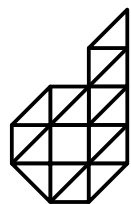

ProgBlox

Plus

programming manual



didacta
advance
didacta.hr

fischertechnik 



The goal is to learn to solve more complex problems by breaking them down into parts (objects) and with logical thinking to learn the basic steps of programming, construction and robotics.

Content:

Connecting the tablet and the control module	1
Tablet	2
Assembling the program on the tablet	3
Control module	4
We learn object programming (introduction)	5
Example 1 - LED light	6
Example 2 - LED light endlessly	7
Example 3 - button and LED light	8
Example 4 - button and LED light + Example 2	9
Example 5 - two buttons one LED light	10
Example 6 - two buttons and two LED lights	11
Example 7 - photo sensor and LED light	12
Example 8 - button and buzzer	13
Example 9 - DC motor control via buttons	14
Example 10 - hair dryer (PHOTO sensor)	15
Example 11 - connection diagram for a vehicle	16
Example 12 - IR sensors	17
Example 13 - the vehicle follows the line	18
Example 14 - vehicle inside the parcour (IR sensor)	19
Example 15 - LED lights - ADVANCED	20
Example 16 - vehicle inside parcour (IR sensor) - ADVANCED	21
Example 17 - COLOR sensor	22
Save the program and restart	23
Example 18 - main program on two sides of the tablet	24
Command cubes - group 1	25
Command cubes - group 2	26
Command cubes - group 3	27

Connecting the tablet and the control module

ProgBlox Plus

In order for the control module and the tablet to be able to communicate, they need to be paired.
This procedure is done only once, for one tablet and module.

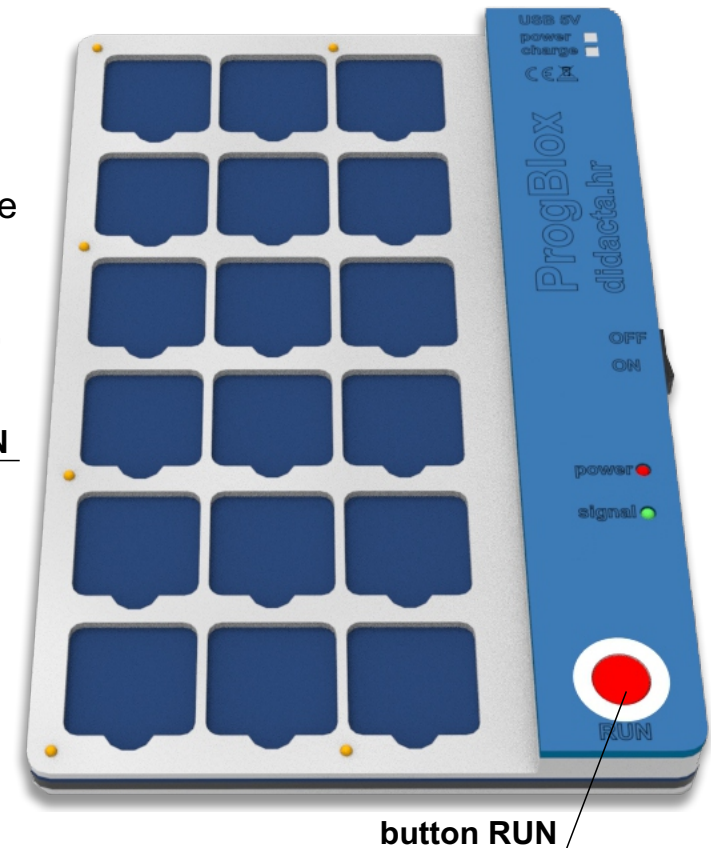
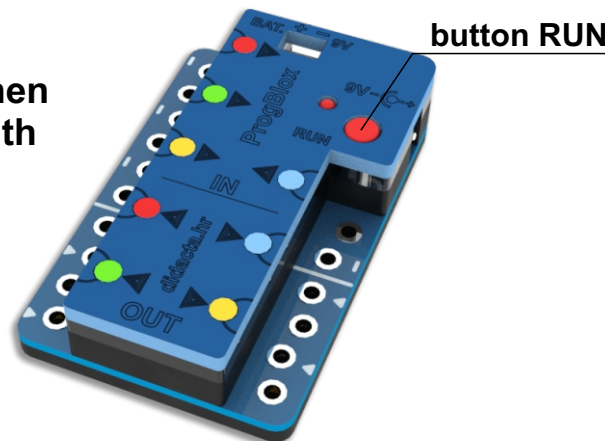
If you want to use the control module with another tablet, you must repeat the process with another tablet.

You can pair one module with only one tablet, but you can pair one tablet with several modules.

1. Press the RUN button on the control module and connect the power supply (LED lights started to turn on and off quickly).
2. Press the RUN button on the tablet and turn it on. Hold the RUN button (on the tablet) pressed for as long as the LED light on the control module does not turn off (devices are paired).

The next time you turn on the tablet, to work with the control module, it is necessary to **hold down the RUN button on the tablet when switching on.**

If the RUN button is not pressed when switching on, the tablet is paired with the robot car from Car Set.

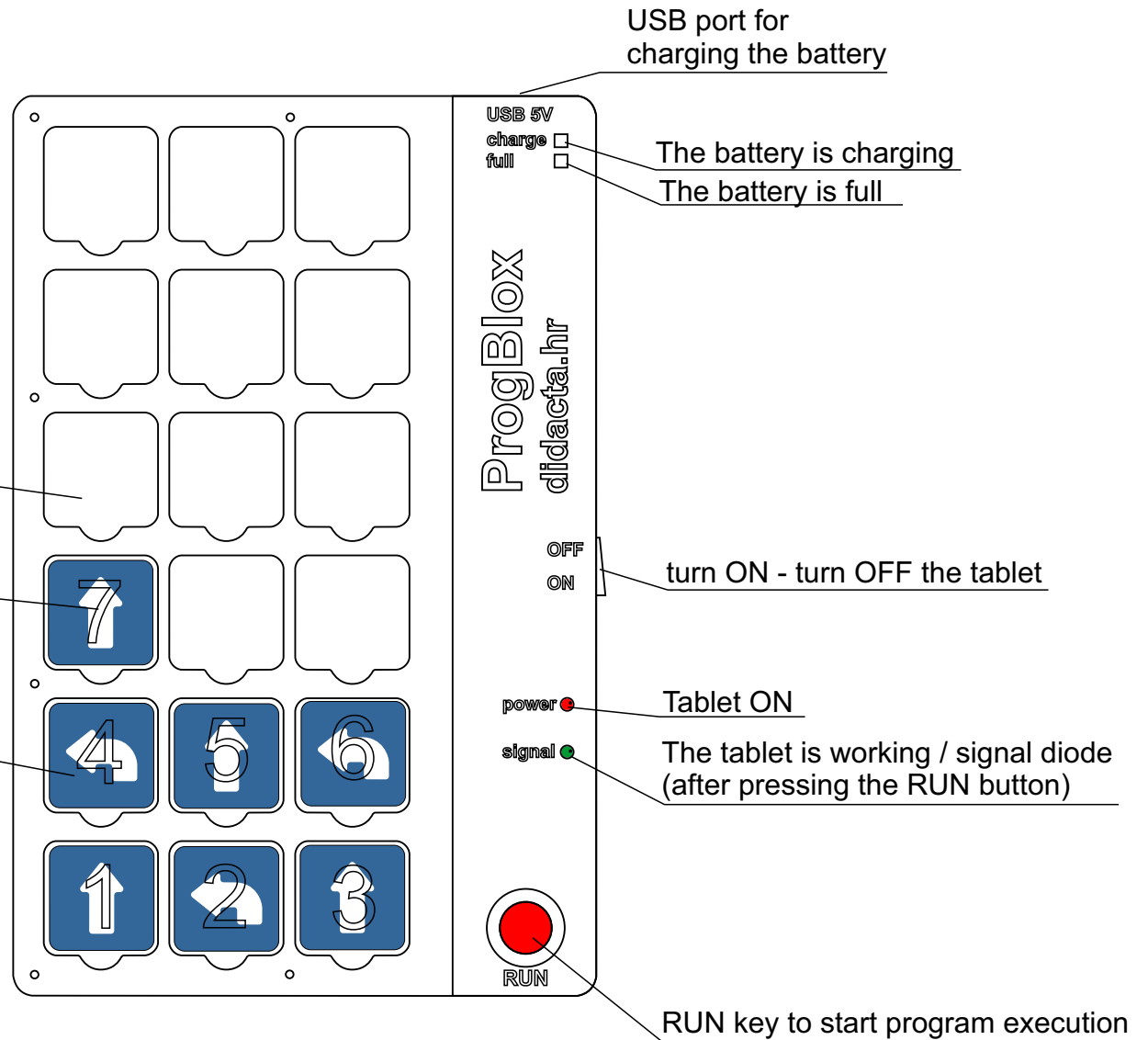


Tablet

- working time more than 4 hours
- **low battery** - signal (green) LED is constantly lit - the battery needs to be charged
- charging the battery via USB cable 5V and mobile phone charger

program fields
the numbers show the program stacking sequence

command cubes
- empty fields within the program do not affect the program



Assembling the program on the tablet

For management through the control module, we use two types of programs:

Main program:

- can be run only once or repeated infinitely
- cannot contain command cubes for input sensor control

Subprograms (object - for INPUT control):

- for each input there can be two subprograms - according to the state of the sensor connected to that input
- one subroutine for the condition when there **is a signal** and another for the condition when there is **no signal**
- each subroutine has a command cube at the beginning to control the state of the input
- the subprogram is started by changing the state of the sensor (there is or is no signal)


MULTIPLE subprograms can be stacked on the tablet at the same time, but ONLY ONE main program.

A single main program and subroutines create a driver program.

After the program or subprogram has been assembled, to activate it, press the RUN button on the tablet.

A red **LED** light on the control module will signal that the program has been loaded into the module's memory and started.

The programs remain in the working memory of the control module until the power supply to the module or the programs are interrupted they don't wipe.

Erasing the program from the working memory can be done via the command cube  or with a short press on the RUN button of the control module.

If necessary, programs can be saved in the permanent memory of the module, where they remain saved even after shutdown modules. Only one driver program can be stored in permanent memory. By saving a new driver program, previously saved ones are deleted.

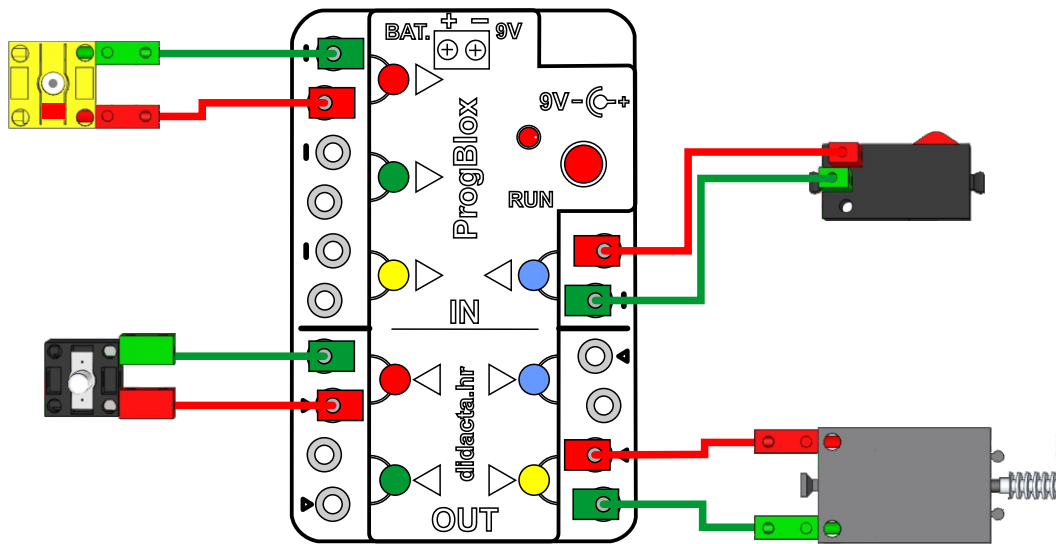
To run an assembled program on the tablet, it is necessary to press the RUN button  on the tablet.

