the colour changes. The two modelling clay contacts are queried for this. If the sensor at pin 23 is touched, the pen colour is set to red. The block **sets pen colour to ...** with the colour field is used for this. Tap on the right colour field and a pipette symbol will appear, with which you can select any colour from the screen. The block will set this colour as pen colour later. Furthermore, touching this modelling clay contact the GPIO pin **r3** is switched on for the red colour of the third RGB LED. The pin is switched off again when the sensor is released.

Similarly, when touching the modelling clay contact at GPIO pin 18, the pen colour is set to green and the green colour component of the RGB LED is switched on. If both modelling clay contacts are touched at the same time, the pins **r3** and **g3** are switched on; the RGB LED lights up in a mixed colour. The object **Lantern** shows the colour switched on last because colours are not mixed here.

If one of the two modelling clay contacts is touch - i.e. both sensor values are set to **1** - the RGB LED lights up in bright white. To do this, the three GPIO pins **r3**, **g3** and **b3** are switched on. The object **Lantern** shows yellow because white is difficult to see on the background. If this condition is no longer true because at least one of the modelling clay contacts is touched, the GPIO pin **b3** is switched off so that the basic colours red, green or mixed yellow (without added blue) are shown again when one or both modelling clay contacts are touched.

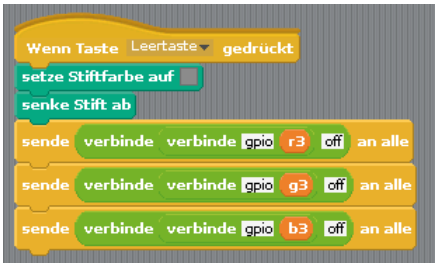
Up to now, the programme cannot switch off the RGB LED entirely. This is done by an additional programme black when the space bar is pressed.



The programme blocks for the new object **Lantern**.

The block **If button ... pressed** reacts to a pressed key on the keyboard. Select the desired key in the list field.

In this case, a grey point at the object **Lantern** is drawn and all three colour components of the RGB LED are switched off.



Pressing the space bar switches the RGB LED off.

