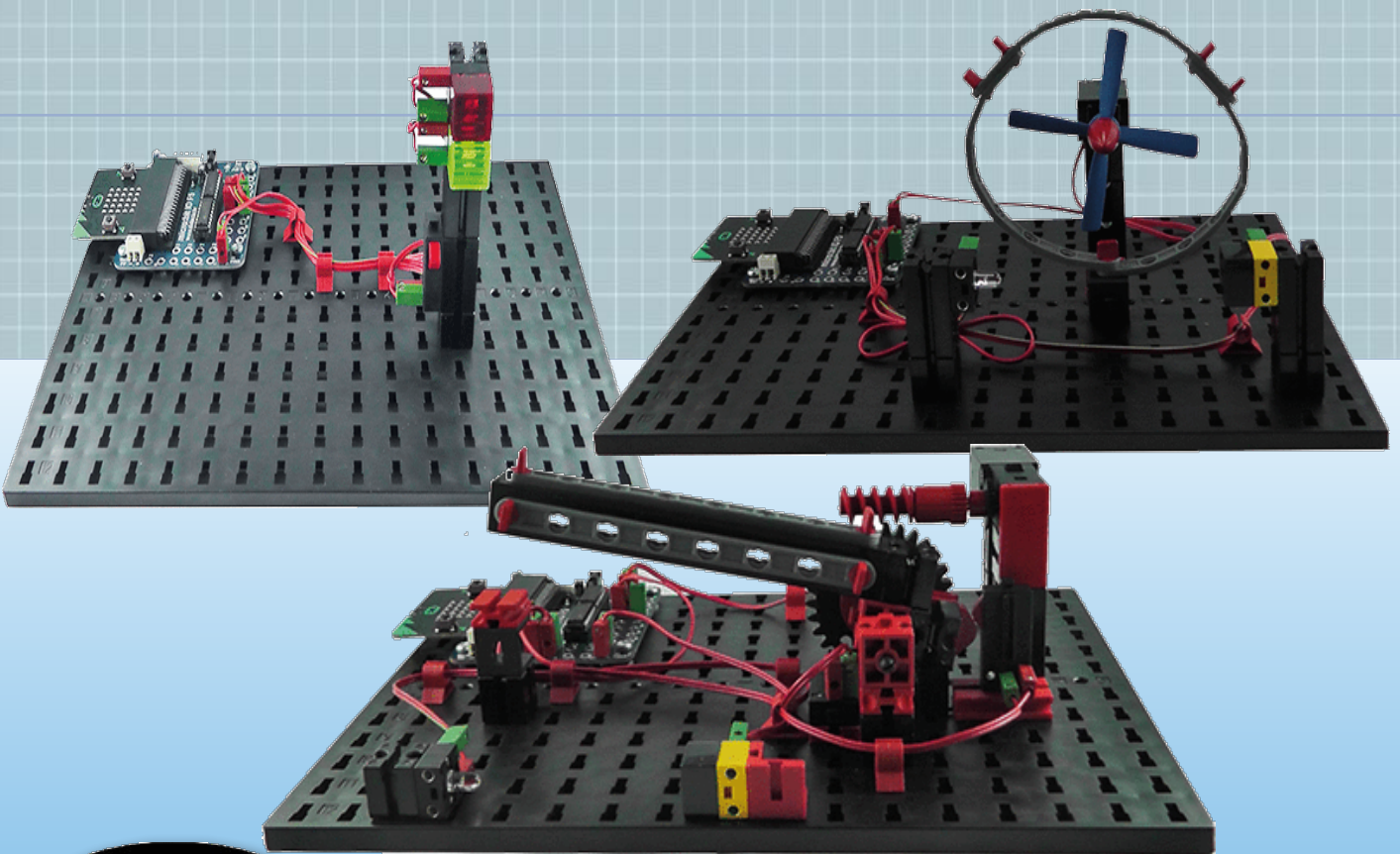
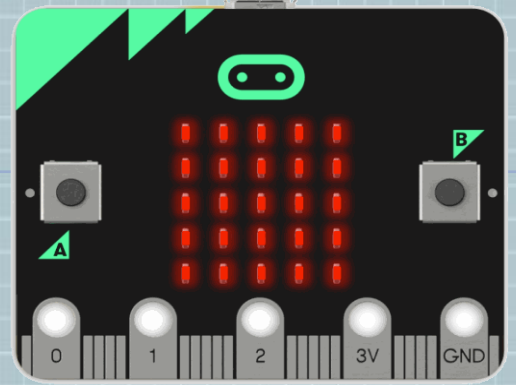


Programming in elementary school

fischertechnik 

micro:bit



3 MODELS

Contents

Preface	p. 3
Preparation	p. 4
Function test	p. 5
Editor	p. 6
Save	p. 6
Load	p. 7
Blocks/Commands	p. 7
Traffic light	p. 8
Pedestrian signal with request button	p. 9
Actuators/sensors	p. 10
Buttons	p. 10
LED	p. 11
Pedestrian signal with flashing light	p. 14
Hand dryer	p. 15
Hand dryer with light barrier	p. 16
Phototransistor	p. 17
Motor	p. 17
Hand dryer with LED display	p. 20
Barrier	p. 21
Parking garage barrier	p. 22
Parking garage barrier with optical display	p. 28
Possible errors	p. 30

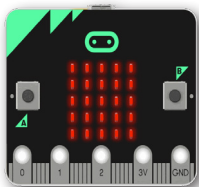
Preface



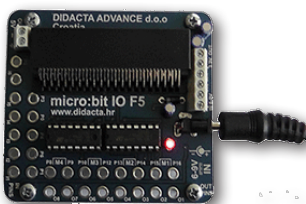
Hello!
 Allow me to introduce myself - my name is RoBo, and I'll be helping you as you work through the different exercises.

Before you start with the exercises, there are a few basic ground rules I need to explain.

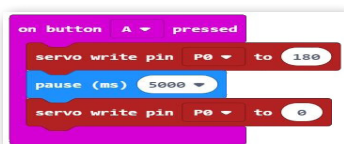
With modules from fischertechnik, you can build the models you want to control.



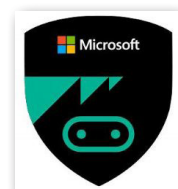
The micro:bit is a small, programmable computer. It has an LED screen, two buttons, a light sensor, a Bluetooth antenna, and an acceleration sensor and compass.



