

# Let's code!

First, open CircuitBlocks and connect your Wheelson to your computer's USB port.





CircuitBlocks should now say "Wheelson connected".

If CircuitBlocks didn't recognize your Wheelson, please check if the USB cable is plugged in properly and if you are using a working USB port on your computer.

If you still cannot get CircuitBlocks to recognize your Wheelson, reach out to us via email at [contact@circuitmess.com](mailto:contact@circuitmess.com).

Let's make your Wheelson's lights blink

In computer programming, a variable is a storage location that contains a value.

Every variable has a specific name and you can store and change the value of a variable.

Let's create our first variable. Find the section named "Variables" and press the "Create variable..." button.



A new window will appear, asking you to name your variable. Let's name it "blinkTime" and click "OK".

code.circuitmess.com says

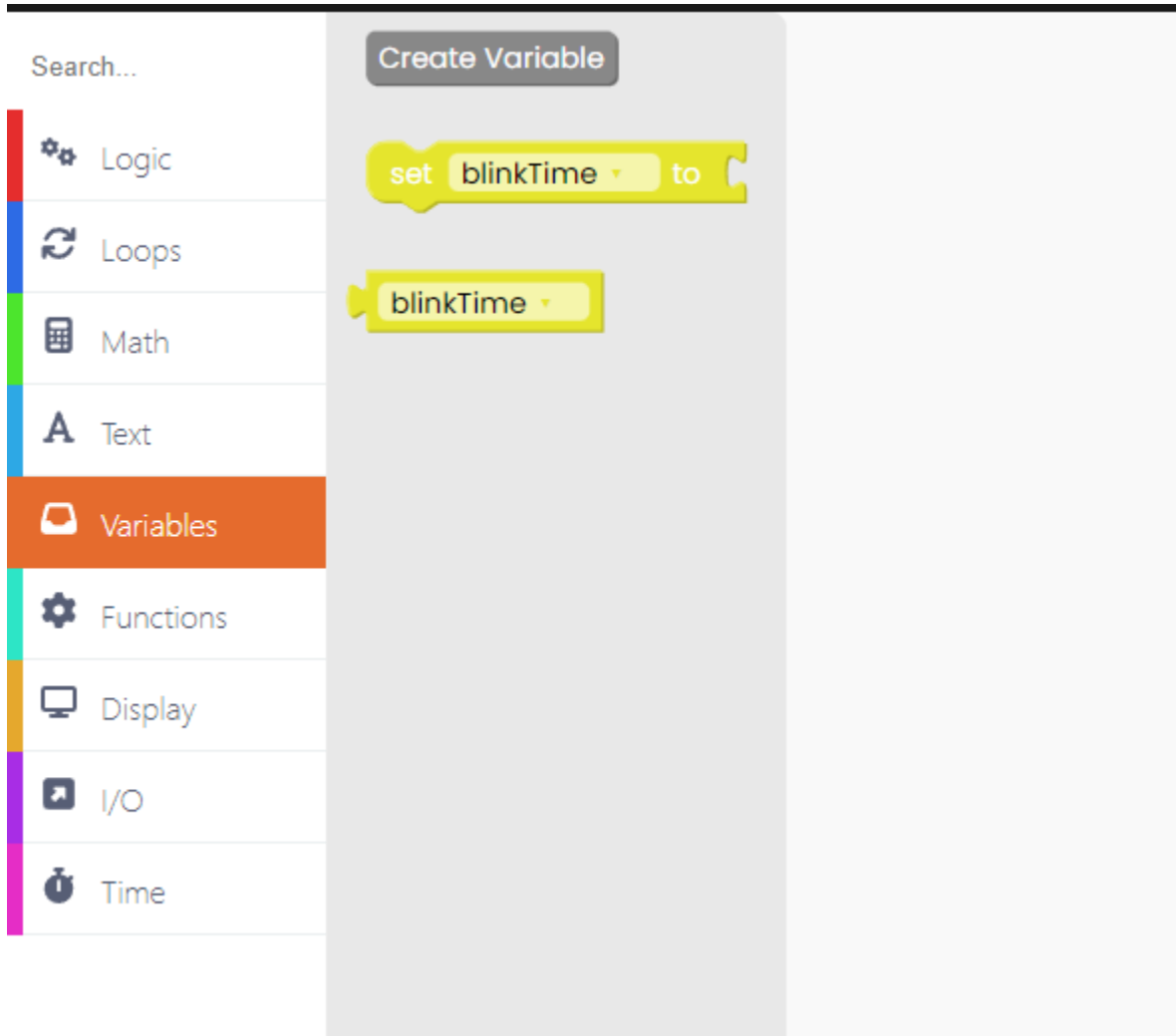
New variable name:

OK

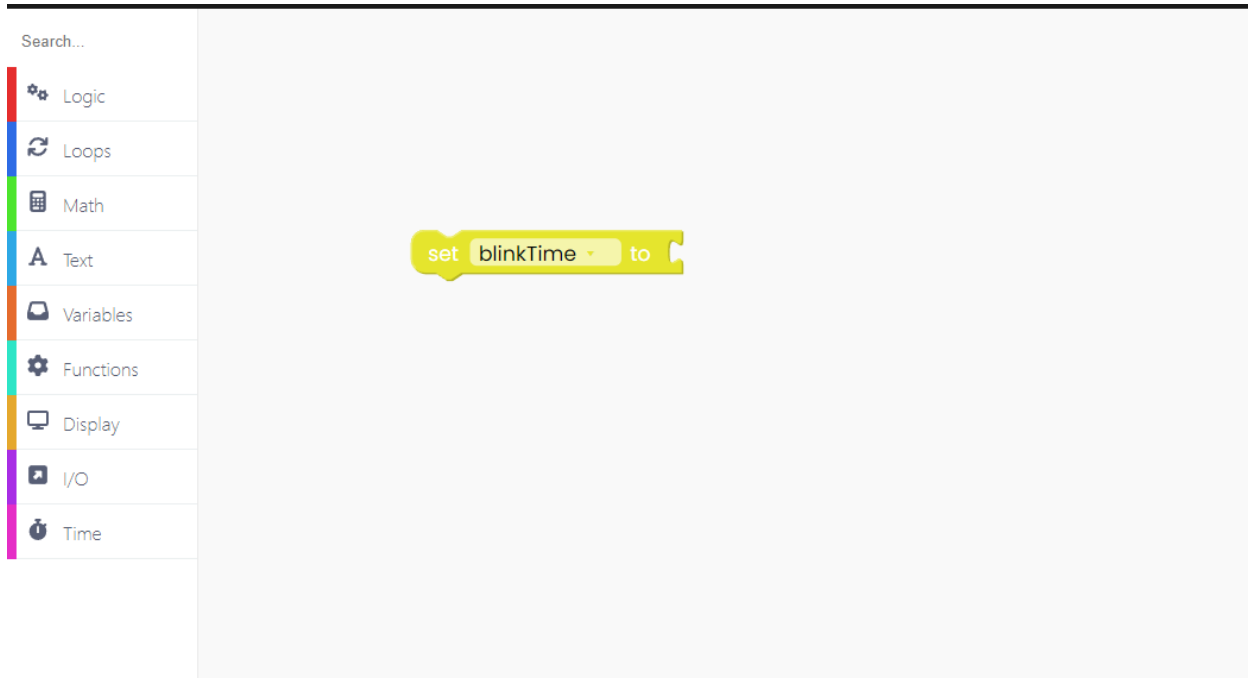
Cancel

After you save the variable, this will create new variables that you can now use.

They will look like this.



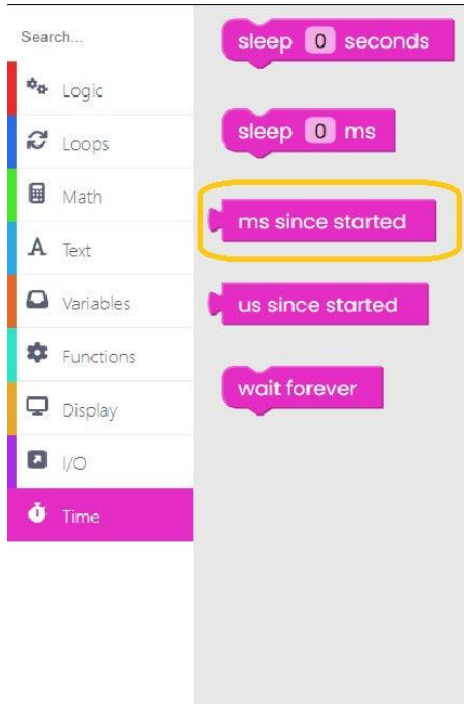
Find the variable “set ‘blinkTime’ to” and drag and drop the block into the drawing area.



When we create a variable, it's undefined - it has no value. We must set a value for every variable when our computer program starts. That's why you'll need the "set variable" block.

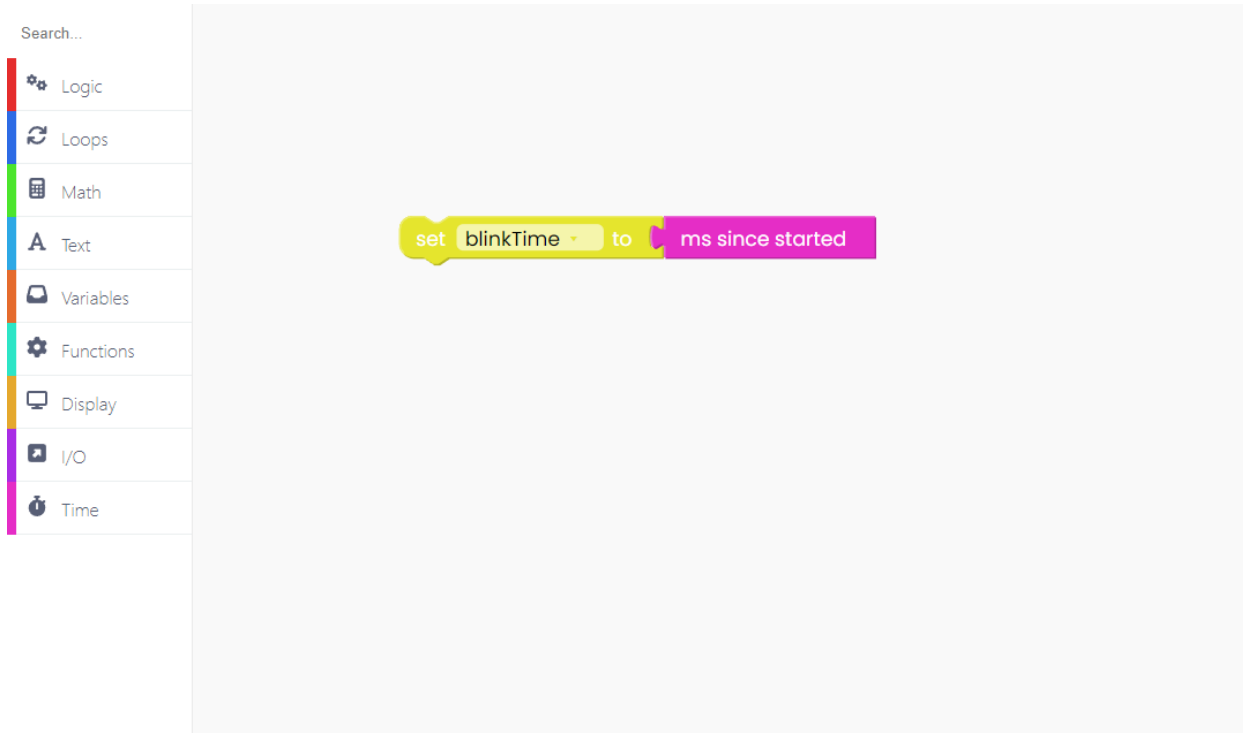
Now you need to define the value that we want to set the variable to.

Find the "Time" section on the left side of the screen and from there take the "ms since started" block.



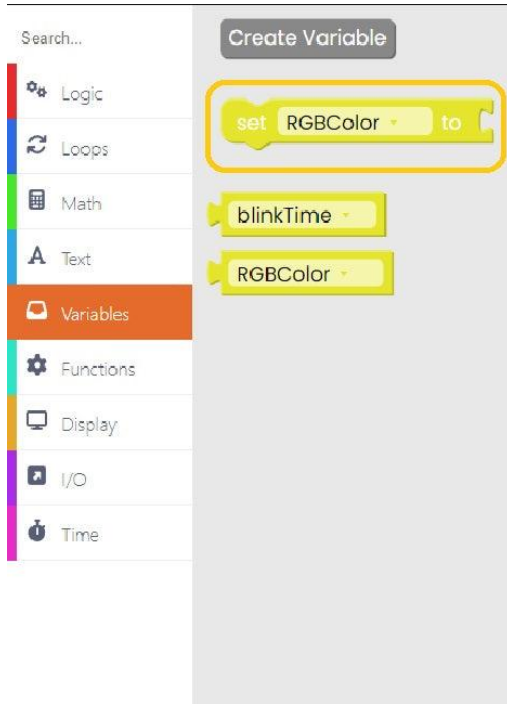
Take the “ms since started” block and drag it into the empty spot on the right side of the “set” variable.



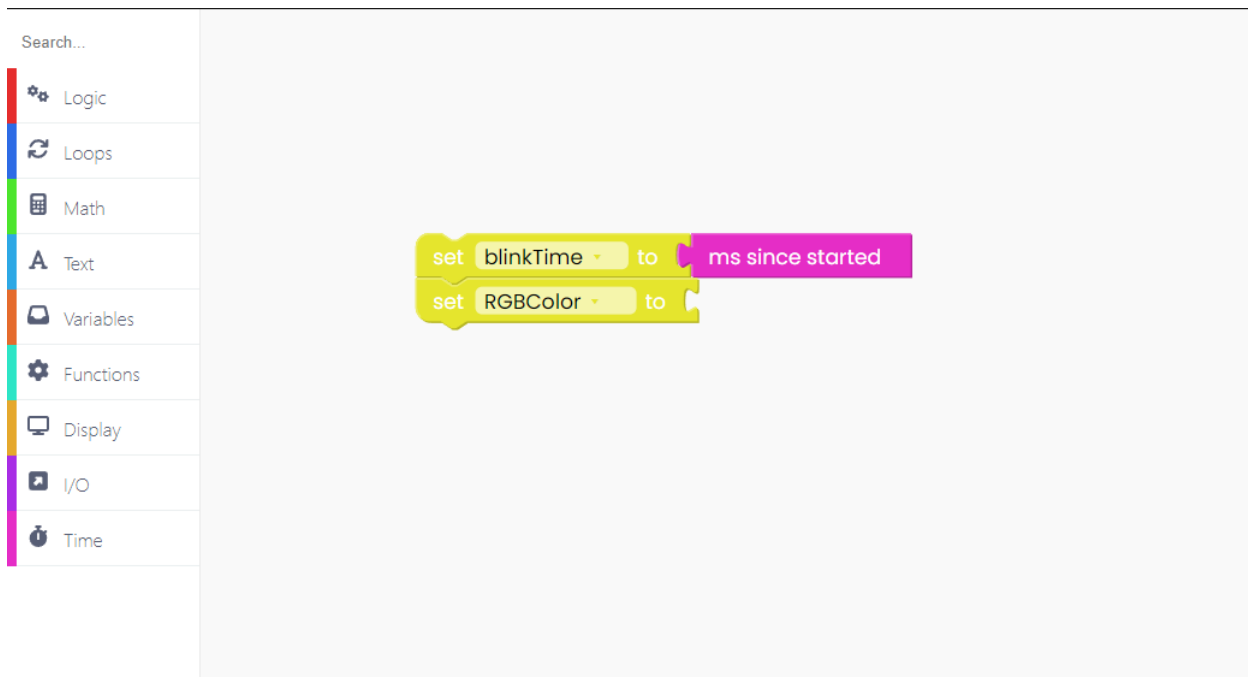


Now, we have to create a new variable that will help us set the RGB colors to the ones we want. Go back to the “Variables” section on the left side and click on “Create variable...” to create a new variable just like we did a few moments ago. Let’s name this variable “RGBcolor”. Once you enter the name, save your variable.

This will generate a few new variables that we can now play around with. For this step, we need the “set ‘RGBcolor’ to” variable.



Drag and drop the "RGBColor" variable right below the previous variable. Take a look at the photo below to see how this should look like.

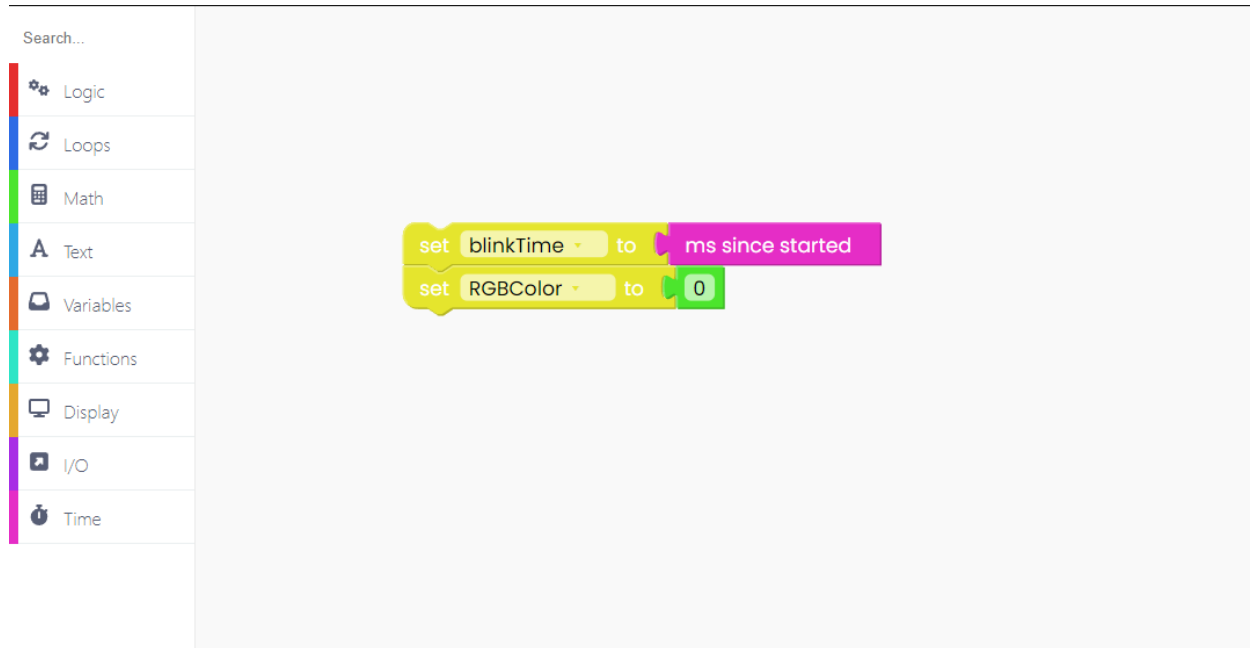




Wheelson's RGB has 7 different colors and we want to show all of them off.

