

Jay-D coding – first steps

Introduction

Installation

Welcome to the Jay-D coding tutorial

Thanks for supporting our work, and we hope you'll learn so much more by following this guide.

We'll use **CircuitBlocks** for coding your newly assembled mix table.

So, what's CircuitBlocks?

CircuitBlocks is a custom-made coding app that we've designed.

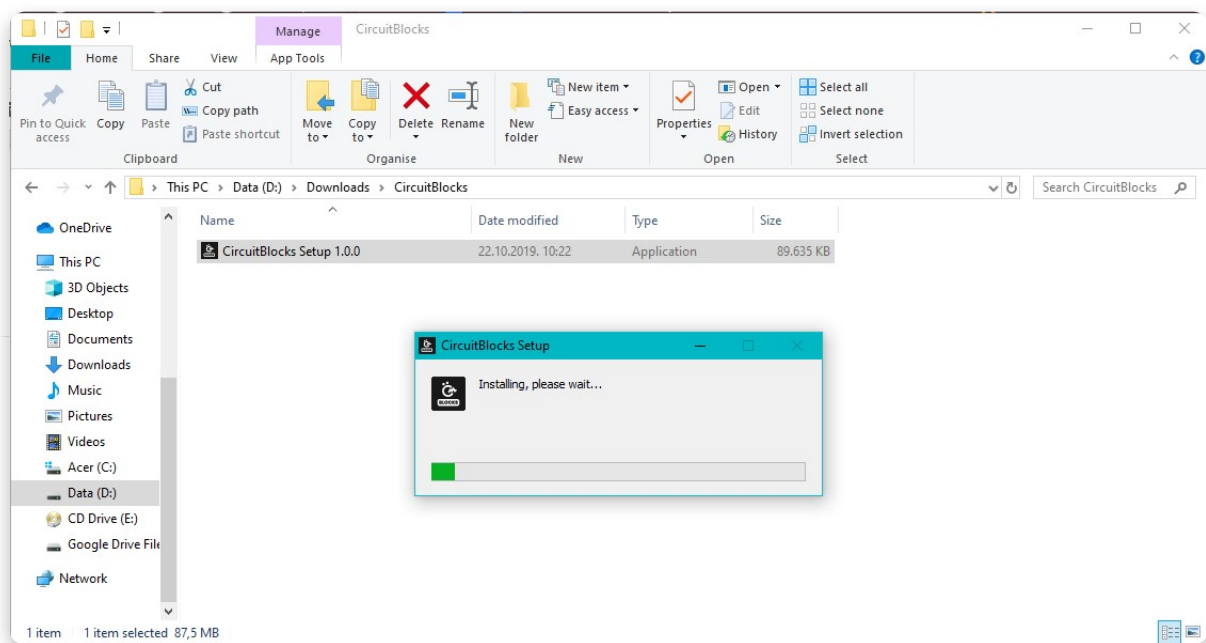
You will code your Jay-D in CircuitBlocks' graphical block-based coding interface that will help you make your first steps in the world of physical computing.

Installation

CircuitBlocks currently runs on Windows, Linux, and Mac OS computers.

If you have a Windows computer

1. Go to the [CircuitBlocks download page](#)
2. **Download the latest version for Windows** – Check if you have a 32 or 64 version. Go to Settings on your PC, click on the System option and find the About section where you'll see the system type.
3. Double-click the downloaded file named "CircuitBlocks"
4. CircuitBlocks will automatically install and create a new desktop shortcut



Your PC is not at risk!



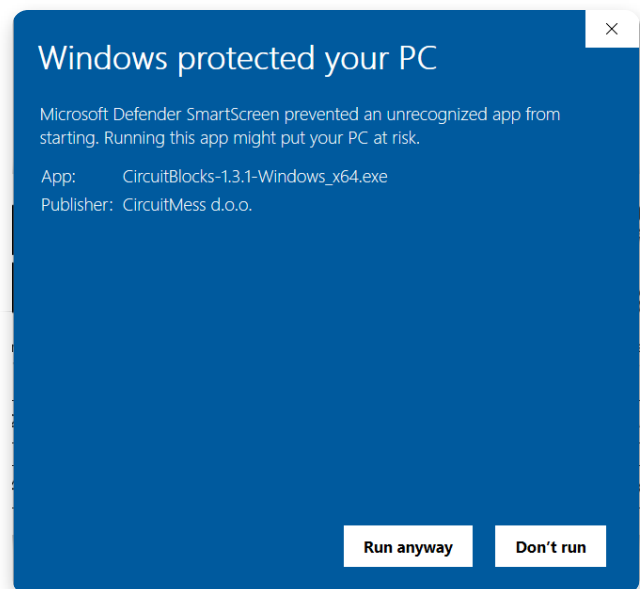
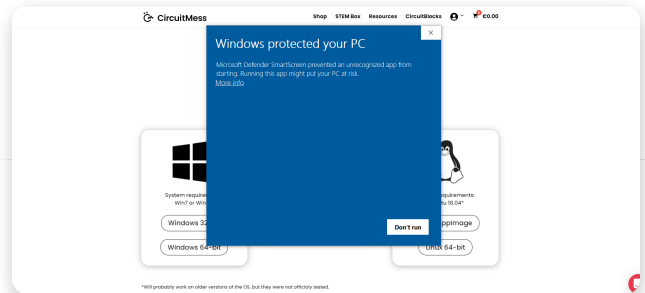
There is a possibility that a notification that says your PC is at risk may pop up when you try to install CircuitBlocks. Don't worry, this happens regardless of CircuitBlocks being safe to run. See the instructions below on how to handle this notification.

This is the message you might get when trying to install CircuitBlock on your PC. Windows reports a threat despite the program being safe to download and run.

Please proceed with the installation by clicking on the 'More info' option.

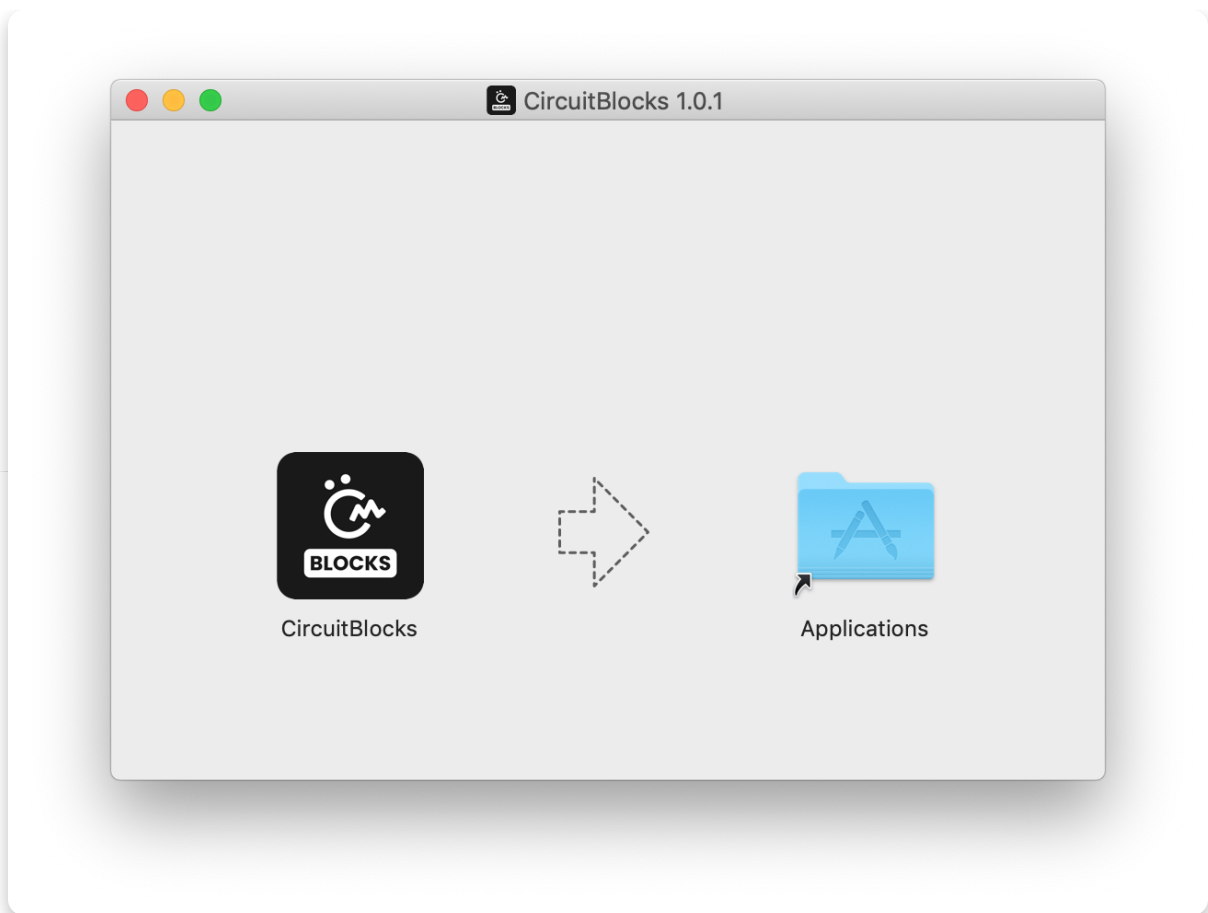
After you click on the 'More info' option, an option to 'Run anyway' should appear at the bottom of the window.

Proceed by clicking on 'Run anyway'.



If you have a Mac computer

1. Go to the [CircuitBlocks download page](#)
2. **Download the latest version of CircuitBlocks** for Mac OS (the file named "CircuitBlocks-1.0.1-Mac.dmg" or similarly should be downloaded)
3. Move the files to the 'Applications' folder
4. CircuitBlocks will be installed automatically



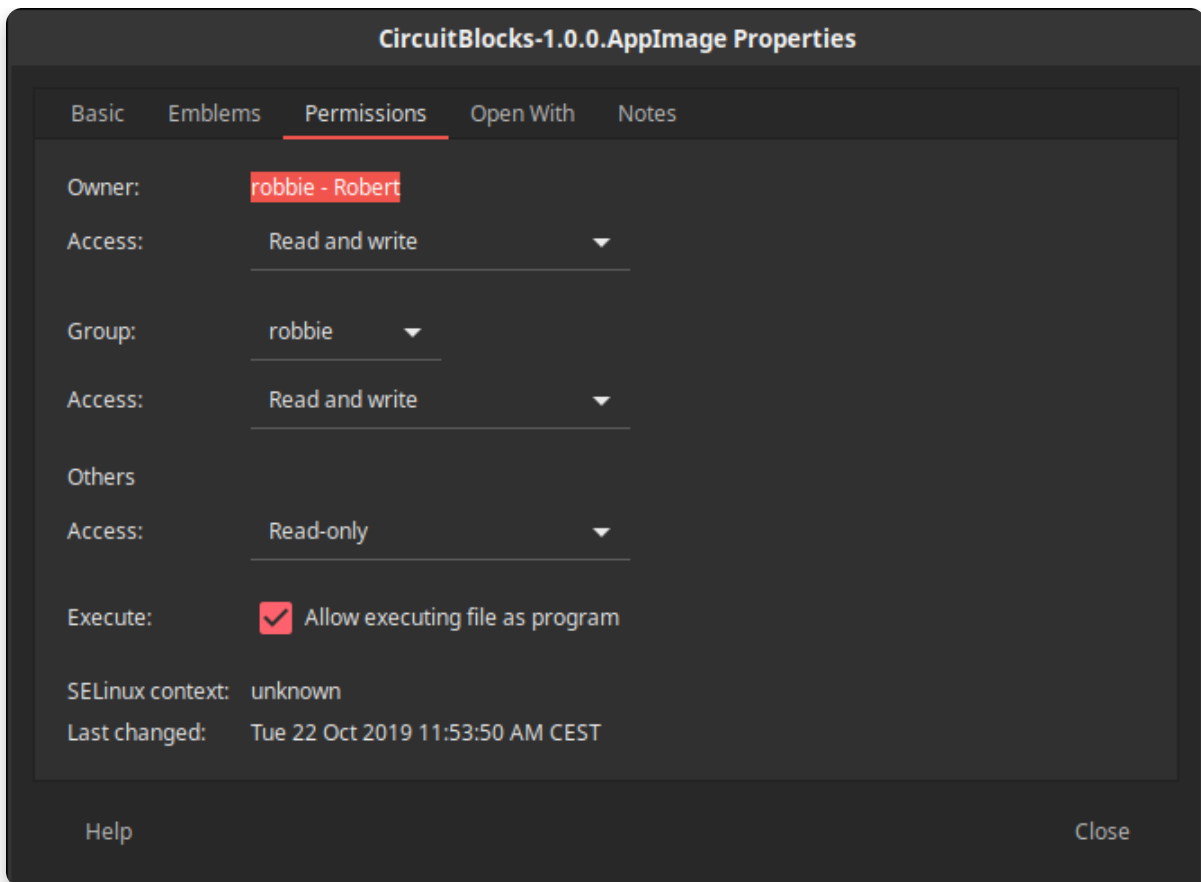
If you have a Linux computer

There are two ways of installing CircuitBlocks on Linux

1. Go to the [CircuitBlocks download page](#)
2. Press the "Linux 64-bit" download button
3. Double-click the file to run the installation (Ubuntu)
or
Open the terminal and write `sudo dpkg -i <path to the downloaded file .deb>` (Other Linux distros)
4. CircuitBlocks will automatically install and create a desktop entry

Stand-alone (AppImage):

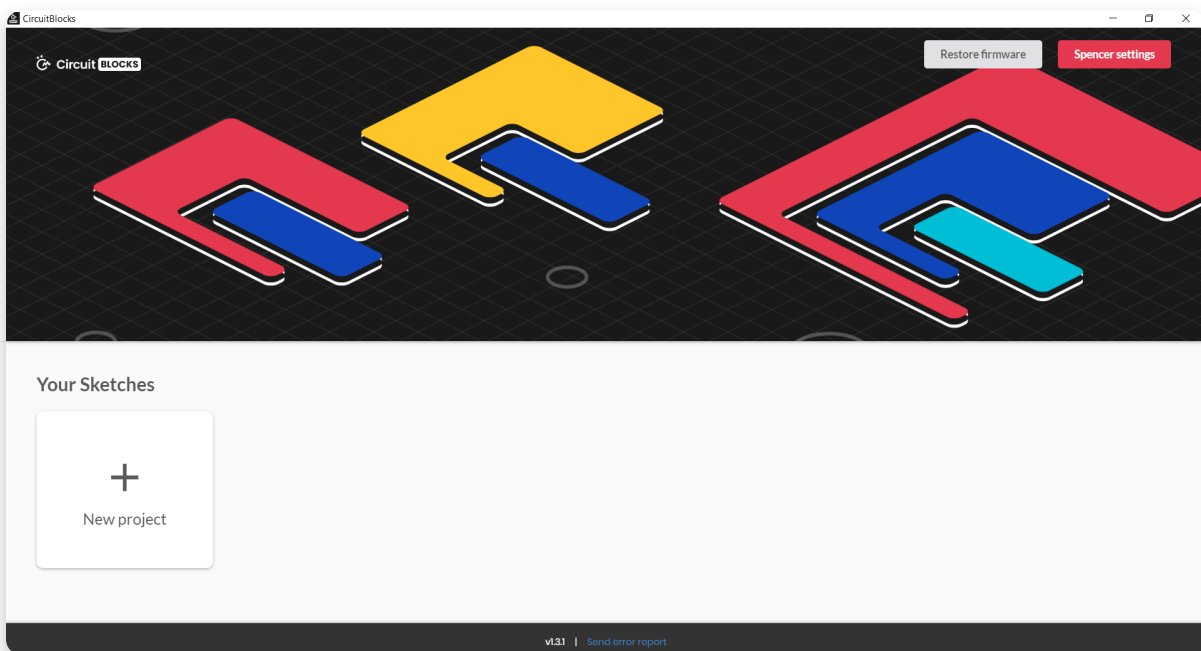
1. Go to the [CircuitBlocks download page](#)
2. Press the "Linux AppImage" download button
3. Right-click on the file and select 'Properties'
4. Go to 'Permissions' and tick 'Allow executing file as program'
5. Double-click the file and the installation will complete automatically



If you encounter any issues with the installation, please reach out to us via email at contact@circuitmess.com and provide a screenshot of your issue and any information you find relevant.

The basics

User interface



When you open CircuitBlocks, you will see a window that looks like this.

It's pretty simple – starting a **new project (we also call them "sketches")** can be done by clicking the 'New project' button.

Saved sketches will appear right next to that button and you can access them at any time.

If you encounter any kind of an issue with CircuitBlocks, press the '**Send error report**' at the bottom of the main screen. You'll get an error report number –

please reach out to us via contact@circuitmess.com and provide your error report number.

Creating a new project (sketch)

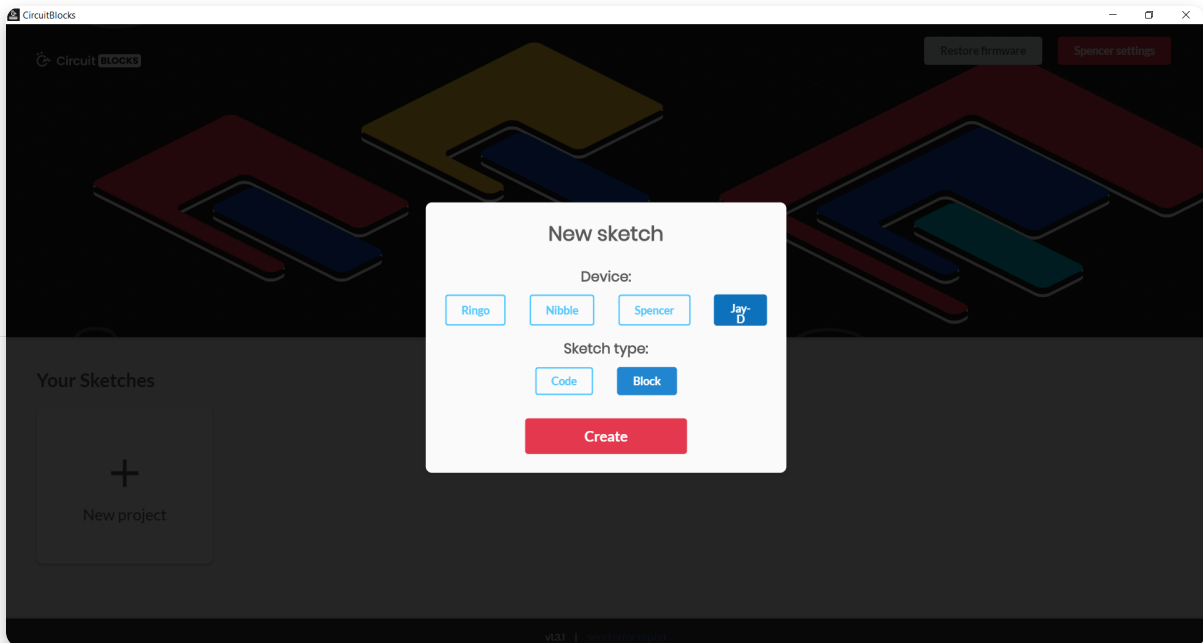
Press on the big "New project" button.

You'll get an option to choose the device and sketch type.

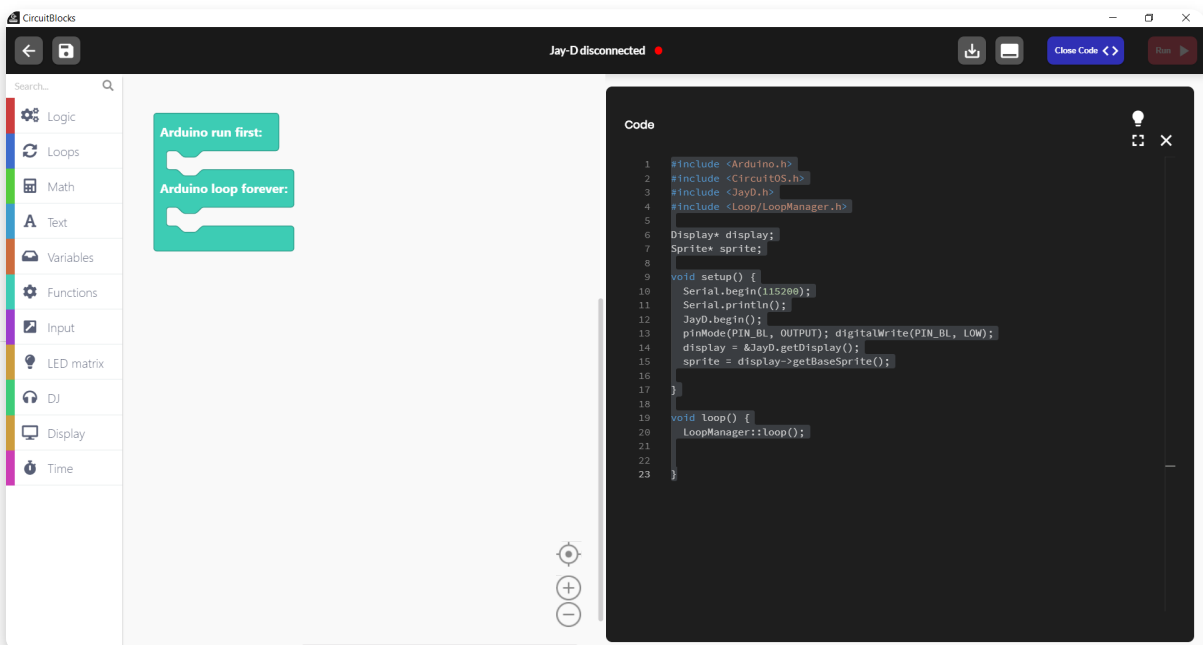
For the device, pick **Jay-D**.

For the Sketch type, choose **Block**.

Press the Create button.



You'll get a screen that looks like this:



On the top of the screen, there is a **toolbar** with a few buttons.

The **block selection bar** is located on the far left - you can take the blocks from there and drop them into the "drawing" area in the middle of the screen.

In the middle of the screen is where you'll be "drawing" your code with colorful blocks.

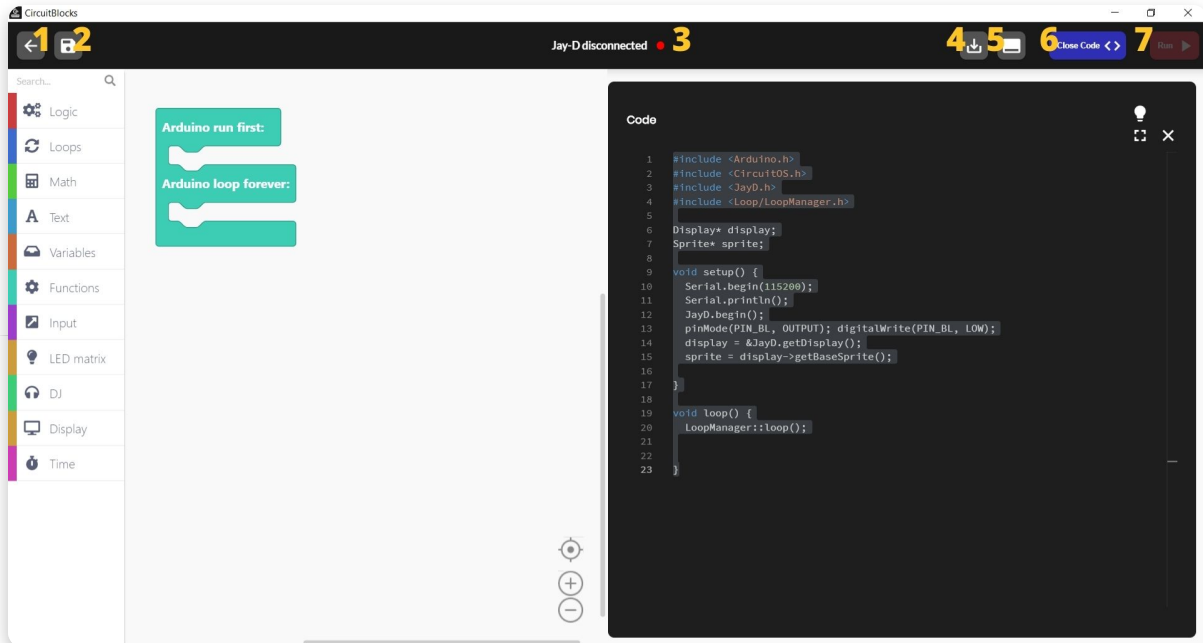
On the right side of the screen, you will see **code written in C++** appear magically by itself when you drag and drop the colorful blocks.

C++ is one of the most popular programming languages, but it's fairly complex to

understand if you've never coded before.

That's why we've created CircuitBlocks - here you can drag and drop colorful blocks that represent parts of code and see what your program would look like in C++. When you get skilled enough, you will be able to switch directly to textual coding in C++ without the need for colorful blocks.

Toolbar



Here's a short explanation of what the buttons in the window toolbar do

1. **Back to the main menu** – returns you to the home screen without saving
2. **Save/Save As** – saves your sketch, make sure to press this button from time to time, and before closing CircuitBlocks
3. **Jay-D connection indicator** - the red dot turns green if your Jay-D is connected to your computer via a USB cable
4. **Export to binary** - saves a binary file of your code to your computer. This is a more advanced function that you won't need for now.
5. **Serial monitor** - this button opens a window that we call the "Serial monitor". "Serial" is a nickname for a type of communication that is happening between Jay-D and your computer. In this window, you will, later on, be able to see the messages that are being sent from Jay-D to your computer via the USB port.
6. **Close code** - with this button, you can close or re-open the code window on the right of the screen. This is useful if you need more screen space for seeing your colored blocks.
7. **Run** - This button will translate the code you have constructed in CircuitBlocks to *machine code* that Jay-D understands (beep boop beep boop 1011100101) and send the code to your Jay-D via the USB port.

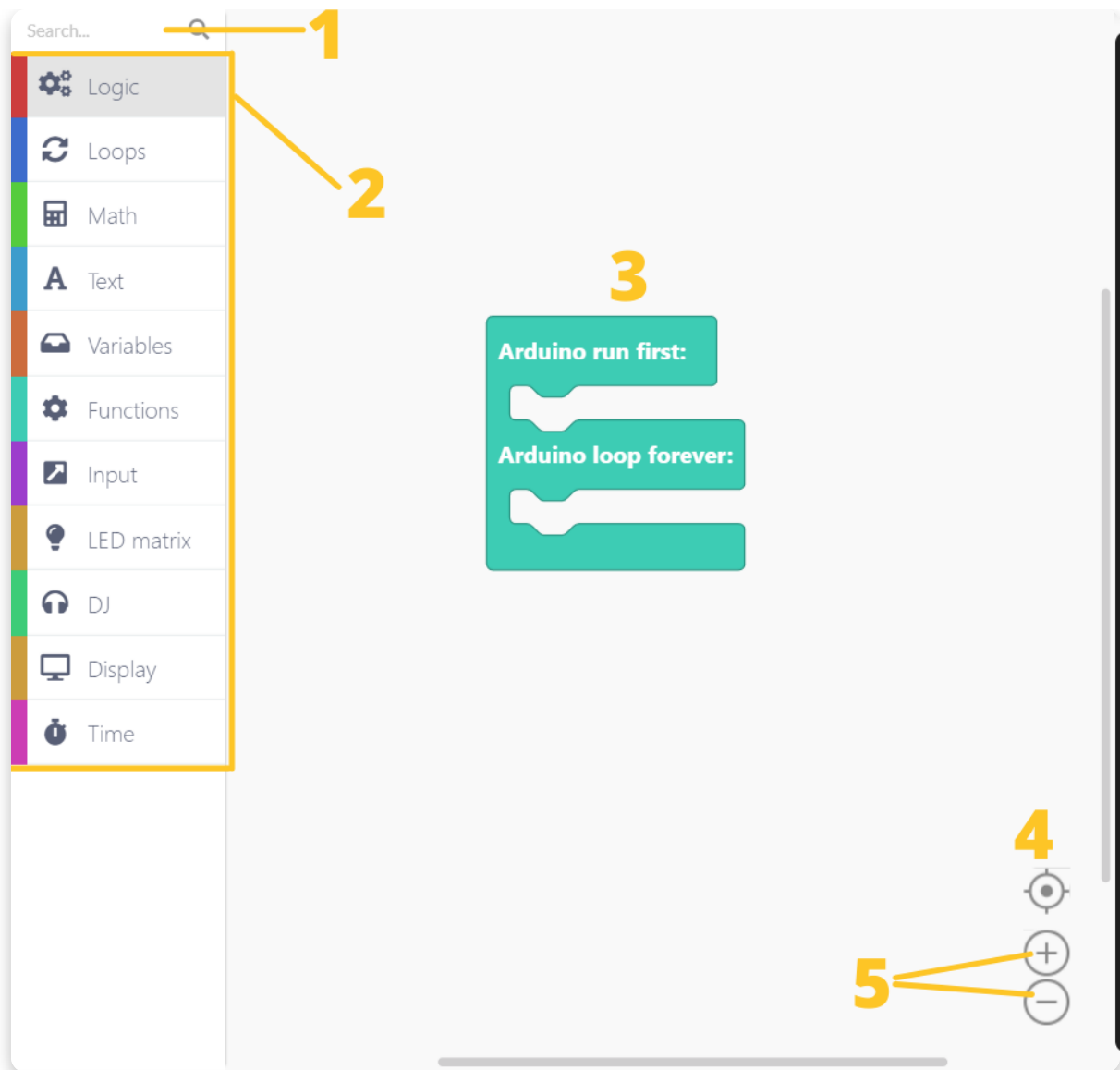
Code window

```
Code
1  #include <Arduino.h>
2  #include <CircuitOS.h>
3  #include <JayD.h>
4  #include <Loop/LoopManager.h>
5
6  Display* display;
7  Sprite* sprite;
8
9  void setup() {
10     Serial.begin(115200);
11     Serial.println();
12     JayD.begin();
13     pinMode(PIN_BL, OUTPUT); digitalWrite(PIN_BL, LOW);
14     display = &JayD.getDisplay();
15     sprite = display->getBaseSprite();
16
17 }
18
19 void loop() {
20     LoopManager::loop();
21
22 }
23
```

The so-called "Code window" has the following parts:

1. **Main code screen** - code written in C++ will appear here as you drag and drop colorful blocks on the left side of the screen. You'll see that some parts of the code are colored in funny colors. Programmers call this *syntax highlighting*. Basically, what is happening is that different categories of code commands are colored differently so that programmers can understand the code more easily.
2. **Light/dark theme switch** - you can toggle the background and text color of the code window with this button.
3. **Expand** - stretches the code window across the entire screen. Press it again to resize it to the half-screen again.
4. **Close** - closes the code window, the same functionality as the 'Close Code' button from the toolbar.

Drawing board



The drawing board is where the magic happens.

It has the following parts

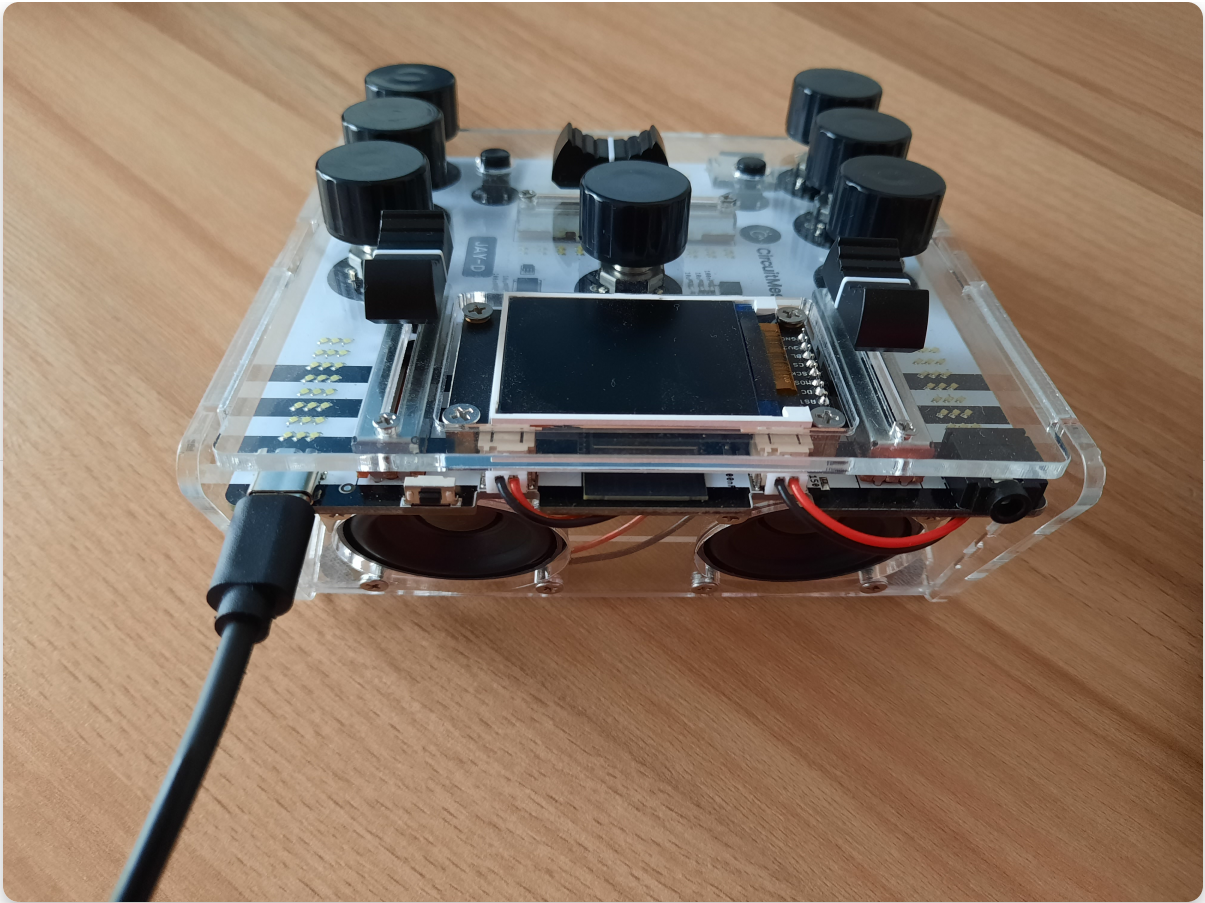
1. **Search bar** – type the name of the component you are looking for here.
2. **Component selector** – the blocks are divided into different categories here. Each category has a specific color designated to it.
3. **Drawing area** – you will drag the blocks from the component selector and drop them into the drawing area. This is how code is made. Easy peasy!
4. **Center tool** – if you get lost when scrolling across the drawing area, press this button and it will center your view on the blocks you have dropped on the drawing area.
5. **Zoom buttons** – zoom in and out of the drawing area.

Step by step

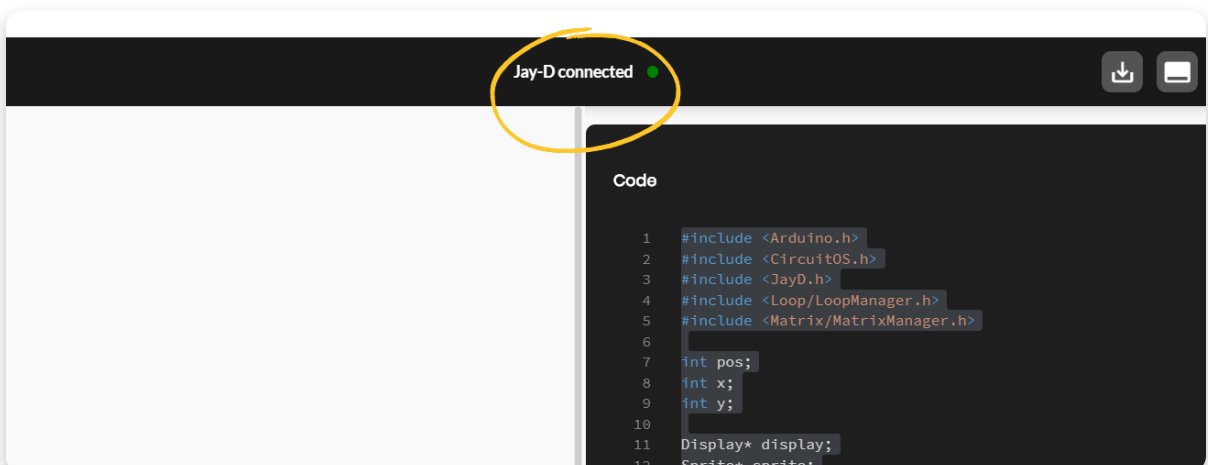
Starting off with LED lights

Ready to start coding?

Firstly, open CircuitBlocks and connect your Jay-D to your computer's USB port.



CircuitBlocks should now say "Jay-D connected".



If CircuitBlocks didn't recognize your Jay-D, please check if the USB cable is

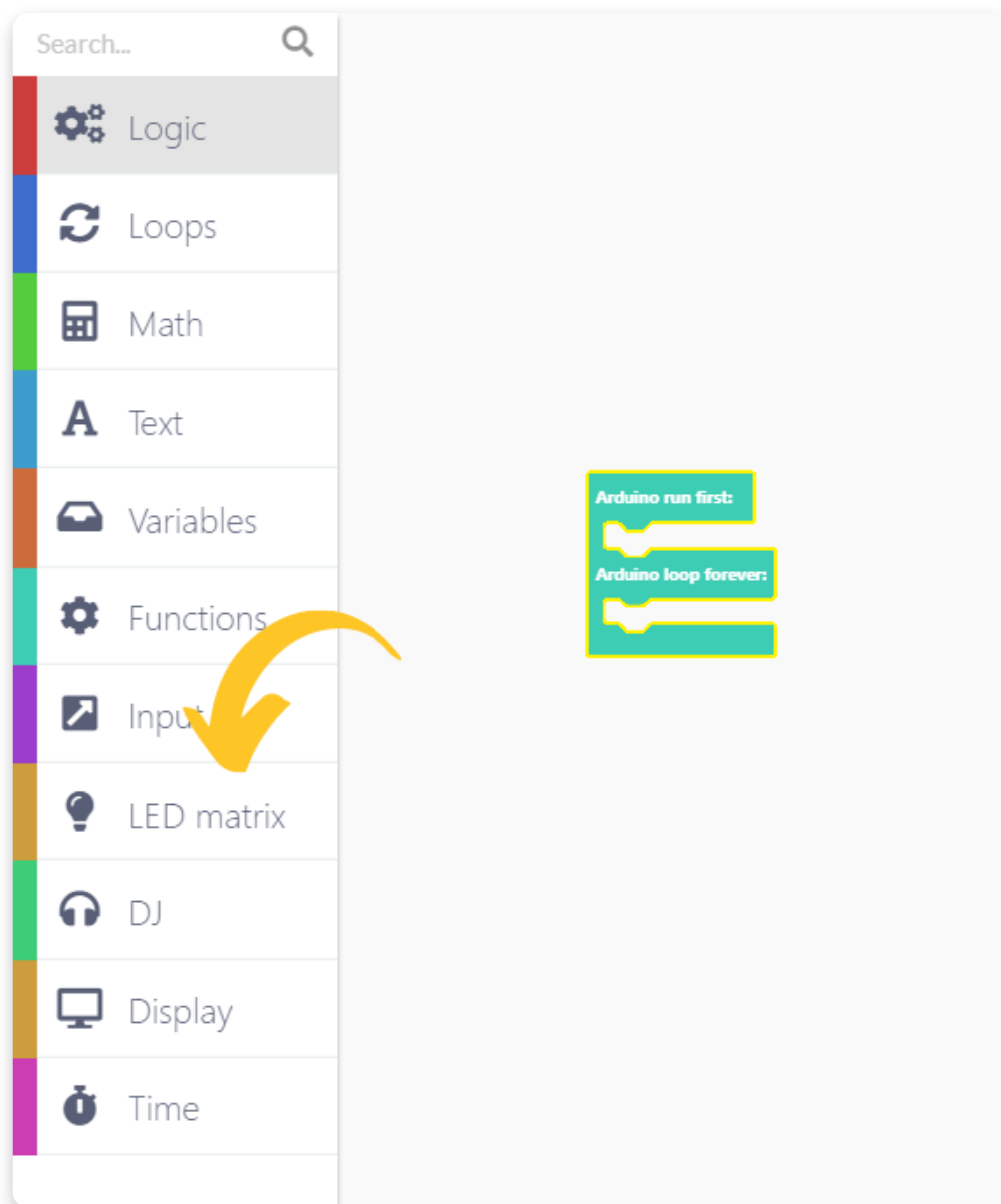
plugged in properly and if you are using a working USB port on your computer.

If you still cannot get CircuitBlocks to recognize your Jay-D, something possibly went wrong with the driver installation on your computer. Drivers are these little programs that help your computer communicate with Jay-D and they sometimes act funny. Reach out to us via email at contact@circuitmess.com if you cannot get your computer to recognize your Jay-D.

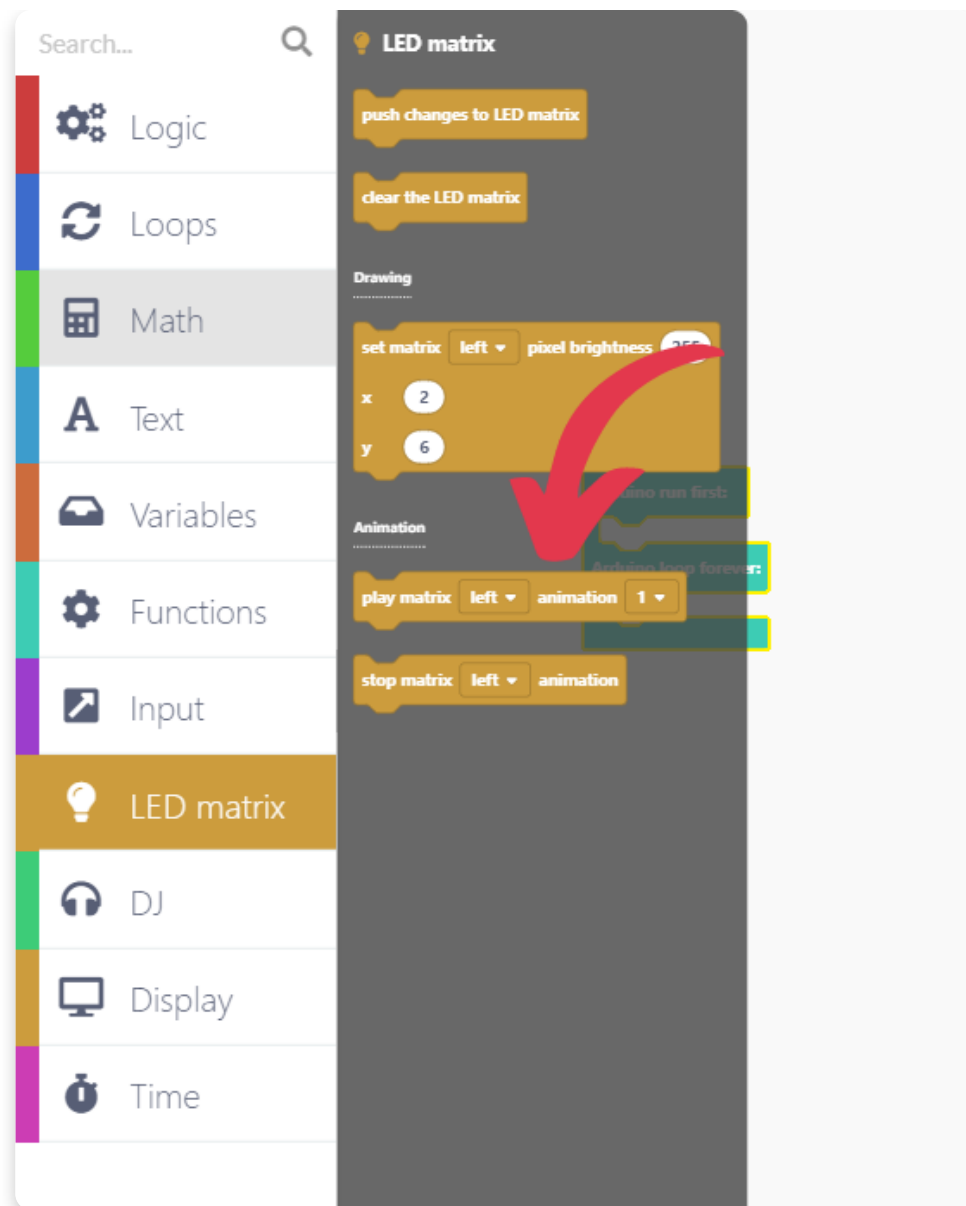
Let's make an animation on Jay-D's LED display

Find the section called "LED matrix" on the left side of the screen.

This section contains a group of blocks that can be used for displaying animations and on Jay-D's LED display (we also call this display the "LED matrix").



Find the block called "Play matrix left animation 1".

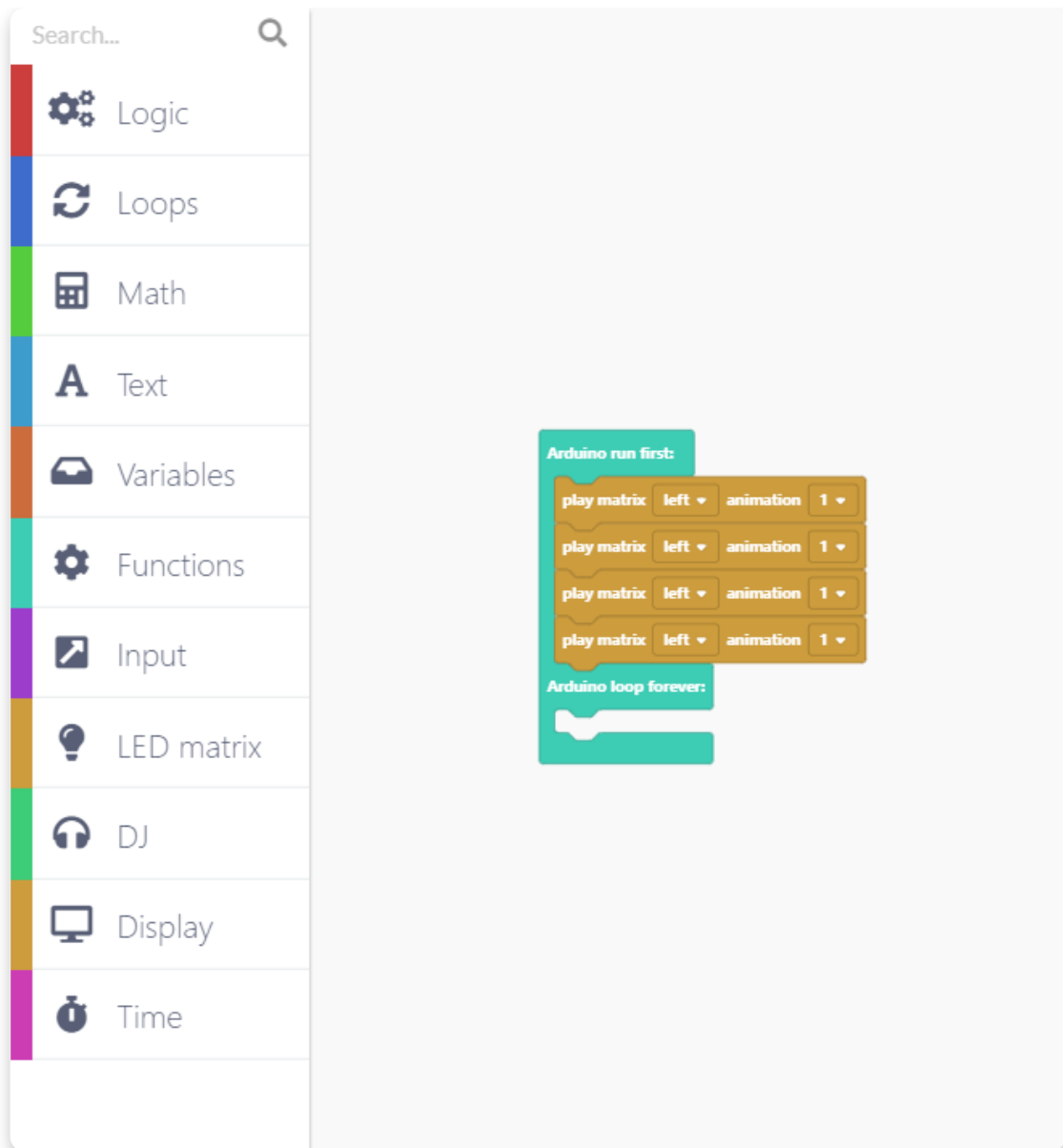


Drag and drop the block into the turquoise block that's already in the drawing area.