The micro:bit is connected to the micro:bit IO F5 via a male multipoint connector. All inputs and outputs can be accessed via sockets. In addition, an external power source can be connected.



With just a few clicks, you can use makecode to quickly create your own programs for the microprocessor and make the model move.



We hope you enjoy completing these exercises.

Sincerely, RoBo

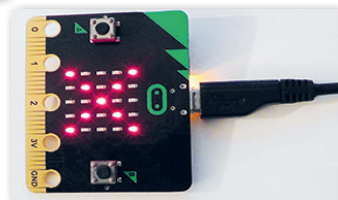
Preparation

Before you begin building the models and start programming, you need to make a few preparations.

First, unpack your micro:bit. It is delivered packaged in a small box.



Important: When you plug it into the connector strip on the adapter for your model later on, the A and B buttons have to face upward.



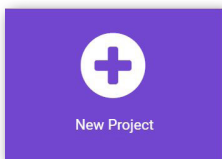
Use a USB cable to connect the micro:bit to a free USB port on your computer. A yellow LED on the bottom of the micro:bit will light up. The LED matrix on the top will display a red X.



Start your web browser and open the program editor under

<https://makecode.microbit.org>

Functional test



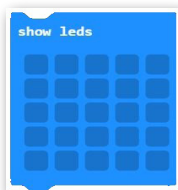
Click “New Project”. The makecode working screen will appear. I will explain it in more detail to you in the next section. Right now, you are just going to test the program in conjunction with the micro:bit.



Use “More...” to change the programming language to “German”.



Click the selection “Basic”. All commands from this block area will appear in this block.

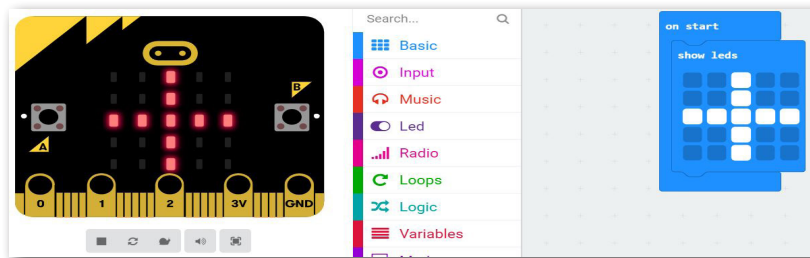
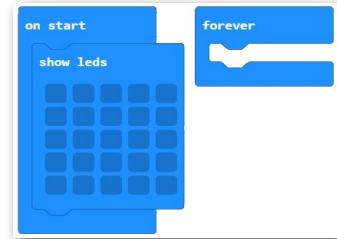


For our test, use the command “show LEDs”.

Drag the command into the empty area for the command “on start”.

When you click the individual LEDs, they will be activated and displayed in the simulation.

Create a plus sign using the LEDs. Your working screen should look like the one shown on the image.

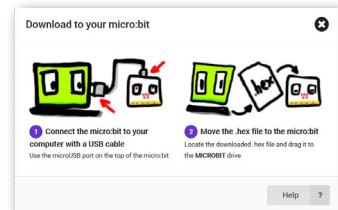


Press the “Download” button to send the program to the micro:bit. When you activate the button, the following line will appear at the bottom of the screen:

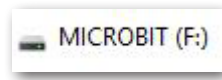


Activate the arrow behind “Save” and select “Save as”.

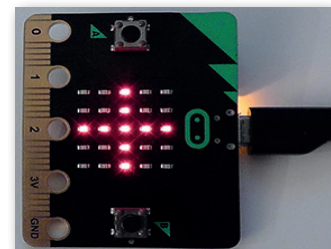
The “Save as” working screen will appear. Here, select the USB drive to which you connected the micro:bit. This is displayed visually.



Important: The file type must always be a HEX file.



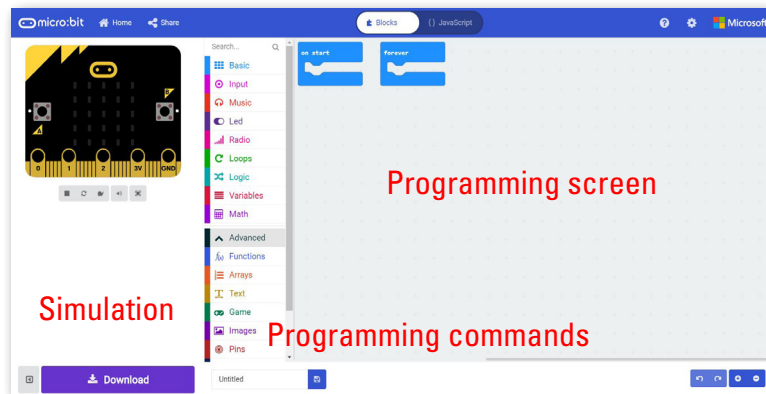
When you click “Save”, the LEDs will go out on the micro:bit and the yellow LED on the bottom will flash. This indicates that the program was transmitted. Once this is done, the plus sign will be displayed.



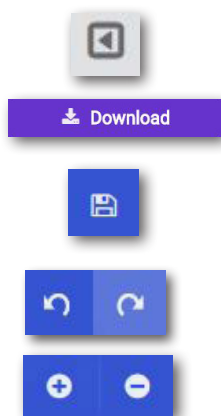
If everything worked correctly, I would now like to explain the Editor to you in a little more detail.

Editor

The screen is divided into three sections: Simulation, Programming commands, and the Programming screen.



Other important buttons:



Start/Stop the simulator

Download the program to the micro:bit

Save the project

Undo

Zoom In/Zoom Out



Stop the simulator

Restart the simulator

Slow-Mo



Mute audio

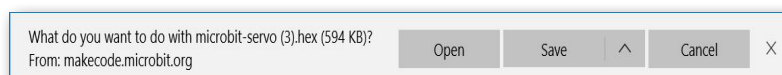


Launch in full screen

Save

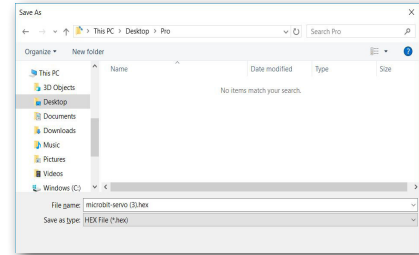


Once you have completed a project, you have to save it. To do so, select the "Save" button.



