

## PREPARING RASPBERRY PI

To put the Raspberry Pi into operation you need:

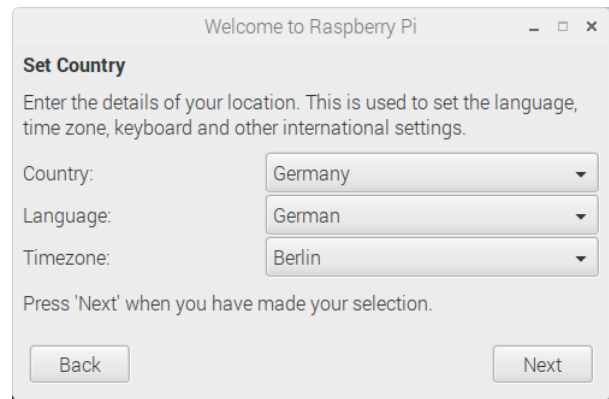
USB keyboard and mouse	MicroSD card with operating system
HDMI cable and monitor	Raspbian (from version 2.8.2)
Network cable or WLAN	Micro-USB mobile phone charger as power supply unit (at least 2,000 mA)

The power supply must be connected last, so that the Raspberry Pi switches on automatically. There is no separate on/off switch.

## OPERATING SYSTEM INSTALLATION IN A NUT-SHELL

**For all those who do not yet have their Raspberry Pi ready for operation with the current Raspbian version, here is the system installation in ten steps:**

1. Download NOOBS at least version 2.8.2 of [www.raspberrypi.org/downloads](http://www.raspberrypi.org/downloads) to your PC and unzip the zip archive to your hard disk.
2. If the SD card has already been used, reformat with the SD formatter in the PC: [www.sdcard.org/downloads/formatter\\_4](http://www.sdcard.org/downloads/formatter_4). Switch on **Format Size Adjustment** (the SD card must be at least 4 GB in size).
3. Copy all files and subdirectories from NOOBS to the SD card.
4. Remove the SD card from the PC, insert it into Raspberry Pi and boot. Select **English** as the installation language at the bottom. This automatically selects the English keyboard as well.
5. Check the preselected Raspbian operating system and click **Install** at the top left. After confirming a security prompt that the memory card is being overwritten, the installation starts, which takes a few minutes. Once the installation is complete, the Raspberry Pi reboots.
6. Since version NOOBS 2.8.2 an automatic configuration wizard starts. In the first dialog box, click **Next** and select Language and Time Zone if they are not automatically set to English.



Welcome to Raspberry Pi

**Set Country**

Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.

Country:

Language:

Timezone:

Press 'Next' when you have made your selection.

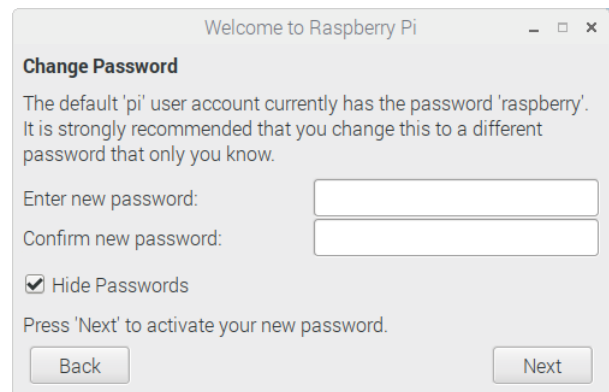
## CONRAD MAKER FACTORY EGG FOR RASPBERRY PI

**Controlling simple electronics using a normal PC or even a notebook – even if only for a few LEDs – can be quite a challenge for any hobby programmer. A PC simply lacks the necessary interfaces for this. In addition, the Windows operating system is unthinkable unsuitable for communicating with electronics.**

The Raspberry Pi is – although it doesn't look like it at first – is a full-fledged computer. Many things work a little slower than you are used to from modern PCs, but the Raspberry Pi is also much smaller and cheaper than a PC.

The experiments in this Tech-Egg are programmed using Scratch. This programming language is pre-installed on the Raspberry Pi and impresses with its low training. You can immediately start programming without any previous knowledge. All experiments work with Raspberry Pi 3 and Raspberry Pi 3 B+.

7. In the next step it is recommended to change the default password, which is not absolutely necessary for the experiments in this Tech-Egg.



Welcome to Raspberry Pi

**Change Password**

The default 'pi' user account currently has the password 'raspberrypi'. It is strongly recommended that you change this to a different password that only you know.

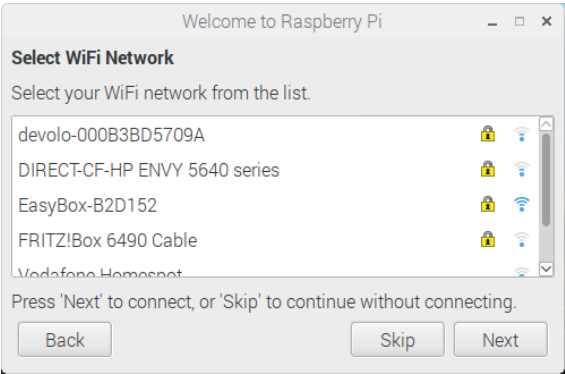
Enter new password:

Confirm new password:

☒ Hide Passwords

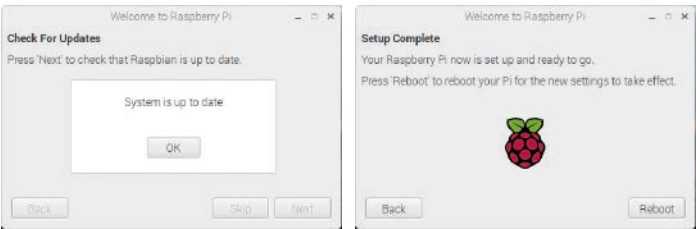
Press 'Next' to activate your new password.

8. For WLAN connection, select network and enter password, for Ethernet connection simply click on **Skip**.



9. The next step is to ask if you want to download updates. Depending on the number and size of updates available since the operating system image, this installation can take up to an hour, but can also be skipped with **Skip**.

10. The last step is to reboot the Raspberry Pi.



Num Lock Button

As with almost all Linux systems, the numeric keypad is switched off by default when Raspbian is started. Press the Num-Lock key on the keyboard to turn it on.