Control module

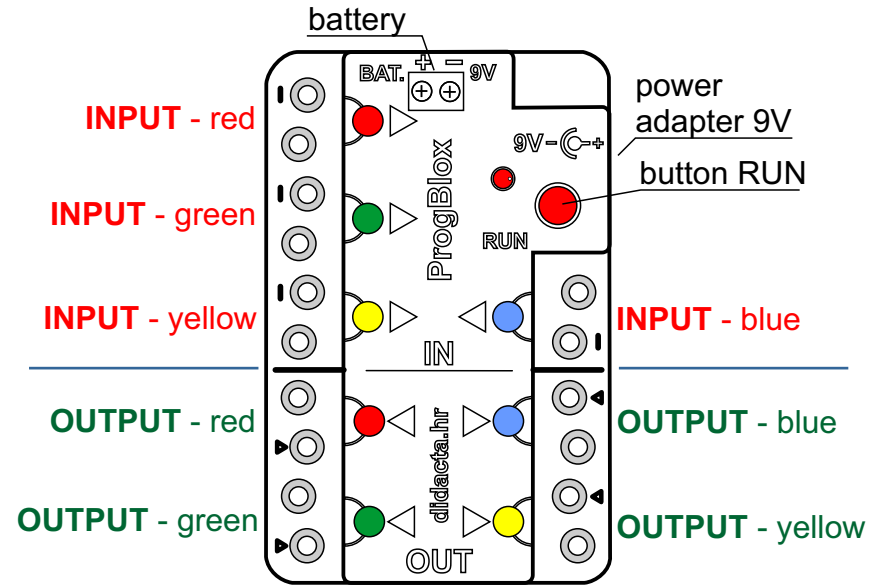
Power supply via 9V adapter or 9V battery.

- 4 x INPUTS for connecting different sensors
 - input signal up to 9V
 - (button, photo sensor, thermal, magnetic, infrared, ...)
- 4 x OUTPUTS for connecting different devices
 - output voltage around 9V (depending on the input voltage)
 - up to 1.5 A per output
 - (DC motor, LED diode (9V), electromagnet, ...)

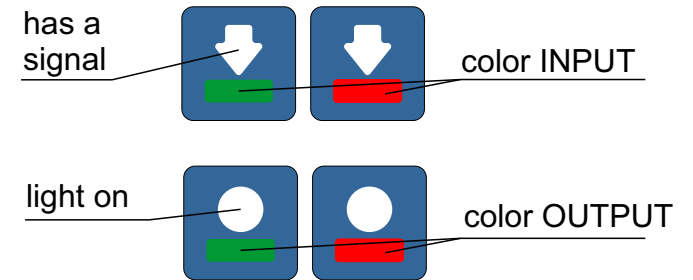
How to connect different input or output devices (TO WORK PROPERLY)



ProgBlox Plus



Command cubes refer to color INPUT or OUTPUT of the control module.



We are learning object programming

Each active part (switch, photosensor,...) of the model that is connected to some input of the interface is an input object, and the object can be a model as a whole (for example, a vehicle). In the main program, we define all actions related to the main process or object (for example, a vehicle). For all other input objects, processes are defined through individual subprograms.

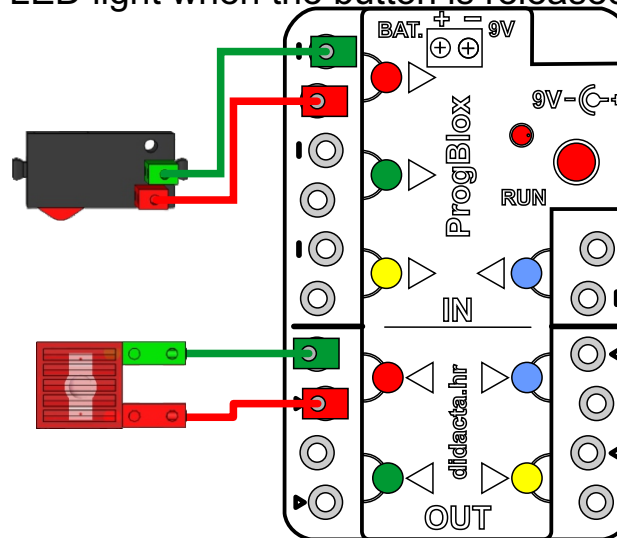
The first example of object programs:

The model has one button (input object) connected to the red input and one LED connected to the red output. The button object can have **two states, positive** (pressed) and **negative** (not pressed).

If we want the LED light to turn on when the button is pressed, and turn off when the button is not pressed, we have to make two subroutines. The first subprogram turns on the LED light (connected to the red output) when the button is pressed (the input has a signal), and the second subprogram turns off the LED light when the button is released (the input has no signal).

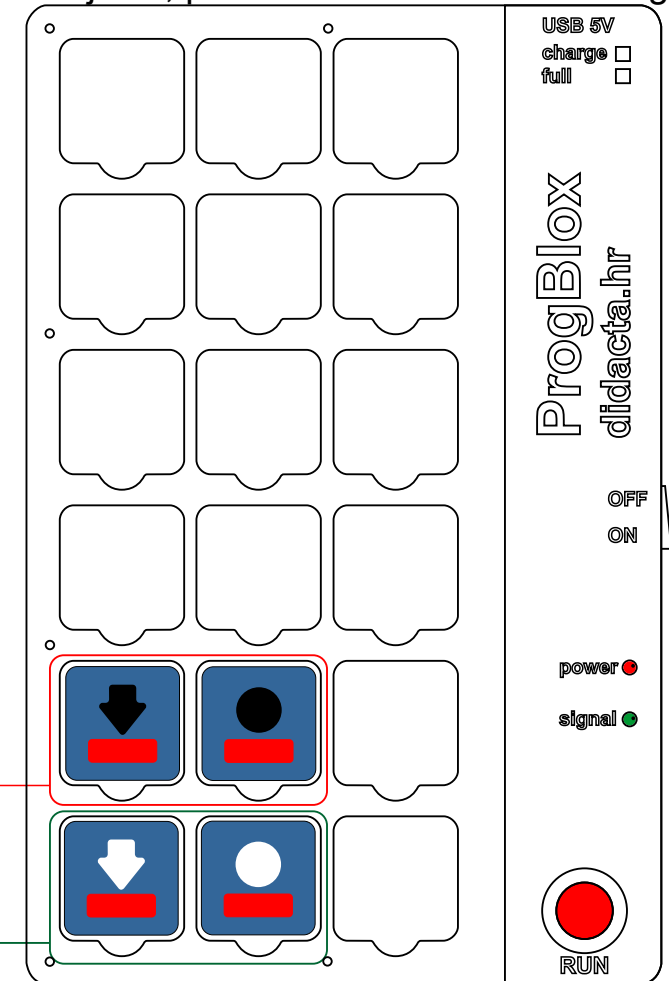
After compiling the program, press the RUN button on the tablet.

Try pressing the switch.




2. subroutine (negative):

1. subroutine (positive):



Example 1 (MAIN PROGRAM):

An example of LED light control via the MAIN program. By pressing the RUN button on the tablet, the LED light turns on for one second and then turns off.

After the exercise, delete the program that is in the working memory, via command cube 



turn on the red LED light



wait a moment (1 sec.)

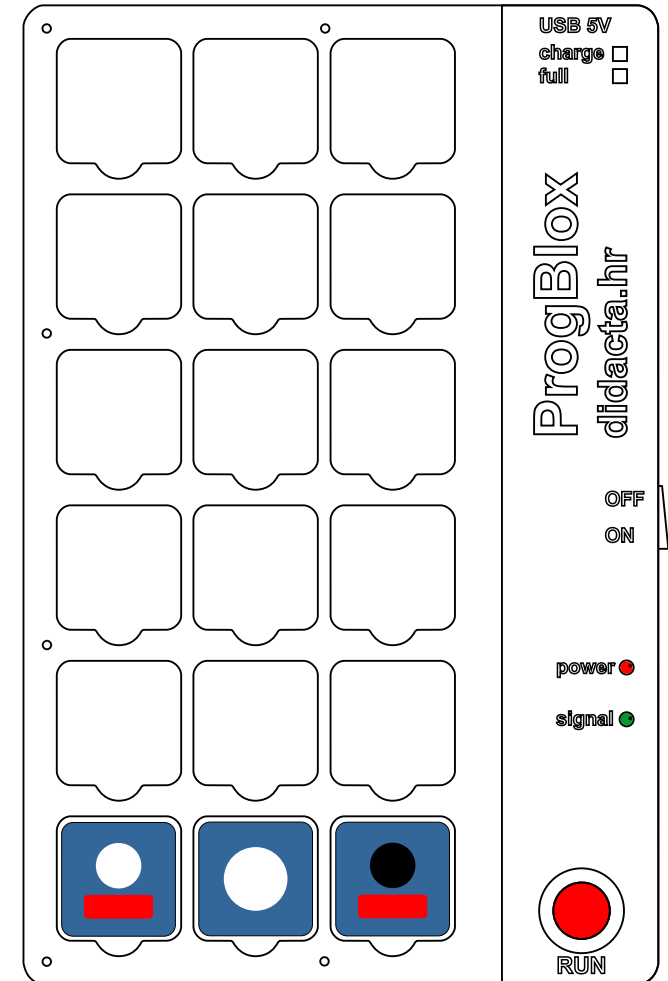
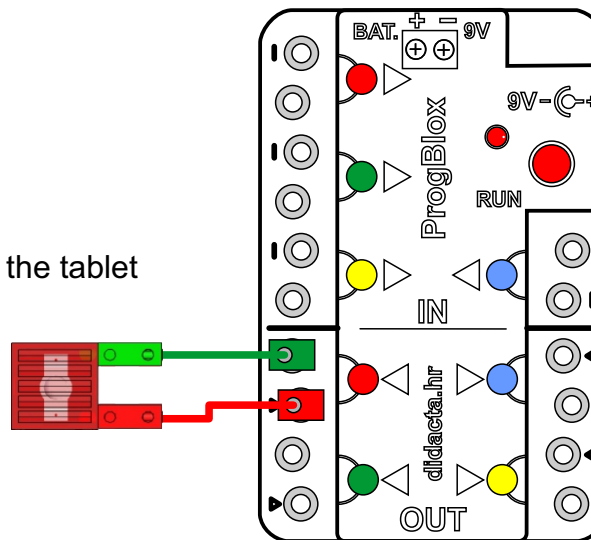


turn off the red LED light



press the RUN button on the tablet






LED on the control module will light up briefly (the program is started) , then it will light up RED LED light. One second later it will turn off.