Click the arrow beside “Save” and then “Save as”.

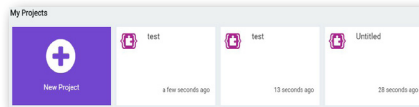
Select a folder or create a new folder where you would like to save your project.



Important: Assign a unique name to your project.



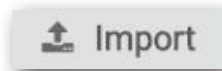
If you would like to use a finished project or continue working on a project, you have to load it from your storage medium. To do so, you need the home page. The home page appears when you launch makecode.



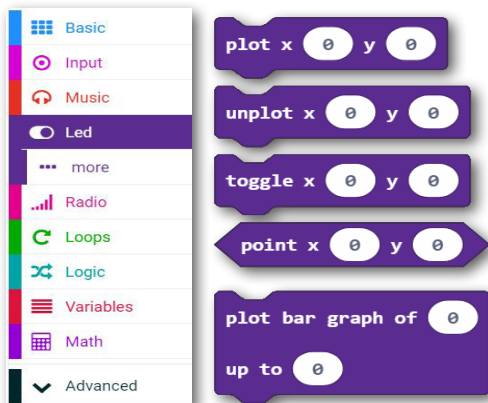
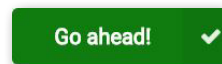
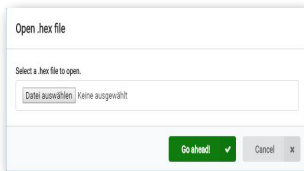
Load

You can find your projects under “My projects”.

However, you can also select the “Import” button. A context menu will appear. Here, select “Import file”.



Another context window will appear. Press the “Browse” button to access your data storage media. Once you have found your project, confirm with “Go ahead!”



The instructions use the terms “Block” and “Commands”. makecode is designed such that all commands are sorted into blocks by function. An example is provided at the left for the block “LED”.

Blocks Commands

OK, now that you are done with that, you can try out your first model.

Traffic light



You have probably seen many different kinds of traffic light systems. You see pedestrian signals or whole crossing systems almost every day, so this principle certainly isn't new to you. To put it simply, in such systems lamps are switched on and off in a certain sequence.

What will make your traffic light unique is that it will be switched on by a button located on the mast.



The technician first switches on the system using a control box. Then the red lamp will light up. Only when the pedestrian presses the request button will the system switch to green, after which they can cross the street.

The next time you are going home from school or taking a walk with your parents, and you pass a pedestrian signal, take a look at it.



Ask yourself the following questions:

- Is the system switched on?
- What lamps are illuminated?
- Does the system have a signal request button?
- What happens when it is pressed?
- How long does the green phase last?
- What happens after the green phase?



Simply write the answers down on a sheet of paper, then use them later when you write your own programming commands.

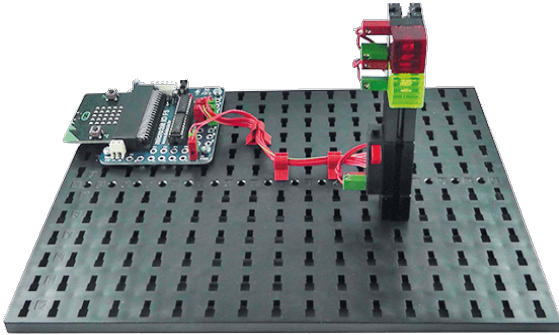
Important: Before you start operating real models, you can always test out your programs first using the programming simulator.



Technical note: You will need a fischertechnik power supply for all models. This is not included in the scope of delivery, and must be purchased from fischertechnik.



Pedestrian signal with request button

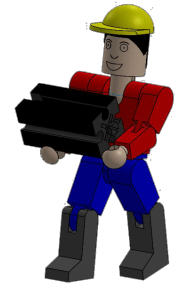


Use the building instructions to build your first model.

Wire the model according to the wiring diagram.

For this model, I am going to give you a task and a secondary task. You then

have to solve these by creating a control program.



The overall system should be switched on using a service switch, "Button A" on the micro:bit. Then the light will be red. Only when the request button on the mast is pressed will the red light switch over to green after 2 seconds. The green phase should last 5 seconds. Then the system will switch back to red and wait for the request button to be pressed once again.

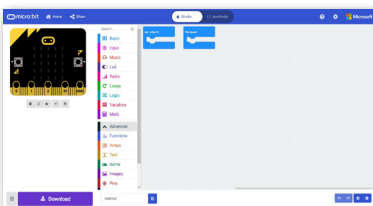
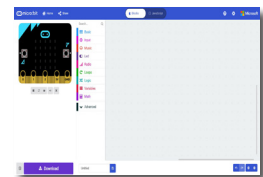
trafficlight1.hex



We will create your first project together.

Next, start the "makecode" editor. You will already be familiar with the screen that appears.

To ensure you can work with all available commands, Press "Advanced" to expand the command list.



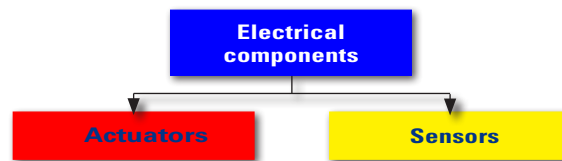
Important: Ensure that you have switched to the "Advanced" area of the makecode blocks by pressing the button.



Actuators Sensors



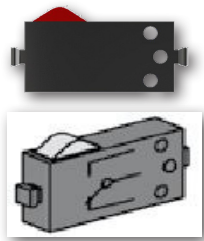
Hang on, now I need to explain the two electrical components installed in the model to you. These are called “actuators” or “sensors”, depending on their function.



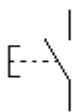
Actuators are called actuators because they are active, they do something like activate a motor or a lamp. Sensors, such as a button, can be used to control actuators.

Buttons

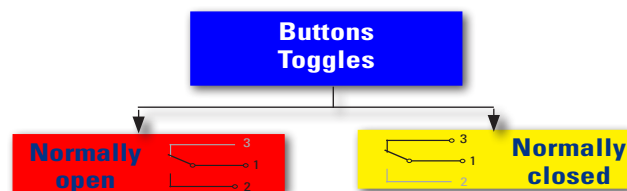
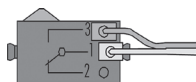
I'd like to start with the sensor button.



Buttons are called touch sensors. If you press the red button, a contact is moved mechanically within the housing and current flows between connections 1 and 3. At the same time, contact between connections 1 and 2 is interrupted. This means you can add the button to your model as a switch.



