Coding for beginners - how to code your Synthia

Let's start! Step by step

In front of you are four sketches that will introduce you to the coding world!

Connect Synthia with your PC, open CircuitBlocks, and follow these steps.

Usually, there are displays on our devices, but we decided to change it a bit and put an LED matrix on Synthia.

First things first!

Let us introduce to you a few important terms:

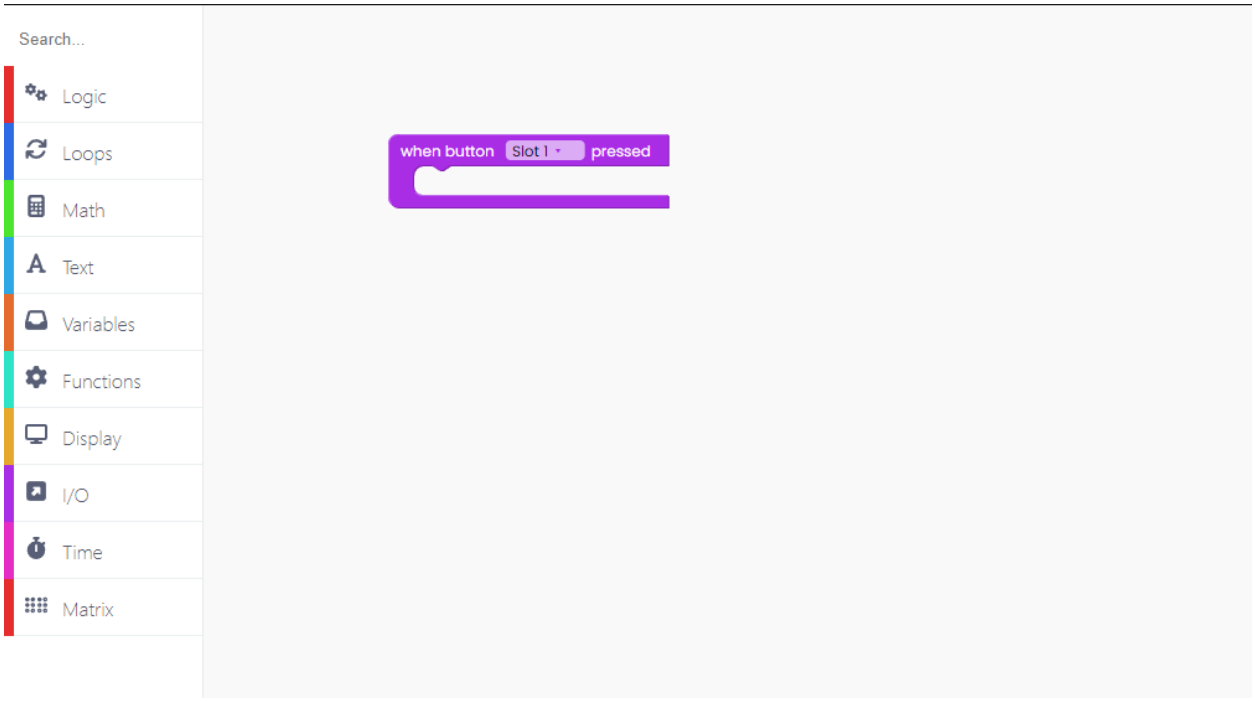
- Track -> those are the LEDs in the middle of Synthia (the one with the largest number of LEDs)
- Cursor -> the white row under the track
- Sliders -> next to the sliders

Click on the new sketch in CircuitBlocks and choose Synthia since that is the device we'll be coding today.

You can name your first sketch "Buttons" because that's what we'll be focusing on right now.

Your Synthia has five pushbuttons, which we'll use to turn on the LEDs on the track matrix.

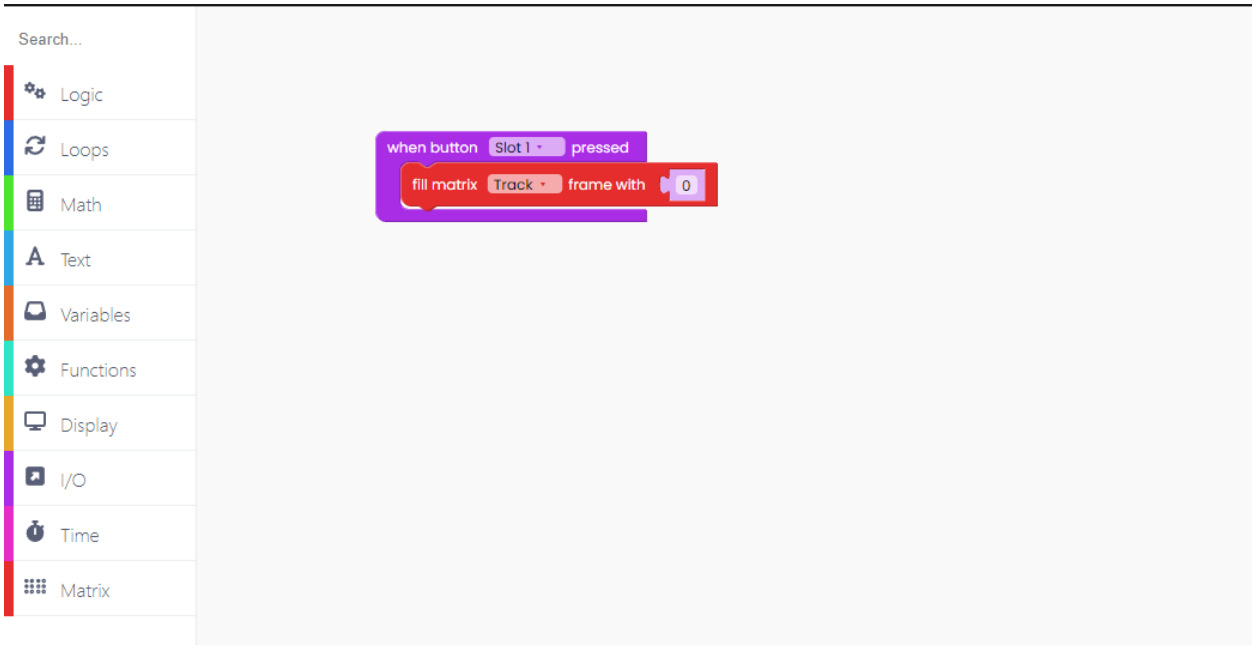
We'll need the I/O block labeled "When button pressed" for this.



The whole code that determines what happens when that specific button is pressed will be included within the I/O block.

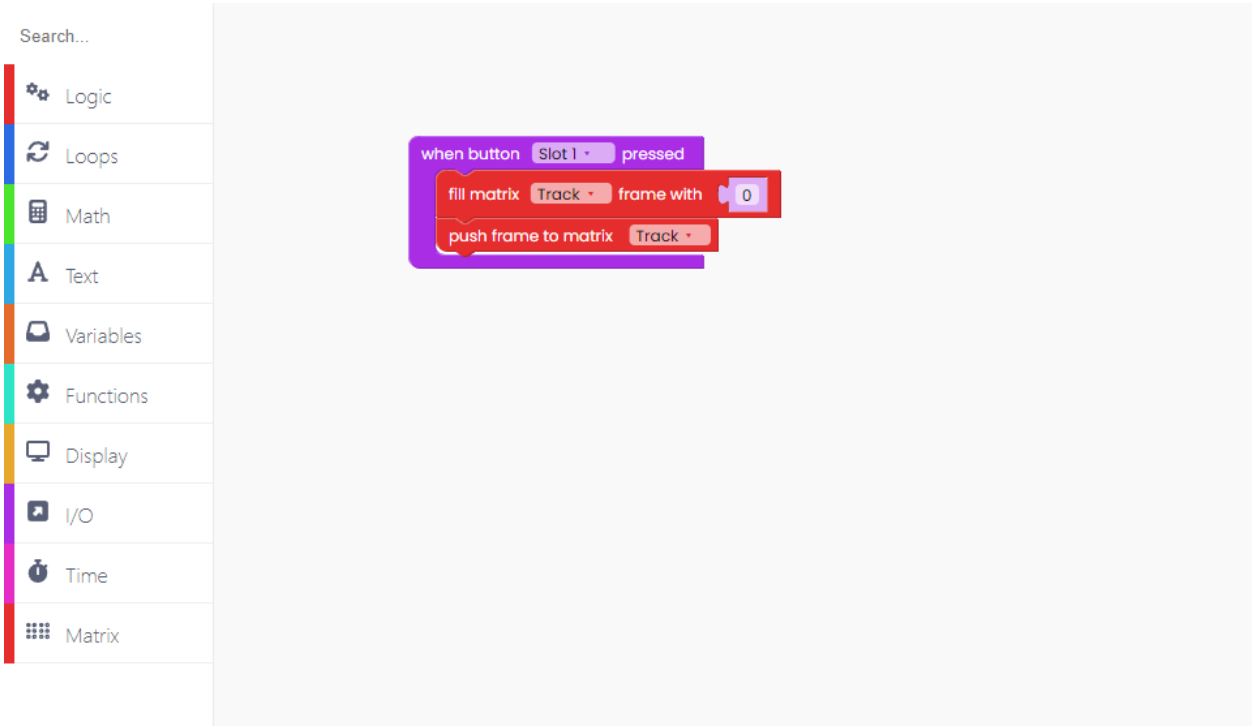
For example, we don't want any of the LEDs to light up when we hit pushbutton number 1 (or slot 1).

Find this "Matrix" block and drag it inside the I/O block:



We must leave the intensity number at 0 if we do not want any of the LEDs to light up.

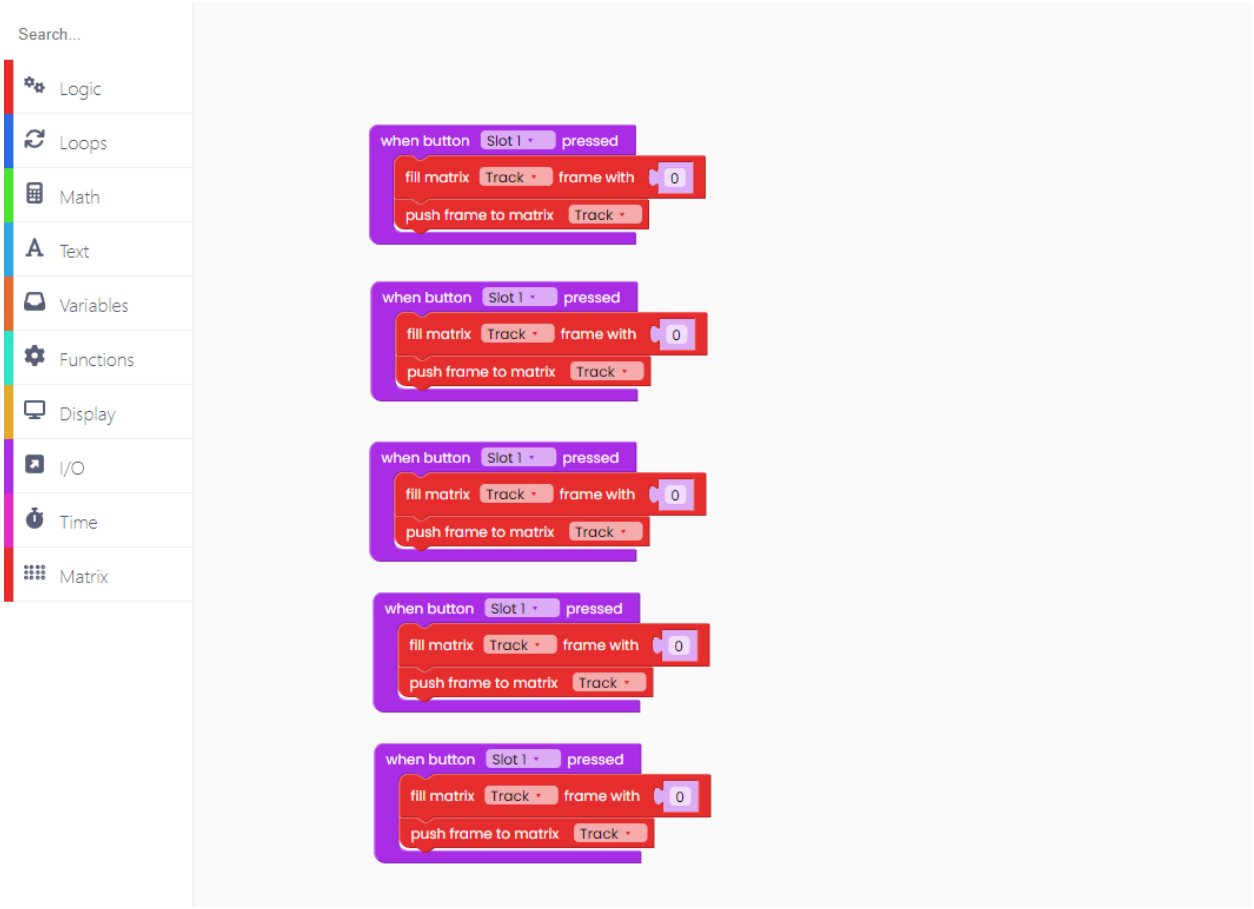
While we are drawing something on the matrix, we must add the "push frame to matrix" block at the end to ensure that this code is visible on the matrix.



You may perform the same thing for any other matrix by clicking on the "Track" button and selecting another matrix from the drop-down menu.