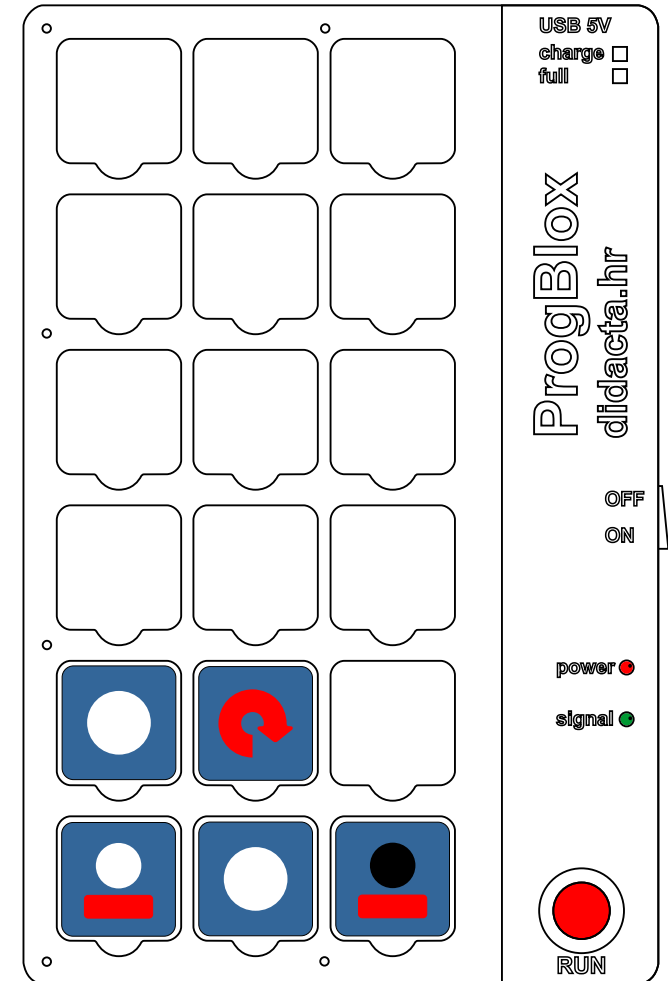
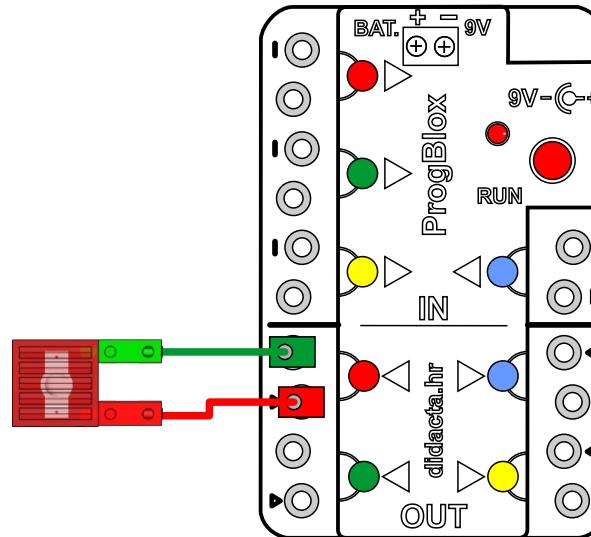



Example 2 (MAIN PROGRAM) - ENDLESSLY

An example of LED light control via the MAIN program.

By pressing the RUN button on the tablet, the LED light turns on for one second and then turns off for one second. The process is repeated Infinity.

-  turn on the LED light
-  wait a moment (1 sec.)
-  turn off the LED light
-  wait a moment (1 sec.)
-  repeat endlessly







Why do we have to put the command cube  after turning off the LED light?

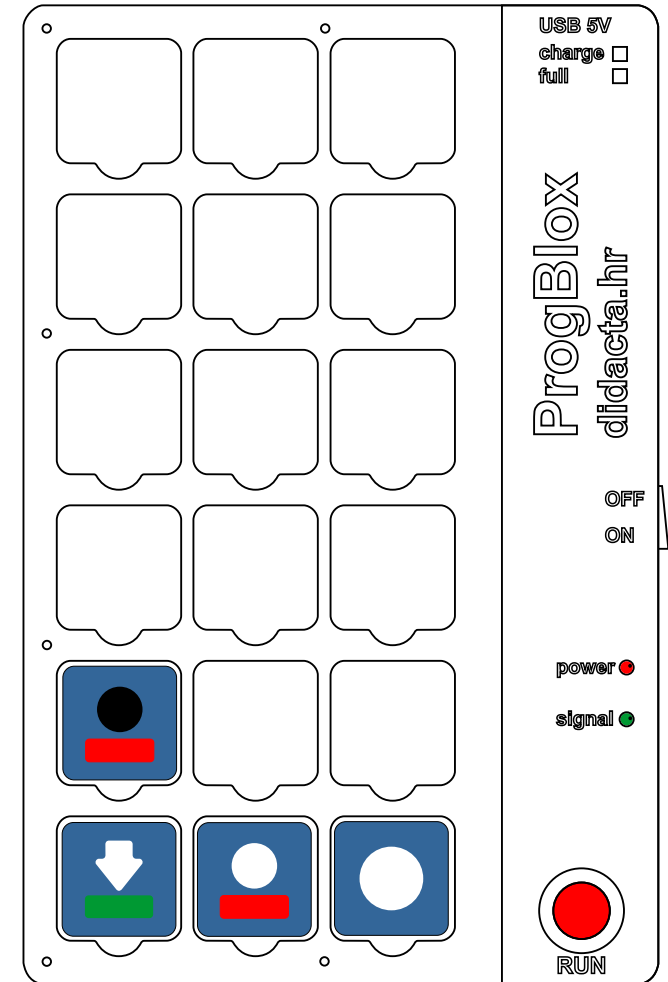
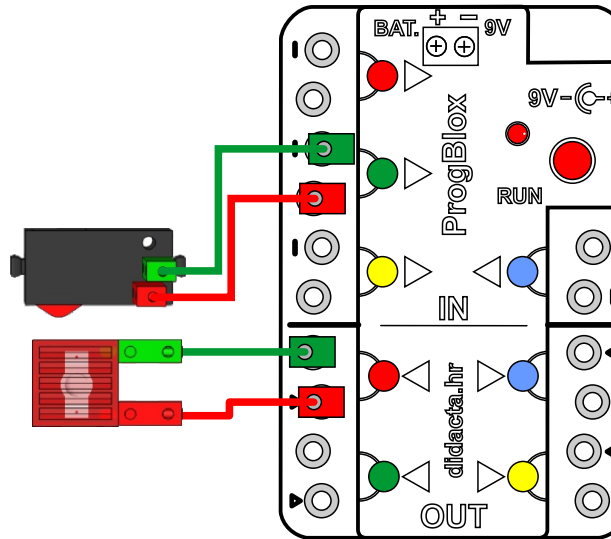
Example 3

One subroutine to control the positive state of the switch.

Add a switch to the previous exercise.

An example of LED light control via switch.

-  pressing the switch (green input)
-  turn on the red LED light
-  wait a moment (1 sec.)
-  turn off the red LED light



Example 4

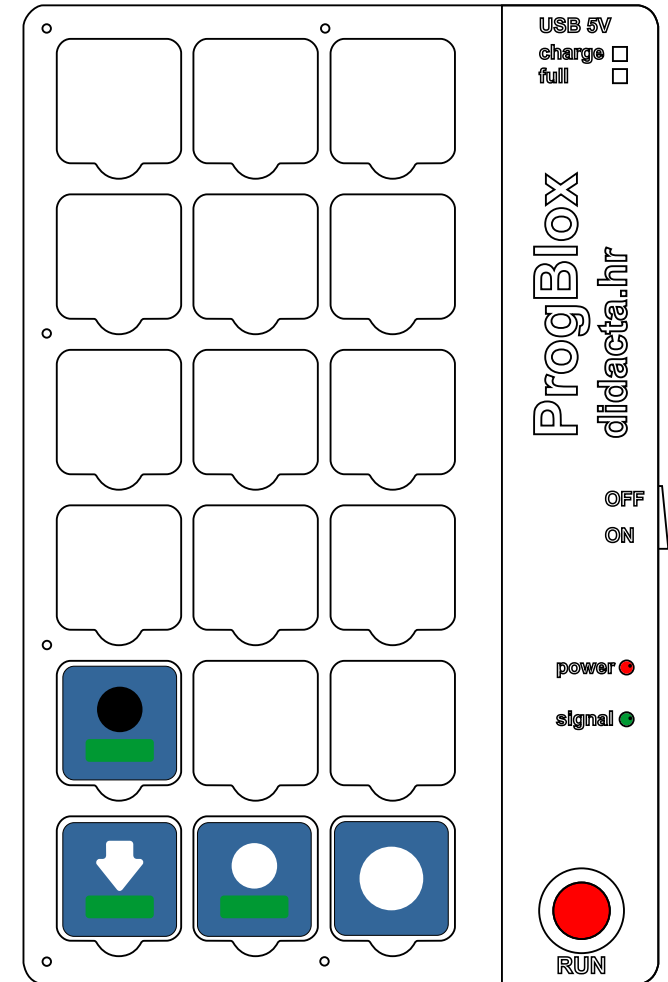
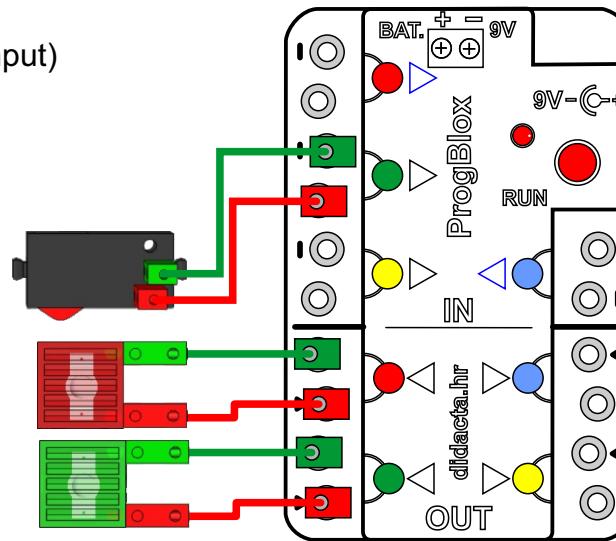
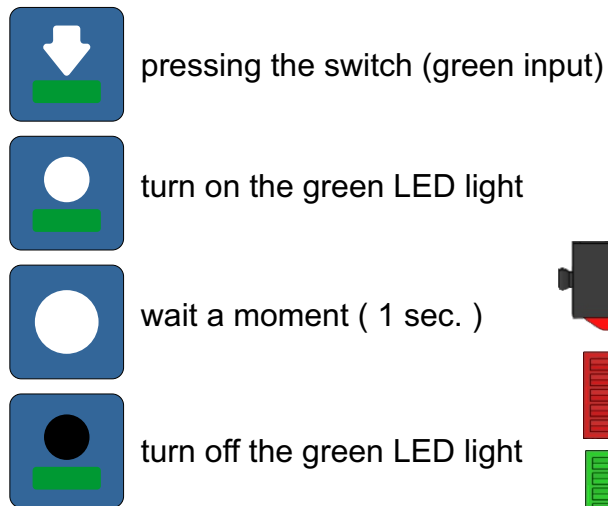
One subroutine to control the positive state of the switch.

An example of LED light control via switch.

You can run the program from Example 2 before or after this subroutine.

Add a green LED light to the previous model.

Try running the main program from **Example 2** with this subprogram.