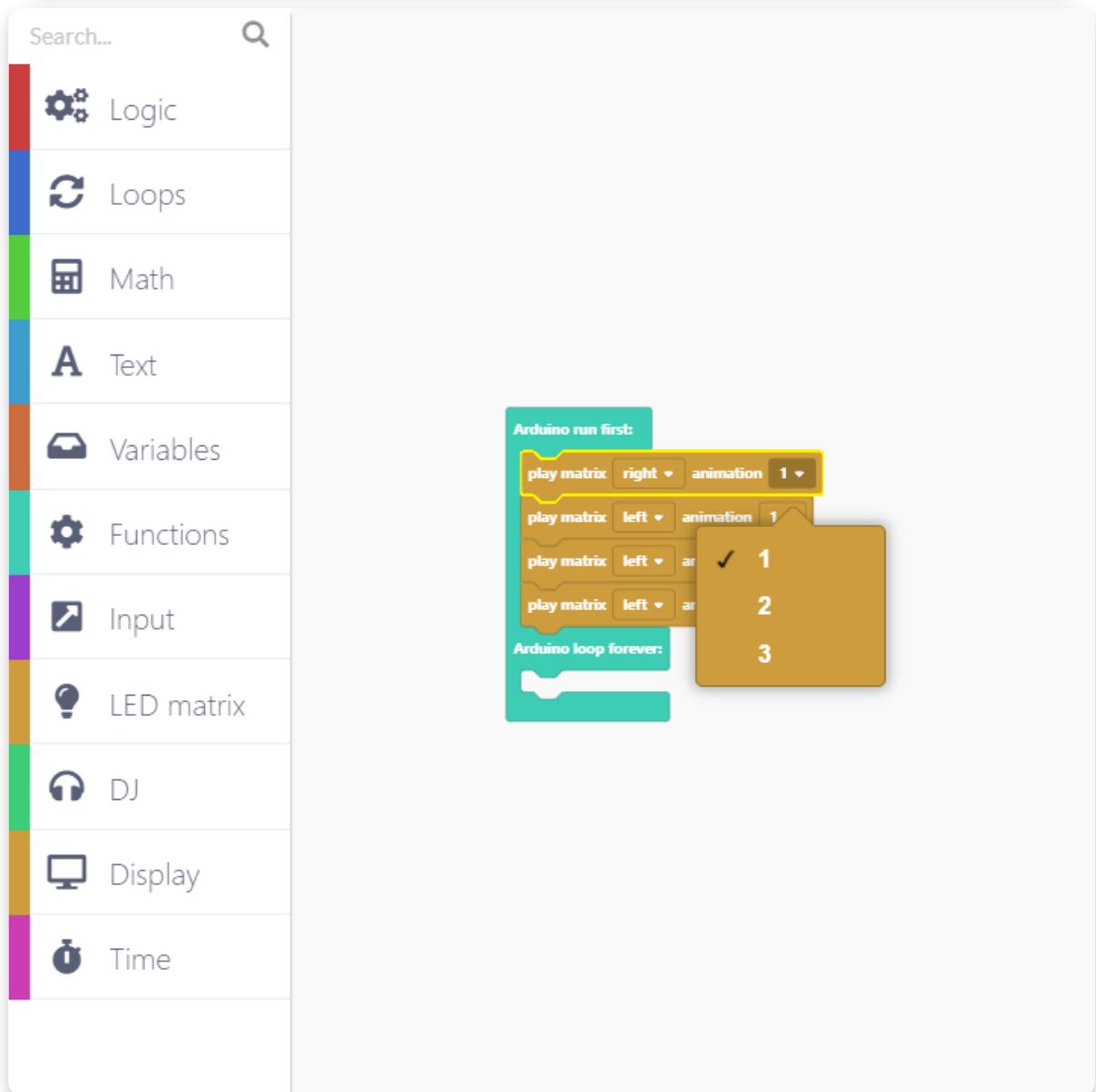
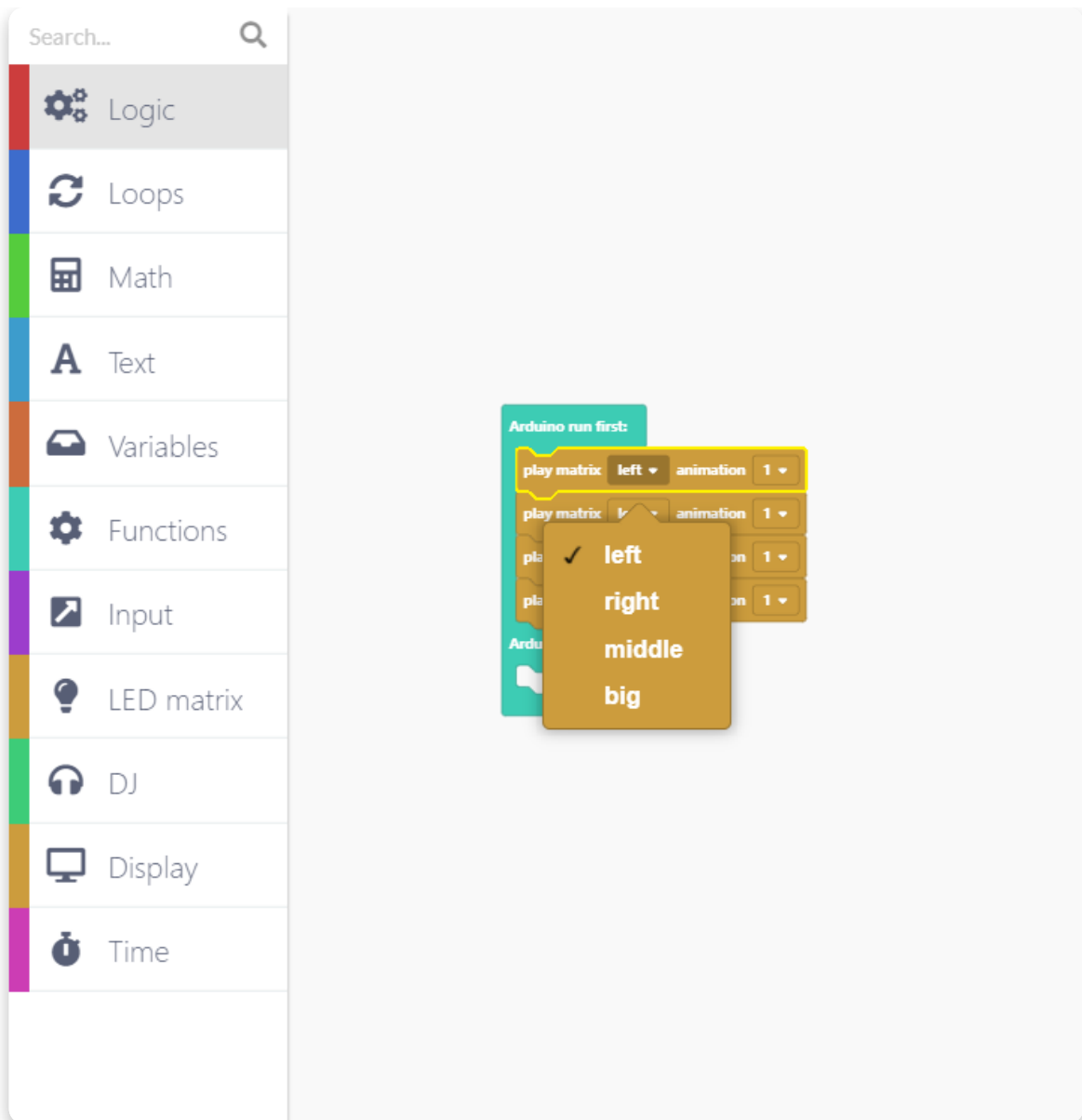
Stick the Play animation block under the text saying "Arduino run first".

We want to put the play animation block here so that the animation starts playing only once when the device starts.

Since Jay-D has 4 LED matrixes, let's repeat that step 3 more times for each of the matrixes. Just drag and drop the block like in the photo below and we'll then arrange the matrixes and animations.



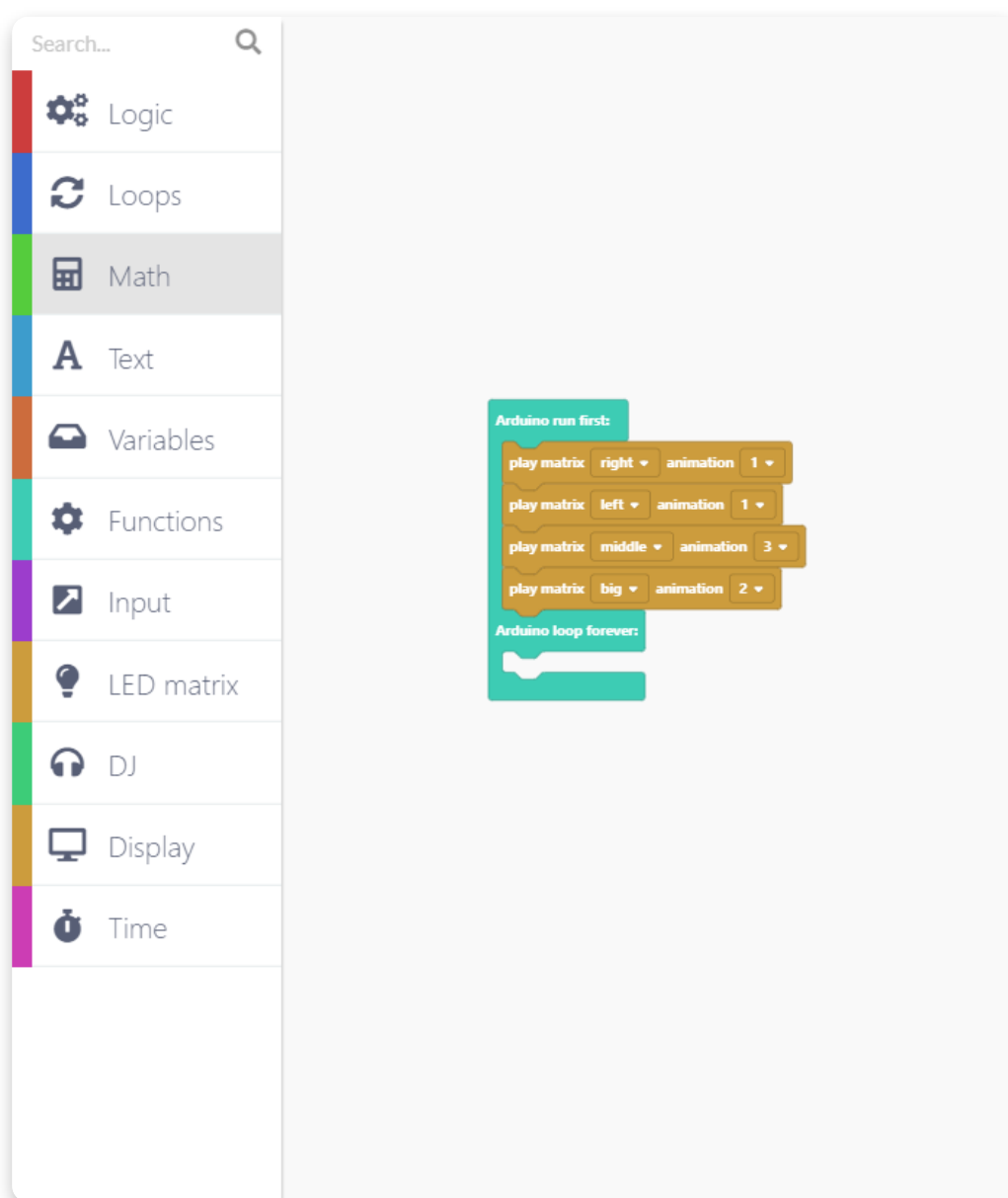
Click on the tiny arrow on the Play animation block and a drop-down menu will appear.



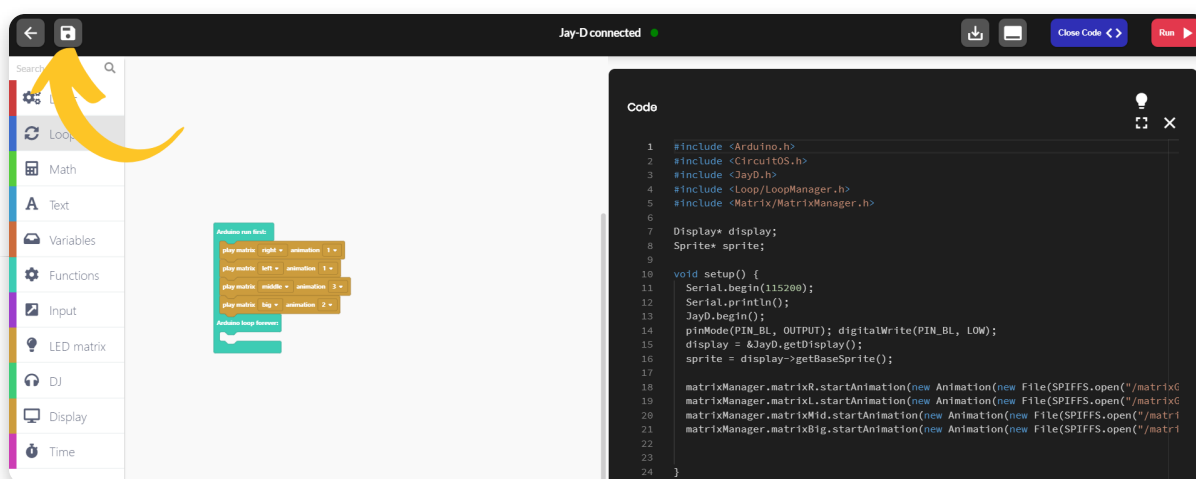
You can start arranging the matrixes (**left, right, middle, big**) and animations (**1,**

2, 3). Those are the GIF animations that are saved on your Jay-D's flash memory and you can't really go wrong with them, so get creative.

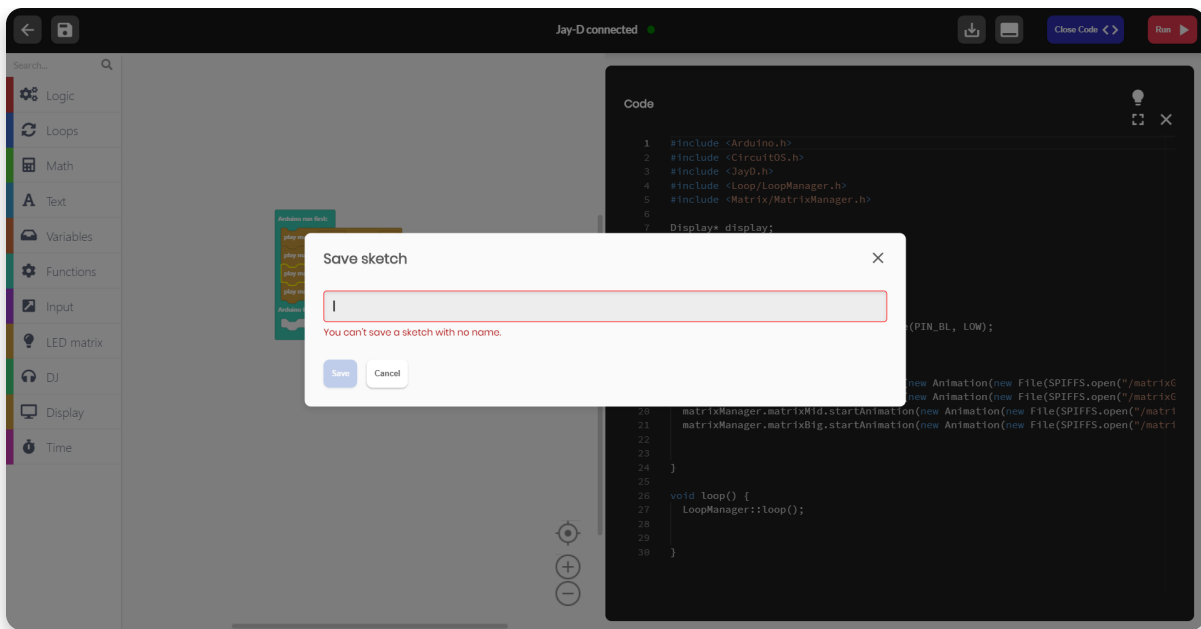
Check out the following example of the blocks with matrixes and animations picked:



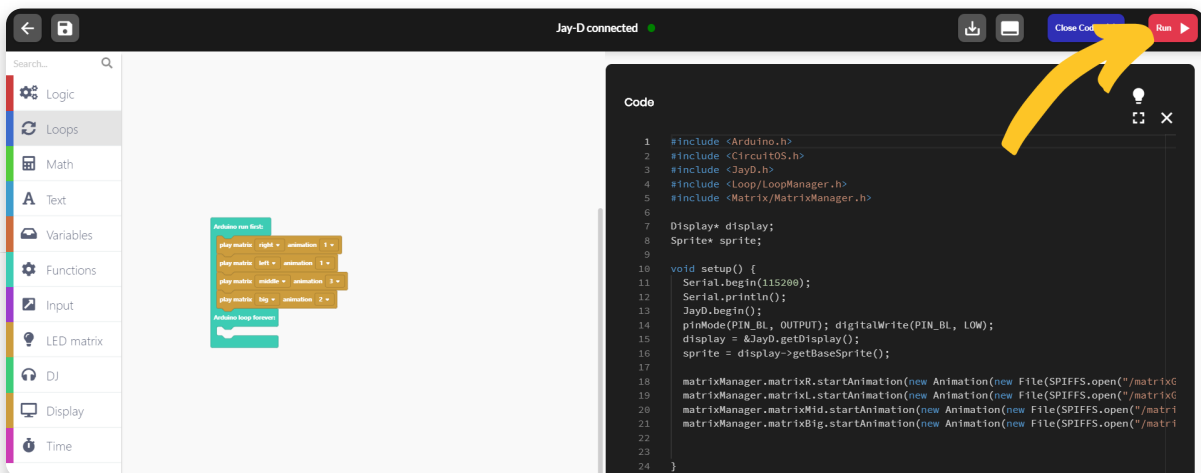
Now would be a good time to save your program. Click on the Save icon on the top left corner of the screen.



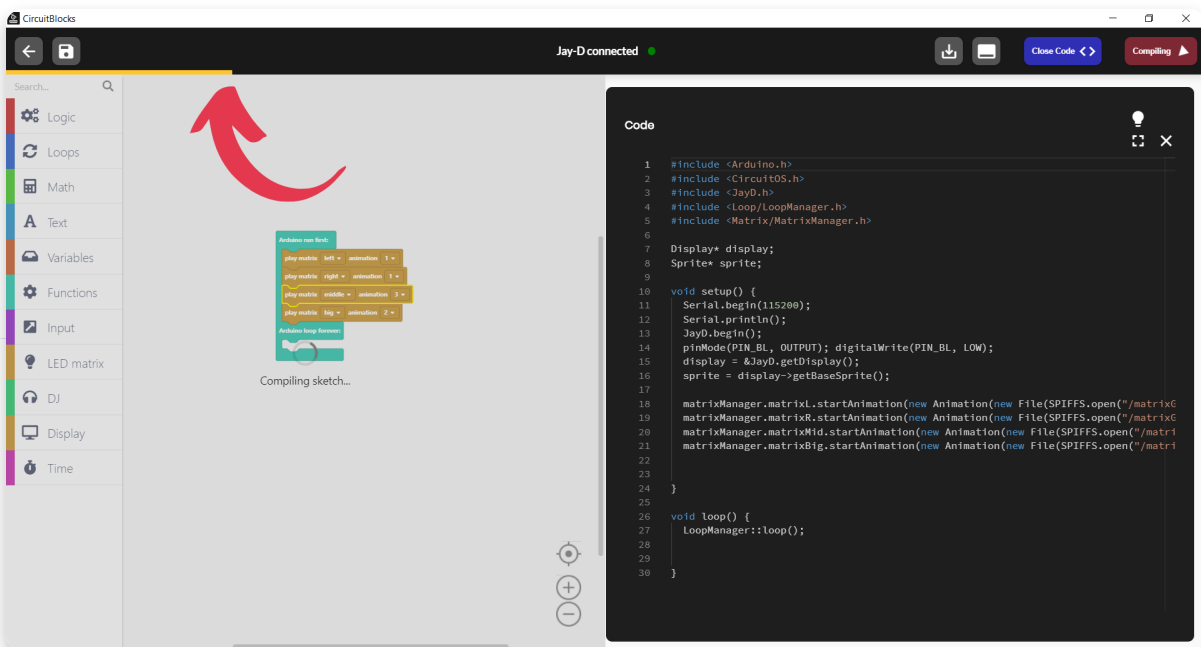
Pick a name for your program and click "Save":



Let's try this code out. Find the big red Run button on the top right and press it.



A yellow loading bar will appear on the top of the screen.



Your program is now going through a process called **compilation**.

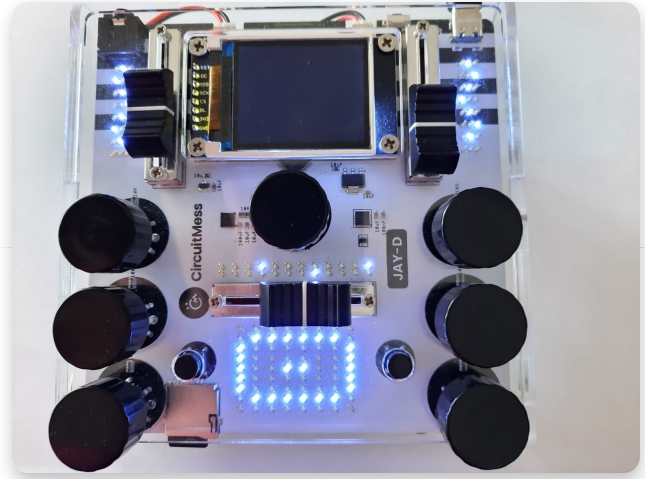
Code compilation (or code compiling) is the transformation from a human-readable form of code (such as the colorful blocks you are looking at) into machine code (a form of code that the computer understands – ones and zeros all squashed together).

Since this is the first time you are compiling code for your Jay-D, it might take a few minutes to compile. This is happening because CircuitBlocks needs to compile all the important core parts of the code needed for your Jay-D and save it to your computer. When this compiles for the first time, it will be saved on your

computer and all of your next programs will compile much faster (yay!).

If your code was compiled and uploaded successfully, you should see the new animations playing in a loop on Jay-D's LED matrix.

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



Something not working? Please press the Send error report on CircuitBlocks' homepage, contact us via email at contact@circuitmess.com and provide us with your error report ID.

LED Matrix

Let's go a bit further with programming the LED lights.

In this step, we'll expand on what we did in the previous step and explore what we can do with different variables.

Start off by creating a new block project for Jay-D and putting three commands that will play animations for left, right, and middle LED display like in the previous step.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- Input
- LED matrix
- DJ
- Display
- Time

Arduino run first:

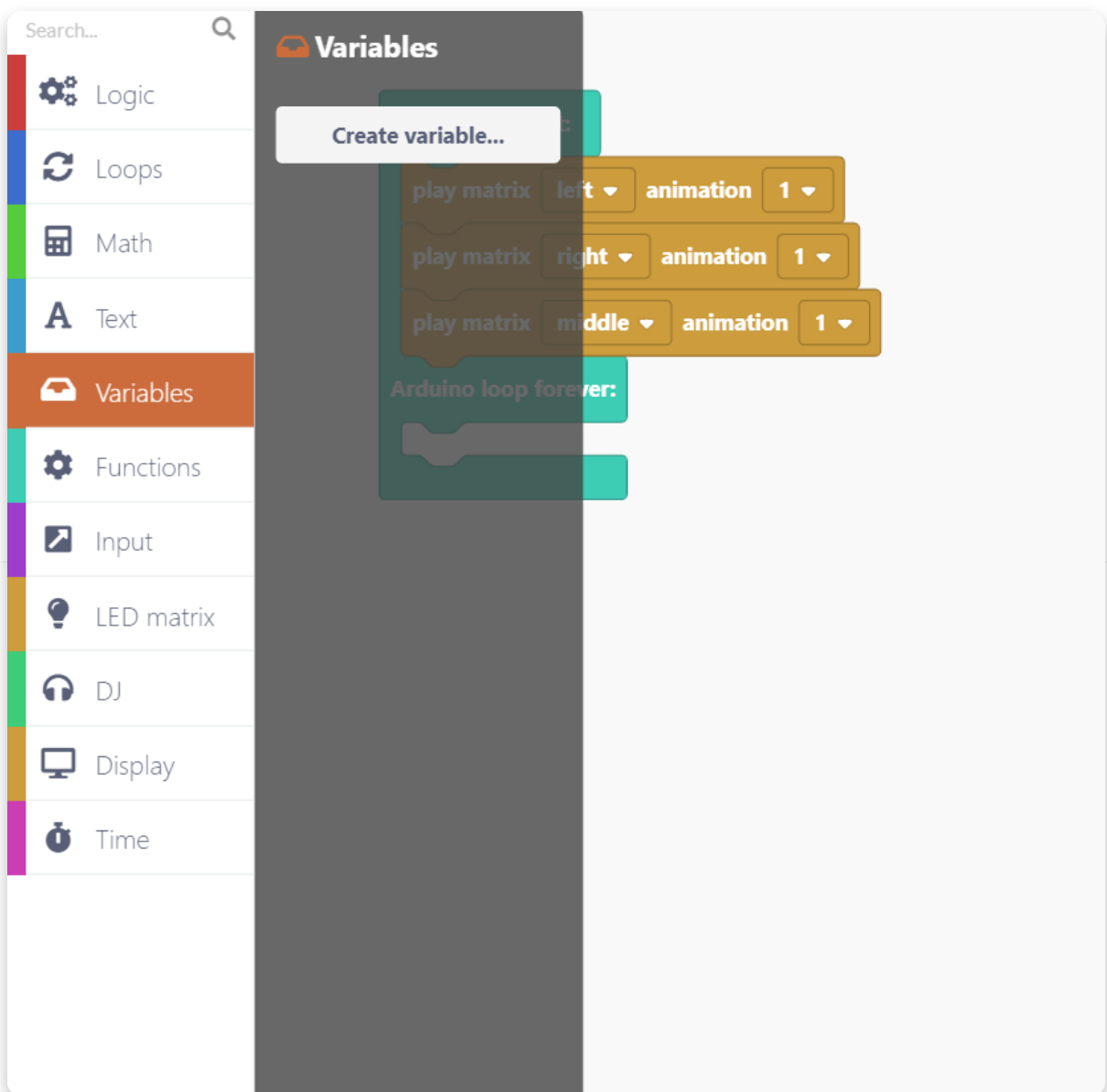
- play matrix left animation 1
- play matrix right animation 1
- play matrix middle animation 1

Arduino loop forever:

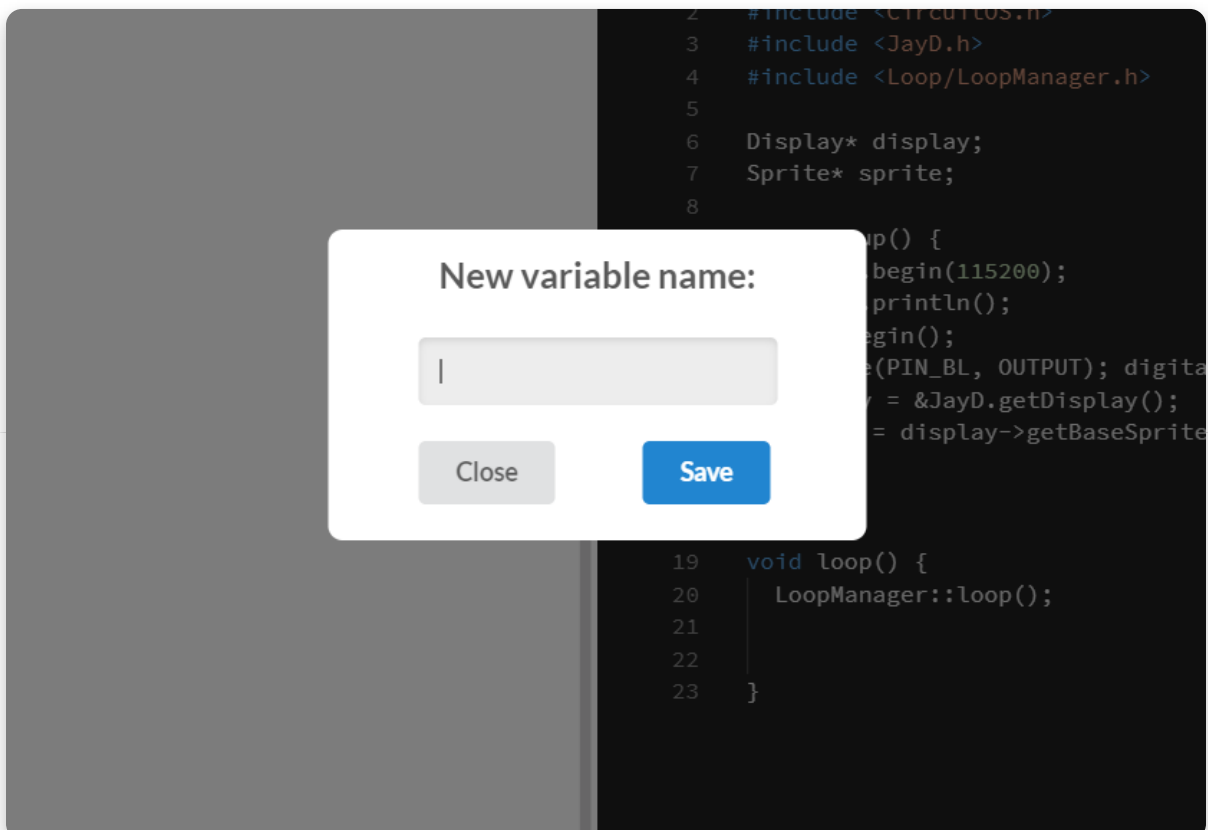
In computer programming, a **variable** is a storage location that contains a value. Every variable has a specific name. You can store and change the value of

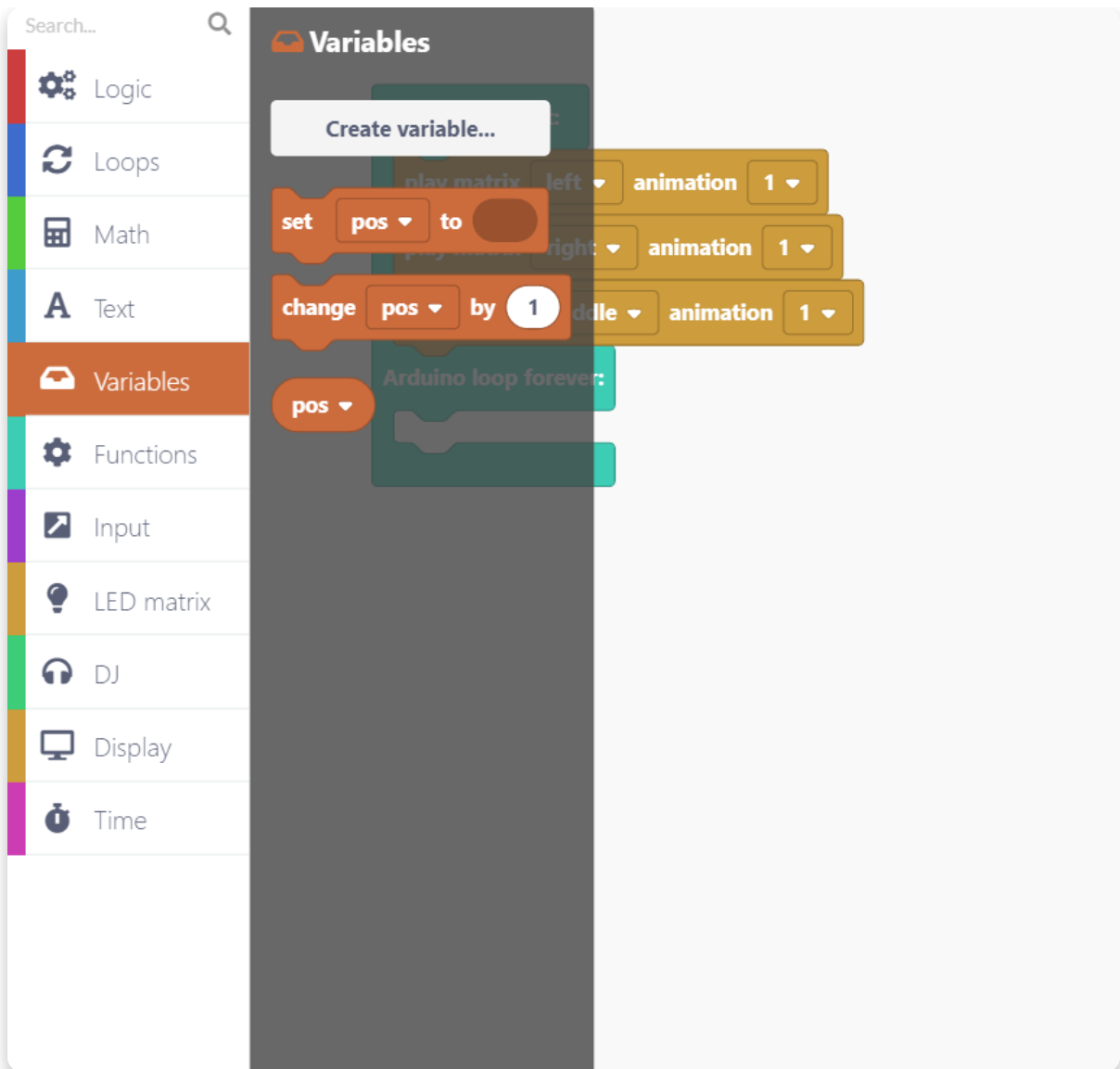
a variable.

Firstly, let's create a variable. Find the section named "Variables" and press the "Create variable..." button.



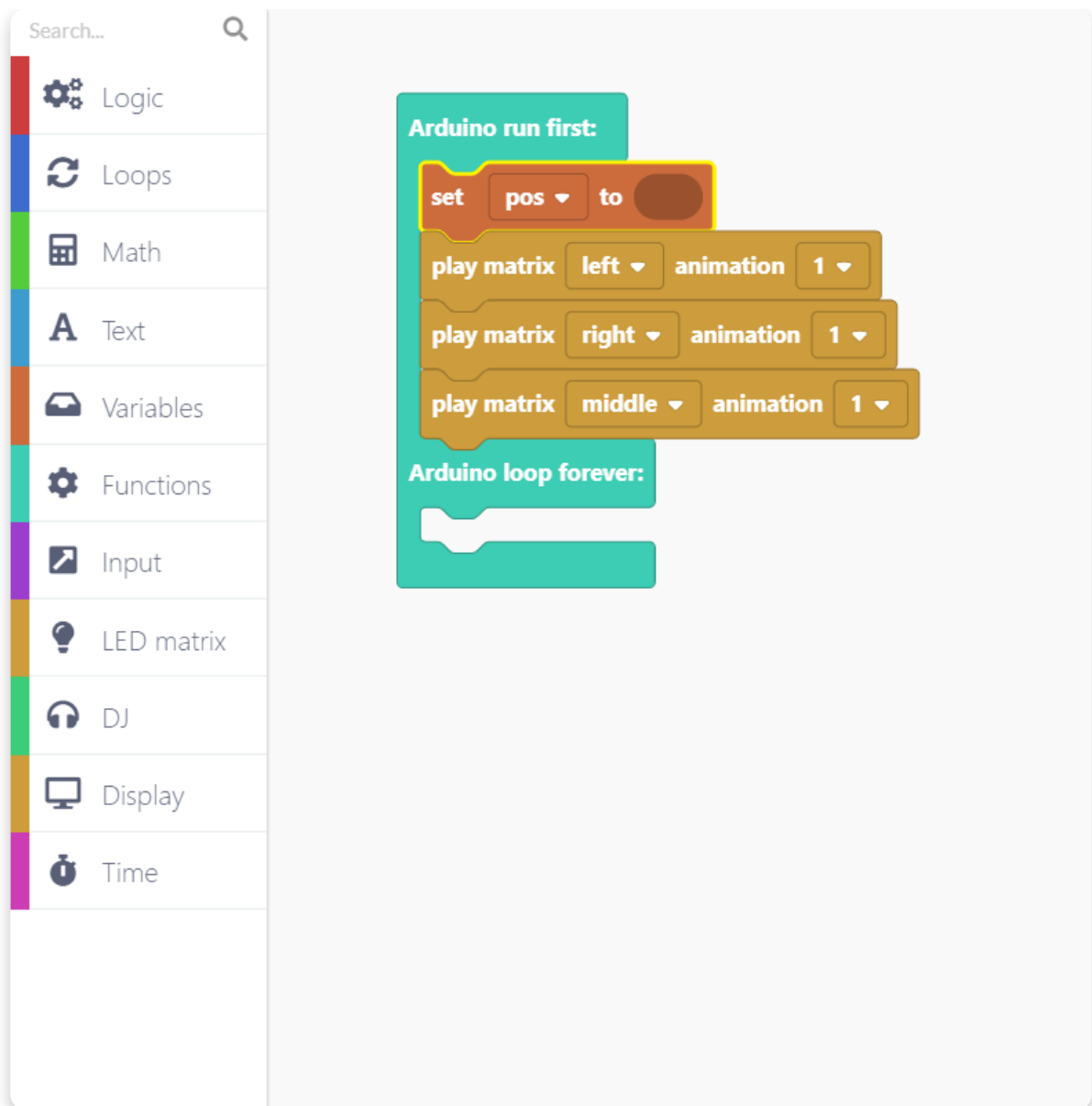
Now you need to give this variable a name. Let's call this one **pos** - short for a position on the display.





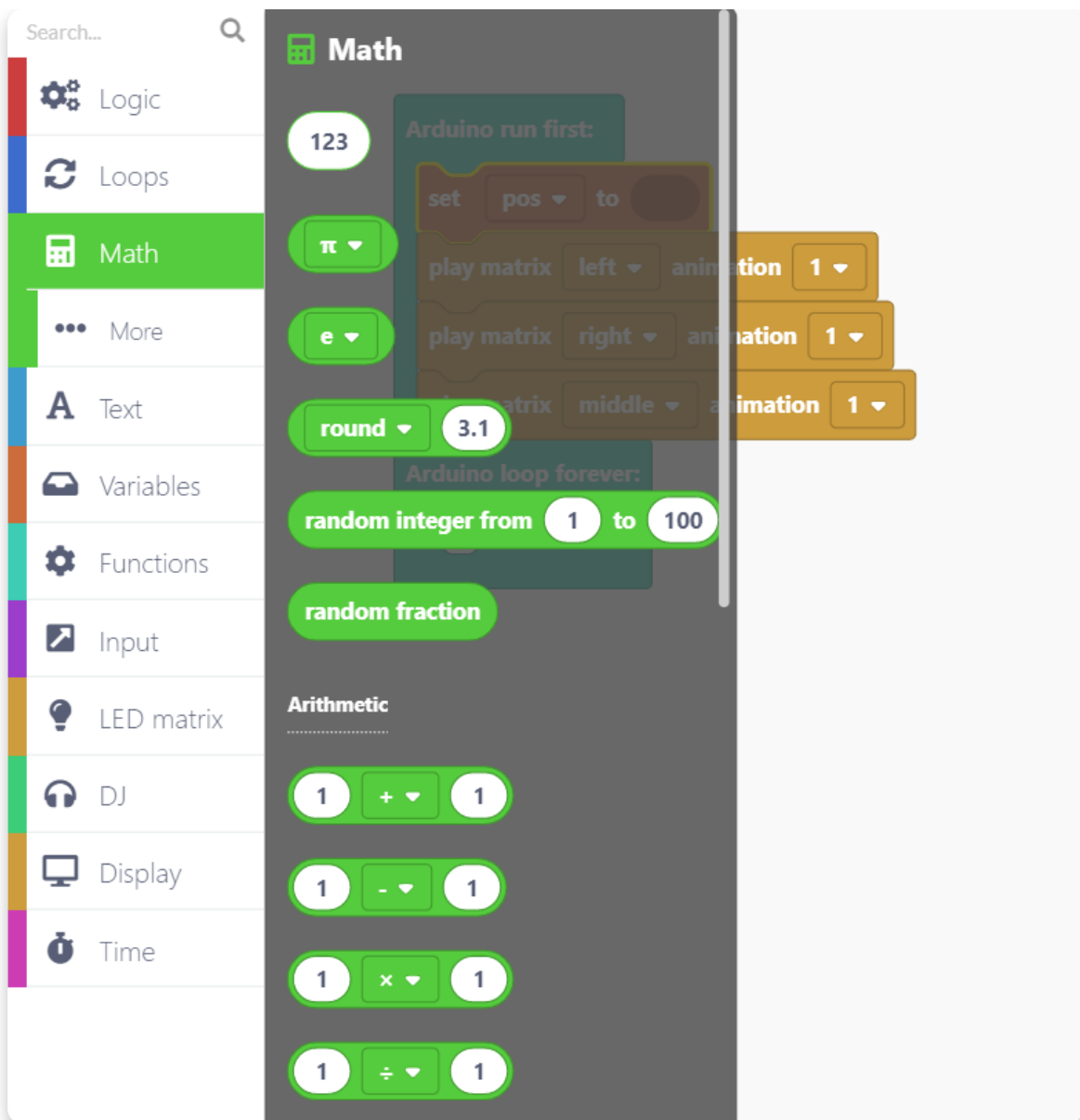
When we create a variable, it's undefined - it has no value. We must set a value for every variable when our computer program starts. That's why you'll need the "set variable" block.

Put this block in the "Arduino run first:" branch.



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

Find the block named "123" in the "Math" section. This block is a numerical value block, and you can type in the numerical value you want once you drop it onto the drawing area.



Place the block here:



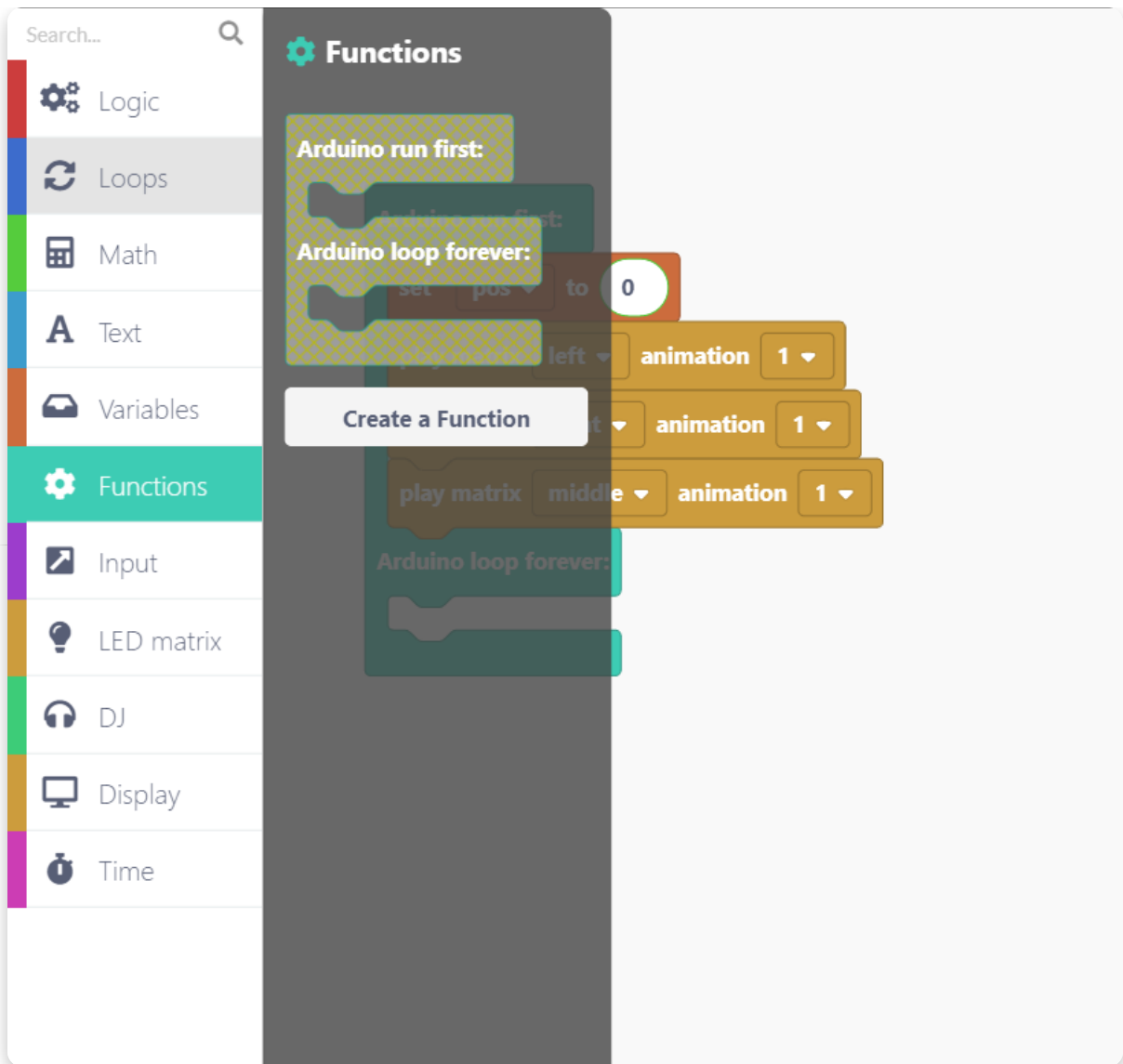
Now click on the block and type in the value. Set the numerical value of the block to 0. You can do this by simply typing the number 0 (zero) on your keyboard.



Now that we have a variable created and set to zero, let's change the variable when a certain event is triggered.

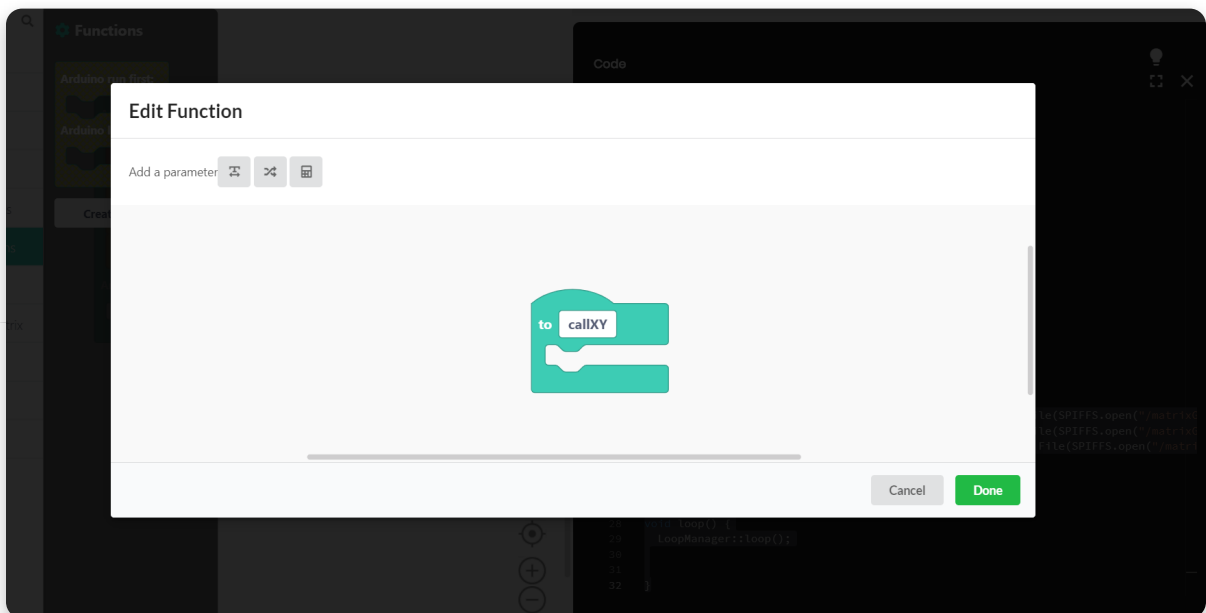
In the next step, we'll show you how to calculate coordinates x and y for the lights on the display.

Open the "Functions" section and choose to "Create a Function".

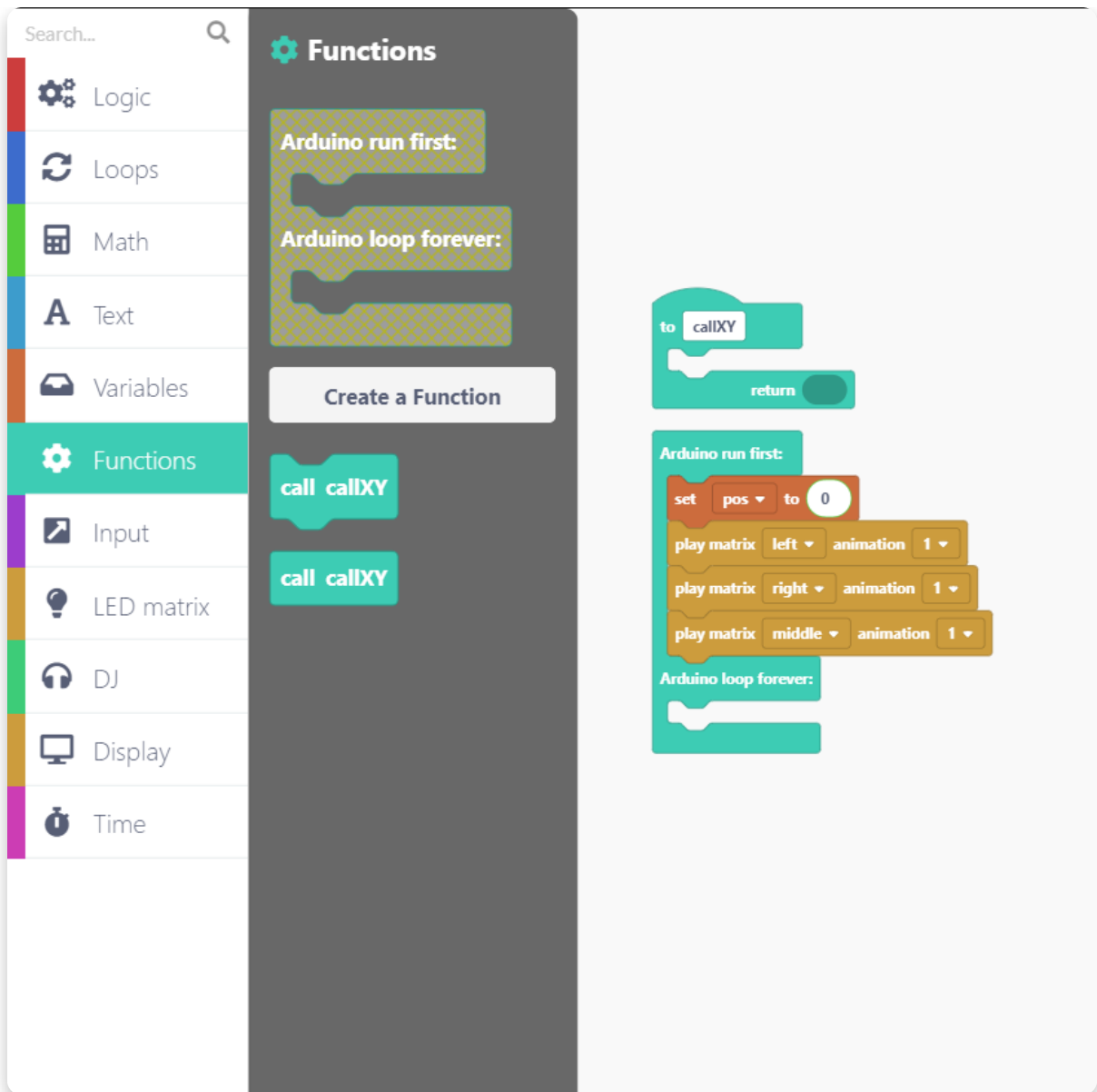


A window like the one in the picture should pop up on your screen. Here, you can name the function you're creating.