Circuit symbols



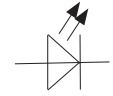
LED

Next, we have an actuator - the light diode or LED



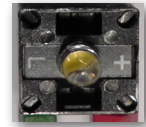
The name LED stands for light-emitting diode. When electrical current flows through the diode (forward direction), it emits light.

There are two LED modules in the construction kit. You can use them as normal lamps, or use them later as signal transducers on a light barrier.



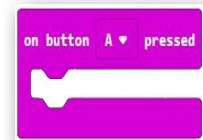
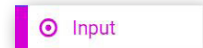
Circuit symbols

Ensure correct polarity when completing an electrical connection.



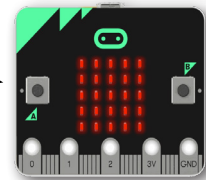
I will explain the other components to you when we get to the specific tasks.

Now we've explained everything important and you can start with your first program.



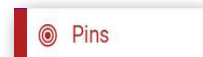
Let's start in sequence. Each program you write always starts with a "Start" command.

In the task, this means that the program is started by pressing the A button on the micro:bit.

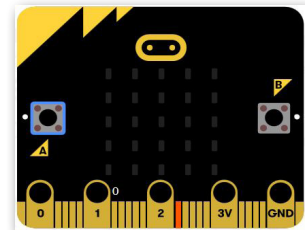
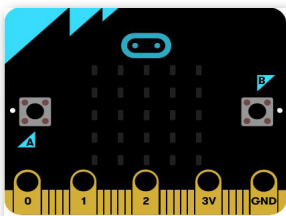


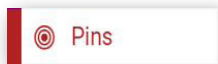
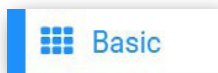
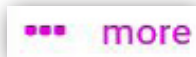
You use the command "on button ... pressed" from the "Input" block. Drag the command to your programming screen.

When the A button is pressed on the micro:bit, the red lamp (LED on PIN13) should be switched on. Dock the command from the "Pins" block in the space between "on button ... pressed". Change the input P0 to P13 and the value 0 to 1.



The first part of the program is complete. You can already see the result on your simulation. Click on the A button. Pin 1 will be switched to on, marked with a logical 1 and shown in red. If you've already connected the model to your computer, the red lamp should be illuminated.





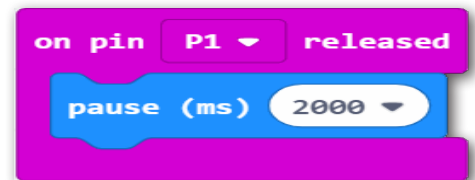
The blocks “on start” and “forever” are not required and can be deleted. Right click the command and select “Delete Blocks” in the context menu.

Next, you will create a program section with the command “on pin P1 released” embedded from the “Input” block. Drag the command to your working screen.

Change the input “P0” to “P1”.

Now a pause command comes next. You can find it in the “Basic” block “pause (ms) xxxx”.

Drag the command to the docking point and change the time input to 2000 for 2 seconds.

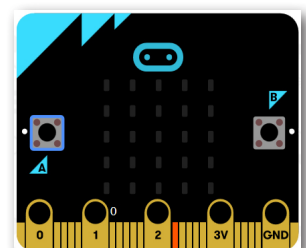


What will happen? if you press the button on “P1”, the red lamp should go out and the green lamp should come on. The red lamp is connected to “Pin13” and the green lamp to “Pin12”. Here, you need the command “digital write pin ...” from the “Pins” block. You dock it under the pause command.

Since you need the command twice, you can simply duplicate it. Right click the command and select “Duplicate” in the context menu. Dock the second command under the last input.

Change the input “P1” to “P13” in the first command and switch from “1” to “0”. In the second command, change “P1” to “P12” and “0” to “1”.

You can test the sub-program once again. Press the virtual A button to simulate it. Pin 13 will be shown in red.



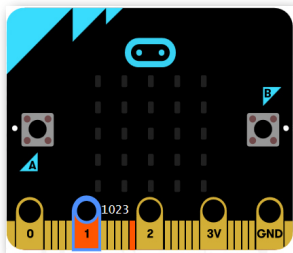
Send the sub-program to the micro:bit (see page 5).



Press the A button on the micro:bit as well. The red lamp will light up.

In the simulation, click Pin1. This will be shown as activated.

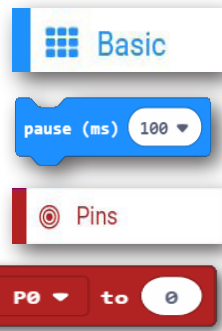
After 2 seconds, the red lamp will go out and P 13 will go out in the simulation. The green lamp or Pin 12 will switch on.



If you press the pedestrian button on the traffic light mast, the red lamp will go out and the green lamp will go on.

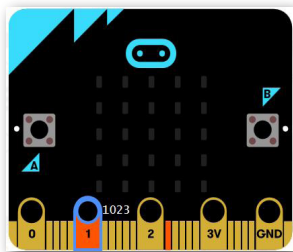
In this task, your job is to make the green lamp go off after 5 seconds, and the red lamp come back on.

Insert the command “pause (ms) xxxx” from “Basic”, followed by 2 commands from “Pins” “digital write pin”.



Change the pause time to 5000 for 5 seconds.

Change the pin assignment to “P12” and “P13”. Switch “P13” to “1” and “P12” to “0”.



Complete a virtual test first here as well. After 5 seconds of pause time, Pin12 is switched off and Pin13 will go back on.

Send the entire program to micro:bit and start it on the model. Here as well, the green lamp will switch to red after a 5 second pause.

“OK, now you’ve taken a big step. Your program is ready and you can test it out”.



Pedestrian signal with flashing light

Let's start the second part of the task.

trafficlight2.hex



After the 5 seconds is complete, the green lamp should flash three times. The flashing frequency is 1 second. Then the light is switched to red.

Loops

```
repeat 4 times
do
```

Here, you will use the command "repeat x times" from the "Loops" block. Dock it after the 5 second pause.

```
pause (ms) 5000
repeat 3 times
do
```

Basic

```
pause (ms) 100
```

Change the repetitions from "4" to "3".

Add a 1 second pause. Then the command "digital write pin P12 to 0" follows.