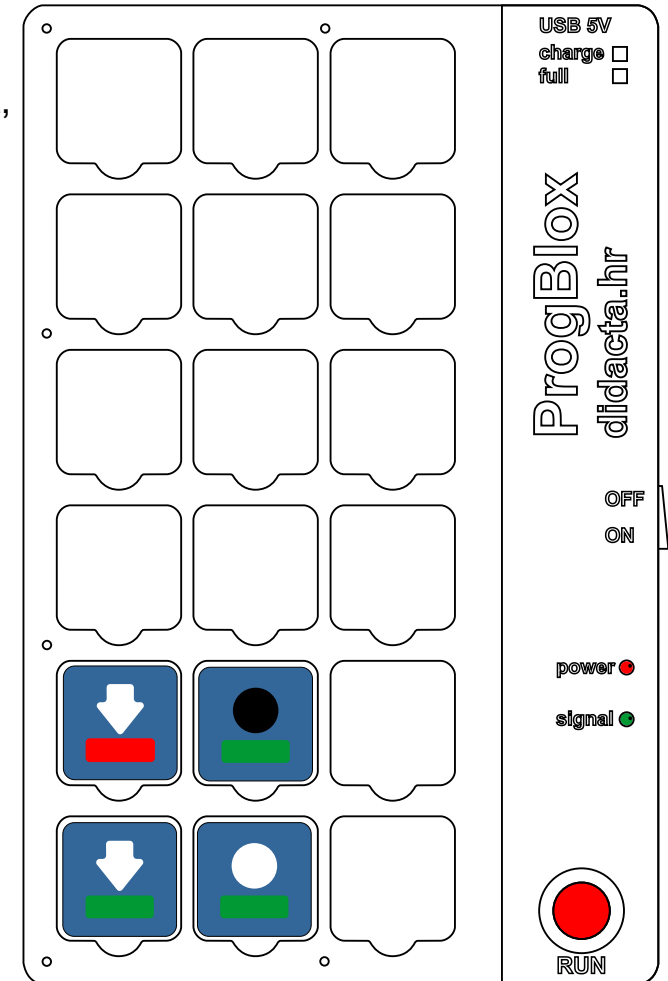
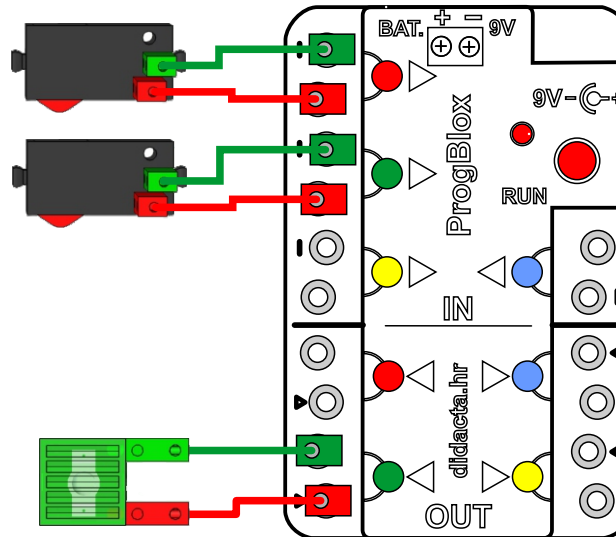
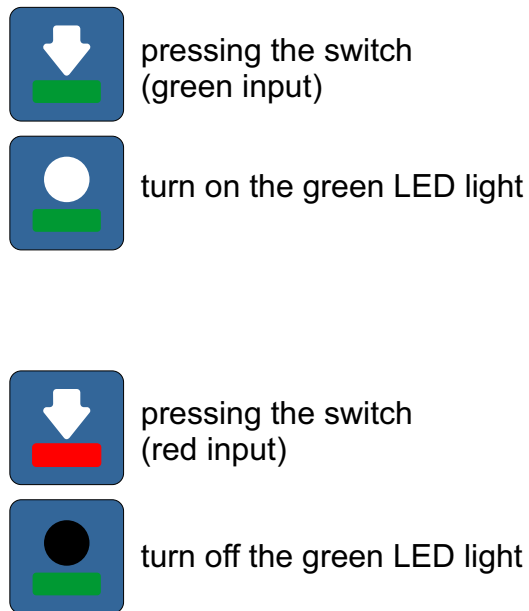


Example 5

Two subroutines to control the positive state of two switches.







LED light control via two switches.

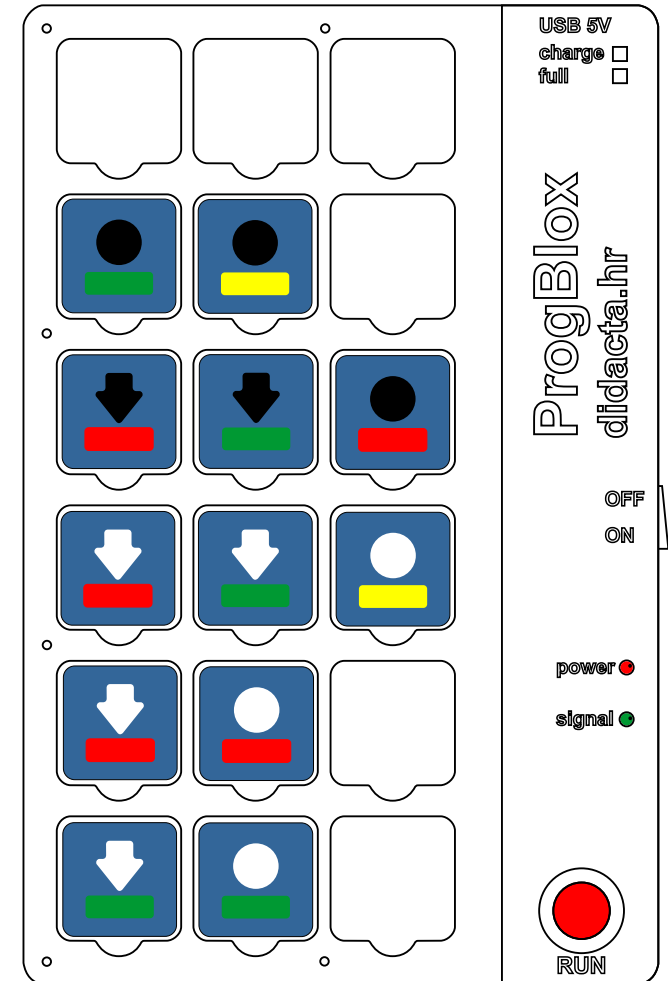
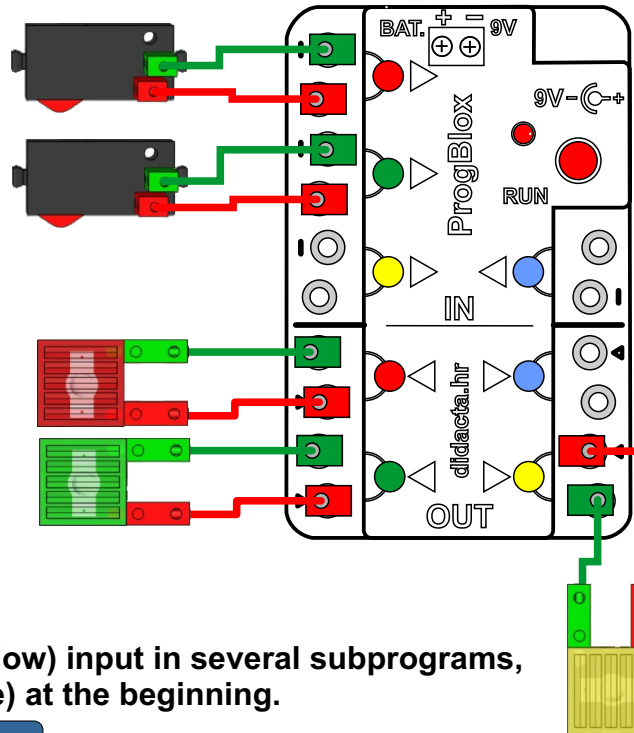
Pressing the switch connected to the green input turns on the green LED light, and pressing the switch connected to the red input turns off the green LED light.



Example 6

Four subroutines. Three for positive button states and one combined for negative state.
LED light control via two switches.

-  by pressing a switch (green input) lights up green LED light
-  by pressing a switch (red input) lights up red LED light
-   by pressing both switches lights up yellow LED light
-   if both switches are not pressed, all LED lights are turned off



NOTE: When you combine the same (yellow) input in several subprograms, put the non-repeating input (red and blue) at the beginning.



For controlling two inputs

simultaneously you can only use two positive



or



two negative combinations.

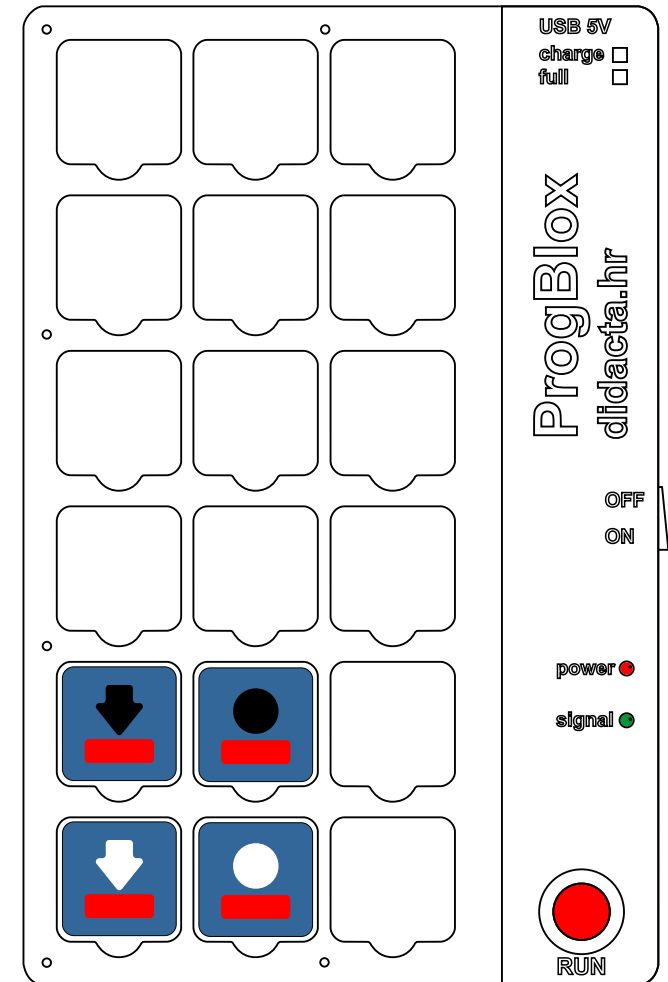
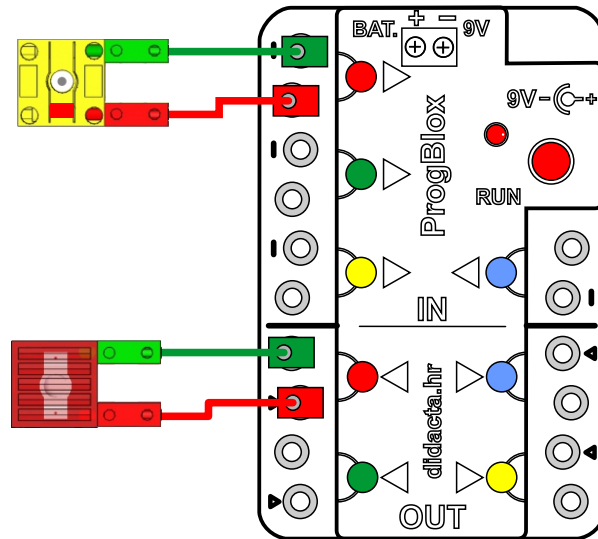
Example 7

Two subroutines.

LED light control via photo sensor.

For practice, you can replace the photo sensor with another sensor (magnetic, thermal,...).