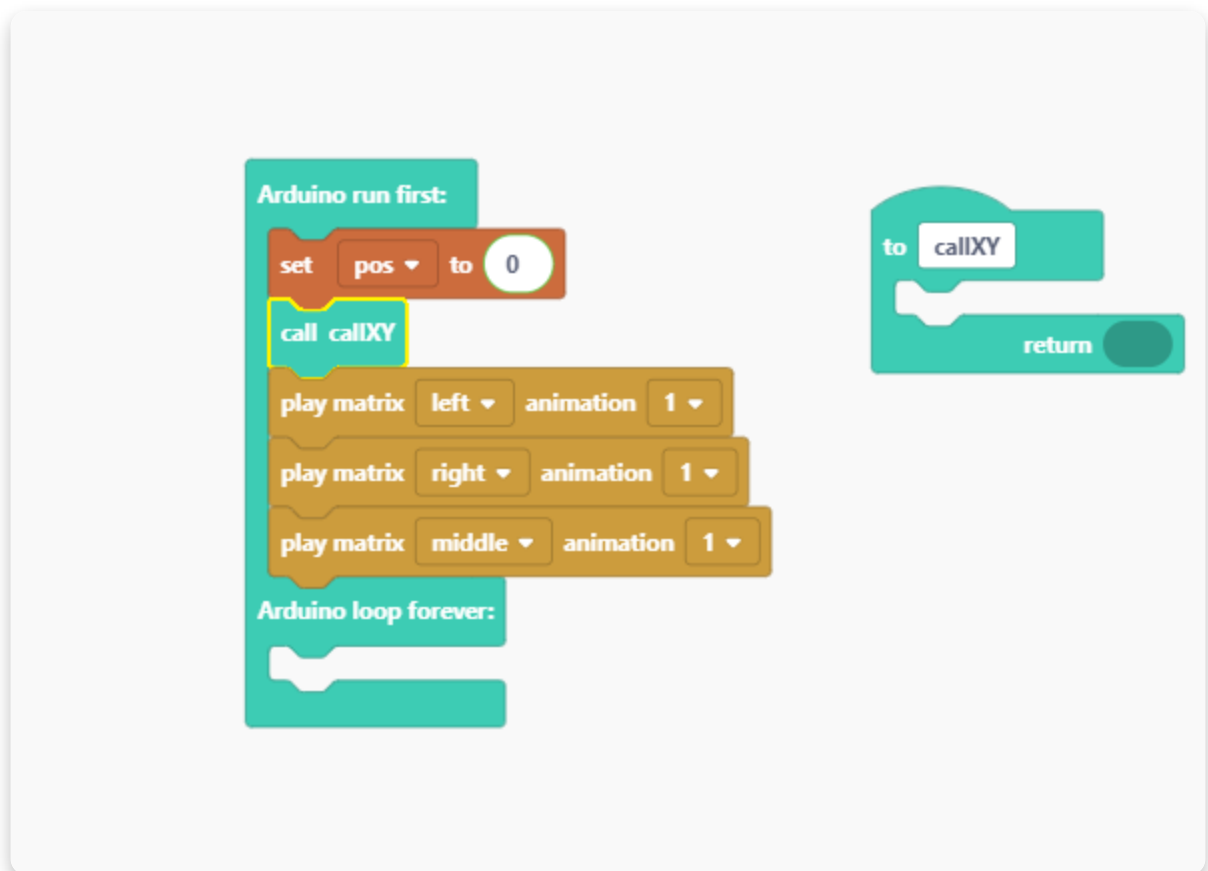
Since we'll use this to calculate coordinates x and y, let's call it "callXY":



After you named the function, it should appear as a block on the screen with a couple more options to drag and drop into the "Functions" section.



Drag and drop the first "call callXY" block under the "set pos" block:

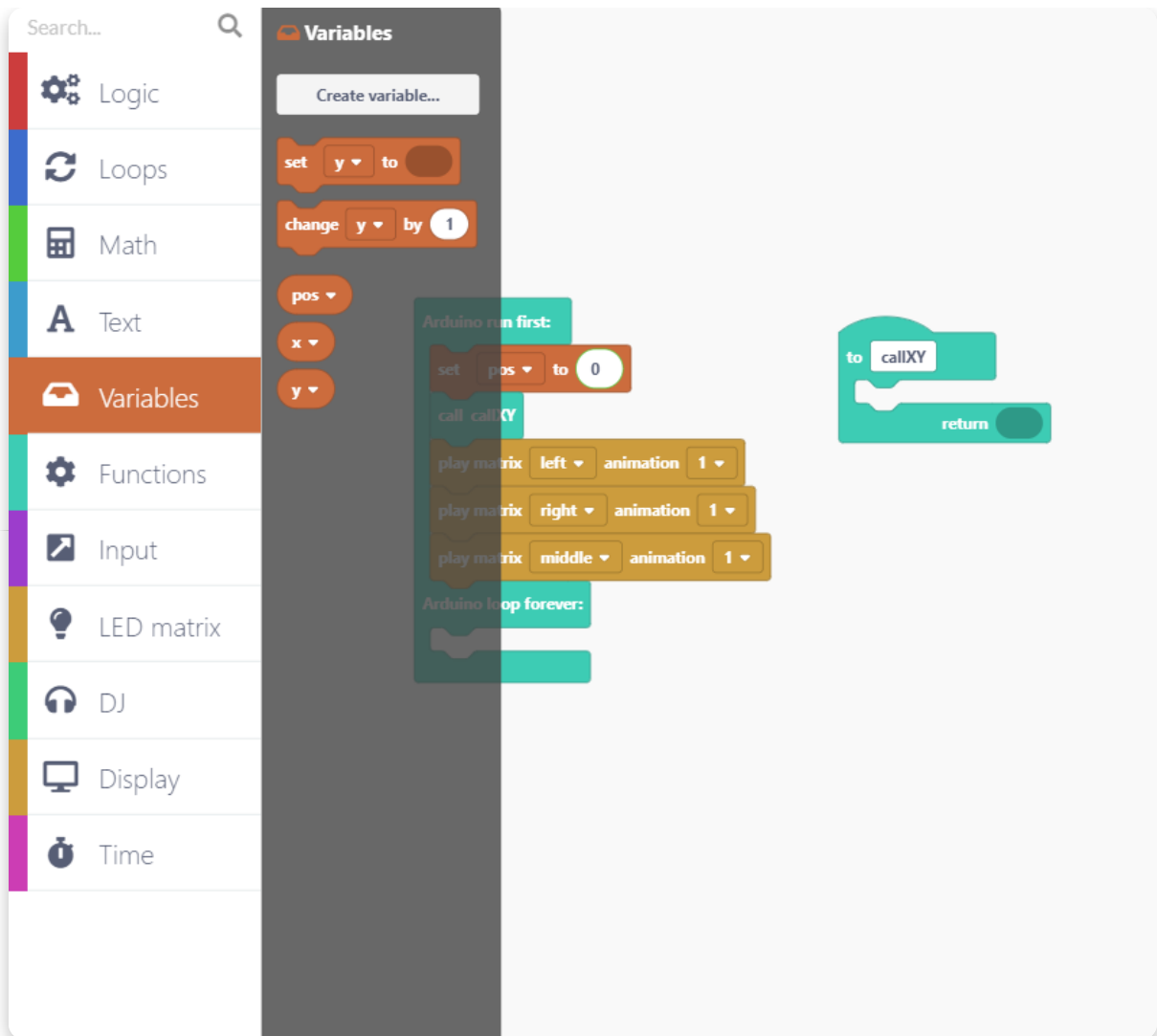


Now let's define what this function does.

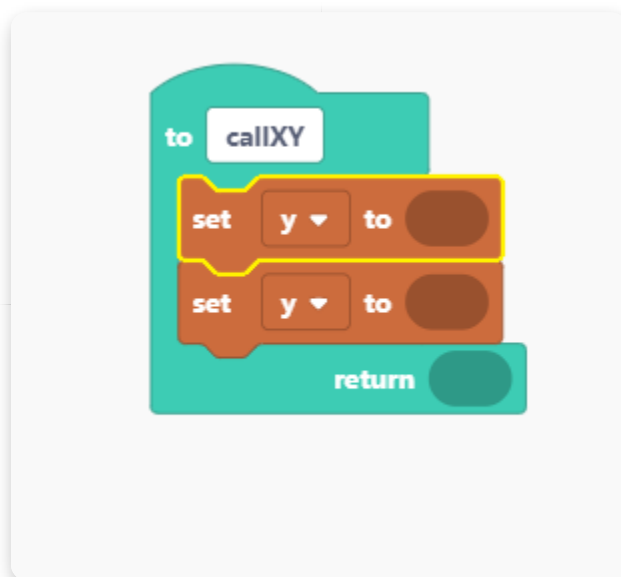
We'll create two new variables just like we did with the "pos" variable.

Go to the "Variables" section and create the following variables: x and y.

The new variables should appear in this section.

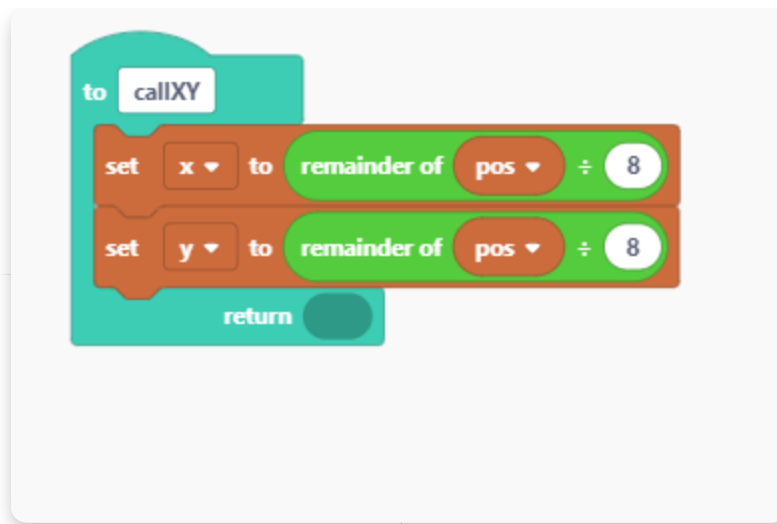


Now drag and drop the "set y to" block here twice:

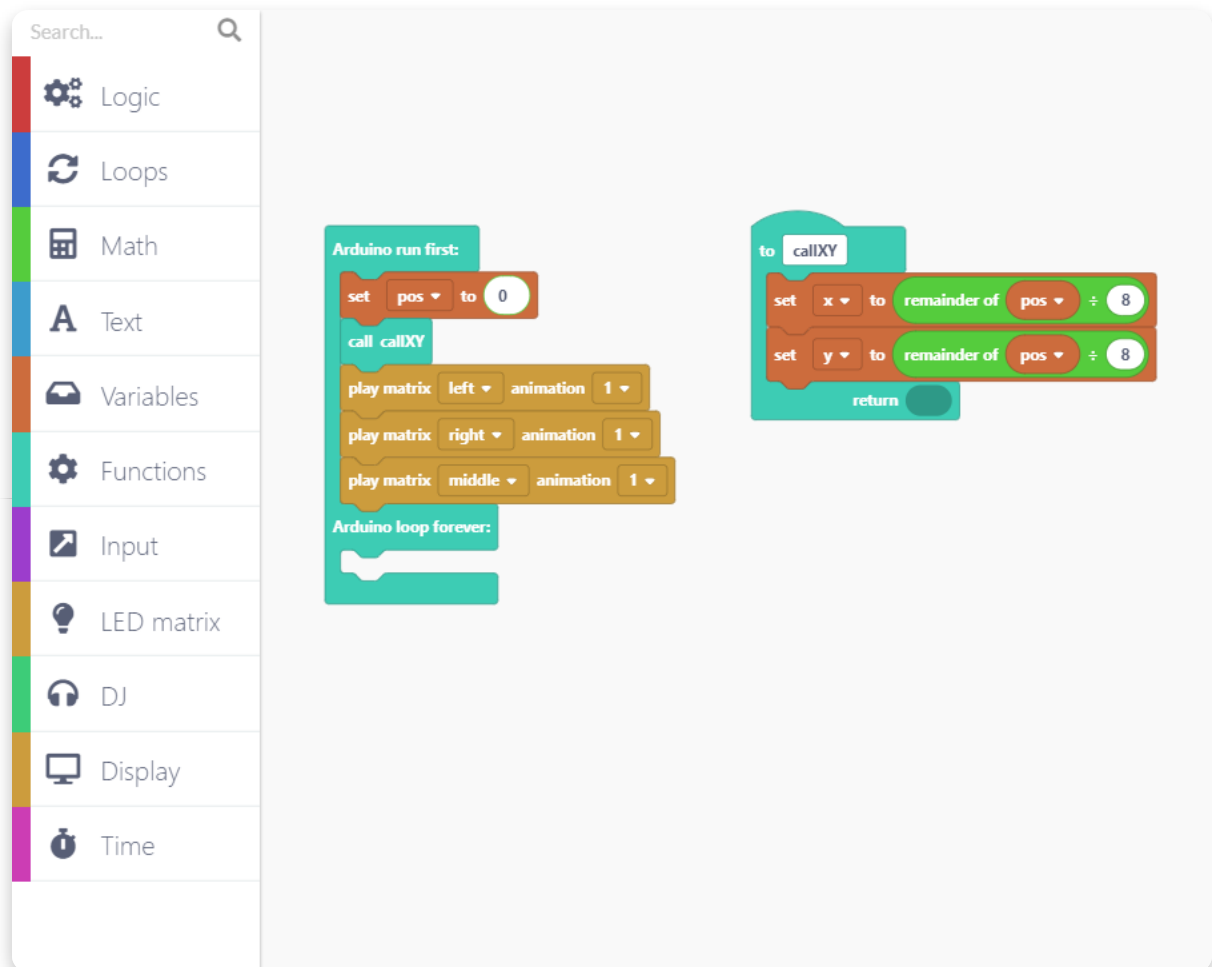


Since the LED matrixes have 8 rows, we'll use this function to turn on the light in the correct order:

- set the coordinates by using the drop-down menu: first choose x and then y
- use the "remainder of" mathematical function to divide the position by 8
- use "pos" from the "Variables" section

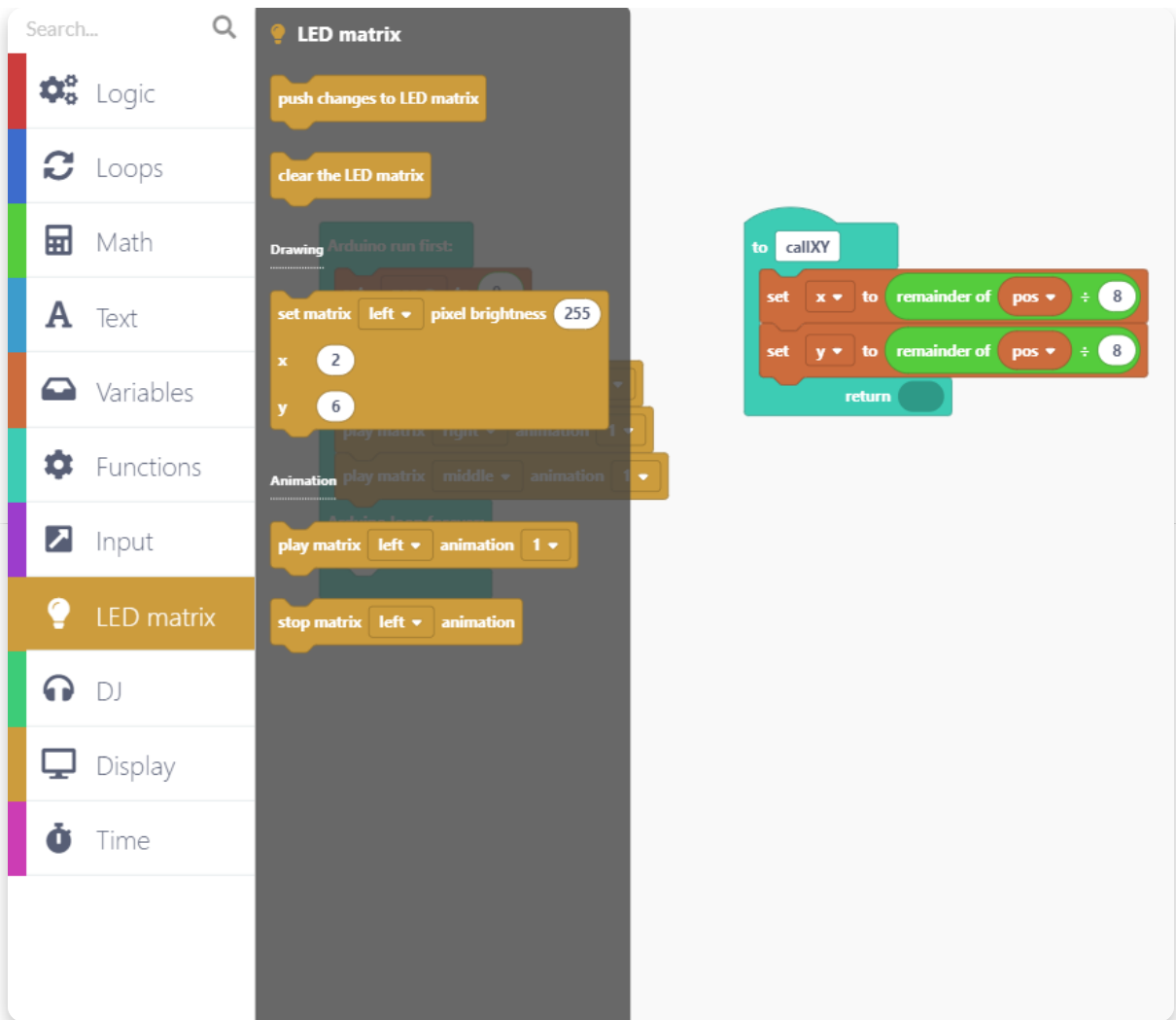


Good job! For now, we have the "Arduino run first" part of the block done.



Let's play with LED lights on a big matrix now.

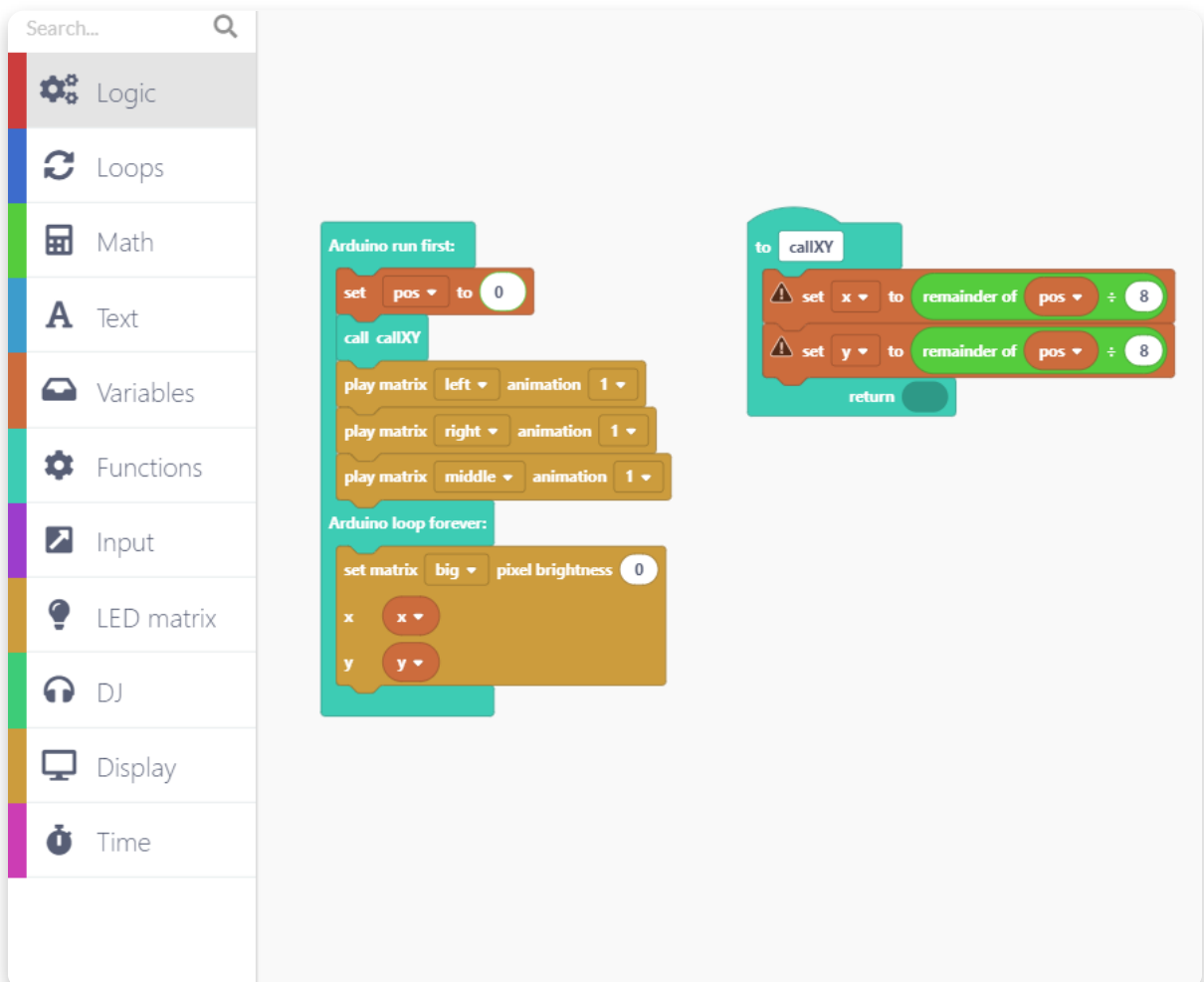
Go to the "LED matrix" section and choose the "set matrix" block.



Place the block after "Arduino loop forever".

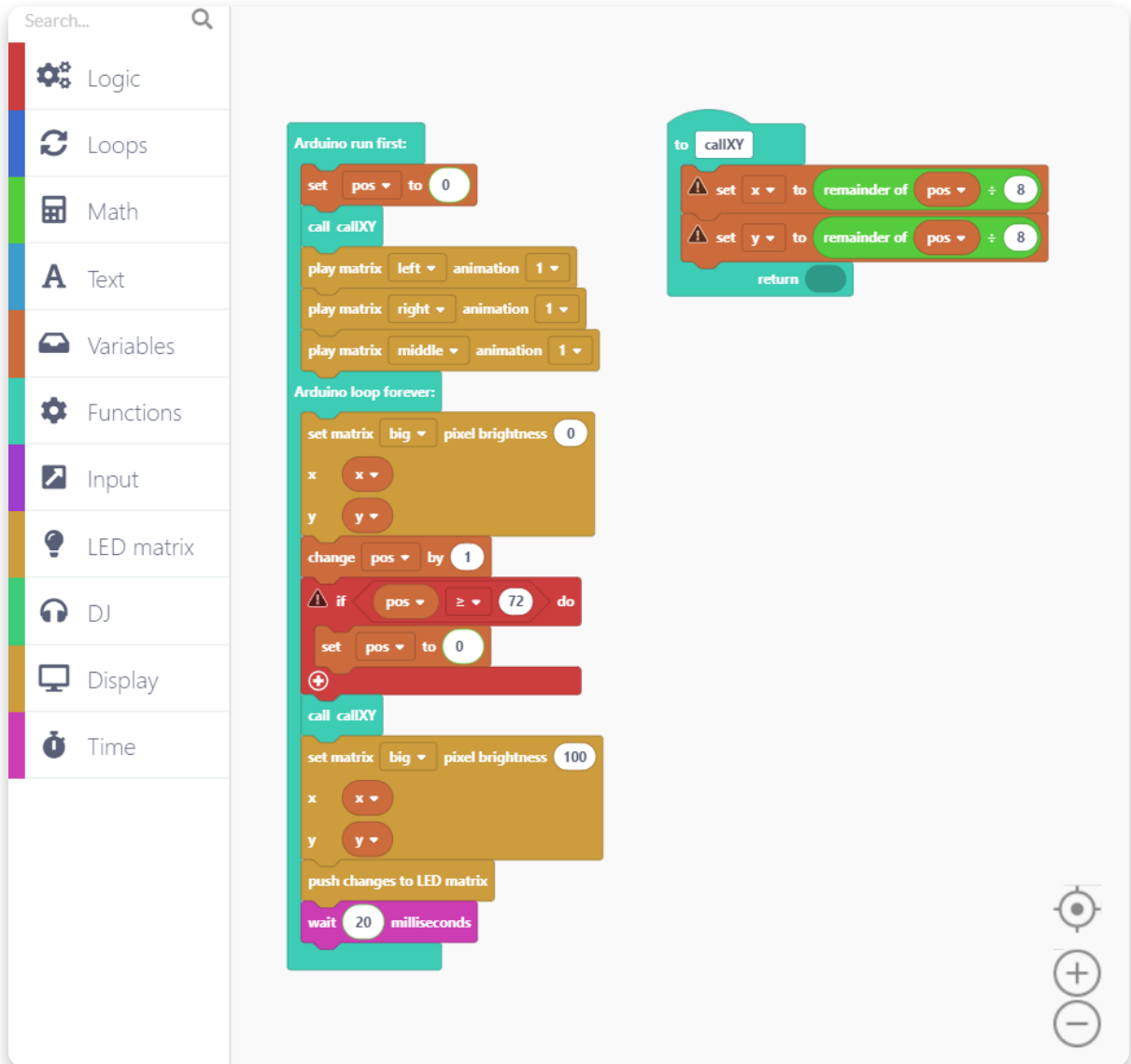
In the drop-down menu, choose the big LED matrix and set the pixel brightness to 0.

In the x and y line, place the x and y variables as in the following photo:



Now add the blocks shown in the following picture. By adding these blocks, you will change the position of the LED lights while using logical, variable, and time

functions.



Now it's time to run the project so you can see your code in action.

Click on the Run button on the top right corner of the screen, give a name to your project if you still haven't, and wait for the CircuitBlock to compile the project.

Let's code some more

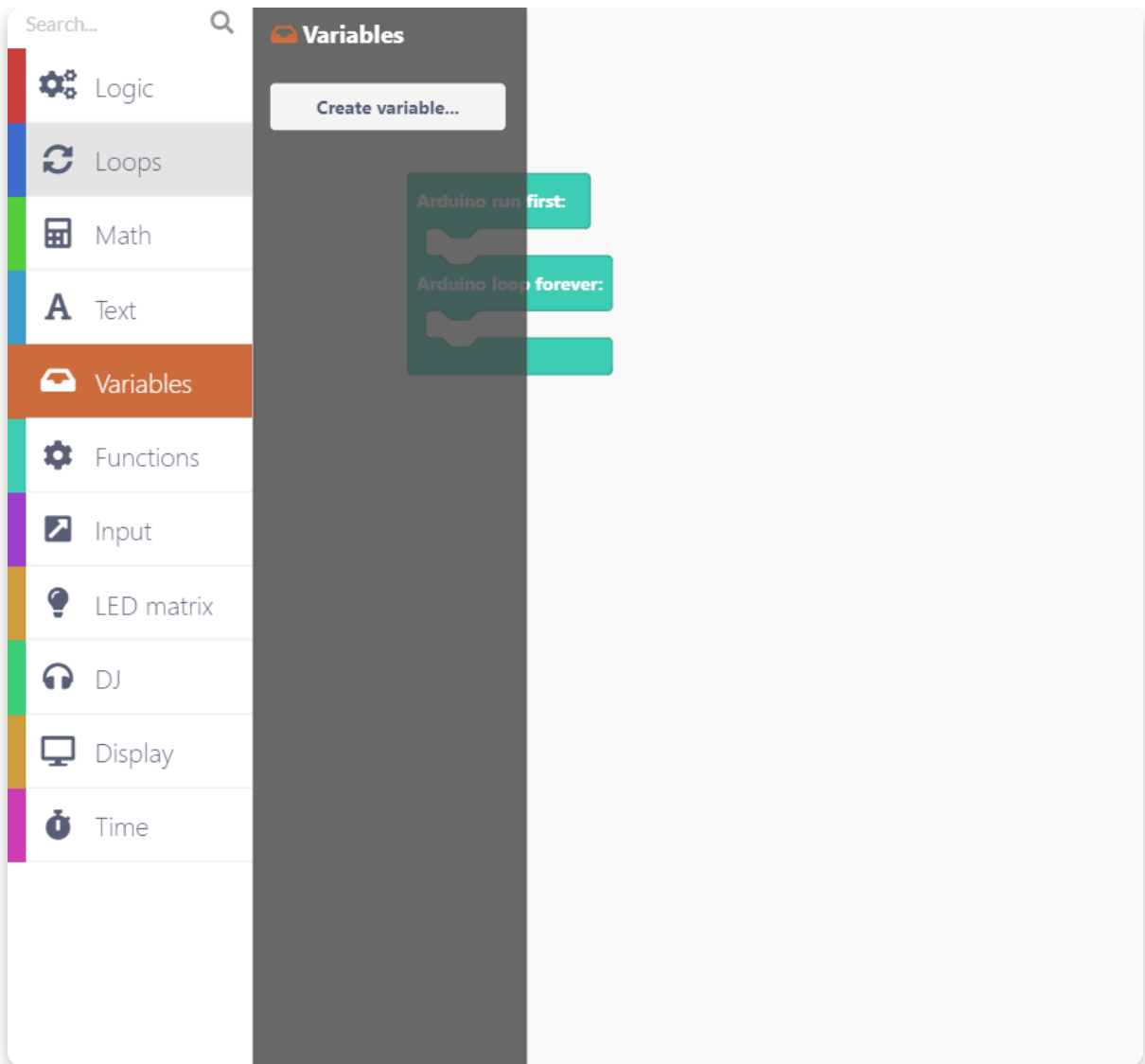
Drawing

Did you know that you can draw on Jay-D's display?

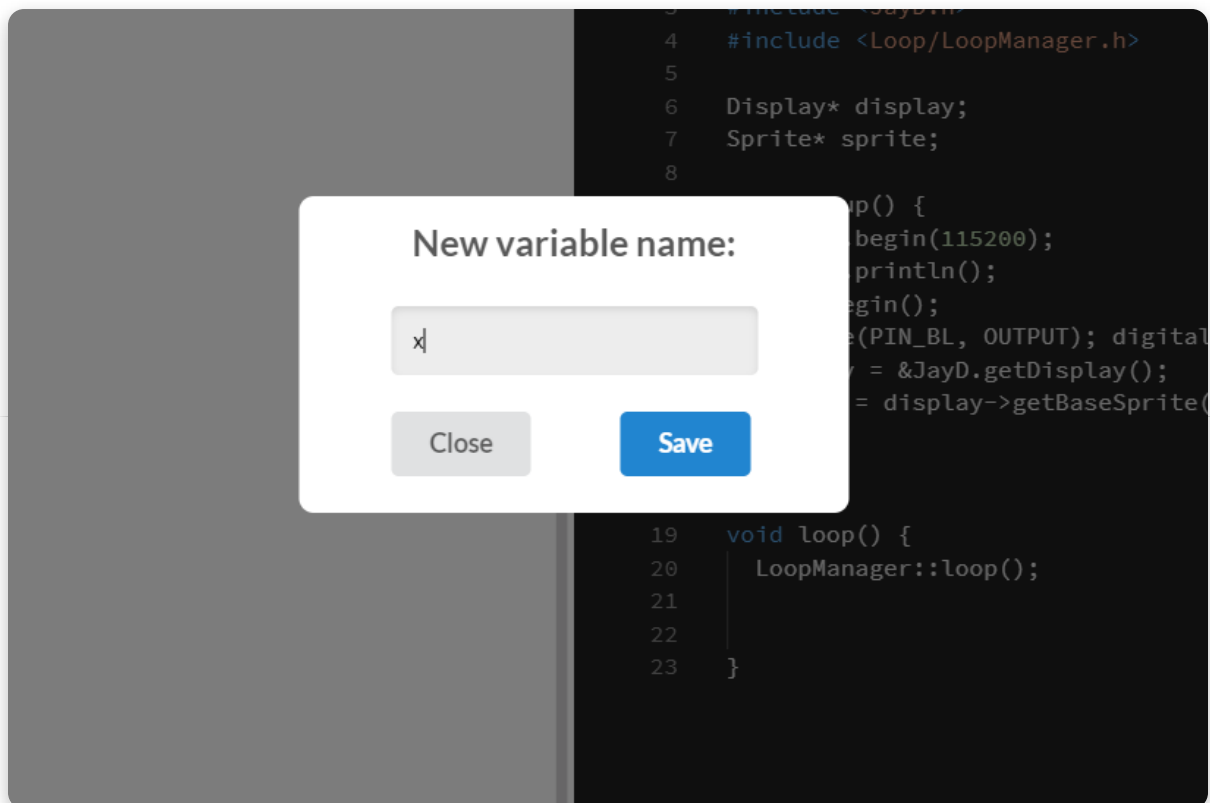
In this step of the coding guide, we'll show you how to set up everything you need for getting creative with Jay-D's drawing pen.

Open a new Jay-D block project.

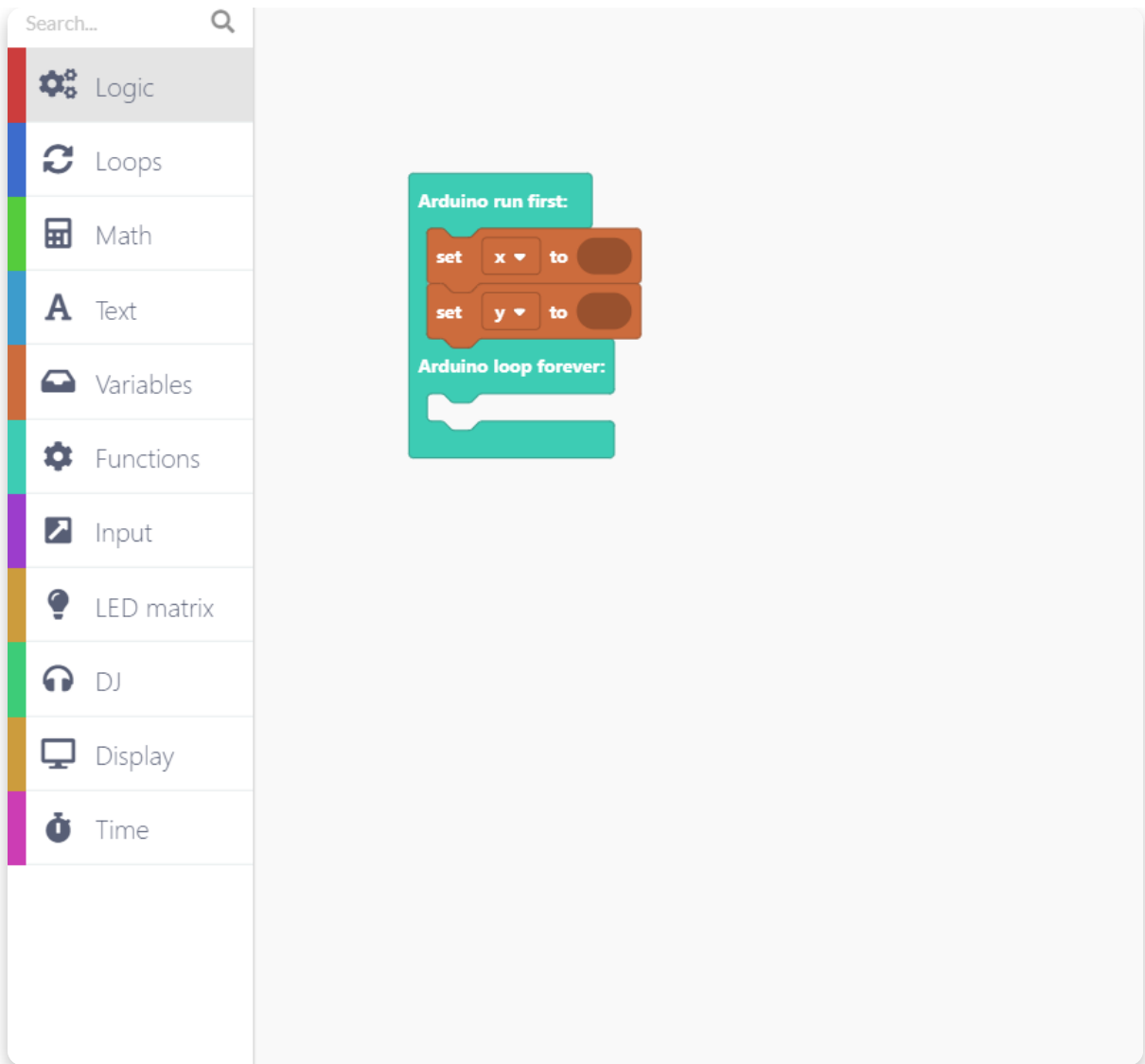
Now create two new variables called x and y. **X means the coordinate on the x-axis while y means the coordinate on the y-axis.**



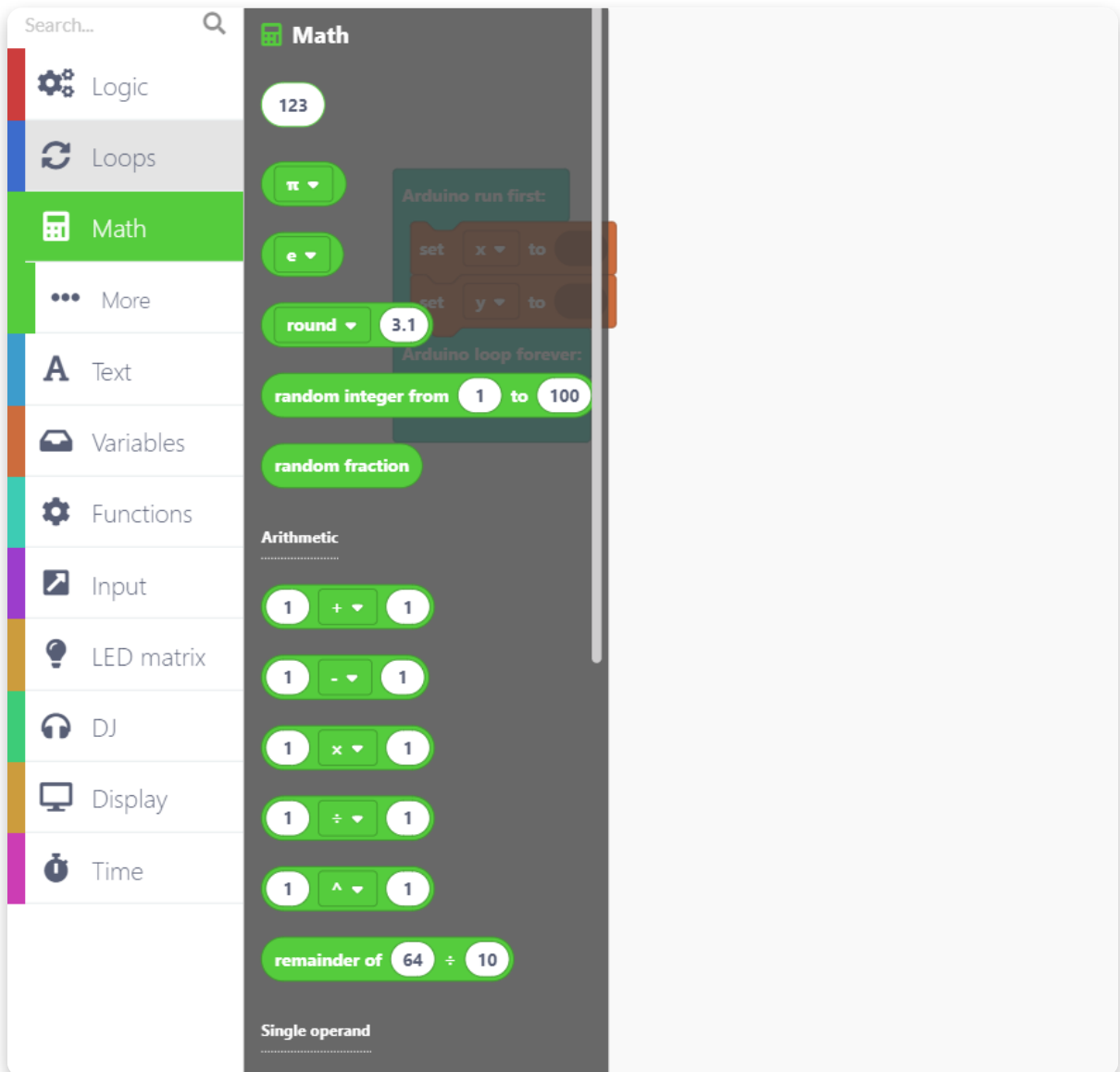
Create two variables: x and y:



Now drag and drop the "set y/x to" block twice here:

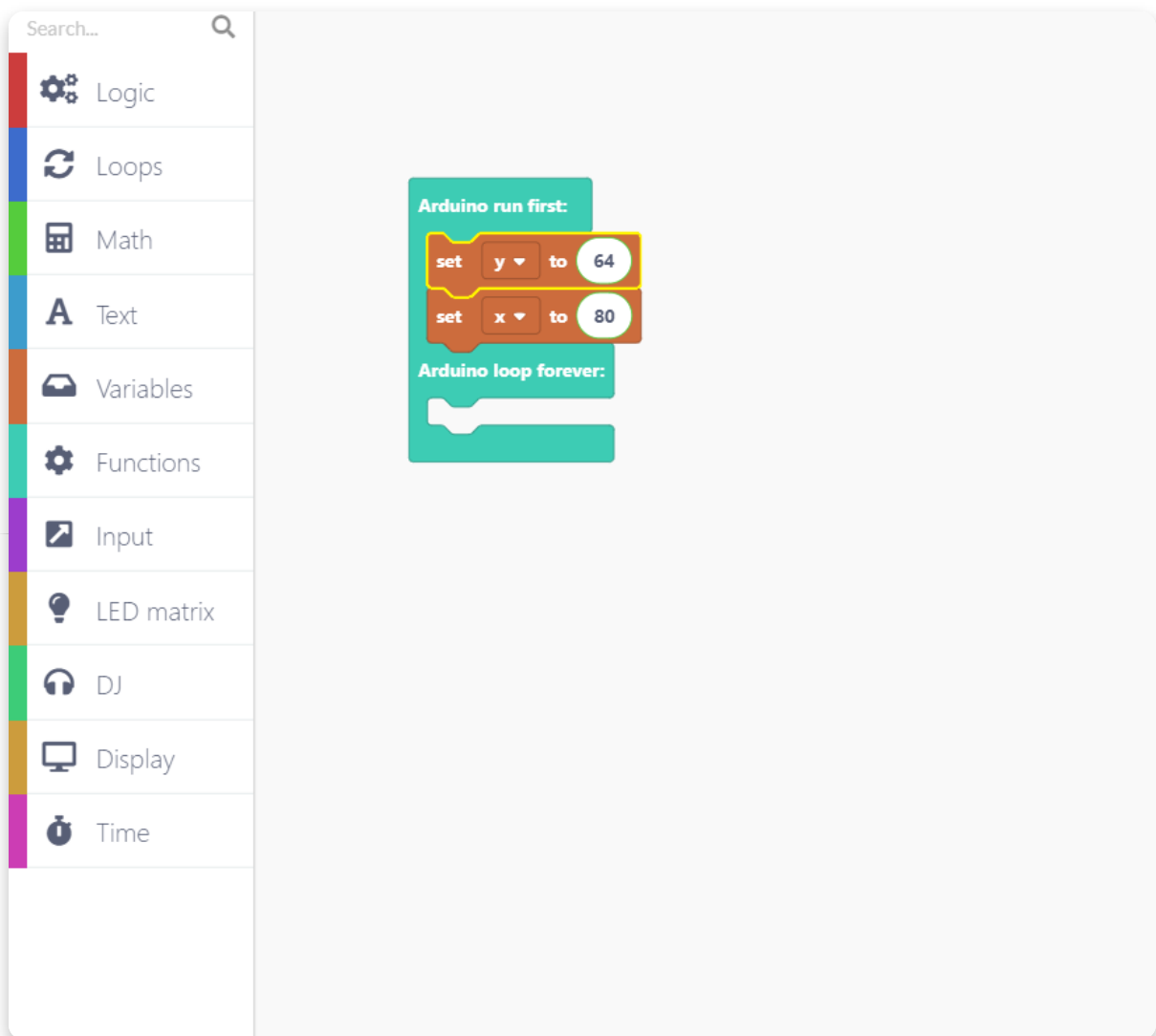


Find the "123" block in the "Math" section, and drop it the previous block that will set the coordinates for the variables.

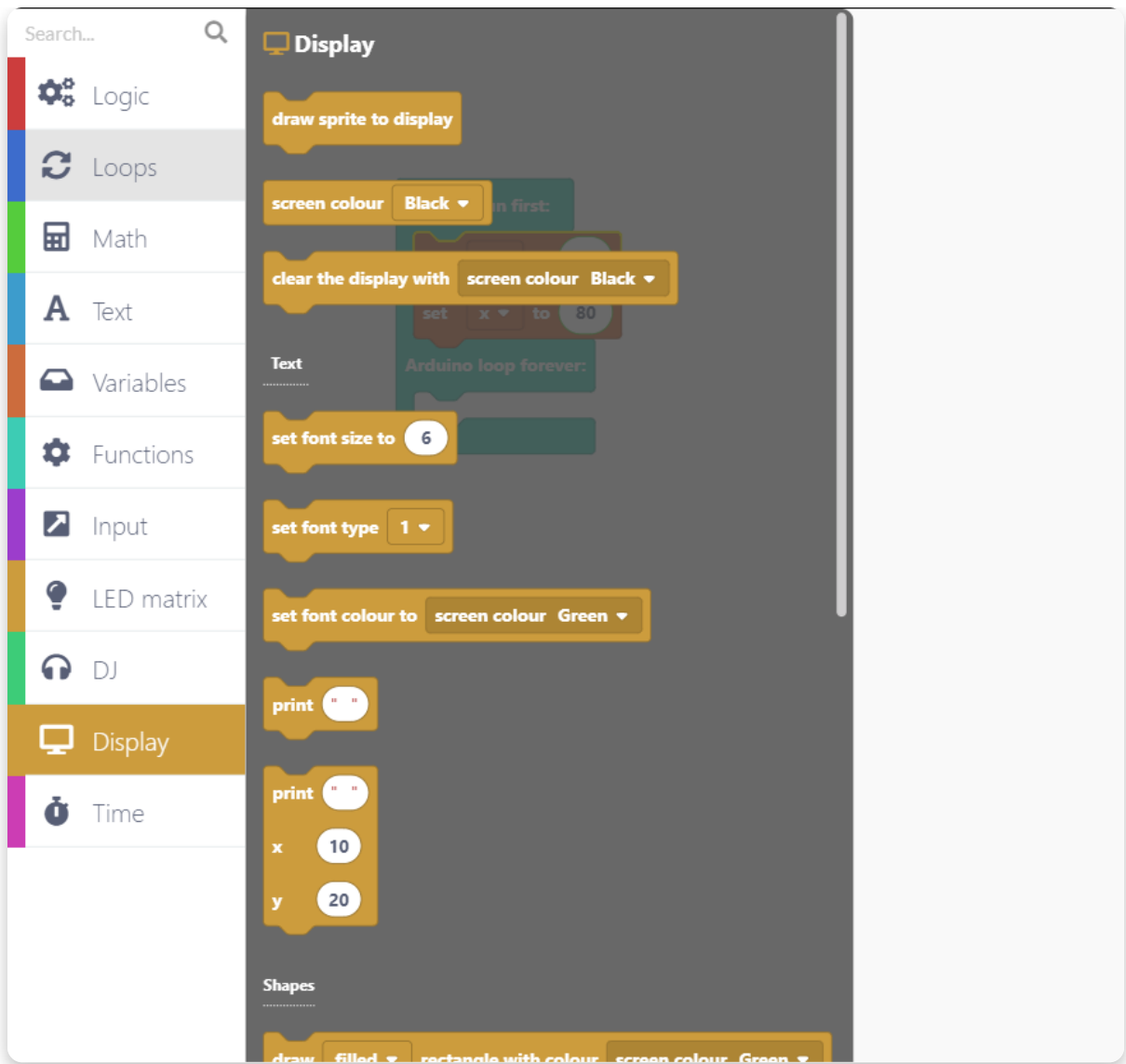


Place the "123" block in both "set x/y to" blocks.