Pins

```
digital write pin P0 to 0
```

Then another 1 second pause command follows, and the command to switch on the P12.

```
repeat 3 times
do
  pause (ms) 1000
  digital write pin P12 to 0
  pause (ms) 1000
  digital write pin P12 to 1
```

After the write command, add a 1 second pause command.

Now the second part of the task is programmed. Save the file on your computer. Test it out virtually, then test it a second time on your model.

```
on button A pressed
  digital write pin P13 to 1

on pin P1 released
  pause (ms) 2000
  digital write pin P13 to 0
  digital write pin P12 to 1
  pause (ms) 5000
  repeat 3 times
  do
    pause (ms) 1000
    digital write pin P12 to 0
    pause (ms) 1000
    digital write pin P12 to 1
  pause (ms) 1000
  digital write pin P12 to 0
  digital write pin P13 to 1
```

Hand dryer

You definitely don't have something like this in your bathroom at home. You have a big towel hanging on a hook. But in public bathrooms, like the bathroom at your school or a restroom at a restaurant, you will often see electric blowers mounted on the wall that blow out hot air to dry your hands.

They are a great invention, especially ones that are so modern you don't even have to push a button to turn them on. Simply hold your hands in front of them and they start up.



Now, you can use these building instructions to create a hand dryer with a touchless on and off switch, then wire it according to the wiring diagram.

Ask yourself the following questions before starting this task:



- How is the hand dryer switched on?
- Does a control light switch on?
- When does the hand dryer switch back off?
- How long is the drying time?

Simply write the answers down on a sheet of paper, then use them later when you write your own programming commands.

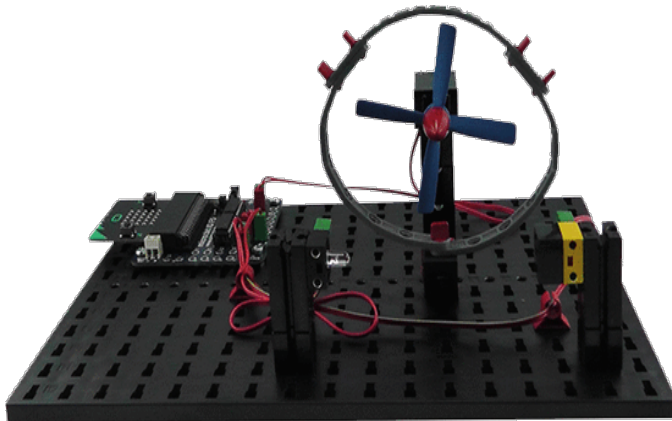


We hope you enjoy working on this task.

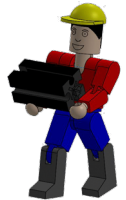
Sincerely, RoBo

Hand dryer with light barrier

Use the building instructions to build your second model.



Wire the model according to the wiring diagram.



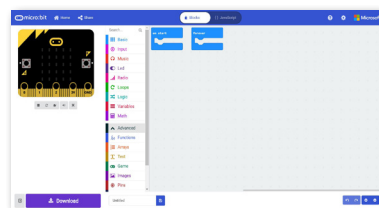
For this model as well, I am going to give you a task and a secondary task. You then have to solve these by creating a control program.

fan1.hex

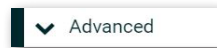


Program the hand dryer so that when a user reaches through the light barrier, the motor with propeller starts up. When they take their hand back out of the light barrier, the motor must stop.

Of course, I will be helping you with this task too.



Just as with the previous task, start your Editor first.



Important: Once again, make sure you have activated the “Advanced” block with the appropriate button.

Before you begin programming, I need to introduce you to another sensor and actuator.



To create the light barrier you will need to switch the fan motor on and off, you will need the LED you have already used, as well as a phototransistor.

A phototransistor is an electronic switch (sensor) that reacts to light. You've probably wondered how the entryway door opens automatically when you go into a department store without any buttons or switches being pressed.

A light barrier is used to open the door. It consists of a light source (transmitter) and a sensor (receiver). In the building kit, an LED component is used as the sensor, and a phototransistor component is used as the receiver.

The motor is a direct current motor (actuator). It converts electrical energy into mechanical energy. This causes the motor axis to rotate.

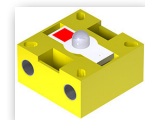
You can control the motor directly with the micro:bit.

Now you've built and wired your model. Next you need to program it.

You have already started the Editor and can start with the "on start" block. It is already shown on the screen, so you can go ahead and add another command.

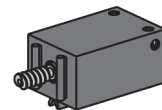
As you can see from the building instructions, the motor is connected to "Pin16" and "Pin15". Pin16 should be the negative terminal, and Pin15 the positive terminal. Because of this, Pin16 is set to "0" on start. The LED is connected to Pin8. Since it should be illuminated forever, you can add the command you need to the "on start" part of the program.

Phototransistor



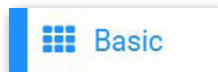
Circuit symbols

Motor



Circuit symbols





You need “forever” for the rest of the program. This command is also already on your programming screen.

Since you are working with a phototransistor, you have to convert or read in the value it delivers from a digital value to an analogue value.

Digital Analogue

Before you continue, I need to explain two terms to you: “digital” and “analogue”. What do they mean?

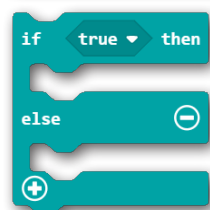
Digital values can only have two conditions, 0 or 1, yes or no. Here’s an example: A lamp is on (1) or off (0).

If you need an analogue value, it can be in a certain area and have multiple values. For instance, if you want to determine a certain temperature range or switch on a fan motor if the outdoor temperature is between 20 and 24 degrees.



First, add the command “digital write pin”. Switch “P0” to “P1”, since you have connected the phototransistor to this pin.

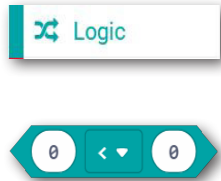
Under the “Pins” block, you will find a command “analog read pin ...”. Place it on the “0” of the first command.



Now you need a command that queries “If an event is true, then action1 should be carried out. If this is not the case (else), action2 should be carried out”.

You can find this command under the “Logic” block, under “if true then --- else”. Dock this command in your program.