When the photo sensor is illuminated (with sufficiently strong light), the red LED light turns on, and when it is not red LED light turns off.




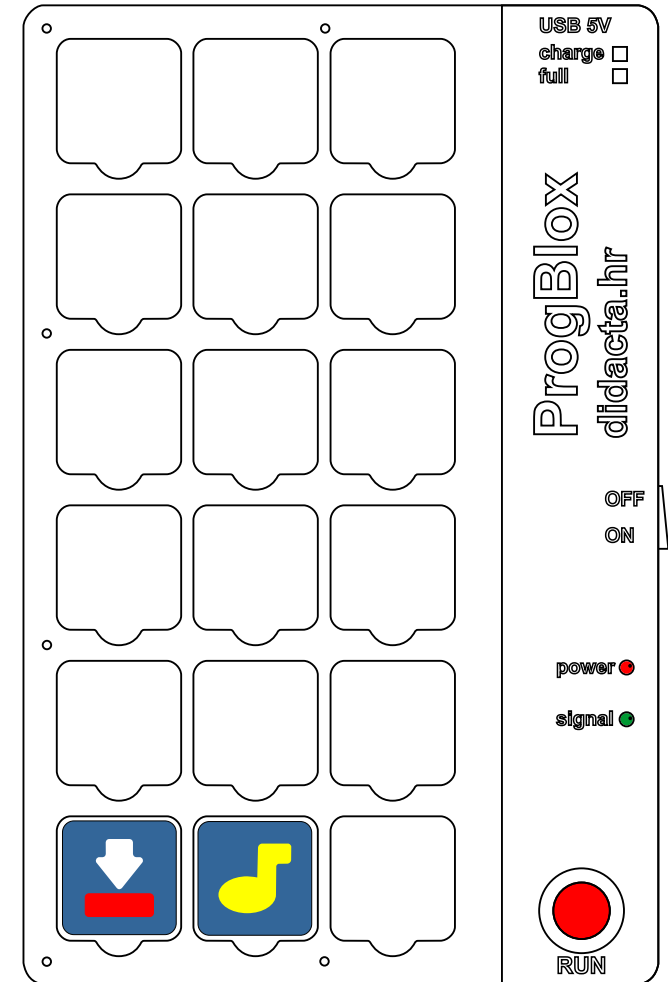
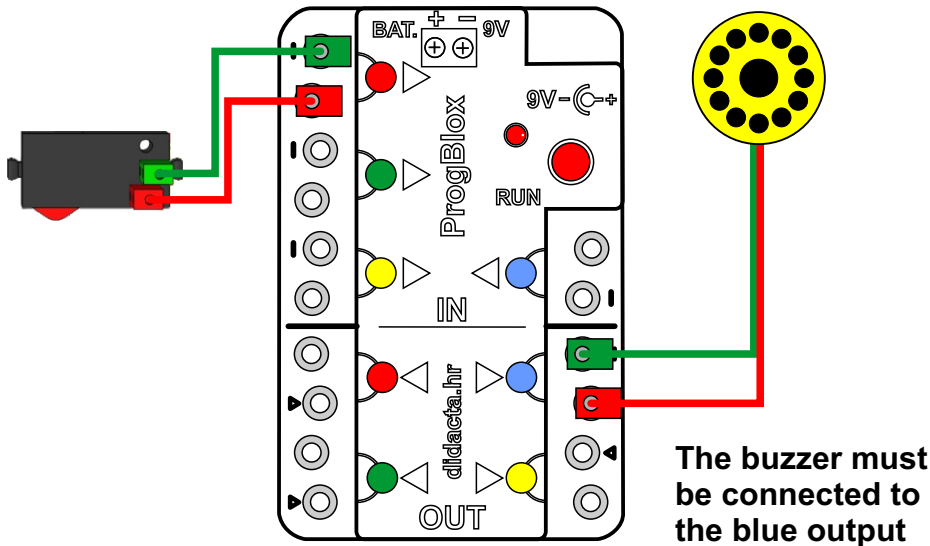
Example 8

One subroutine.

Buzzer control via buttons.

For the buzzer, there are two command blocks that can be used to perform two different melodies.

Try another melody  .

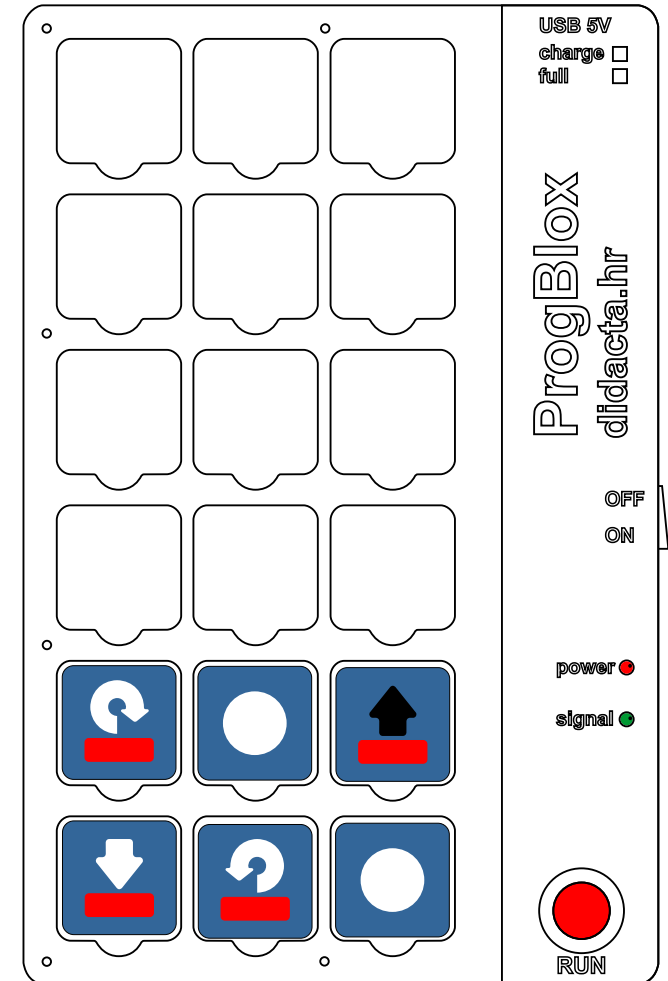
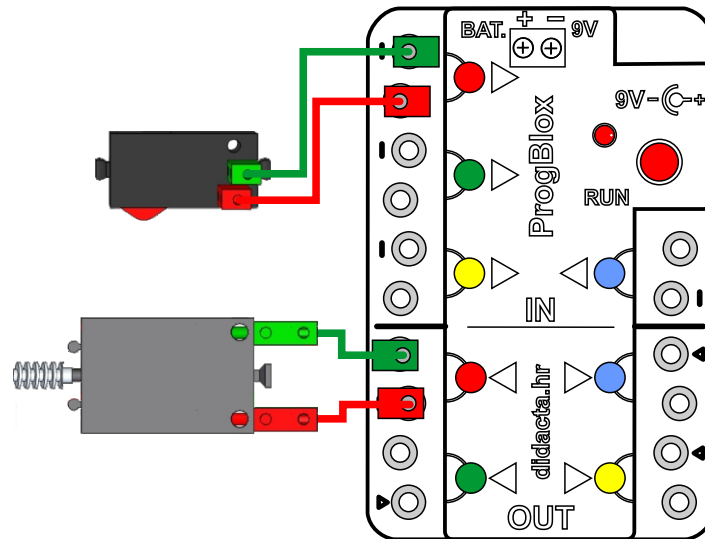


Example 9

One subroutine.

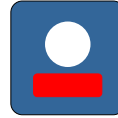
DC motor control via switch.

Pressing the switch will cause the motor to rotate in one direction for one second and then rotate in the other direction for one second.



Example 10 - HAIR DRYER - PHOTO sensor

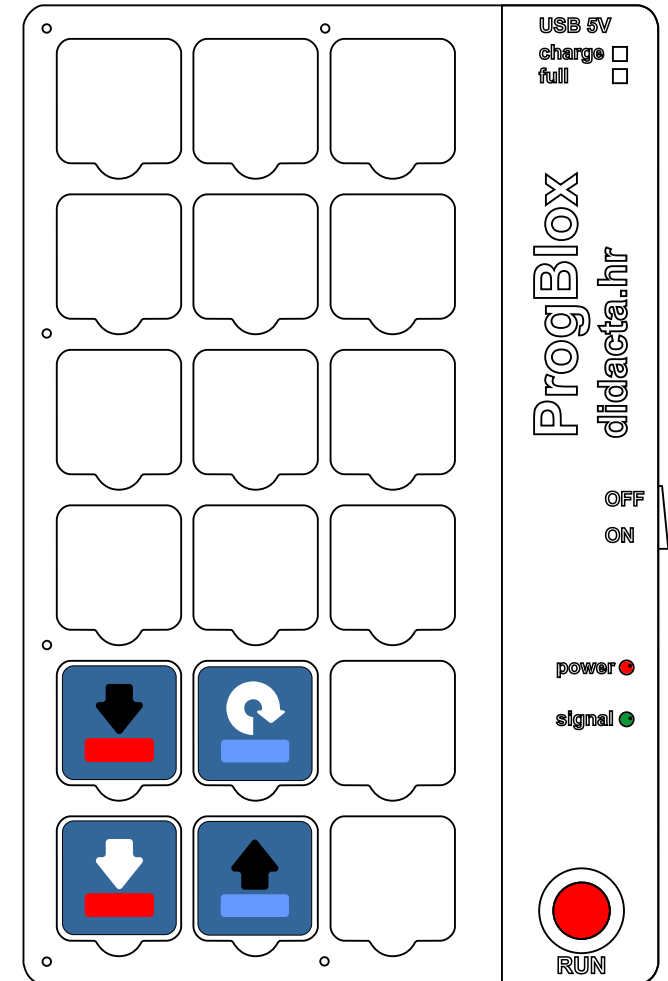
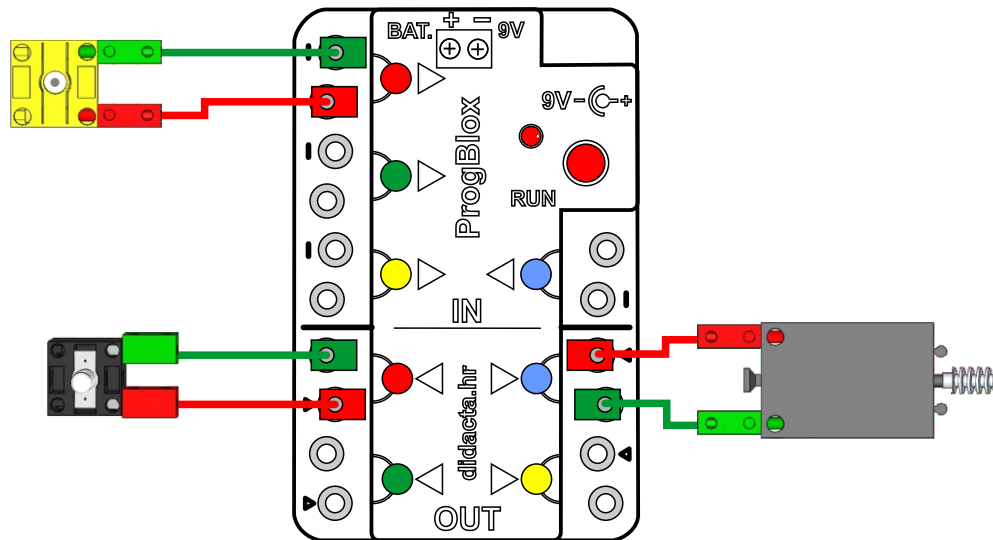
Two subprograms and a main program.