# THE COMPONENTS

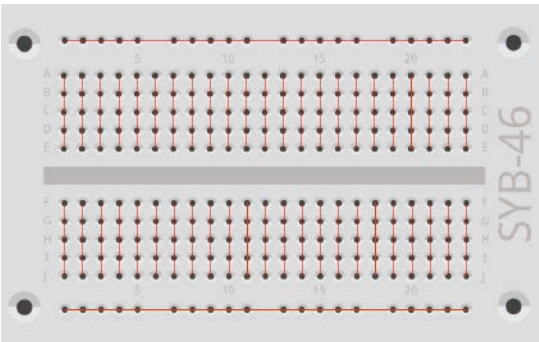
These components are included in the Tech-Egg:

- 1 pinboard
  - 7 LED with built-in series resistor
  - 1 resistor 20 MΩhm

The most important components briefly explained
- 1 piece of plasticine
  - 8 GPIO connection cable
  - Jumper wire

## PINBOARD

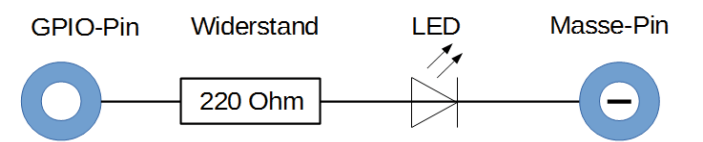
We use a plug-in board for assembling electronic circuits quickly. Here electronic components can be inserted directly into a hole grid. With this plug-in board, all outer longitudinal rows are connected to each other via contacts (X and Y). These contact rows are often used as positive and negative poles for the power supply of the circuits. In the other contact rows, five contacts (A to E and F to J) are connected crosswise, with a gap in the middle of the circuit board. This allows larger components to be inserted and wired to the outside.



The connections on the pinboard.

## LEDs

LEDs light up when current flows through them in the direction of flow. LEDs are represented in circuits with an arrow-shaped triangle symbol, which indicates the flow direction from the positive pole to the negative pole or to the ground line. An LED lets almost any current through in the direction of flow; it has a very low resistance. To limit the flow current and thus prevent the LED from burning through, a 220 ohm series resistor is usually installed between the GPIO pin used and the anode of the LED or between the cathode and ground pin. This resistor also protects the GPIO output of the Raspberry Pi from high currents. The LEDs in the Tech-Egg already has the series resistor built in and can therefore be connected directly to the GPIO pins.



Circuit diagram of an LED with series resistor.

Connect LED in which direction?

The two connecting wires of an LED have different lengths. The longer wire is the positive pole, the anode, the shorter one the cathode. Easy to remember: The plus sign has one stroke more than the minus sign, making the wire a little longer. In addition, most LEDs are flattened on the minus side, comparable to a minus sign. Also easy to remember: Cathode = short = flat edge.

## RESISTOR

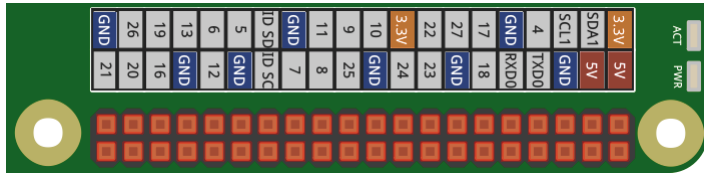
Resistors are used for limiting current on sensitive electronic components and as series resistors for LEDs. The unit of measurement for resistors is Ohm's. 1,000 ohms correspond to one kilo-ohm, abbreviated kΩhm. 1,000 kΩhm corresponds to a megohm, abbreviated MΩhm. The omega character Ω is often used for the unit ohm. The coloured rings on the resistors indicate the resistance value. With a little practice, these are much easier to recognise than tiny numbers that can only be found on old resistors. Most resistors have four such colour rings. The first two colour rings denote the digits, the third a multiplier and the fourth the tolerance. This tolerance ring is mostly gold or silver coloured – colours that do not appear on the first rings. Thus the reading direction is always unambiguous. The tolerance value itself hardly plays a role in digital electronics. The table shows the meaning of the coloured rings on resistors.

Colour:	Resistance value in ohm			
	1. Ring (tens)	2. Ring (one)	3. Ring (multiplier)	4. Ring (tolerance)
silver			10 <sup>-2</sup> = 0.01	±10 %
gold			10 <sup>-1</sup> = 0.1	±5 %
black		0	10 <sup>0</sup> = 1	
brown	1	1	10 <sup>1</sup> = 10	±1 %
red	2	2	10 <sup>2</sup> = 100	±2 %
orange	3	3	10 <sup>3</sup> = 1,000	
yellow	4	4	10 <sup>4</sup> = 10,000	
green	5	5	10 <sup>5</sup> = 100,000	±0,5 %
blue	6	6	10 <sup>6</sup> = 1,000,000	±0,25 %
violet	7	7	10 <sup>7</sup> = 10.000.000	±0.1 %
gray	8	8	10 <sup>8</sup> = 100,000,000	±0.05 %
white	9	9	10 <sup>9</sup> = 1,000,000,000	

This Tech-Egg contains a 20 MΩhm resistor (red-black-blue). It does not matter in which direction a resistor is installed. With LEDs, on the other hand, the installation direction plays an important role.

## GPIO CONNECTION CABLE

The coloured connection cables all have a plug on one side and a socket on the other side, which fits on a GPIO pin of the Raspberry Pi. The plugs are plugged into the plug-in board. The programmable GPIO pins on the Raspberry Pi have numbers, the ground pins are marked by GND in the figure.



Configuration of the GPIO pins.

## Precautionary measures

You should under no circumstances connect any GPIO pins to each other and wait and see what happens. Not all GPIO pins can be programmed freely. A few are fixed for power supply and other purposes. Some GPIO pins are directly connected to the processor terminals, a short circuit can completely destroy the Raspberry Pi. If two pins are connected to each other via an LED, a protective resistor must always be connected in between. LEDs with built-in series resistors are an exception. For logic signals, always use pin 1, which supplies +3.3 V and can be loaded up to 50 mA. Pin 6 is the ground line for logic signals. Pin 2 and 4 provide +5 V for power supply of external hardware. Here you can take as much power as the USB power supply of the Raspberry Pi supplies. However, these pins must not be connected to a GPIO input.

## JUMPER WIRE

The wire is used to create short connection bridges, which are used to connect rows of contacts on the plug-in board.

## THE SCRATCH PROGRAMMING LANGUAGE

Scratch is pre-installed on the Raspberry Pi in the menu under Development and is considered one of the easiest programming languages to learn. A program is clicked together using blocks that resemble a puzzle, so you don't have to remember command names and syntax rules.

## THE NEW SCRATCH 2

The Scratch programming language has been pre-installed in version 1.x ever since the first Raspbian version was released. For PCs there is the new Scratch 2 version with many more possibilities. Among other things, it allows you to create your own function blocks. Scratch 2 runs online on the PC in the browser. However, this requires more computing power than a Raspberry Pi can currently offer. Since version NOOBS 2.4.0, a version of Scratch 2 is pre-installed in the Raspbian operating system, which runs offline without a browser and thus runs smoothly with the capabilities of a Raspberry Pi 3. With Scratch 2, hardware control via the GPIO interface has become much easier. However, some important functions for GPIO control are not yet supported. Therefore, we continue to use the proven Scratch 1.4 for all projects in this Advent calendar.

## PREPARING SCRATCH

Click on the globe at the top left of the scratch logo and select English. The selected language remains saved, so you don't have to select it every time. Scratch 1.4 provides support for various hardware components on the GPIO port, which must



Switching Scratch 1.4 to English

be activated once for each program via the menu item Edit/Start GPIO Server.

You can see that GPIO support is active by the fact that this menu item changes to Stop GPIO server. Check this for every new scratch program.