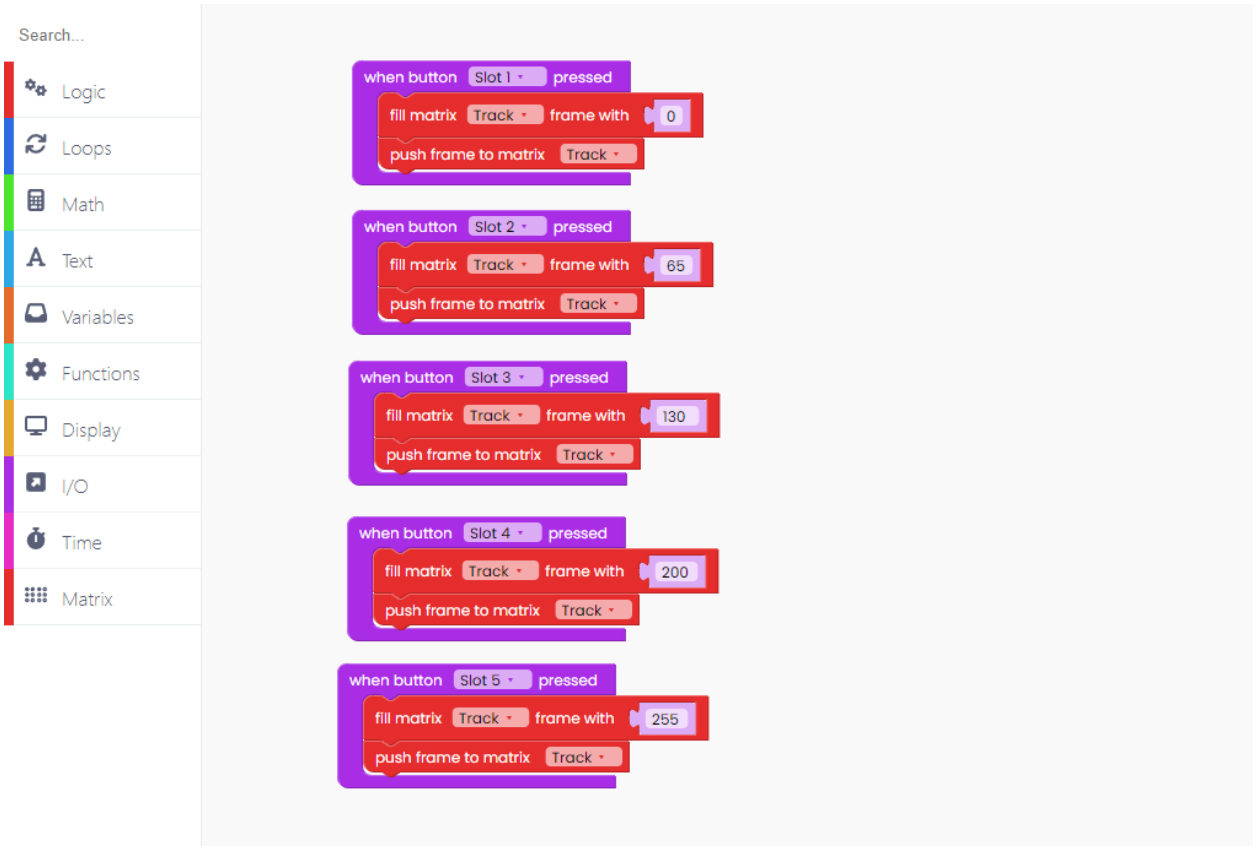
We want to code similar thing for each of the five pushbuttons.

This I/O block can be duplicated four more times to do this.



But, hey, the blocks are all the same now.

Don't worry. We'll change them right away.



As you can see, we coded every pushbutton and experimented with different intensities.

Another thing to remember is that if you create something with inputs (pushbuttons, sliders, encoders, etc.), you must include the "loop forever" block and place the "scan buttons" block inside it.

This ensures that your code is constantly scanned and executed correctly.

Click on the Run button, press the pushbuttons, and check the code.

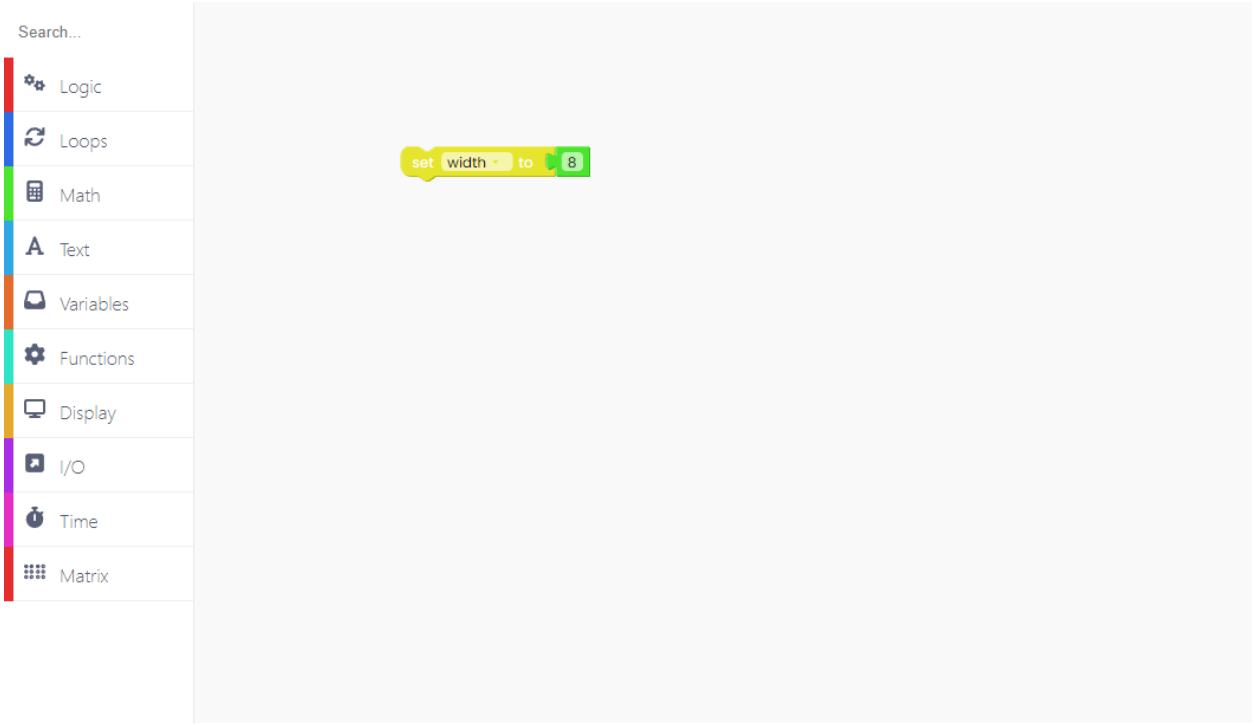
Play with the encoders

In the second guide, we'll experiment with another type of input: encoders.

We'll go through how to code the pre-drawn circle on the Track matrix so that you may adjust its width with the encoders.

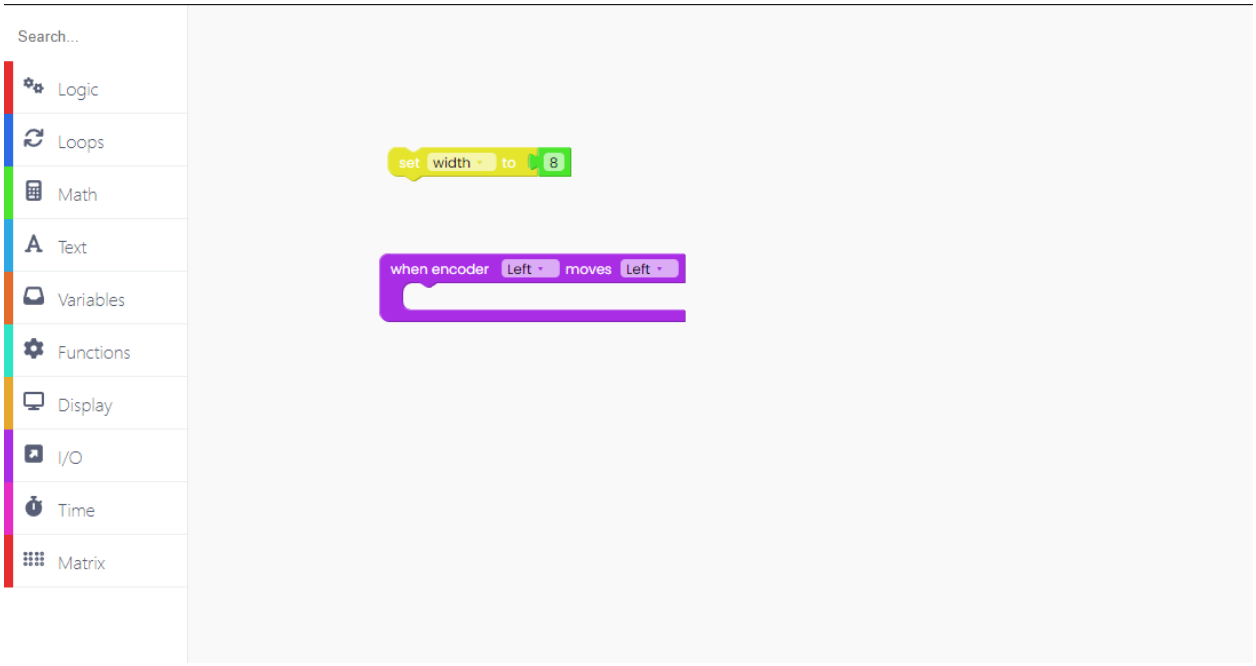
Make a new variable called "width" first.

We'll set the value of width at 8 at the beginning.



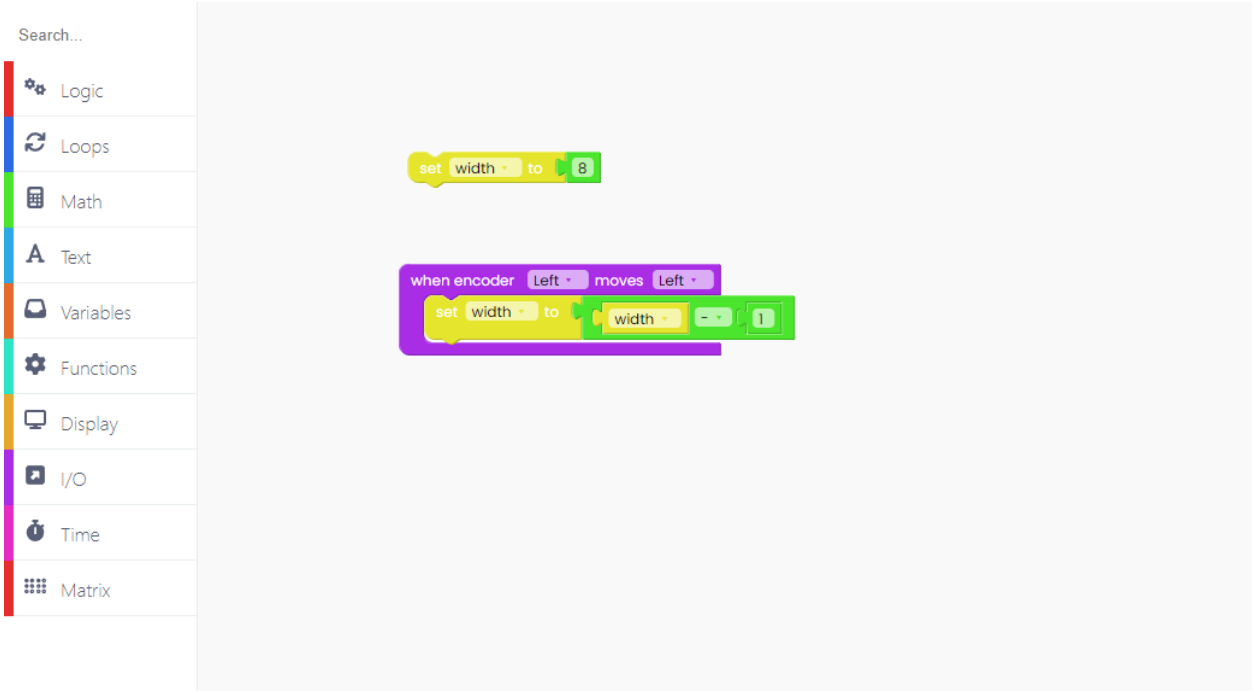
Now we'll write code to see what happens to this variable when we move the left encoder to the right or left.

You'll need I/O blocks, similar to the ones you used in the last example.

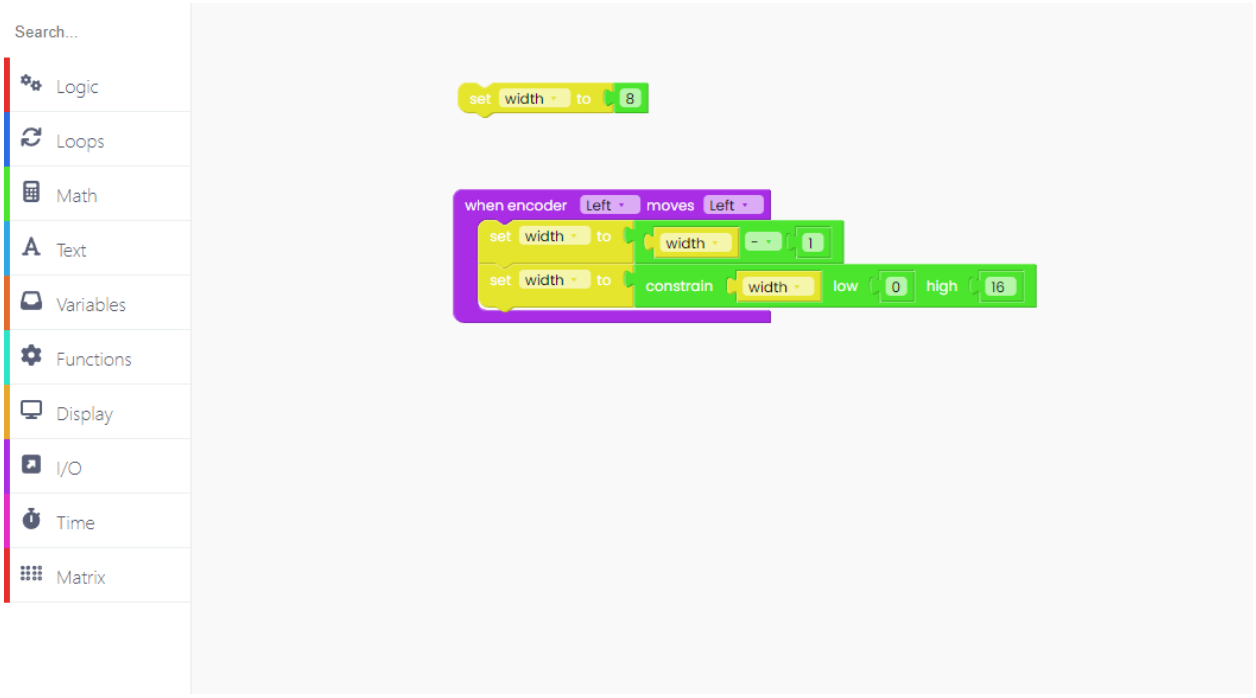


We want the circle to narrow as we move it to the left.