First, start **main program** to turn on the LED light:

At the beginning, the main program turns on the LED light that illuminates the photo sensor.

If the photo sensor has a signal, the motor stops working (the output signal on the blue output is interrupted). By interrupting the light that illuminates the photo sensor, the fan motor is started, as long as the photo sensor has no signal.



Example 11 - connection diagram for a vehicle

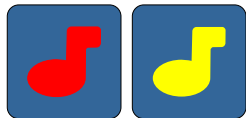
Scheme of connecting the device to the module, for proper operation of the program (if everything is well connected) and movement of the vehicle.

You can check the correct rotation of the motor with the FORWARD command cube.

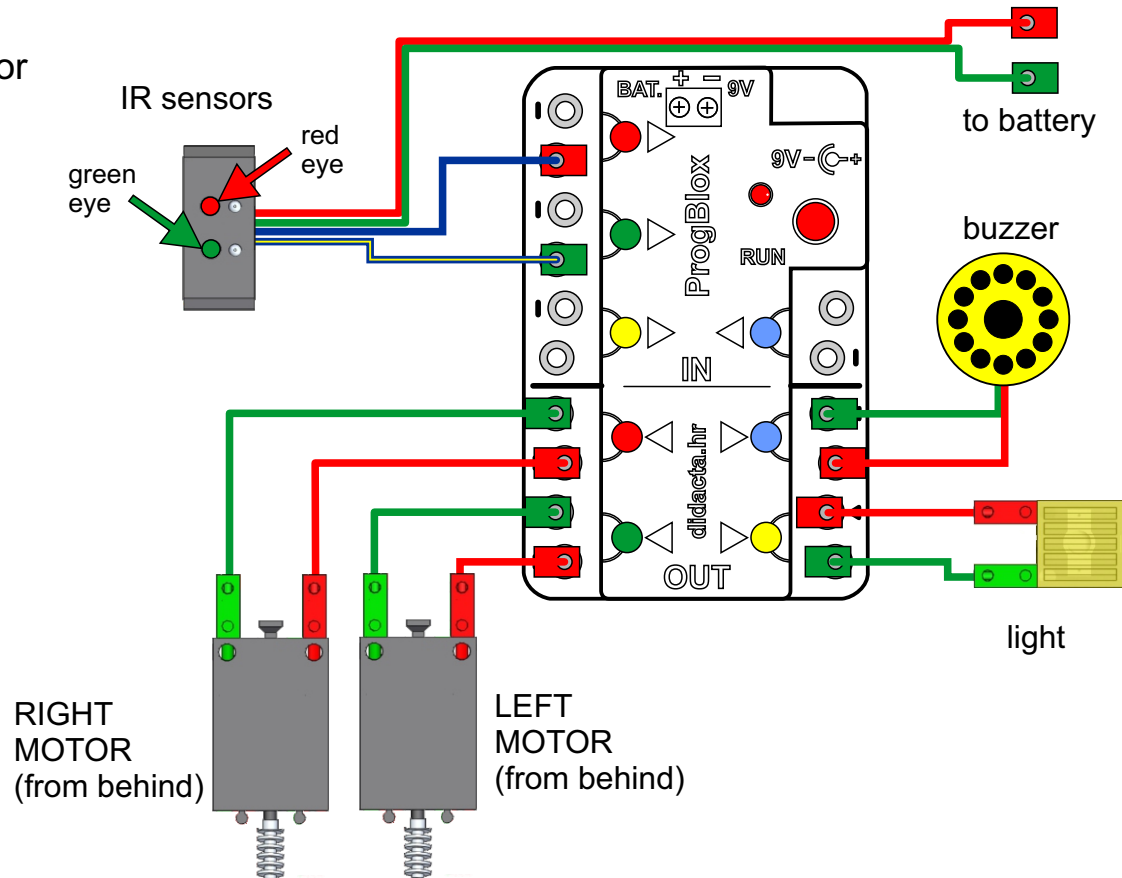
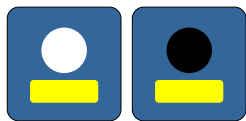


The vehicle should be moving forward.

You can control the buzzer via command cubes for melody 1 and 2:



The correctness of connecting the LED light can try via command cubes:




Example 12 - IR sensors

NOTE: The IR sensor works in reverse to the IR sensor on the vehicle


Test the subroutine by moving the IR sensor over the black line.

Try it with one sensor, then another.

subroutine 1




when the green eye is above black backgrounds (line)




turn on the yellow LED light

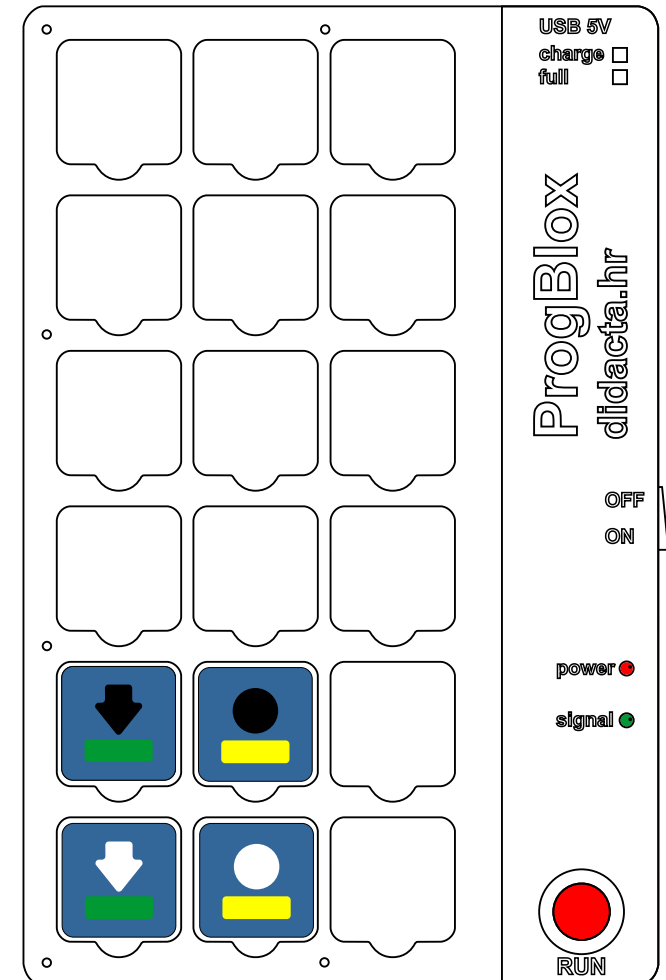
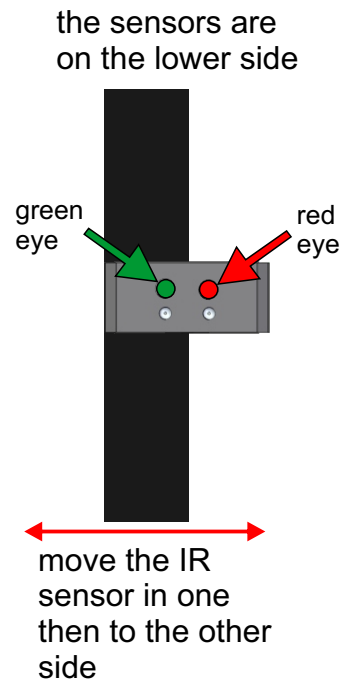
subroutine 2



when the green eye is above white background



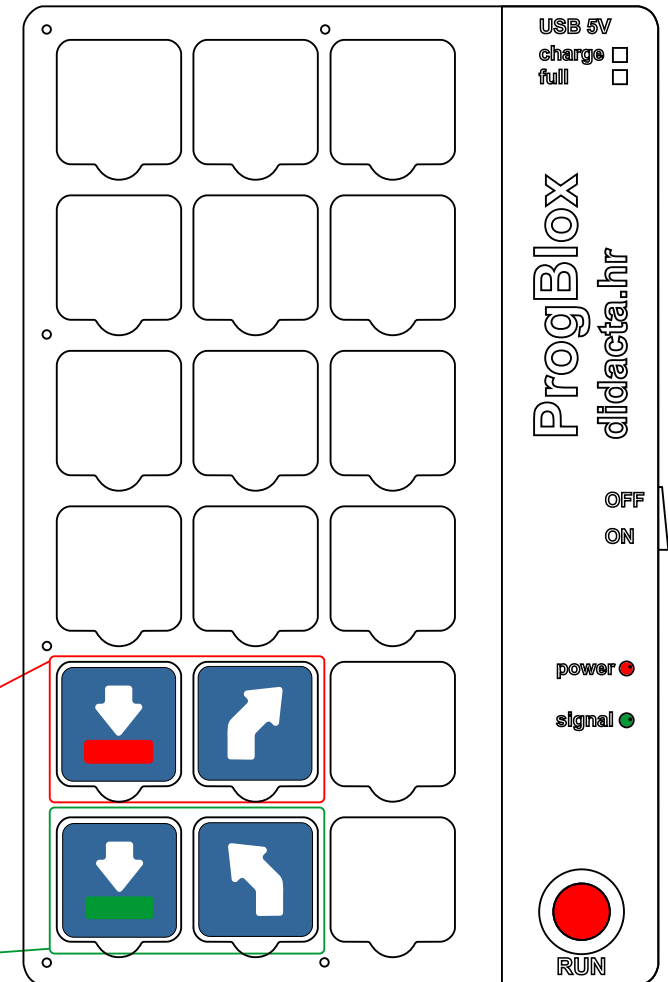
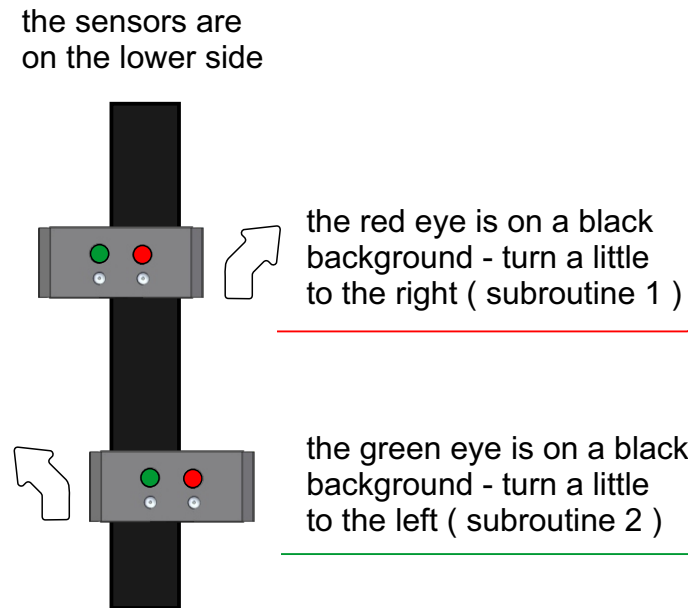
turn off the yellow LED light



Example 13 - THE CAR FOLLOWS THE LINE

If the IR sensors (EYES) are working properly, you can make a program to drive along the line.

Width of the black line: 1.5 - 2 cm.