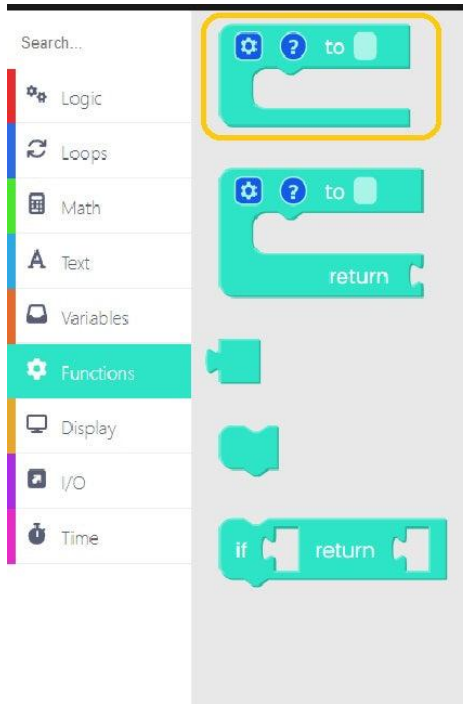
We'll start with the first color and move our way to the last one. Our first color will be set to '0,' so let's add that value to the variable.

To do that, go to the "Math" block section and find the first block with numbers. Change the number to 0.



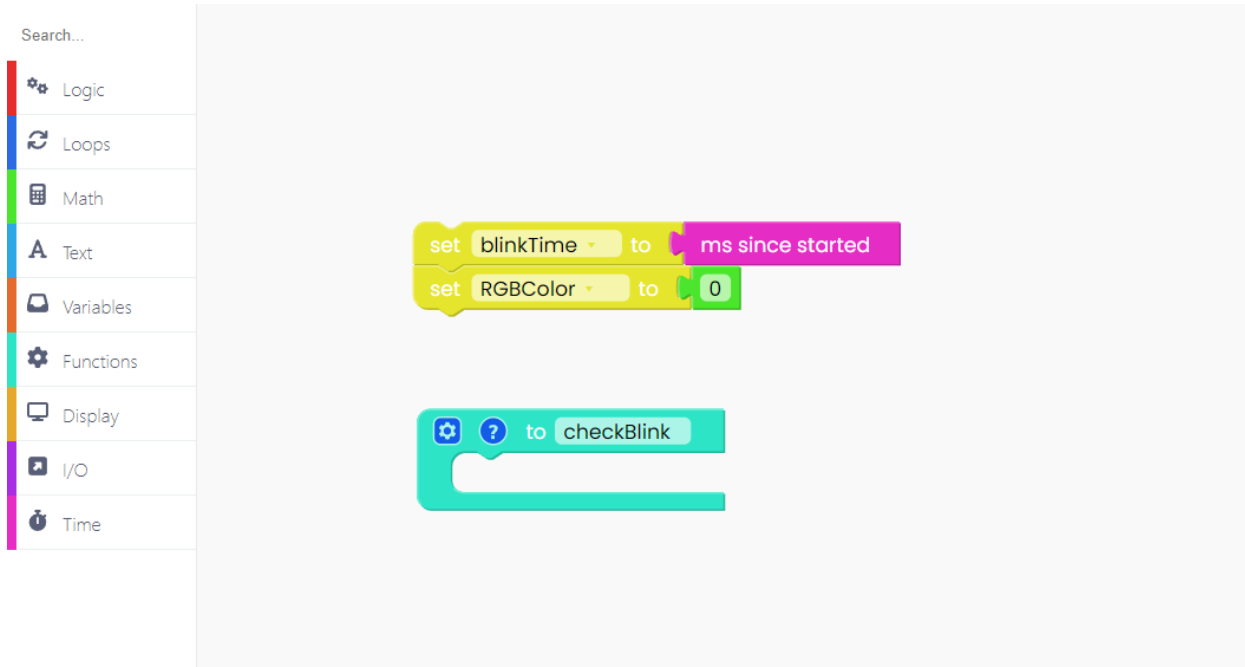
Good job so far!

Now let's create a function. Find the "Functions" section on the left side of the screen and click on this block:

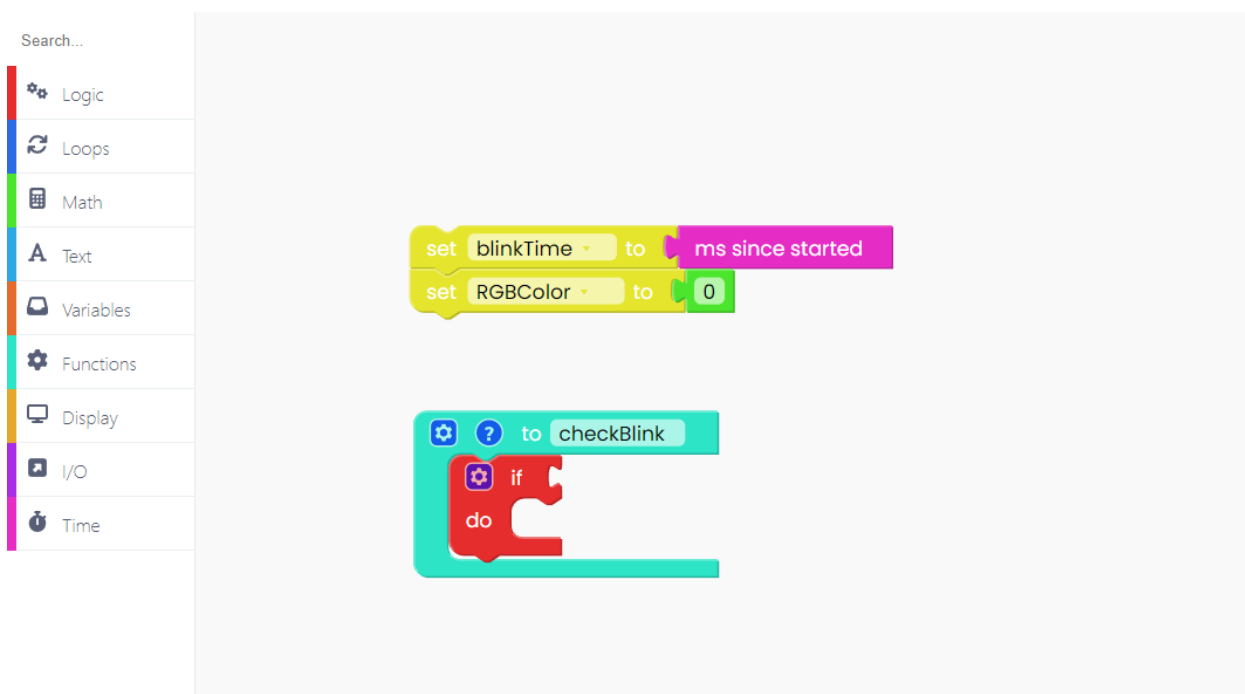


You can name the function by simply writing down the name in the empty space.

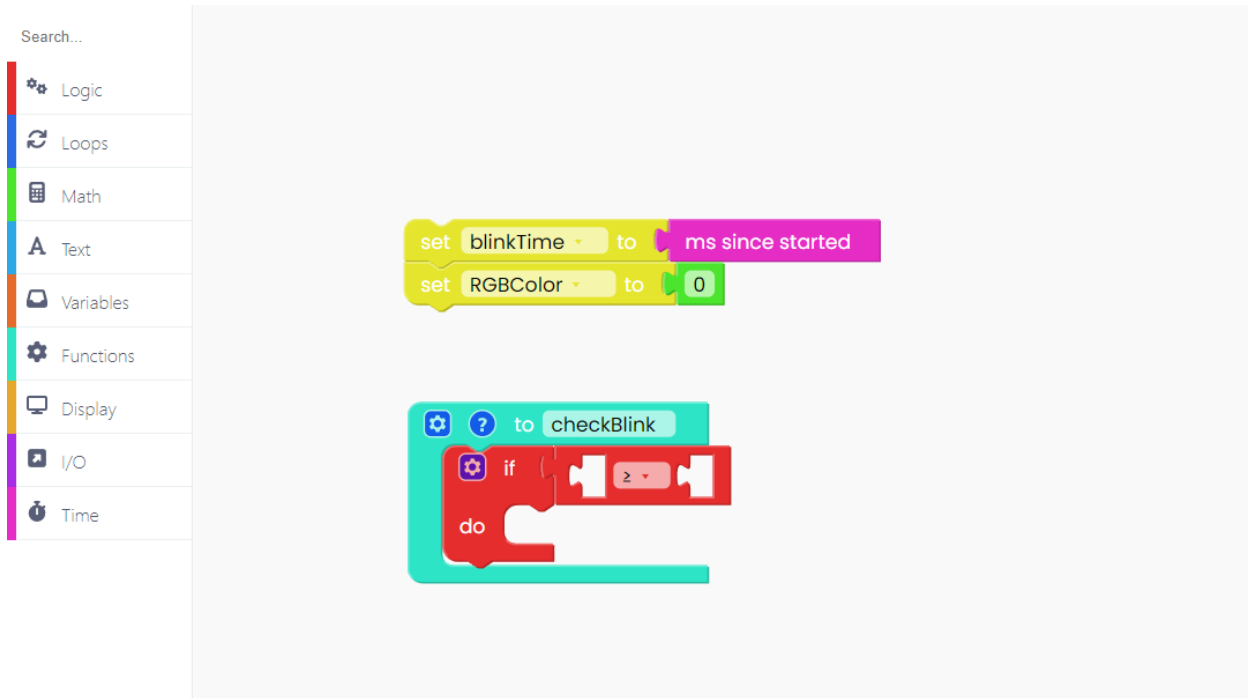
We named it "checkBlink".



Now we'll define what this function does. Let's add some logic to it. From the "Logic" section on drag and drop the "if" block inside of the "checkBlink" function.



Add this comparison block from the “Logic” section. Change the comparison symbol in the middle to the greater than or equal to symbol ( $\geq$ ) from the drop-down menu.



Then, from the “Math” section, add the block with the ‘-’ symbol between the two numbers.

Search...

- Logic
- Loops
- Math**
- Text
- Variables
- Functions
- Display
- I/O
- Time

123

$\pi$

e

round

random integer from to

random fraction

constrain low high

### Arithmetic

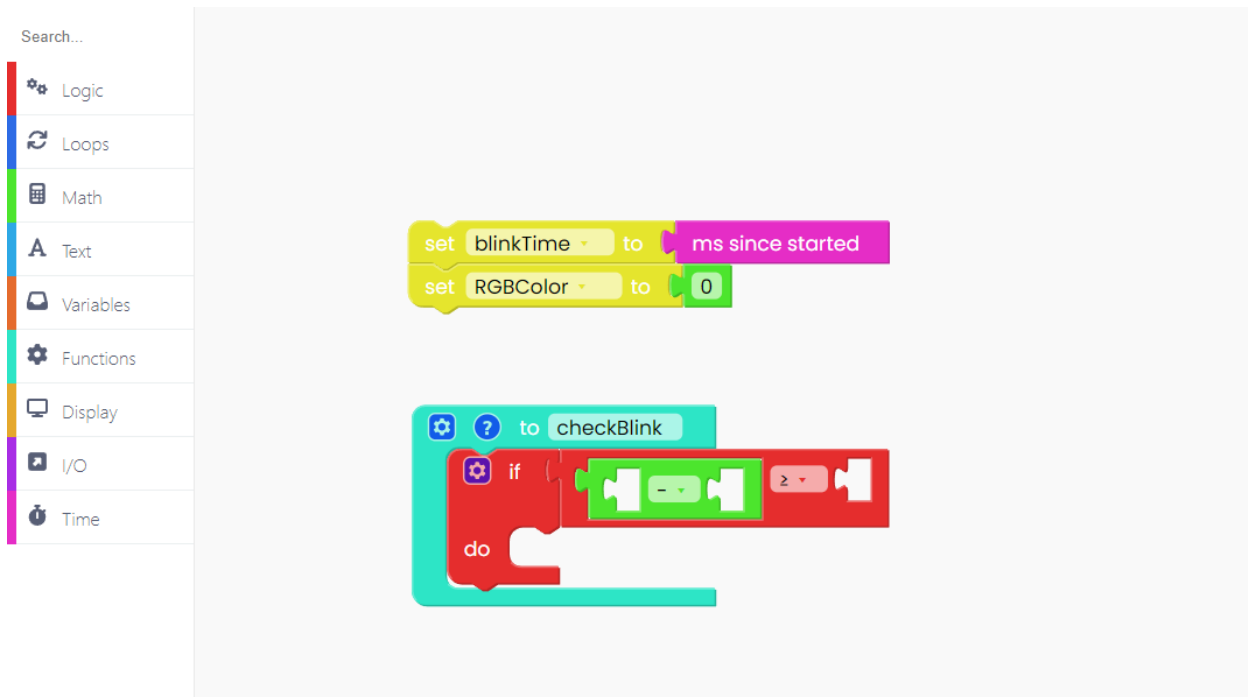
+

-

x

÷

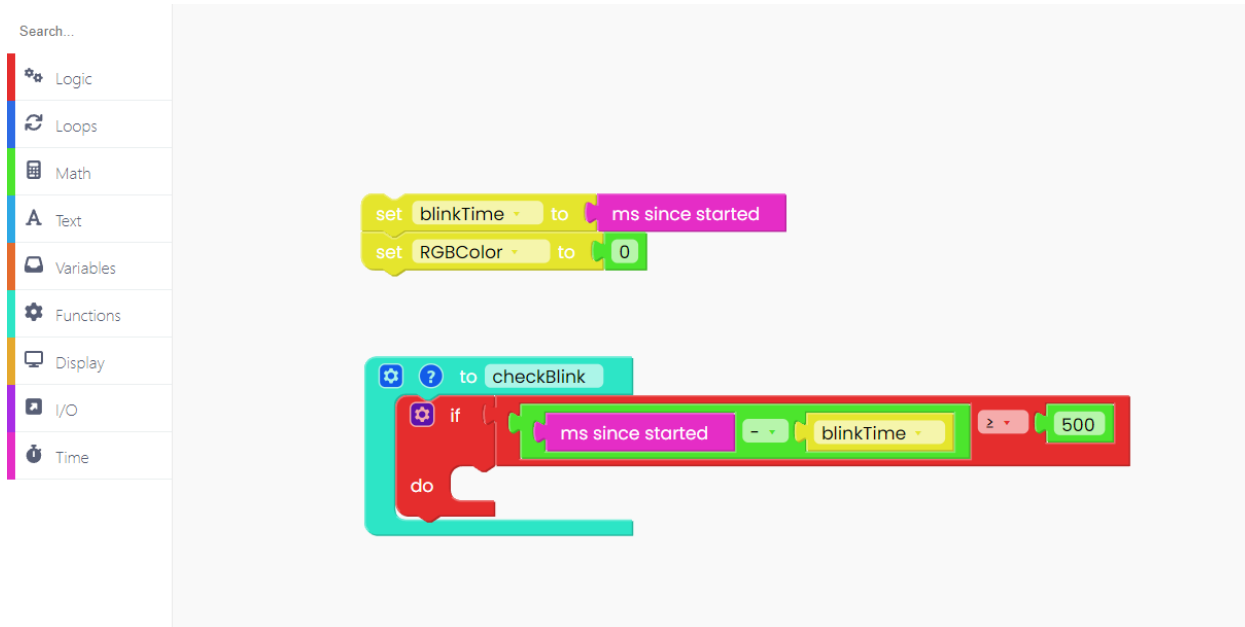
^



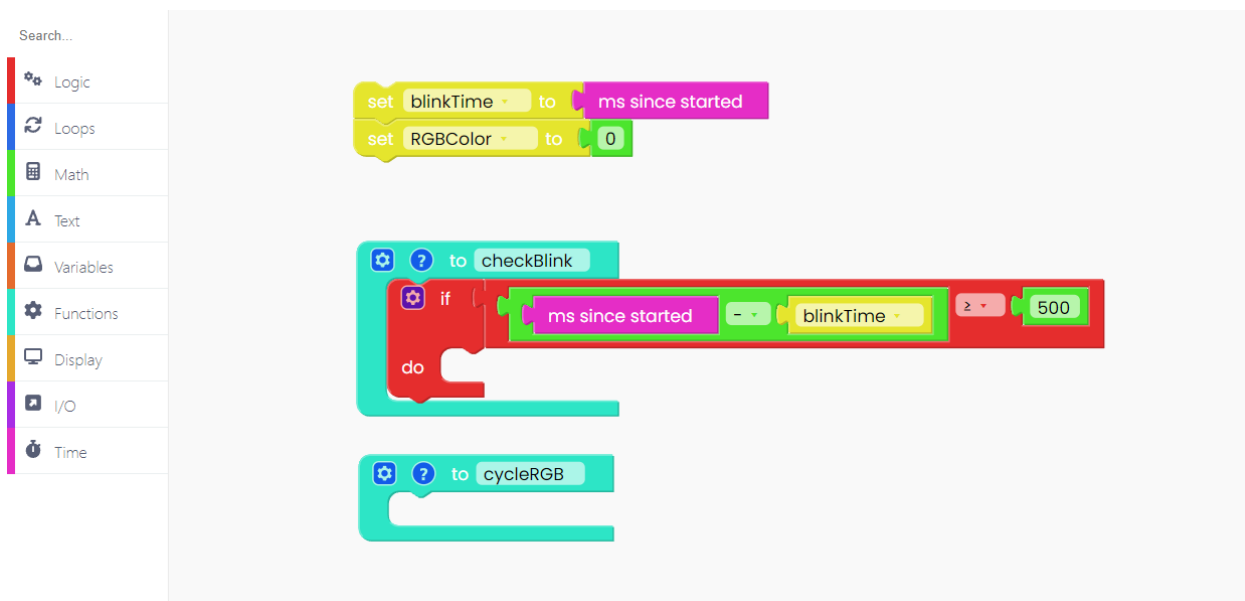
After that, we'll go to the "Time" section on the left of the screen and select the "ms since started" block and add it to the left slot of the "Math" block we just dropped.