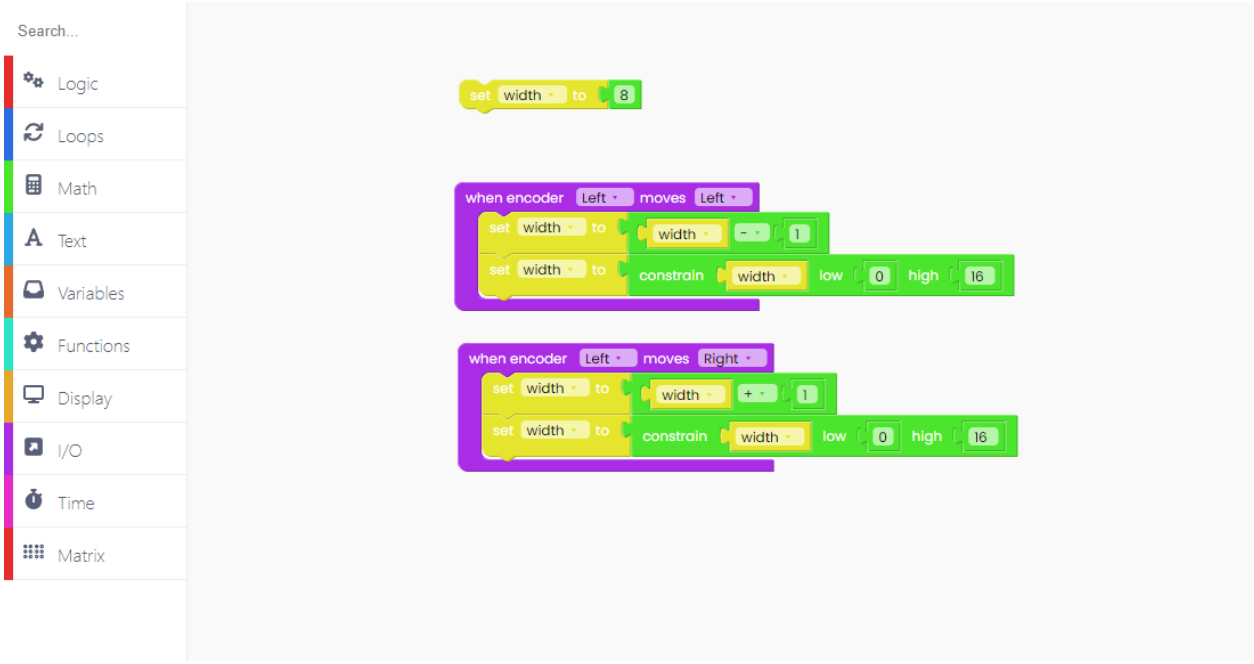
As a result, we'll set the value of the variable "width" to the difference between the variable's value and 1.



However, because that is the maximum, we must limit the number to between 0 and 16.



Let's do this for turning the Left encoder to the right..

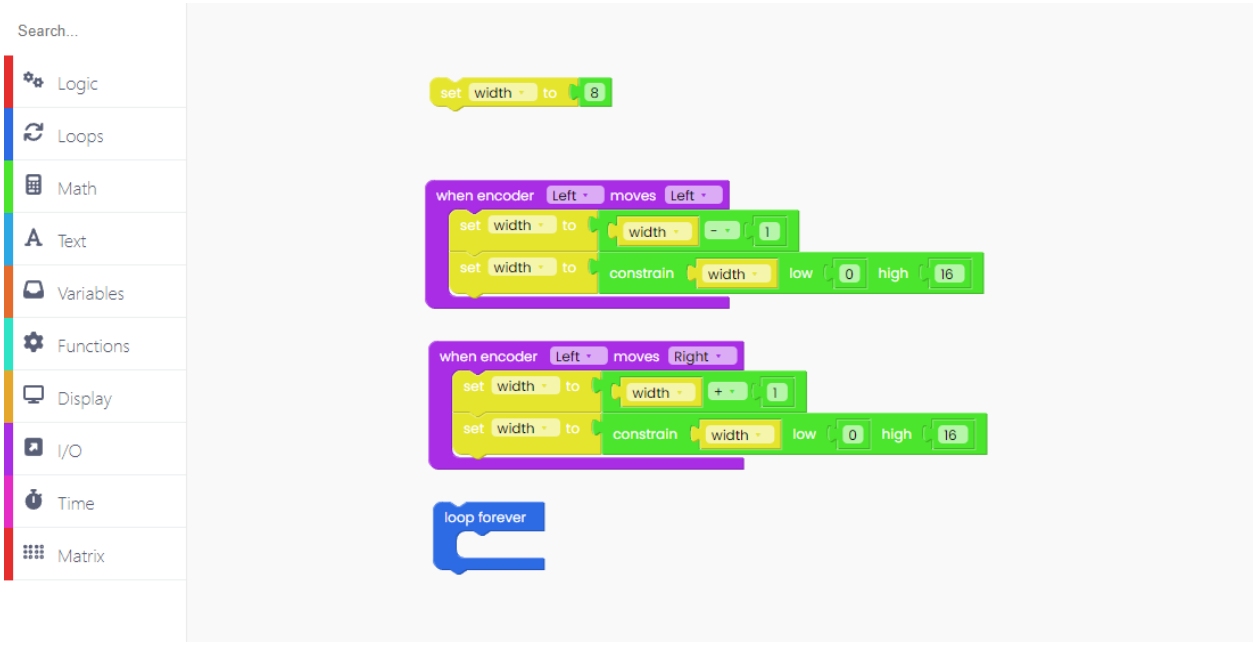


This time, we'll increase the width value by one.

Now is the time to draw what will be happening on the matrix.

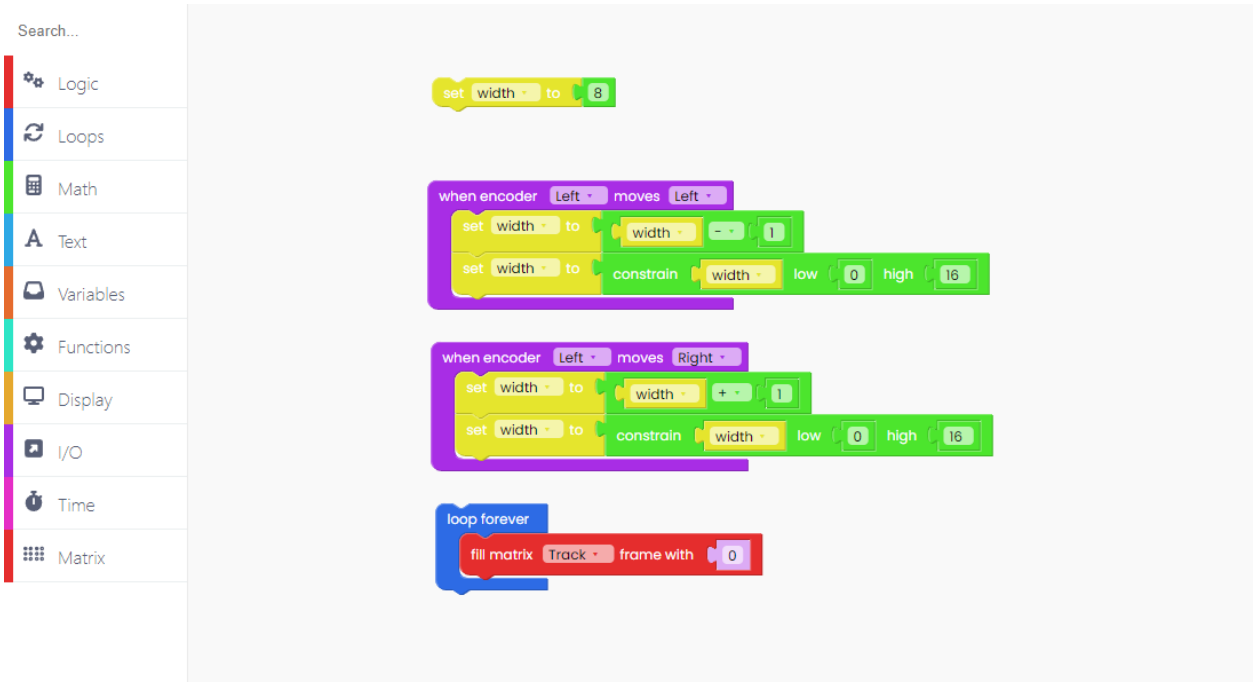
As mentioned before, we want to draw the circle.

Firstly, find the "loop forever" block inside the "Loops" block section.



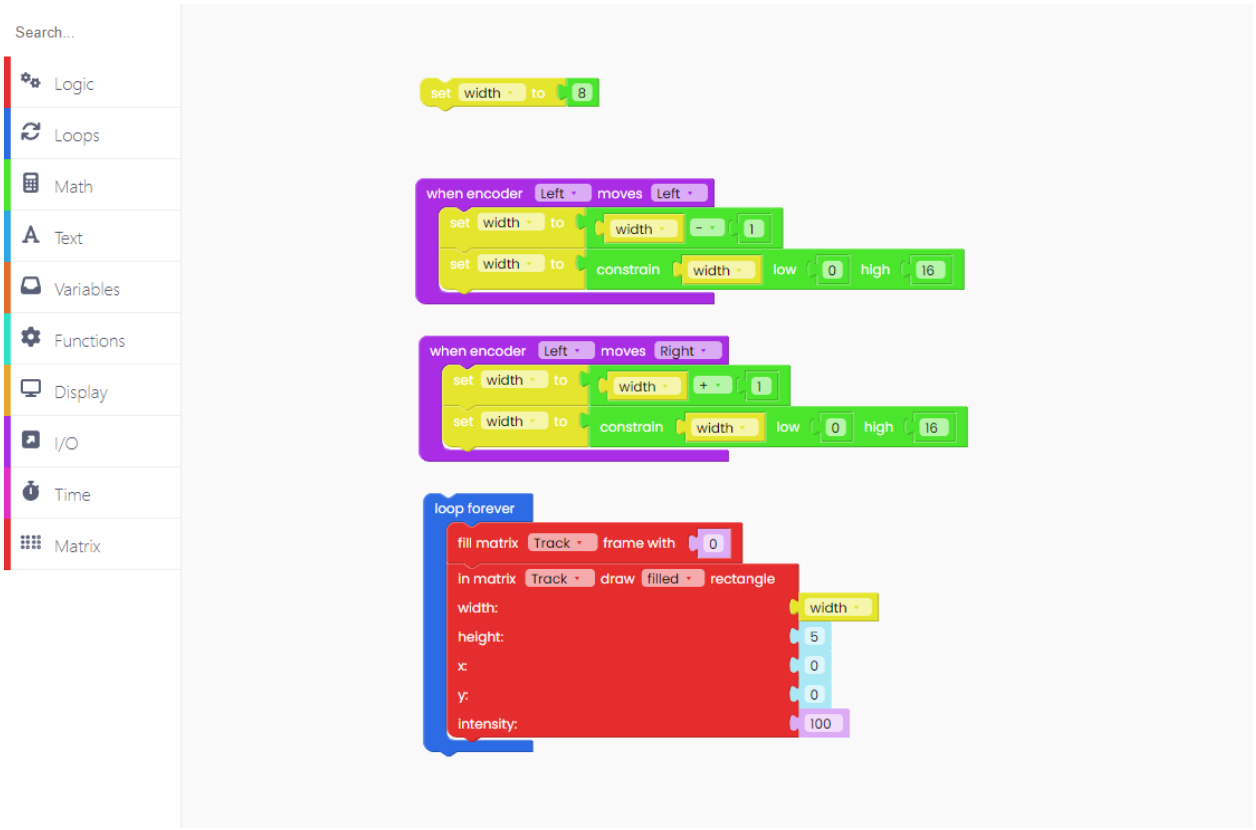
To begin, we must turn off all of the LEDs.

Find the "fill matrix frame with 0" block to do so.



With this block, we reduced the intensity of all LEDs on the Track matrix to 0 - turning them off.

Now, in the "Matrix" block section, locate this large block and place it into the "Loop forever" block.



We use the variable "width" for the width, which means that it will be 8 at first but can change based on how the encoders are turned.

The height is set to 5, the x and y coordinates are set to 0, and the LED intensity is set to 100.

As we saw in the last example, you must include the "push frame to matrix" block every time you draw something on the matrix.

You must also remember the "scan encoders" block because you are working with them.

Start turning the encoders after clicking the Run button.

Let's draw!

Once again, click on the new sketch and choose Synthia.

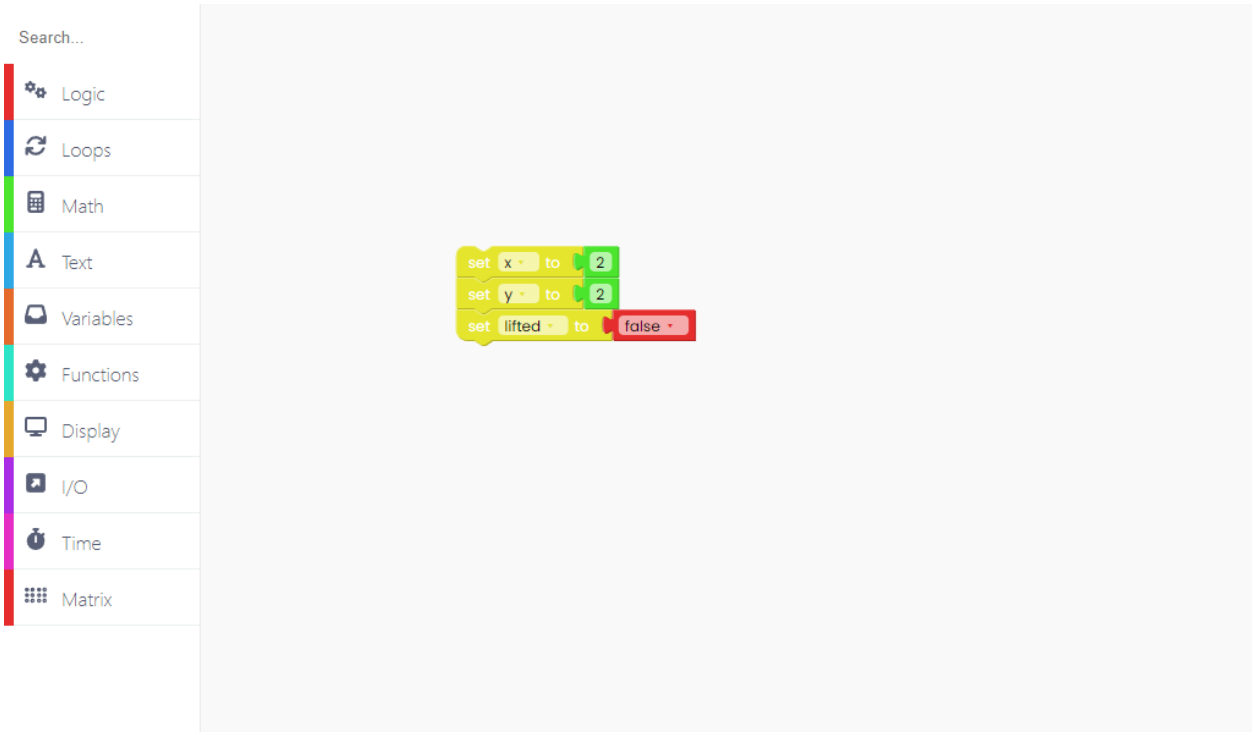
Firstly, we'll make three variables and call them "x", "y", and "lifted".

As you already know, "x" and "y" variables determine the current position of the LED in the matrix.

The "lifted" variable is a new one; with it, you determine if the pen is not lifted and ready to draw. This is the boolean variable, meaning it can be true or false.

Let's set the values for these variables.

For that, we'll use the "set to" block from the Variables block section.