Start Scratch 1.4 GPIO Server

## Downloading programs

The programs used in the Tech-Egg can be downloaded here: [www.buch.cd](http://www.buch.cd). Enter the code xxxx for this product in the input field.



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



Start the file manager on the Raspberry Pi. This automatically displays the home directory at startup. Right-click on the downloaded zip file and select **Unzip here** from the context menu.

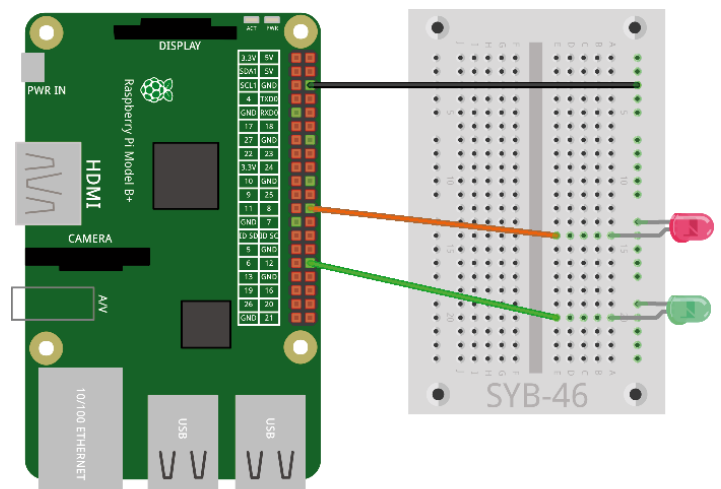
You will find this manual in the download archive as a colour PDF, so you can easily recognise the individual lines and the illustrations of the scratch programs on the circuit diagrams.

## TWO LEDS FLASH ALTERNATELY

The first experiment causes two LEDs to light up alternating between red and green. The whole thing is controlled via an endless loop in Scratch.

## COMPONENTS

- 1 Plug-in board
- 1 LED red with series resistor
- 1 LED green with series resistor
- 3 GPIO connection cables

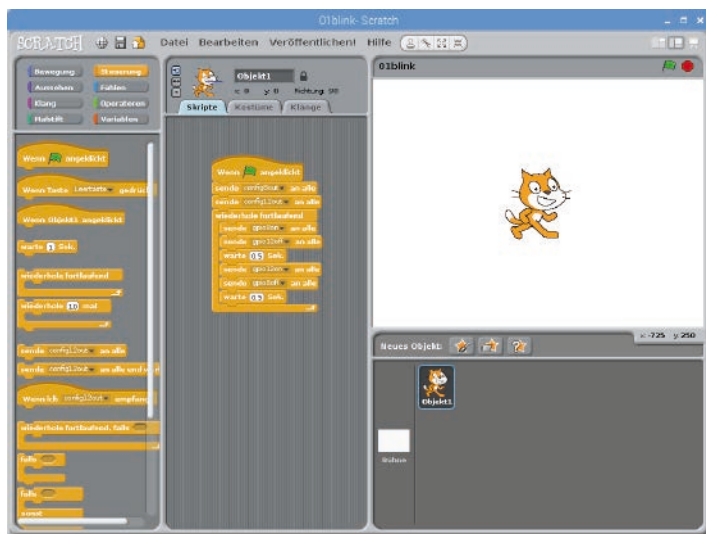


Two LEDs flash on the Raspberry Pi.

Make sure that the LEDs are installed correctly. The flat side is in the picture above. The circuit configuration makes use of the contact strip on one long side of the plug-in board as a ground contact. The cathodes of all LEDs are plugged in here and connected to a GND pin on the Raspberry Pi using a cable.

## THE PROGRAM

In Scratch you don't have to type any program code when programming. The blocks are simply attached to each other by dragging and dropping. The Block Palette in the left part of the scratch window contains the available blocks sorted by type.



This scratch program **01blink** controls both LEDs.

You can assemble the program on the screen or use the program 01blink from the download. To do this, from the File/Open menu, select the pibutton in the next dialog box to select the personal home directory where the downloaded programs are located.

Click on the yellow symbol Control in the upper left corner of Scratch. The blocks for Control are then displayed in the Block Palette on the left. For this first program all we need are these yellow blocks.

Drag the blocks you need from the Block Palette into the Scripts Area in the middle of Scratch.



The block when (green flag) clicked is used to start a program. The following script elements are run when you click on the green flag in the top right corner of Scratch. The block is round at the top, so it does not fit under any other block. It must always be set first.



The GPIO commands are sent via the Scratch block broadcast... The respective pin name and corresponding keywords are entered in the text field. To do this, click in the text field in the block, select New/edit... and enter the text.

At the start, the GPIO pins 8 and 12 are defined with config8out and config12out as outputs. Each GPIO pin can be either output or input.



A forever loop ensures that the two LEDs continue to flash endlessly until the user clicks on the red stop icon in the upper right corner of Scratch. All blocks within the loop are repeated.



After the red LED on pin 8 is switched on and the green LED on pin 12 is switched off, the program waits half a second. For this, Scratch offers its own block wait...secs.



The green LED at pin 12 is then switched on in the same way and the red LED at pin 8 is switched off. After another half second, the cycle repeats from the beginning.

If you want the LEDs to flash faster, shorten the times in the two **wait...sec** blocks within the loop. If you want them to blink more slowly, extend the waiting times. Like many American programs, Scratch uses the dot instead of the comma, as is common in English. You therefore write a half second waiting time as **0.5** and not **0,5**.

## SAVING THE PROGRAM

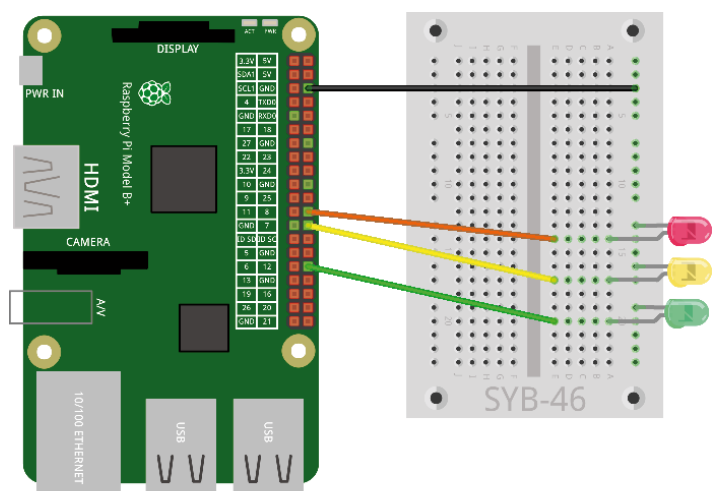
Don't forget to save the finished program for later use using the disk symbol in the upper left corner.

## SIMPLE TRAFFIC LIGHT CIRCUIT

**Switching two LEDs on and off may be exciting at first, but you don't really need a computer. A traffic light with its typical light cycle from green to yellow to red and then from red-yellow to green via a light combination is a bit more exciting.**

### COMPONENTS

- 1 Plug-in board
- 1 LED red with series resistor
- 1 LED yellow with series resistor
- 1 LED green with series resistor
- 4 GPIO connection cable



fritzing

Traffic light with three LEDs.