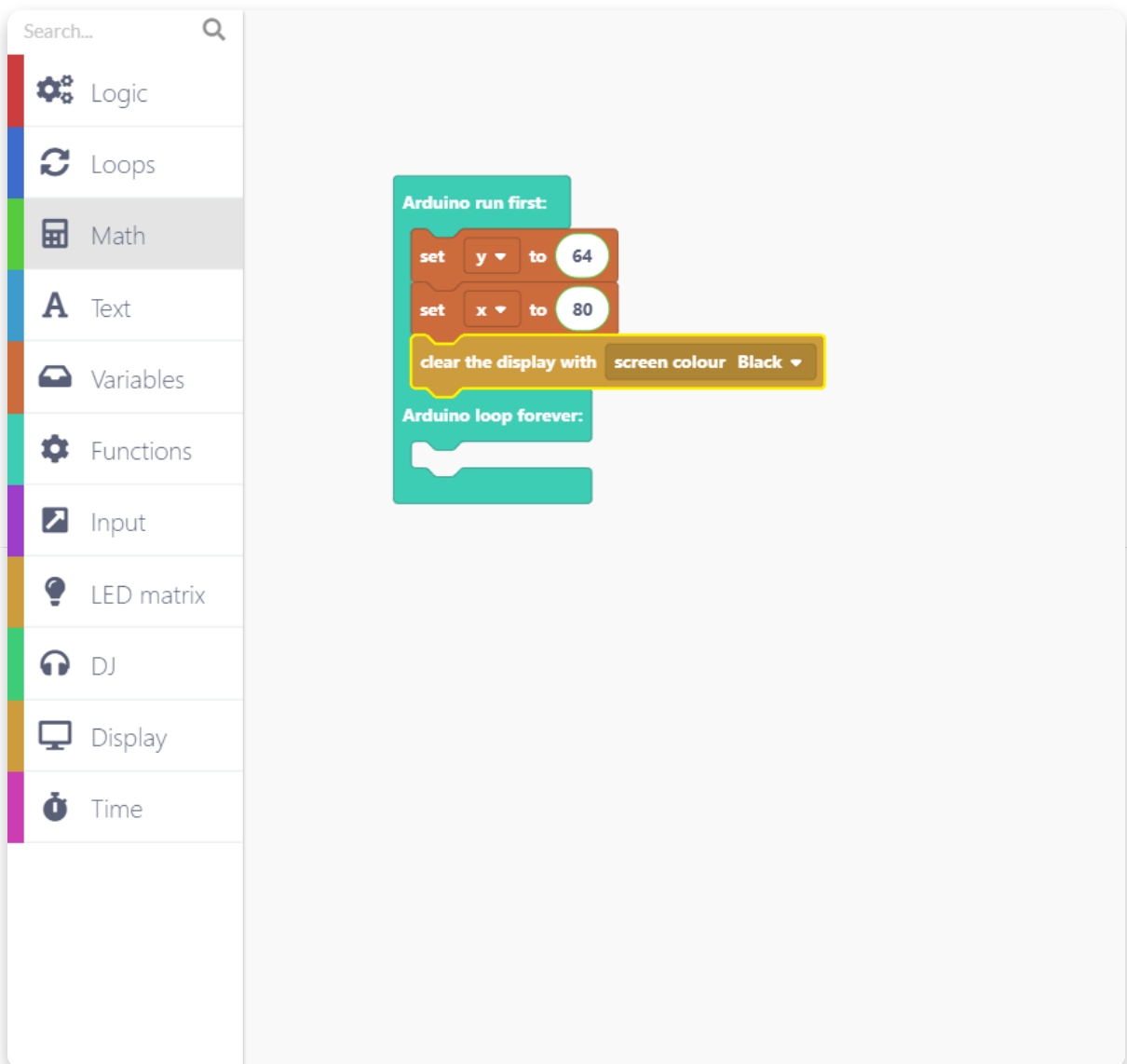
Since the display's dimensions are 128x160, let's set the starting dot in the center of the display. Set the y variable to 64 and the x variable to 80.



To set the background color to black so you can draw on it, go to the "Display" section and find a block that says "clear the display with screen color".



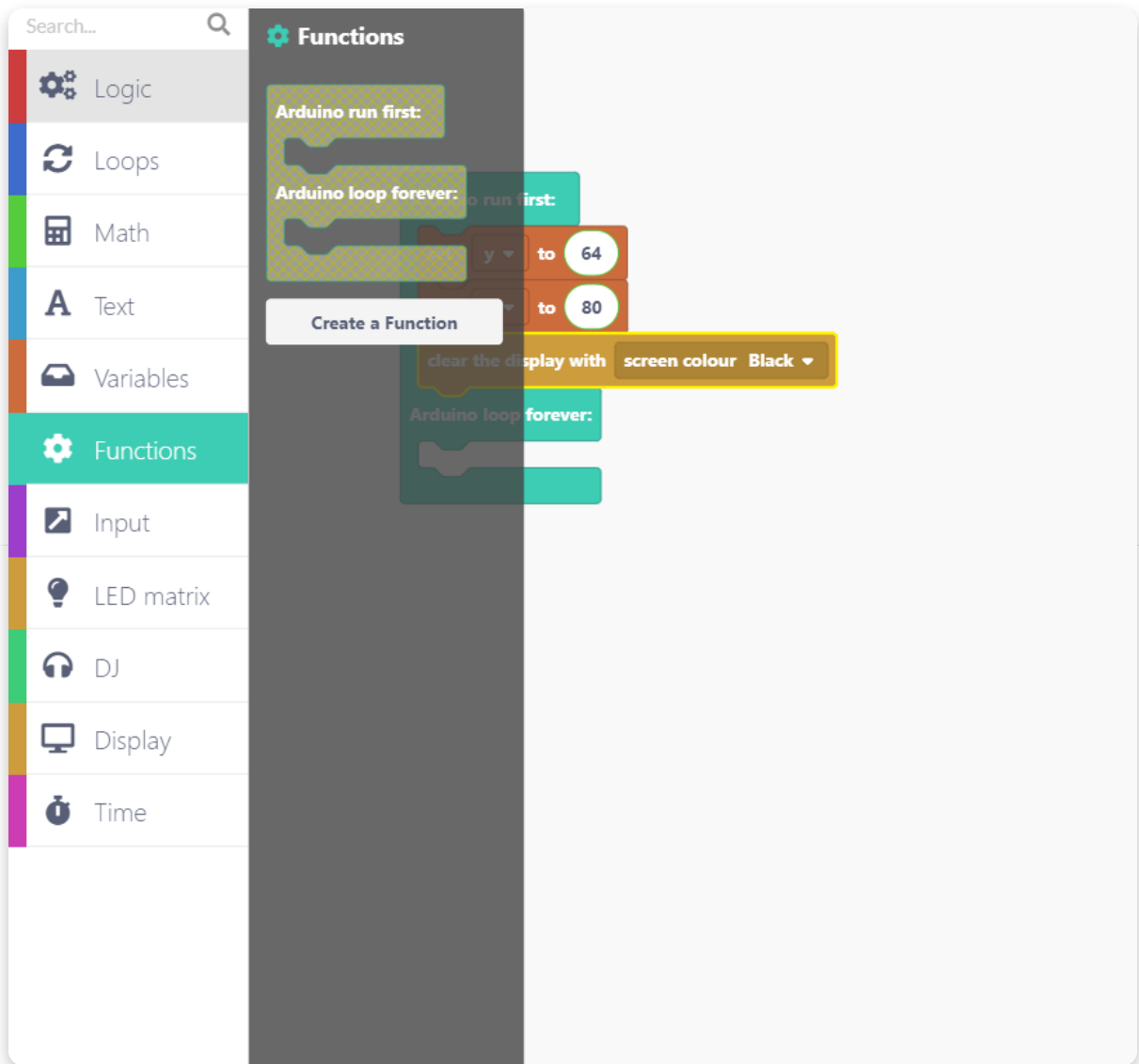
Drag and drop that block here:



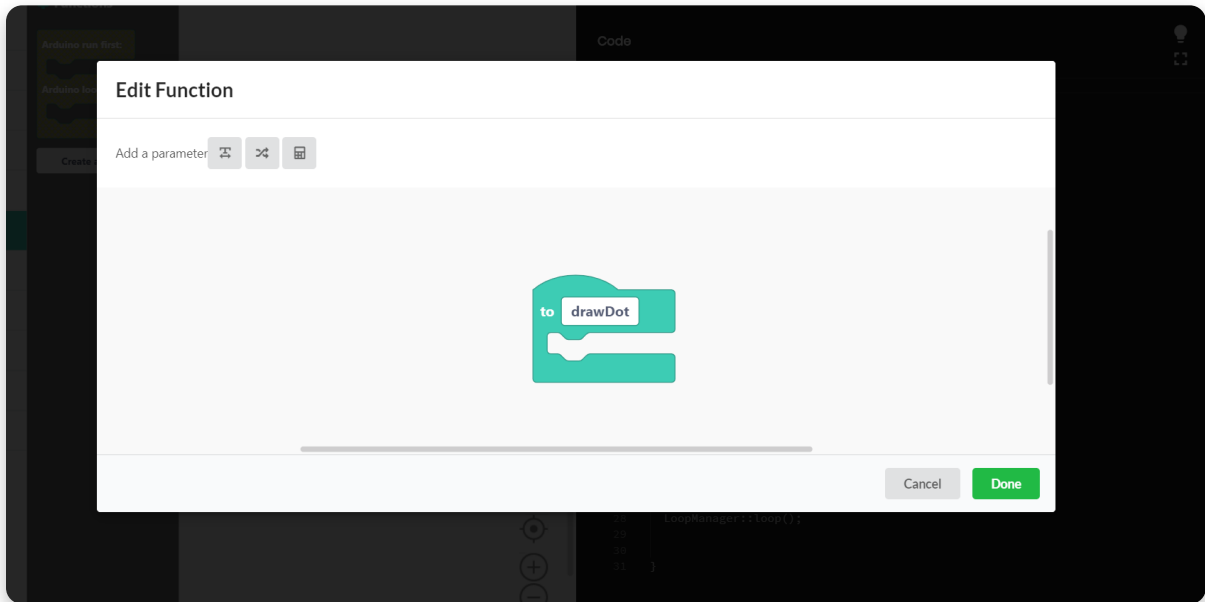
Now when we've set the starting point of our pen and the background to black,

Let's create a function that will draw the dots with a color of your choice.

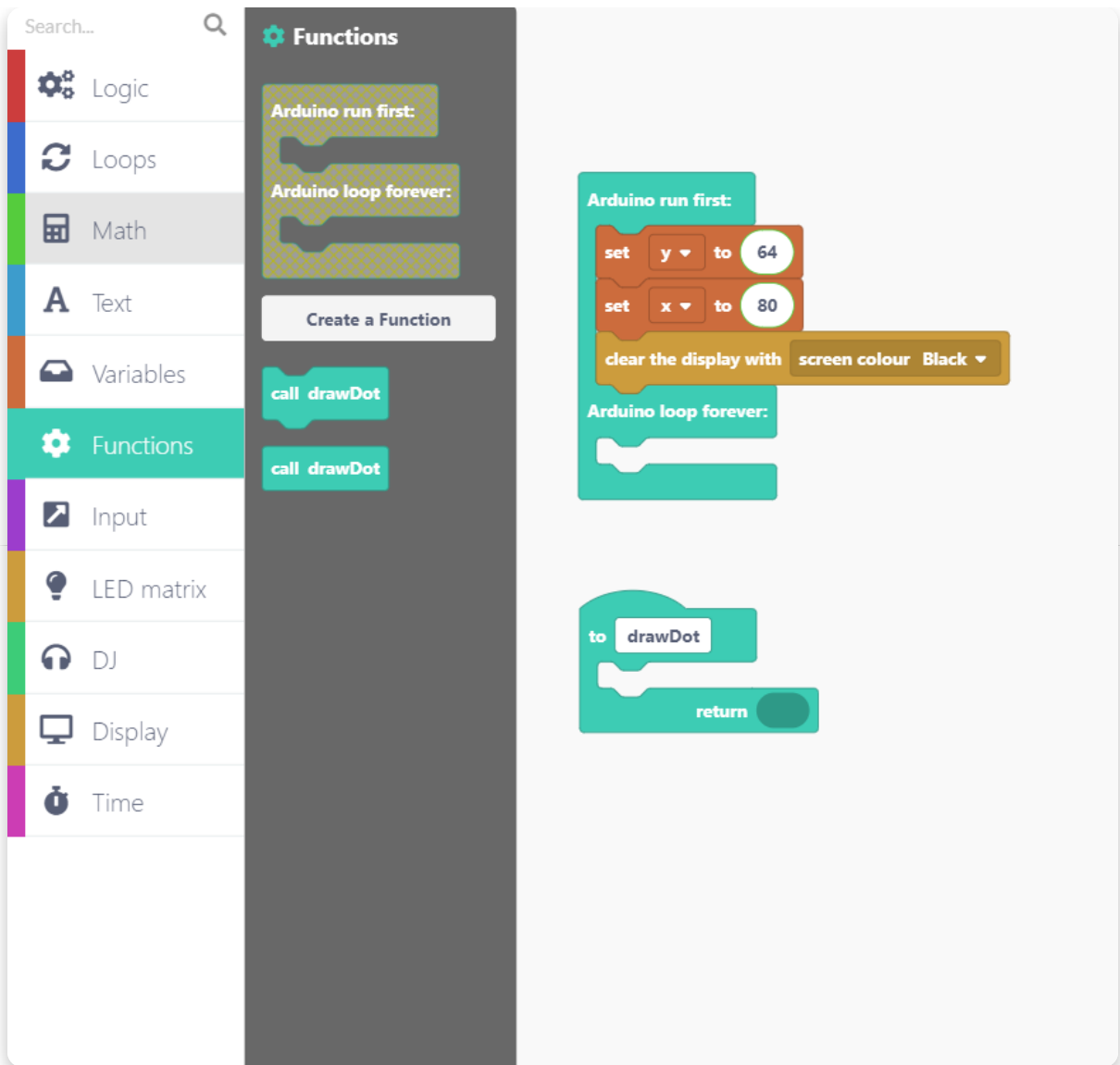
Find a "Create a Function" option in the "Functions" section.



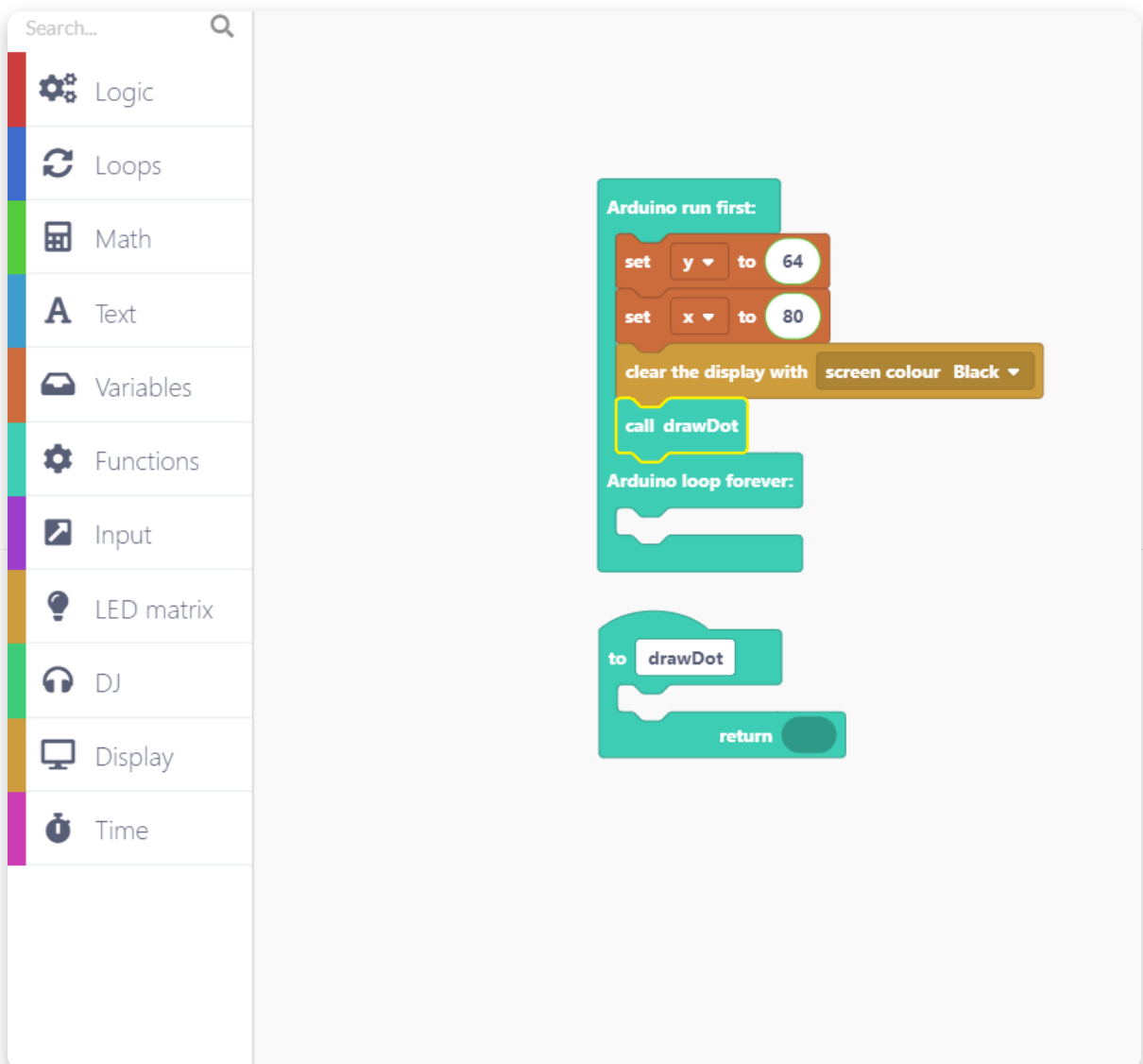
Let's call this function "drawDot":



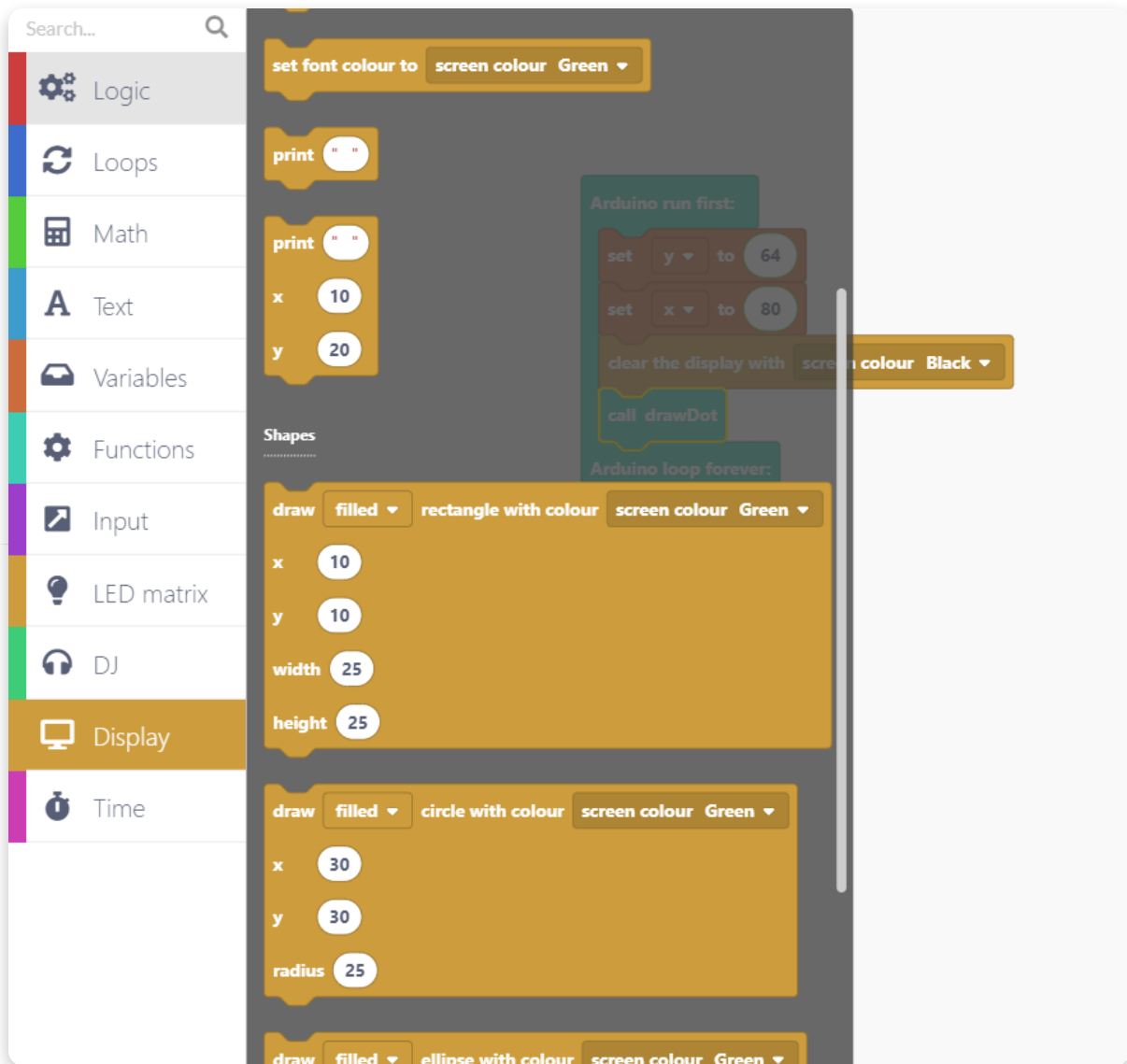
When you name the function, it will appear on the screen and a few more options will appear in the "Functions" section.



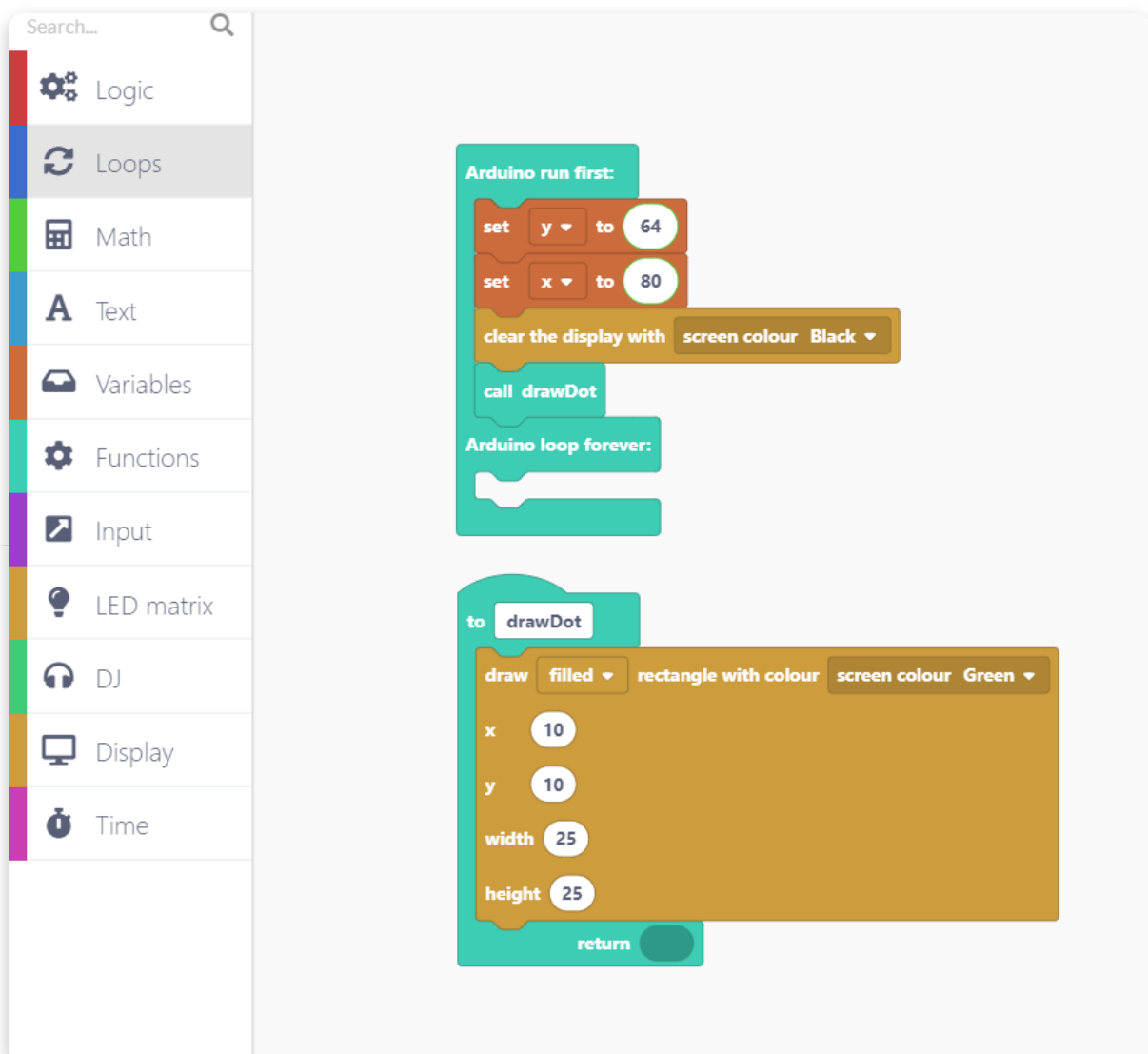
Drag and drop the "call drawDot" function and place it after the clear the display one:



In the following step, we have to define what does this drawDot function does.
Go to the "Display" section and find a shape that draws a rectangle with a color.



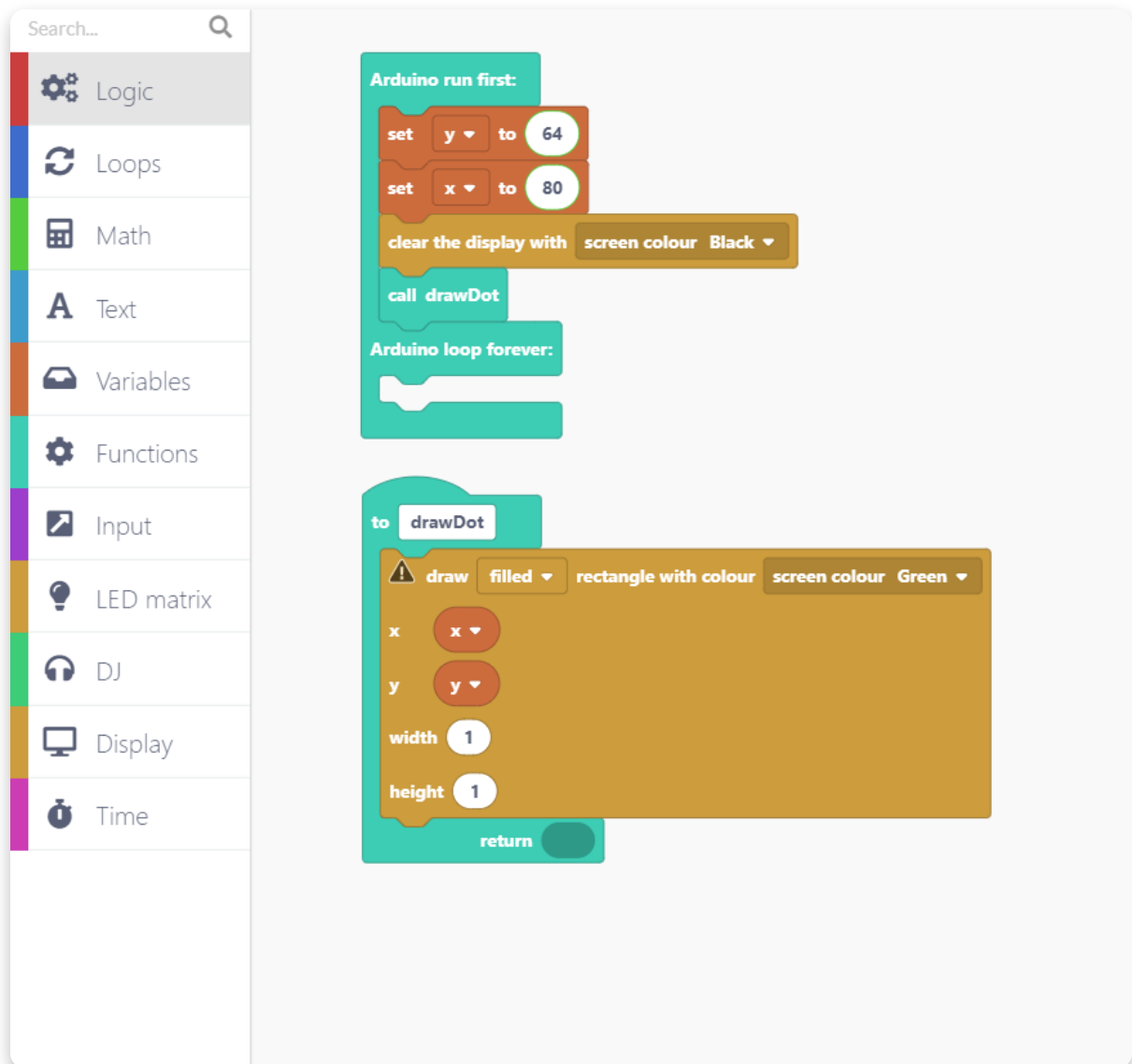
Drag and drop this block in the drawDot function.



When you place the block into the function, do the following:

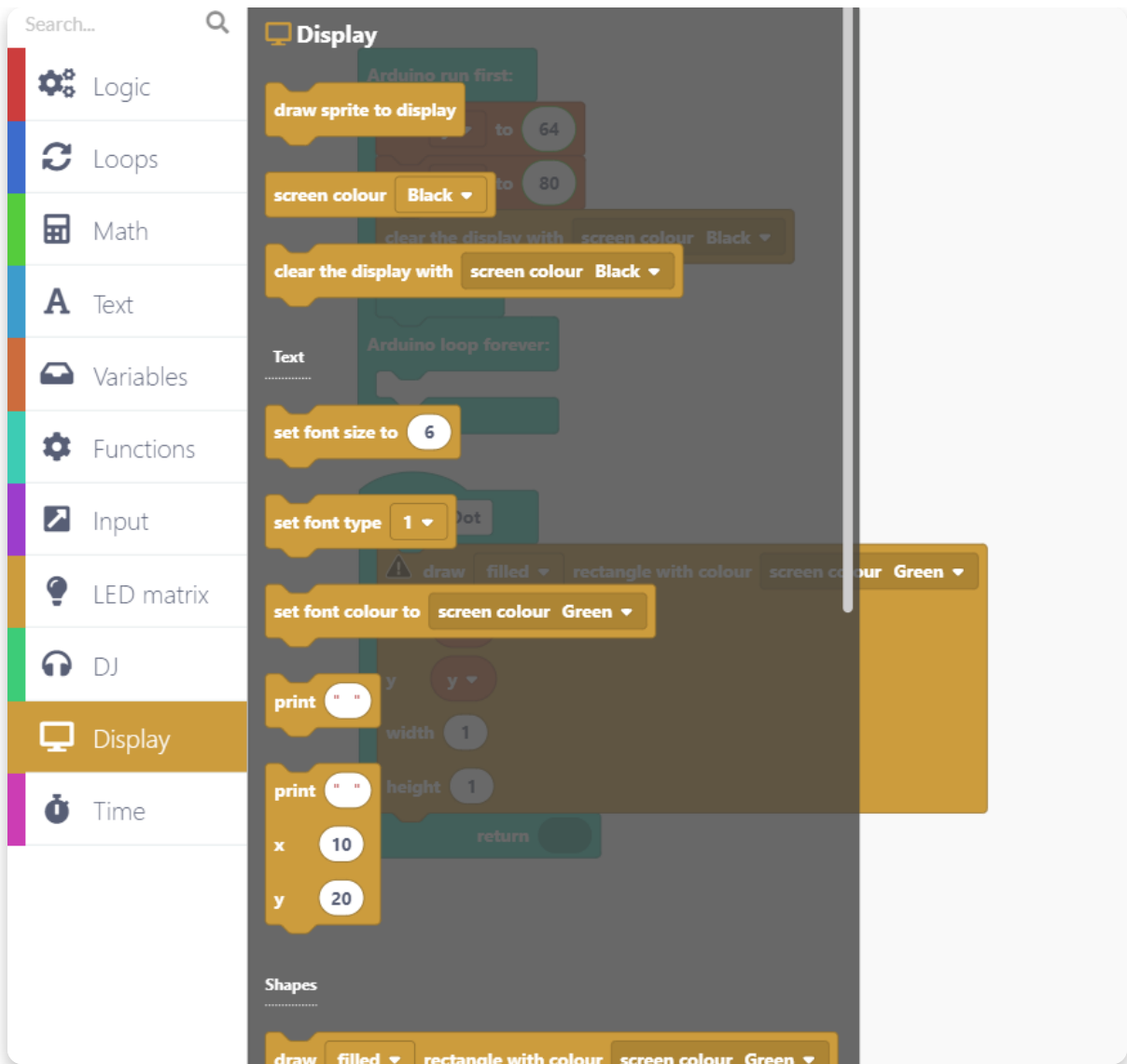
- Set the "filled" option in the first drop-down menu and choose a color you'd like. We'll keep the color green for now.
- Place the x and y variables (from the "Variables" section) into white boxes next to x and y.
- Set the width and height to 1 since we'll be drawing a small dot.

Take a look:

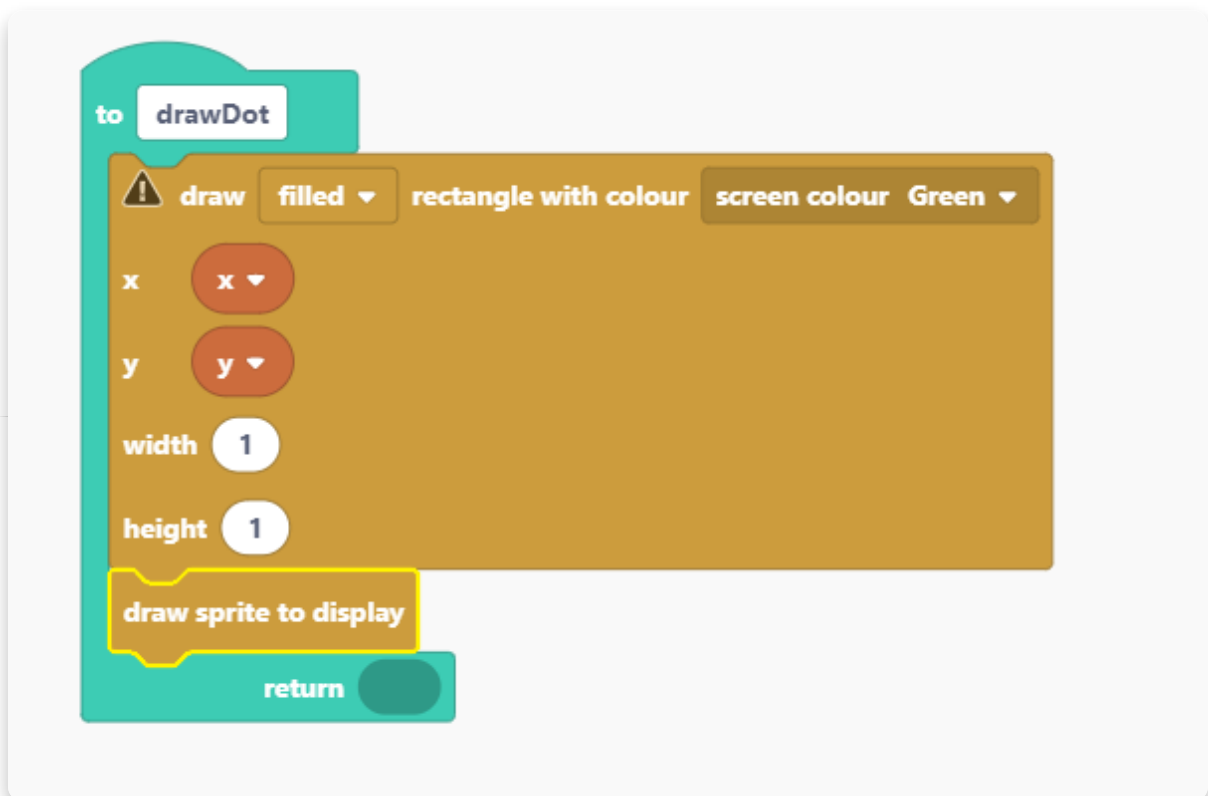


To get those functions working, we'll use another block from the "Display" section that will apply the "drawDot" function.

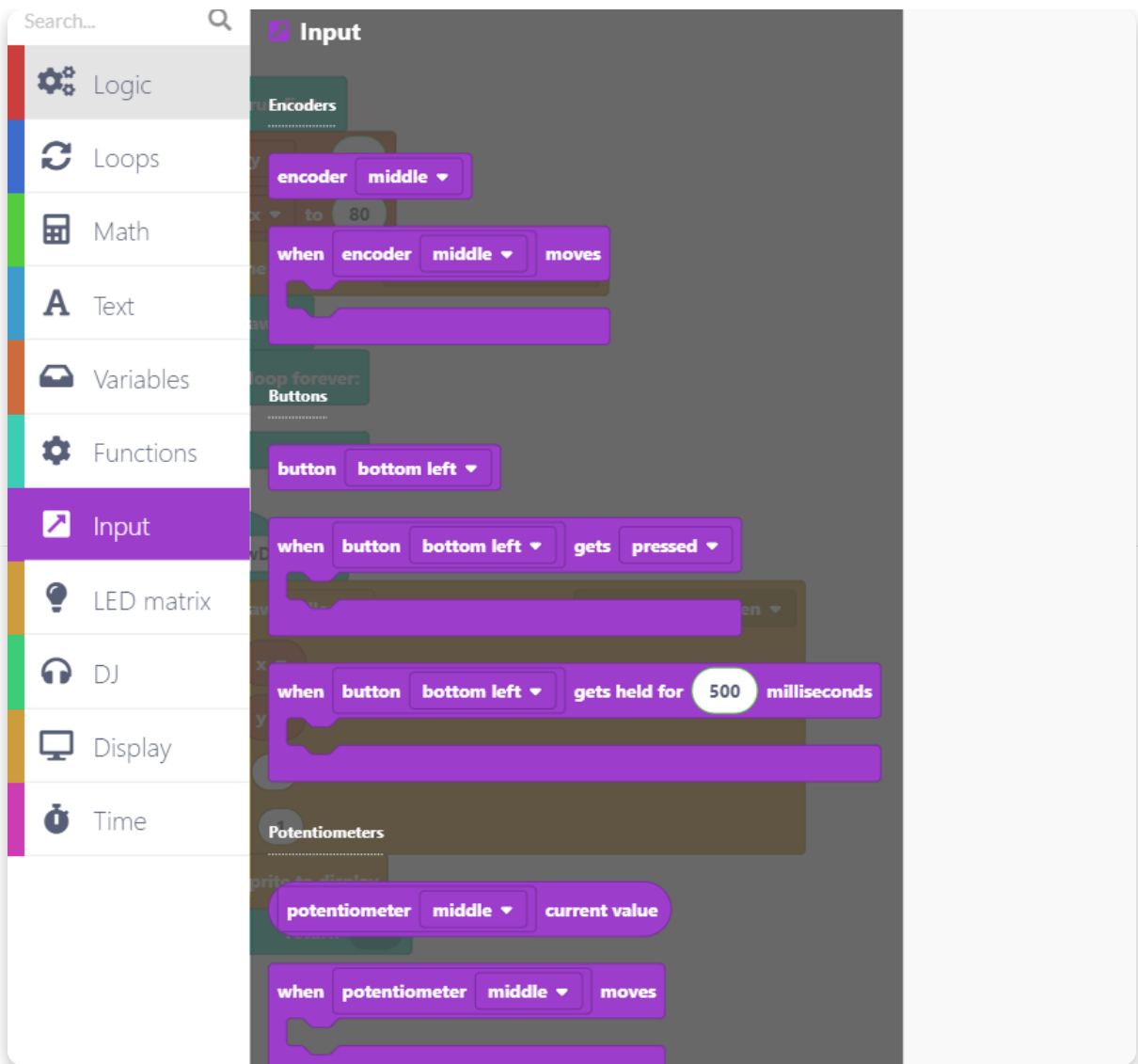
Open the "Display" section and find the "draw sprite to display" block.



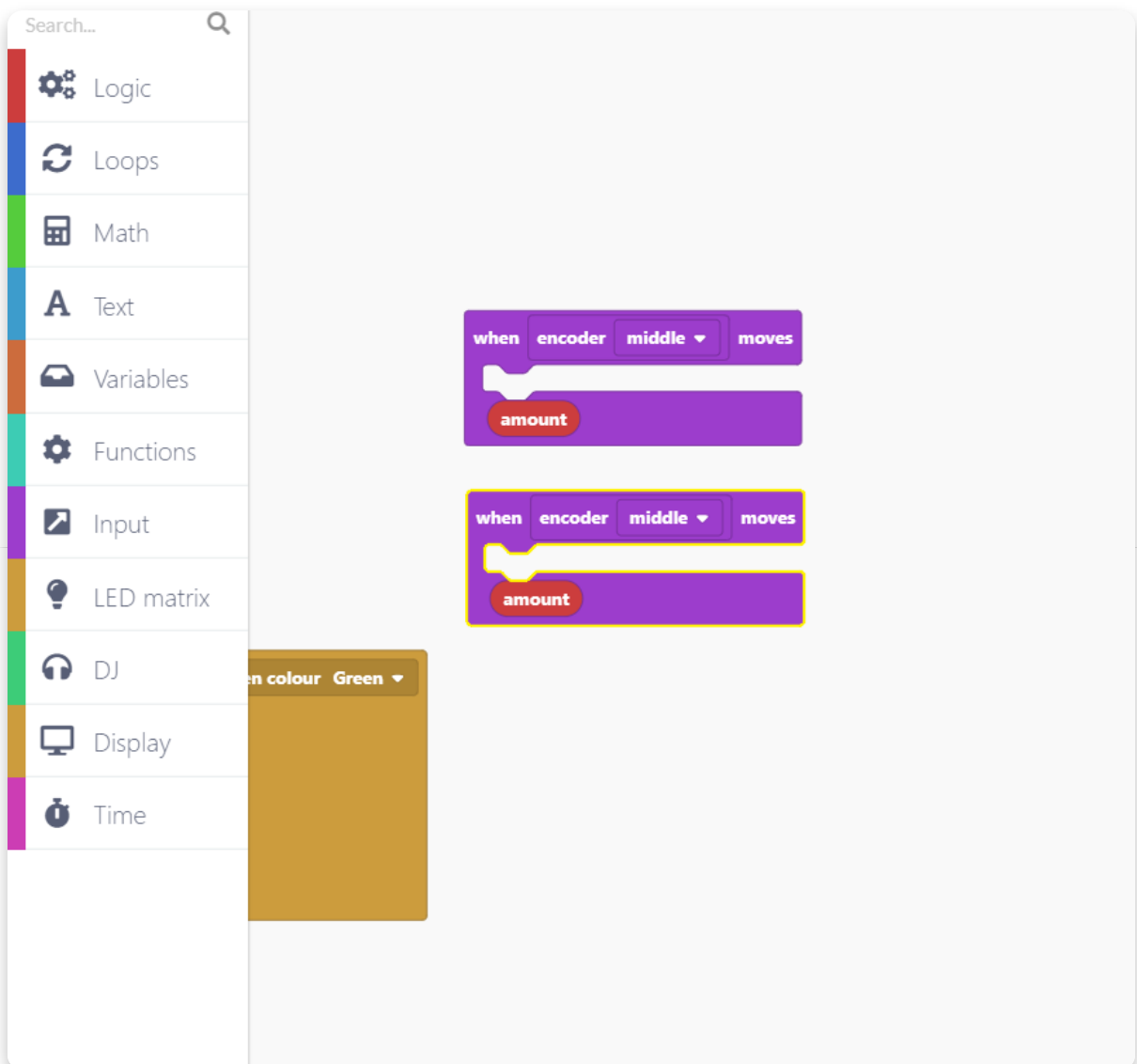
Drag and drop it at the end of the "drawDot" function:



Almost everything is ready for drawing, but there's one more thing that we need. We need a certain input that will serve as a pen. In this case, let's use the encoders. One will be for drawing on the x-axis and the other one for the y-axis. For this, we are using the "Input" section and two identical blocks that say "when encoder middle moves".

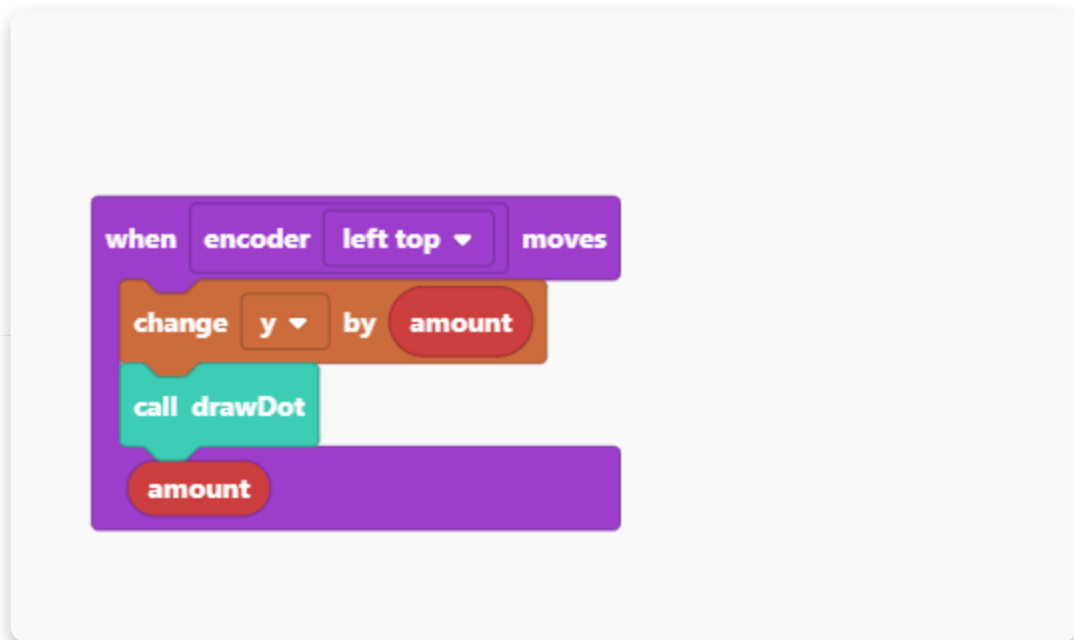


Drag and drop this block twice:



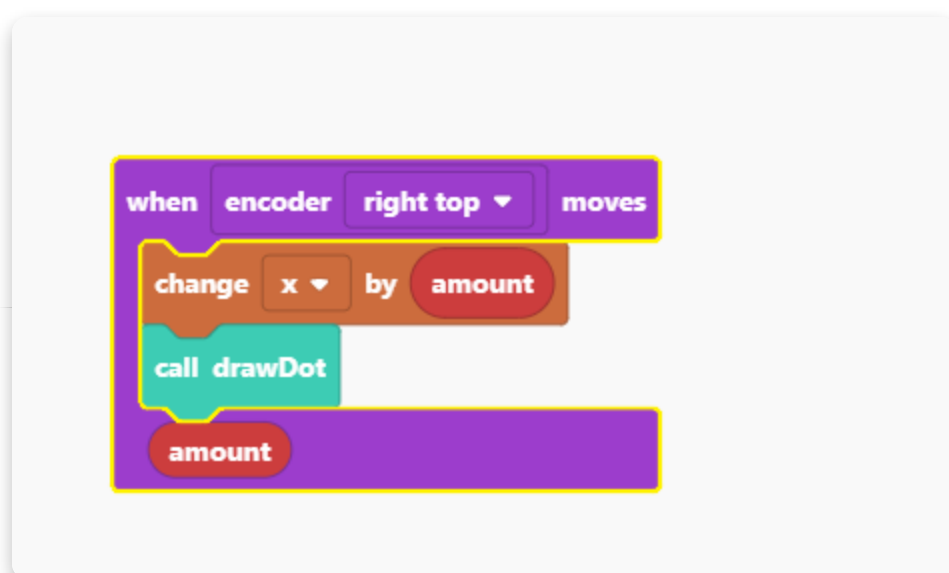
For moving along the y-axis, let's select the following:

- Choose the left top encoder in the drop-down menu.
- Go to the "Variables" section and drag and drop the "change" block into the function. Select y in the drop-down menu.
- Drag the "amount" value from the bottom of the function to the "change" block.
- Go to the "Functions" section and place the "call drawDot" block at the end of this block.



For moving along the x-axis, let's select the following:

- Choose the right top encoder in the drop-down menu.
- Go to the "Variables" section and drag and drop the "change" block into the function. Select x in the drop-down menu.
- Drag the "amount" value from the bottom of the function to the "change" block.
- Go to the "Functions" section and place the "call drawDot" block at the end of this block.