I have already written the whole program for you and discovered that the photoconductive cell switches at an analogue value of 500. This value must be queried in the program. You can find the command under the "Logic" block.



```

forever
  digital write pin P1 to analog read pin P1
  if < 0 < 0 then
  else
  
```

Now add the query "if "analog read pin P1" "< smaller" "500"" to your program.

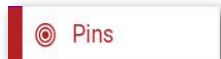


You can duplicate the command here and insert it at the right spot.

```

forever
  digital write pin P1 to analog read pin P1
  if analog read pin P1 < 500 then
  else
  
```

Insert the command "digital write pinP15" for "else". Change to "1". Duplicate the command and insert it under "else". Change to "0".



Now you have completed task 1. Save and test out the program.



```

on start
  digital write pin P16 to 0

forever
  digital write pin P1 to analog read pin P1
  if analog read pin P1 < 500 then
    digital write pin P15 to 1
  else
    digital write pin P15 to 0
  
```

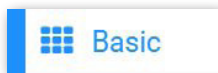
Hand dryer with LED display

Here is another task you need to solve.

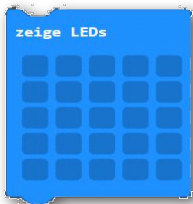
fan2.hex



There is an LED matrix on your micro:bit with 25 LEDs. You should now use it to display the current switching status of the motor. If the motor is running, an + should appear. Otherwise the matrix should be out.

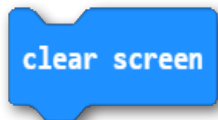


It is actually very simple. You can find the command "Show LEDs" in the "Basic" block. Drag it before the command "otherwise".

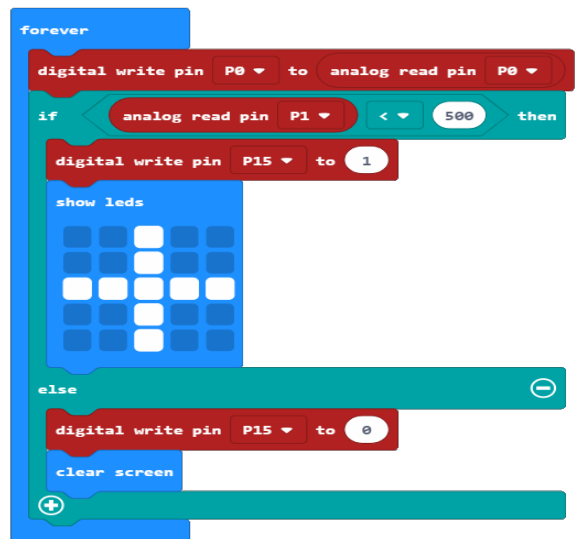
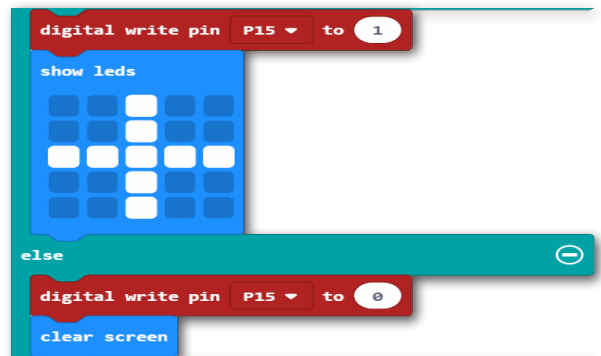


Click the relevant LEDs. These are shown in white.

Insert the command "clear screen" in "else" from the "Basic" and "...more" block.



Now you are done. Save the program once again and test it on your model.



4

Barrier

In many cities, you can find parking garages or large parking lots that use a parking barrier to control vehicles driving in or out. If the parking garage is completely full, for instance, a parking guidance system displays this. On some streets, you will see electronic displays that tell you which parking garages still have space, and which are full.



There are different ways to activate a barrier.

For instance by entering a number code, a code card, or using a light barrier like the one you will be using in your model.



Ask yourself the following questions before starting this task:

- When does the barrier open?
- Does a control light switch on?
- When does the barrier close again?
- How long is the barrier open?



Simply write the answers down on a sheet of paper, then use them later when you write your own programming commands.



We hope you enjoy working on this task as well.

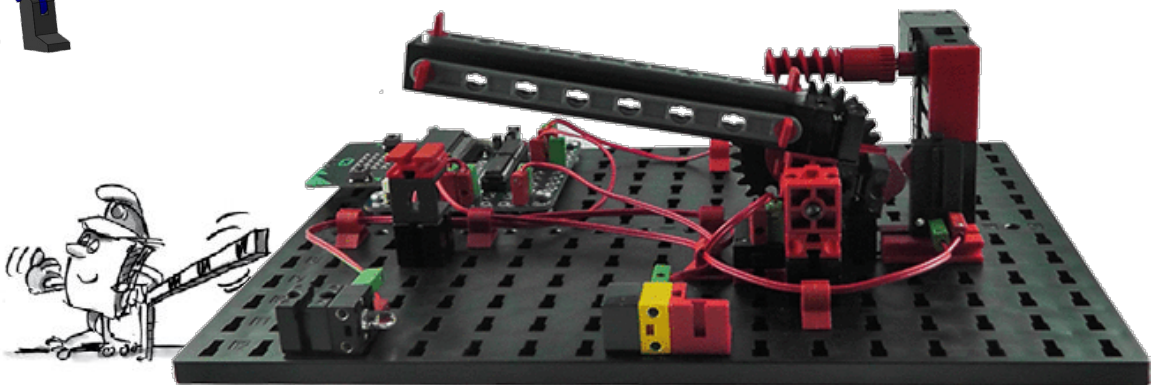
Sincerely, RoBo

Parking garage barrier



Use the building instructions to build your third model.

Wire the model according to the wiring diagram.



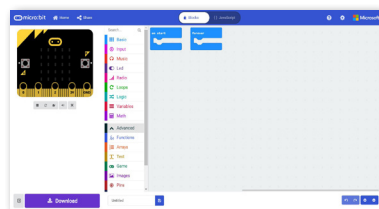
For this model as well, I am going to give you a task and, since you're already an expert programmer, three secondary tasks. You then have to solve these by creating a control program.

barrier1.hex



When a vehicle drives through the light barrier and interrupts the light current, the barrier should open after 1 second. It should remain open for 5 seconds and then close again.

Of course, I will be helping you with this task too.