In the right slot of this same "Math" block, we'll add the "blinkTime" variable block from the "Variables" section.

Finally, we'll change the number '123' to '500' to make the lights change color every 0.5 seconds. This is how it should look like:



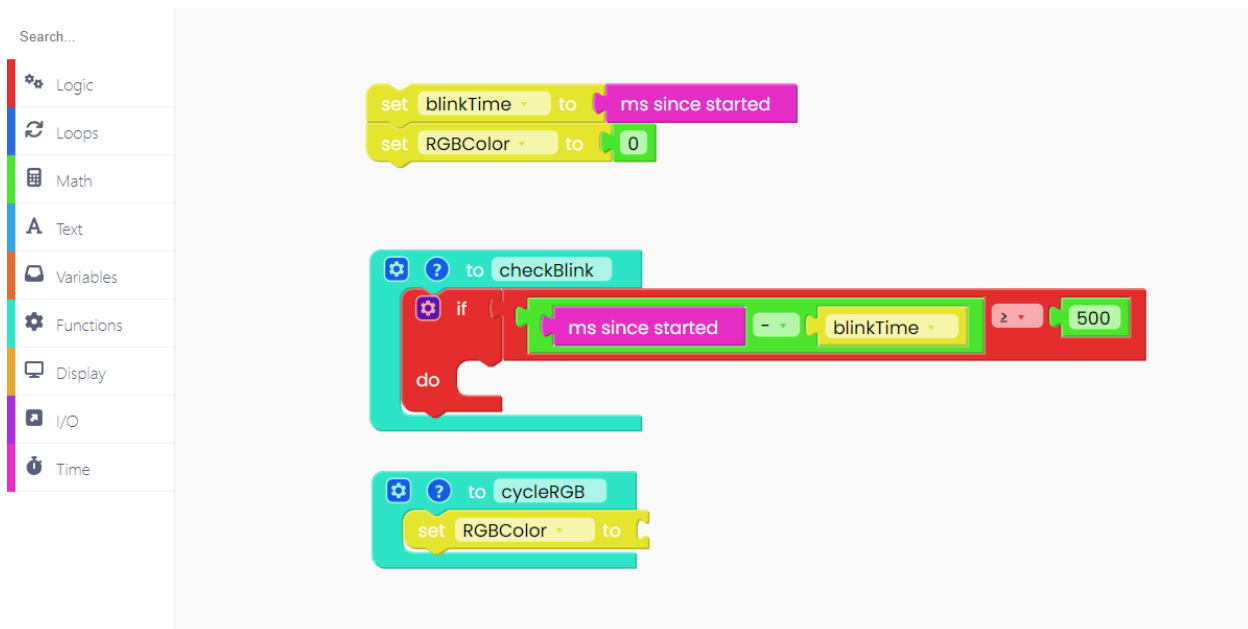
Let's create a new function called "cycle RGB". Go to the "Functions" section and click on "Create a Function". Just like the last time, choose a name for your function - let's go with "cycleRGB" this time. Click "OK," and the function will show up on the screen.





Let's add some variables to the function.

From the "Variables" section, get the "set 'RGBcolor' to" block and drop it in the empty slot of the "cycleRGB" function.

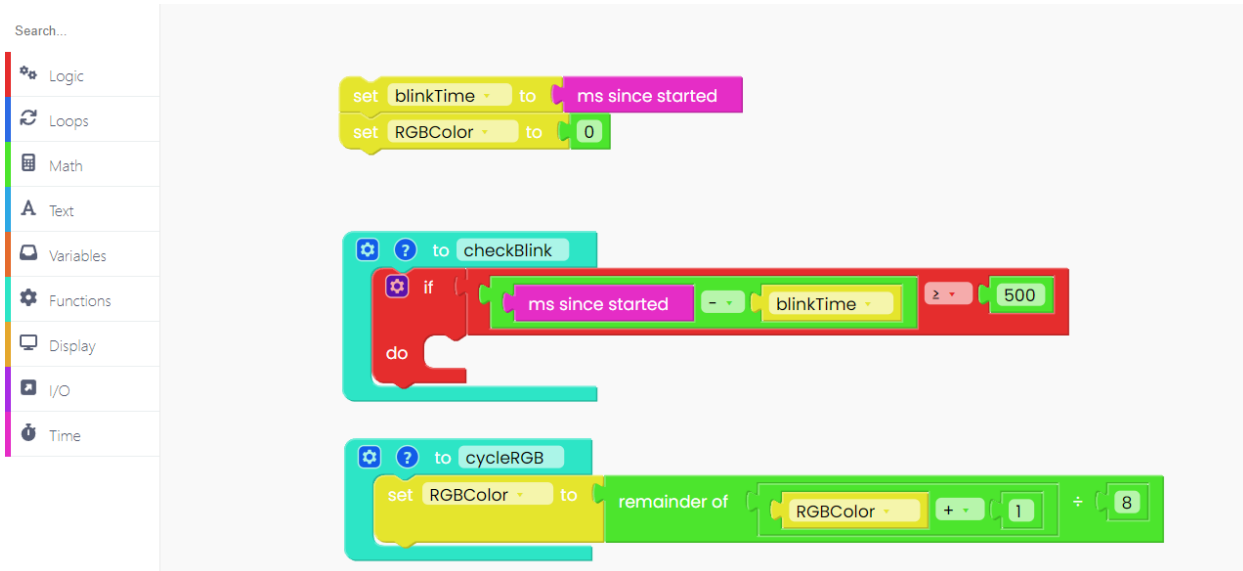


We'll fill the empty space on the right side of the "set" function with a block from the "Math" section.

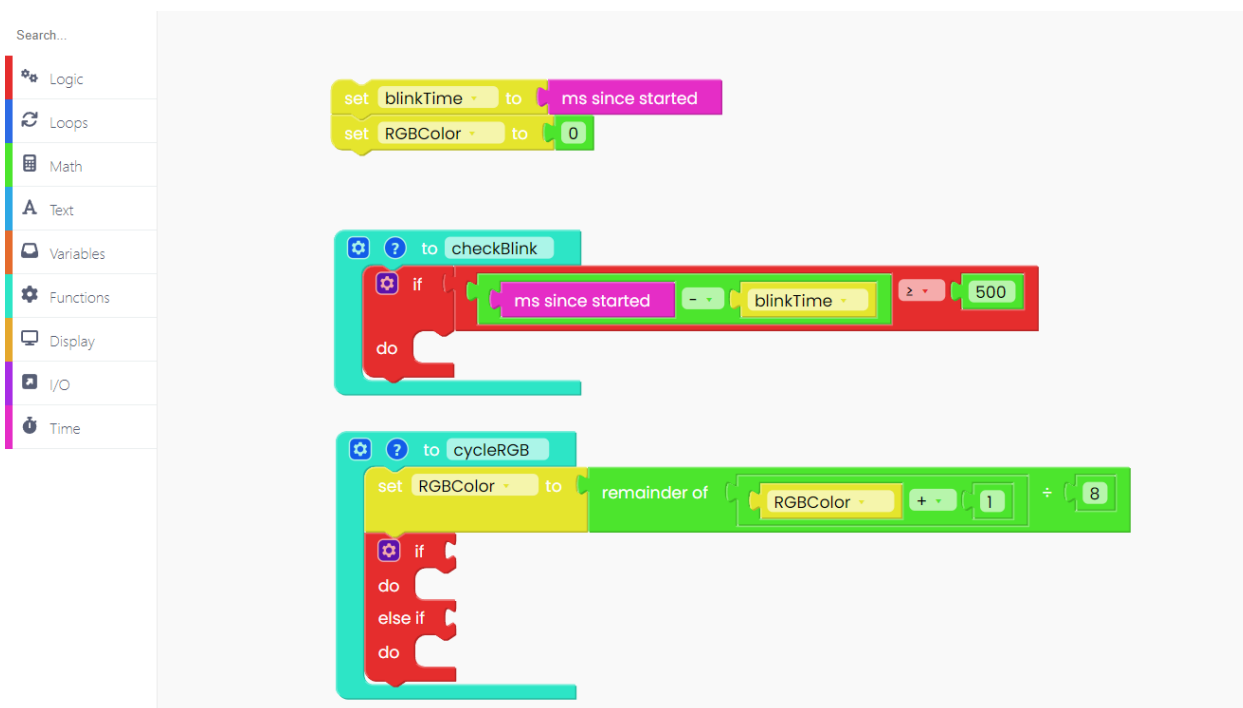
Find the "remainder of" block and drop it into the empty slot.

By default, there are some random numbers written in the "remainder of" block.

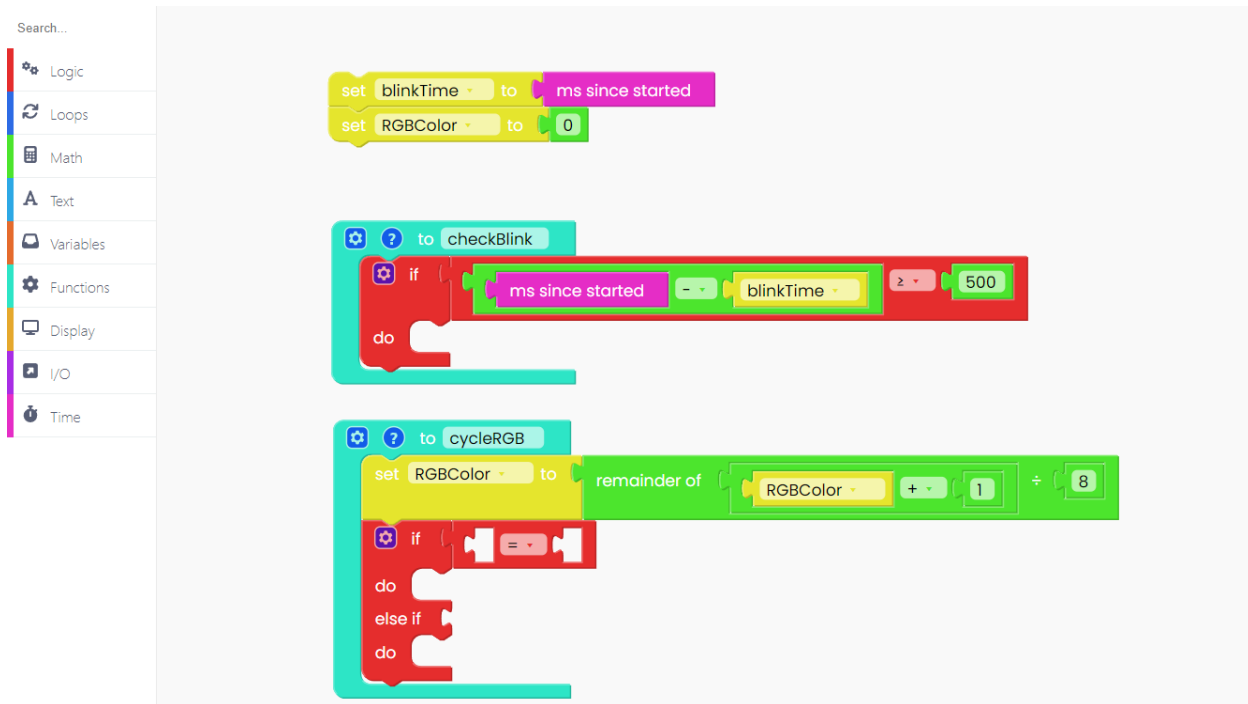
We'll switch the number '64' with another block from the "Math" section. Find a block that has number '1' on each side with a '+' symbol in the middle. Then, drag and drop that block in place of the number '64'.



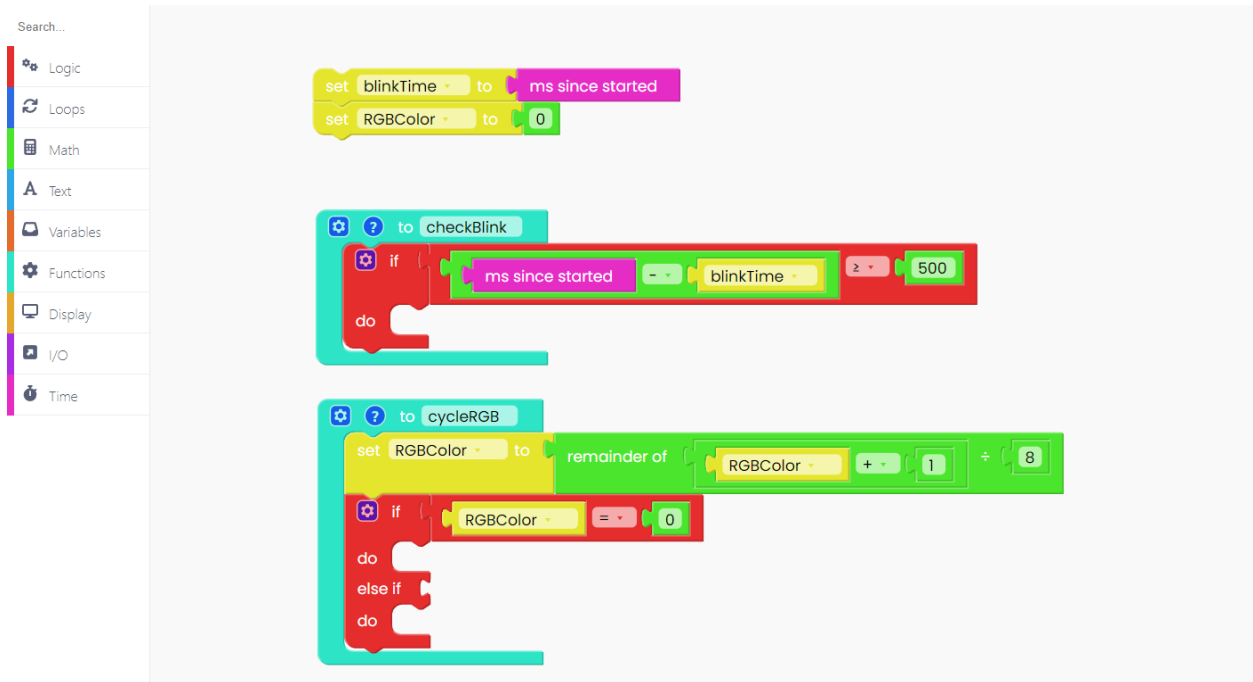
Next, go to the “Logic” section and drag and drop the “if, else” block right under the variable we just created.



Let's add some comparison logic to this new block. From the "Logic" section find a block that compares whether two number values are equal to each other and drag and drop it in place of the "true" value that's there.

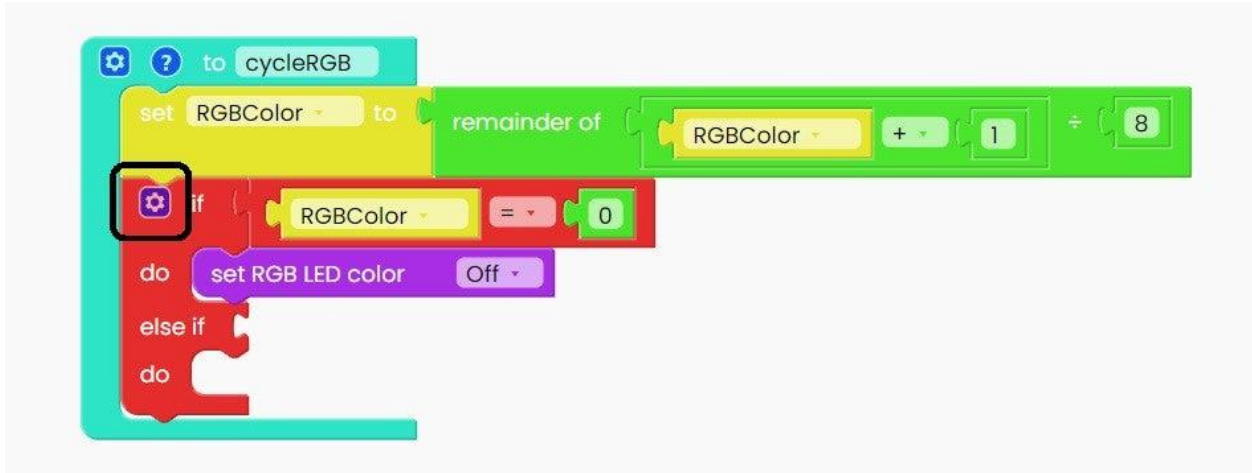


Let's add the "RGBcolor" variable in the left slot of the comparison logic. Let's also put the number '0' on the right side.