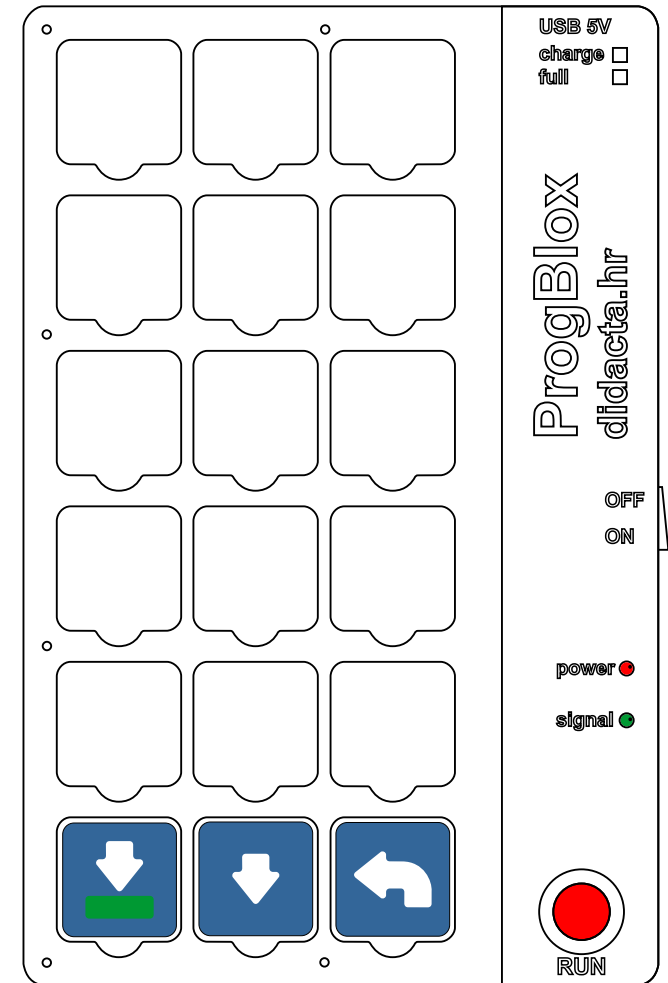
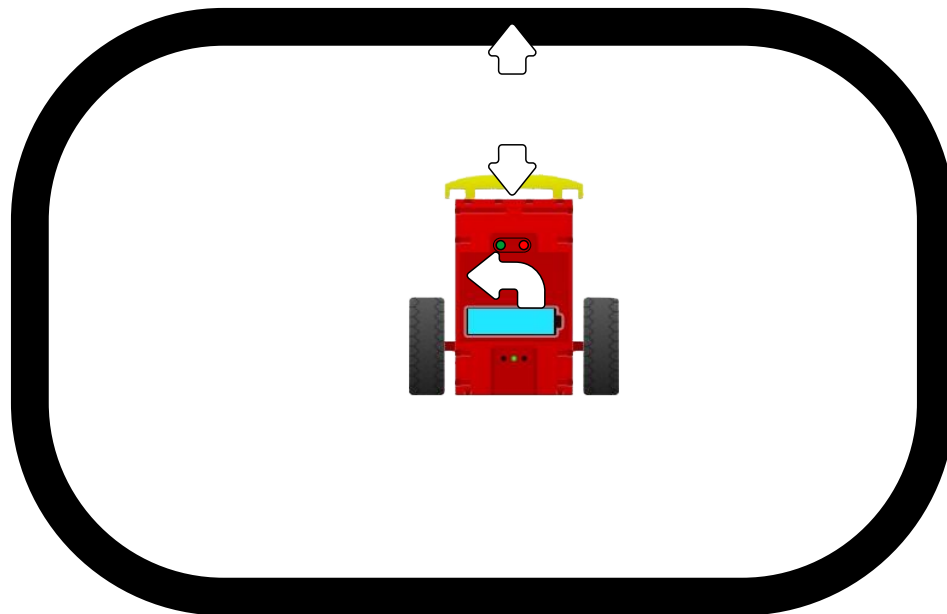


Example 14 - THE VEHICLE IS MOVING WITHIN THE PARCOUR

For the display, we use the vehicle model from the Car Set.

- in the example you can use green or red eye (sensor)

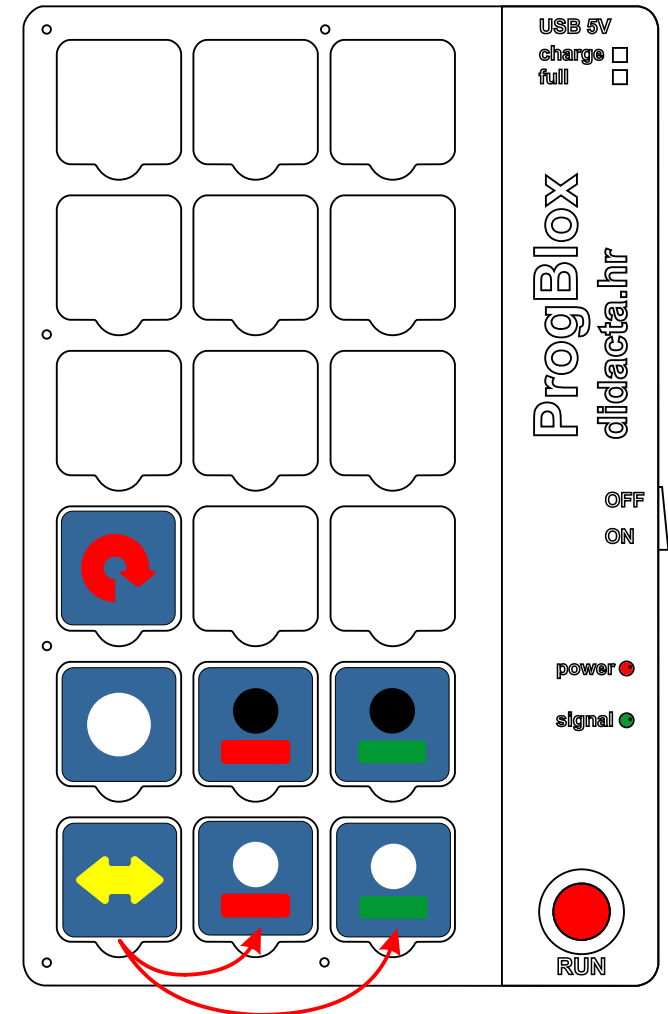
Main program:  



Example 15 - LED LIGHTS - ADVANCED

The red and green LED lights turn on alternately. The decision is made via the decision command cube. The "decision" command cube transfers control of program execution to the control module. The control module executes the first (red light) or second (green light) program step behind the "decision" cube through a random decision process.

THE CONTROL MODULE DECIDES WHICH LED LIGHT WILL LIGHT UP , RED OR GREEN

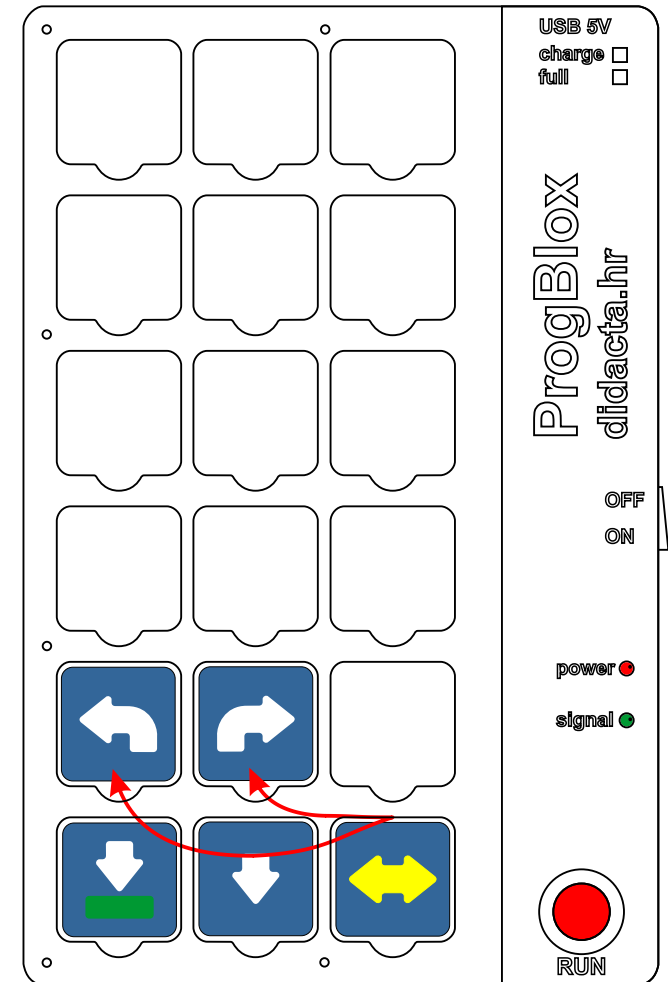
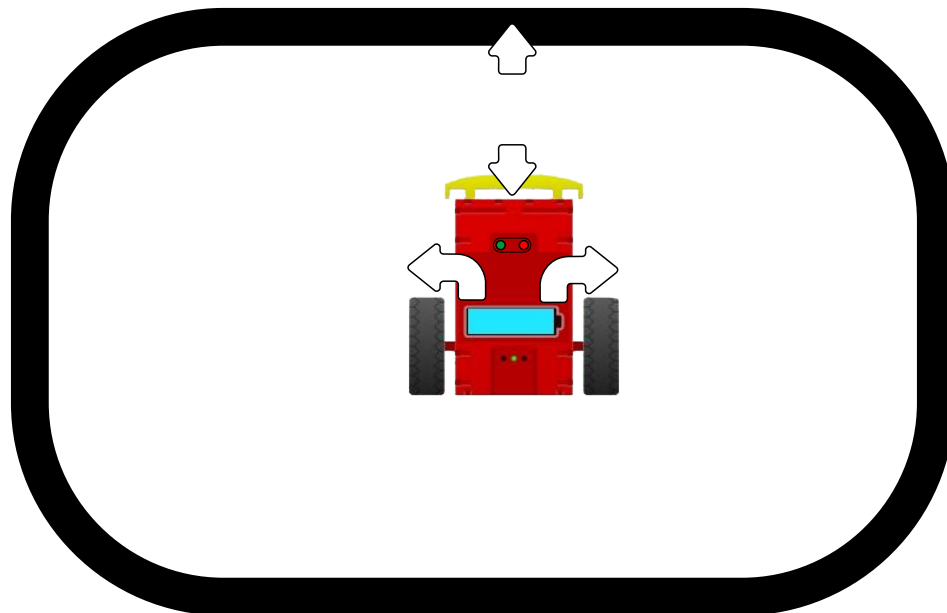
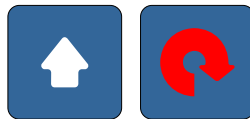


Example 16 - THE VEHICLE IS MOVING WITHIN THE PARCOUR - ADVANCED

For the display, we use the vehicle model from the Car Set.