24. Day

24. Day

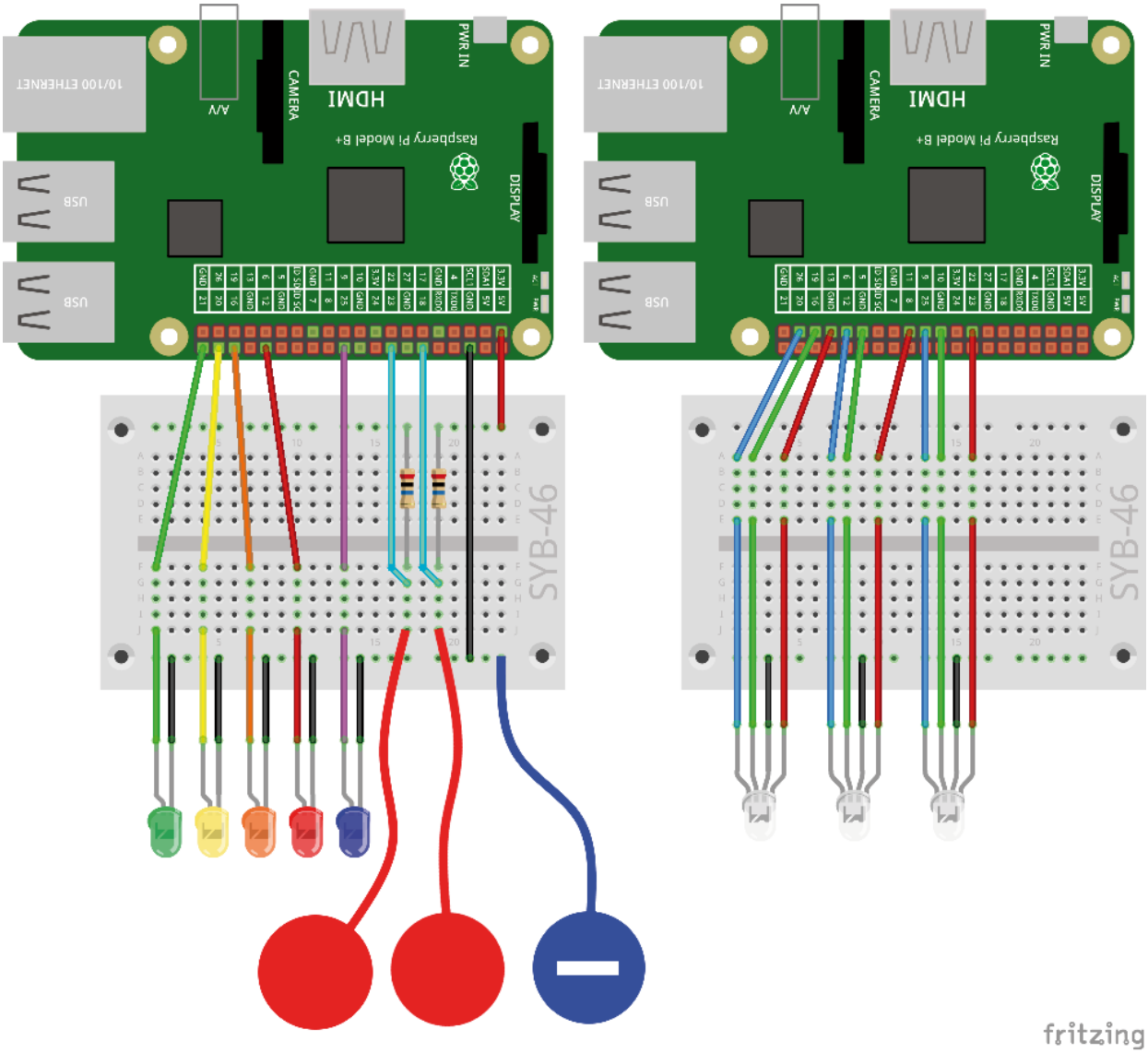
Today in the advent calendar

· 1 download code

Christmas songs on the Raspberry Pi

On the last day, the programme is expanded by another two interesting functions. When clicking on one of the five LED objects on the stage, five Christmas songs are played. The LED in the lantern can be changed interactively on the stage using a rainbow-coloured bar.

Components: 1 patch panel SYB-46; 1 green LED with series resistor; 1 yellow LED with series resistor; 1 orange LED with series resistor; 1 red LED with series resistor; 1 purple LED with series resistor; 3 RGB LEDs with series resistor; 2 20-Mohm resistors; 18 short GPIO connection cables (Raspberry Pi – patch panel); 22 long GPIO connection cables (patch panel – LEDs); 3 modelling clay contacts



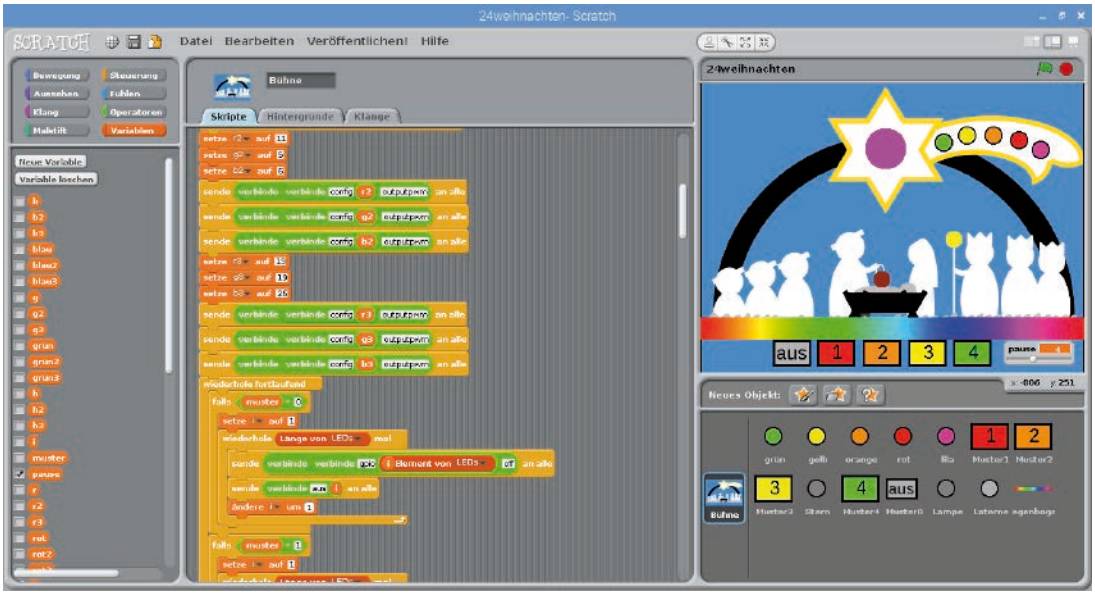
Two modelling clay contacts control the running light and three independently switchable RGB LEDs.