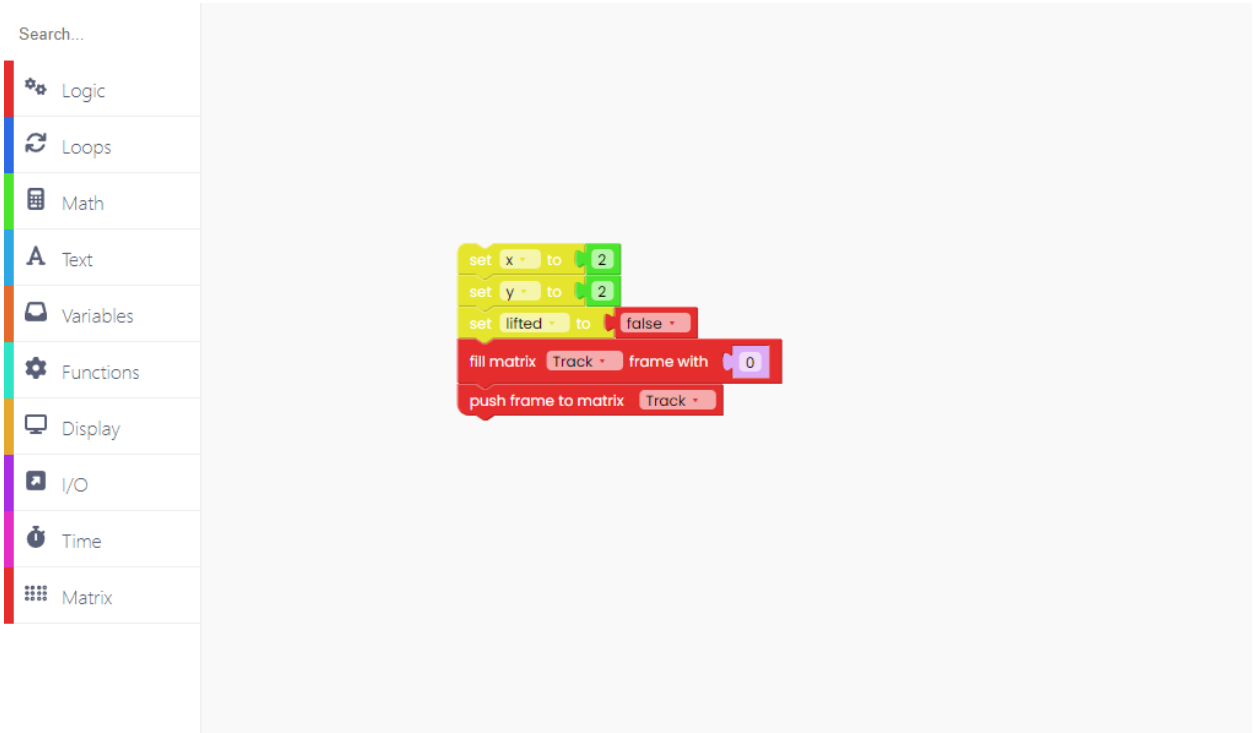
"X" variable will be set on 2, just like the "y" one, and the "lifted" variable will be set to false.

If the "lifted" variable is set on false, that means the pen was not lifted (ready to draw), and if it's set on true, the pen is lifted and it is not ready to draw.

Another important thing is to clear the matrix every time while turning on the device so you can draw on it.

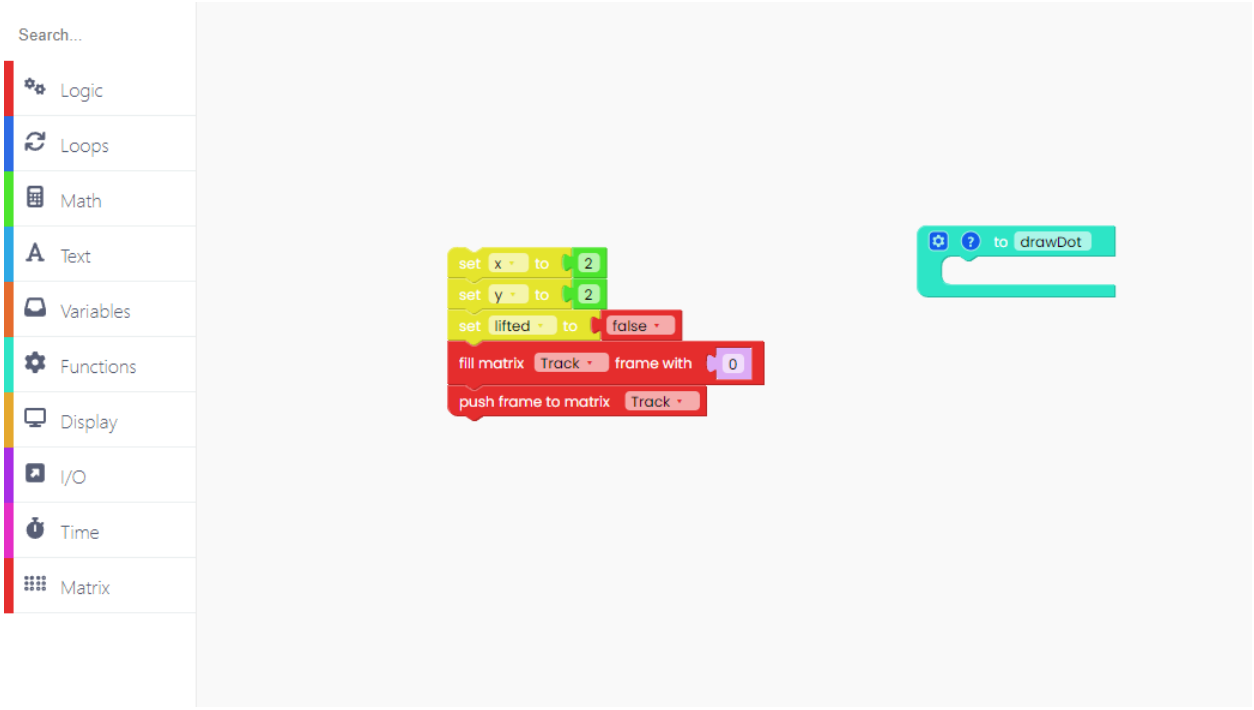
We need the "fill matrix frame with 0" to clear the matrix.

Don't forget to put the "push frame to matrix" below it:

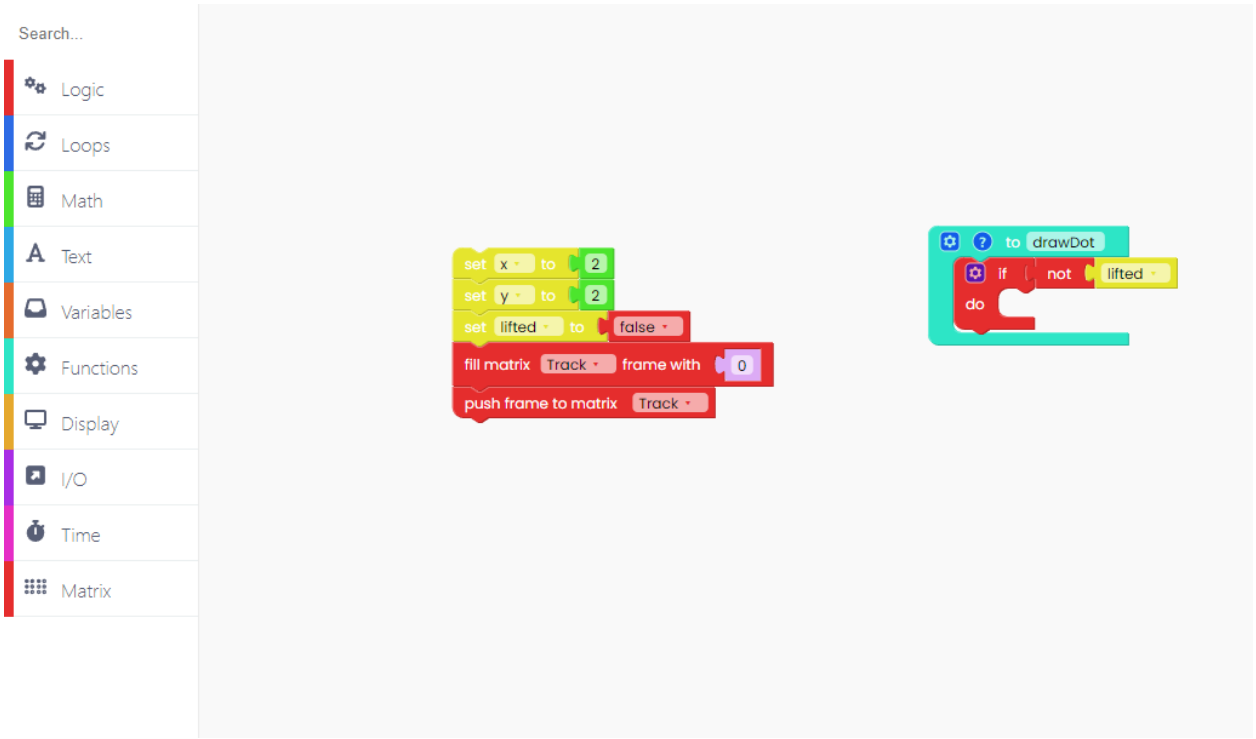


Now that we set the main variables let's create a new function and call it "drawDot".

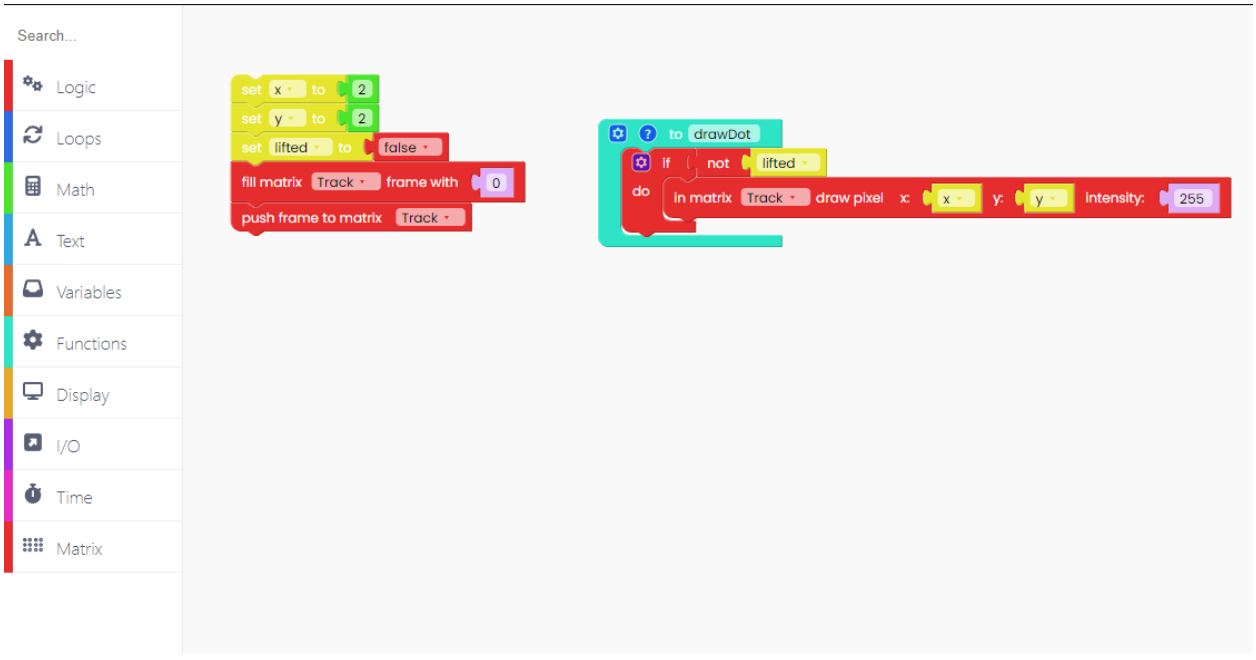
This function draws the current position of the dot on the matrix.



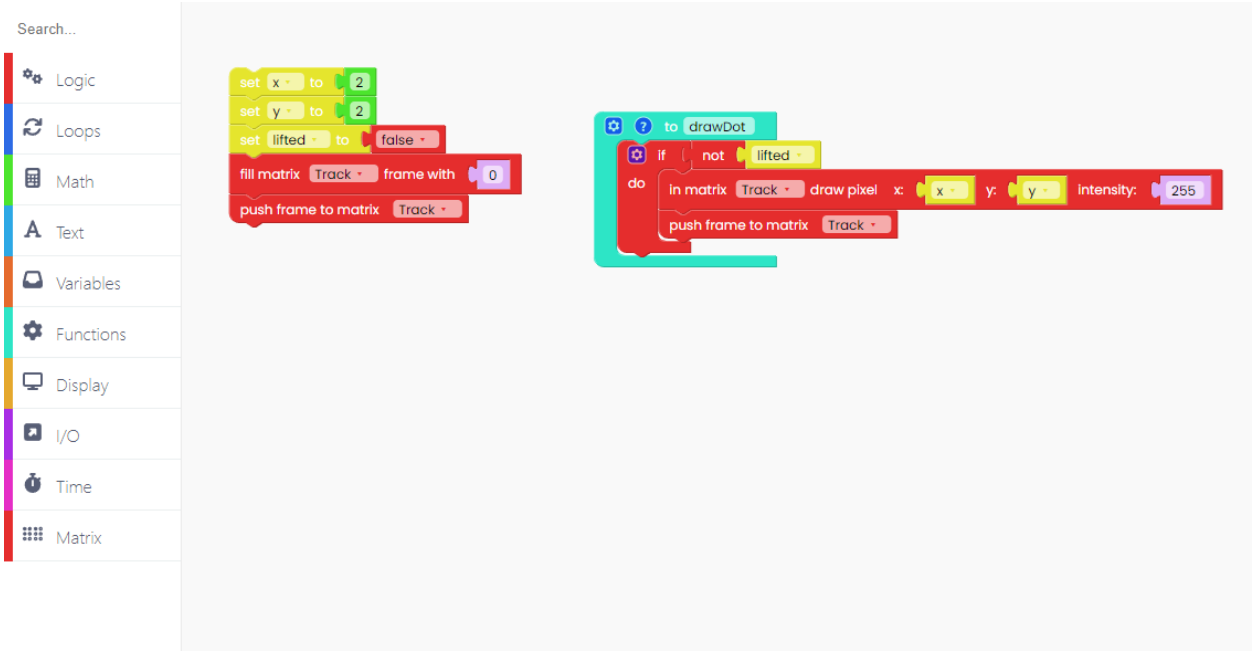
Take the "if-not" block from the Logic section of the blocks, and drop it in the "drawDot" function.



Another thing to do is to set matrix pixel brightness to the maximum - 255!