THE CONTROL MODULE DECIDES ON THE VEHICLE TURN DIRECTION

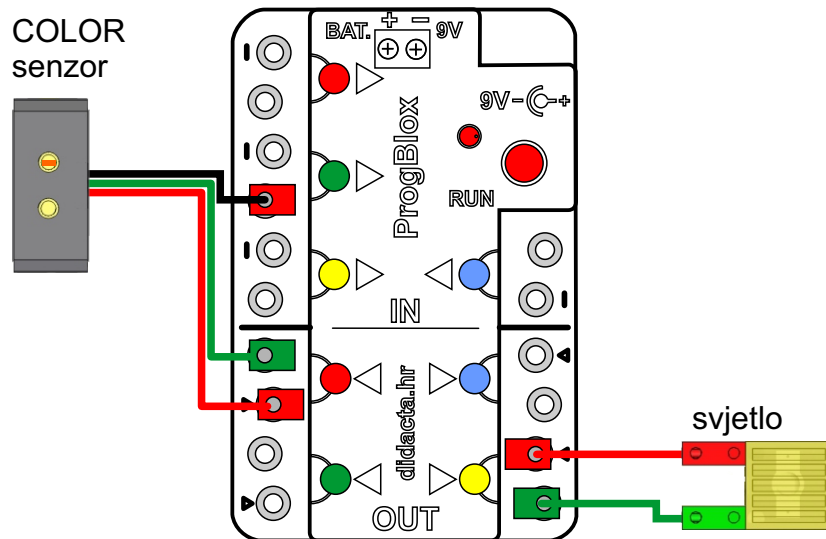
Main program:



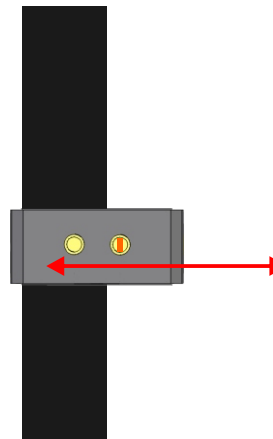
Example 17 - COLOR sensor

NOTE: the color sensor recognizes black and white surfaces

You can connect the sensor's power supply to the battery or to one of the outputs. If you connect to the red output (example), don't forget to use the command cube for the lit red LED light in the main program.

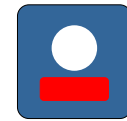


the sensor are on the lower side



move the COLOR sensor in one, then to the other side

MAIN PROGRAM:

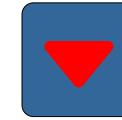


			USB 5V charge full
			full
			ProgBlox didacta.hr
			OFF ON
			power signal
			RUN

Saving programs

Save the program and restart

To save the program in the permanent memory of the control module, the command cube is used at the end of the program (subprogram) or independently.



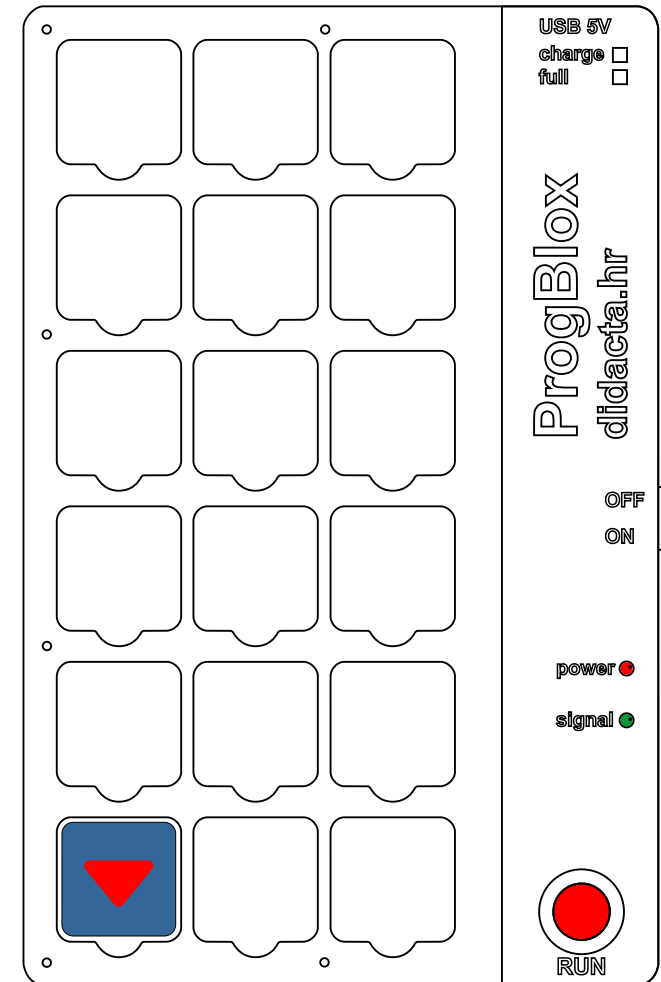
Try the programs first, then save them.

To start saved programs from the permanent memory of the control module, it is necessary:

- turn on the power supply of the control module
- press the RUN button briefly

The saved programs from permanent memory are started only with the first press of the RUN button.

Each subsequent press of the RUN button (of the module) deletes all programs from the working memory of the control module (programs saved in permanent memory are not deleted).

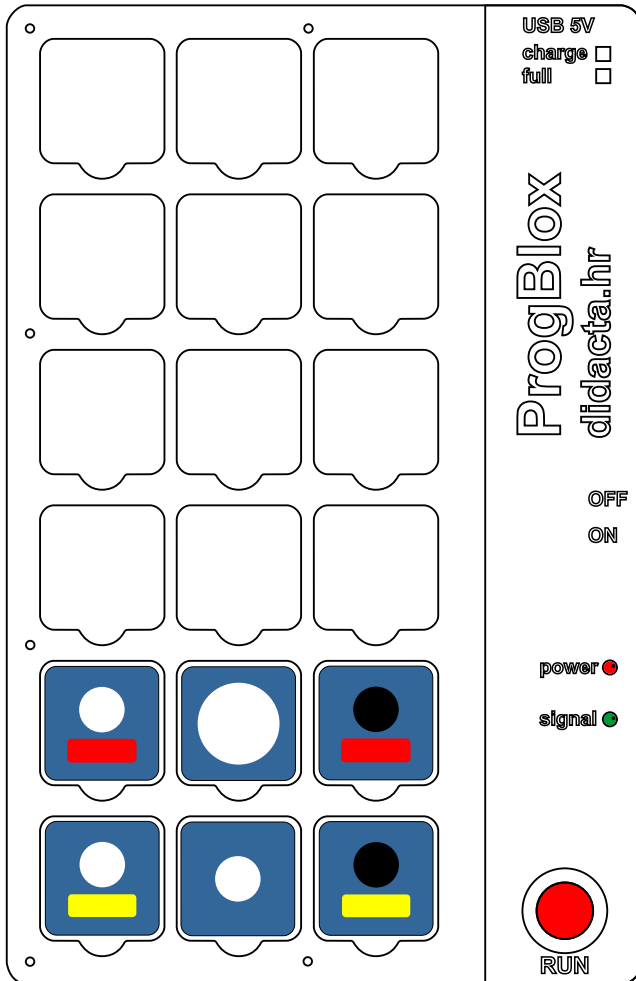


Example 18 - MAIN PROGRAM ON TWO SIDES OF THE TABLET - TRAFFIC LIGHT

**FIRST PART OF THE PROGRAM
- PAGE 1**

1. Assemble the first part of the program and save it in the memory of the CONTROL module by pressing the **RUN** key.

Subsequent changes to the first part of the program do not change the second part and vice versa.



**THE SECOND PART OF THE PROGRAM
- PAGE 2**

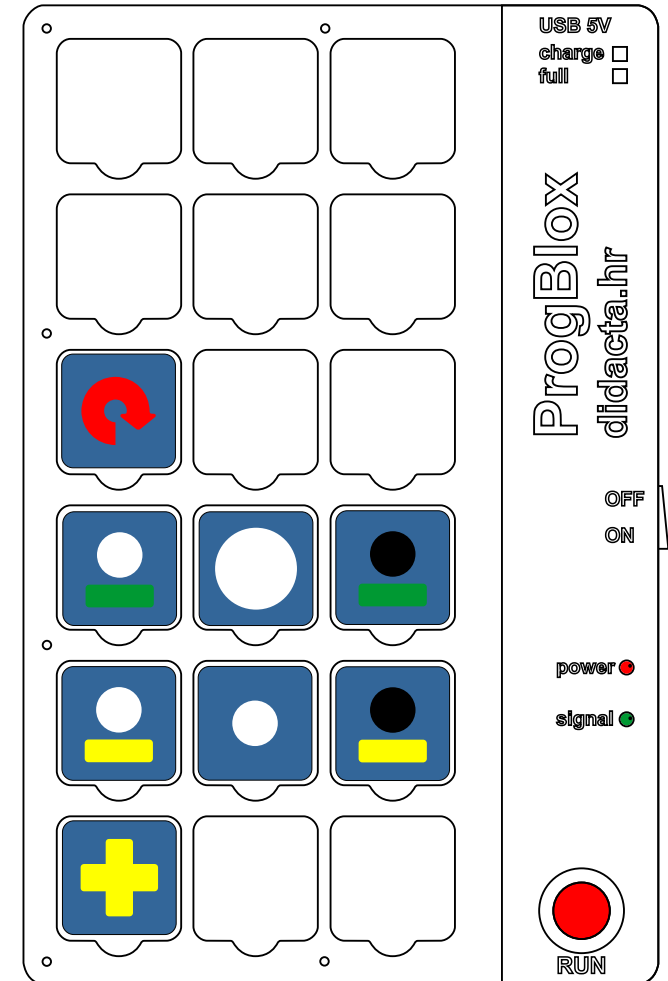
2. Assemble the second part of the program and save it in the memory of the module by pressing the **RUN** key.

3. If you want to **SAVE** the entire program (first and second part) in the permanent memory of the module, add a command block



for saving at the end of the second part of the program.

The complete program will be saved in the permanent memory of the control module.



GROUP 1 - for vehicle movement control and other command cubes



wait a little longer (2 sec.)



wait a long time (15 sec.)



rotate the vehicle to the left side halfway



rotate the vehicle to the right a little



wait little (1 sec.)



wait a lot longer (10 sec.)



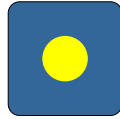
rotate the vehicle to the right side halfway



rotate the vehicle to the left a little



wait very little (0.5 sec.)



wait longer (5 sec.)



drive in the forward direction for a long time



drive in the forward direction for a short time



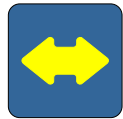
delete all programs from the working memory of the c.module



drive in the direction back long



drive in the reverse direction for a short time



decision - command cube first or second in sequence



rotate the vehicle to the right completely



rotate the vehicle to the left completely

FOR MAIN PROGRAM ONLY



















the main program, the second part



repeat endlessly the main program

GROUP 2 - for lights and inputs

-  turns on the LED light connected to the red output
-  turns on the LED light connected to the green output
-  turns on the LED light connected to the yellow output
-  turns on the LED light connected to the blue output
-  turns off the LED light connected to the red output
-  turns off the LED light connected to the green output
-  turns off the LED light connected to the yellow output
-  turns off the LED light connected to the blue output

-  red INPUT HAS signal
-  green INPUT HAS signal
-  yellow INPUT HAS signal
-  blue INPUT HAS signal
-  red INPUT NO signal
-  green INPUT NO signal
-  yellow INPUT NO signal
-  blue INPUT NO signal

command cubes for INPUTS CANNOT be used in the main program

GROUP 3 - for DC motors and outputs



run the DC motor connected to the red output to the LEFT



run the DC motor connected to the green output to the LEFT



run the DC motor connected to the yellow output to the LEFT



run the DC motor connected to the blue output to the LEFT



run the DC motor connected to the red output to the RIGHT



run the DC motor connected to the green output to the RIGHT



run the DC motor connected to the yellow output to the RIGHT



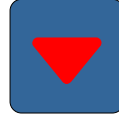
run the DC motor connected to the blue output to the RIGHT



melody 1



melody 2



SAVE programs



red OUTPUT stop - same function



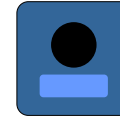
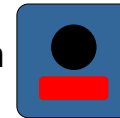
green OUTPUT stop - same function



yellow OUTPUT stop - same function



blue OUTPUT stop - same function



ProgBlox Plus



ProgBlox

Plus

programming manual



TRAVNIČKA 18, 40000 ČAKOVEC, CROATIA (EU)