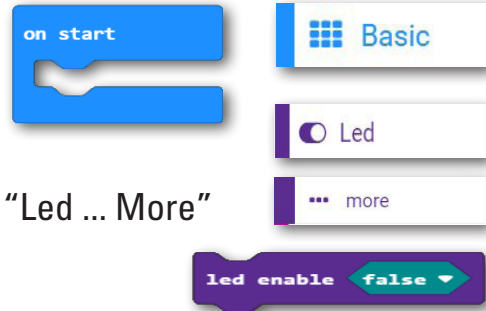
Log into the internet and start your Editor just as you did for the previous tasks.

▼ Advanced



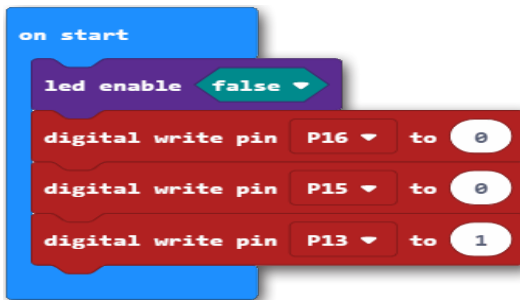
Important: Once again, make sure you have activated the “Advanced” block with the appropriate button.

Here as well, you will start with the program section "on start".



To ensure the program works correctly later on, you must first insert the command "led enable" from the "Led ... More" block.

To start the program, the motor connections on "P16" and "P15" should be set to "0".



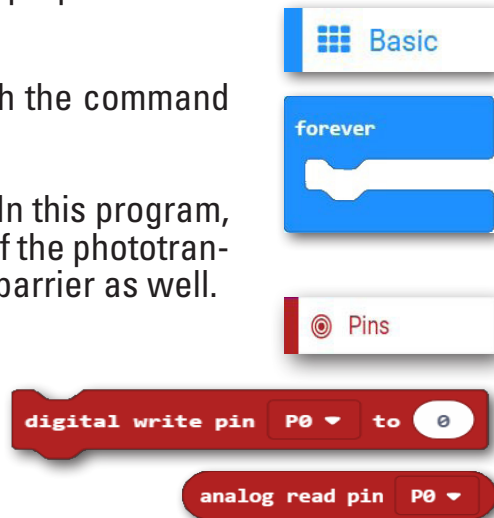
You have connected the LED for your light barrier to "Pin13". This should be illuminated after you start the program. Insert the command and change its properties.



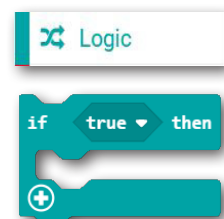
The main part of the program once again starts with the command "forever".

Look at the program for the hand dryer once again. In this program, you query the light barrier and determine the value of the phototransistor. You can use this command sequence for the barrier as well.


Just as for the hand dryer, drag the command you need into your program. The phototransistor is connected to "Pin1", just like the barrier. Change the value accordingly.



The next command is the "if ... Then" command from the "Logic" block. Drag the command for a query to the "true" rhombus. Change the value after the "<" to 500.



micro:bit



```

    forever
        digital write pin P1 to analog read pin P1
        if analog read pin P1 < 500 then
            +
    
```

If the value for the light barrier is less than 500, the barrier should open. To do so, insert "digital write pin" twice for "then". Change the first command to "P16" and to "1" and the second command to "P15" and "0".



```

    forever
        digital write pin P1 to analog read pin P1
        if analog read pin P1 < 500 then
            digital write pin P16 to 1
            digital write pin P15 to 0
        +
    
```

Loops

When the program is started, the barrier will open or the motor will start. It should run until the switch on "Pin3" is closed.



To do so, you need the command "while ... do" from the "Loops" block. Drag it under the last command.

Replace the "true" rhombus with a query from the "Logic" block.

Logic

Insert the command "digital value of ..." as the first variable. Change the pin assignment to "P3".

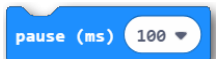


Pins

In the empty program section, insert the command "pause (ms)" with a value of 100 from the "Basic" block.



Basic



```

    forever
        digital write pin P1 to analog read pin P1
        if analog read pin P1 < 500 then
            digital write pin P16 to 1
            digital write pin P15 to 0
            while digital read pin P3 = 0
                do
                    pause (ms) 100
            digital write pin P16 to 0
        +
    
```

Test the sub-program. If you interrupt the light barrier, the barrier opens, activates the switch and stops.

In the task, your job is to ensure that when the light barrier is interrupted, the barrier opens after 1 second. Add this command to the right spot and change the wait time to 1 second.

The task also requires that the barrier close again after 5 seconds. To do so, add the command "pause (ms)" once again. Then the motor should turn in the other direction until the button on "Pin0" is pressed.



Basic

pause (ms) 100

```

pause (ms) 5000
digital write pin P16 to 0
digital write pin P15 to 1
while digital read pin P0 = 0
do
  pause (ms) 1
digital write pin P15 to 0
  
```

Simply duplicate the commands you need from the first part of the program.

Change the value of "P16" to "0" and the value of "P15" to "1". Change the value of the button from "P3" to "P0".

To ensure the barrier remains closed, you also need to change the last command from "P16" to "P15".

```

forever
  digital write pin P1 to analog read pin P1
  if analog read pin P1 < 500 then
    pause (ms) 1000
    digital write pin P16 to 1
    digital write pin P15 to 0
    while digital read pin P3 = 0
    do
      pause (ms) 1
    digital write pin P16 to 0
    pause (ms) 5000
    digital write pin P16 to 0
    digital write pin P15 to 1
    while digital read pin P0 = 0
    do
      pause (ms) 1
    digital write pin P15 to 0
  
```



Test the entire program. If you interrupt the light barrier, the barrier opens, activates the switch and stops. The barrier closes again after a specified wait time.

What happens if the barrier is open when the program starts? You would need to close it manually. But why don't we do this by adding to our program?

barrier2.hex



If the barrier is open when the program starts, it should close first.