## THE PROGRAM

The program 02ampel01 does not really contain anything new compared to the previous program.



The Scratch program 02ampel01 controls the traffic light.

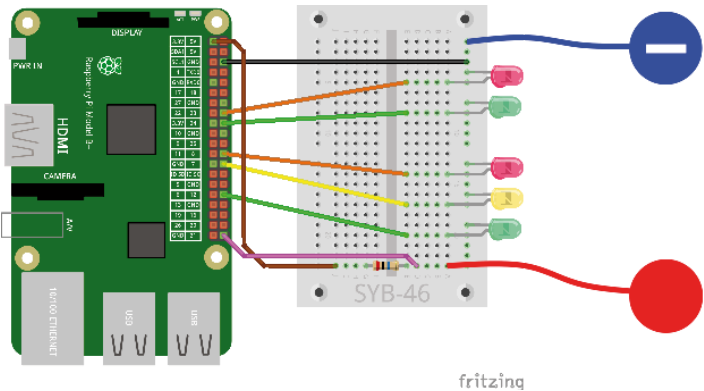
The first broadcast... blocks sets pins 12, 7 and 8 as outputs. Several GPIO commands can be sent in one block. The second broadcast... block creates a defined starting position if the LEDs from an earlier program still shine. The green LED at pin 12 is then switched on, the yellow LED at pin 7 and the red LED at pin 8 is switched off. The forever loop starts with the 2-second-long green phase. This is followed by yellow and red after 0.6 seconds. The red phase lasts 2 seconds again, and the traffic light switches to green following red/yellow. The loop then starts all over again. This cycle is repeated until the user clicks on the red stop icon.

## PEDESTRIAN TRAFFIC LIGHT WITH TOUCH SENSOR

The next experiment adds a pedestrian traffic light to the simple traffic light circuit, which indicates a green phase for pedestrians during the red phase of the traffic light. The traffic light cycle no longer runs automatically, but is started by touching a sensor button, as with a typical pedestrian traffic light.

### COMPONENTS

- 1 Plug-in board
- 2 LEDs red with series resistor
- 1 LED yellow with series resistor
- 2 LEDs green with series resistor
- 1 20-MOhm resistor (red-black-blue)
- 8 GPIO connection cables
- 2 plasticine contacts

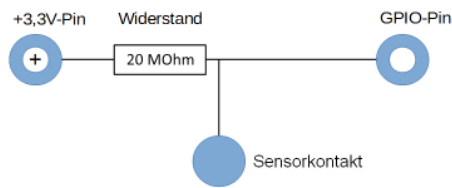


Pedestrian traffic light with touch sensor. Today traffic lights, door openers, light switches and machines are often controlled with contact sensors that you only have to touch.

Pushbuttons that really need to be pressed are becoming increasingly rare.

## TOUCH SENSOR FROM PLASTICINE

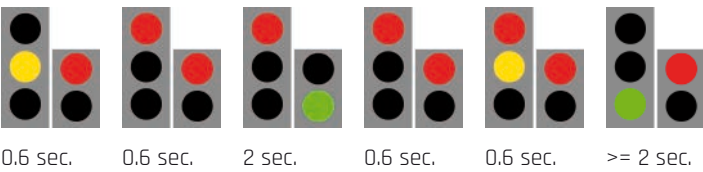
The GPIO pin switched as input is connected to +3.3 V via an extremely high-impedance resistor (20 MOhm), so that there is a weak but clearly defined high signal present. A person who is not floating freely in the air is always earthed and delivers a low level current through his electrically conductive skin. If this person now touches a sensor contact, the weak high signal is overcome by the significantly stronger low signal of the hand and pulls the GPIO pin to the low signal.



The level of resistance between the hand and the other material depends on many things, including shoes and floors. Barefoot in wet grass is the best connection to the earth's ground, but it also works well on stone floors. Wooden floors insulate more strongly, plastic floor coverings are often even positively charged. Plasticine conducts electricity almost as well as human skin. It is easily moulded into any shape, and a plasticine contact is much easier to handle than a simple piece of wire. The surface with which the hand touches the contact, is clearly larger. So it is not so easy as with a "loose contact". Insert a piece of stripped wire into a piece of plasticine. Insert the other end of the wire into the clipboard. If the sensor contact does not work, there is another plasticine contact for the circuit which can be connected to the ground rail of the plug-in board. This is marked in the drawing by a minus sign. Touch this and the actual sensor at the same time. The ground connection is made in any case.

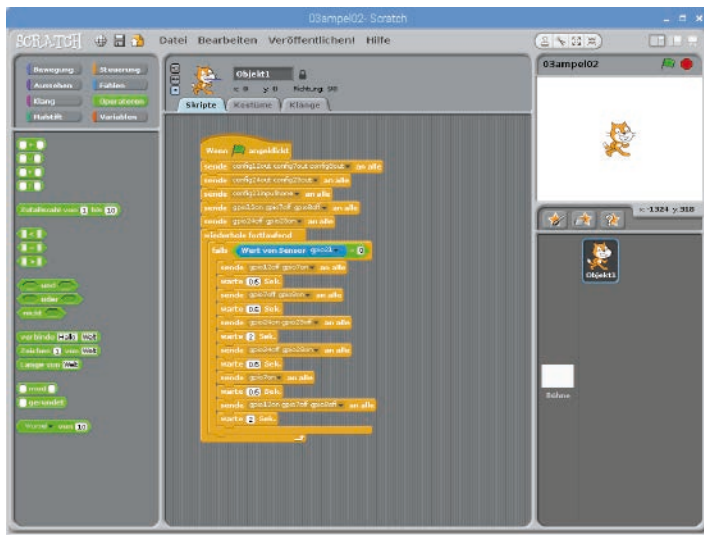
## THE PROGRAM

The program 03ampel02 switches the pedestrian traffic light. Click on the green flag to start the program. The traffic lights shine green, the pedestrian lights red, just like real traffic lights and would do so for hours if no pedestrian came and pressed the button. Now touch the red plasticine contact. The traffic light cycle will now start. In our program, as with a real traffic light, it consists of 6 different light patterns, which shine for different durations.



After the last light pattern - pedestrian red, traffic light green - the traffic light returns to its normal state. However, the programme must ensure that this too is always adhered to for a minimum period. Even if pedestrians keep pushing the button, cars have to be allowed to drive. In our model traffic light this is 2 seconds, with a real traffic light of course much longer.





The Scratch program **03ampel02** controls the pedestrian light.

At the beginning the three pins 12, 7 and 8 for the traffic lights as well as 24 and 23 for the pedestrian lights are defined as exits. Pin 21 is set as the input for the touch sensor. ...inpullnone switches off the pull-down resistor built into the GPIO pin, as this would impair the function of the touch sensor.

The traffic light is then switched on, green for cars, red for pedestrians. The other three LEDs are switched off. Switching off is not really necessary at the beginning. It only serves to start the program with a clearly defined state.