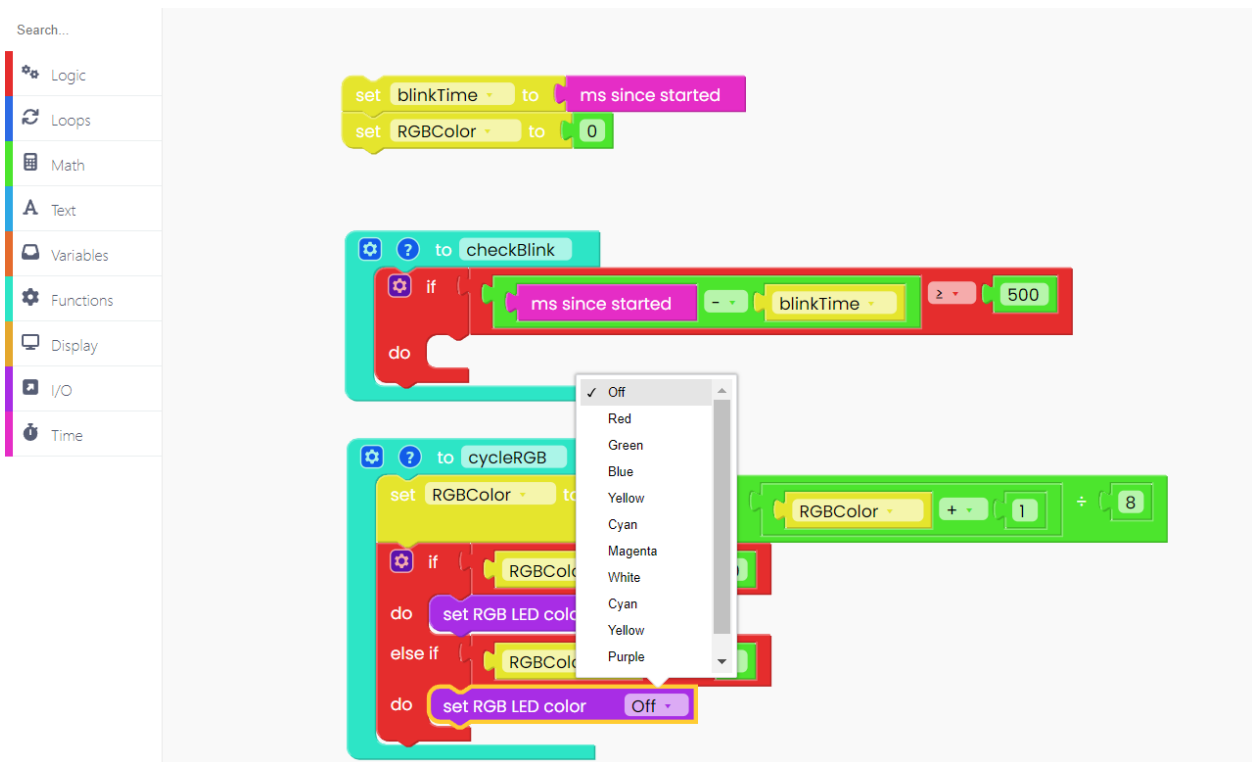


Now find the “I/O” section on the left side of the screen. From there drag and drop the “set RGB LED color” block in the empty slot of the “if else” block. Just like this:

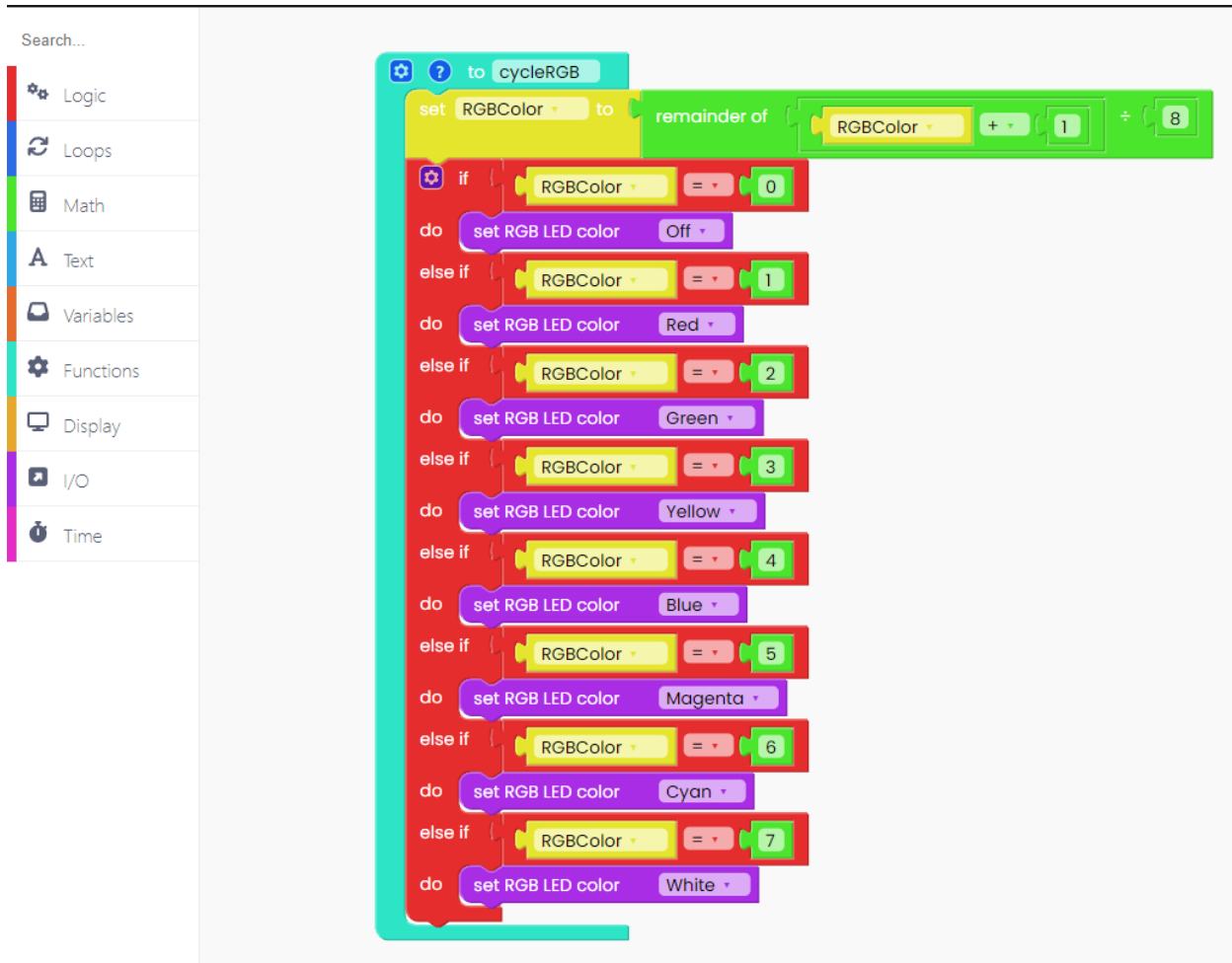
Let’s do the same for all of the colors. To add a new check click on the little ‘gear’ icon on the upper left side of the block.



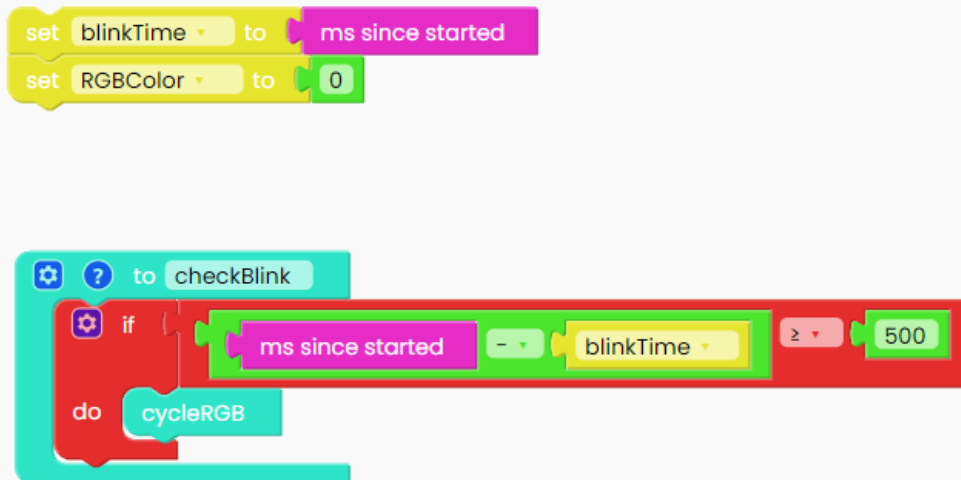
Then, drag and drop the same comparison block as in the previous step and add the "RGBcolor" variable. This time instead of '0' add the value '1' and set the "RGB LED color" to "red" by selecting the red color from the drop-down menu.



Repeat this same process for the remaining 6 colors. With each new block, make sure to increase the value by 1 and change the color into a different one. Just like this:

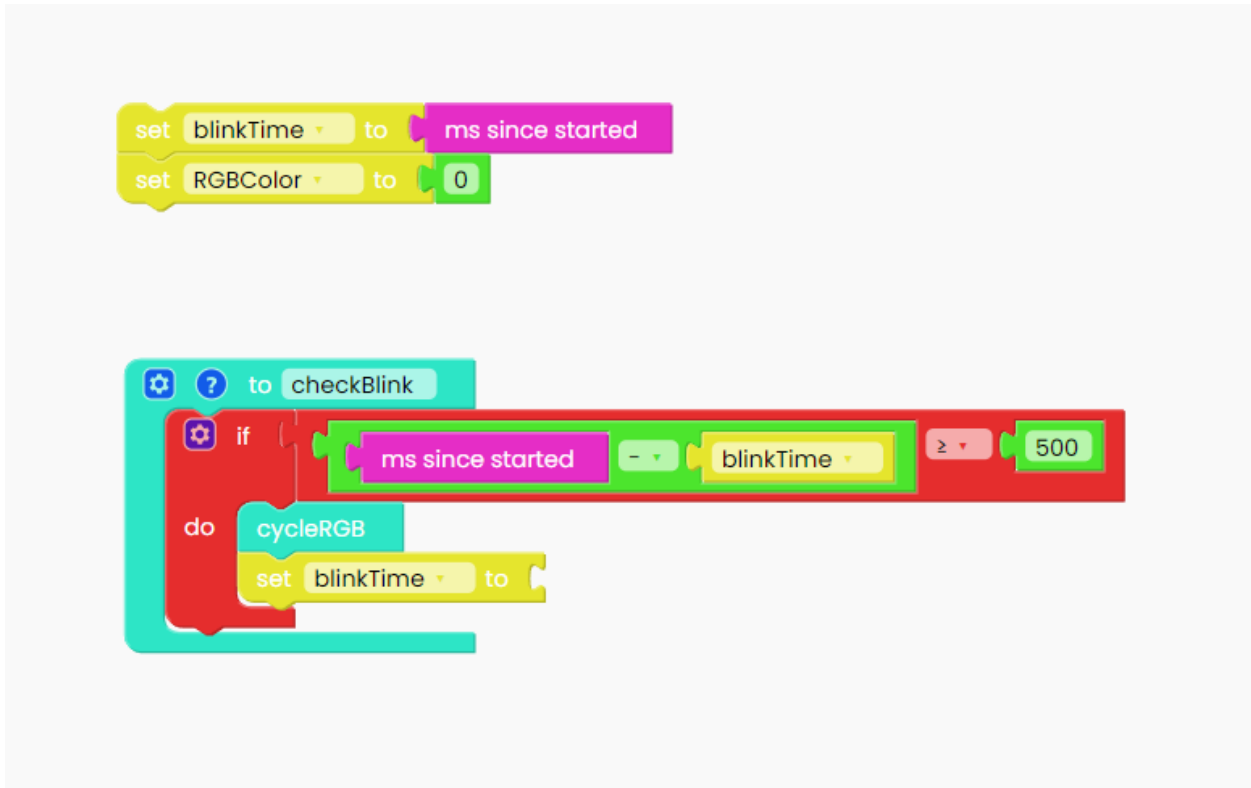


Now let's go back to our "checkBlink" function. Add the "cycleRGB" function from the "Functions" section to the empty slot of the "if" logic.

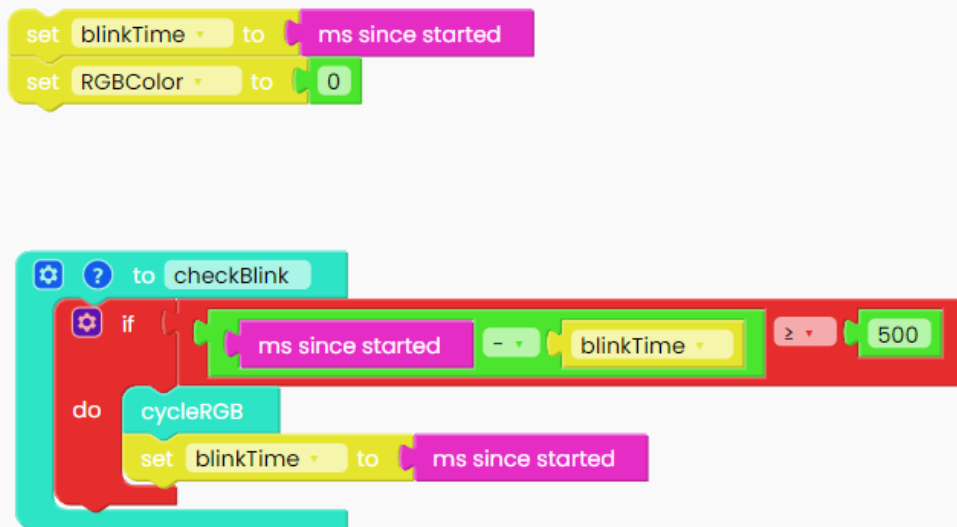


Now go to the “Variables” section and drag the “set” variable.





We'll now add a block from the "Time" section. Look for the "ms since started" block and place it inside the variable.



There's one final block we need to add. Go to the "Loops" section and drag and drop the "Loop forever" block.

Add the "checkBlink" function inside it.

The image shows a Scratch script with the following blocks:

- set `blinkTime` to `ms since started`
- set `RGBColor` to `0`
- to `checkBlink` function:
  - if `ms since started` is greater than or equal to `blinkTime` (with `500` in the input field):
    - do:
      - `cycleRGB`
      - set `blinkTime` to `ms since started`
- loop forever:
  - `checkBlink`

This is how your sketch should look like in the end:

```

set blinkTime to ms since started
set RGBColor to 0

to checkBlink
  if ms since started > blinkTime - 500
    do
      cycleRGB
      set blinkTime to ms since started
end if

loop forever
  checkBlink
end loop

to cycleRGB
  set RGBColor to remainder of (RGBColor + 1) / 8
  if RGBColor = 0
    do set RGB LED color to Off
  else if RGBColor = 1
    do set RGB LED color to Red
  else if RGBColor = 2
    do set RGB LED color to Green
  else if RGBColor = 3
    do set RGB LED color to Yellow
  else if RGBColor = 4
    do set RGB LED color to Blue
  else if RGBColor = 5
    do set RGB LED color to Magenta
  else if RGBColor = 6
    do set RGB LED color to Cyan
  else if RGBColor = 7
    do set RGB LED color to White
  end if
end to cycleRGB

```

This sketch will make you Wheelson's RGB light blink in different colors every 0.5 seconds and it will draw the camera on the screen. So, let's test it out and see if it works!

Click on the Run button and check it out.

If your code was compiled and uploaded successfully, Wheelson's white LED headlights and the RGB light underneath the screen should blink in different colors every 500 milliseconds and you should be able to see the camera on the screen.



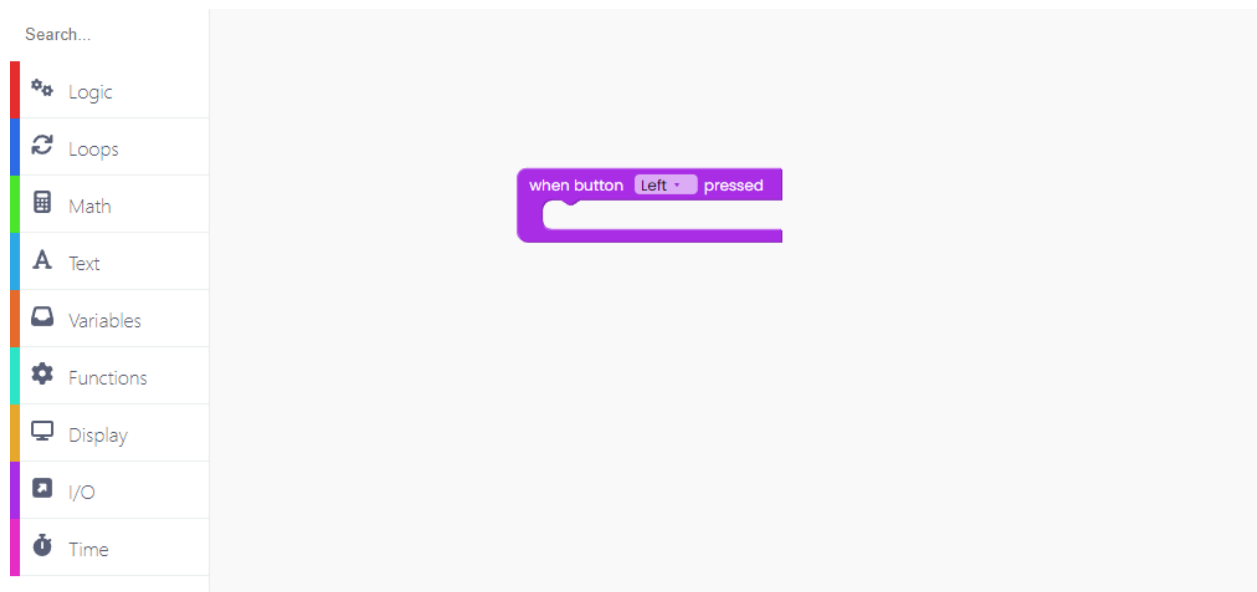
Is everything ok? Great, let's move on.

# Let's get movin'!

In this chapter, we'll learn how to code your Wheelson's motors to go forward or backward, or to move the wheels in place.

This seems like a lot of work, but don't worry.

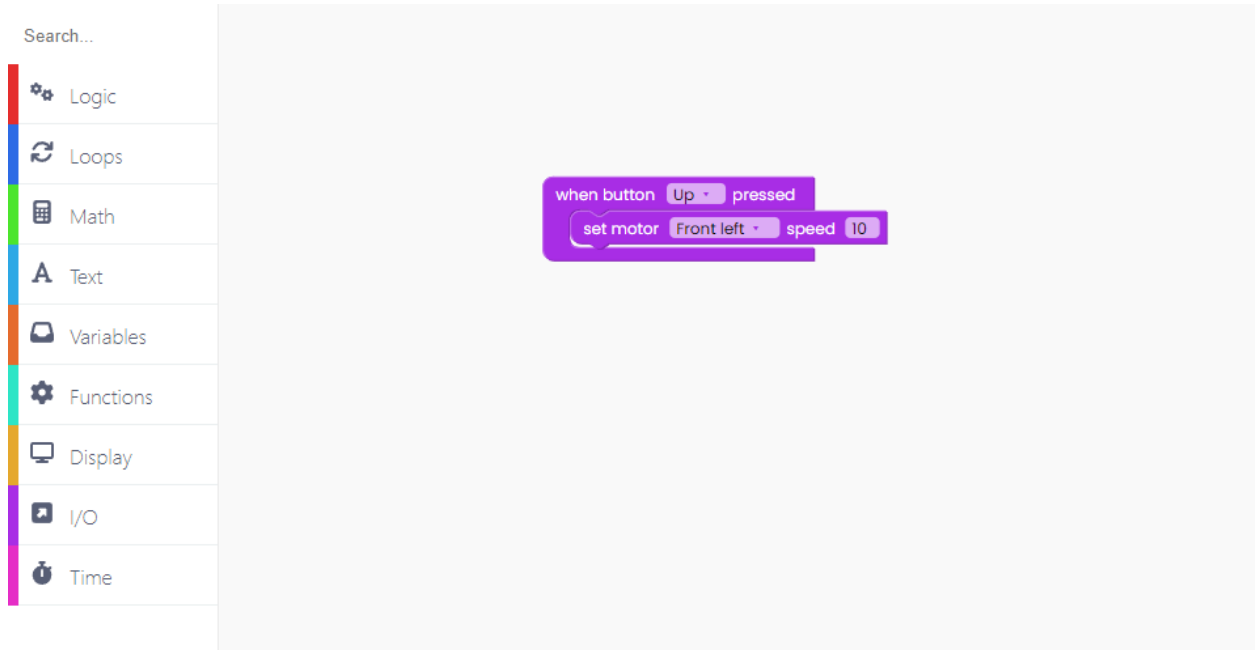
To begin, find this I/O block labeled "When button left pressed" and drop it on the drawing area.