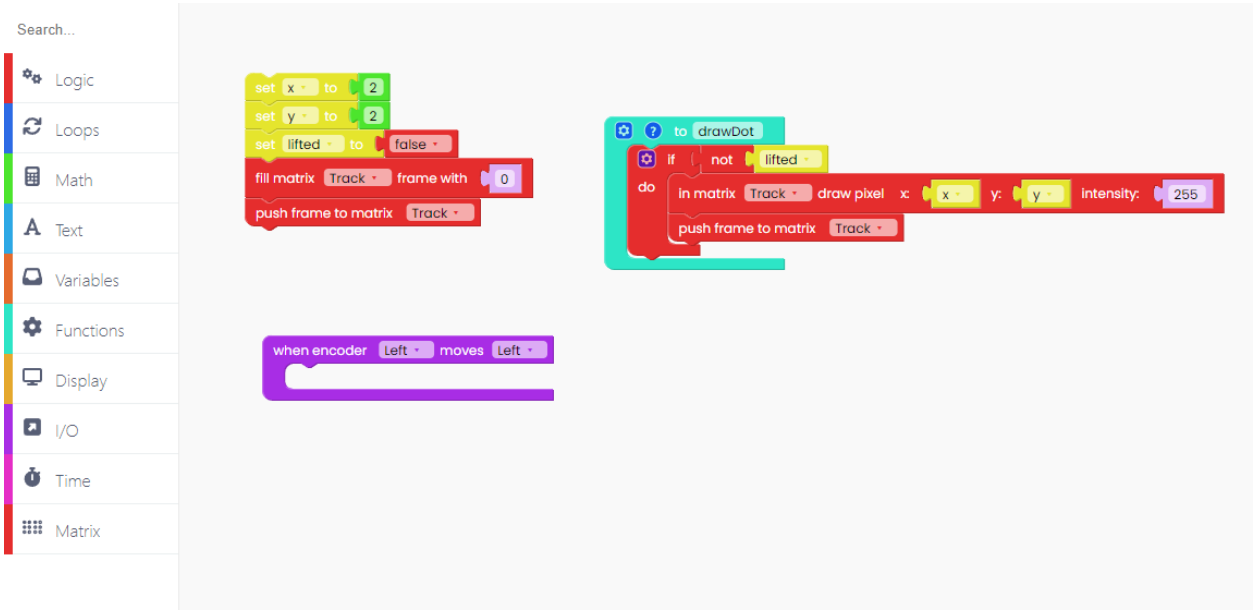


In the end, drag, and drop the "push frame to matrix" block to ensure your code will execute properly.



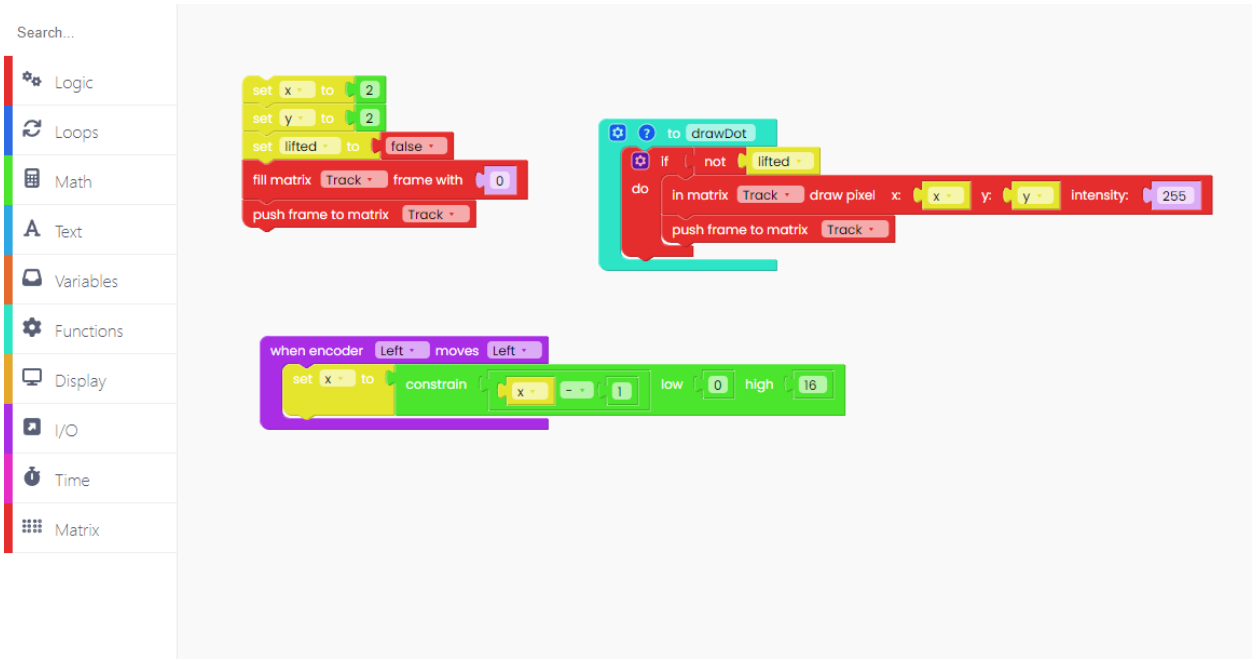
You'll use encoders for drawing on the matrix. Because of that, now is the time to use those purple blocks from the I/O block section.

With the left encoder (x variable), we'll change the horizontal position of the dot on the matrix. With the right encoder (y variable), we'll change the vertical position of the dot.



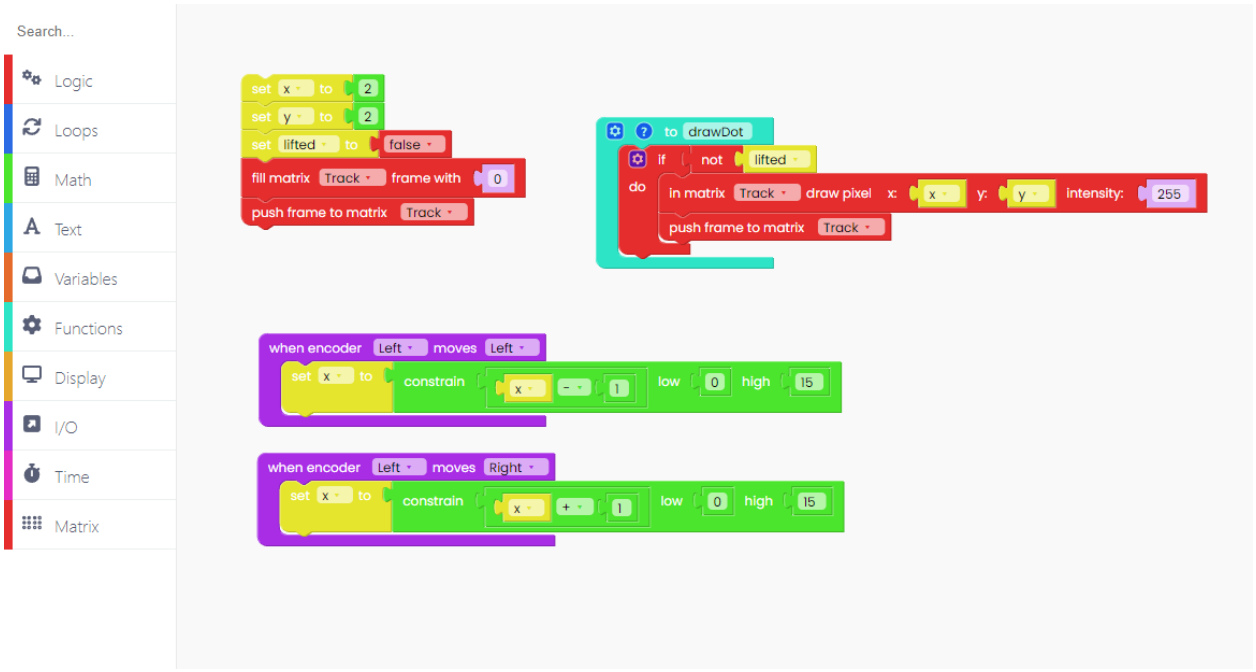
Remember the previous sketch?

We'll do the same steps here.

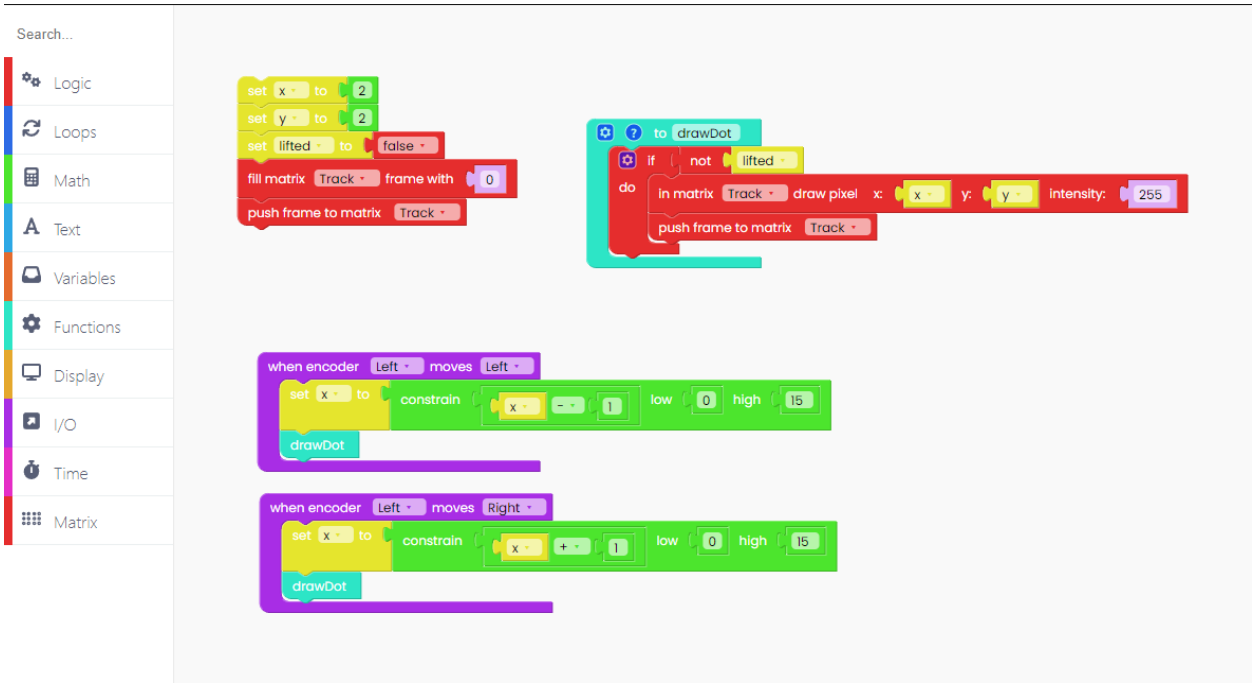


So, we'll make sure to add the amount variable (-1 if we rotate the encoder to the left, or 1 if we rotate the encoder to the right) to "x" or "y", and to limit this from 0 to 15.

Let's duplicate this block for when the left encoder is turned to the right!

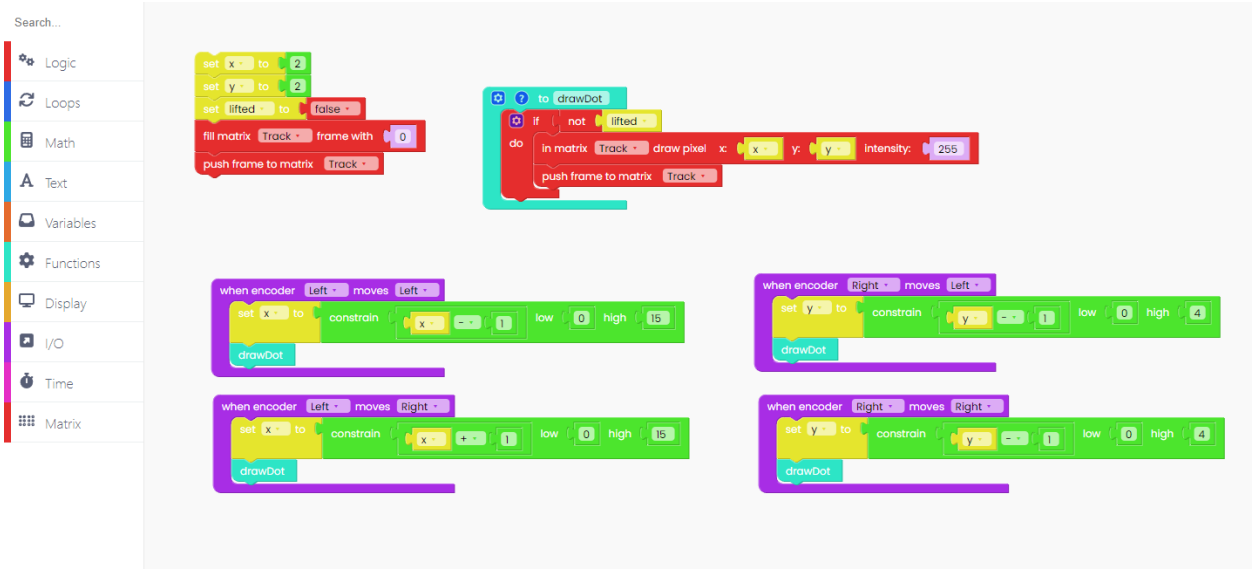


Add the "drawDot" function at the end of both I/O blocks to ensure that the dot moves in the matrix.



Duplicate the blocks and code what happens

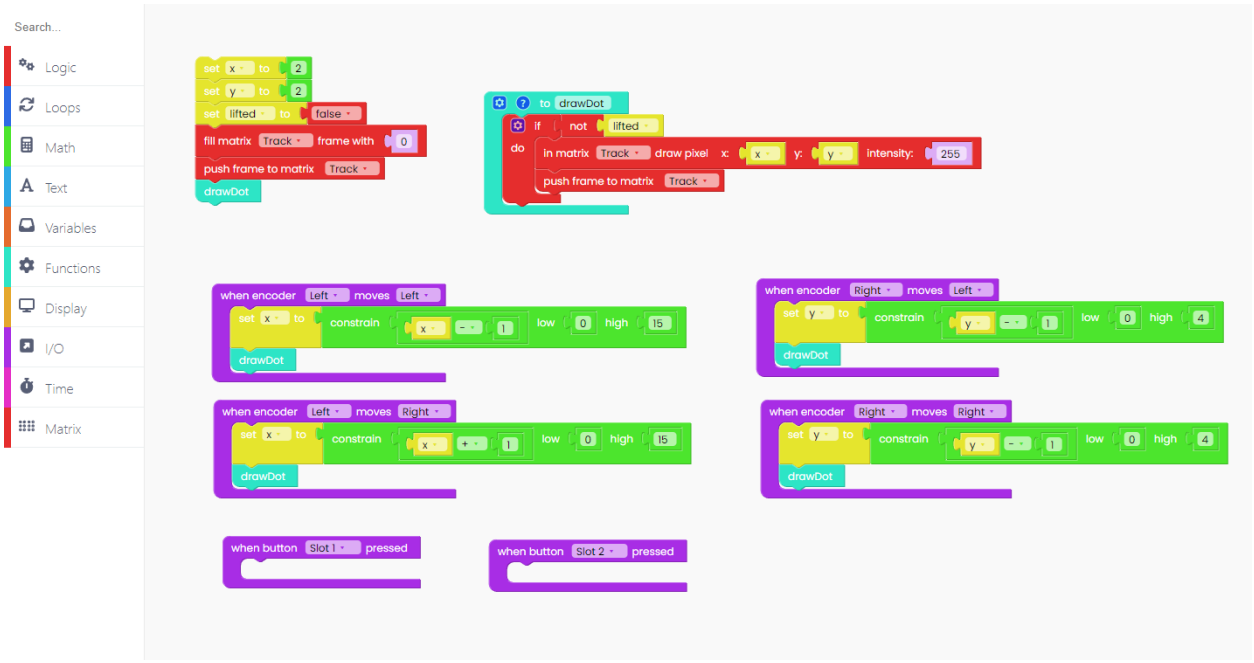
Also, keep in mind that you're changing the matrix's width with the left encoder, so the limit is from 0 to 15. On the other hand, you're changing the height of the matrix with the right encoder, so the limit is set from 0 to 4.