As in the last program, a forever loop now begins, which searches the plasticine contact at GPIO pin 21. If it returns the value 0, it is touched and pulled to low level.



For this, we use the scratch block if, which works similar to an if query in other programming languages. For the query itself, within the if block an oblong field with pointed ends is provided. A block from the green block palette Operators must be inserted here. Select the block with the equals sign and drag it to the placeholder field in block if.



This operator is always true if the two values left and right of the equal sign are the same. In our case, the value of GPIO pin 21 should be 0. The block 'Sensor Value' from the blue block palette Sensing is used to query GPIO inputs.



Drag this block to the placeholder field on the left of the green EQUAL-TO operator. Select the sensor gpio21 in the list field of the blue block. In addition to some predefined sensors, all GPIO pins defined as inputs are available for selection. Now click once on the green flag in the upper right corner to start the unfinished program. This defines the GPIO pins and gpio21 appears in the selection list. Stop the program again using the red stop sign. Then enter the value 0 in the green equality operator on the right.

Now the different light patterns of the traffic lights are switched on one after the other at intervals of 0.6 seconds. While the pedestrian light is green, it waits 2 seconds. The minimum duration of the green phase of the traffic light is also 2 seconds.

## LEDS WILL FLASH RANDOMLY

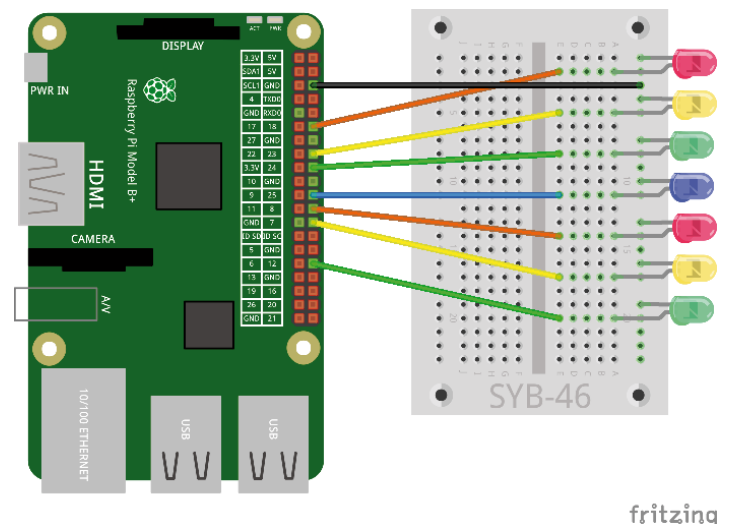
**Seven LEDs will flash in random order, and several may be on at the same time. That's how it works: A randomly selected LED is switched on alternately and then another randomly selected LED is switched off.**

### How are random numbers generated?

It is commonly thought that nothing can happen randomly in a program – so how can a program be able to generate random numbers? If you divide a large prime number by any value, you get numbers from the umpteenth decimal place that are very difficult to predict. If you increase the divisor regularly the number also change randomly. Although the result may seem random, it can be reproduced at any time by an identical program or by running the same program several times. But if we now take a number assembled from some of these digits and divide it again by a number that results from the current seconds of time or the contents of any memory location of the computer, we get a result that cannot be reproduced and is therefore called a random number.

## COMPONENTS

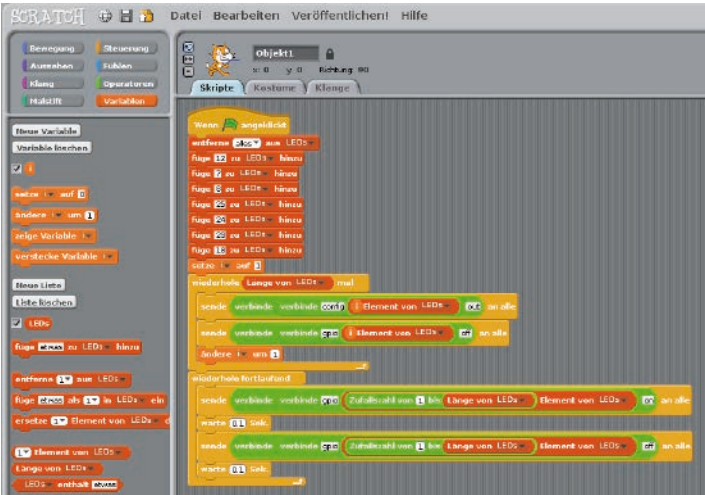
- 1 Plug-in board
- 2 LEDs red with series resistor
- 2 LEDs yellow with series resistor
- 2 LEDs green with series resistor
- 4 GPIO connection cable



7 LEDs are flashing.

# THE PROGRAM

The program 04leds contains some new elements compared to the previous programs.



The Scratch program 04leds makes 7 LEDs flash randomly.

For this program, on the block pallet create variables a variable named i and a list named LEDs. The variable i is used for a loop iteration counter. Traditionally, programmers always use i, j, k, ... for counters.

## Variables and lists in Scratch

Variables are small memory locations where a program remembers a number or something else. When the program is closed, these variable memories are automatically emptied. Variables have to be created in Scratch on the block palette **Variables** by clicking on **New variable** before they can be used. You can then drag the symbol of the newly created variable from the block palette to a designated field of a block in the program. The block palette also contains various blocks for reading and changing variables. A list is a special type of variable. In such lists, several values can be stored and addressed via the position within the list. The list elements are incremented starting with 1. The number of the last element corresponds to the length of the list.

Initially, the list LEDs is completely emptied, which means it no longer contains any data from the last run program. The contents of variables and lists remain in memory even if the program is stopped and restarted using the red stop symbol. The seven pin numbers of the GPIO pins used for the LEDs are then entered into the list one after the other. The variable i is set to 1 as loop iteration counter and a loop is started, which runs seven times, according to the length of the LEDs list. Scratch provides a block Length of..., which always contains the length of a selectable list.



In order to define and switch off each pin as an output, the texts in the broadcast... blocks are reassembled for each LED. The block join on the green block palette Operators joins any two strings to each another. Two nested join... .. blocks connect three strings; the word config, the number of the GPIO pin corresponding to the current loop iteration counter and the word out to define the pins as outputs. The pins are switched off in the same way using gpio, the pin number and off. Then the loop counter is increased by 1 and the next pin is defined as output and switched off.