Let's change the "left" to "up".

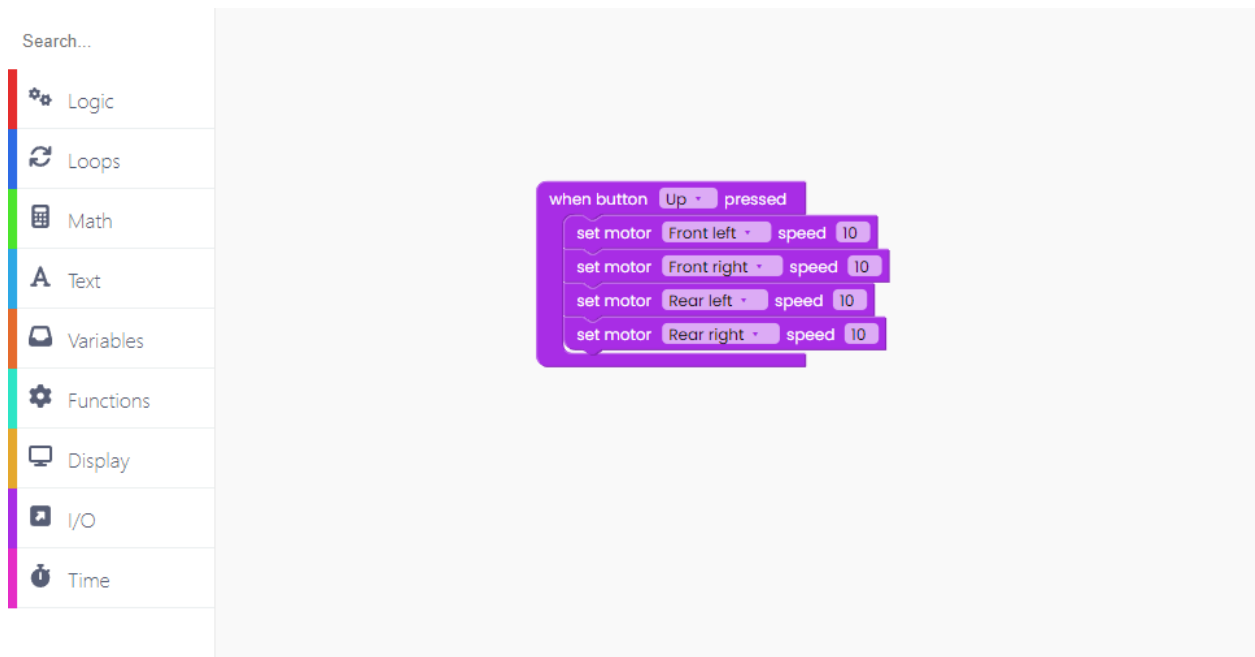
Now we'll code what happens to each motor when you click the button.

To do that, find this I/O block:



Make sure the speed value for this button is set to 10.

Repeat this for all four motors.

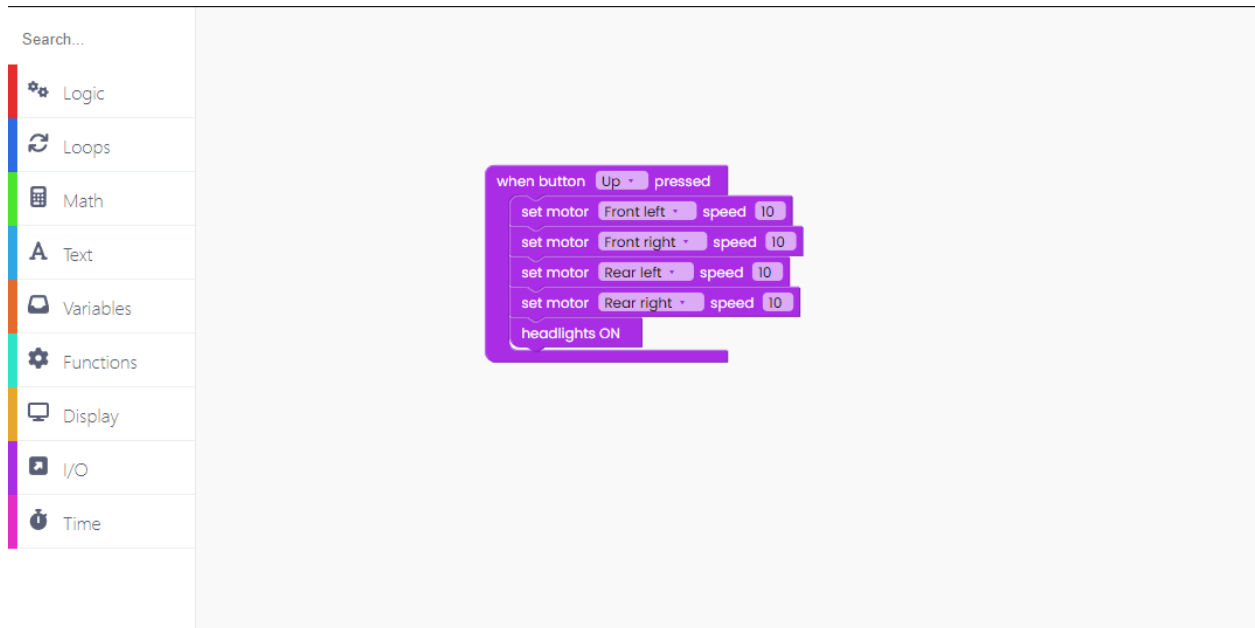


So, what we did here is we coded your Wheelson to move forward once the Up button is pressed.



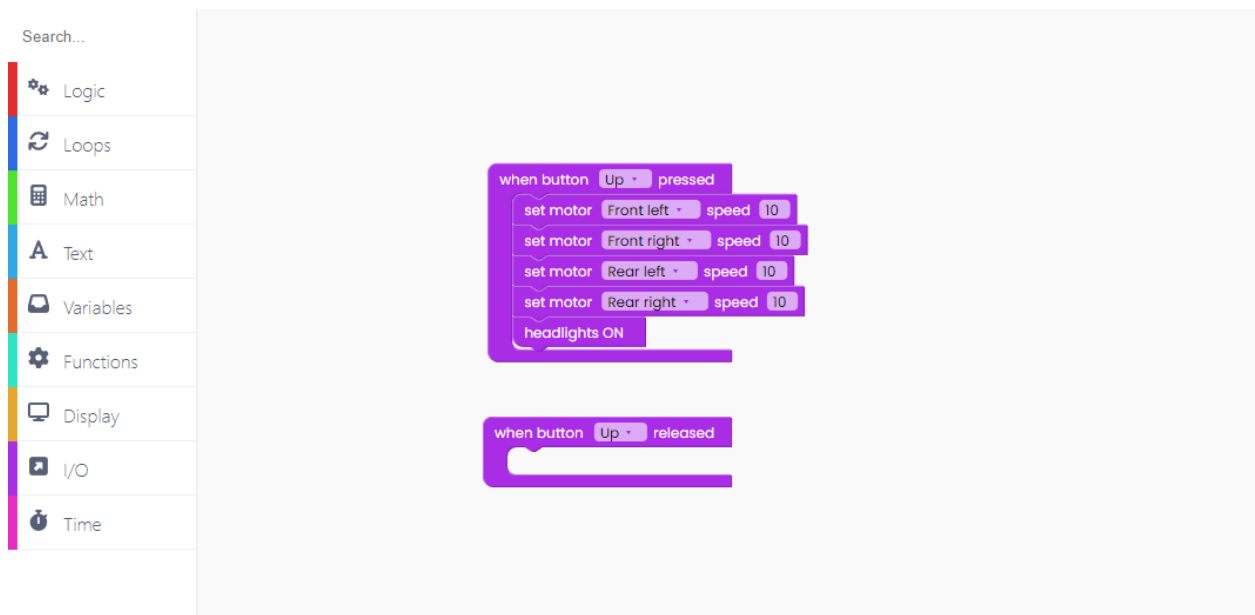
Another thing we'd like to do is turn on the headlights.

For that, you'll need this I/O block:



The screenshot shows a block-based programming environment. On the left is a sidebar with a search bar and categories: Logic, Loops, Math, Text, Variables, Functions, Display, I/O, and Time. The main workspace contains a purple block titled "when button Up pressed". Inside this block, there are four "set motor" blocks stacked vertically, each with a dropdown menu set to "Front left", "Front right", "Rear left", and "Rear right" respectively, and a "speed" field set to "10". Below these is a "headlights ON" block.

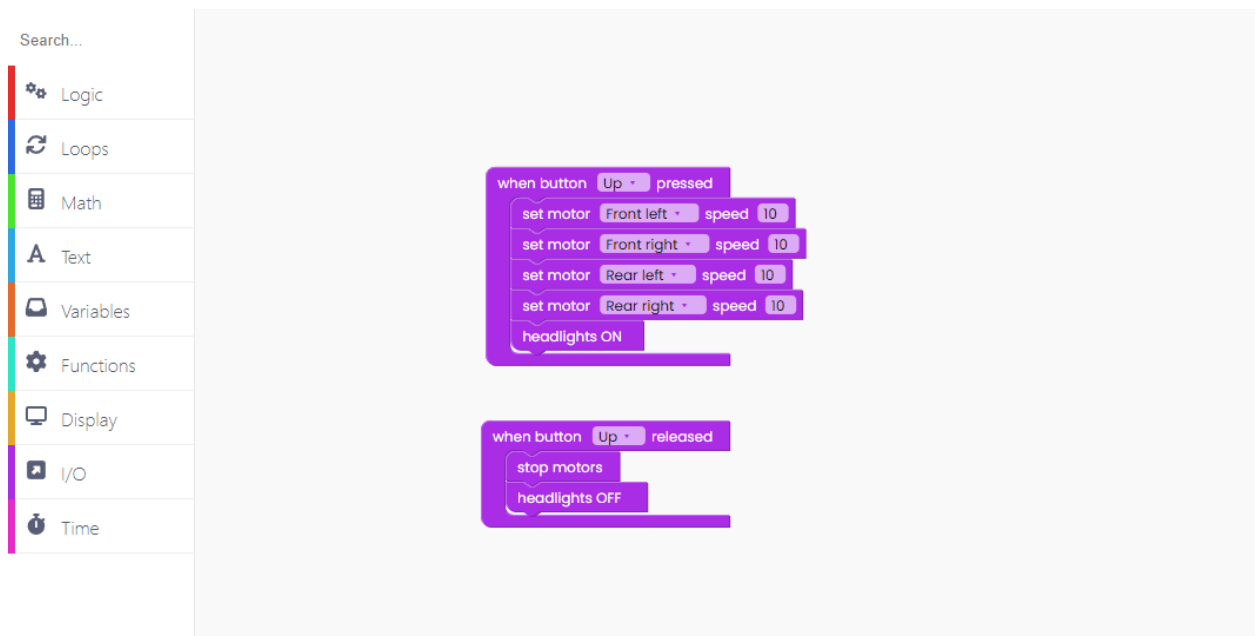
Let's see what happens when we release the button we're holding.



The screenshot shows the same block-based programming environment. The "when button Up pressed" block from the previous image is still present. Below it is a new purple block titled "when button Up released". This block is currently empty, intended for the user to add logic to stop the motors and turn off the headlights.

We want the motors to stop and the headlights to turn off once we release the button.

Find these two blocks and place them within the "When button up released" block to do this.



Let's do it again for three more pushbuttons: down, left, and right.

When we press the "down" button, we want the Wheelson to start moving backward and that's why we need to add the minus sign before the speed value.

We also want to turn on the headlights.



We want the motors to stop and the headlights to switch off when we release the button in

question.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- Display
- I/O
- Time

when button Up pressed

- set motor Front left speed 10
- set motor Front right speed 10
- set motor Rear left speed 10
- set motor Rear right speed 10
- headlights ON

when button Down pressed

- set motor Front left speed -10
- set motor Front right speed -10
- set motor Rear left speed -10
- set motor Rear right speed -10
- headlights ON

when button Up released

- stop motors
- headlights OFF

when button Down released

- stop motors
- headlights OFF

So we coded the back-and-forth movement.

Let's code moving the wheels in place.

We'll do that with the right and left buttons.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- Display
- I/O
- Time

when button Up pressed

- set motor Front left speed 10
- set motor Front right speed 10
- set motor Rear left speed 10
- set motor Rear right speed 10
- headlights ON

when button Down pressed

- set motor Front left speed -10
- set motor Front right speed -10
- set motor Rear left speed -10
- set motor Rear right speed -10
- headlights ON

when button Up released

- stop motors
- headlights OFF

when button Down released

- stop motors
- headlights OFF

when button Right pressed

- set motor Front left speed -10
- set motor Front right speed 10
- set motor Rear left speed -10
- set motor Rear right speed 10
- headlights ON

We must set two of the motors to move backward and two to go, forwards, in order for Wheelson to move in place.

As always, turn the headlights on.

When we release the right button, we want the motors to stop working and the headlights to turn off.

The image shows a Scratch code editor interface with a sidebar on the left containing categories: Logic, Loops, Math, Text, Variables, Functions, Display, I/O, and Time. The main workspace contains four event-driven code blocks:

- when button Up - pressed:** set motor Front left - speed 10, set motor Front right - speed 10, set motor Rear left - speed 10, set motor Rear right - speed 10, headlights ON.
- when button Down - pressed:** set motor Front left - speed -10, set motor Front right - speed -10, set motor Rear left - speed -10, set motor Rear right - speed -10, headlights ON.
- when button Up - released:** stop motors, headlights OFF.
- when button Down - released:** stop motors, headlights OFF.

Below these, there are two more event-driven code blocks:

- when button Right - pressed:** set motor Front left - speed -10, set motor Front right - speed 10, set motor Rear left - speed -10, set motor Rear right - speed 10, headlights ON.
- when button Right - released:** stop motors, headlights OFF.

Repeat the process for the left button, but make sure to make different motors going back and forth.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- Display
- I/O
- Time

The image displays six Scratch code blocks arranged in a 3x2 grid, each triggered by a button event. The top-left block is for the 'Up' button pressed, setting Front left and Rear left motors to speed 10, and Front right and Rear right motors to speed 10, with headlights ON. The top-right block is for the 'Down' button pressed, setting Front left and Rear left motors to speed -10, and Front right and Rear right motors to speed -10, with headlights ON. The middle-left block is for the 'Up' button released, stopping all motors and turning headlights OFF. The middle-right block is for the 'Up' button released, stopping all motors and turning headlights OFF. The bottom-left block is for the 'Right' button pressed, setting Front left and Rear left motors to speed -10, and Front right and Rear right motors to speed 10, with headlights ON. The bottom-right block is for the 'Left' button pressed, setting Front left and Rear left motors to speed 10, and Front right and Rear right motors to speed -10, with headlights ON. The bottom-most block is for the 'Right' button released, stopping all motors and turning headlights OFF.

The final step is to include the block that determines what happens when the left button is released.



Finally, we must include the "loop forever" block with the "scan buttons" inside it to guarantee that the buttons are always scanned and that the code is executing properly.

Please remember to include this section every time you code something with the buttons.

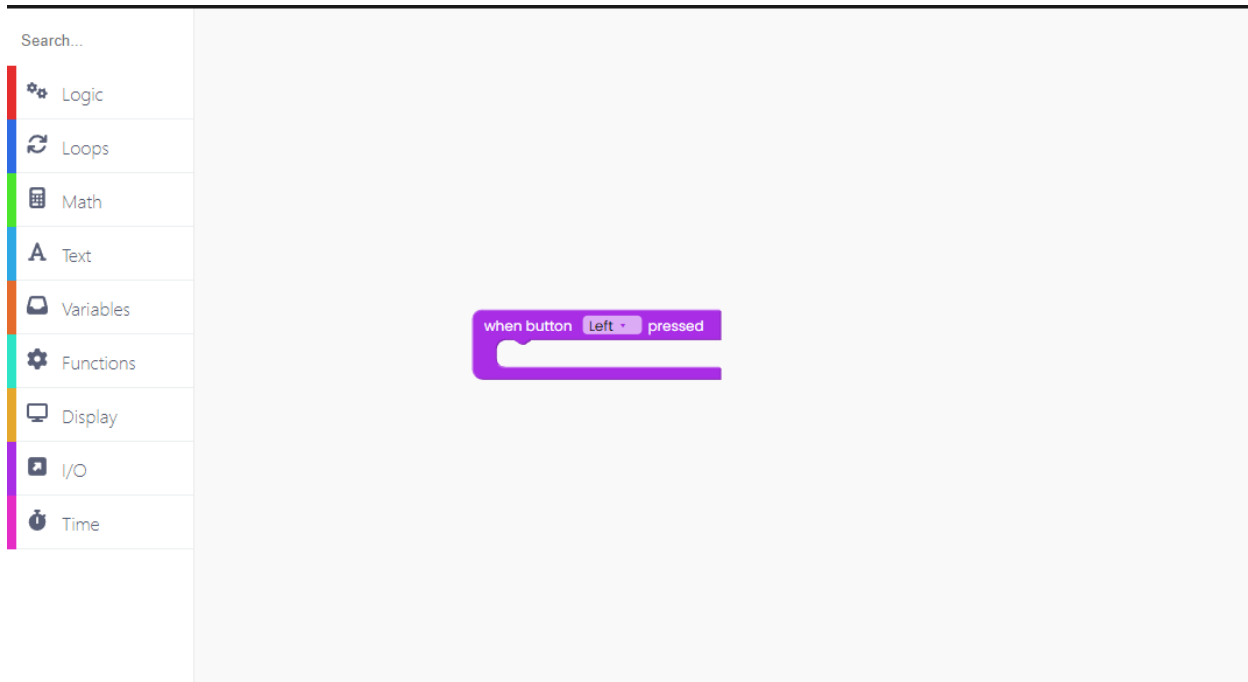
Click on the Run button and try it out.