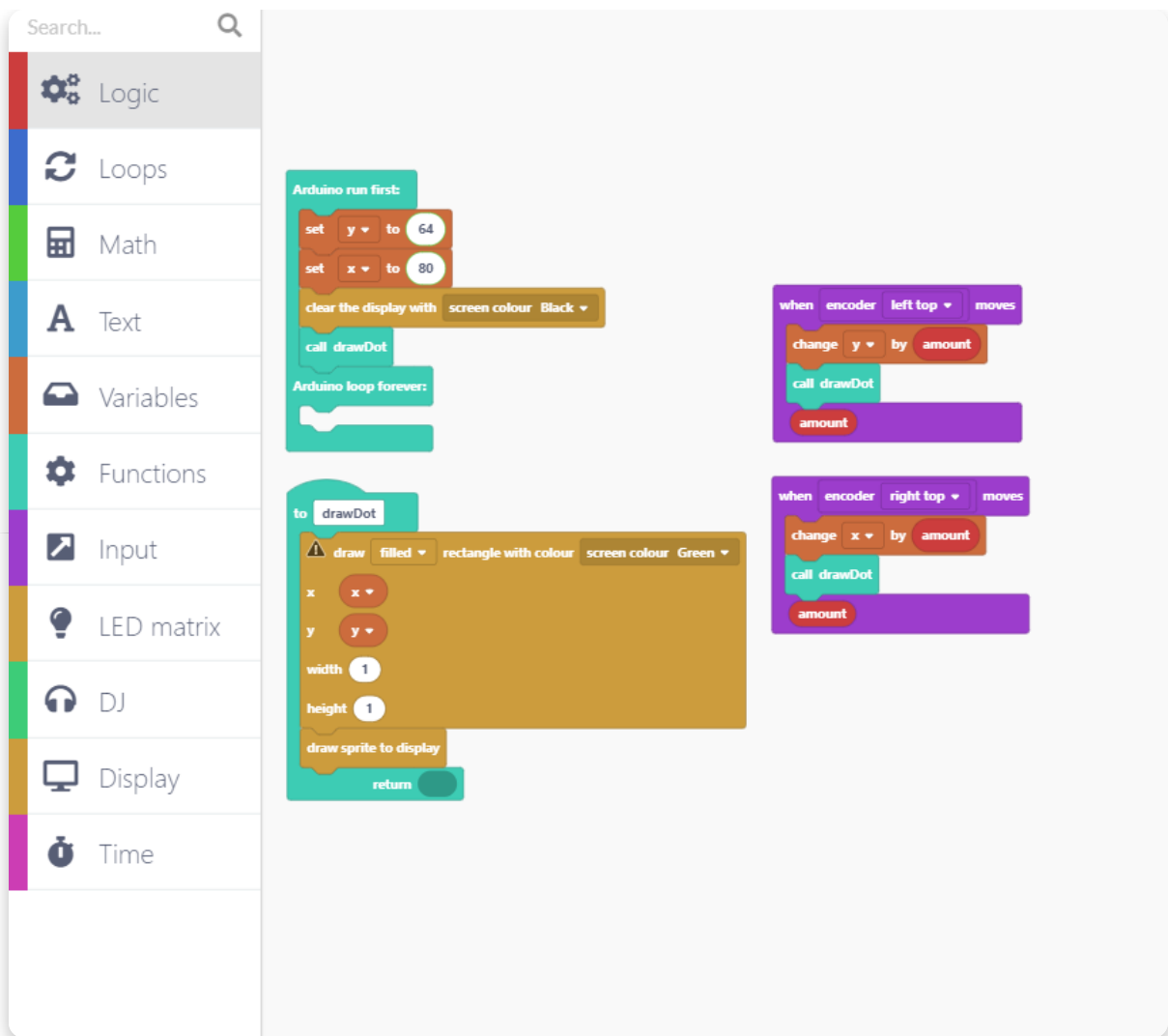


And that's it. Your Jay-D is a blank canvas now.

You just have to Save the project, give it a name and then Run it.

Running the project might take some time since CircuitBlock first needs to compile the core libraries for the newly-made sketch.

This is how the project looks at the end.



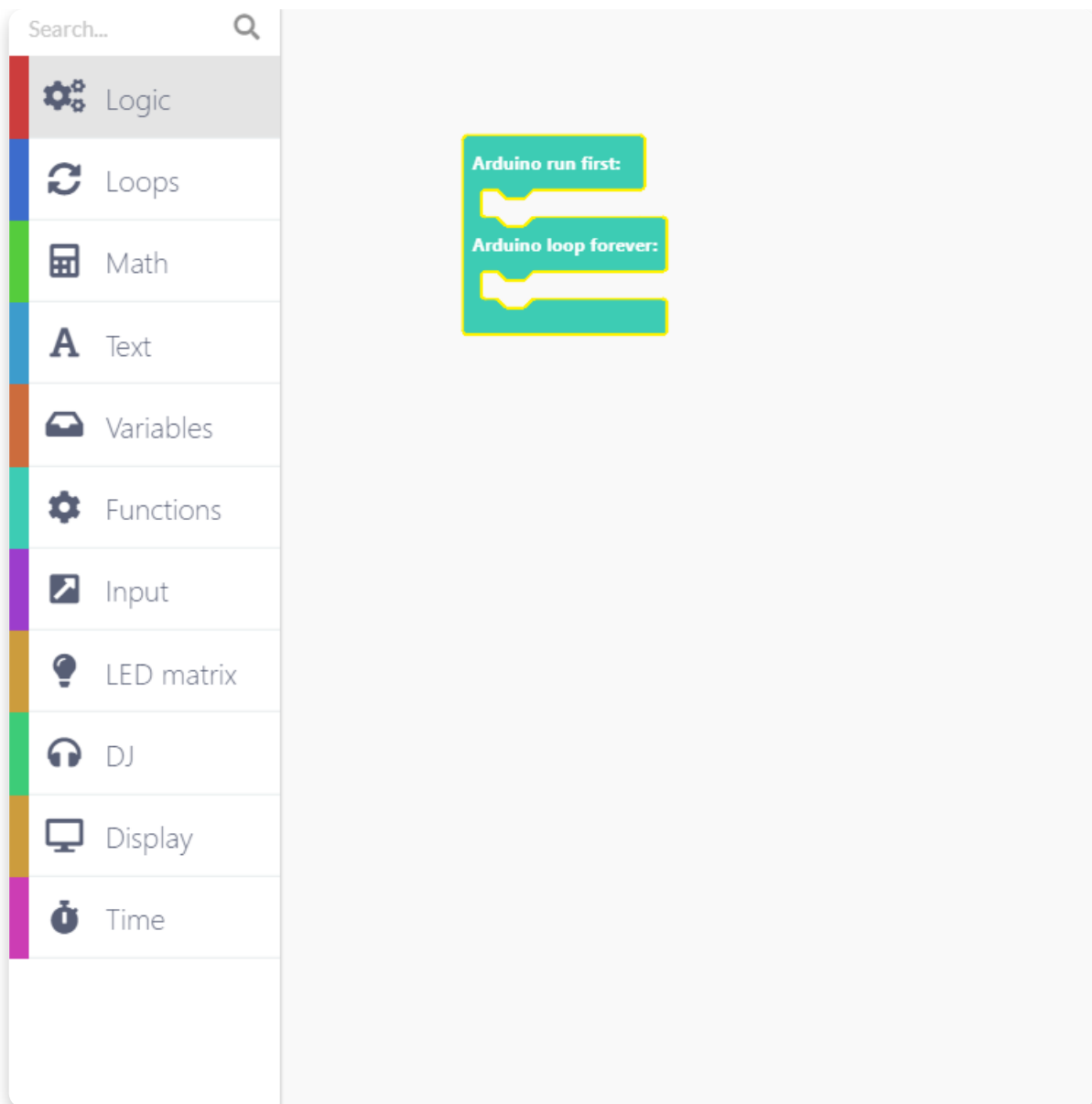
Remixing

Welcome to the last step in this guide, where we'll show you one more sketch example you can try out in CircuitBlocks.

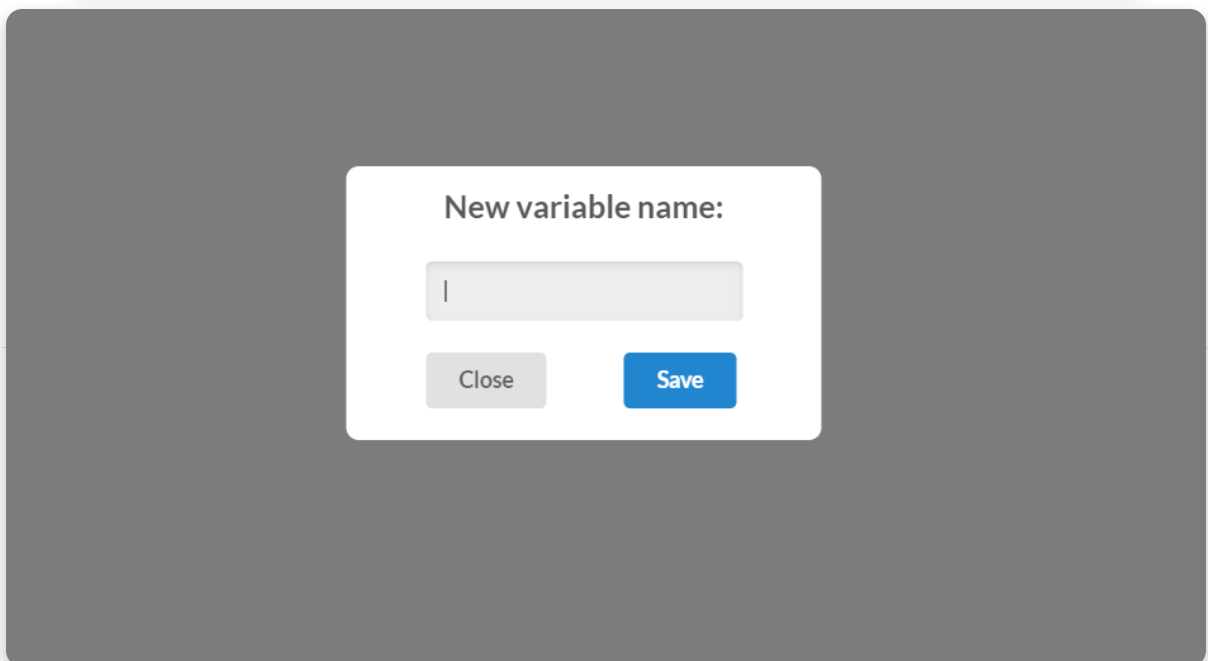
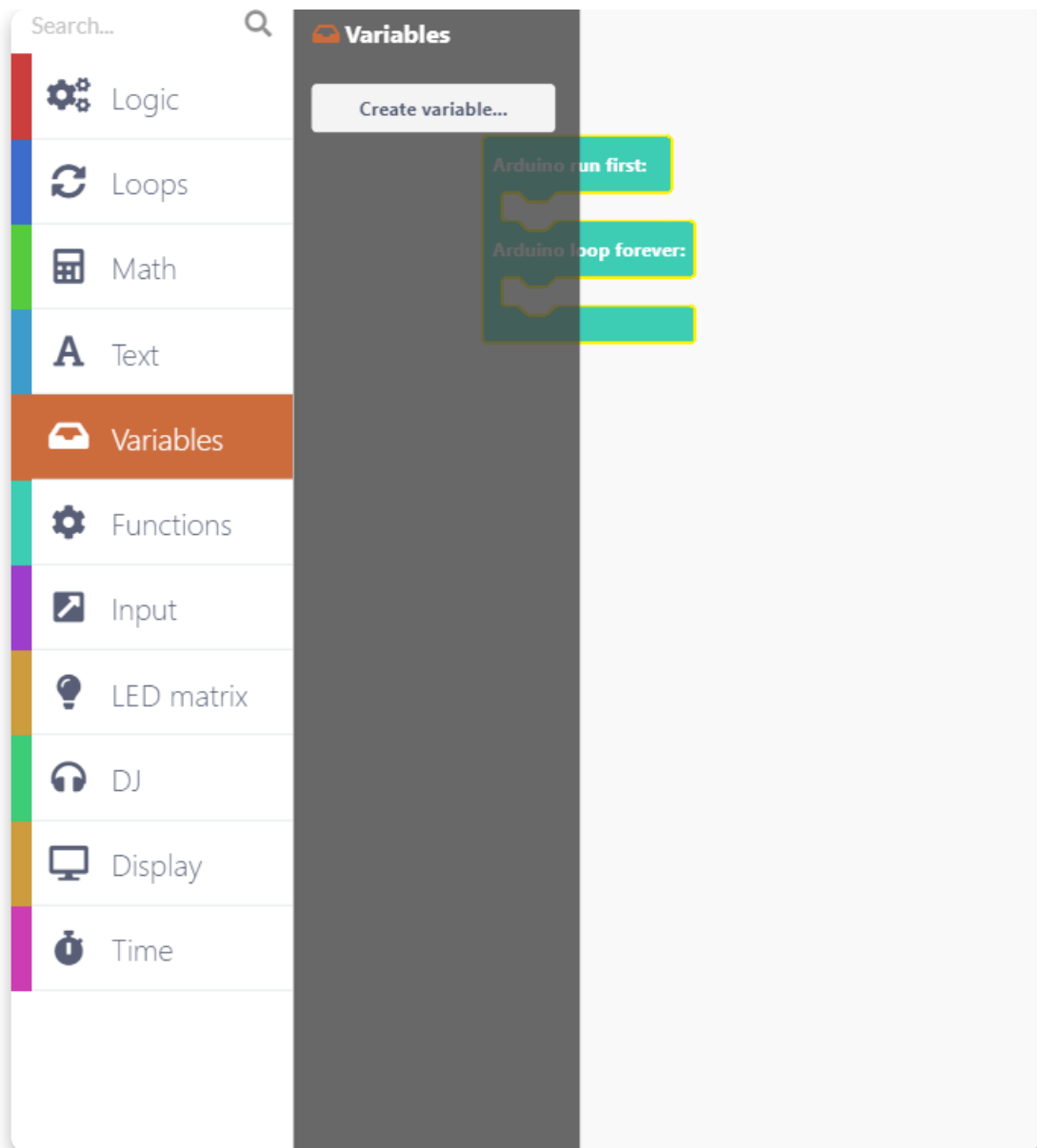
This time, we're going to create a DJ interface where you can remix one song, add different effects and intensities.

Also, we'll explore options for printing letters on display. This function will enable us to see the names of the available sound effects on the LCD display.

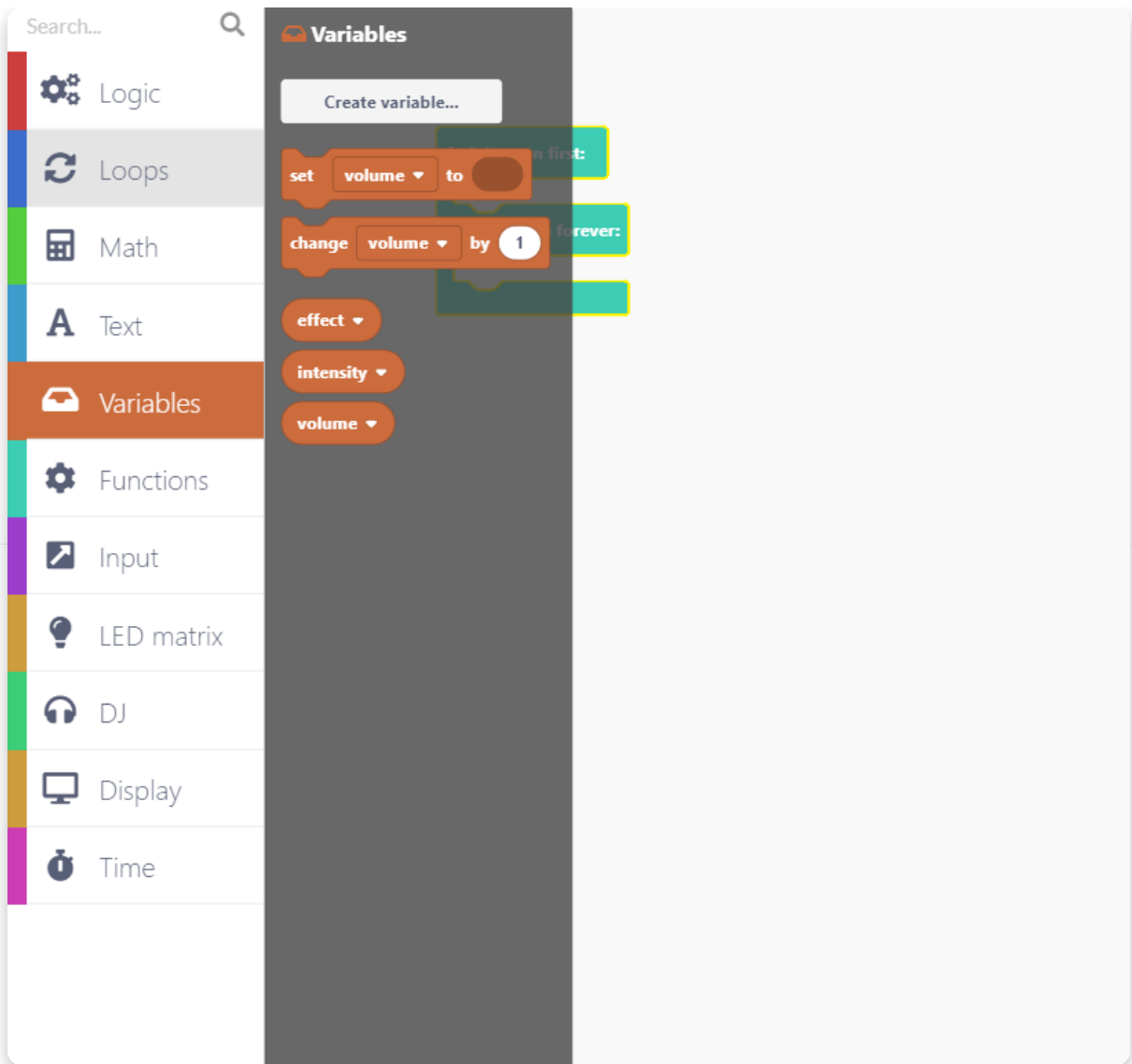
Let's start with creating a new Jay-D block project.



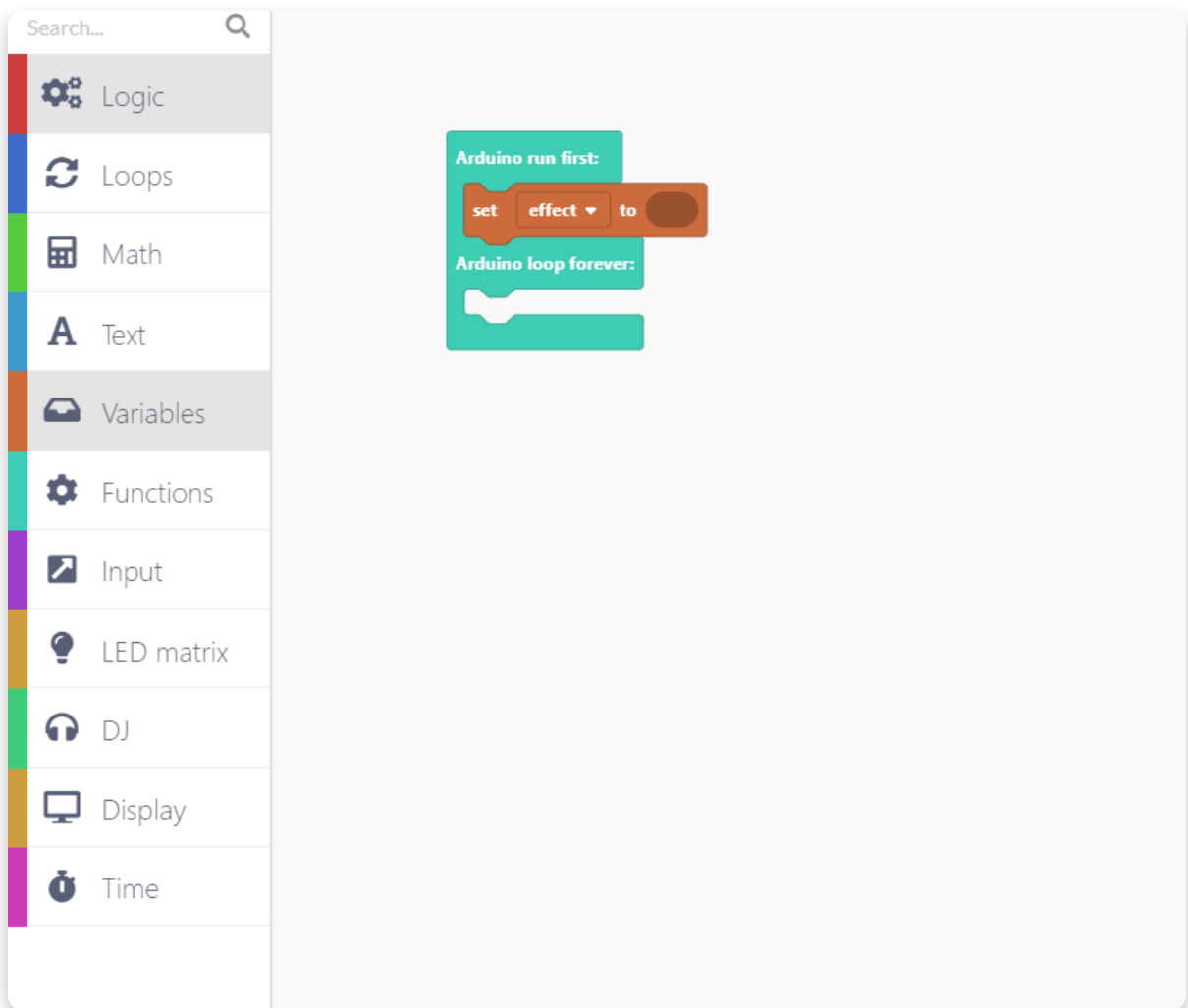
Now let's create three variables that we'll use. Name the variables effect, intensity, and volume.



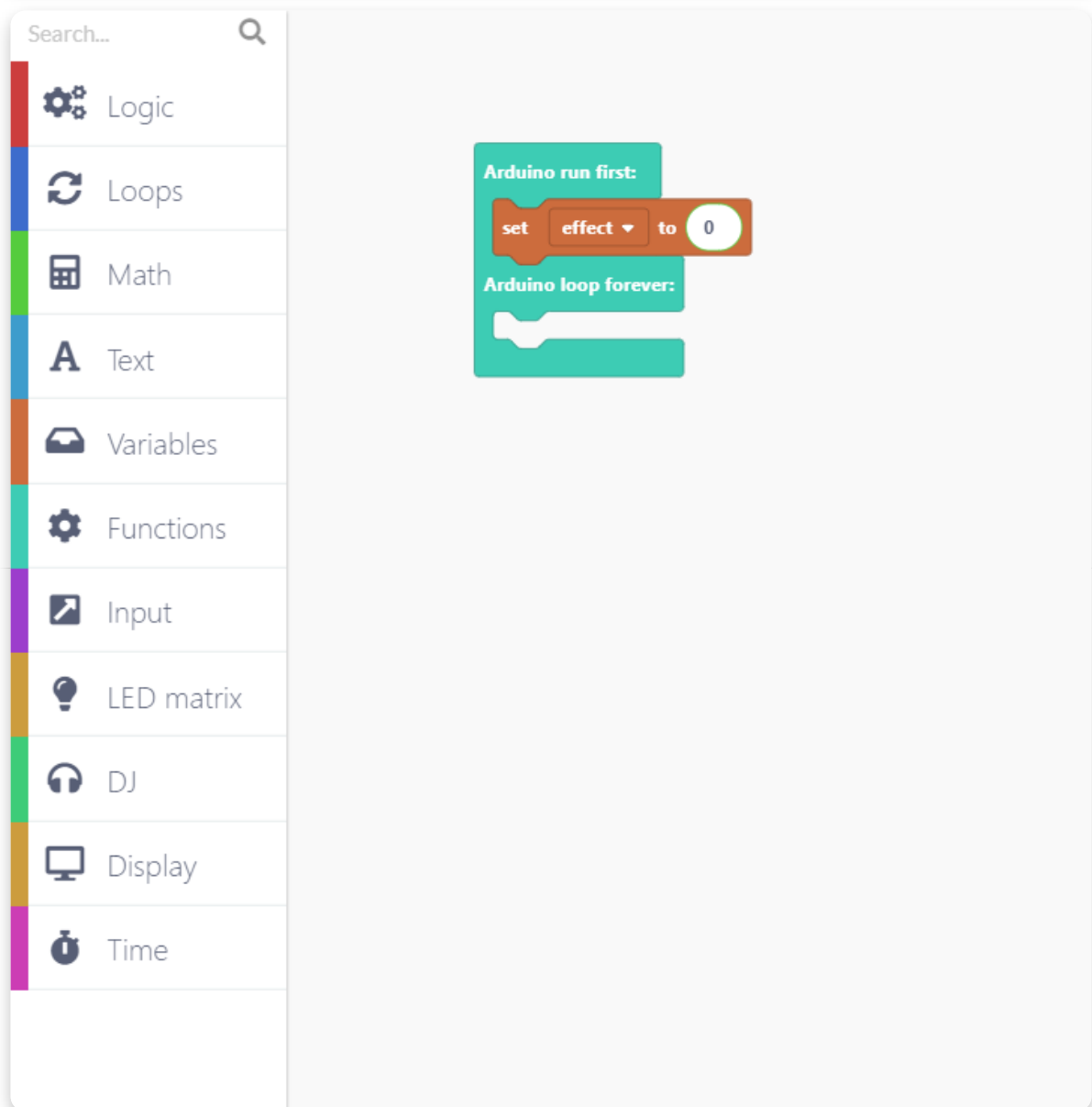
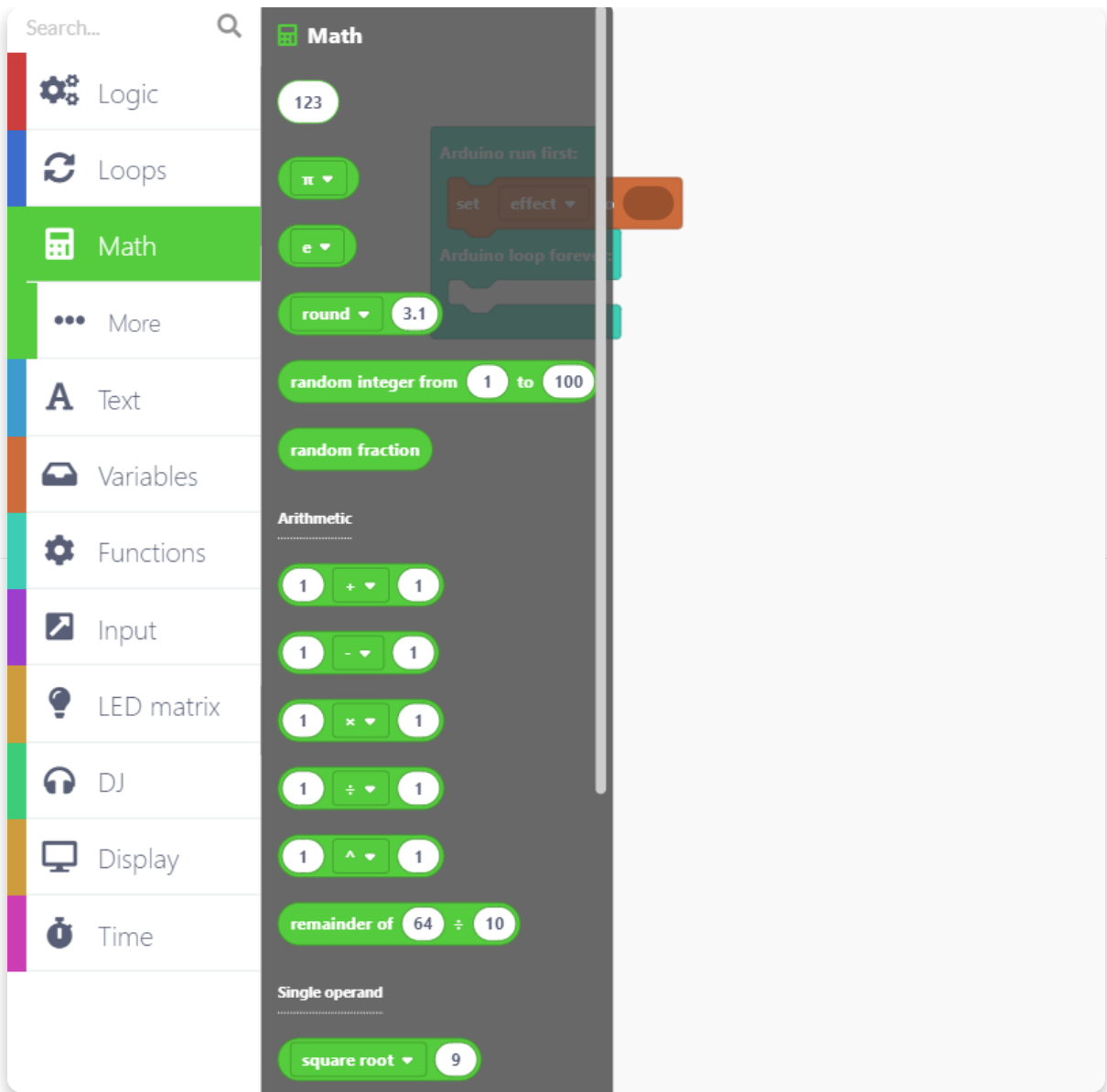
Repeat this step three times, so you have the following variables available:



Drag and drop the "set" block under Arduino run first and choose the "effect" in the drop-down menu:



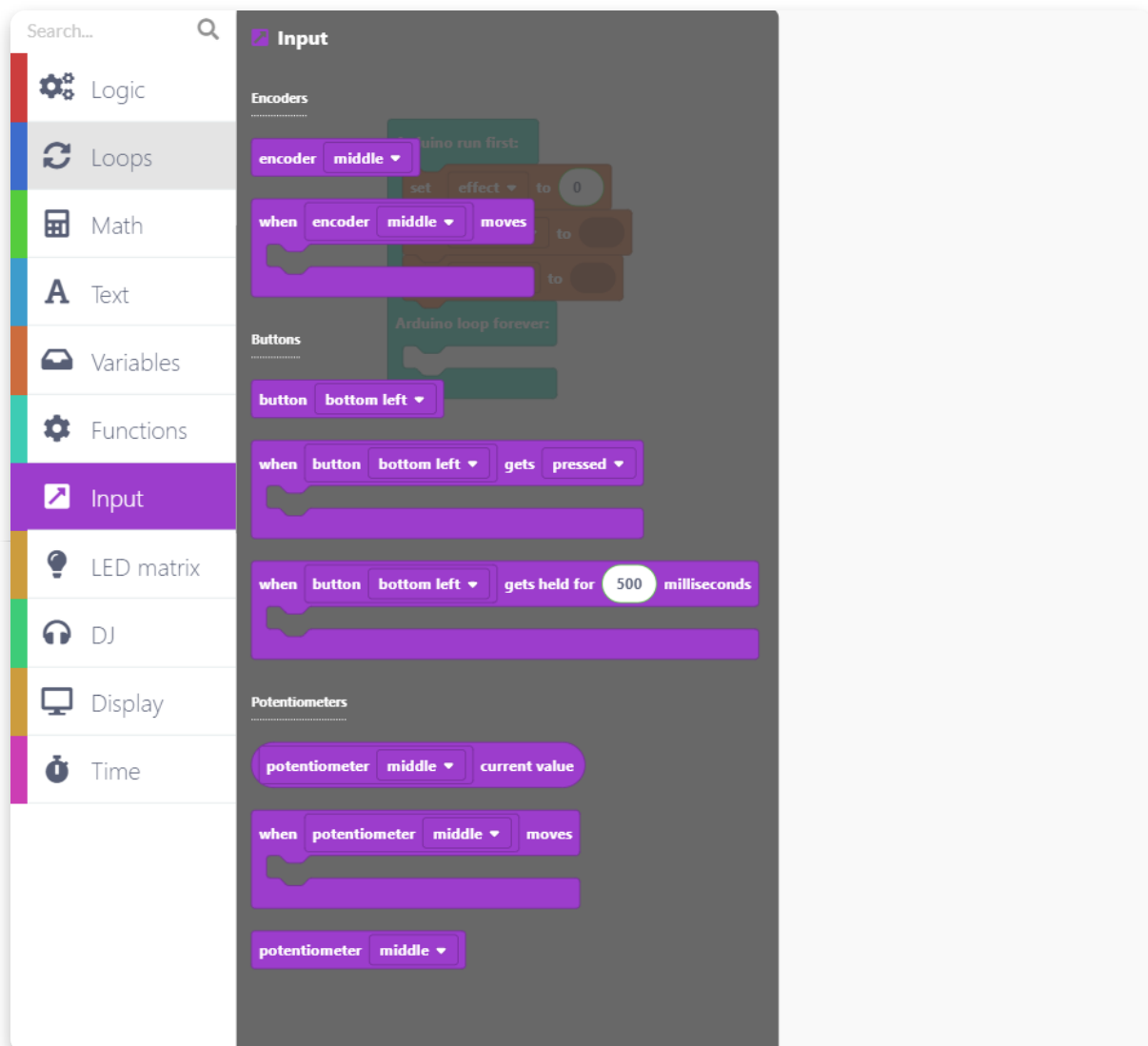
In the "Math" section, choose the "123" block and insert it into the previous block. Put the 0 value instead of 123.



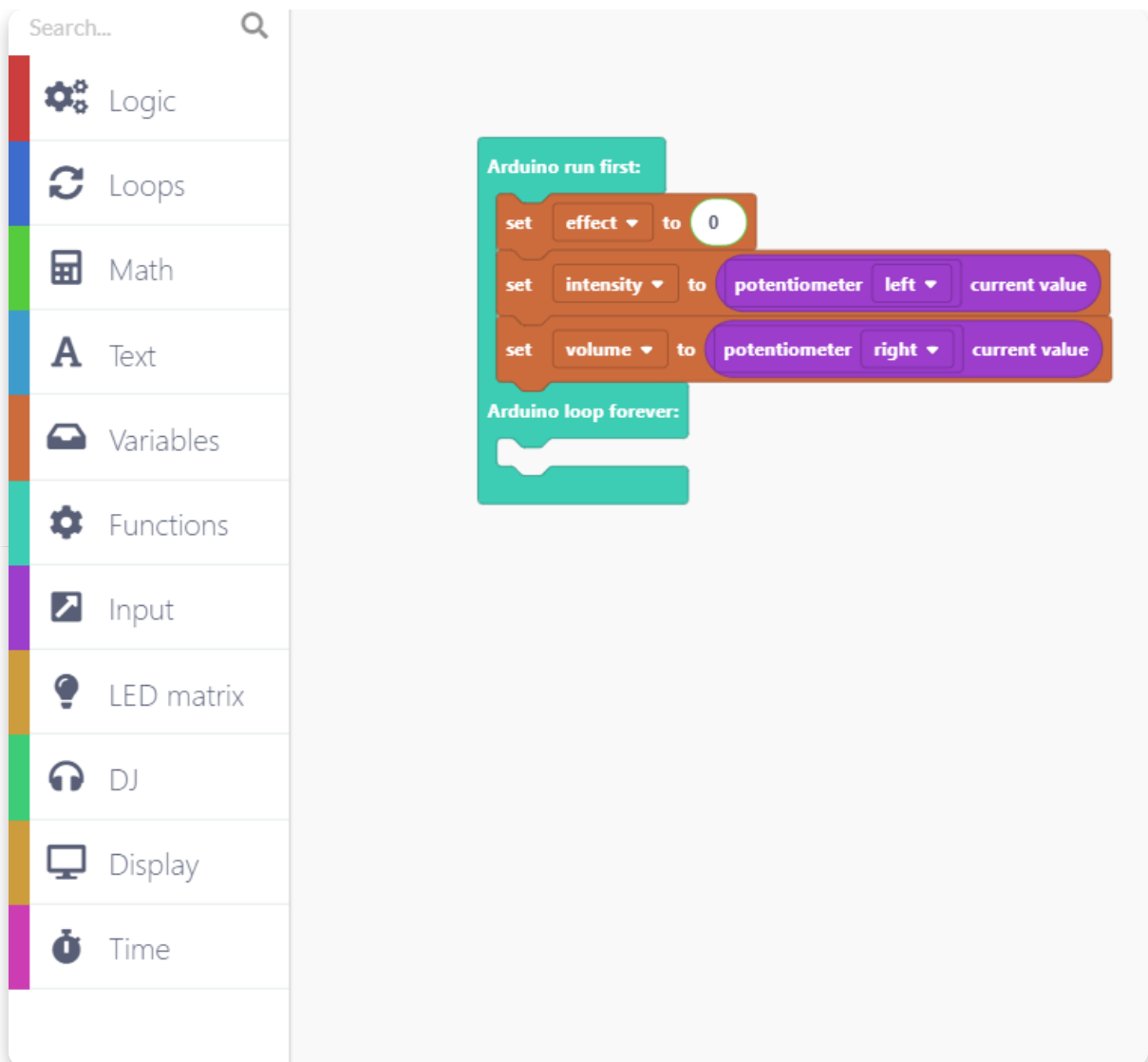
Now drag and drop two more "set" blocks from the "Variable" section.

Open the drop-down menus and choose intensity in one block and volume in the other block.

Intensity and volume will be controlled by the slider potentiometers. So, go to the "Input" section and find the block that says "potentiometer middle current value".



Put this block into the variable block for intensity and volume. Set the intensity potentiometer to be the left one and the volume potentiometer to be the right one.

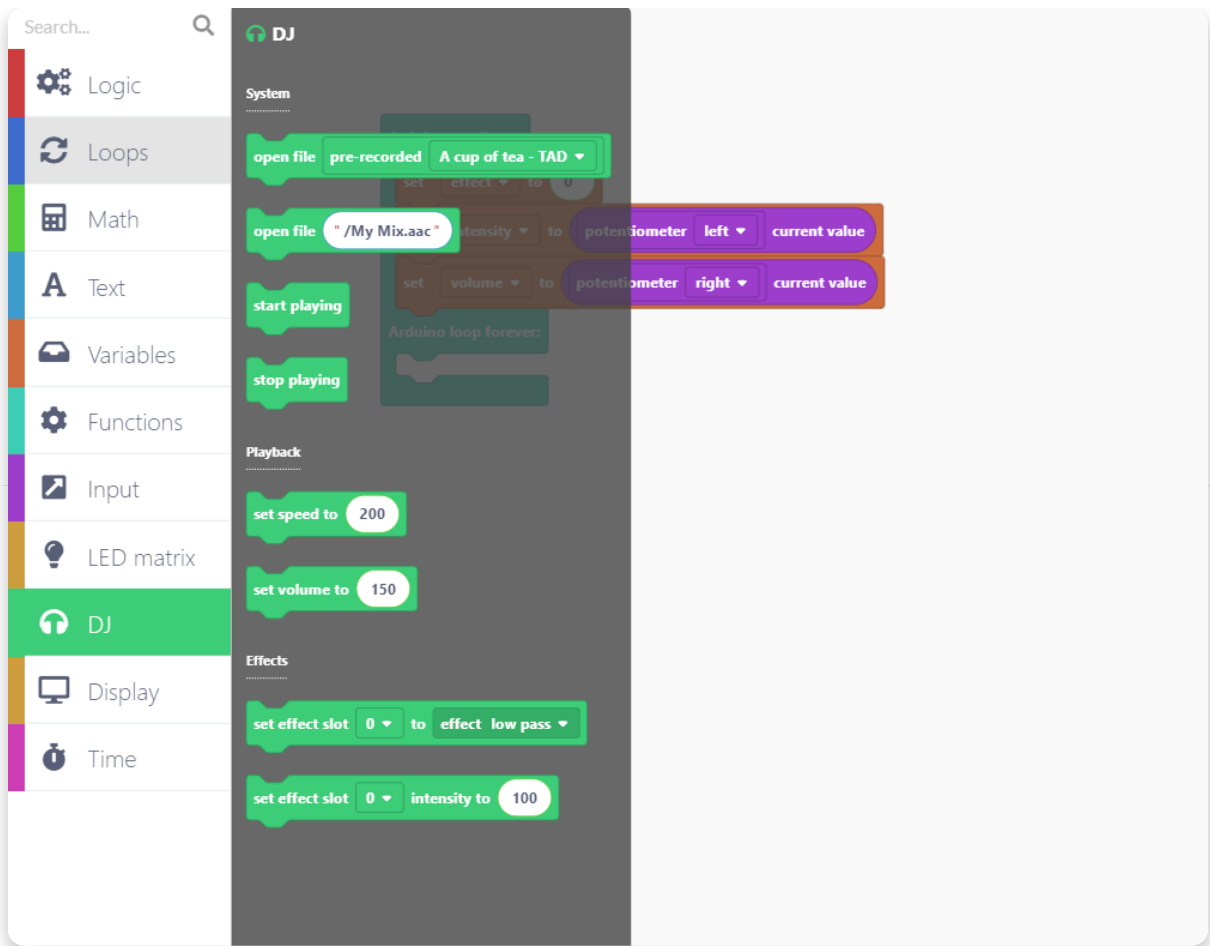


Let's open a file on the memory card. This is a song that you're going to remix.

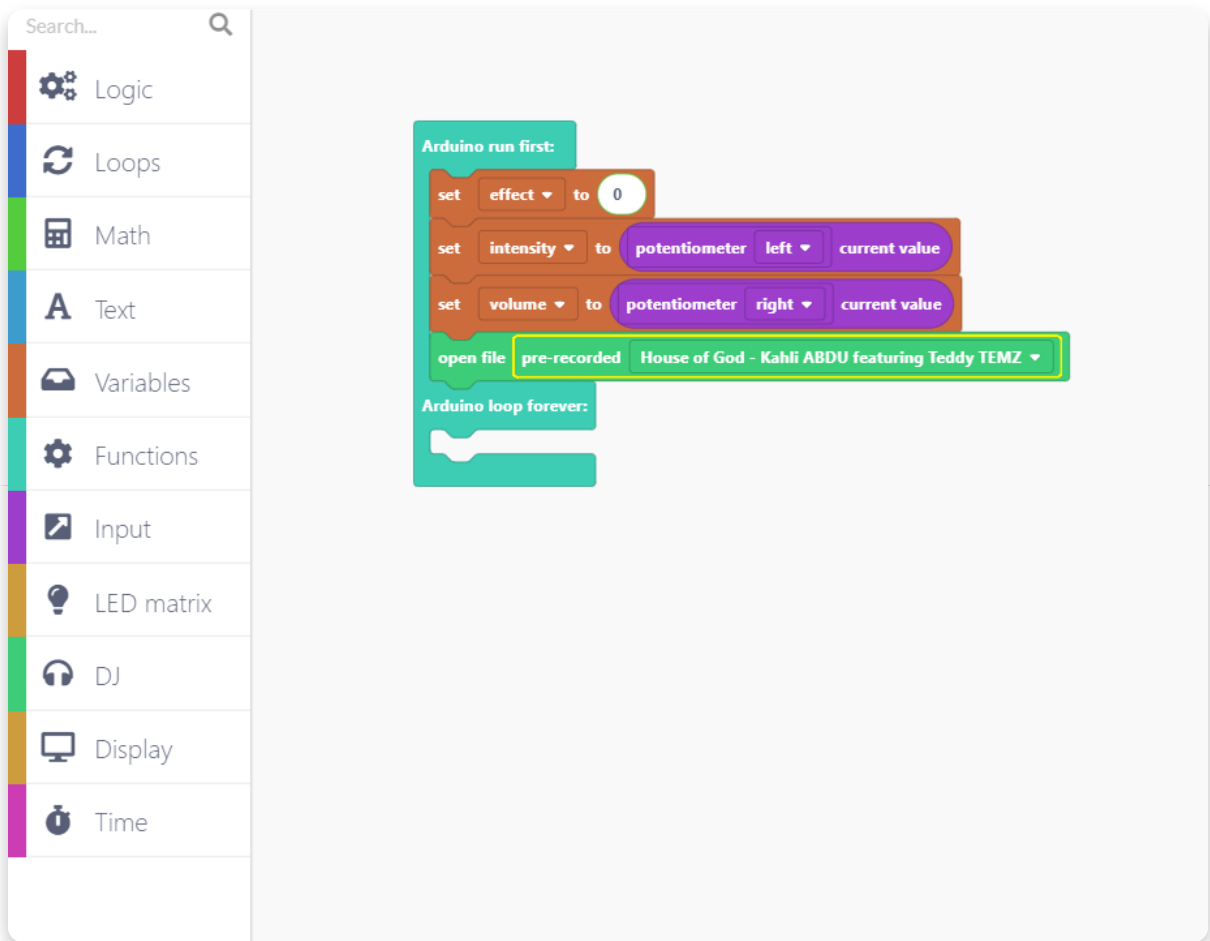
Go to the "DJ" section and find a block "open file pre-recorded". This block will give you an option to choose from songs that were pre-recorded on the memory card.."

If you'd like to use a song that you added yourself, choose a block that says "open file (name of the song)".

You can put this block after the variable blocks that we added earlier.

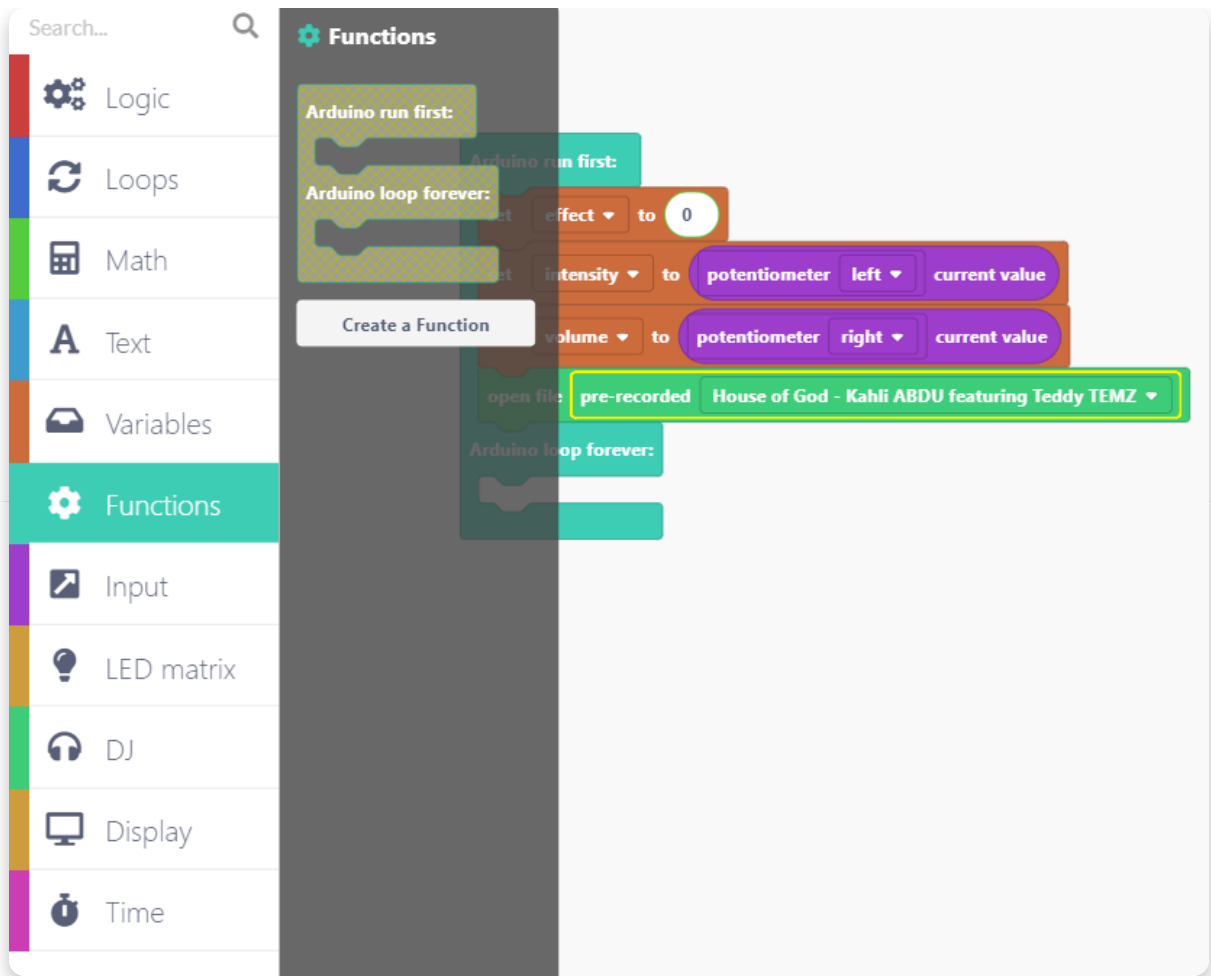


Open a drop-down menu in this block and choose any song you'd like to remix.

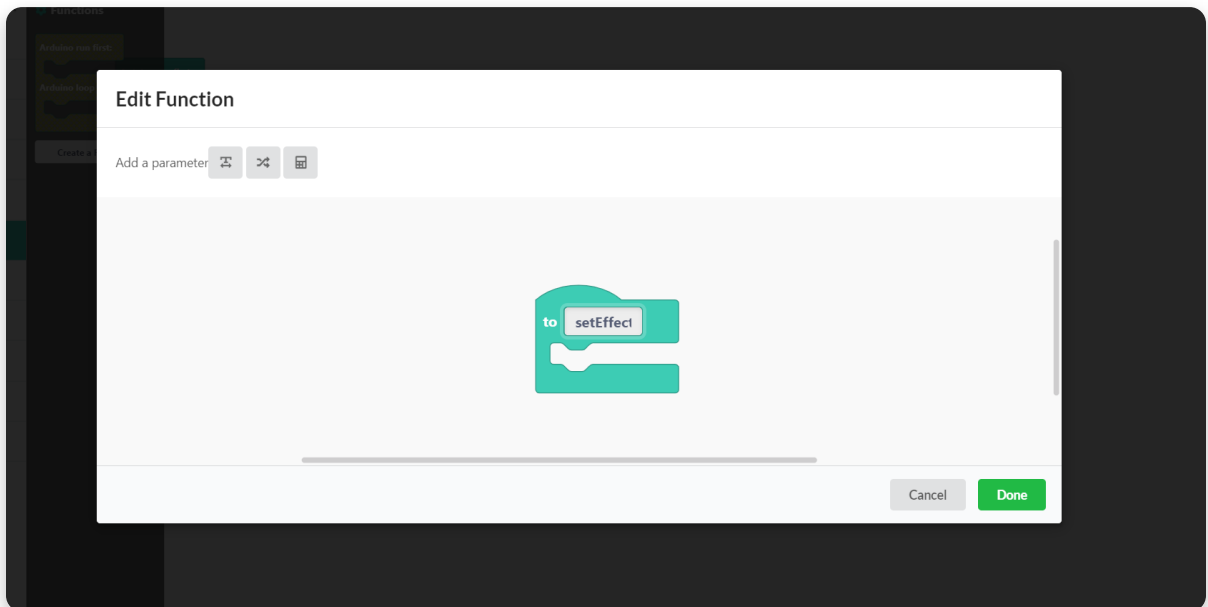


After you have chosen a song you want to remix, let's create a function that will set the effects for remixing.