Zufallszahl von 1 bis 10

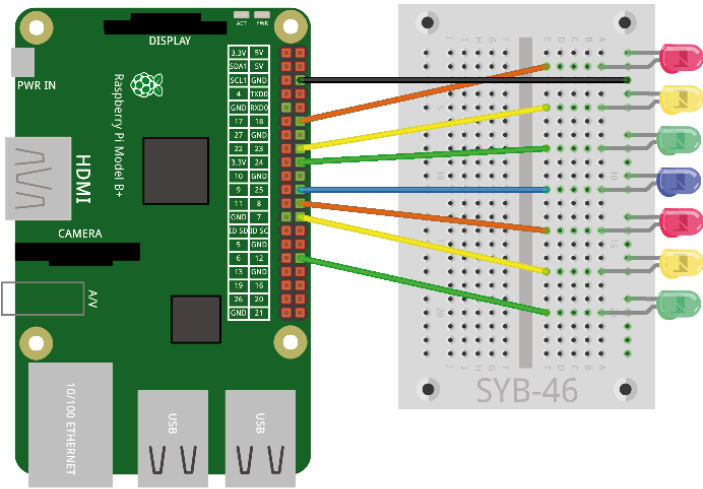
Thereafter, a forever will start running on a randomly selected LED and turning off a randomly selected LED after 0.1 seconds. Here too, the texts in the broadcast... blocks are rebuilt every time. A block pick random... to... from the green block palette Operators selects a random LED. The random number must be between 1 and the length of the list LEDs, in our case 7. Since the LED to be switched on and the LED to be switched off are selected randomly and independently of each other, different numbers of LEDs can be switched on at the same time. It is even possible that nothing changes during a loop run.

# RUNNING LIGHT WITH ADJUSTABLE SPEED

The circuit design for this experiment is similar to the previous one, but the LEDs do not flash randomly, but rather alternately one after the other. The chaser effect is caused by the fact that an LED shines for a short time and once it is switched off, the LED next to it immediately turns on. After the last LED in the row has turned off, the first one turns on again.

## COMPONENTS (THE CONSTRUCTION CORRESPONDS TO THE PREVIOUS EXPERIMENT)

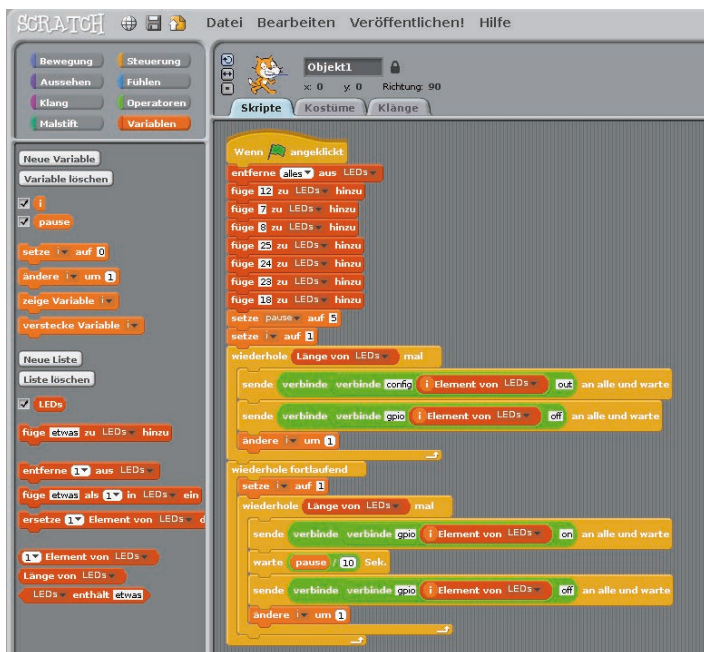
- 1 Plug-in board
- 2 LEDs red with series resistor
- 2 LEDs yellow with series resistor
- 2 LEDs green with series resistor
- 1 LED blue with series resistor
- 8 GPIO connection cables



7 LEDs as running lights.

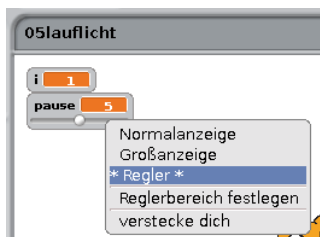
## THE PROGRAM

The program 05lauflicht lets the LEDs light up as running lights. The speed can be adjusted interactively while the program is running. For this, the program uses an additional variable pause, which indicates the pause between two switching operations.



The Scratch program **05laufflicht** lets LEDs light up as running lights.

You can rebuild the program from the previous one, since the start is largely the same. A block set pause to 5 from block pallet variables sets pause to 5, which corresponds to half a second in the program. An endless loop switches the LEDs on and off one after the other. Since only integers can be set interactively as variable values in Scratch, the program divides the value of the variable pause by 10. To do this, use the block ... / ... from the block pallet Operators. Drag the variable symbol pause from the block palette Variables into the left field of the operator and enter 10 on the right. To change the value of the variable in the program interactively, activate the check mark to the left of the variable symbol on the Variables block pallet. The variable will then appear on stage and always displays the actual value. Right-click the variable on the stage and select slider from the context menu. A slider appears with which the variable value can be adjusted.



Then click on set slider min and max and set the maximum value to 10, which corresponds to a pause of 10 seconds for later splitting.



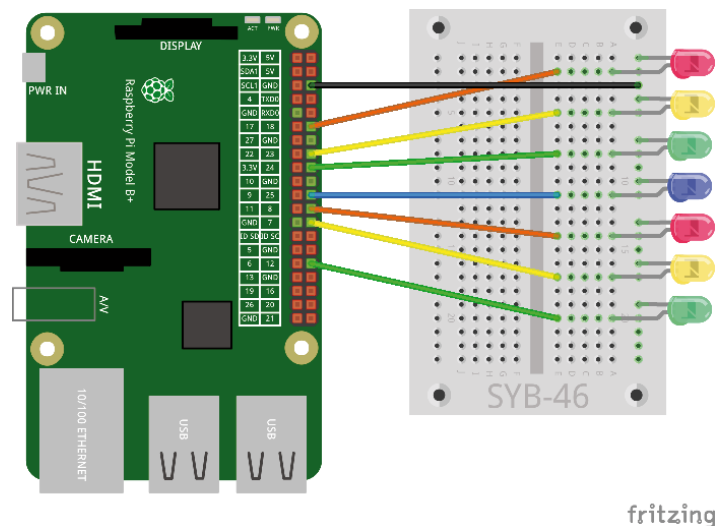
Start the program. The LEDs automatically start to flash one after the other. Now you can use the slider to adjust the speed of the variable pause on stage.

## LIGHT CONTROL PANEL IN SCRATCH

A simple light control panel switches the LEDs in Scratch by clicking. The Scratch control panel also displays the status of the LEDs.

### COMPONENTS (THE CONSTRUCTION CORRESPONDS TO THE PREVIOUS EXPERIMENT)

- 1 Plug-in board
- 2 LEDs red with series resistor
- 2 LEDs yellow with series resistor
- 2 LEDs green with series resistor
- 1 LED blue with series resistor
- 8 GPIO connection cables

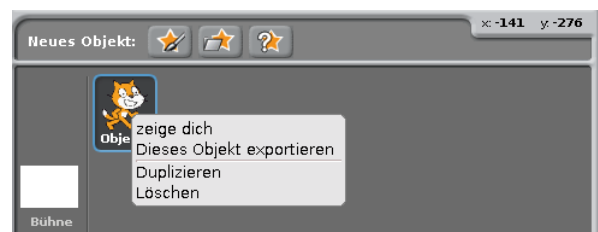


fritzing

Control 7 LEDs with a control panel.