The programme

The programme blocks on the stage are known in principle. This time, the third RGB LED at the GPIO pins **r3**, **g3** and **b3** are set up as PWM outputs as was done for the other pins. A new programme block on the stage reacts to the message **hsv2rgb3** and calculates the RGB values for the third RGB LED. You can simply duplicate this block again and then replace the variables.

For the object **Rainbow** import the graphic `rainbow.png` from the downloads. When clicking on this object with the mouse, the RGB LED should light up in the corresponding colour.

Because the object can be easily moved when clicked, it will move back to its original position when it was clicked. After that, the H-value of the desired colour is determined based on the x-position of the mouse. The scratch stage is 480 units wide and has x-coordinates from **-240...240**. The block **Mouse x-position** reads the current x-coordinate of the mouse. Based on this value, which can be between -240 and +240, a value **h3** between 0 and 360 is calculated, **sends hsv2rgb3 to all and wait** starts the calculation of the corresponding RGB colour values. They are then shown as PWM signals on the RGB LED.



New programme blocks on the stage.

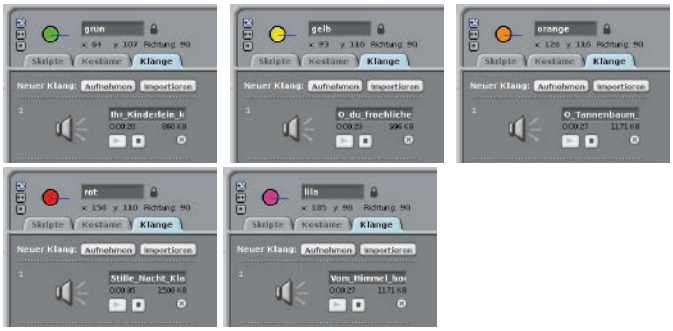
The object **Lantern** also reacts to the message **hsv2rgb3** and draws a coloured dot in the corresponding colour. The value **h3** within the range from 0 to 360 is converted into a colour value in the range between 0 and 200 for this purpose. Pressing the space bar switches the RGB LED off.

Scratch can play MP3 files on the Raspberry Pi that were imported into the programme. To do this, import one Christmas song for each of the objects **green**, **yellow**, **orange**, **red** and **purple** on the tab **Sounds**. The download code for the songs can be found behind today's window of the advent calendar.

When such an object is clicked, the corresponding Christmas song will be played.

The programme blocks for the new object **Rainbow**.

To do this, add a block **If ... clicked** to each object in addition to the blocks for the costume change. First, songs that might still be playing should be stopped so that several songs are not played at the same time. To do this, you can find the block **stop all sounds** on the block pallet **Sound**. After that, a block **play sound ... completely** will play the song imported to this object.



Importing Christmas songs for the LED objects.

Overview

Here is an overview of all programme functions:

- The four buttons with the numbers make the LEDs and the LED objects on the tail of the star flash in different patterns.
- The controller **pause** controls the flashing speed.
- The button **off** switches all LEDs and LED objects off.
- The first modelling clay contact controls the colour of the RGB LED and the coloured point on the star.
- The second modelling clay contact controls the colour of the RGB LED and the coloured point on the lamp of the king.
- Clicking on the rainbow-coloured bar controls the colour of the RGB LED and the coloured points on Joseph's lantern on the manger.
- Pressing the space bar switches this RGB LED off.
- Clicking on one of the five LED objects plays a Christmas song.

Merry Christmas!

Christmas songs are played when the objects are clicked.