## Buttons

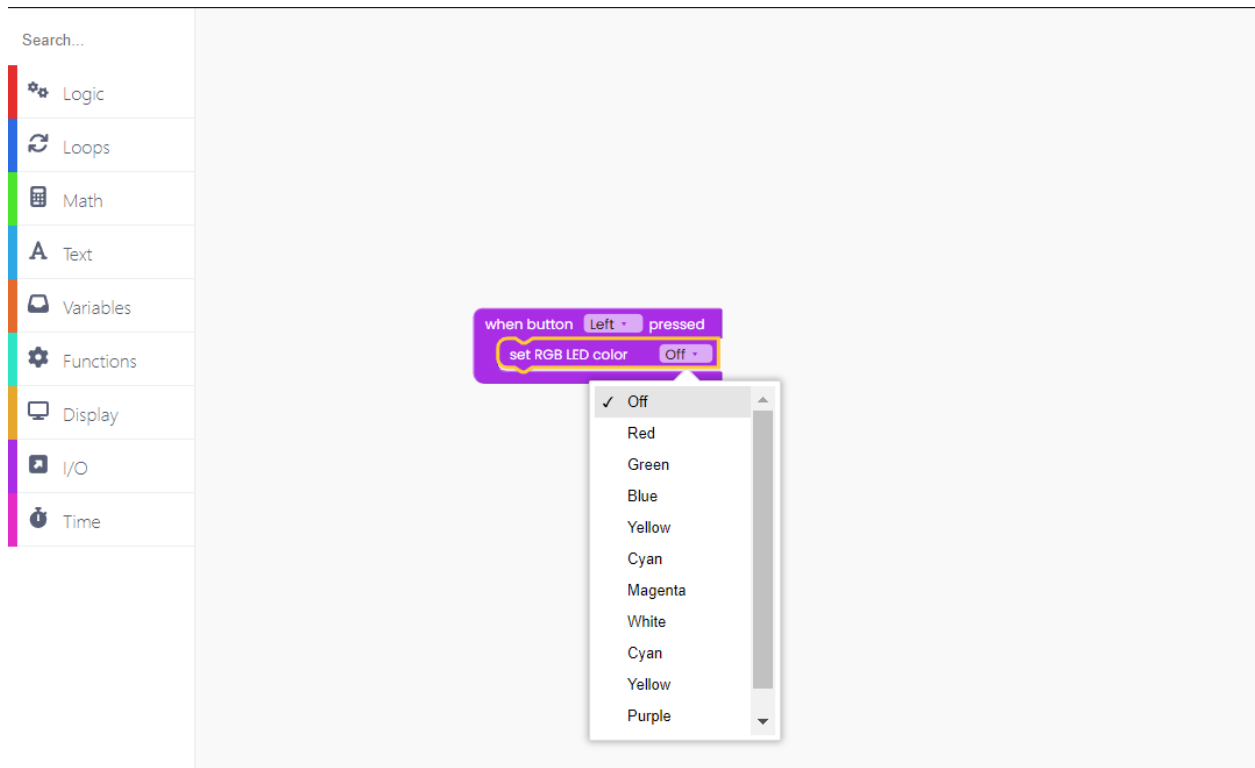
Let's write code to light the LED in the chosen color when a specific button is pressed.

This will be done mainly in the I/O block section.

Drag and drop the "When button left pressed" block from the I/O block section here:

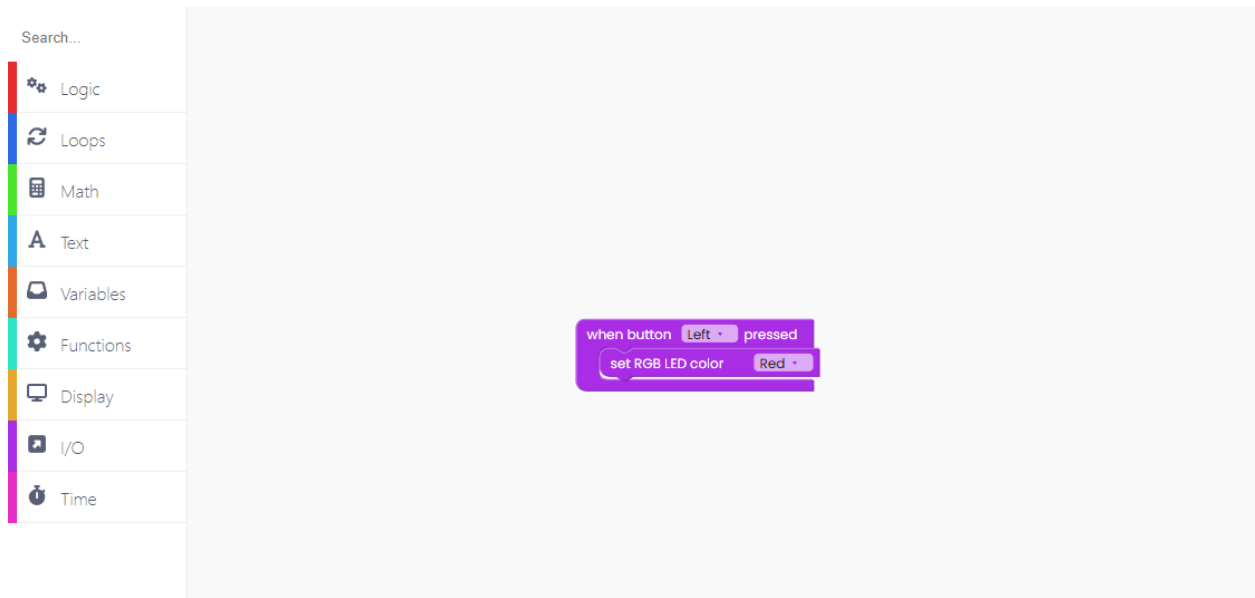


Then, find the "set RGB LED color off" block and insert it in the block above.



Click on "off" and you'll see a drop-down menu showing you all the different colors you can choose from.

For the first one, choose red.





You can duplicate this 5 times - one block for each button.

You can duplicate the block by right-clicking it and selecting "Duplicate".

This is what your code should look like right now:



Change which button you press and which color the LEDs light up.

As a result, when each button is pressed, the LED changes color:

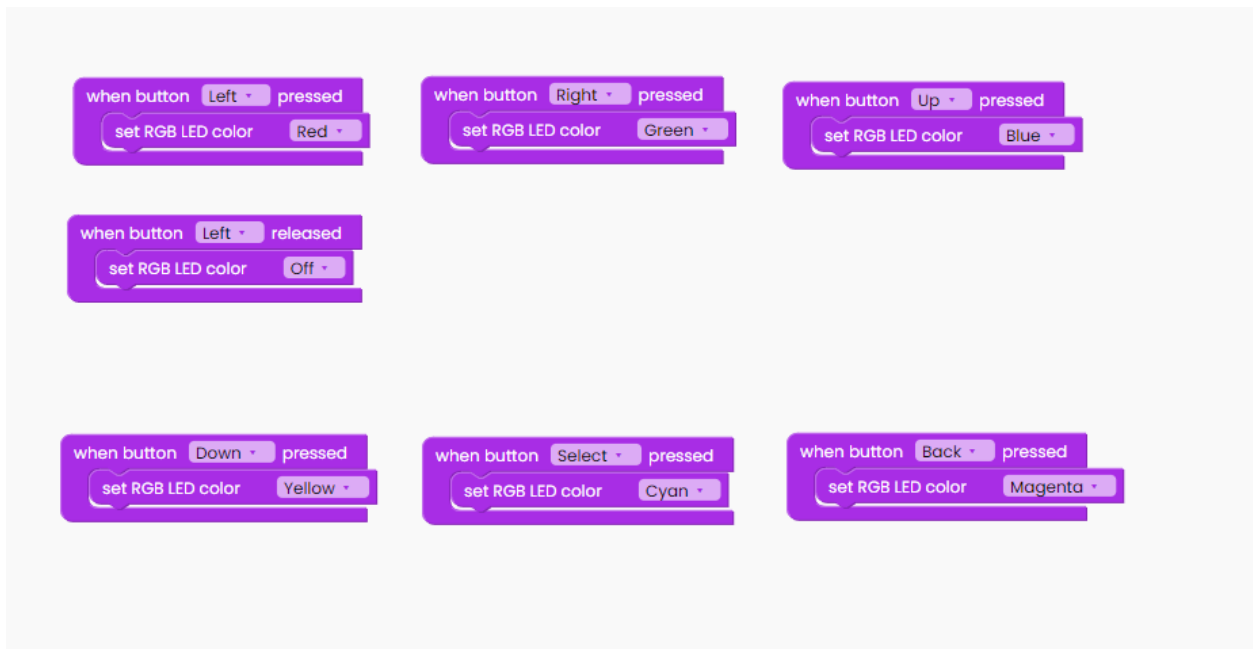


Return to the I/O block section and select the "When button left released" block. Drag it onto the drawing area.



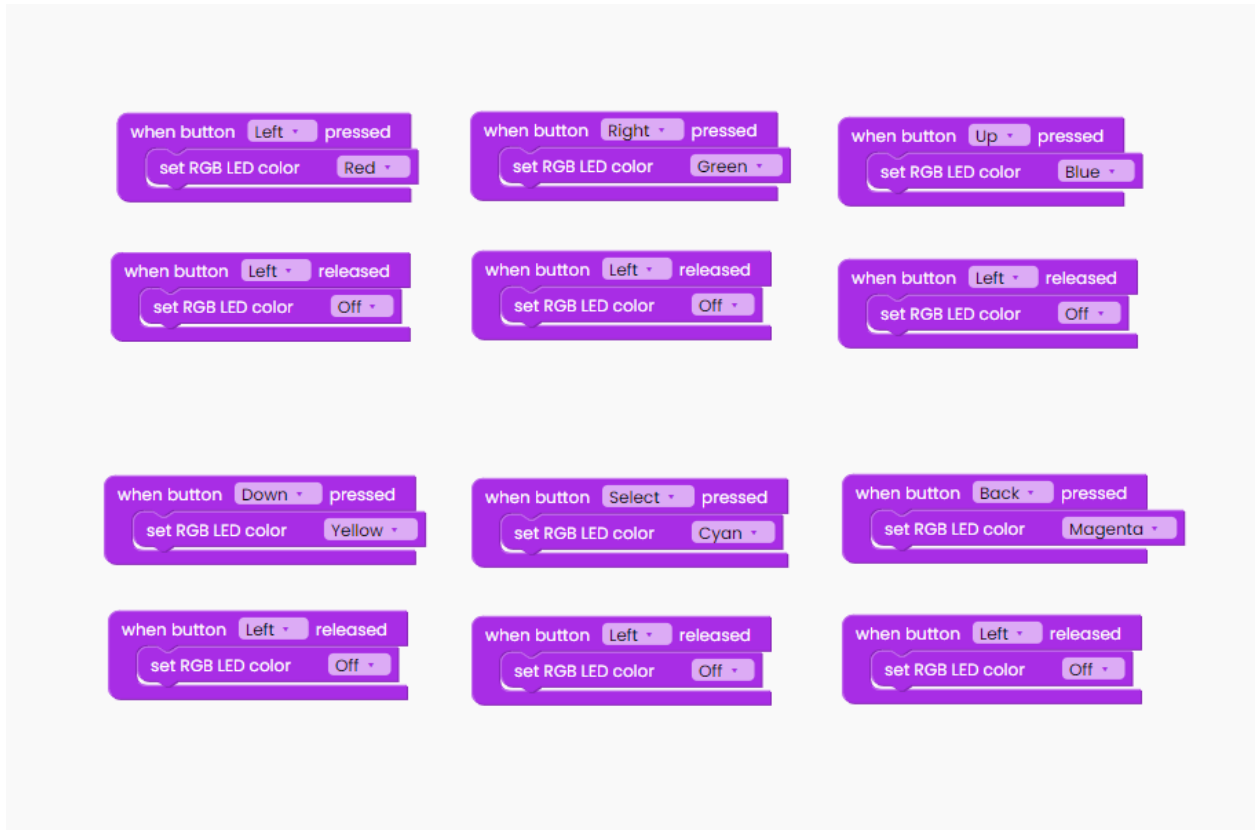
As you can assume, for each button-pressed action, we'll code what happens when the button is released.

Each "released" block will include the identical "set RGB LED color" block.

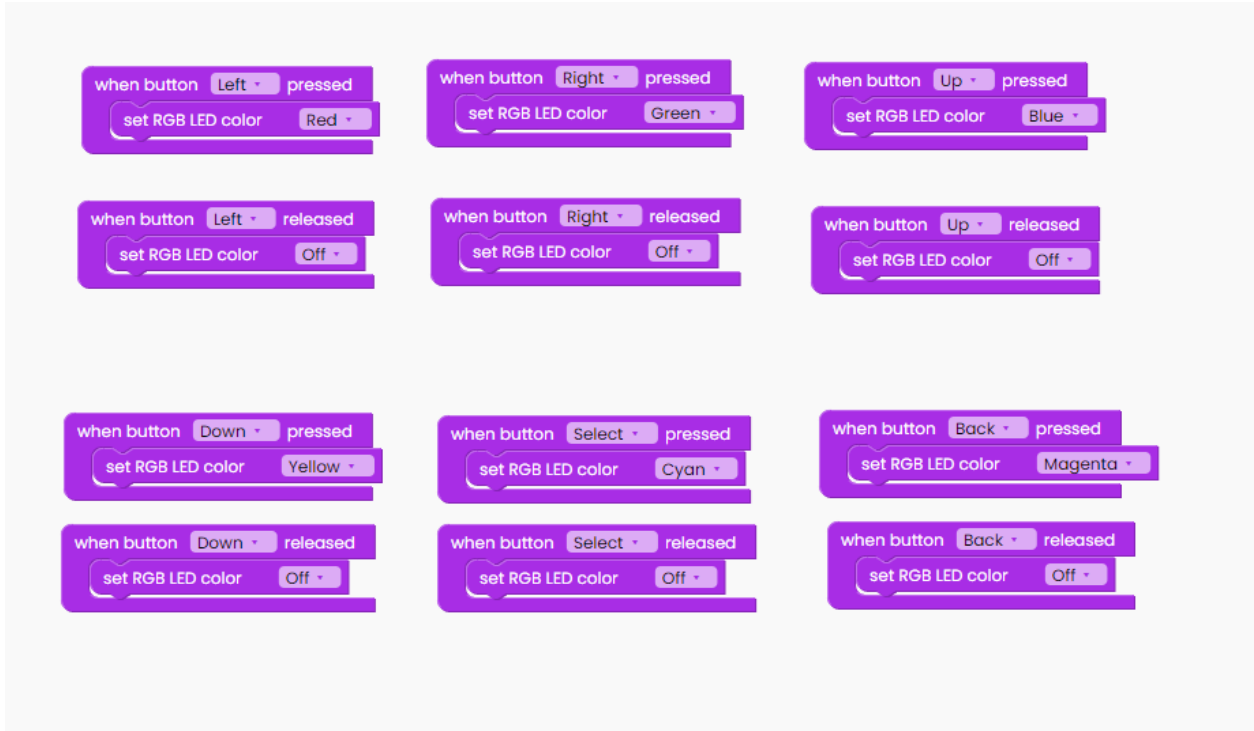


Let's duplicate the "released" block 5 times.

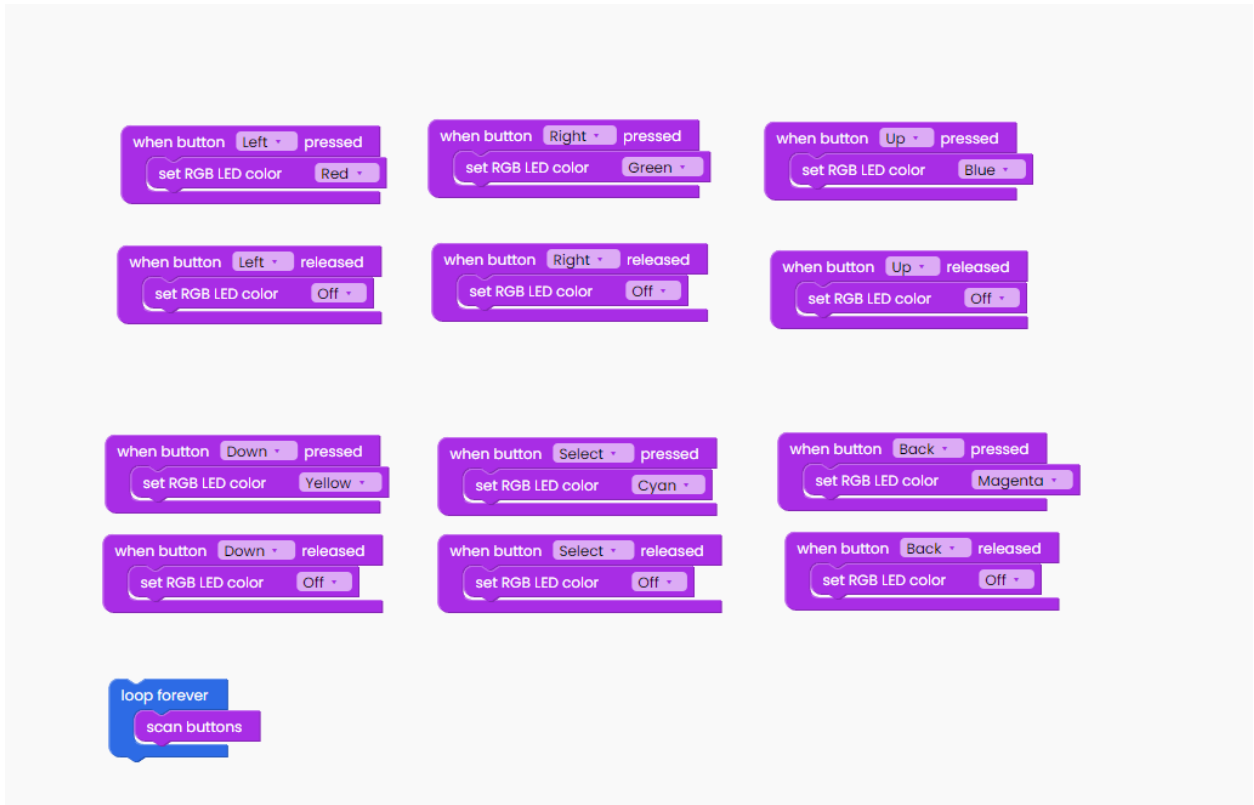
This is what your code should look like right now:



We want the LED to turn off when we release the button, so we'll keep the "off" section of the block, but we'll replace the buttons in question.