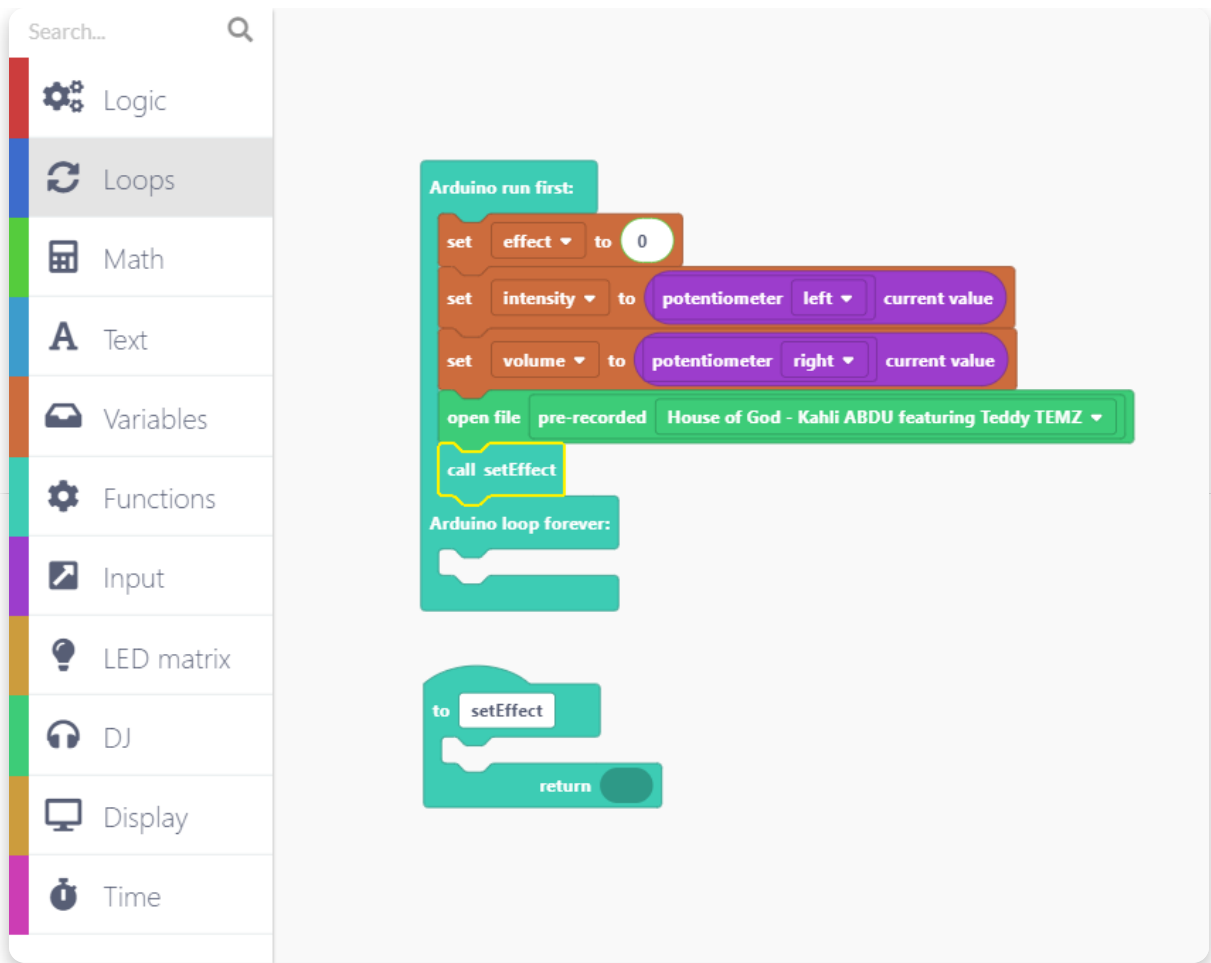
Go to the "Functions" section and create a new function.



Let's call it "setEffect":

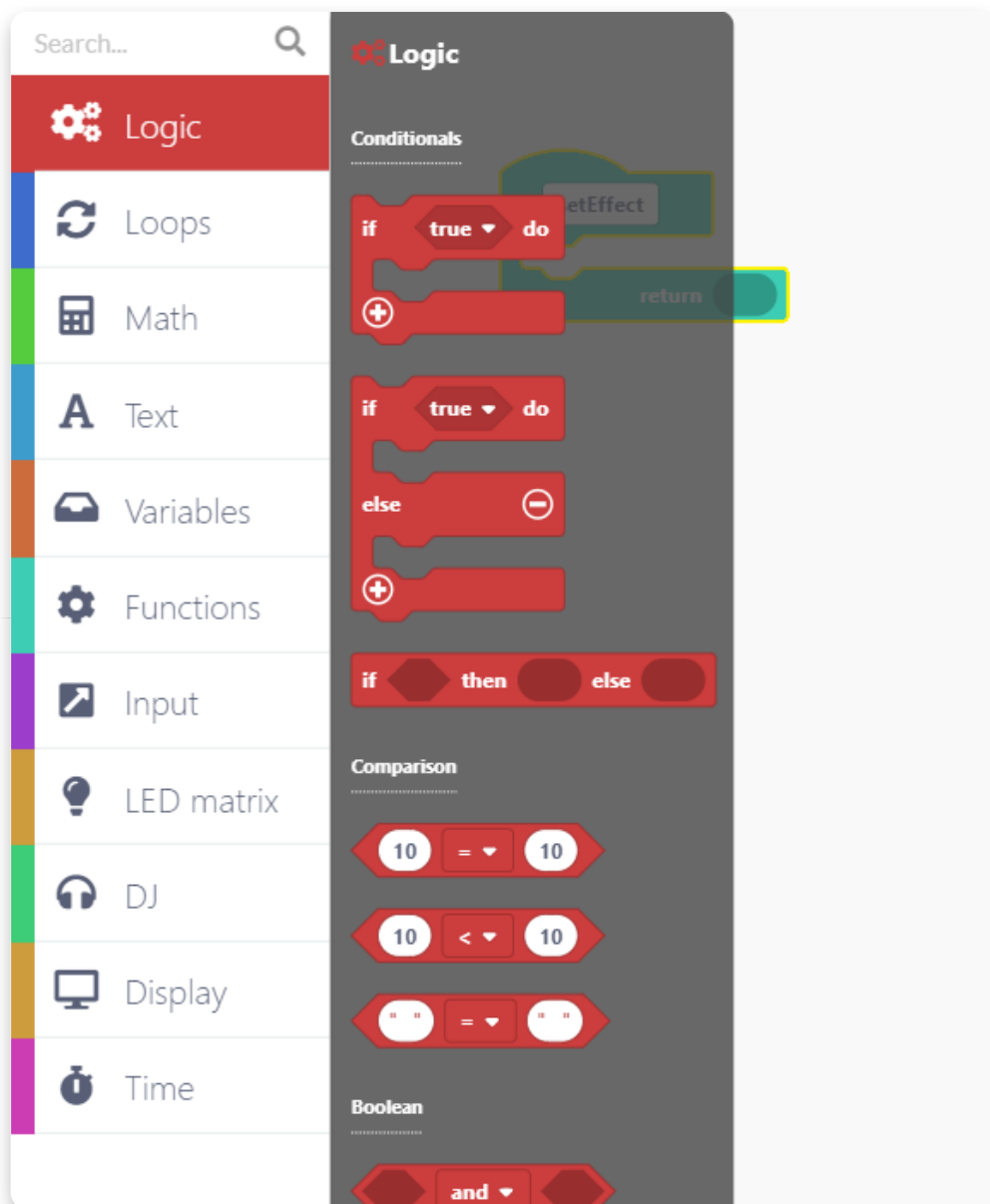


The function will appear in the sketch, but don't forget to put the "call effect" in the first block after the "open file" block.

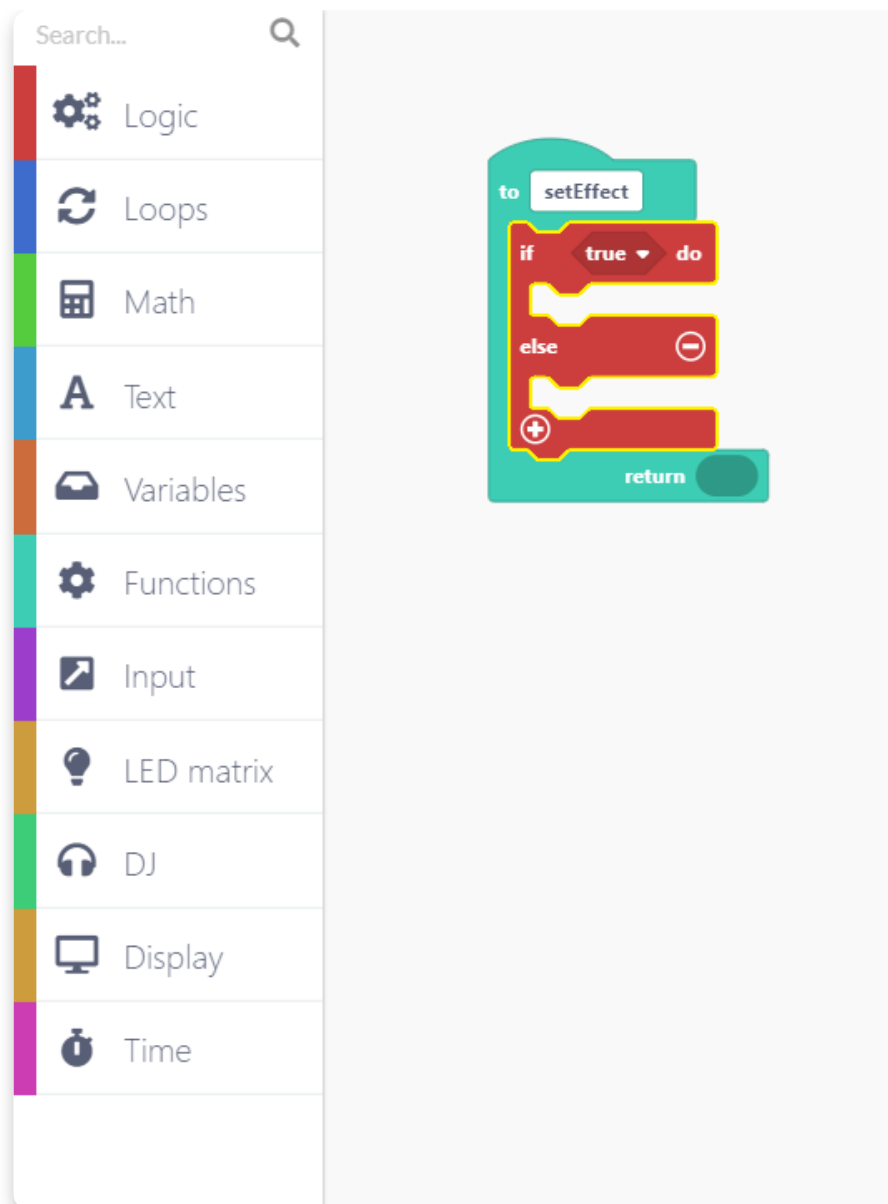


We'll now use a logical function for setting the effect. There will be 5 options, so bear with us.

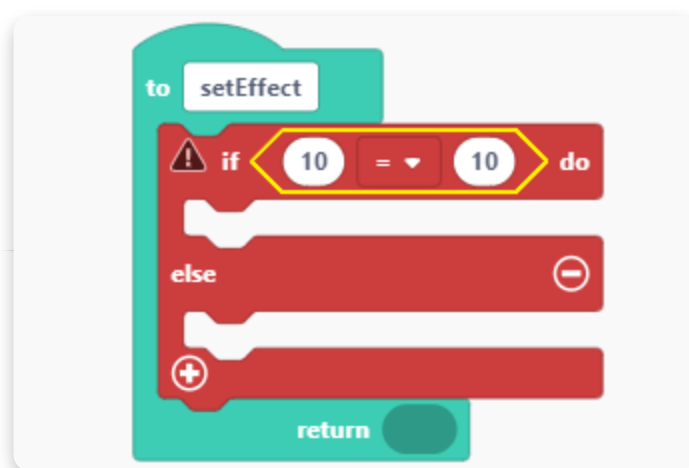
Go to the "Logic" section and choose a second block with the *if* and *else* option.



Drop it into the "setEffect" block.

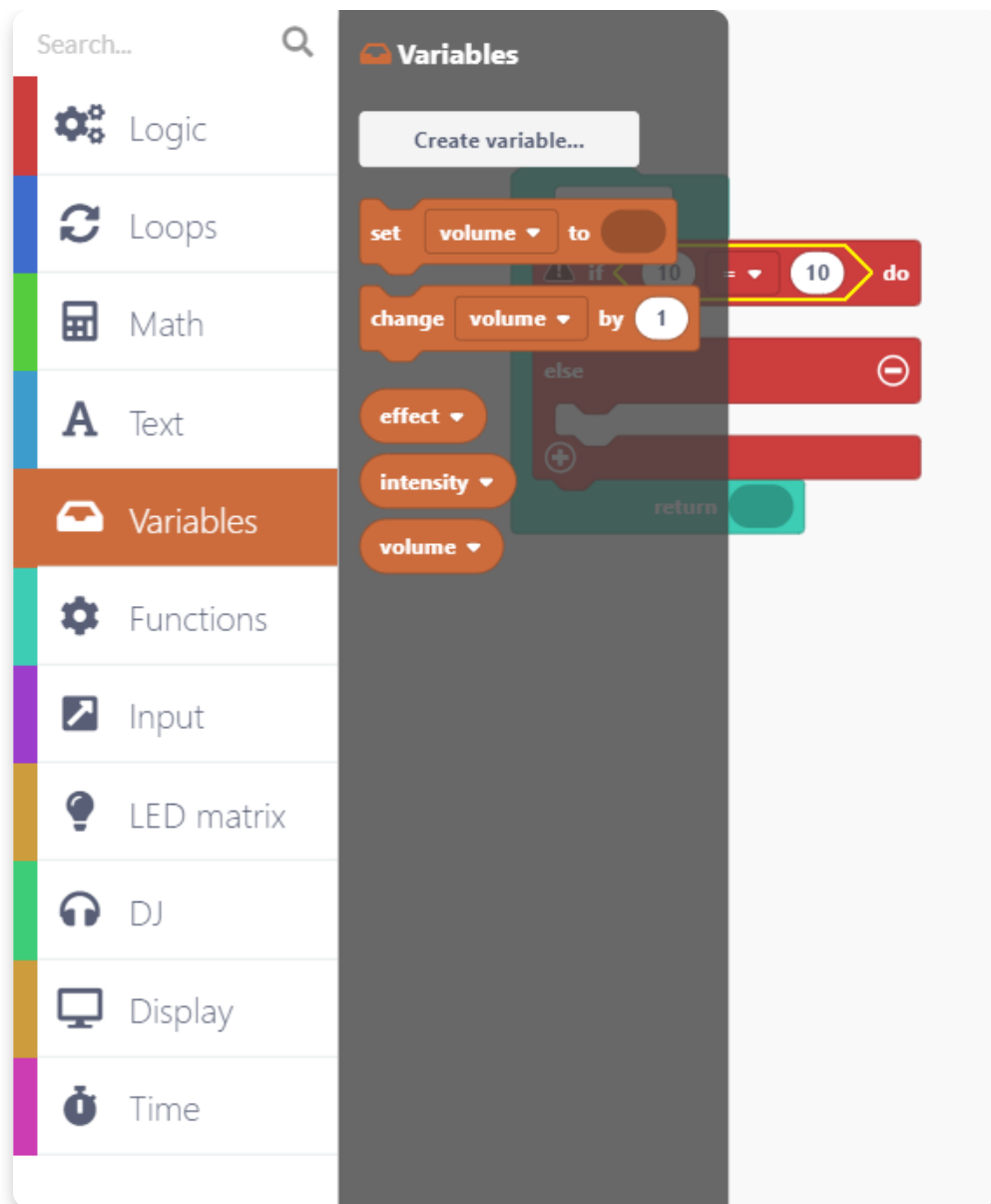


We'll now insert a new logic block instead of the "true" option. So, go to the "Logic" section and drag and drop the following block:



This is a comparison function, so we'll need to use the "effect" variable instead of numbers. Each effect will have its number, so we will assign those numbers by using this function.

Go to the "Variables" section and put the "effect" variable in the first window in the comparison.

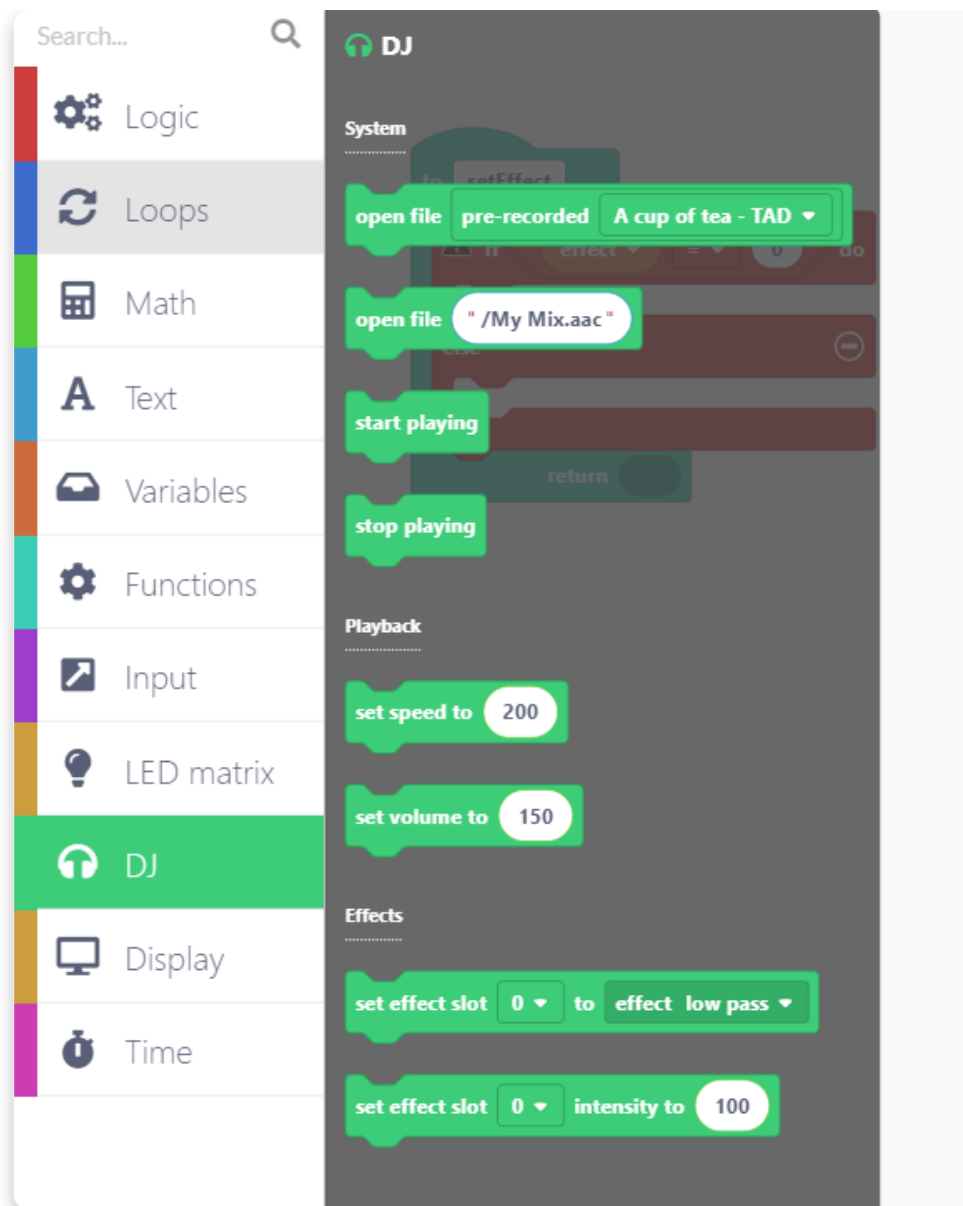


Make sure to keep the equal sign and type in the 0 in the other window of the comparison.



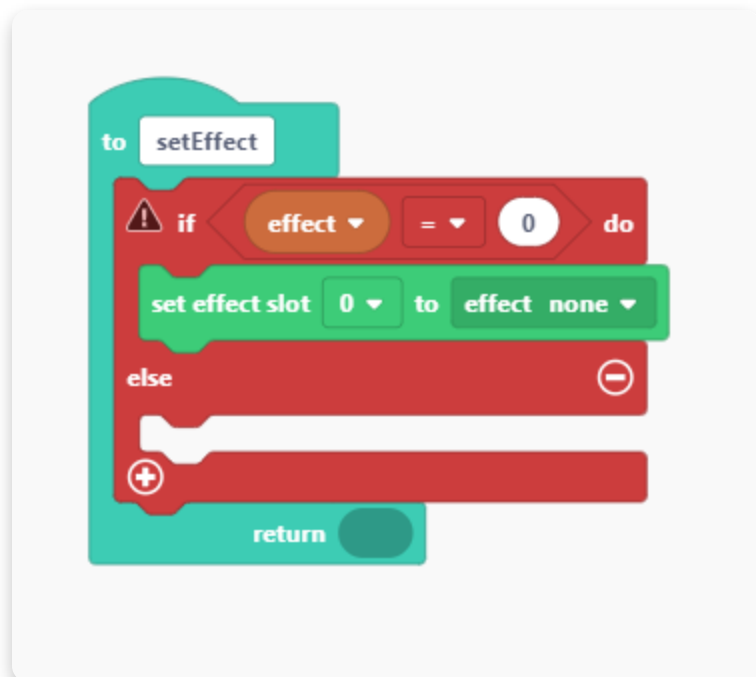
We just set the comparison for the effect to be equal to zero. If that is the case, we need to select an action that will take place.

Go to the "DJ" section and find a block that says "set effect slot 0 to effect low pass".



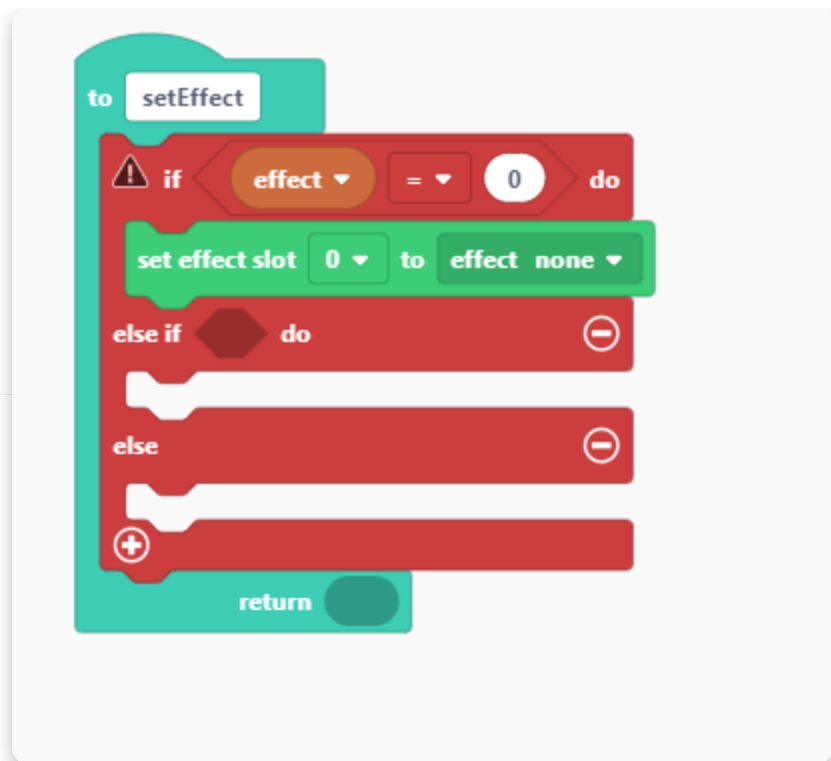
Drag and drop this one here:

Make sure to choose 0 in the first drop-down menu and effect none in the other drop-down menu.



Setting the none effect option is done. Let's repeat this a couple of more times, but adding different effects this time.

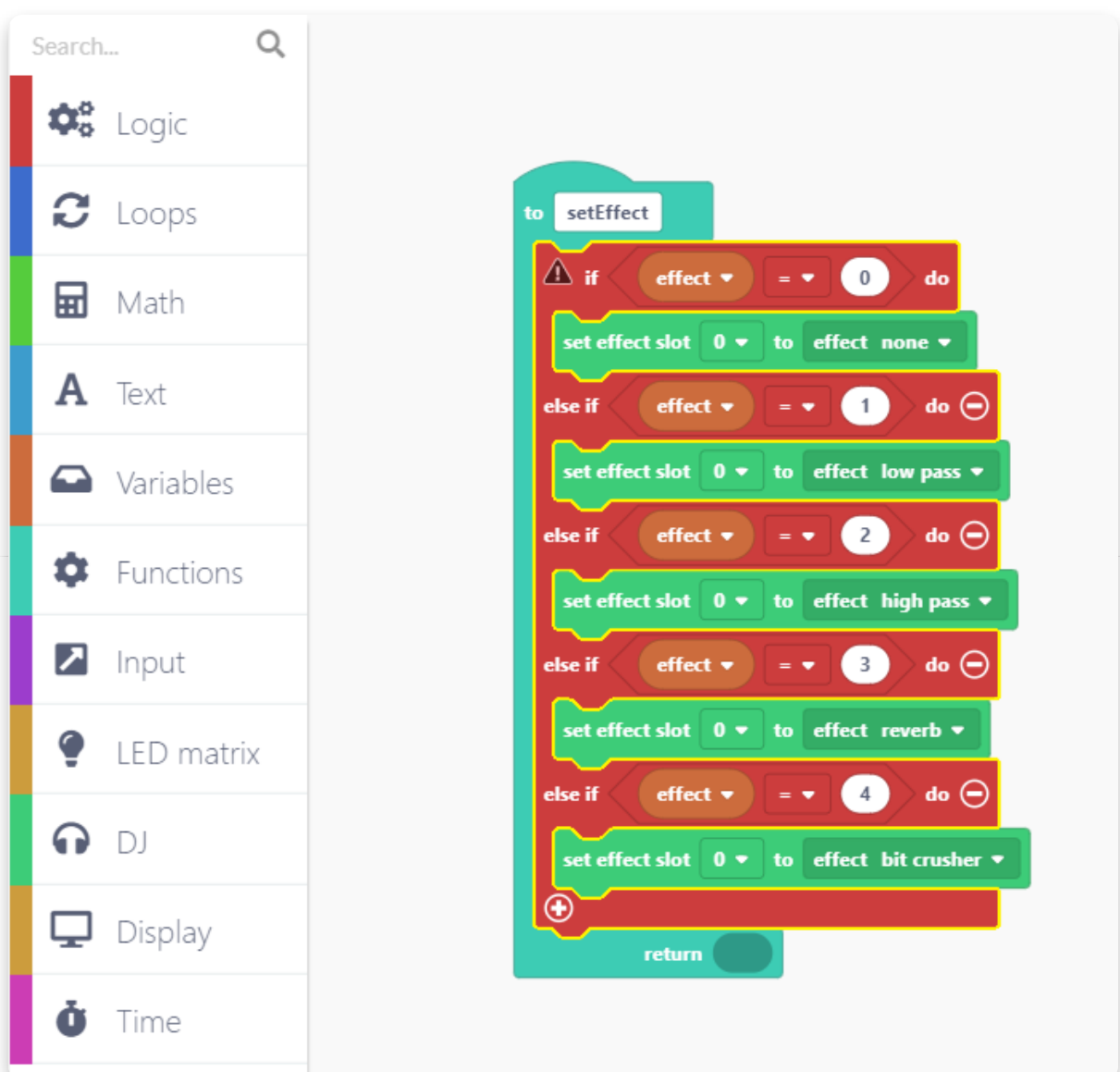
Every time you want to expand the logical block and add another effect, click on the small plus sign at the bottom of the block:



We repeated the previous step four more times for four new effects:

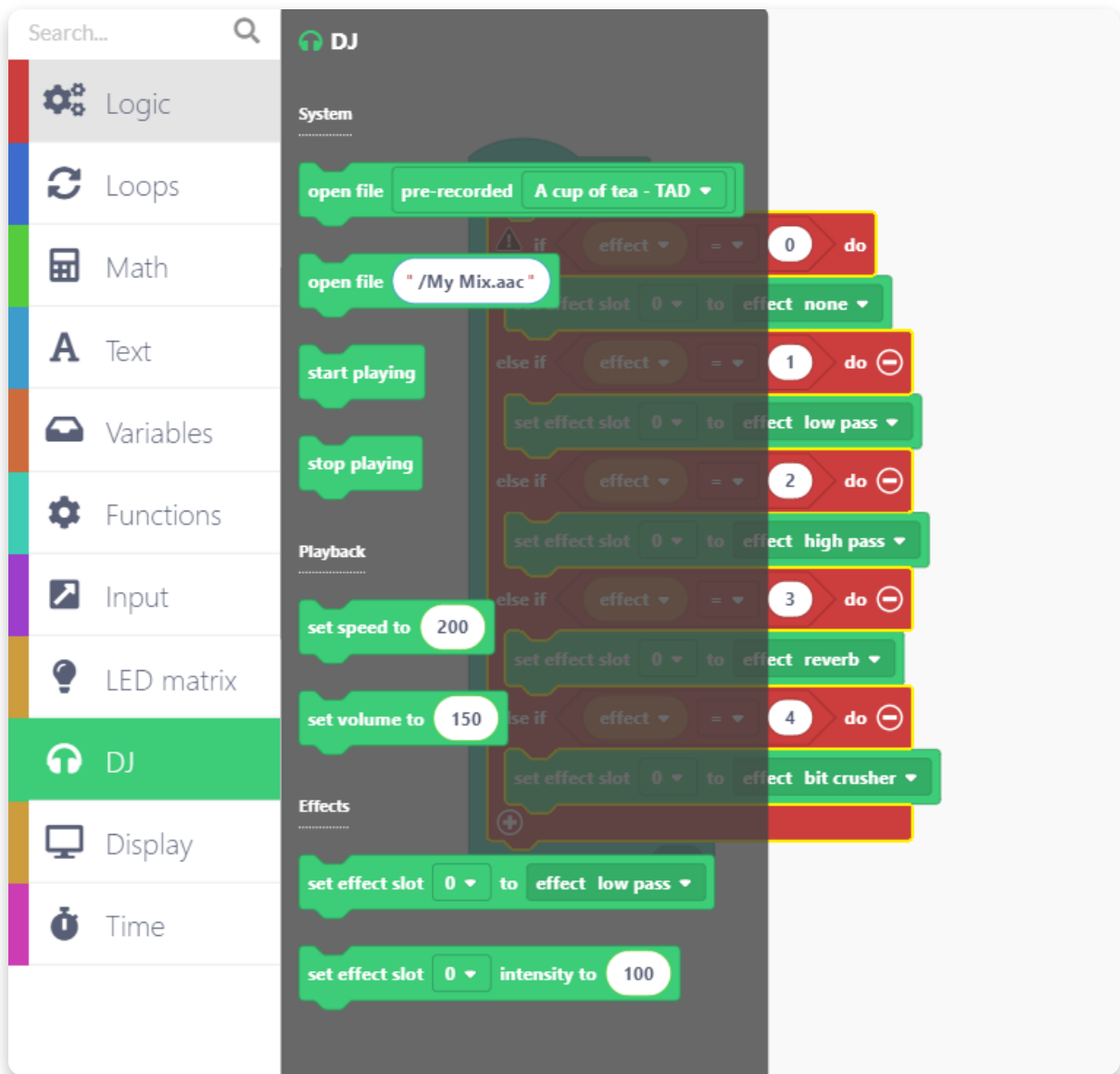
1. low pass
2. high pass
3. reverb
4. bit crusher

Close this block by clicking on the small minus sign next to "else".

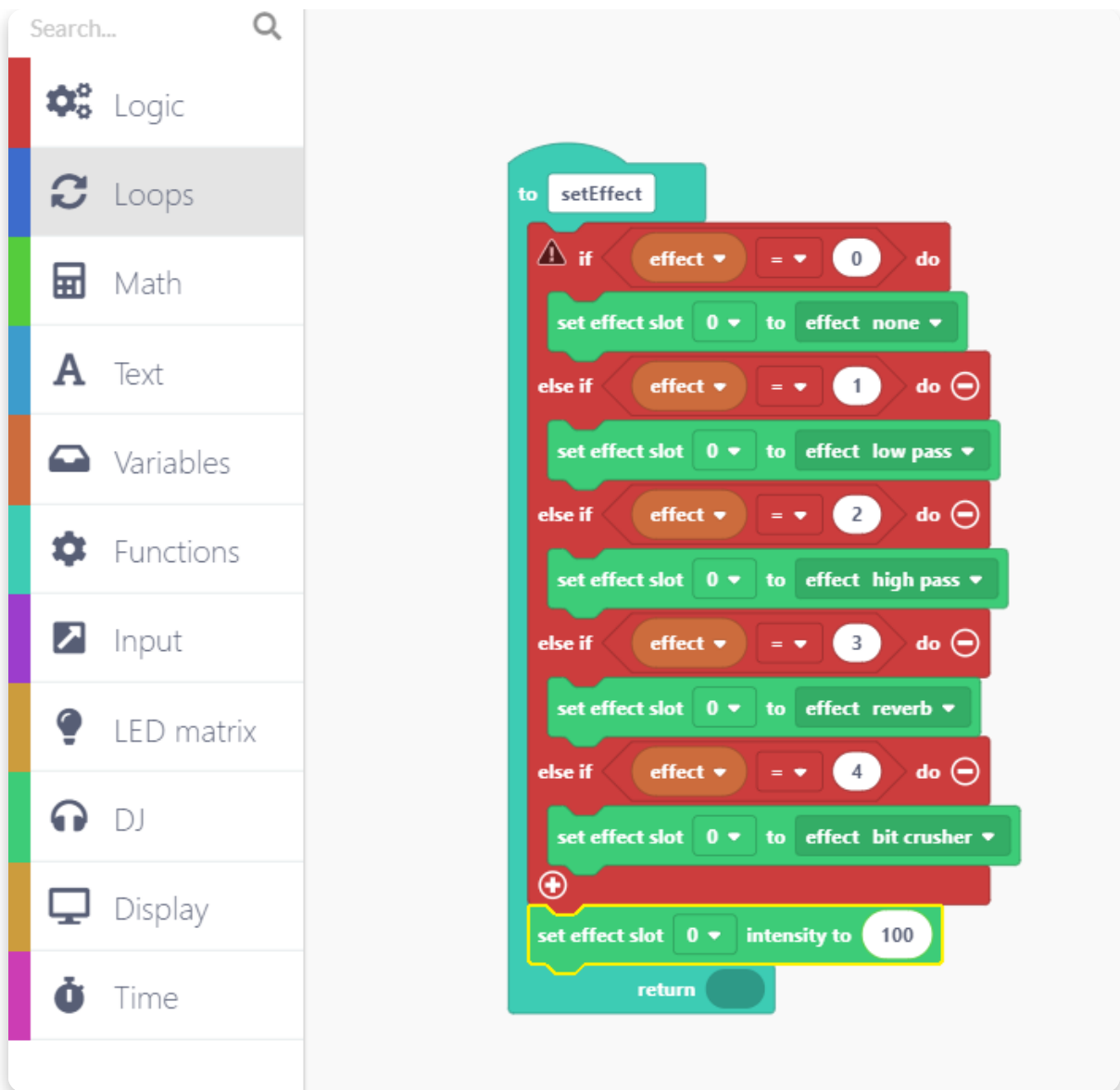


Lastly, we'll add the block that will connect the effect with the intensity we used as a variable previously.

Go to the "DJ" section and find the last block that says "set effect slot 0 intensity to 100".

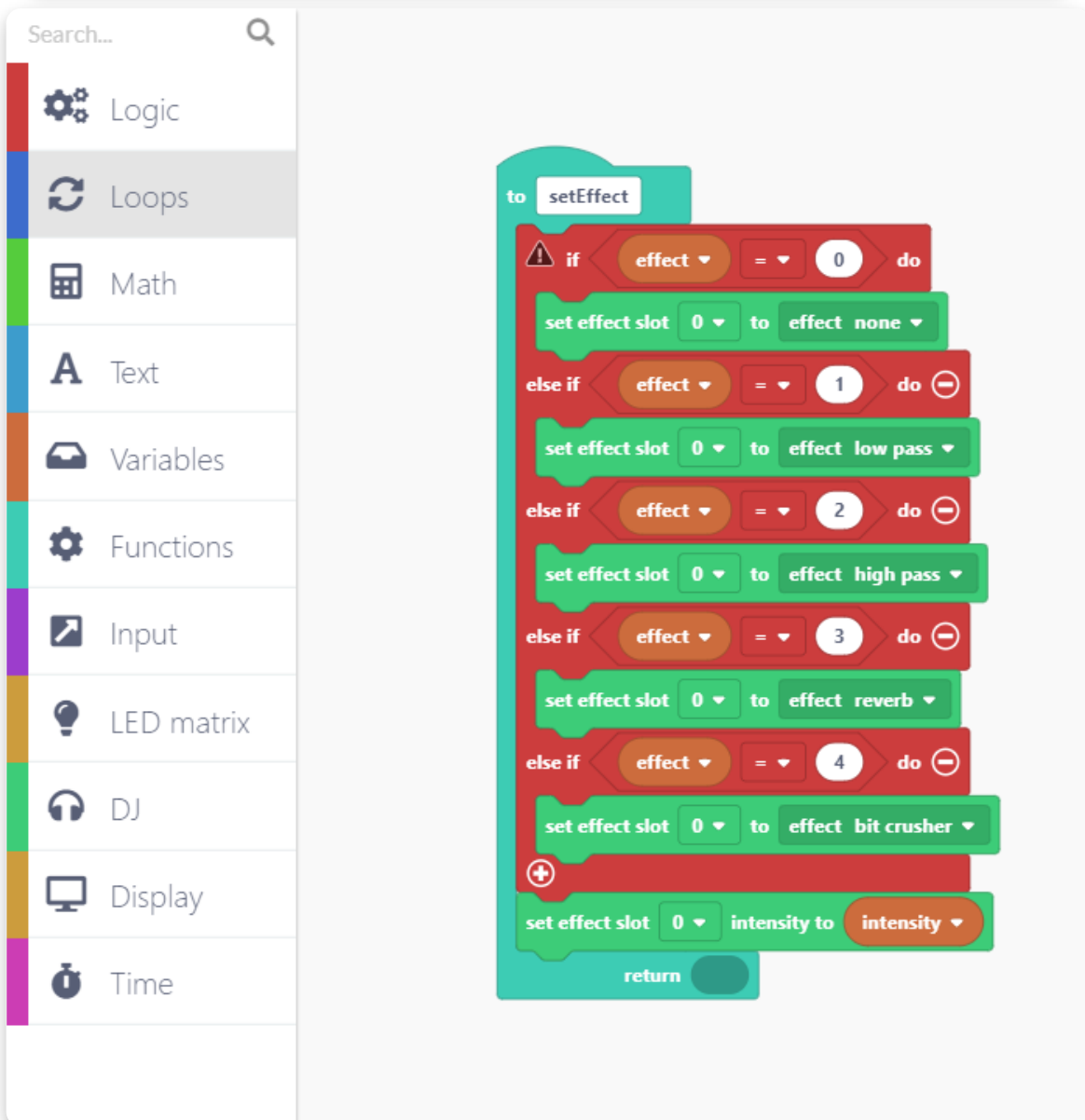
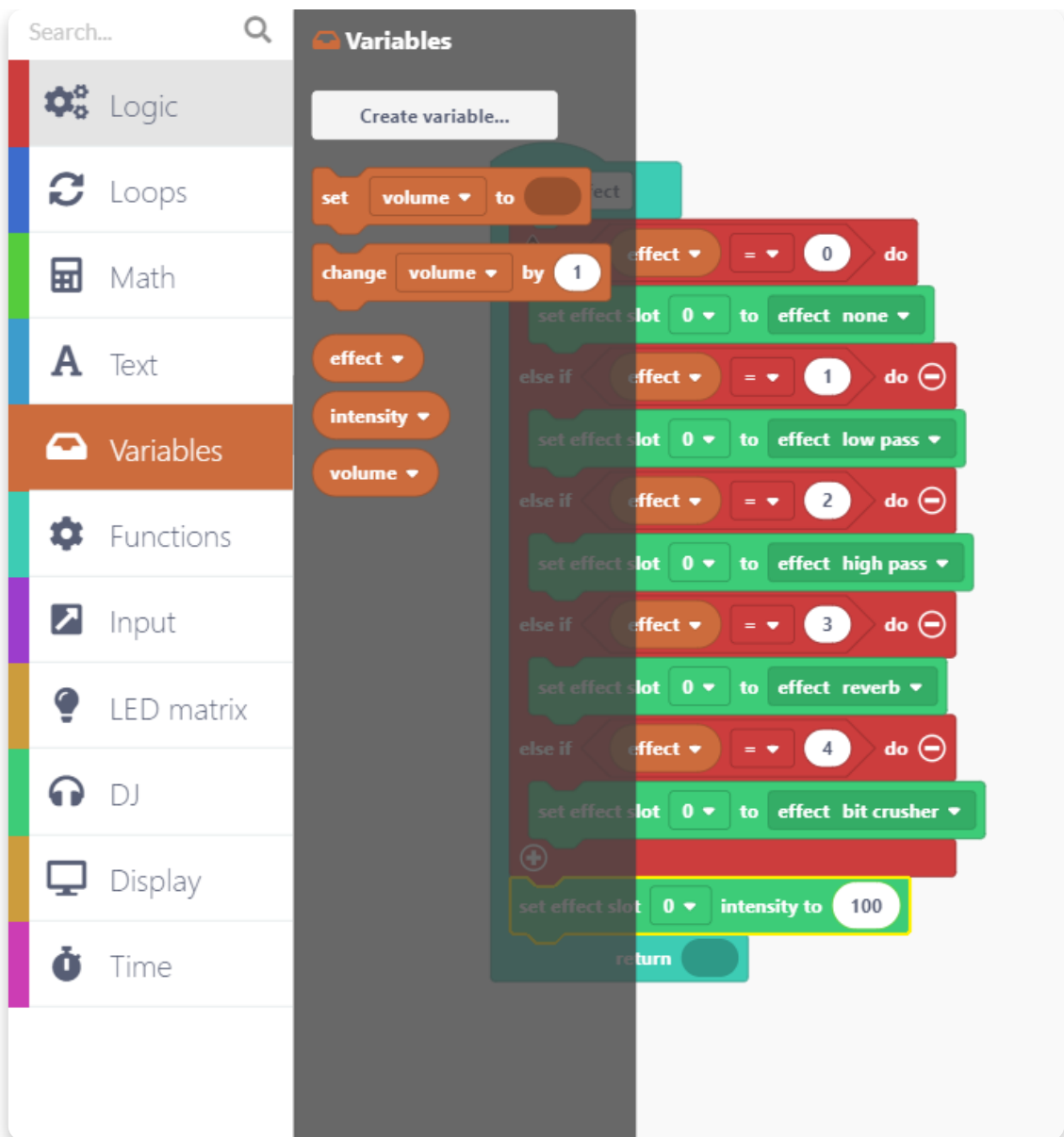


Put this block here:



Instead of "100", we'll use the intensity variable. So, go to the "Variables" section and drag and drop the intensity variable in the last block.

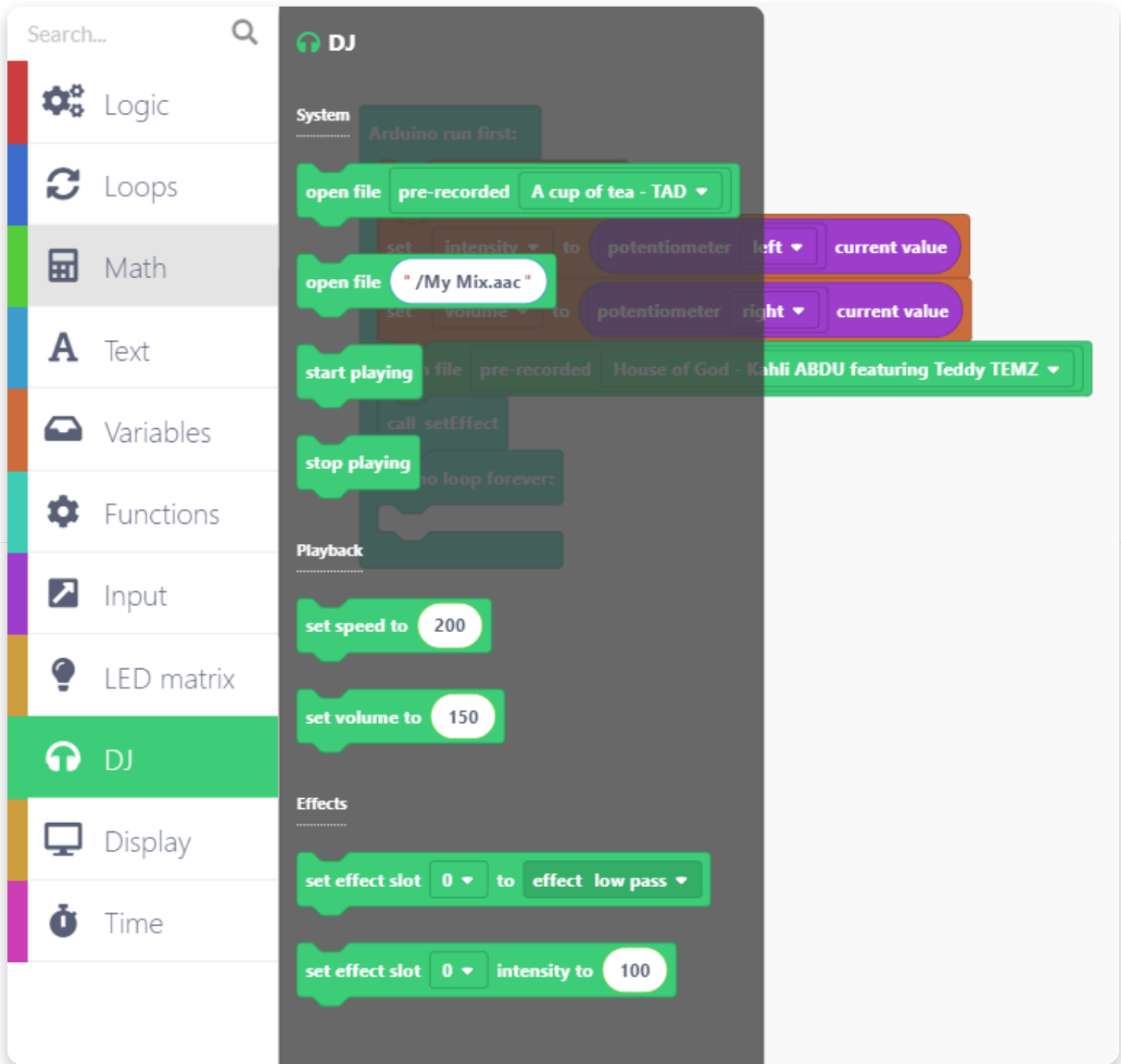
If the number pops up outside the block when you drop the intensity variable, just drag it on the side where the sections are to delete it from the sketch.



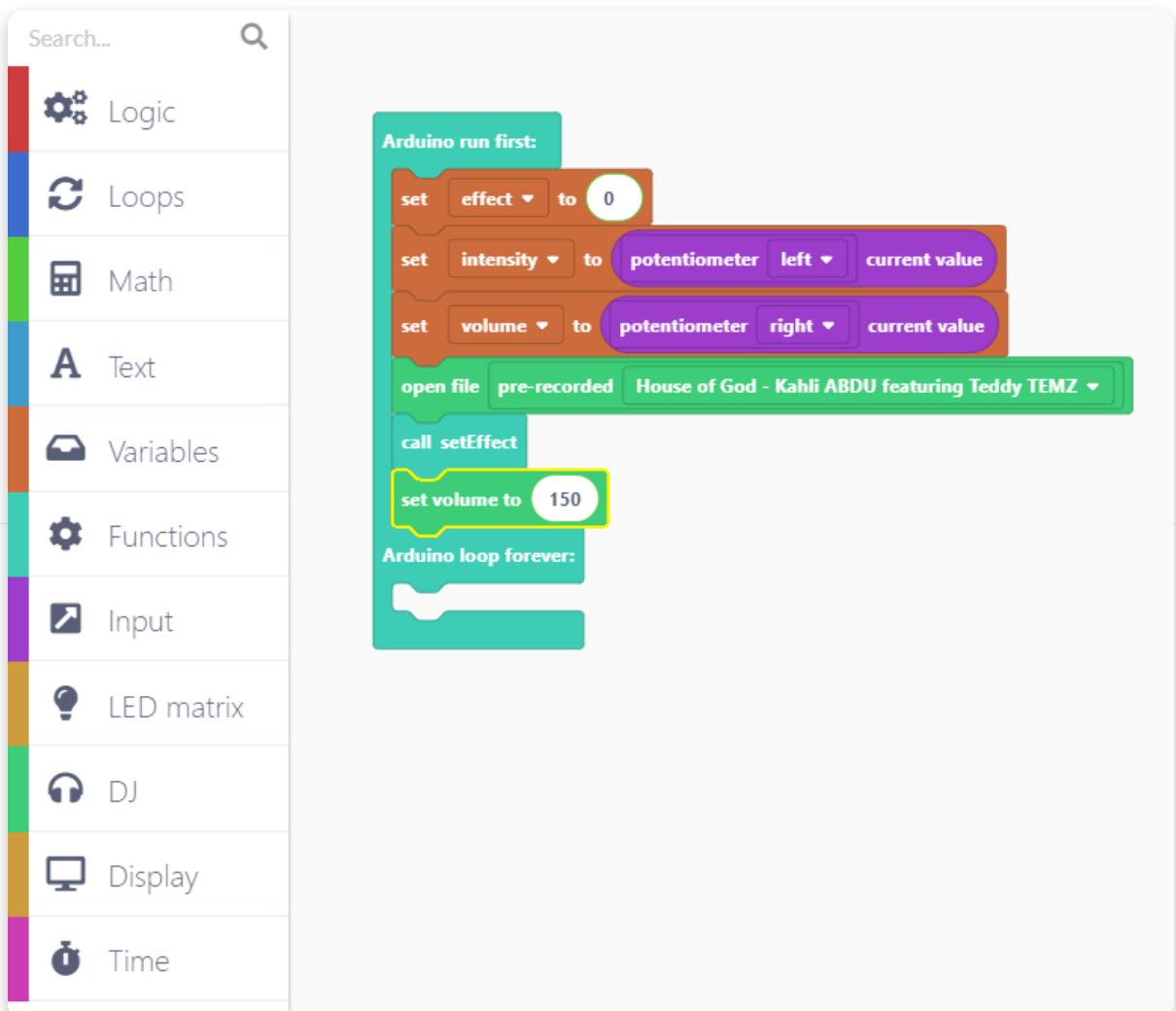
The function "setEffect" is done!