Duplicate the program block into the start program from the "forever" section.

```
digital write pin P16 to 0
digital write pin P15 to 1
while digital read pin P0 = 0
do
  pause (ms) 100
```

Now you can test out the program. Mechanically move the barrier to a middle position. After the program has been downloaded, first the barrier will close, and then process the "forever" part of the program.



```
on start
  led enable false
  digital write pin P13 to 1
  digital write pin P16 to 0
  digital write pin P15 to 1
  while digital read pin P0 = 0
  do
    pause (ms) 100
  digital write pin P15 to 0
```

Now, the entire program is as follows.

```

on start
  led enable false
  digital write pin P13 to 1
  digital write pin P16 to 0
  digital write pin P15 to 1
  while digital read pin P0 = 0
  do pause (ms) 100
  digital write pin P15 to 0

forever
  digital write pin P1 to analog read pin P1
  if analog read pin P1 < 500 then
    digital write pin P16 to 1
    digital write pin P15 to 0
    while digital read pin P3 = 0
    do pause (ms) 100
    digital write pin P16 to 0
    digital write pin P15 to 0
    pause (ms) 5000
    digital write pin P16 to 0
    digital write pin P15 to 1
    while digital read pin P0 = 0
    do pause (ms) 100
    digital write pin P15 to 0
  
```

Save the program on your computer. Use a new name, such as Barrier2.



I have another idea. What if you added a service switch (button A) to the system so you can use it to start the system?



Add a program section you can use to start the system.

barrier3.hex

```

on start
  while true
  do
  
```

Of course, you will add the command you need to the "on start" part of the program. Insert the command "while ... do" from the "Loops" block before LED activation.

Replace the "true" rhombus with the command "not" from the "Logic" block.

To query button A, use the command "Button A is pressed" from the "Input" block.

Add a duplicate of "pause (ms) 100" to the space for "do".

- Loops
- while true
- do
- Logic
- not
- Input
- button A is pressed

```

on start
while not button A is pressed
do
  pause (ms) 100
led enable false
digital write pin P13 to 1
digital write pin P16 to 0
digital write pin P15 to 1
while digital read pin P0 = 0
do
  pause (ms) 100
digital write pin P15 to 0
  
```

Now you can test out the program. Move the barrier to a middle position. After the program has been downloaded, it will wait for button A. Once it is pressed, first the barrier will close, and then process the “forever” part of the program.

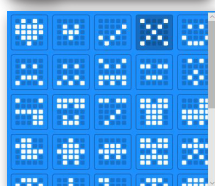
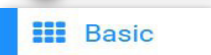
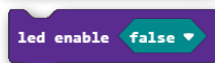


Save the program on your computer. Use a new name, such as Barrier3.

Parking garage barrier with optical display

I have another programming addition for you.

barrier4.hex



The position of the barrier should be indicated visually on the LED field. When the barrier is closed, an X will appear, and when it is open a check.



First, add the command “led enable” from the LED block to the “Start block”.

Dock the command using the “while ... do” command and change the value from “false” to “true”.

Then insert the command “showicon” from the “Basic” block. Click the arrow and select the display of an X. This command sequence is used to switch the barrier to closed on start.

```

led enable true
show icon X
while digital read pin P0 = 0
  
```



```

digital write pin P9 to 0
show icon [grid icon]
pause (ms) 5000
digital write pin P16 to 0
digital write pin P15 to 1
show icon [check icon]
    
```

You need the command “show icon” in two places in the “forever” program section. Once when the barrier is open and then once again when it is closed. Insert the command and change the LED display.

```

Basic
show icon [grid icon]
    
```

Test the program with the addition. Move the barrier to a middle position. After the program has been downloaded, it will wait for button A. Once it is pressed, the graphic display (X) will appear and the barrier will close, and then it will process the “forever” part of the program. If the light barrier is interrupted, the barrier will open and switch the LED display to the check for 5 seconds. When the barrier is closed once again, the LED display also changes.



Save the program on your computer. Use a new name, such as Barrier4.

```

on start
while not button A is pressed
do
pause (ms) 100
led enable false
digital write pin P13 to 1
digital write pin P16 to 0
digital write pin P15 to 1
led enable true
show icon [X icon]
while digital read pin P0 = 0
do
pause (ms) 100
digital write pin P15 to 0

forever
digital write pin P1 to analog read pin P1
if analog read pin P1 < 500 then
digital write pin P16 to 1
digital write pin P15 to 0
while digital read pin P3 = 0
do
pause (ms) 100
digital write pin P16 to 0
digital write pin P15 to 0
show icon [grid icon]
pause (ms) 5000
digital write pin P16 to 0
digital write pin P15 to 1
while digital read pin P0 = 0
do
show icon [check icon]
pause (ms) 100
digital write pin P15 to 0
    
```


If something isn't working right ...

... hopefully you can find a solution for your problem in this table.

Problem	Possible cause	Fault correction
1. makecode software does not connect to the micro:bit	USB cable is not connected	Insert USB cable
2. Button does not work	Electric plug inserted on the wrong connections of the button or micro:bit.	Use connections 1 and 3 on the button. Insert the micro:bit plug onto the two sockets for I5, I4 or I1.
3. Phototransistor does not work	Electric plug plugged in incorrectly	On phototransistor: Insert red plug on the side with the red dot, green plug on the side without a marking.
	Light barrier LED does not light up.	Connect LED to I1 and P1, ensuring correct polarity
	LED illuminates but misses the phototransistor	Move the LED so that it strikes the phototransistor
4. Motor does not turn	Motor not connected to the micro:bit	Connect the motor to the micro:bit as described in the wiring diagram for the model
	Motor connected to the wrong motor output on the micro:bit	Use the wiring diagram to check which output M1 of the motor belongs on and connect it to this output
5. Motor turns in the wrong direction	Red and green exchanged on electric plugs	Exchange red and green plugs on the motor
		Change the direction of rotation for the motor in the control program Pin15/Pin16
8. Problem not described here	Not found	Contact fischertechnik directly, for instance at: www.fischertechnik.de

Finally, I would like to give you an important web address. If you want to learn more about the micro:bit, you can find further information at

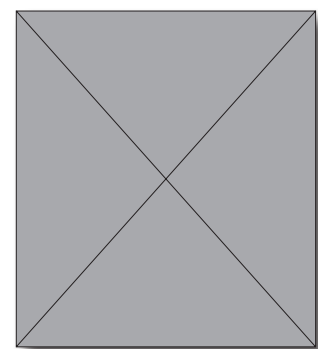
<https://microbit.org>



Well, it seems we've reached the end. I wish you lots of success programming the ft models with makecode.

1st edition 2019

fischertechnik GmbH Tel: (+49) 7443 12 - 4369
Klaus-Fischer-Strasse 1 E-Mail: info@fischertechnik.de
72178 Waldachtal



Hermann Weininger,
Head teacher and
master electrician