The "scan buttons" block inside the "loop forever" block is the last item we need to add. This will ensure that this action is carried out at all times.



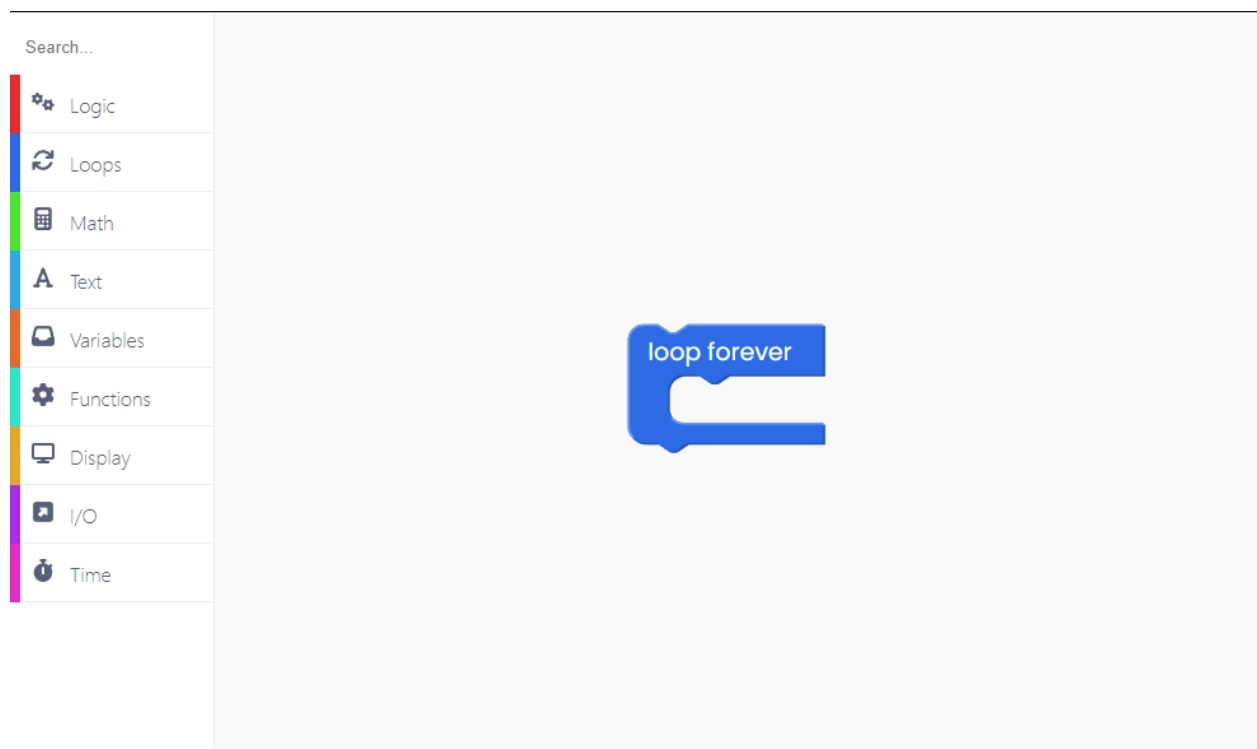
Click on the Run button, and start pressing and releasing the buttons.

## Simple Timer

Now we'll learn how to code the ongoing switching of the back and front lights, as well as the filling of the display with black and white.

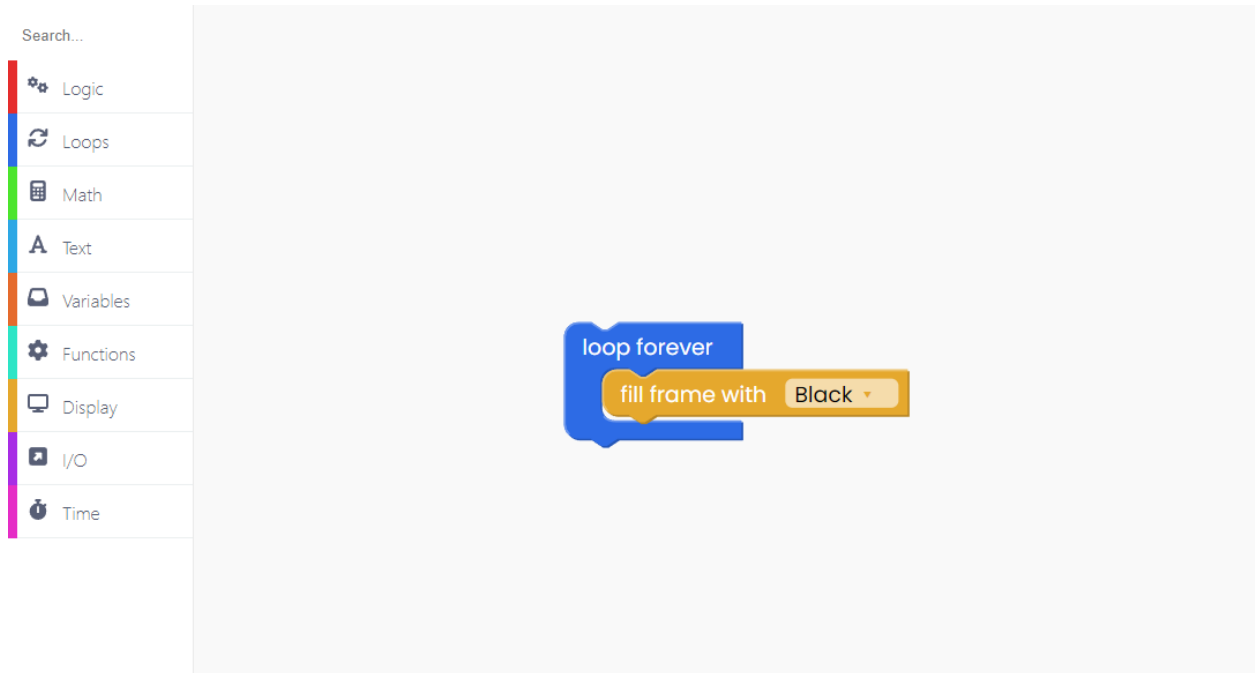
Let's kick off!

To begin, look for the "loop forever" block inside the "loops" block section.

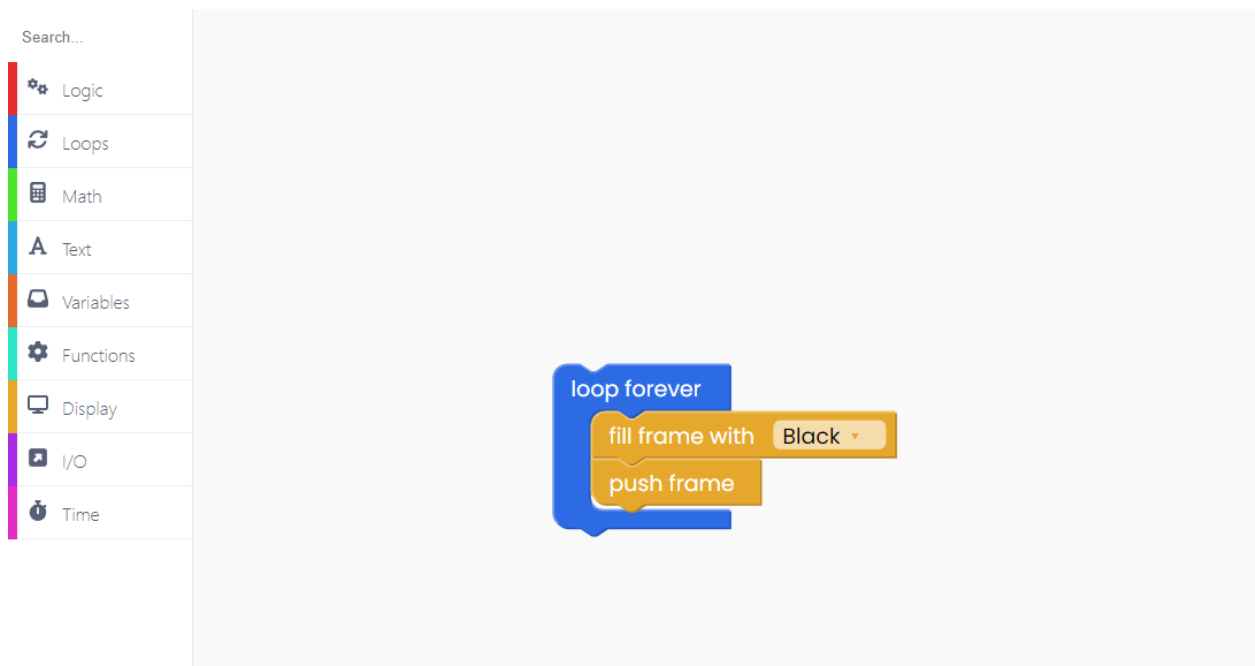


The first step is to make the display completely dark.

You can do that by adding the "display" block.

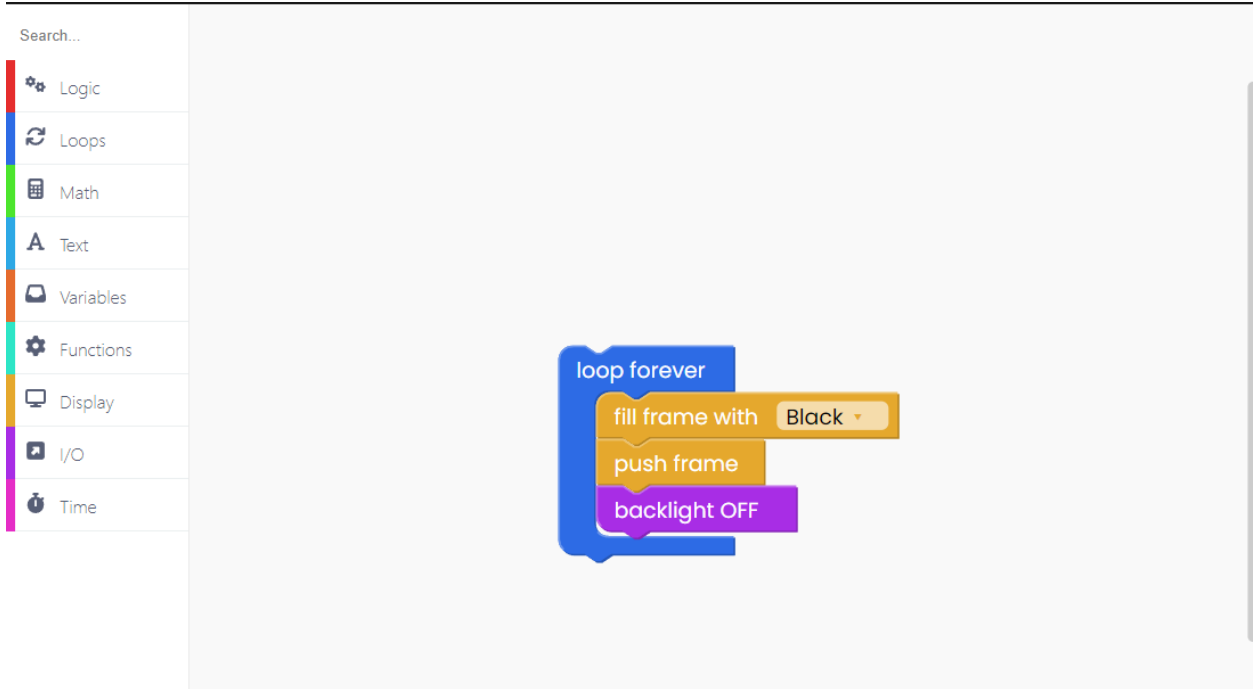


Find another "display" block labeled "push frame" and place it below like this:



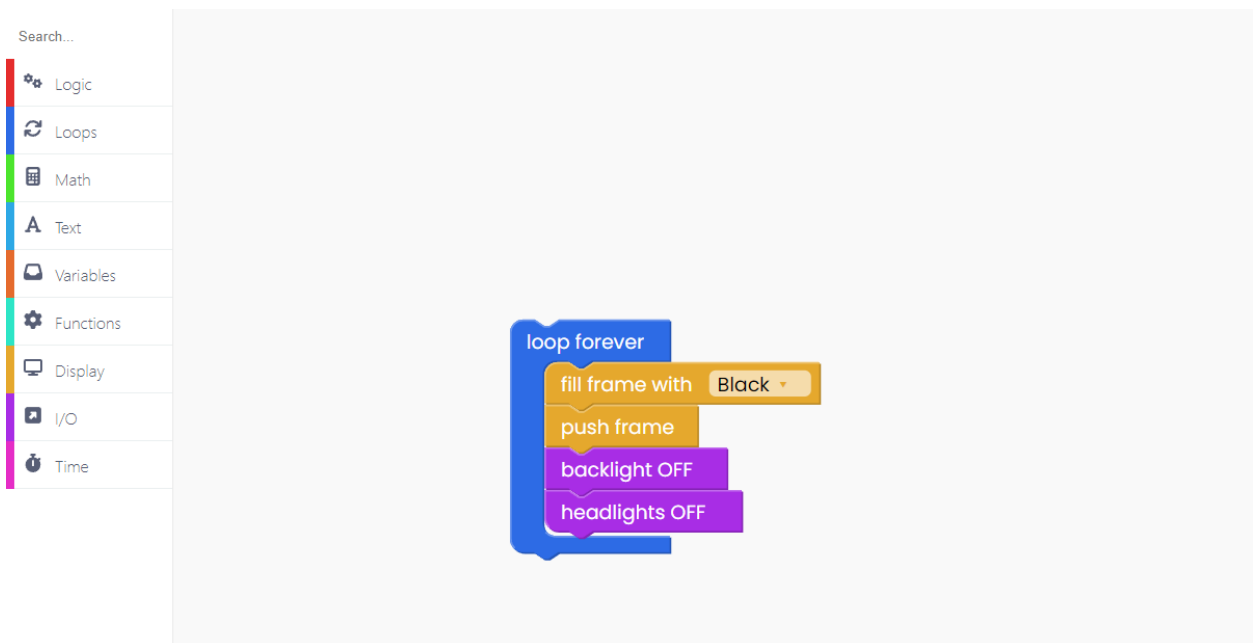
When this happens, we will turn off the backlights on your Wheelson.

For that, you have to use an I/O block labeled "backlight off".



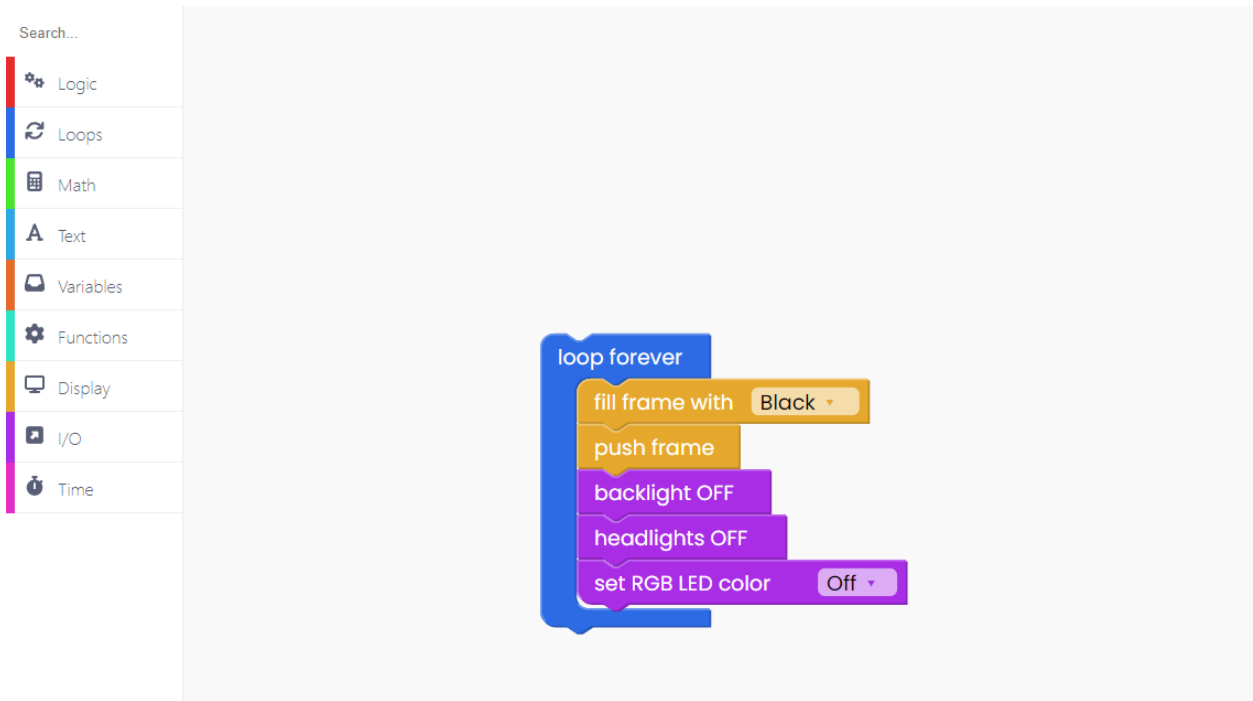
We'll also switch off the frontlights.