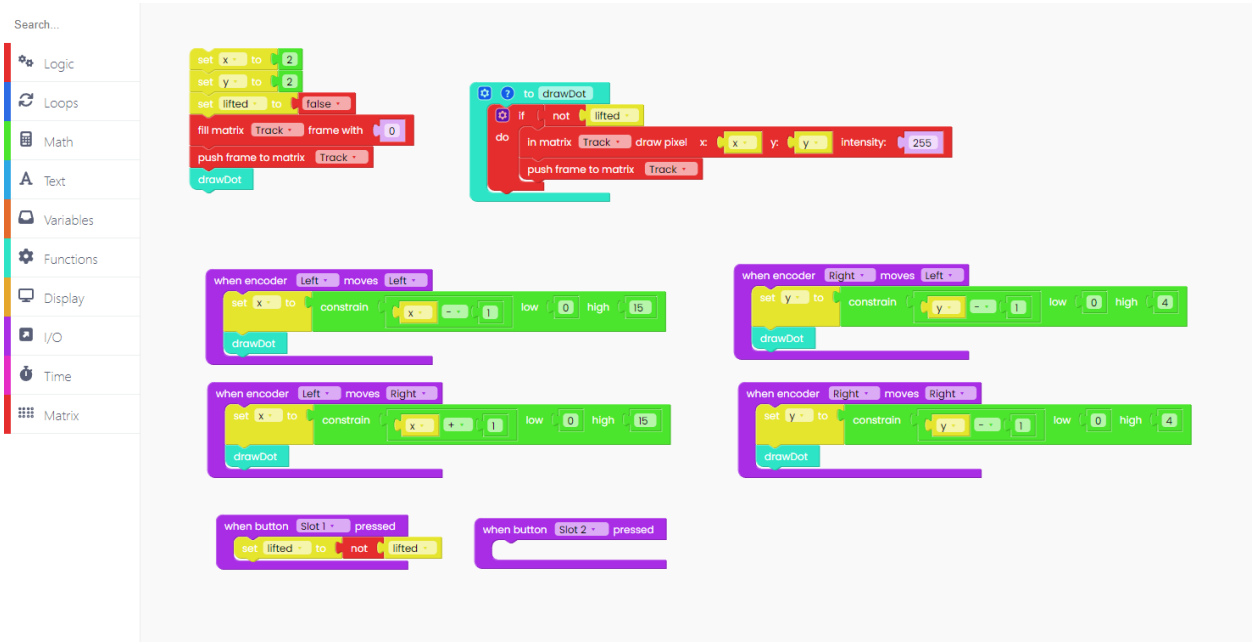
The last thing we will do is set what will happen if the pushbuttons get pressed.

We'll do this for the pushbuttons number 1 and 2.

For that, you'll need these blocks from the I/O block section:

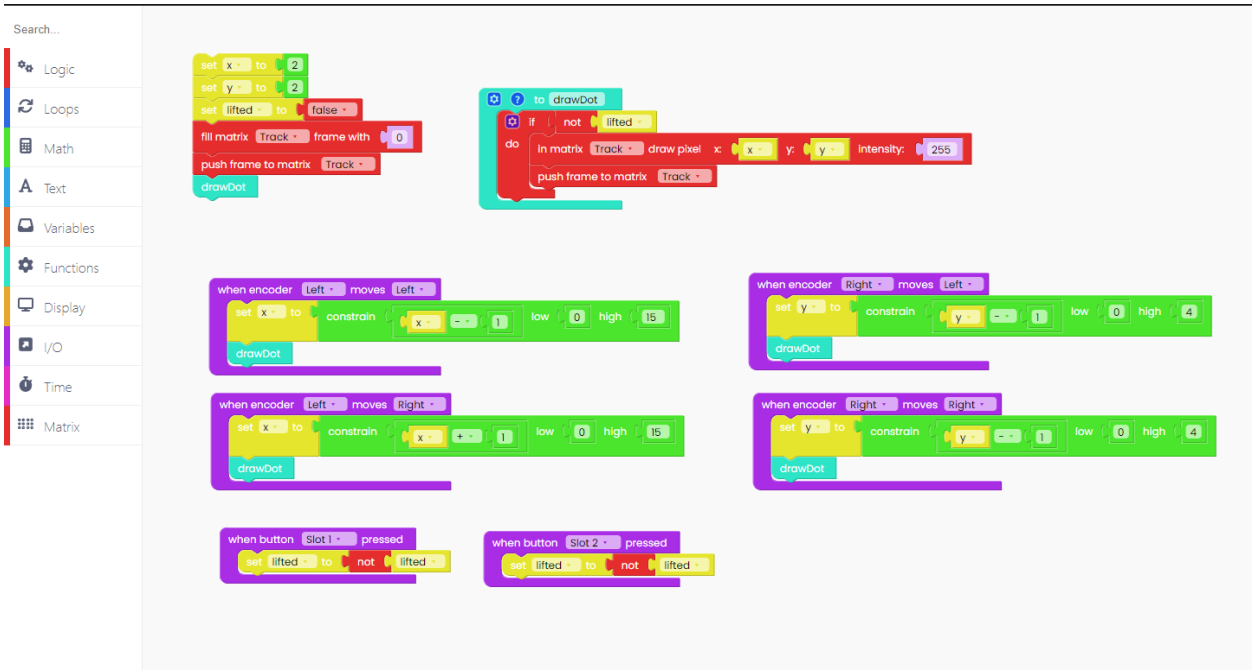


While pressing the pushbuttons, we want to change the "lifted" variable.



So, as the blocks say - if you press the pushbutton no.1, the variable will go from lifted to not lifted.

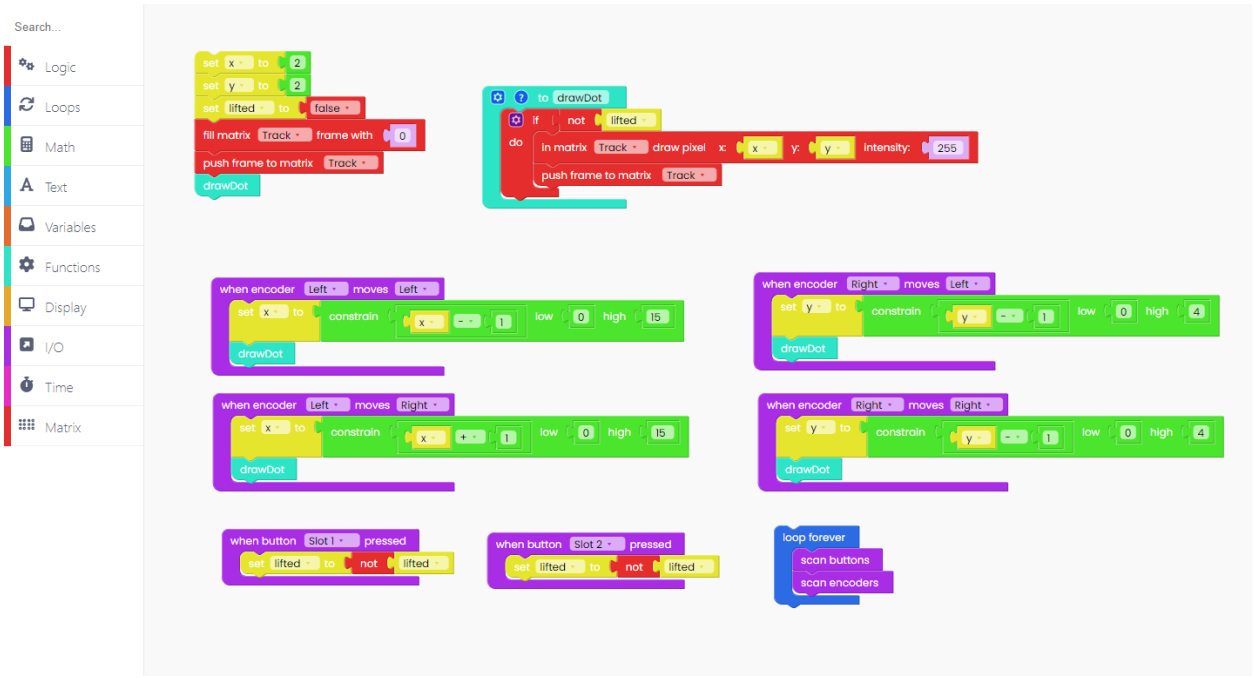
Let's duplicate the blocks for pushbutton no.2.



As you can see, pressing pushbutton no.2 will do the same.

So, if the variable was lifted, it would change to not lifted. Of course, this goes both ways, so if the variable is not lifted, pressing on the encoders will change it into lifted.

The only thing left to do is to add the "Loop forever" block and add the "scan buttons" and "scan encoders" blocks inside it for properly executing your code.



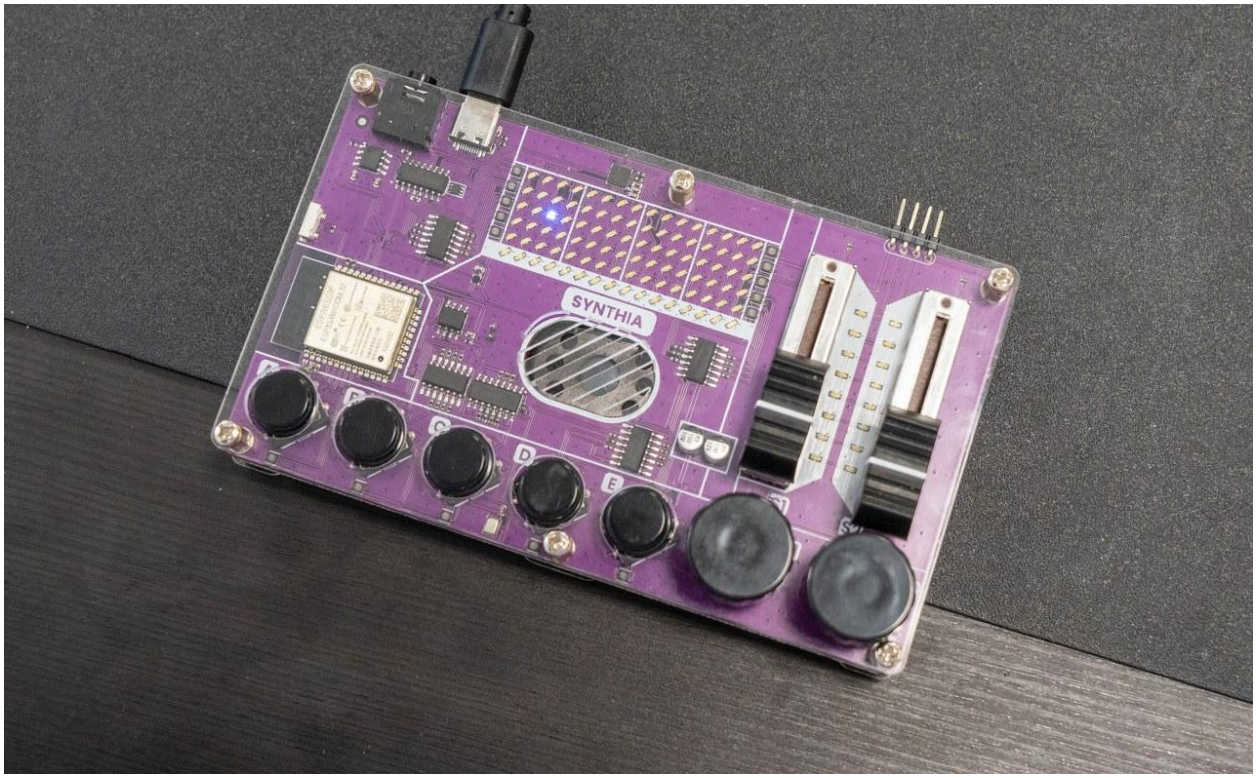
Congratulations!

You successfully finished all the sketches and are on the way to becoming a real programmer!

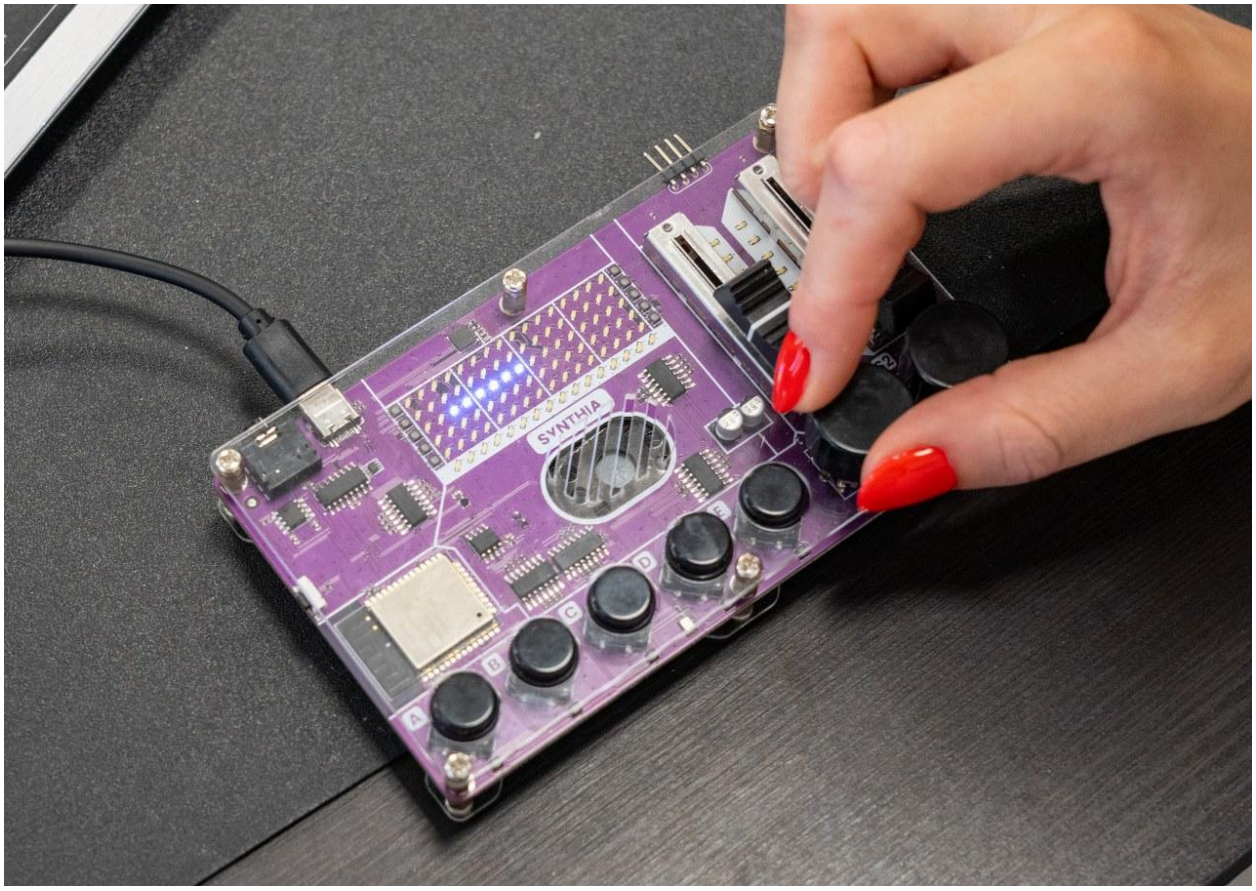
We are almost done with this guide, but make sure to check the next chapter because we'll show you how to delete all the changes you have made and how to restart your device.