Let's return to the Arduino run first block and add the volume block.


Open the "DJ" section and find the "set volume to" block.



Drop it here:



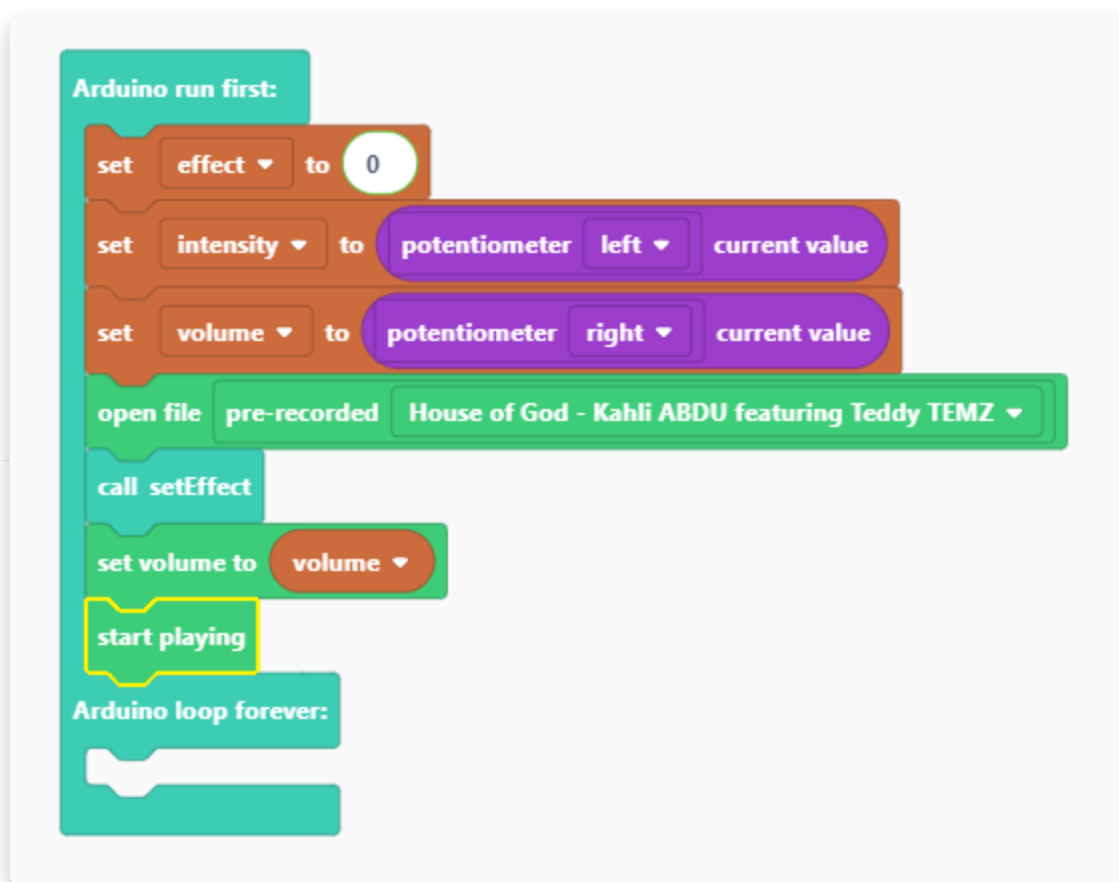
Like with the intensity variable, let's replace the number with the volume variable here as well:



```
Arduino run first:  
set effect to 0  
set intensity to potentiometer left current value  
set volume to potentiometer right current value  
open file pre-recorded House of God - Kahli ABDU featuring Teddy TEMZ  
call setEffect  
set volume to volume  
Arduino loop forever:
```

Open the "DJ" section again and find a block that says "start playing".

Put it here:

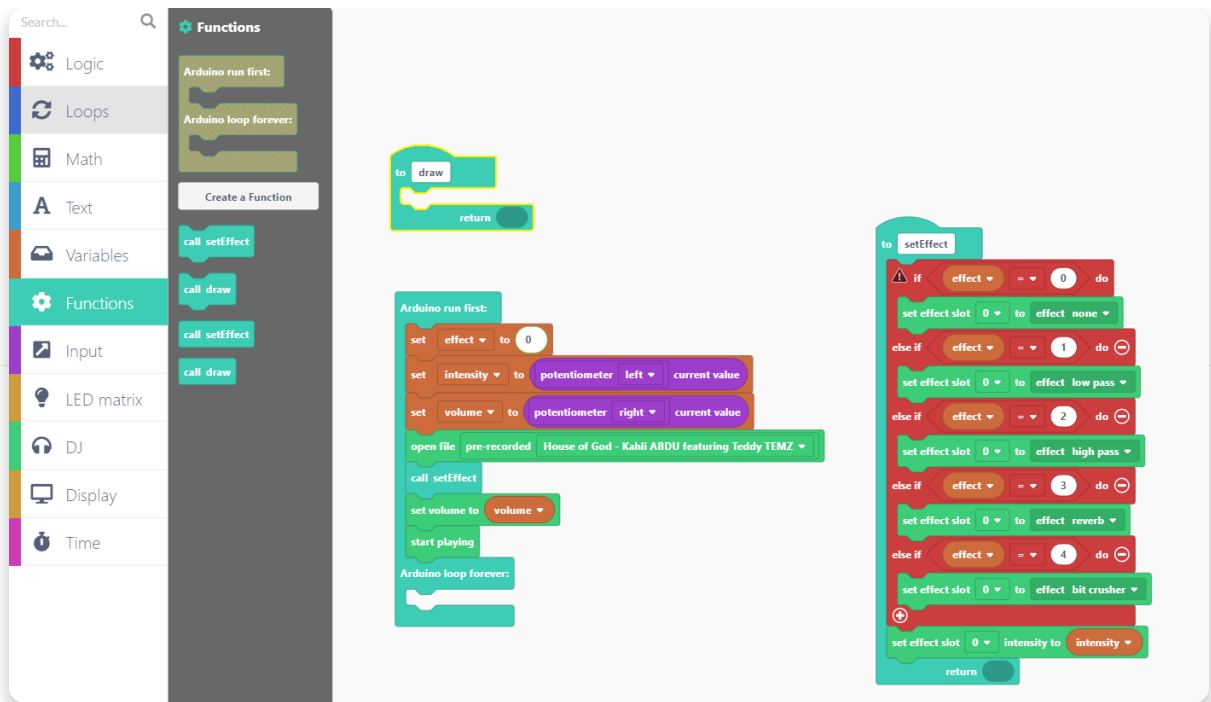


```
Arduino run first:  
set effect to 0  
set intensity to potentiometer left current value  
set volume to potentiometer right current value  
open file pre-recorded House of God - Kahli ABDU featuring Teddy TEMZ  
call setEffect  
set volume to volume  
start playing  
Arduino loop forever:
```

We need to create one more function.

Let's call it "draw" since this function will print the effect names and intensity on the display.

After you create the function, it will appear in the sketch.

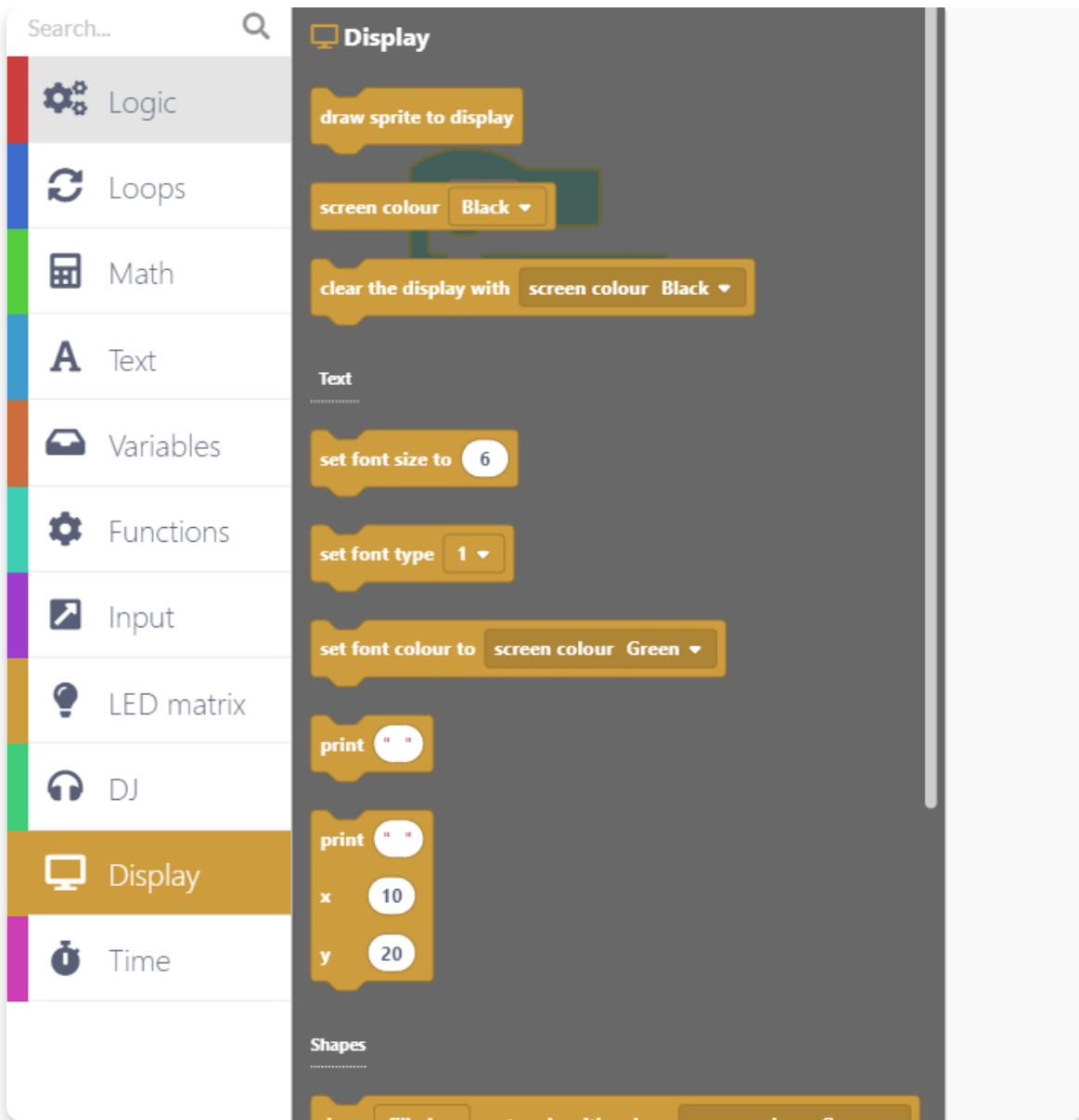


Don't forget to drop the "call draw" block here:

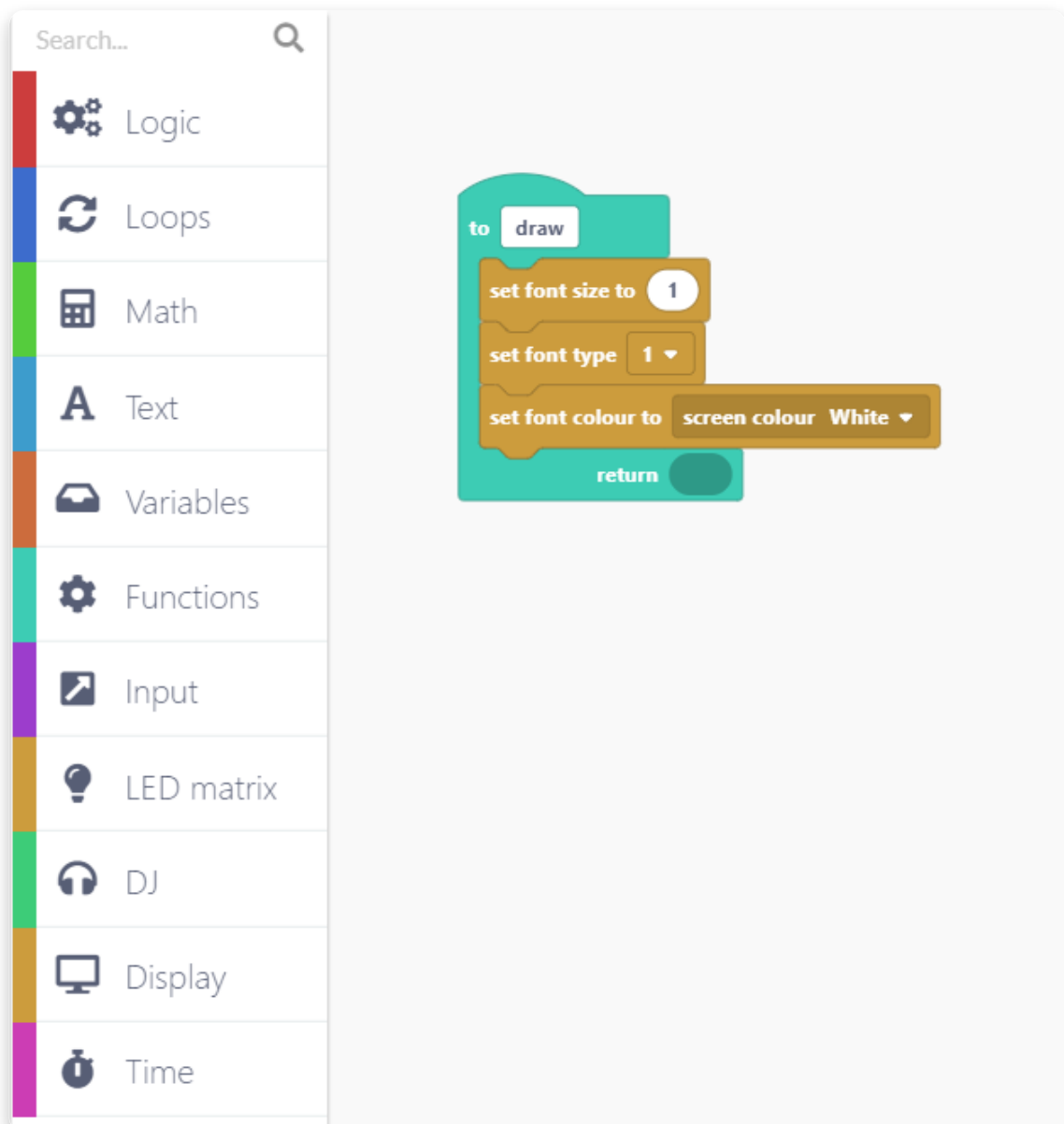


Let's edit the "draw" function now.

We'll use the blocks in the "Display" section to set the font size, type, and color.



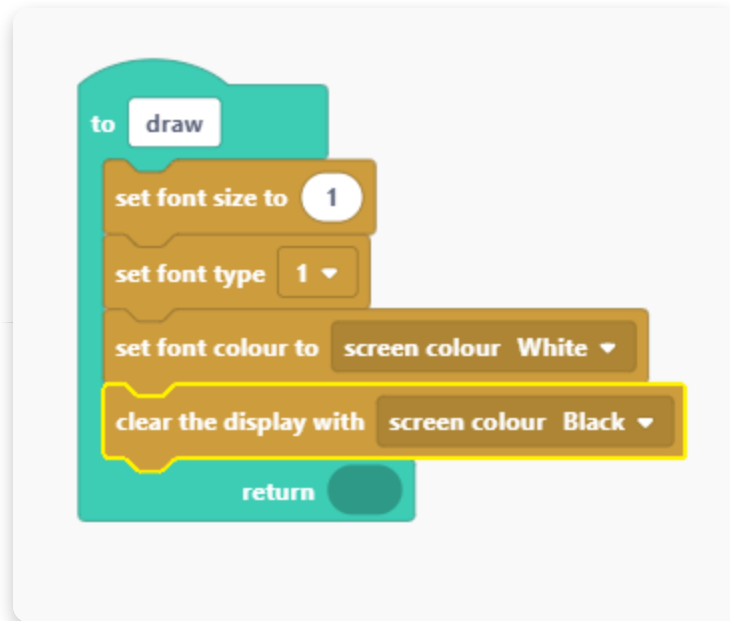
Set the following options for the text on the screen:



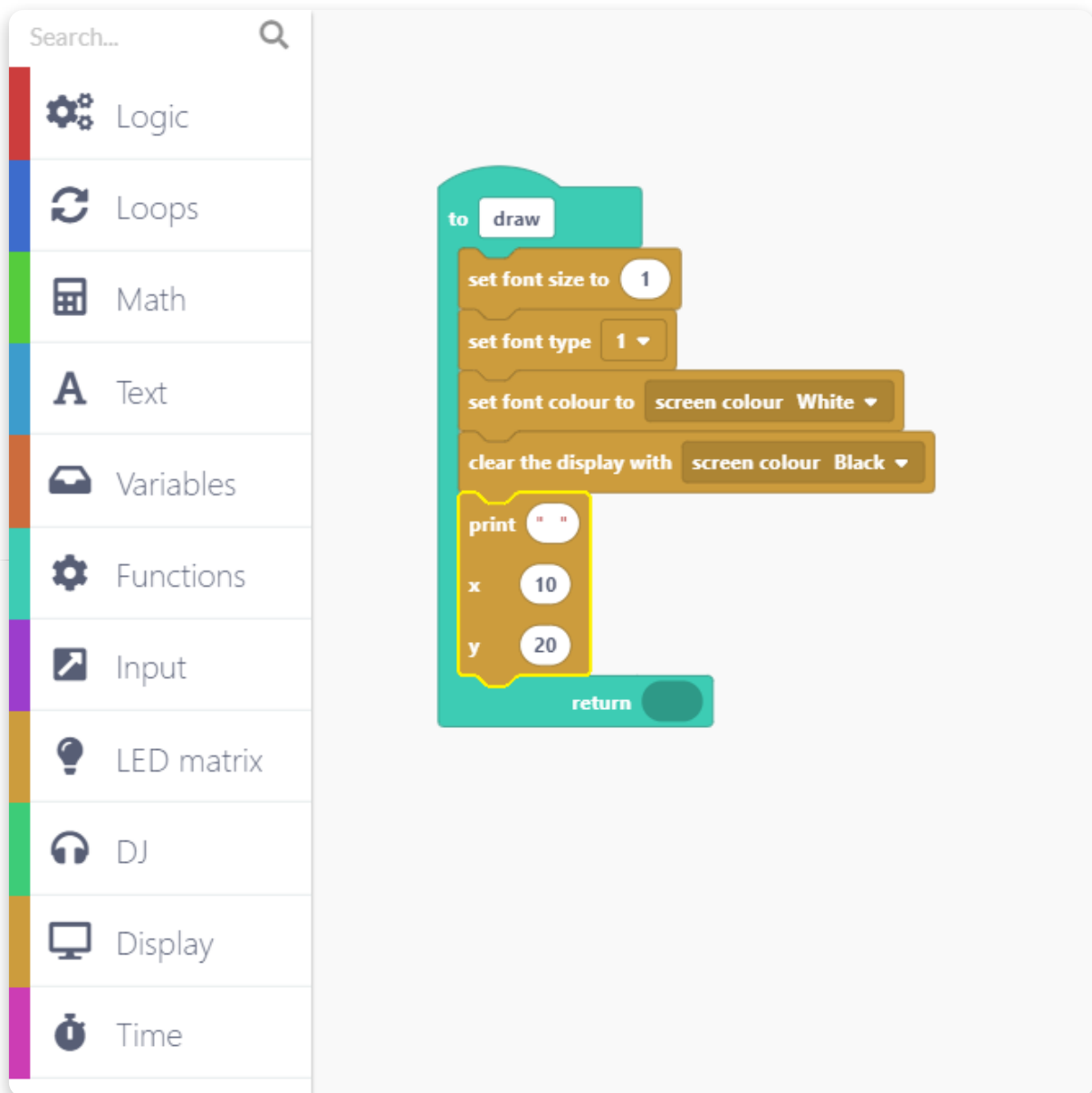
Since we chose the white font color, let's set the background to black.

Go to the "Display" section and find the block that says "clear the display with screen colour Black".

Put the block here:

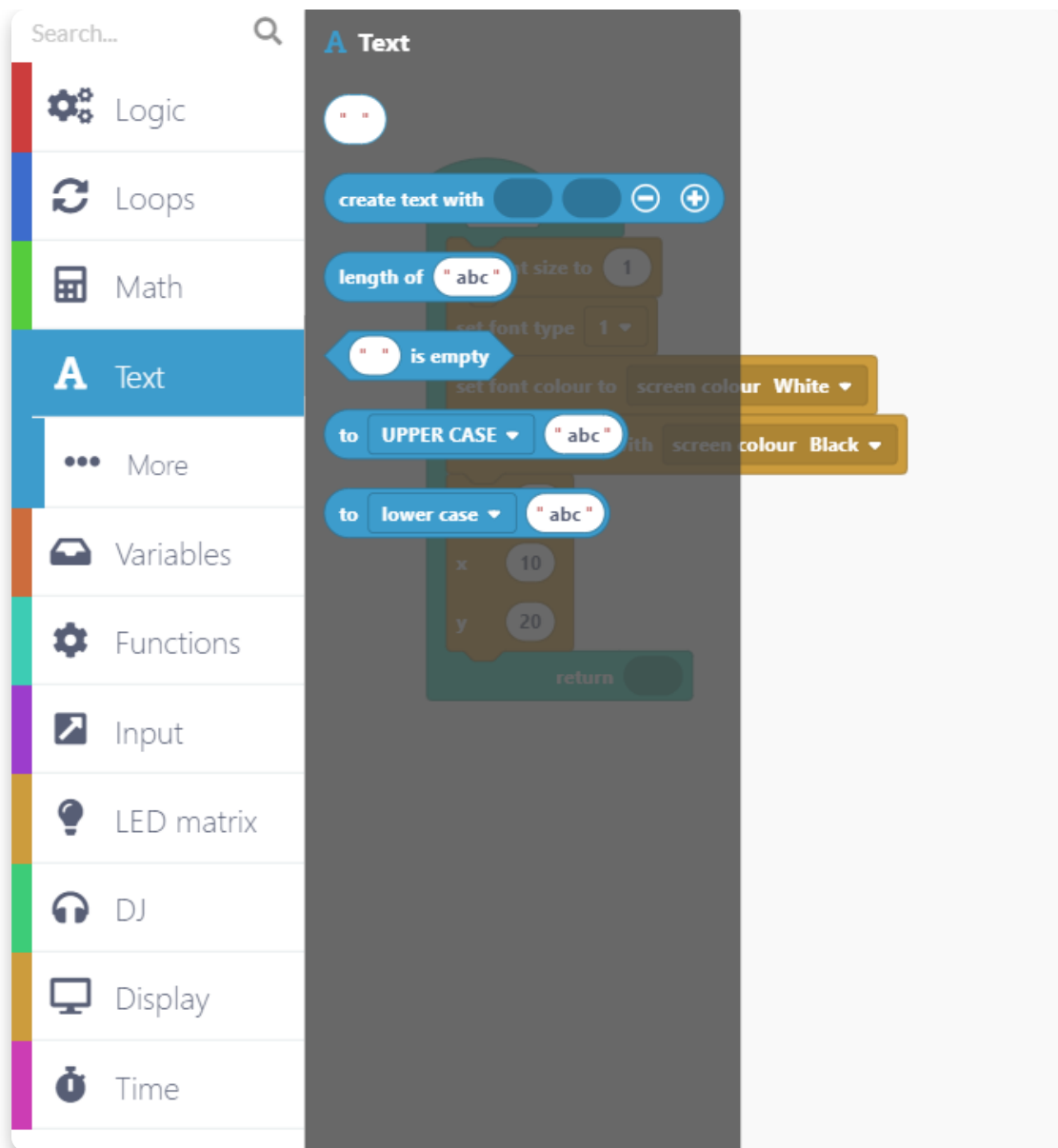


Go to the "Display" section and find the print block:

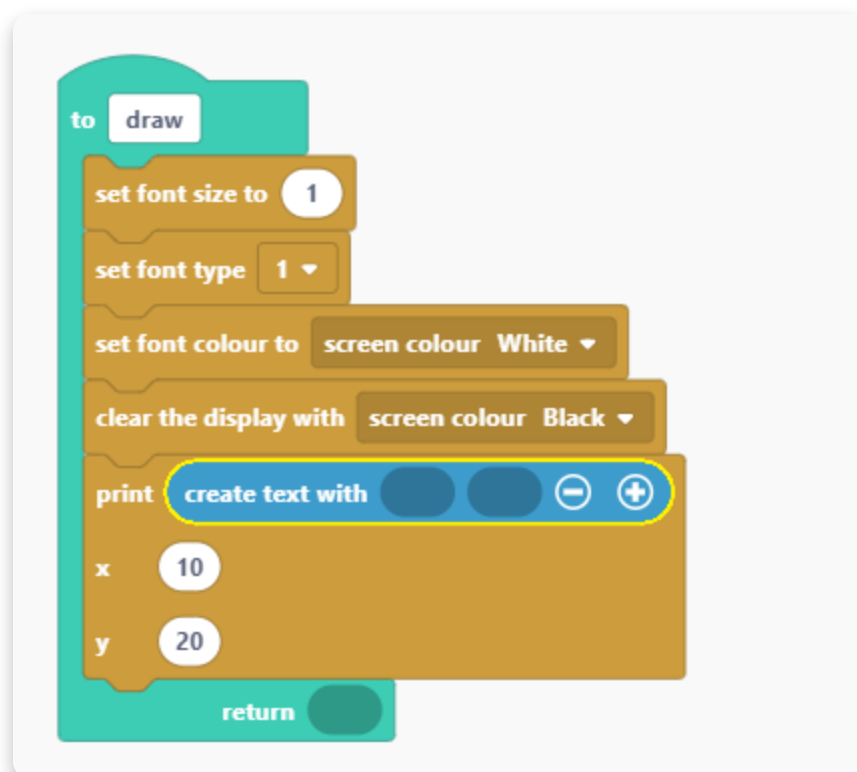


We're going to edit this so that it prints the word Volume and the volume value as well.

In the "Text" section, find the block that says "create text with".

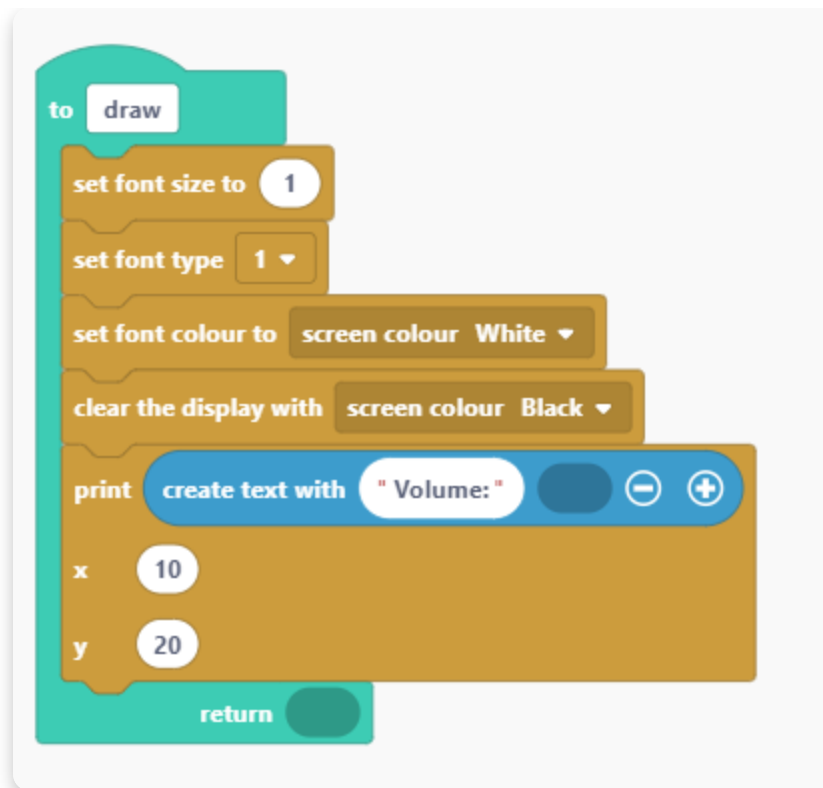


Put this block here:



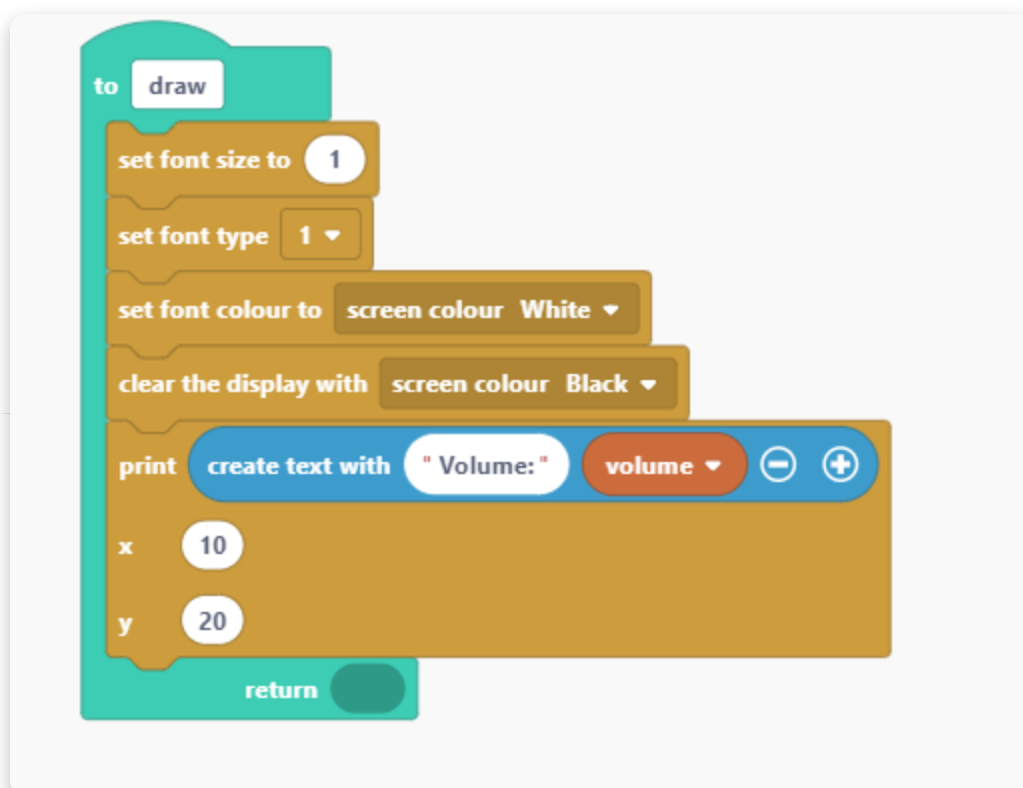
Open the "Text" section again and drag and drop the blank text block in the first window of the previous text block that's in the sketch.

Write "Volume:" in the text box like this:



Put the volume variable in the next window in the text box. This will print the actual volume value.

x and y are the coordinates of the text "Volume:". Set the x to 10 and y to 20.



To print the effect names, we'll have to use the logic function. This step is similar to the one where we assigned the numbers to the effects.

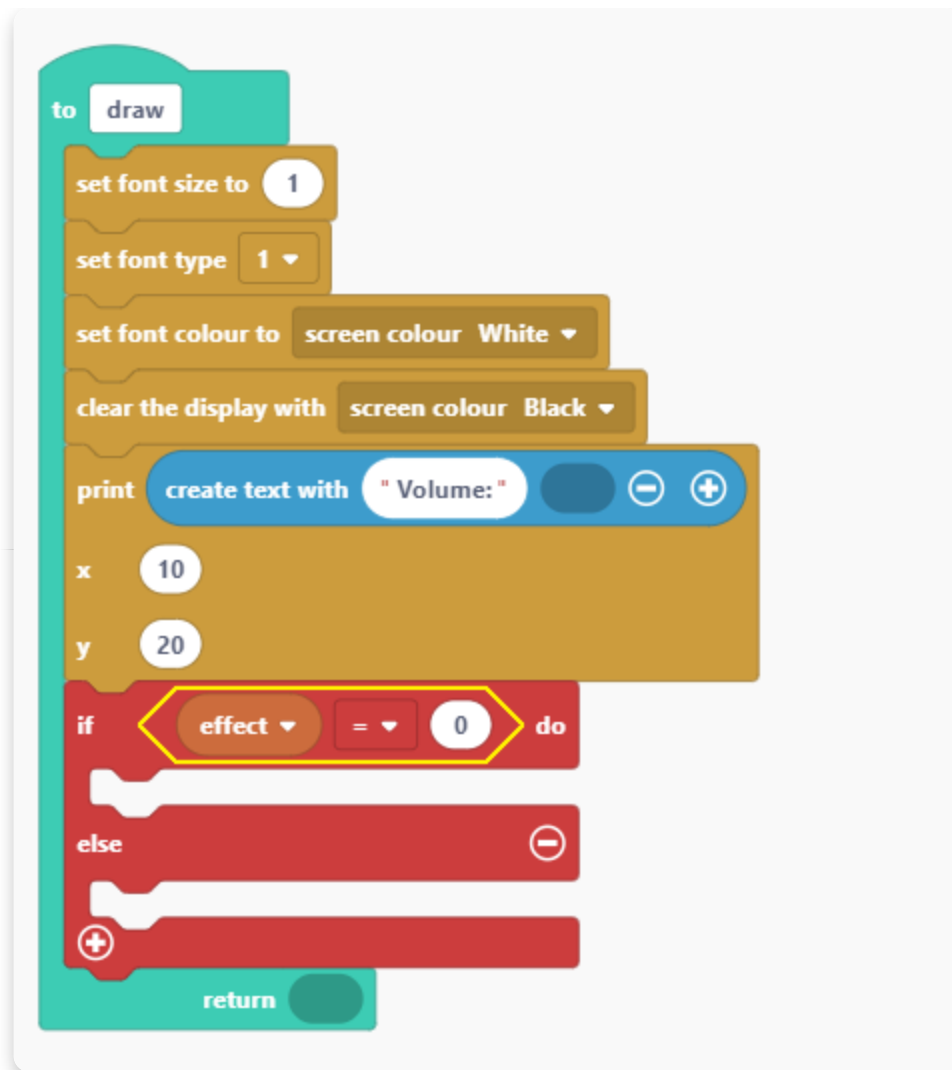
Put this logical function here:

```
to draw
  set font size to 1
  set font type 1
  set font colour to screen colour White
  clear the display with screen colour Black
  print create text with "Volume:"
  x 10
  y 20
  if true do
  else
  return
```

Instead of "true", let's use the comparison block.

```
to draw
  set font size to 1
  set font type 1
  set font colour to screen colour White
  clear the display with screen colour Black
  print create text with "Volume:"
  x 10
  y 20
  if 10 = 10 do
  else
  return
```

Insert the effect variable in the first window and type 0 in the second window.

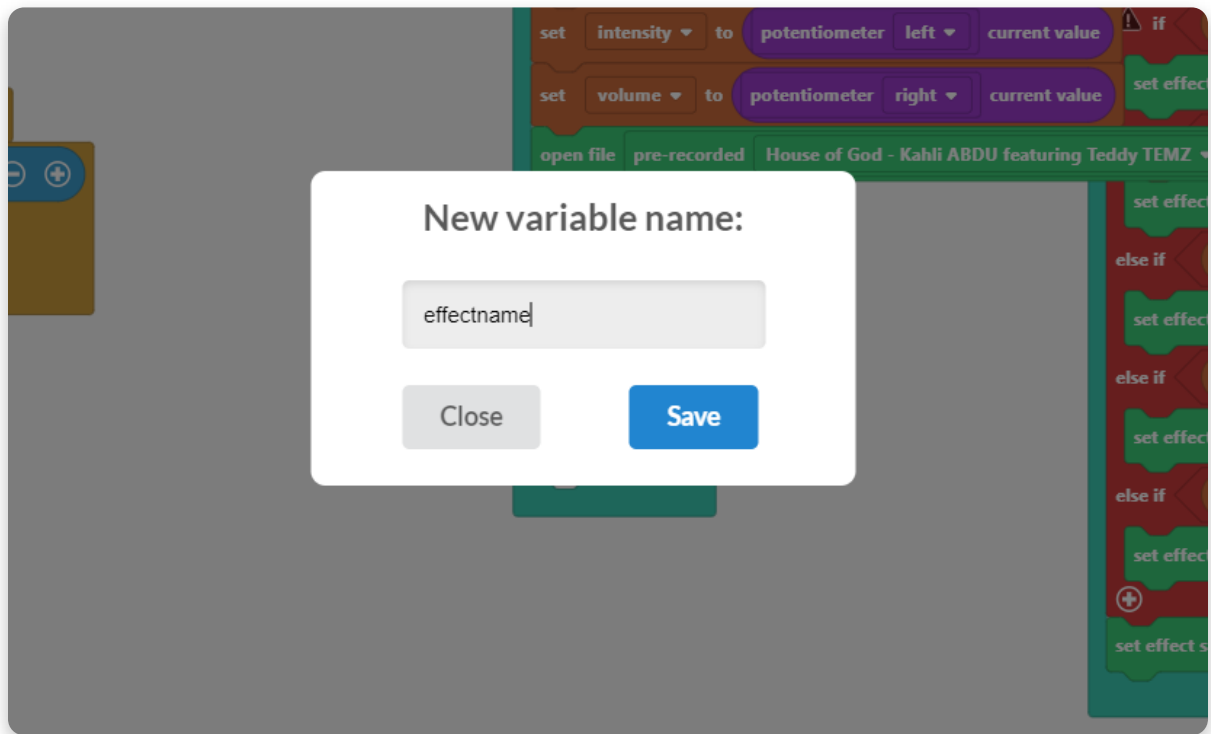
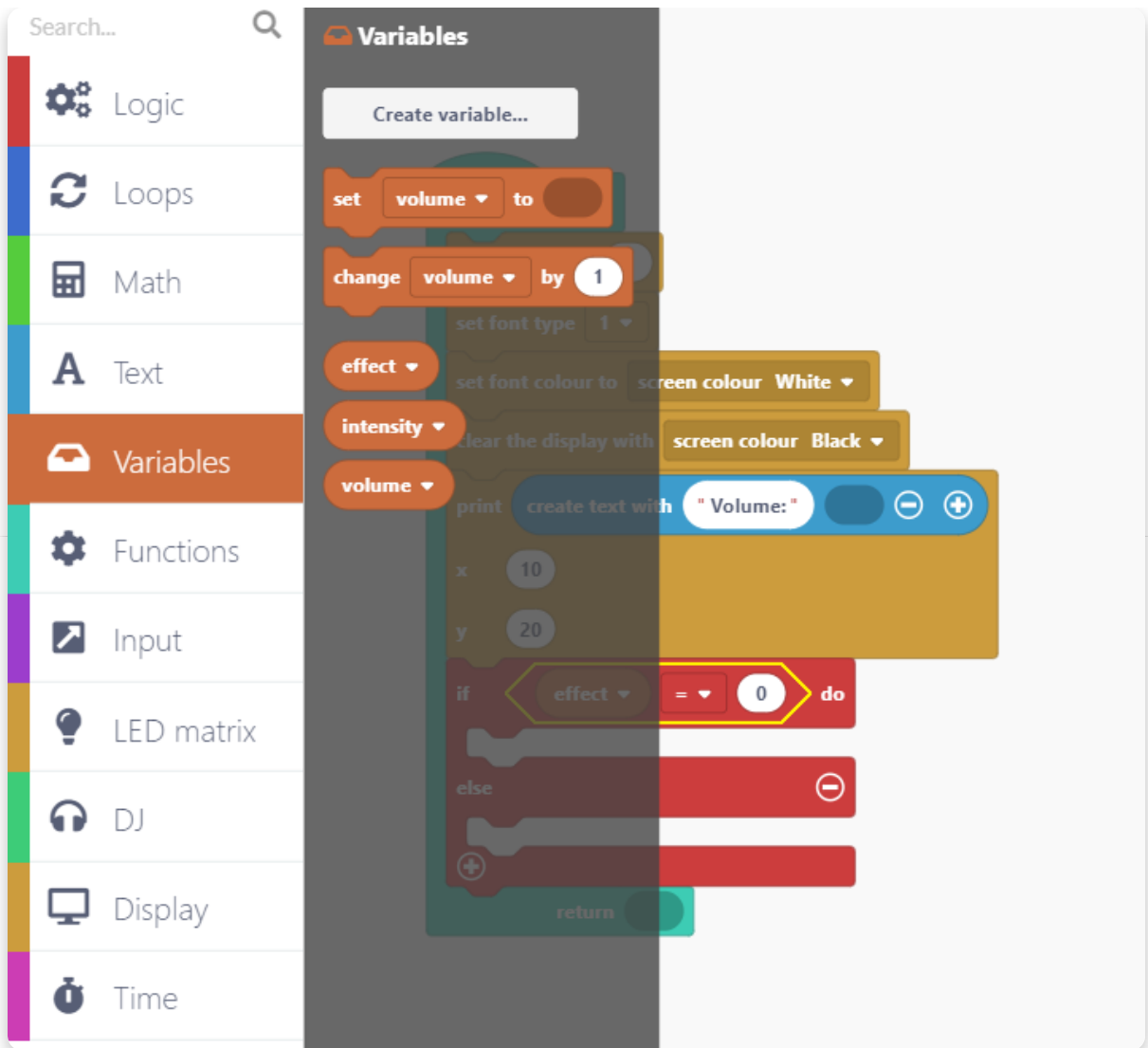


This means that if the effect is set to 0, then the display needs to show an indicator that there is no effect.

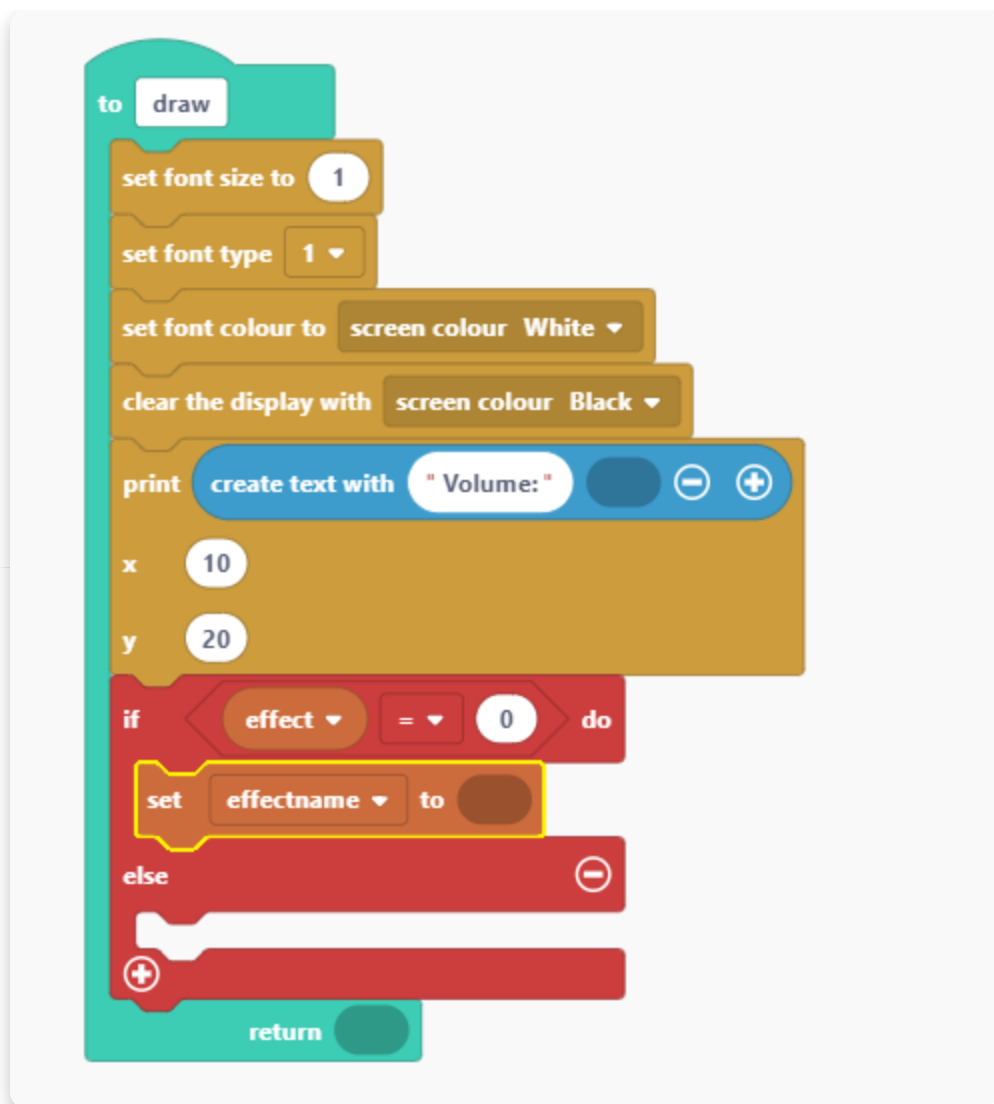
Let's add this condition now.

We need a new variable "effectname" that we'll later use in the functions for printing the effect names.

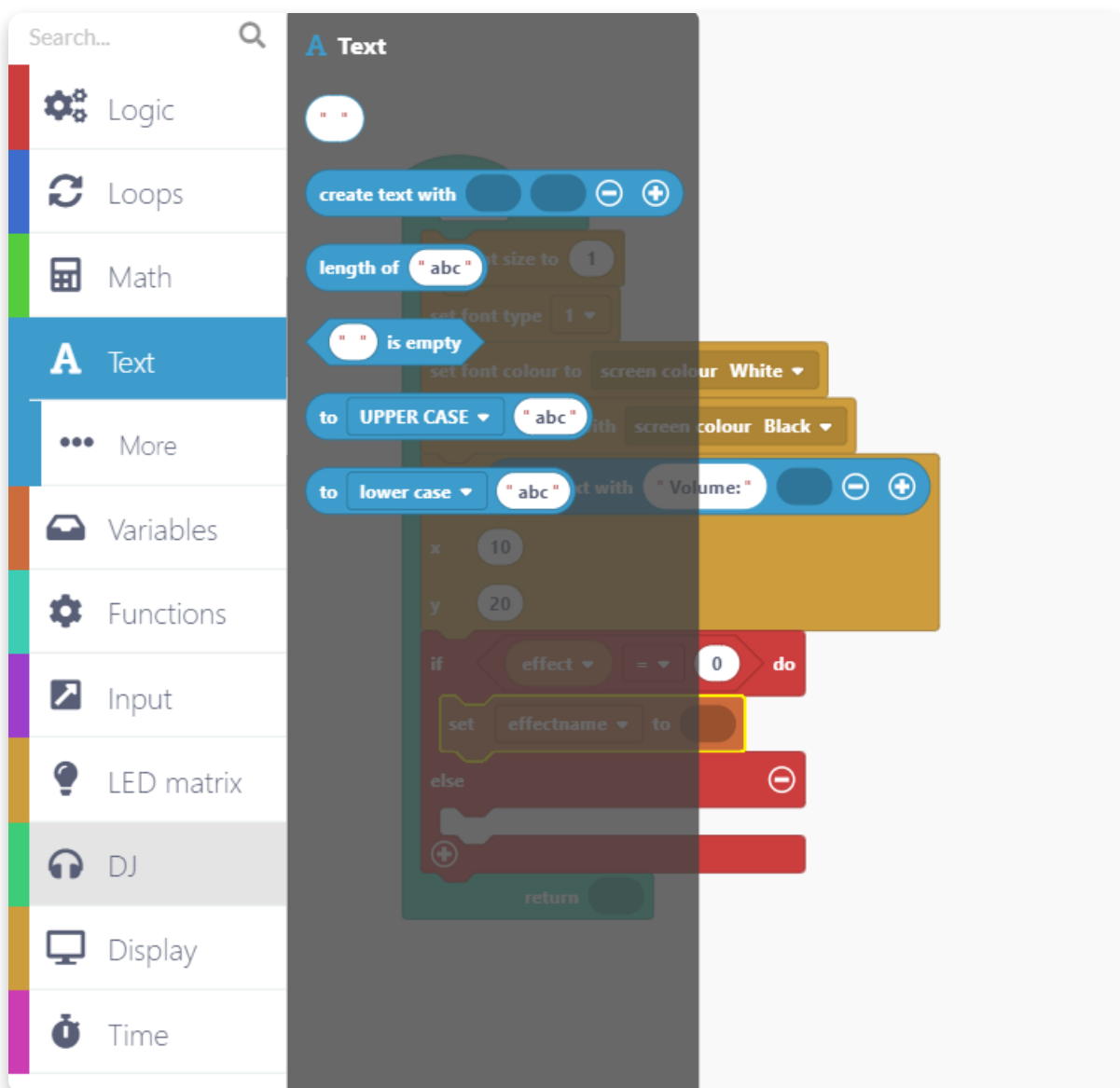
Create the variable:



Now add the "set effectname to" the block here:



Go to the "Text" section and find the blank text box block:



Add the text block and write "none" in it. This indicates that if the effect is set to 0, the effect is none.

```
to draw
  set font size to 1
  set font type 1
  set font colour to screen colour White
  clear the display with screen colour Black
  print create text with " Volume:"
  x 10
  y 20
  if effect = 0 do
    set effectname to "none"
  else
  return
```

Now we need to repeat this step for all the names of the effects.