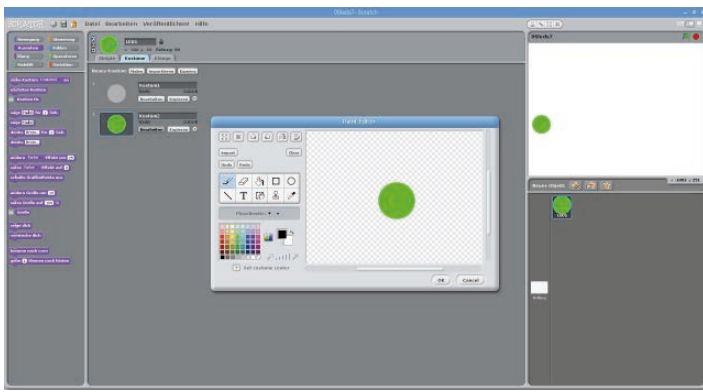
### THE PROGRAM

The Scratch stage, where only the cat could be seen so far, can display much more than just variables. Here you can create graphical objects (also called sprites) that move or respond to interaction with the user. Each program can contain several objects/sprites. In the program 06leds7, each object/sprite has its own program blocks. First delete the cat, which is not used in this program. To do this, right-click on the cat in the lower right object/sprite window and select Delete from the context menu.



Now create a new object/sprite for the first LED. Once this has been completely designed and programmed, you can duplicate it for the other LEDs and slightly adjust it. You can paint this object directly in Scratch. Click on the symbol Paint new sprite in the sprite window. Scratch contains a simple painting program with which you can paint a grey circle as a new sprite. Give this object/sprite the name LED1 at the top of Scratch's main window.





You can change the appearance of every object/sprite in Scratch using so-called costumes. Select the object/sprite LED1 and switch to the tab Costumes in the script window. Copy the existing costume by clicking on Duplicate.

Edit the second costume with the paint program. Fill the grey area green to indicate that the LED is on.

The object LED1 gets its own program blocks. If the green flag is clicked, the object should also appear grey on the screen once it is switched off. Hover over the block If green flag clicking on a block switch costume... to from the Looks block pallet. Select costume1 in the selection field of this block.

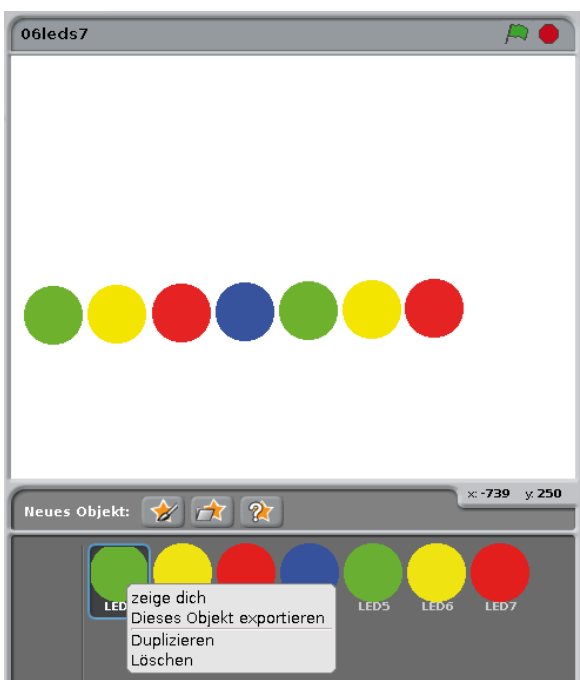


If the object itself is clicked, it should light up in colour on the screen. At the same time, the corresponding LED should turn on. By clicking once more, the the LED and the screen object will turn off.

Use block If LED1 is clicked from the Controller tab. When you click on it, first the costume is changed, then it is checked whether the coloured costume2 for the activated LED is currently visible.

In this case, the LED at pin 12 is switched on. If, on the other hand, the costume1 for the inactivated LED is visible, the LED at pin 12 is switched off.

After all program blocks have been completed, duplicate the object LED1 by right clicking in the object/sprite window. Rename the new object to LED2.



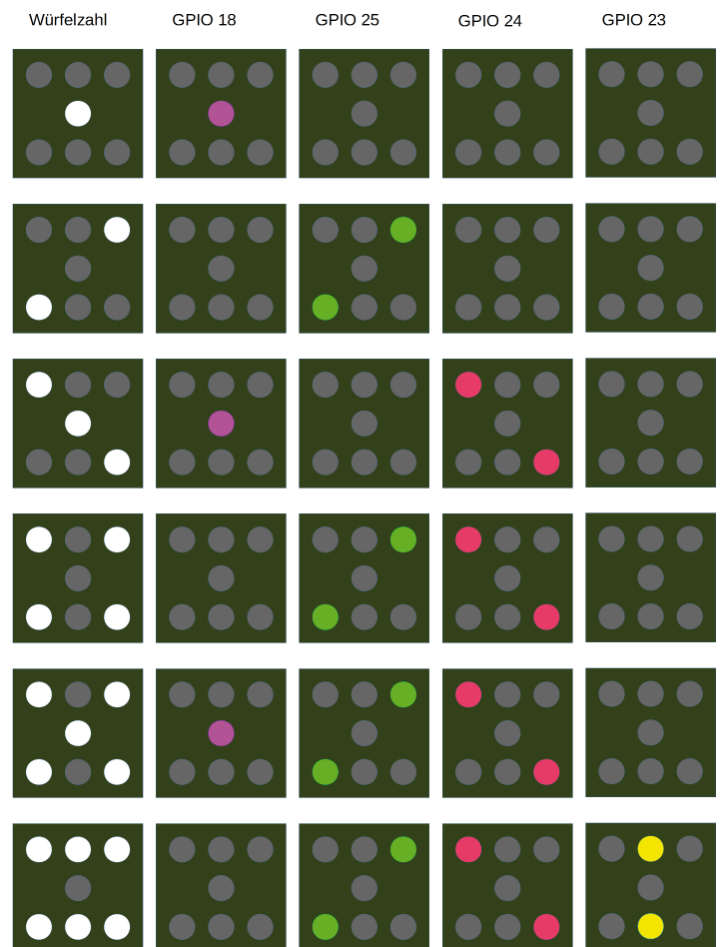
Edit the costume 2 in the paint program. Fill the circle with the colour yellow. Then change the used GPIO pin in the script block to 7. Duplicate more blocks, then change the colours of the costume2 to the colours of the corresponding LEDs and pin numbers accordingly.

Start the program by clicking on the green flag. All LEDs are off; the objects appear in grey. If you click on one of the objects, it appears in colour and the corresponding LED shines until the next click. This allows you to turn all LEDs on and off individually.