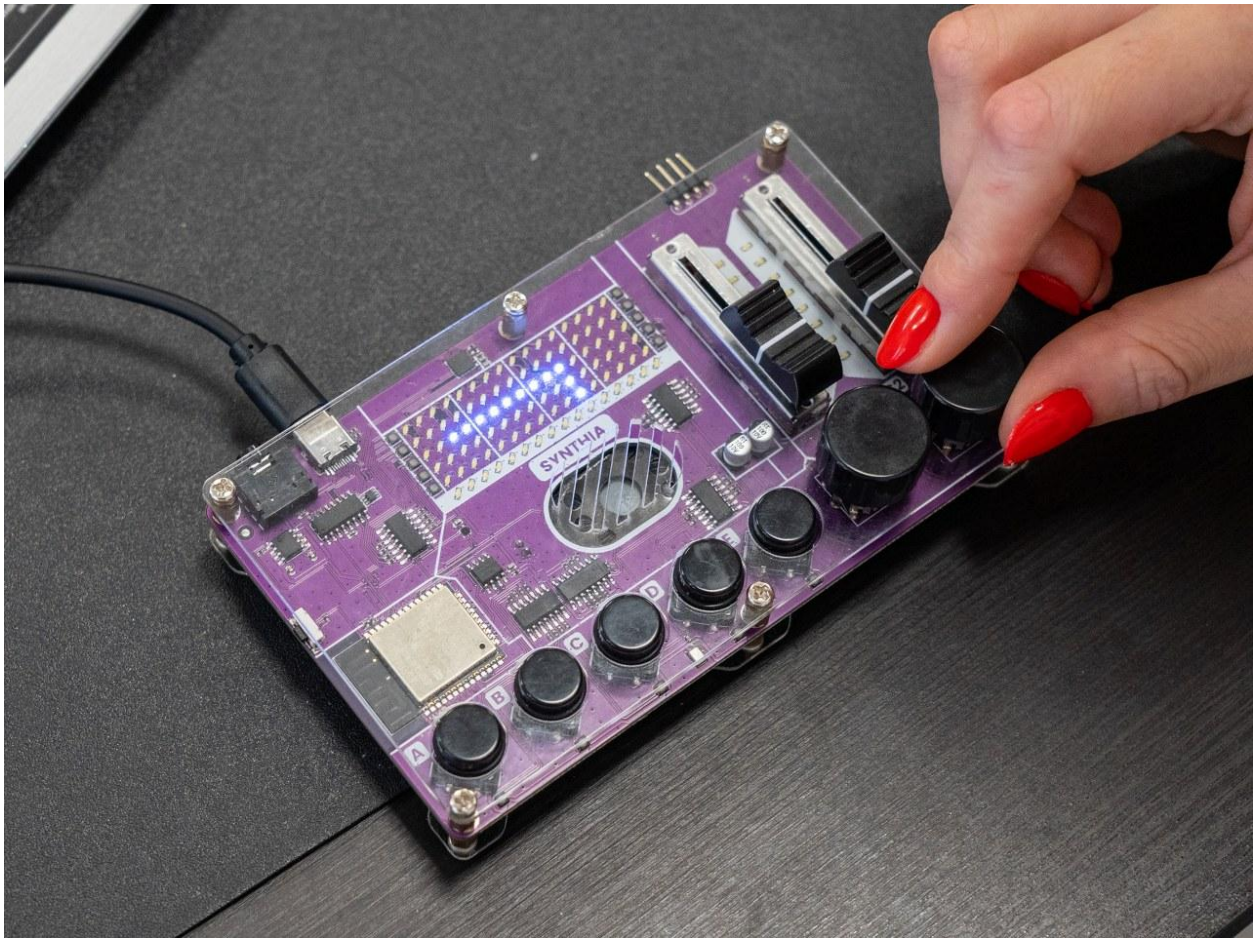
Once you run the code and the device turns on, you'll see this on Synthia.

So, a little dot (LED on the LED matrix) that is lightning up is the one we set at the beginning of the sketch. That is your starting point while drawing on the matrix.



If you rotate the left encoder, you'll be able to draw horizontally, and if you rotate the right one, you'll be able to draw vertically.


SYNTHIA
A
B
C
D



However, if you click on any of the two pushbuttons we coded, the pen will stop working, and if you click on it once again, it will start working.

Slide it down!

The last example we prepared for you will be about the sliders on your Synthia.

The purpose of this code is to have the left slider turn on and off the LEDs in the Track matrix and the right slider turn on and off the LEDs in the Sliders matrix.

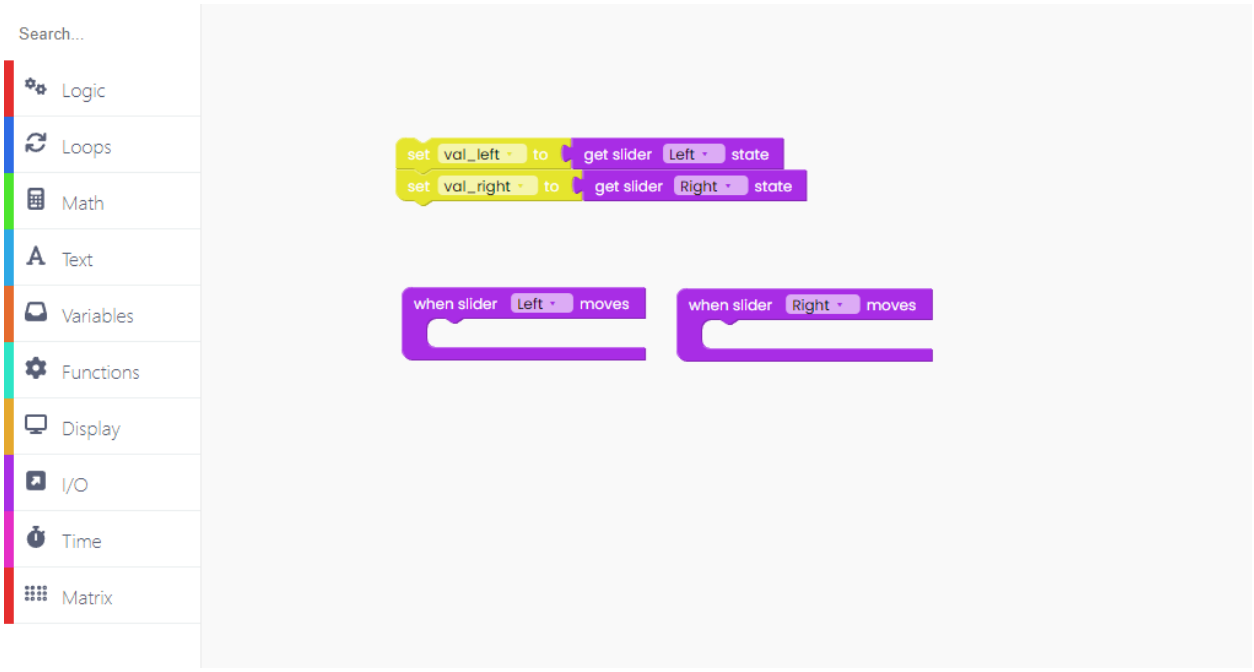
To do so, we must first set the values of the left and right sliders.

Let's create two variables and call them "val_left" and "val_right".

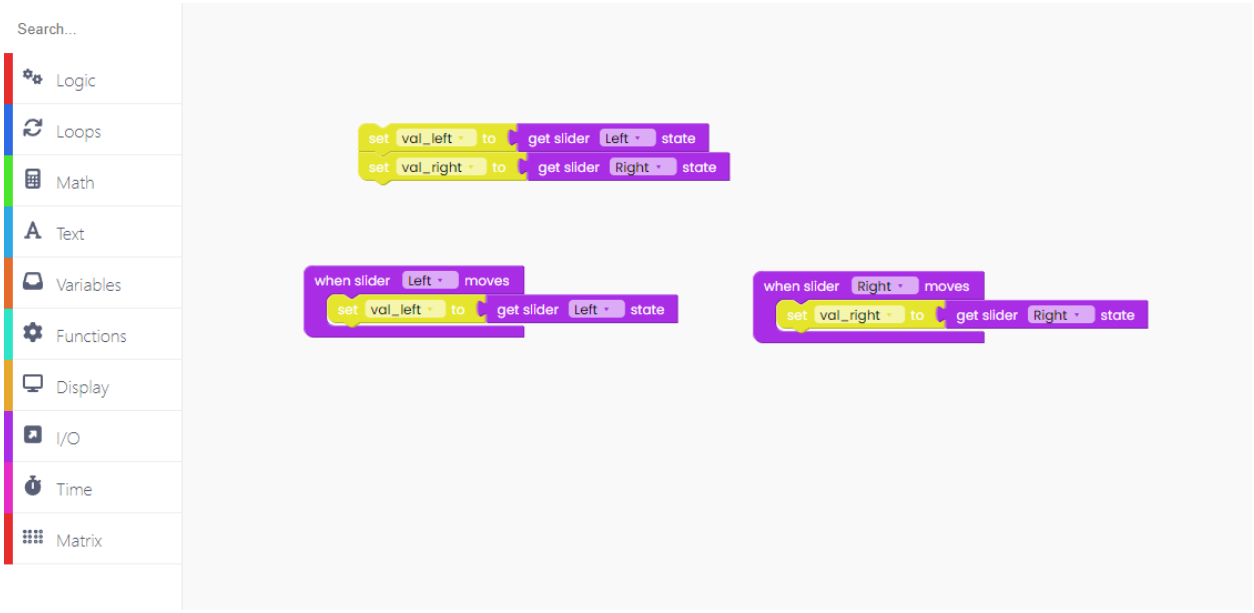
The value will be determined by the position of your sliders at the time.

Now, let's see what happens when sliders move.

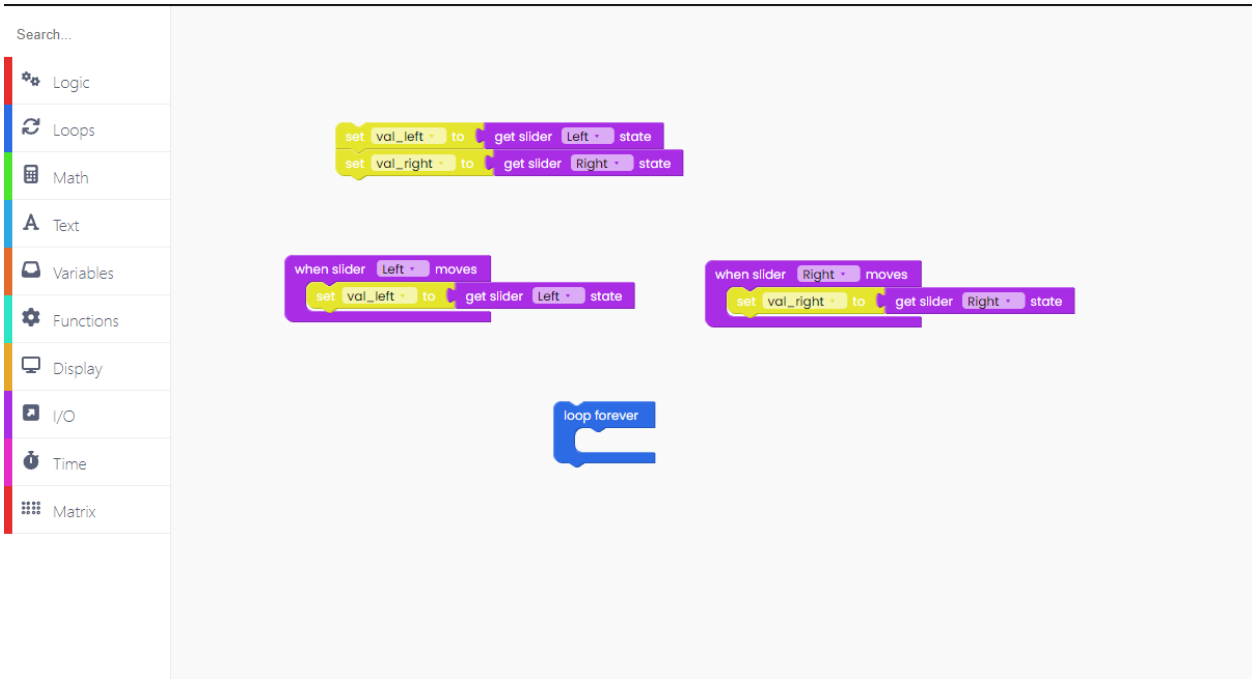
You'll need I/O blocks for that.



When we move the slider, its value changes to reflect its current position.