Find another I/O block and place it below:



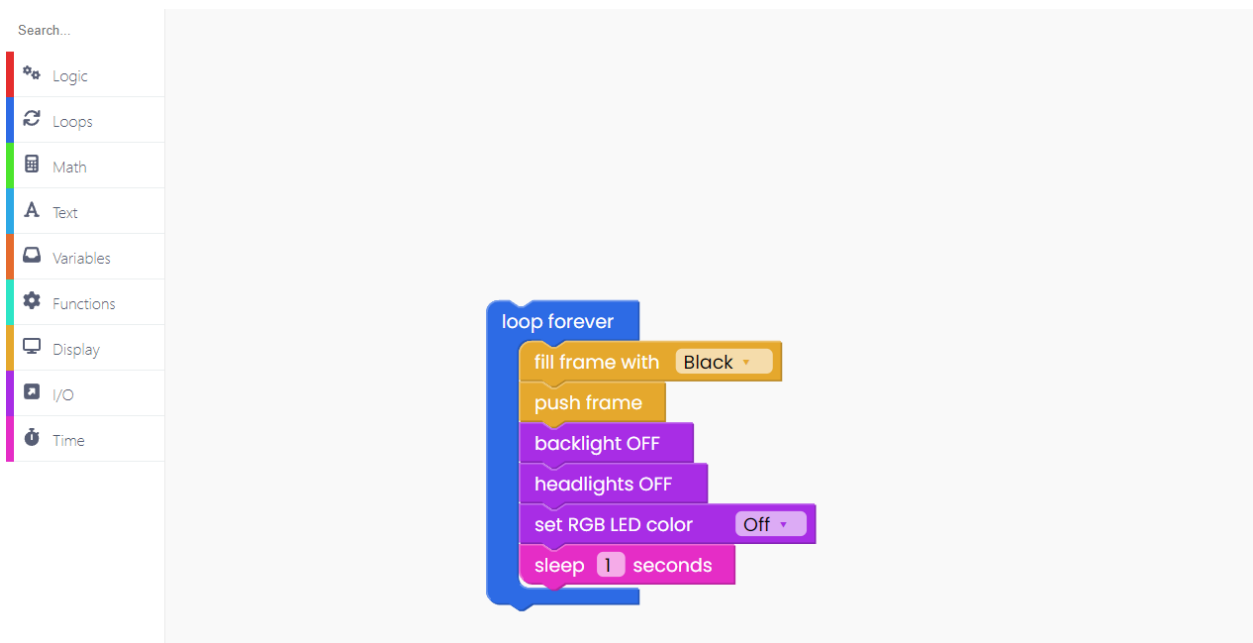


Let's turn off the LED as well.

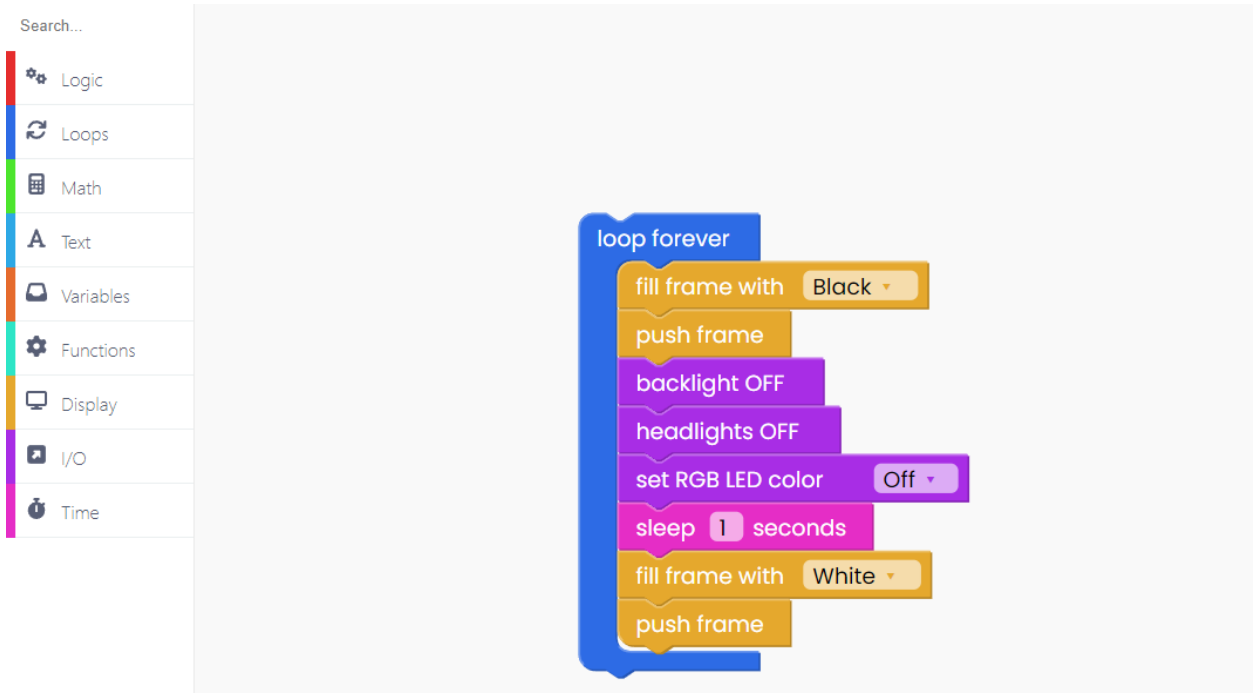


Now, jump to the "time" block section and find the "sleep 0 seconds" block.

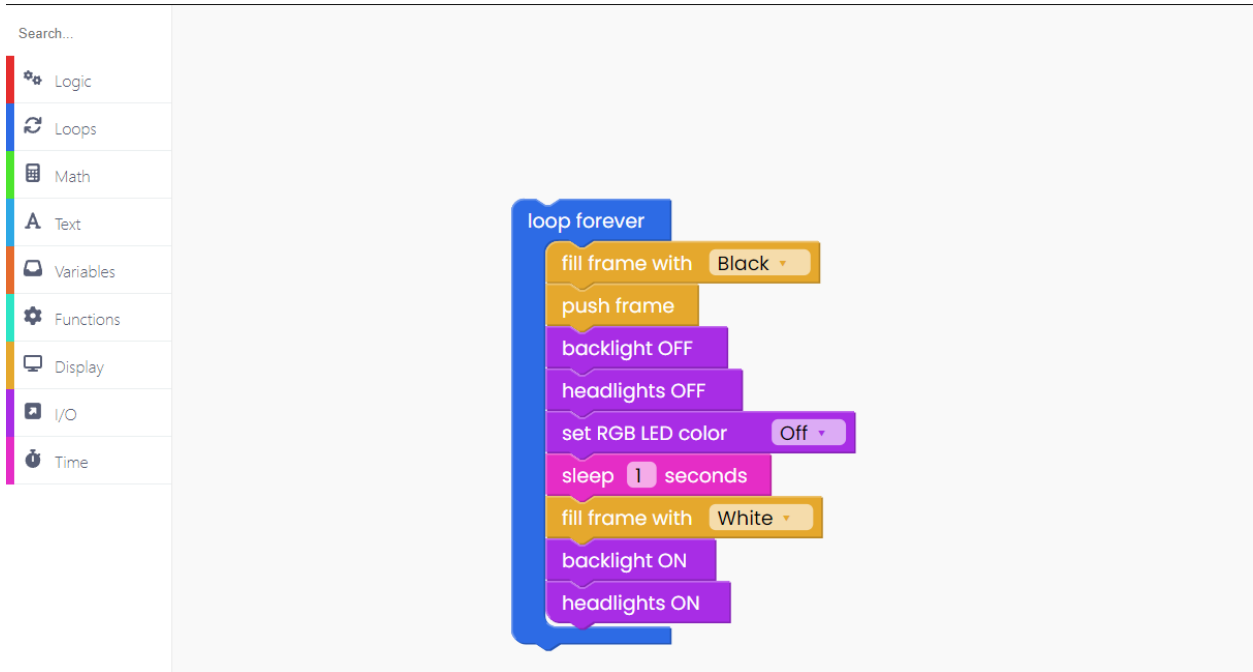
We'll put the code to sleep for one second.



We want the display to turn white after 1 second.



We also want all of the lights to turn on.



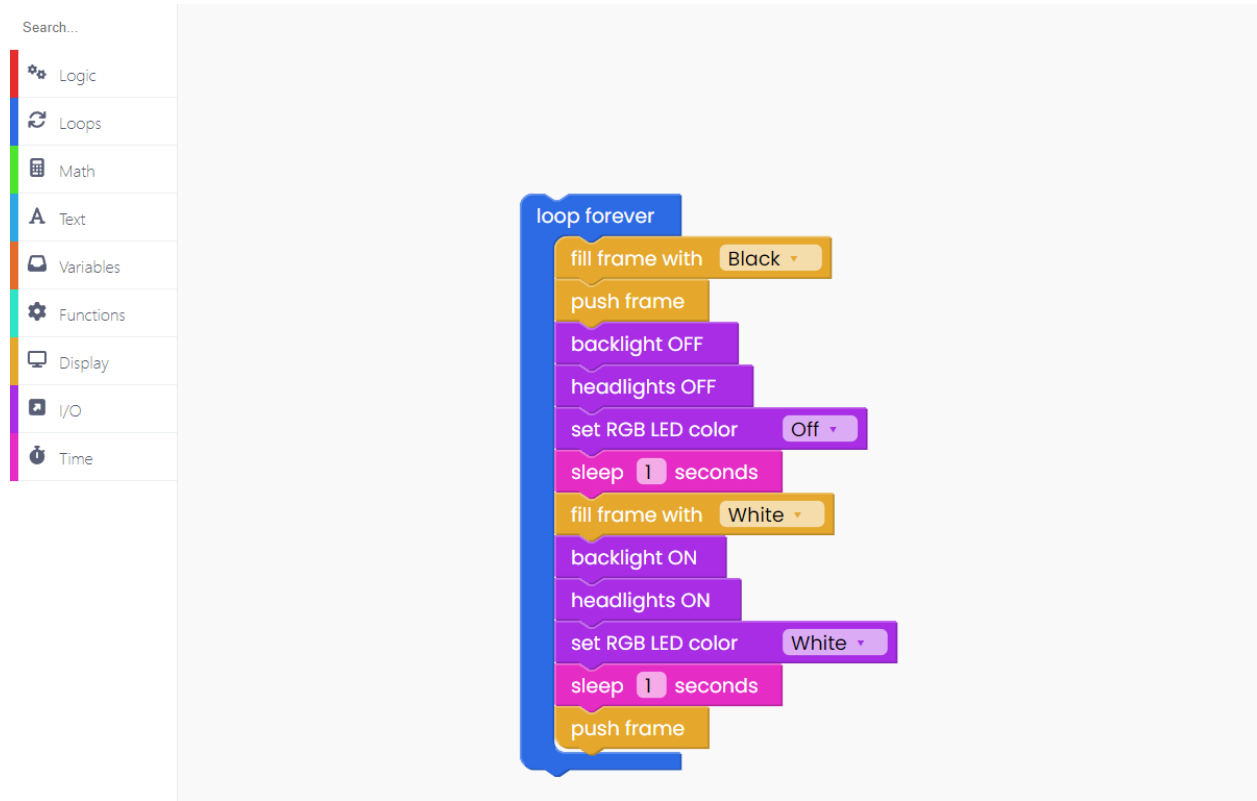
Once the back and front lights are turned on, the LED should light white.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- Display
- I/O
- Time

```
loop forever
  fill frame with Black
  push frame
  backlight OFF
  headlights OFF
  set RGB LED color Off
  sleep 1 seconds
  fill frame with White
  backlight ON
  headlights ON
  set RGB LED color White
```

In the end, add another "time" block and the "push frame" block.

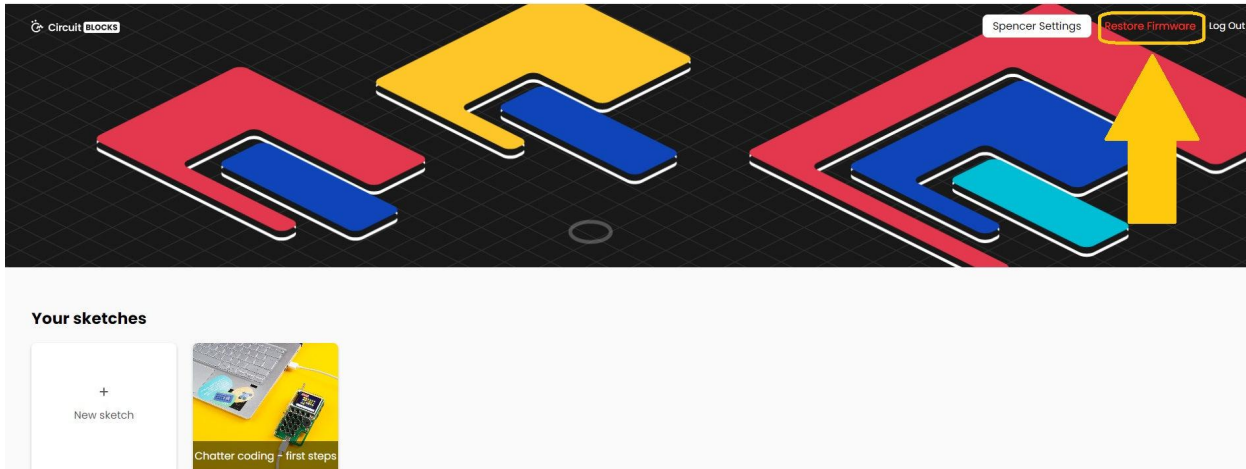


Click on the Run button, and check the lights and the display.

## Restoring Wheelson's firmware

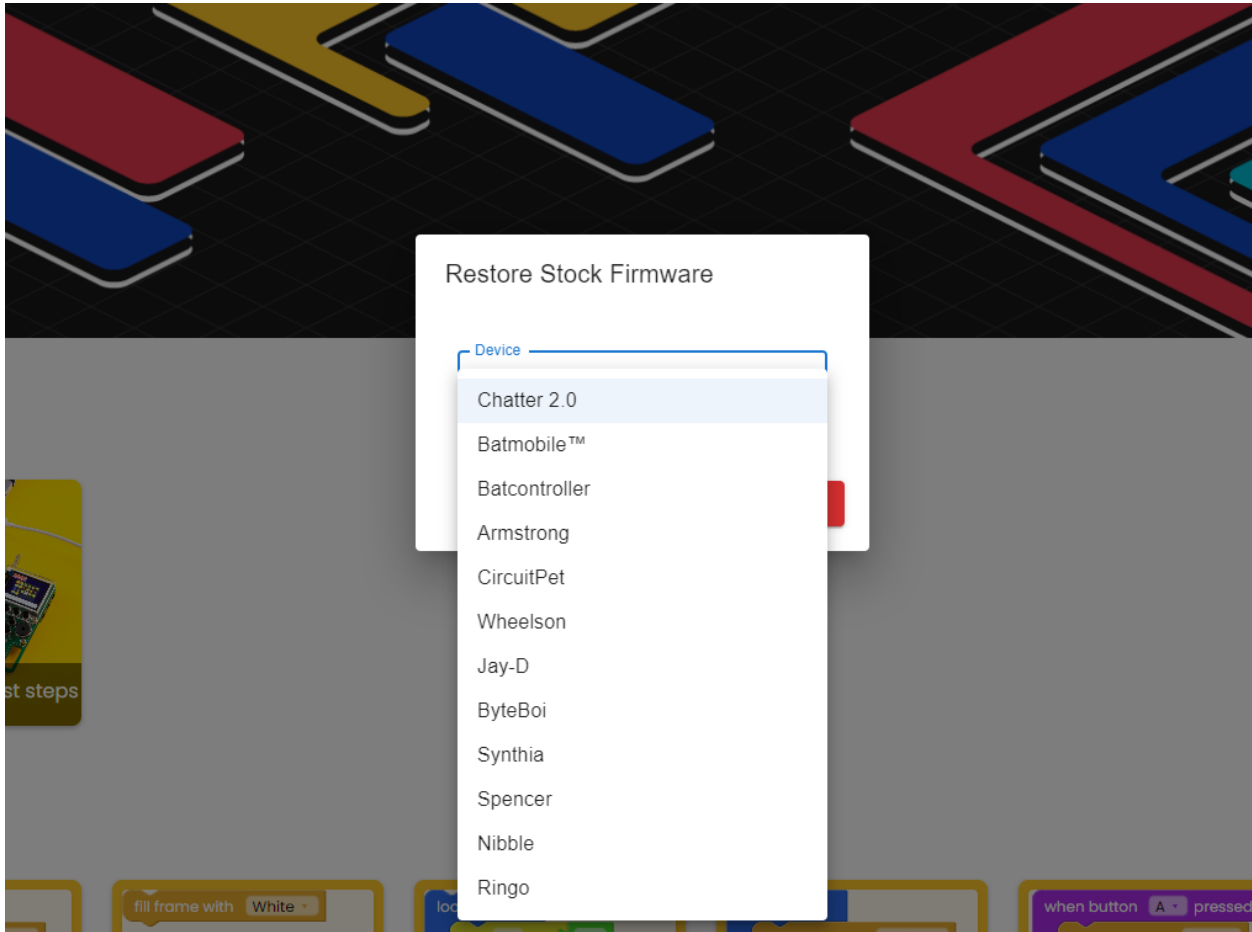
If you want to restore the firmware on your Wheelson, follow these steps.

This is quite simple, just connect your Wheelson to the USB port of your computer and press the "Restore firmware" button on the top right.



You will be prompted with a window where you need to choose the device that you are restoring the firmware for.

Choose Wheelson, of course.



Wait for a few seconds, and your Wheelson will be back and running like usual.

## What's next?

You've reached the end of our first Wheelson coding tutorial, congratulations!

We hope you're as excited as we are about Wheelson's future since there are so many cool things we want to do with it in the future firmware and CircuitBlocks updates.

In the meantime, continue exploring on your own and show us what you've done with Wheelson's lights, wheels, marker, and ball detection by sharing it on the CircuitMess

community forum: <https://community.circuitmess.com/>

If you need any help with your device, as always, reach out to us via [contact@circuitmess.com](mailto:contact@circuitmess.com) and we'll help as soon as we can.

Thank you and keep making!