## LED DIE

**Everyone knows the typical die for use in games, which has 1-6 dots; everyone has them at home. An electronic die is however much cooler, with the glowing dots controlled by a plasticine contact - and then not just 1-6 LEDs in a row, but in the arrangement of a die for games.**

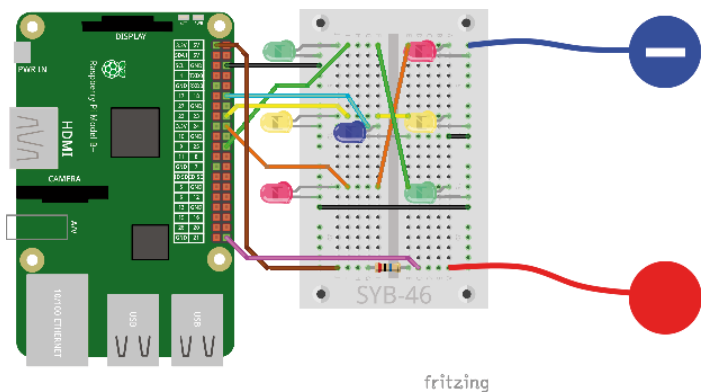
**Game dice have dots in the typical square arrangement, requiring 7 LEDs controlled in four groups, as depicted in the illustration. To control the LEDs we only need 4 rather than 7 GPIO pins, because a die uses pairs of dots to display even numbers.**



The dot symbols on a die are created by LED groups

## COMPONENTS

- 1 Plug-in board
- 2 LEDs red with series resistor
- 2 LEDs yellow with series resistor
- 2 LEDs green with series resistor
- 1 LED blue with series resistor
- 1 20-MOhm resistor (red-black-blue)
- 7 GPIO connection cables
- 5 wire jumpers (different lengths)
- 2 plasticine contacts

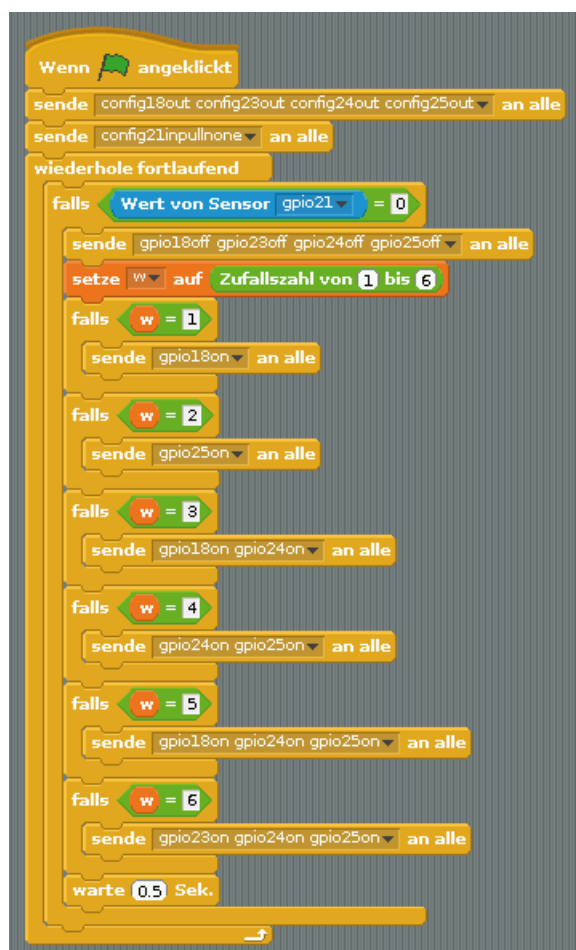


Die with 7 LEDs controlled via a sensor contact.

For LEDs switched at the same time, we use the same colours. For this circuit you need some wire jumpers. Cut the wire to the appropriate length using a small wire cutter. To be able to insert the wires easier into the plug-in board, it is advisable to cut the wires slightly diagonally to create a kind of wedge. Then remove the insulation at both ends using a sharp knife over a length of about 5 mm.

## THE PROGRAM

The program 07die simulates a game die. At the start, all LEDs are off. Briefly touch the two touch sensors to roll the dice. Once you release the touch sensors, the rolled number remains displayed until the touch sensors are pressed again.



The program 07die activates a game die.

Once the player clicks on the green flag, the five GPO pins used are initialised first. Here we use pins 18, 23, 24, 25 as outputs for the LEDs and pin 21 as input for the 'plasticine contact'.

The closed loop will now begin, waiting for the plasticine contacts to be touched. For this purpose, GPIO pin 21 is scanned in each run.

If the GPIO pin has the value 0, the instructions within the if block are executed. First, the program switches off the four LEDs. At the start, the LEDs are all off, but later a displayed dice result remains lit until the plasticine contacts are touched again.

Then a random number between 1 and 6 is generated and stored in the variable w. If you turn the switch to the left of the variable w on the block palette Variables, this variable is automatically displayed in a small orange field on the stage with the cat. This way you can always see the diced number and easily check if the LEDs are working. This number field is very small. Right-click and select Large Readout from the menu. Then the number is easier to read.

After the number is rolled, six if blocks will follow for each possible die value. Each of these block switches on the corresponding combination of LEDs when a certain number is rolled.

Irrespective of the result on the die, the program should always wait half a second after the plasticine contact has been thrown, to avoid two different dice actions being triggered in quick succession by so-called key bounce. This is done by a block wait 0.5 sec. after the last if block. This is processed in each case.

All circuits and programs presented here have been developed and tested with the greatest possible care. Nevertheless, errors cannot be completely eliminated. The publisher and author are liable in cases of intent or gross negligence in accordance with the statutory provisions. Otherwise, the publisher and author shall only be liable under the Product Liability Act for injury to life, limb or health or for culpable breach of material contractual obligations. The claim for damages for the violation of essential contractual obligations is limited to the foreseeable damage typical for the contract, unless there is a case of mandatory liability under the Product Liability Act.

### Attention! Eye protection and LEDs:

Do not look directly into an LED from a close distance, as a direct view can cause retinal damage! This is especially true for bright LEDs in clear housings and especially for power LEDs. With white, blue, violet and ultraviolet LEDs, the apparent brightness gives a false impression of the real danger to your eyes. Special care should be taken when using converging lenses. Operate the LEDs as described in the instructions, but not with higher currents.

### Dear customers,

This product has been manufactured in accordance with the applicable European directives and therefore bears the CE mark. The intended use is described in the enclosed manual.

For any other use or modification of the product, you are solely responsible for compliance with the applicable rules. Therefore, you should construct the circuits exactly as described in the instructions. The product may only be passed on along with these instructions.

The symbol of the crossed-out dustbin means that this product must be recycled separately from household waste as electronic waste. Your local authority will tell you where to find the nearest free collection point.

All rights reserved, including photomechanical reproduction and storage in electronic media. The creation and distribution of copies on paper, data carriers or the Internet, in particular as PDF, is only permitted with the express permission of the publisher and could otherwise lead to prosecution.

WEEE-REG.-NO.:  
DE 21445697

Most product names of hardware and software as well as company names and logos mentioned in this work are generally also registered trademarks and should be considered as such. The publisher essentially follows the spellings of the manufacturers when it comes to product designations.

Author: Christian Immler GTIN: 4019631150431 N° 1891587

Not suitable for children under 14!



MAKERFACTORY  
distributed by Conrad Electronic SE  
Klaus-Conrad-Str. 11 92240 Hirschau  
www.makerfactory.com

© 2019 Franzis Verlag GmbH  
Richard-Reitzner-Allee 2  
D-85540 Haar, Germany  
2019/01