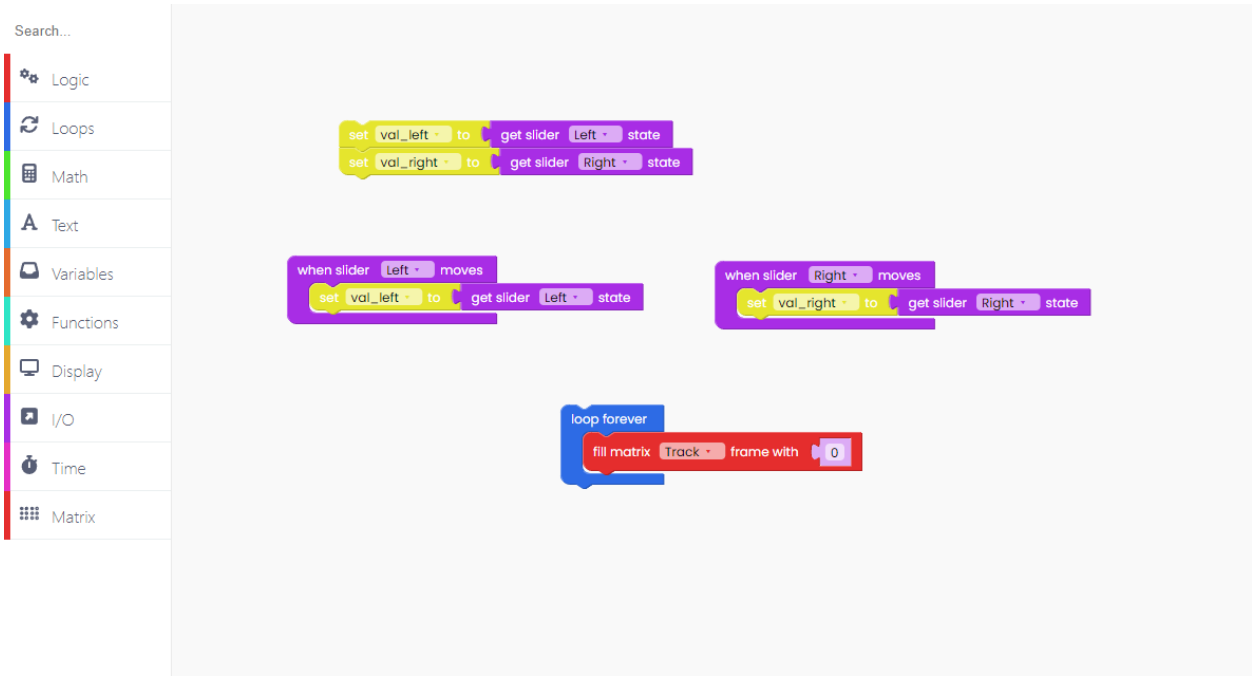
Let's code the part that will be happening all the time, in loops.



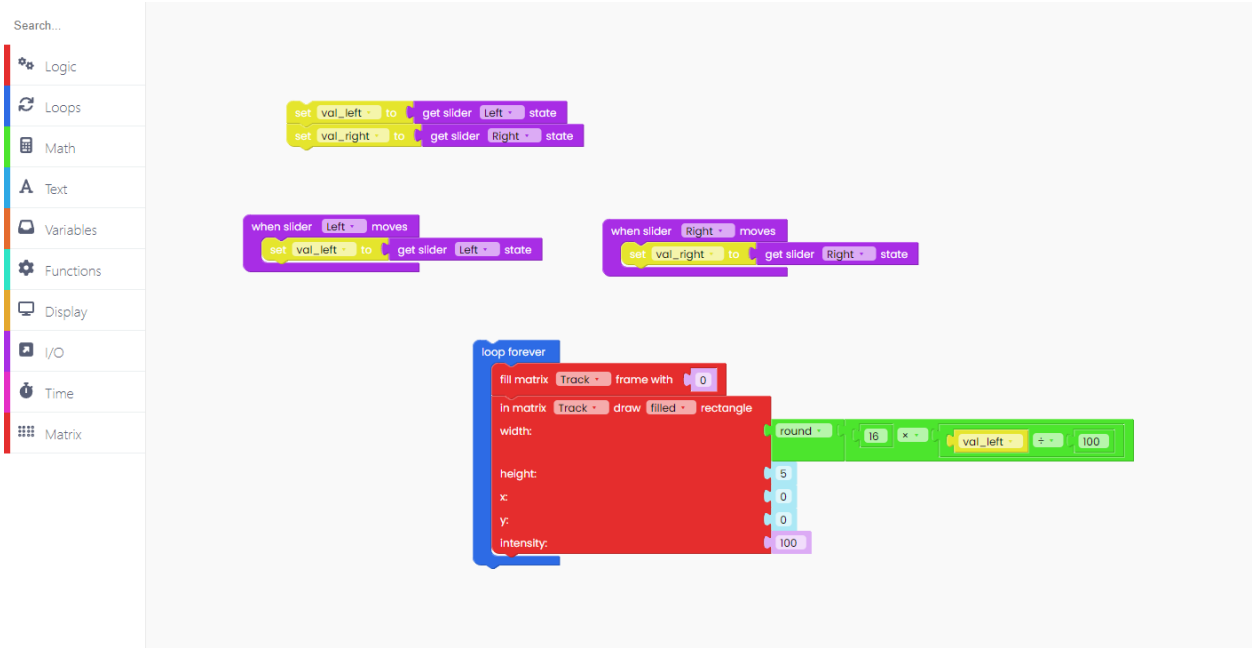
As always, the initial step is to clear one matrix.

For that, we'll need the "fill matrix frame with 0".

The first matrix we'll work on is the Track matrix.



Now, let's draw a filled rectangle in the Track matrix.

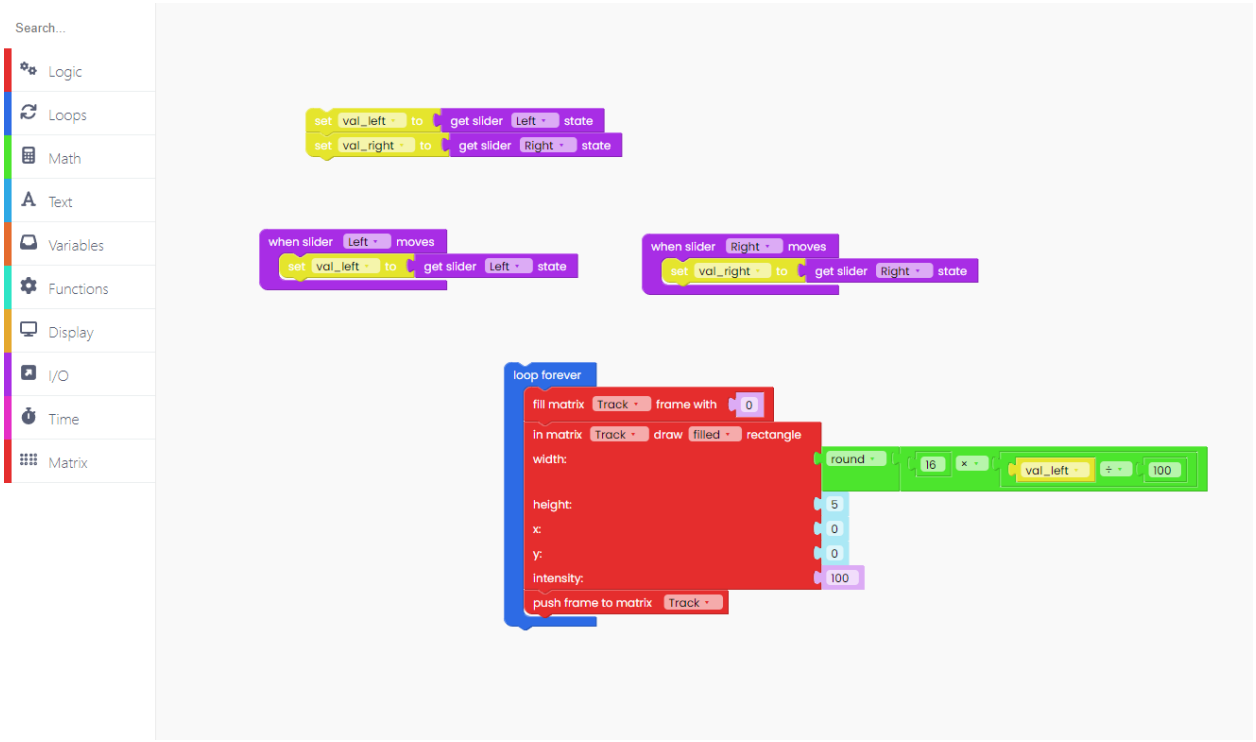


The LED intensity is set to 100, and the height is set to 5 (maximum).

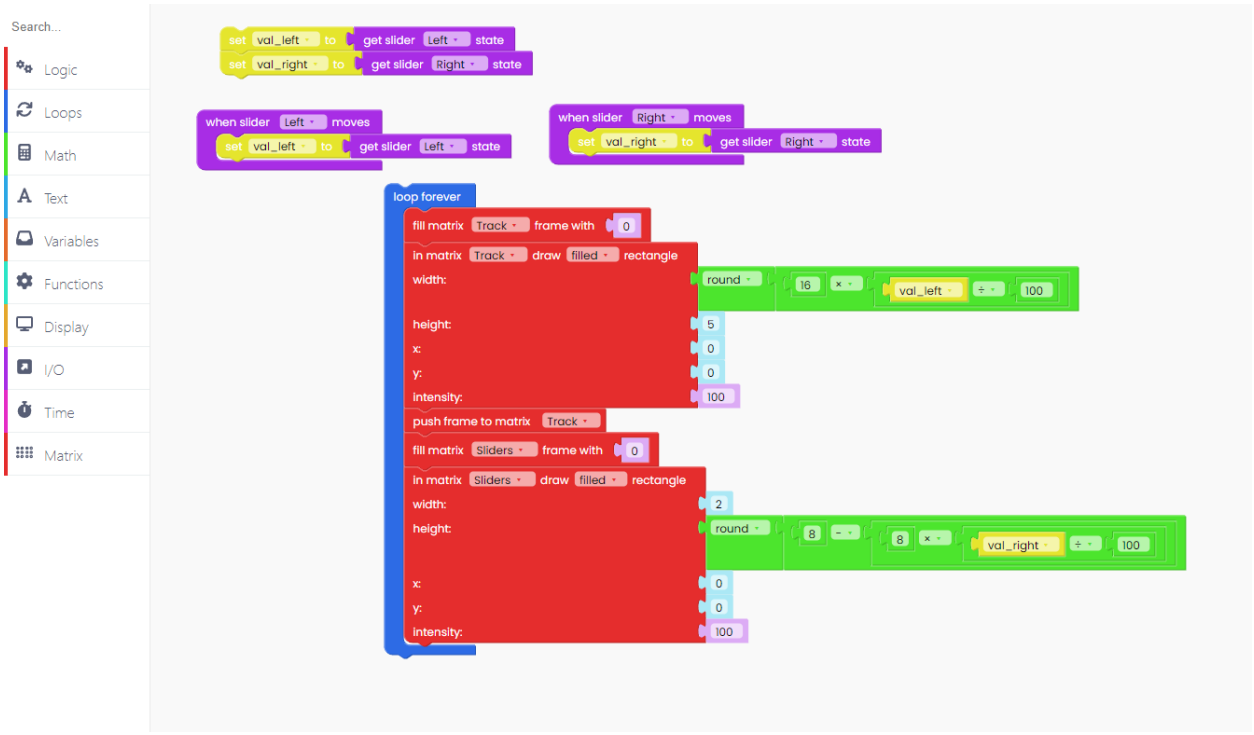
The width is set to the relative number equal to this math operation.

The maximum width of the matrix is 16, which we must divide by 100 because the sliders have a range of 0 to 100.

To ensure that this rectangle appears, add the "push frame to matrix" block.



Now, let's do the same for the Sliders matrix.



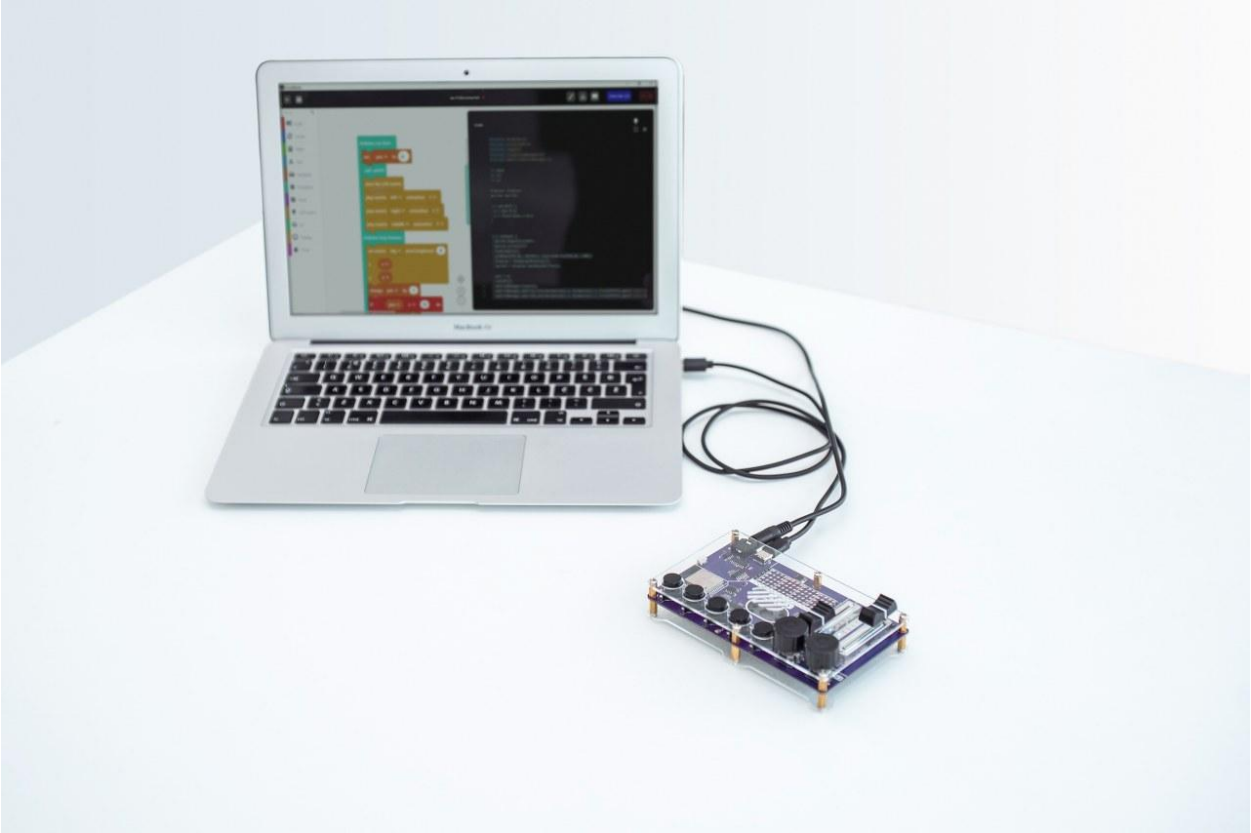
We adjusted the height and width this time.

Remember to include the "push frame to matrix" and "scan sliders" blocks at the end.

Click on the Run button and check it out.

What's next?

You've reached the end of our first Synthia coding tutorial; congratulations!



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

In the meantime, continue exploring on your own and show us what you've done with your Synthia by sharing it on the CircuitMess community forum or via our Discord channel.

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

Thank you, and keep making!