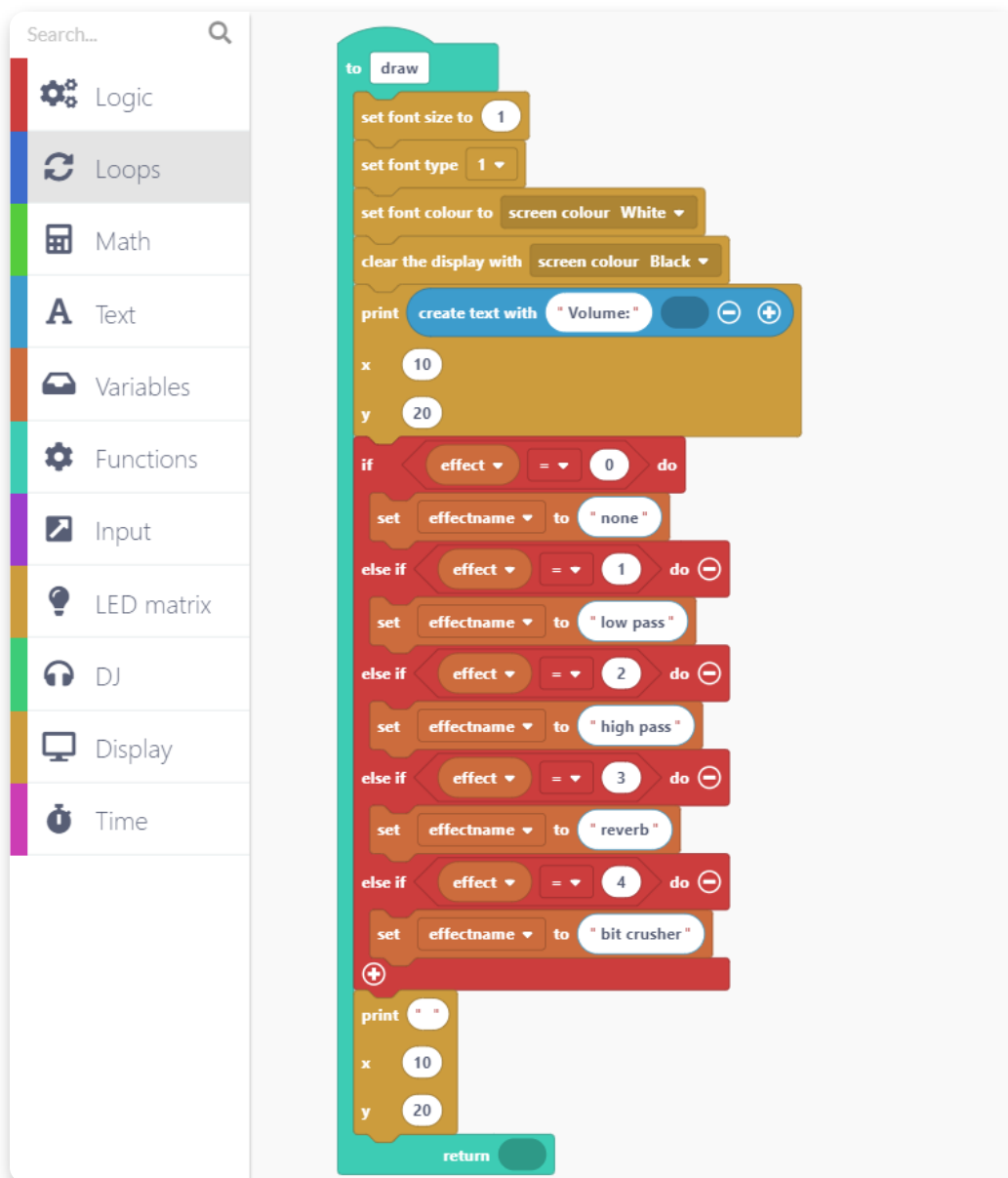
Extend the block by clicking on the plus sign at the bottom and close the block at the end by clicking on the small minus sign.

```
Search...
Logic
Loops
Math
Text
Variables
Functions
Input
LED matrix
DJ
Display
Time

to draw
  set font size to 1
  set font type 1
  set font colour to screen colour White
  clear the display with screen colour Black
  print create text with " Volume:"
  x 10
  y 20
  if effect = 0 do
    set effectname to "none"
  else if effect = 1 do
    set effectname to "low pass"
  else if effect = 2 do
    set effectname to "high pass"
  else if effect = 3 do
    set effectname to "reverb"
  else if effect = 4 do
    set effectname to "bit crusher"
  return
```

When you entered all the logical functions in this block, it's time to add the block to print the effect name on display.

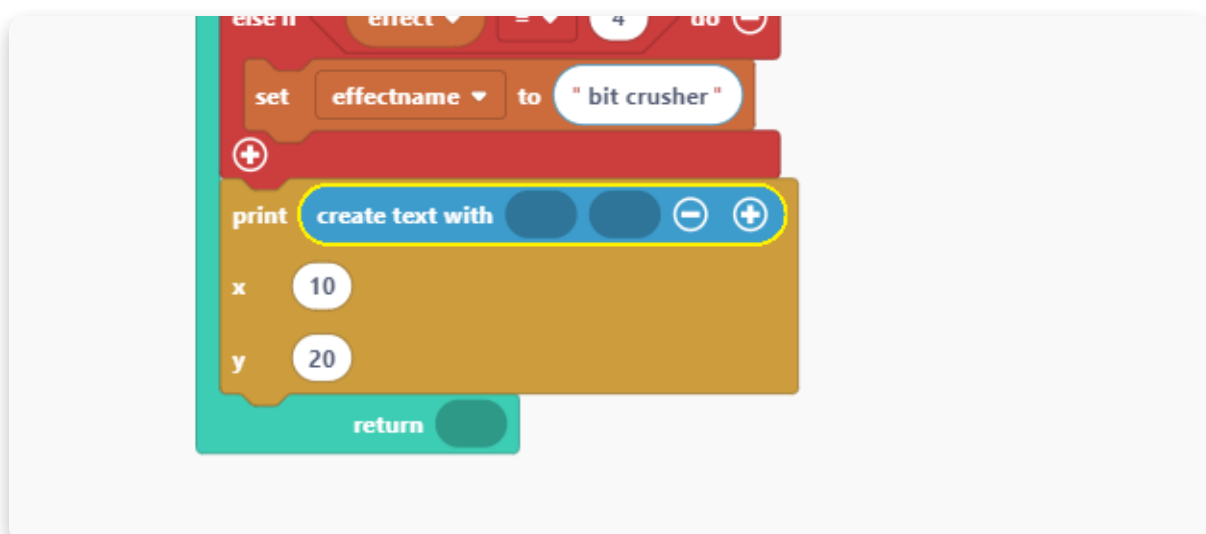
Go to the "Display" section and use this block:



```
to draw
  set font size to 1
  set font type to 1
  set font colour to screen colour White
  clear the display with screen colour Black
  print create text with " Volume: "
  x 10
  y 20
  if effect = 0 do
    set effectname to "none"
  else if effect = 1 do
    set effectname to "low pass"
  else if effect = 2 do
    set effectname to "high pass"
  else if effect = 3 do
    set effectname to "reverb"
  else if effect = 4 do
    set effectname to "bit crusher"
  print " "
  x 10
  y 20
return
```

To print the effect name, go to the "Text" section and choose the "create text with" block.

Put it here:



```
set effectname to "bit crusher"
+
print create text with
x 10
y 20
return
```

Add a text window from the "Text" section and write "Effect:".

```
else if effect = 4 do
  set effectname to "bit crusher"
  print create text with "Effect"
  x 10
  y 20
return
```

Put the "effectname" variable in the next window in the block.

Also, set the coordinates of the text.

Set the x to 10 and y to 40.

```
else if effect = 4 do
  set effectname to "bit crusher"
  print create text with "Effect" effectname
  x 10
  y 40
return
```

This function should now print the effect name on display after it prints the volume value.

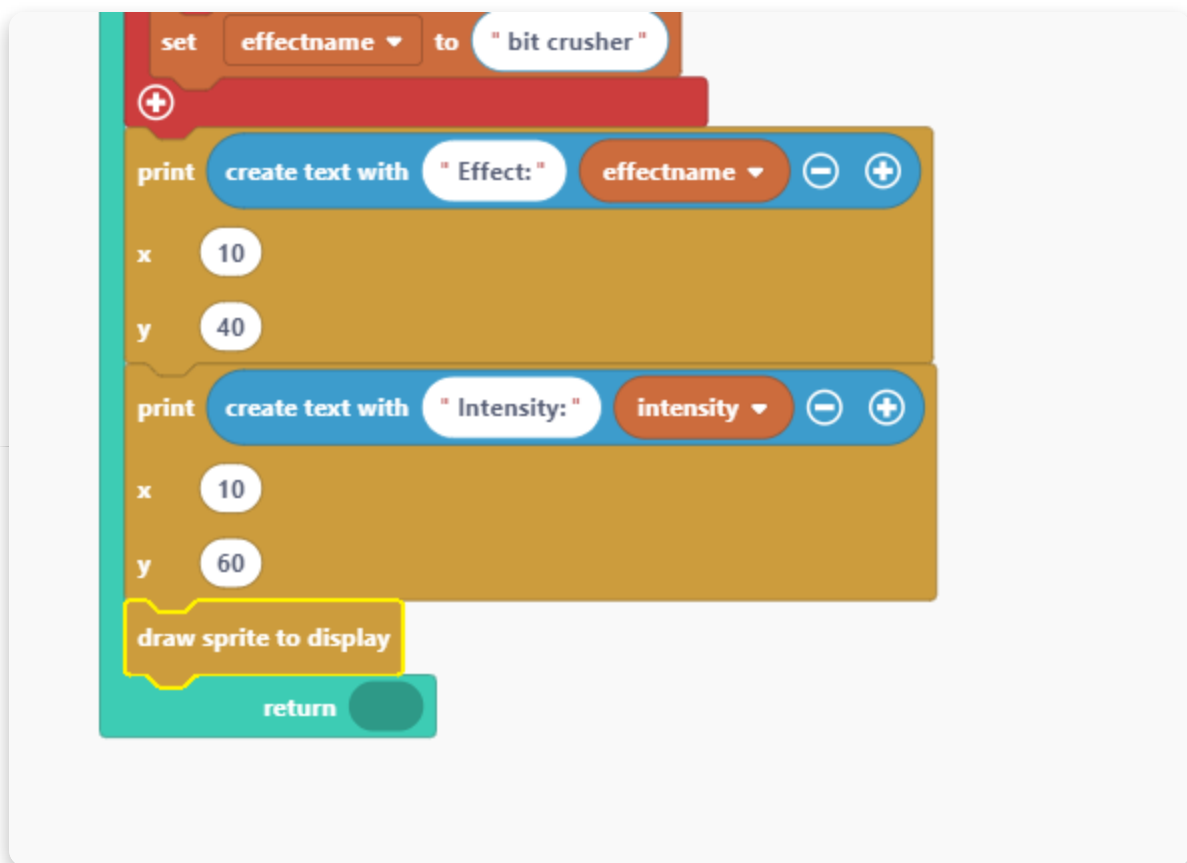
All it's left is to print the intensity value.

Put another "print" block and edit it like this:

```
else if effect = 4 do
  set effectname to "bit crusher"
  print create text with "Effect:" effectname
  x 10
  y 40
  print create text with "Intensity:" intensity
  x 10
  y 60
return
```

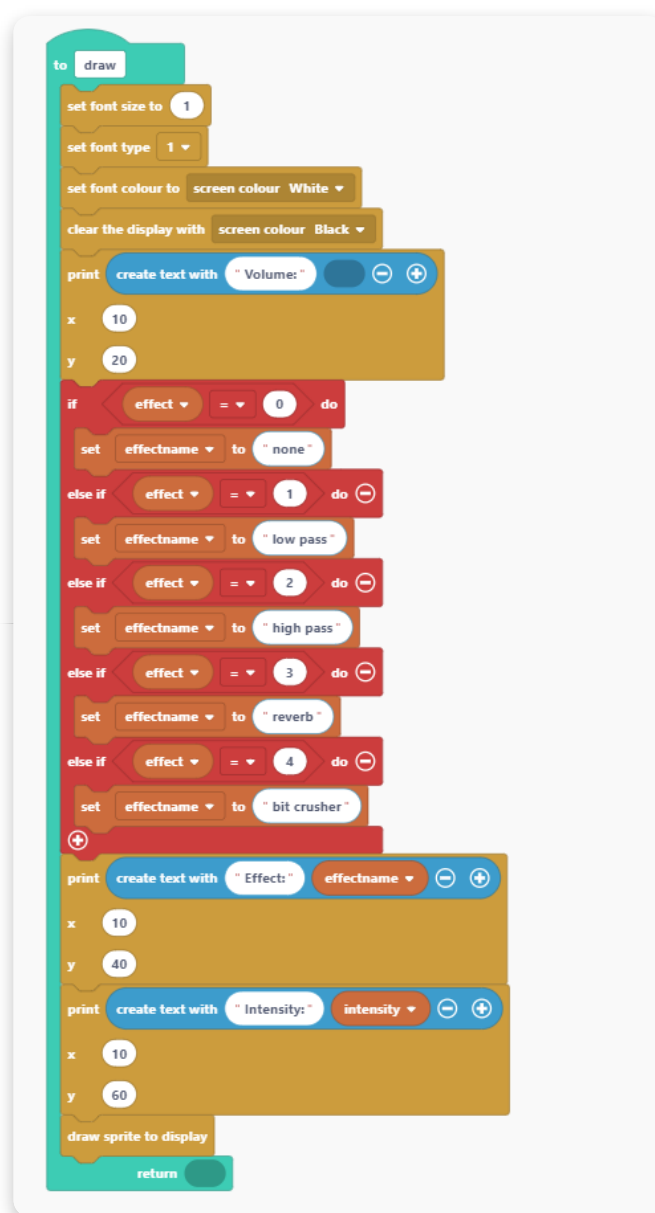
To finalize this "draw" function, we'll use the block "draw sprite to display".

You can find it in the "Display" section.



```
set effectname to "bit crusher"
print create text with "Effect:" effectname
x 10
y 40
print create text with "Intensity:" intensity
x 10
y 60
draw sprite to display
return
```

This is the finalized "draw" function:



```
to draw
set font size to 1
set font type 1
set font colour to screen colour White
clear the display with screen colour Black
print create text with "Volumes:"
x 10
y 20
if effect = 0 do
set effectname to "none"
else if effect = 1 do
set effectname to "low pass"
else if effect = 2 do
set effectname to "high pass"
else if effect = 3 do
set effectname to "reverb"
else if effect = 4 do
set effectname to "bit crusher"
print create text with "Effect:" effectname
x 10
y 40
print create text with "Intensity:" intensity
x 10
y 60
draw sprite to display
return
```

And this is what you should have up until now:

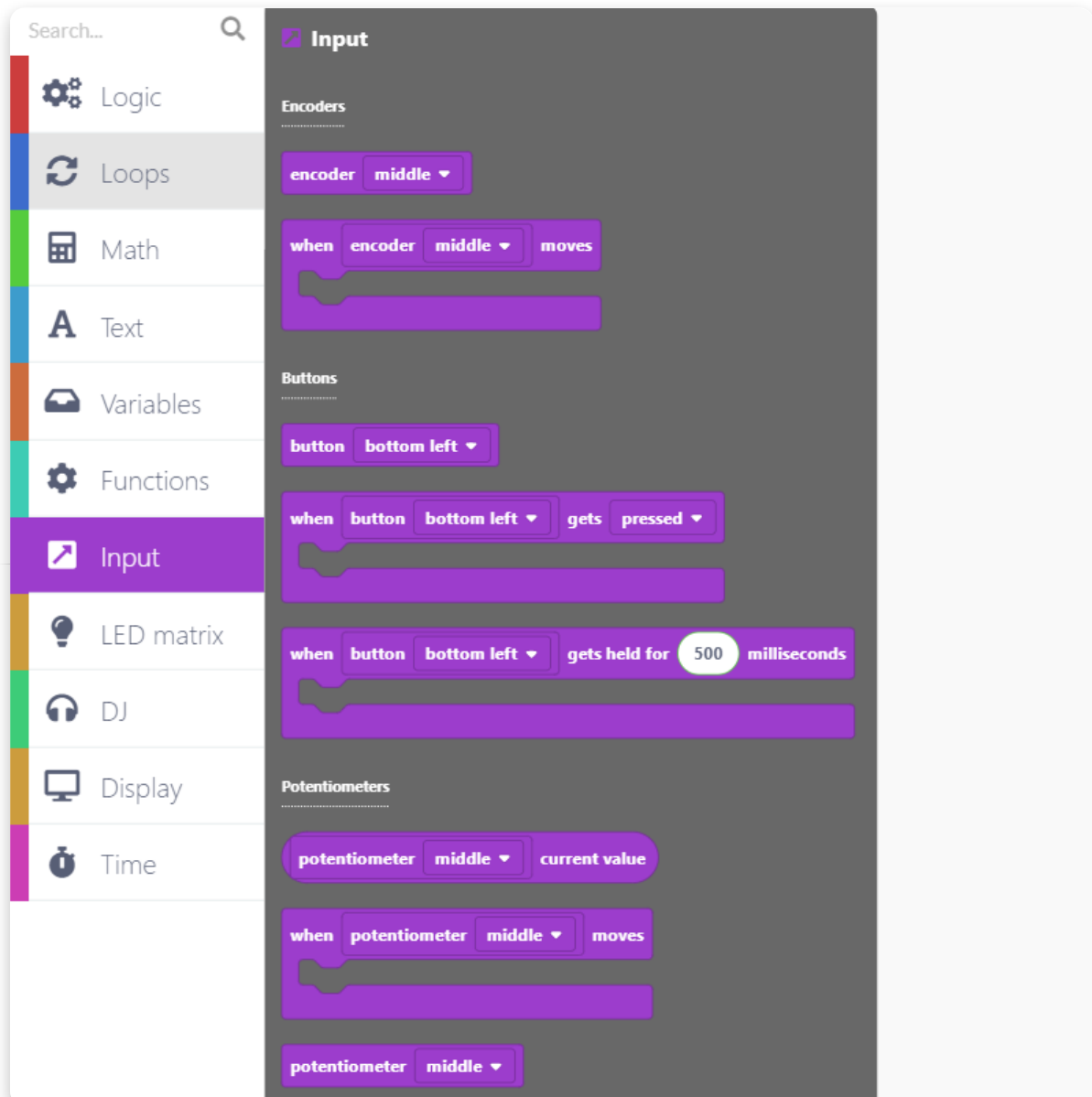


The only thing remaining now is to define the input functions for the middle encoder that will regulate the effect and potentiometers that will regulate volume and effect intensity.

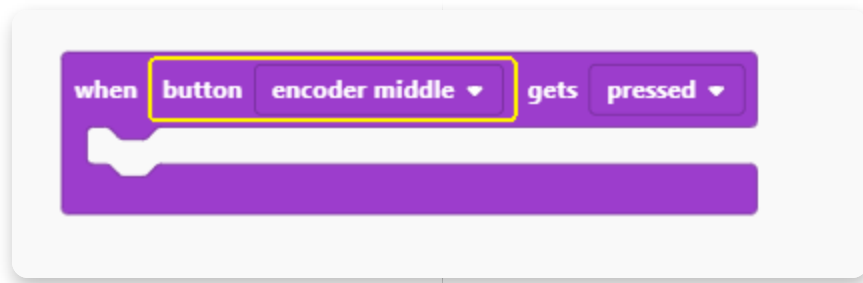
Let's start with changing the effects. We'll do this by pressing the middle encoder.

So, let's go to the "Input" section and find the block that says "when button bottom left gets pressed".

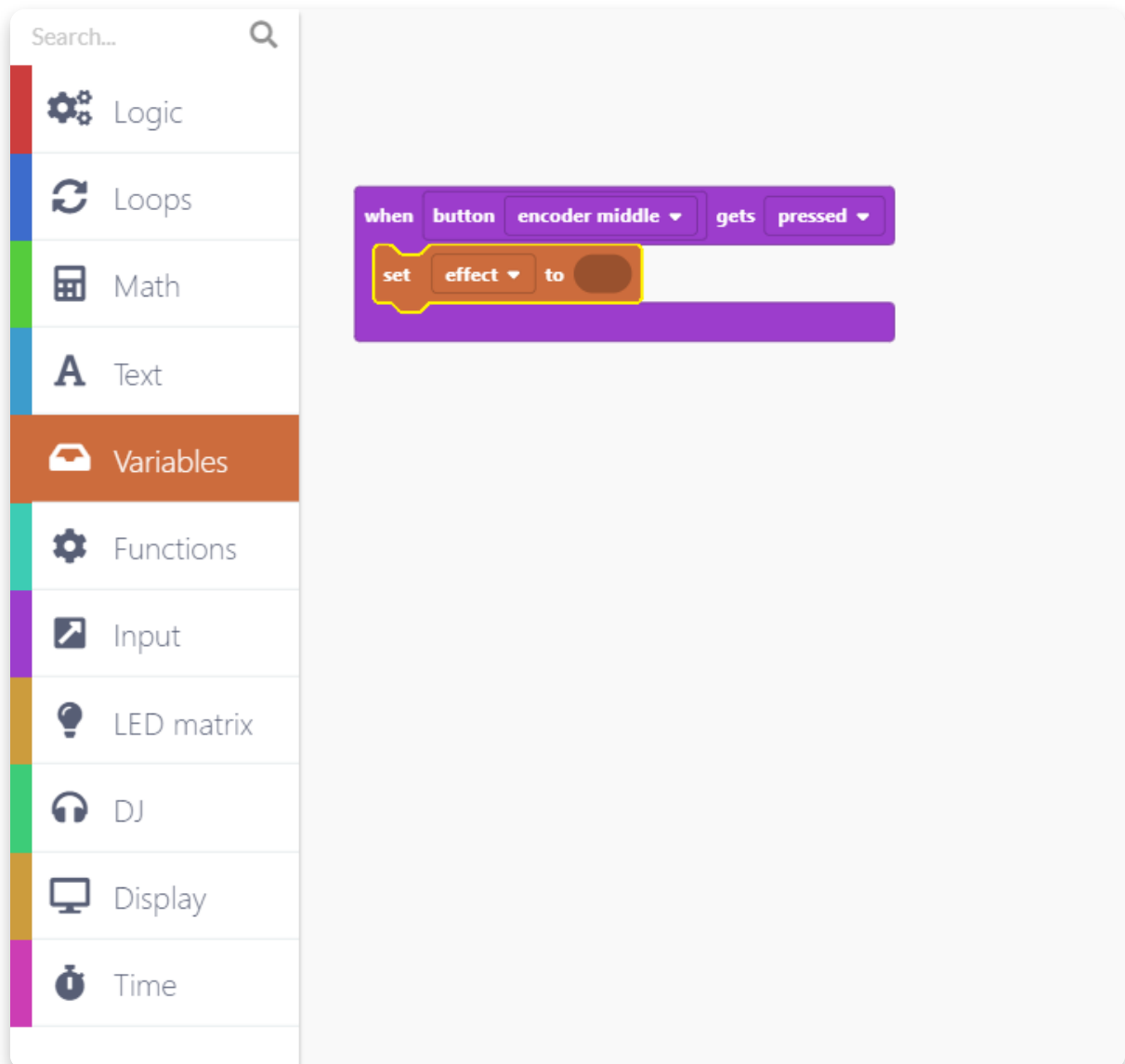
Drag and drop this block to the sketch.



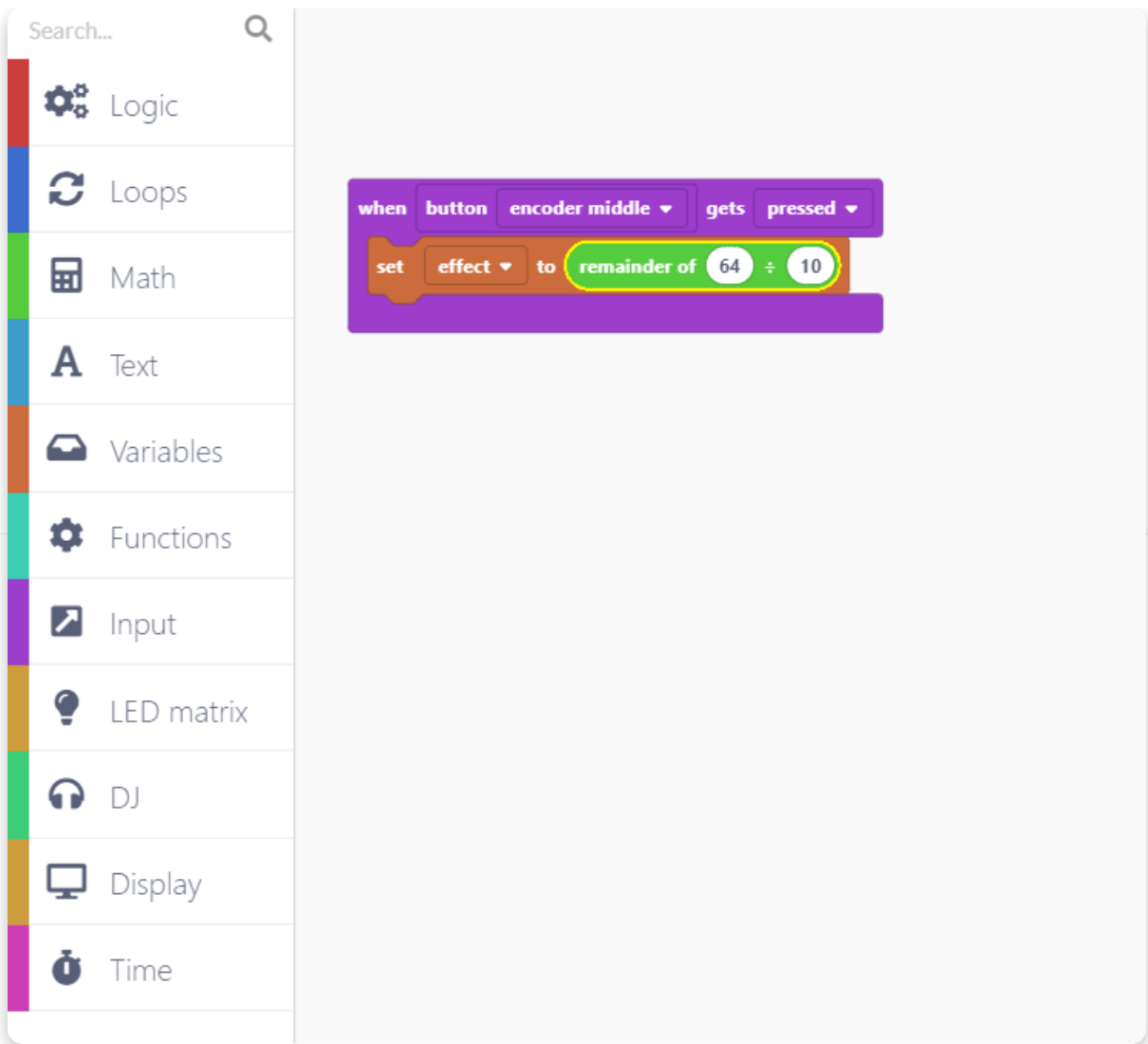
Choose the "encoder middle" option in the first drop-down menu and "pressed" in the second drop-down menu.



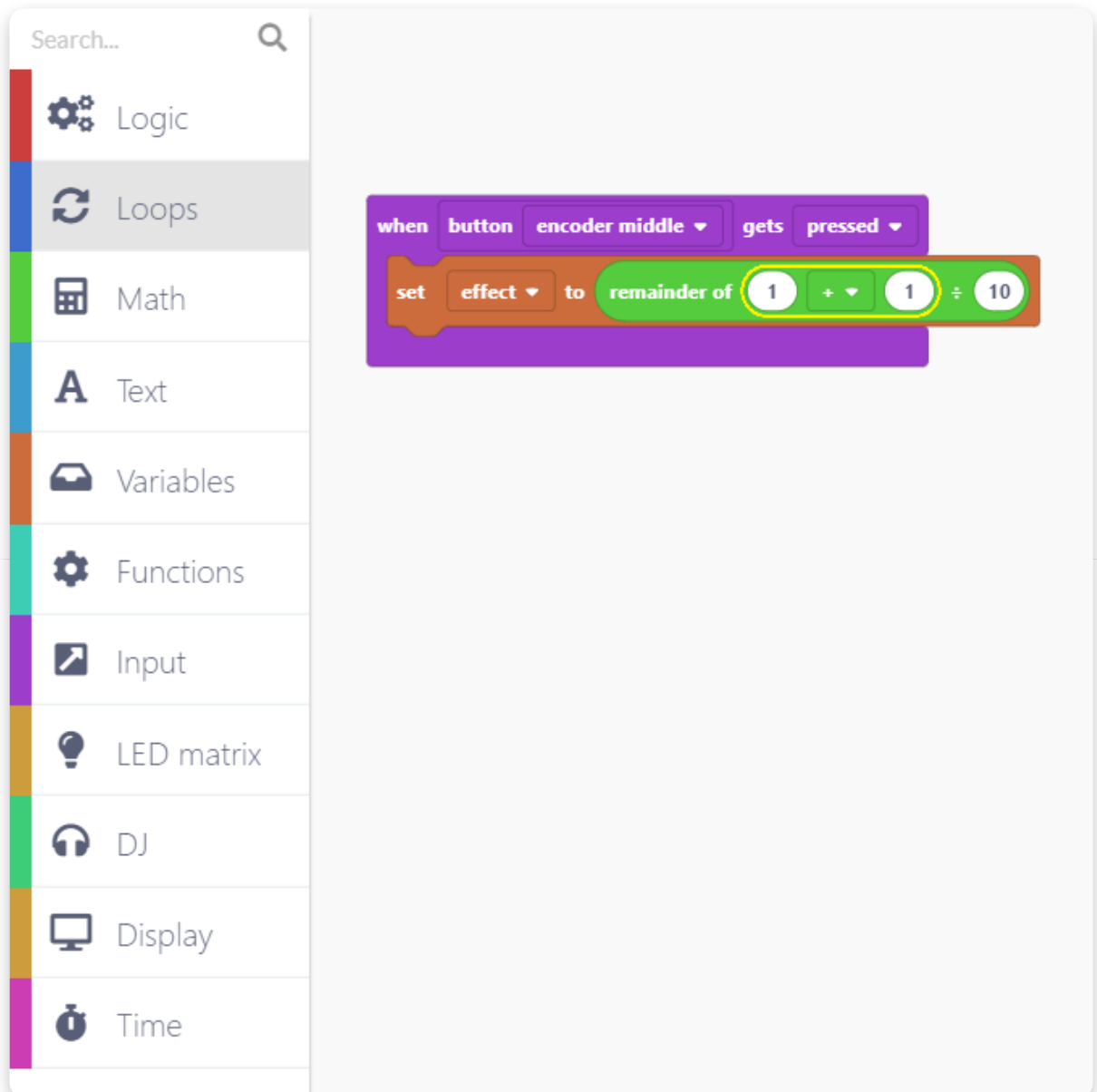
To choose an effect when pressing the middle encoder, use the following function from the "Variables" section:



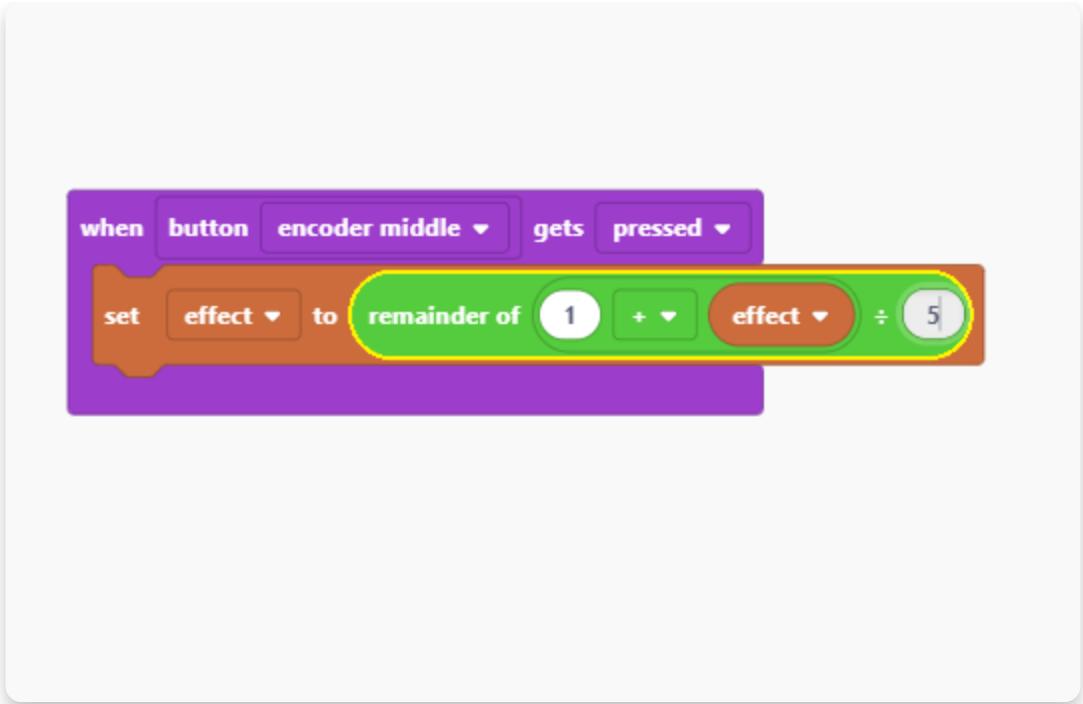
Since we have 5 options for effect (none, low pass, high pass, reverb, and bit crusher), we're going to use the "remainder" block from the "Math" section for dividing the number of times the middle encoder gets pressed by 5.



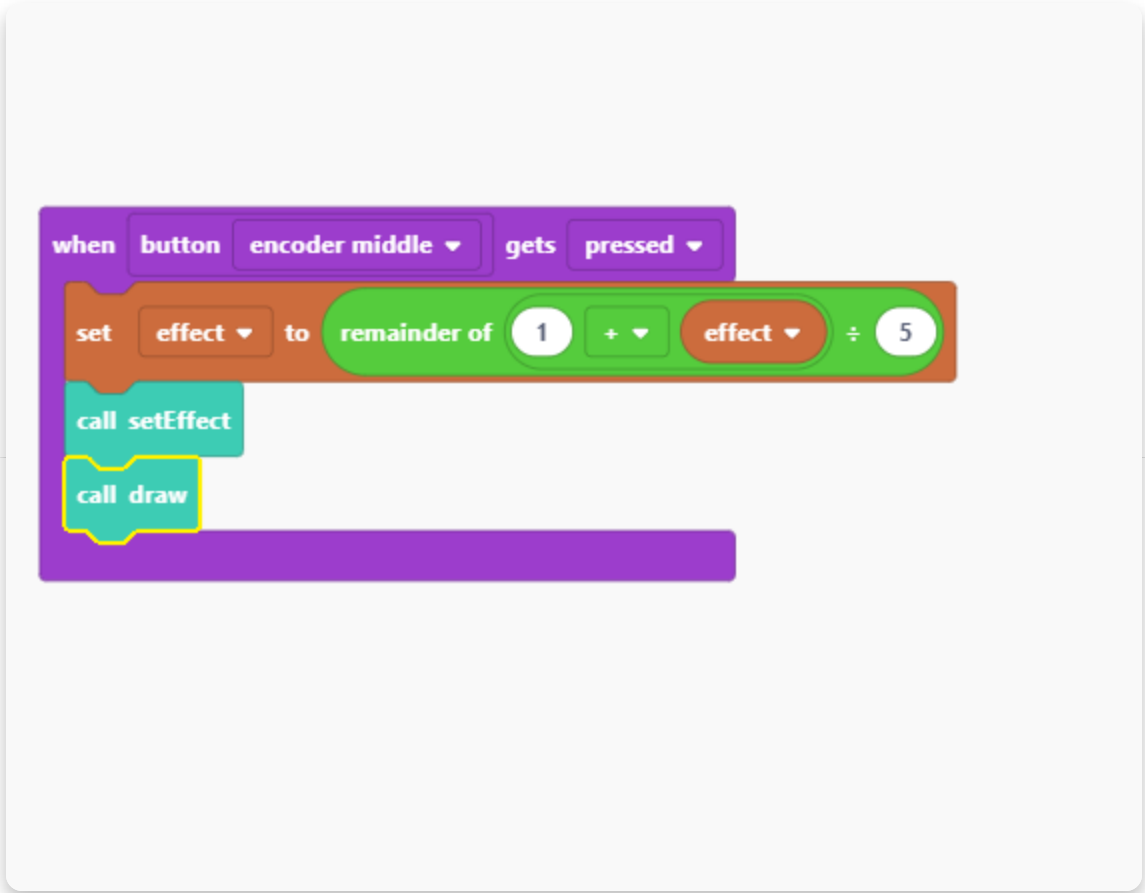
Add the division block in the first window of the remainder block:



Edit the block like this by adding the effect variable:



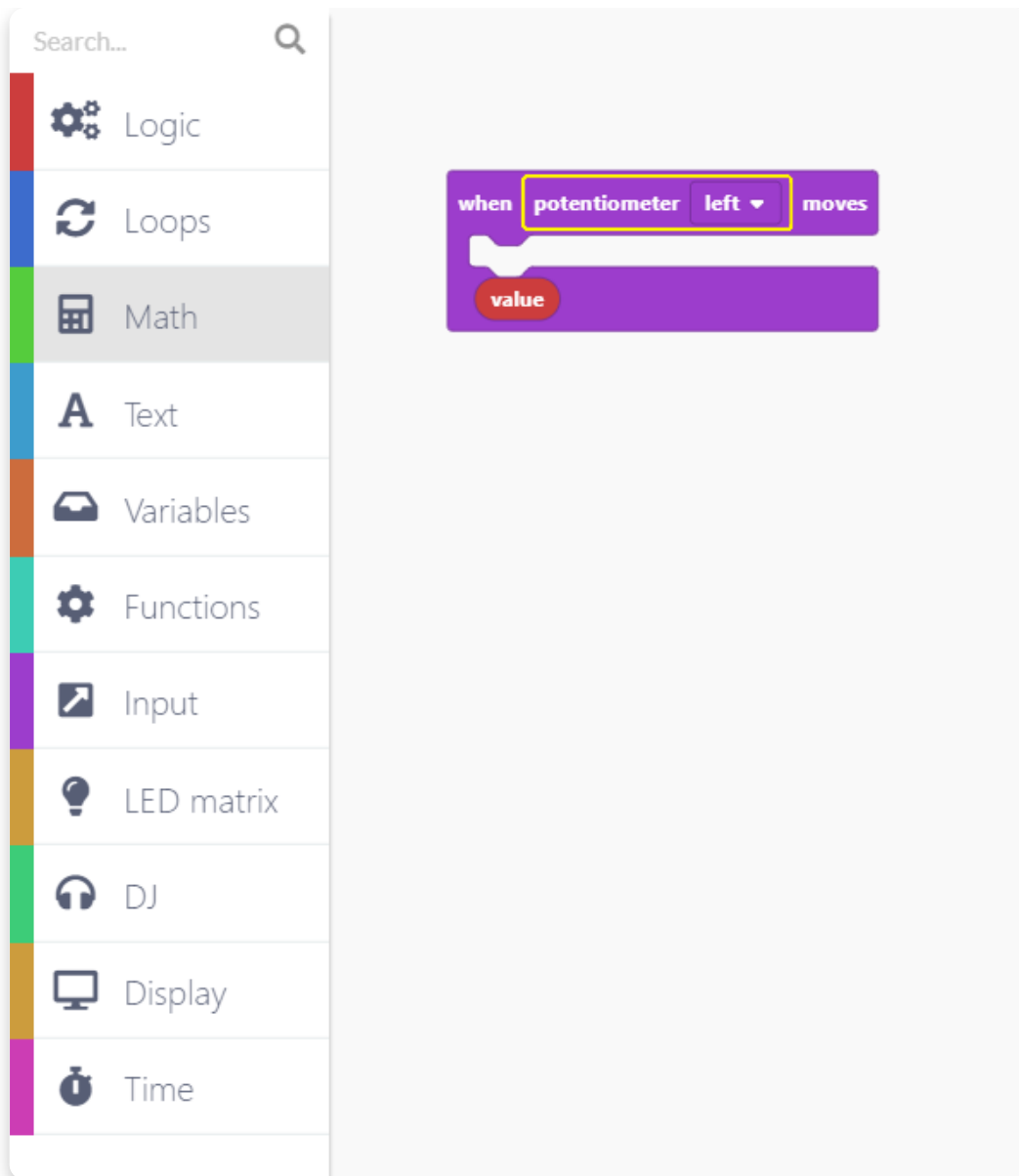
Now add two of the functions that we created previously: setEffect and draw.



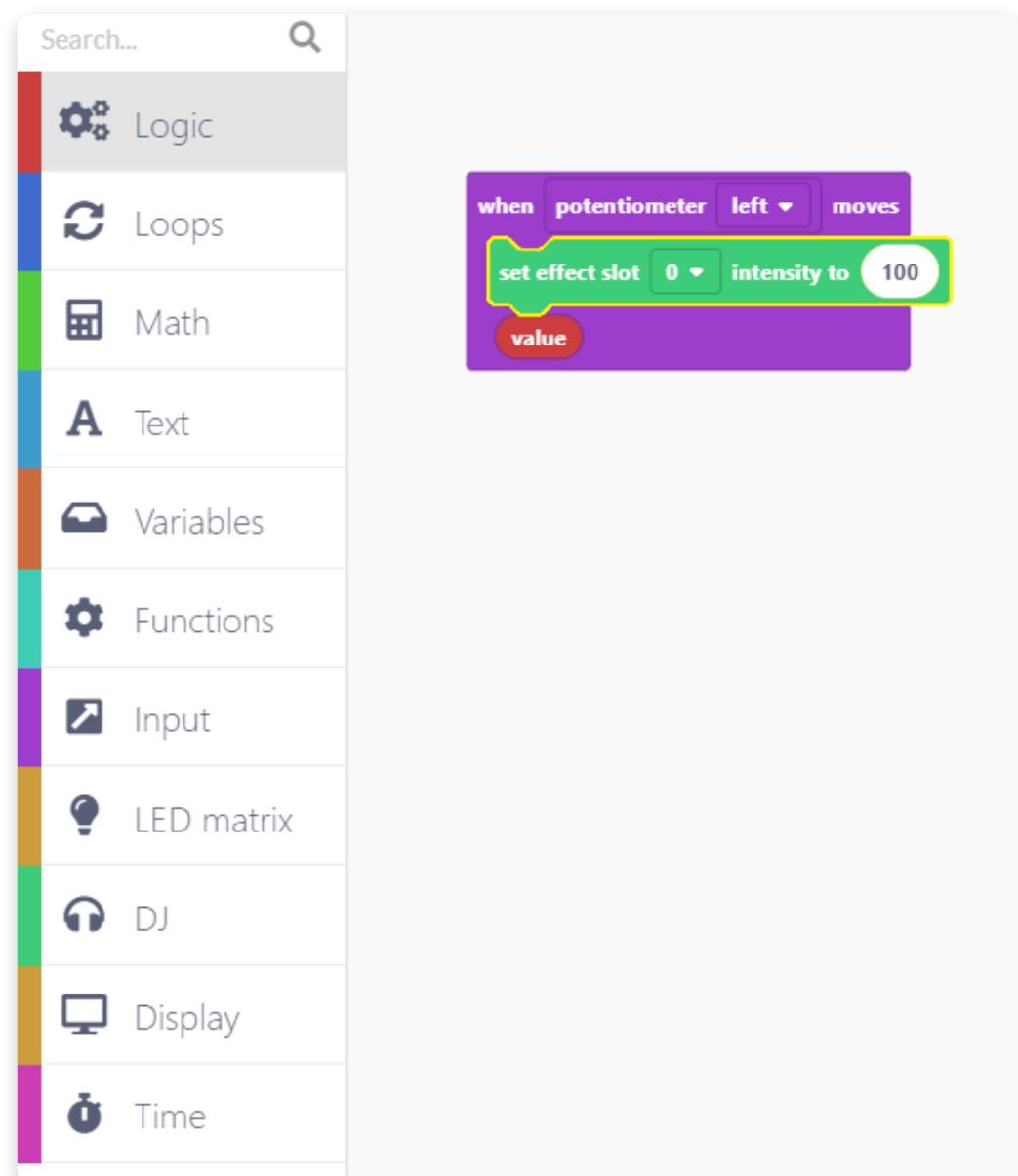
This "Input" is done.

Let's create one that will set the left potentiometer to control the intensity of the effect.

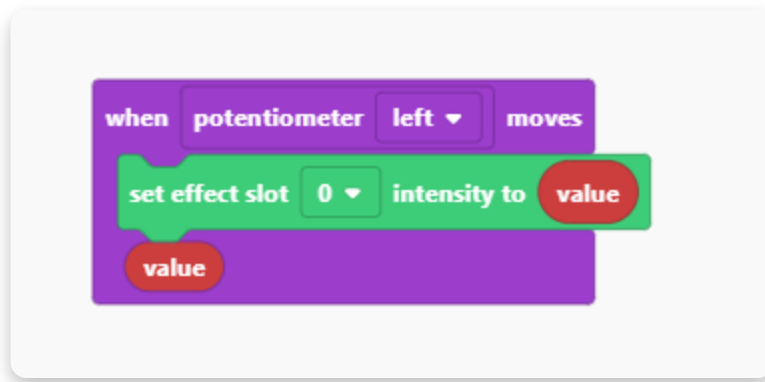
Find this potentiometer block in the "Input" section and edit it like this so it's set to the left side:



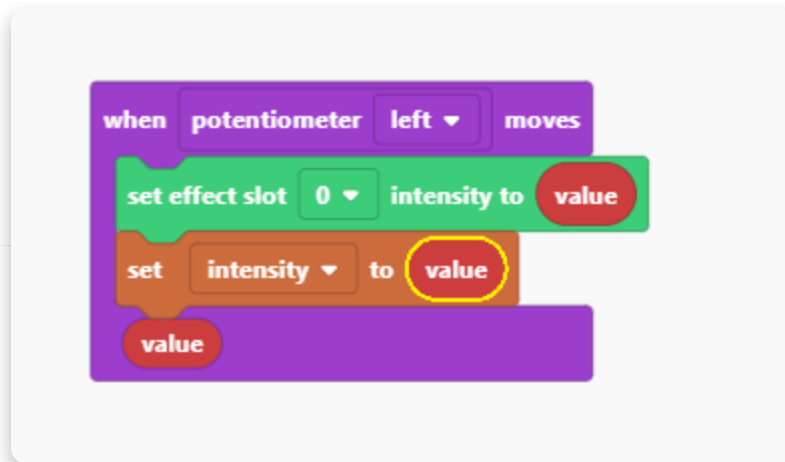
After this, add a block that is setting the intensity from the "DJ" section:



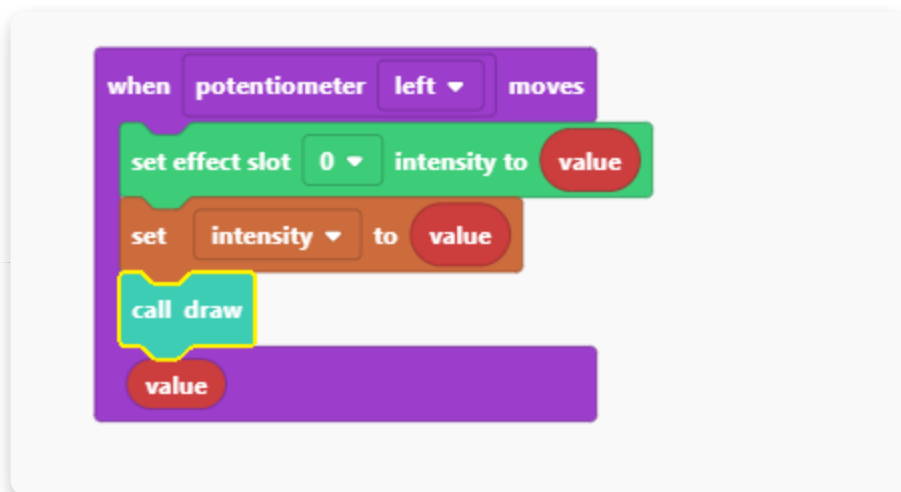
Drag the value block from the bottom of the Input block and put it in the window instead of the value 100.



Add the variable function that will set the intensity to the value that's set like this:

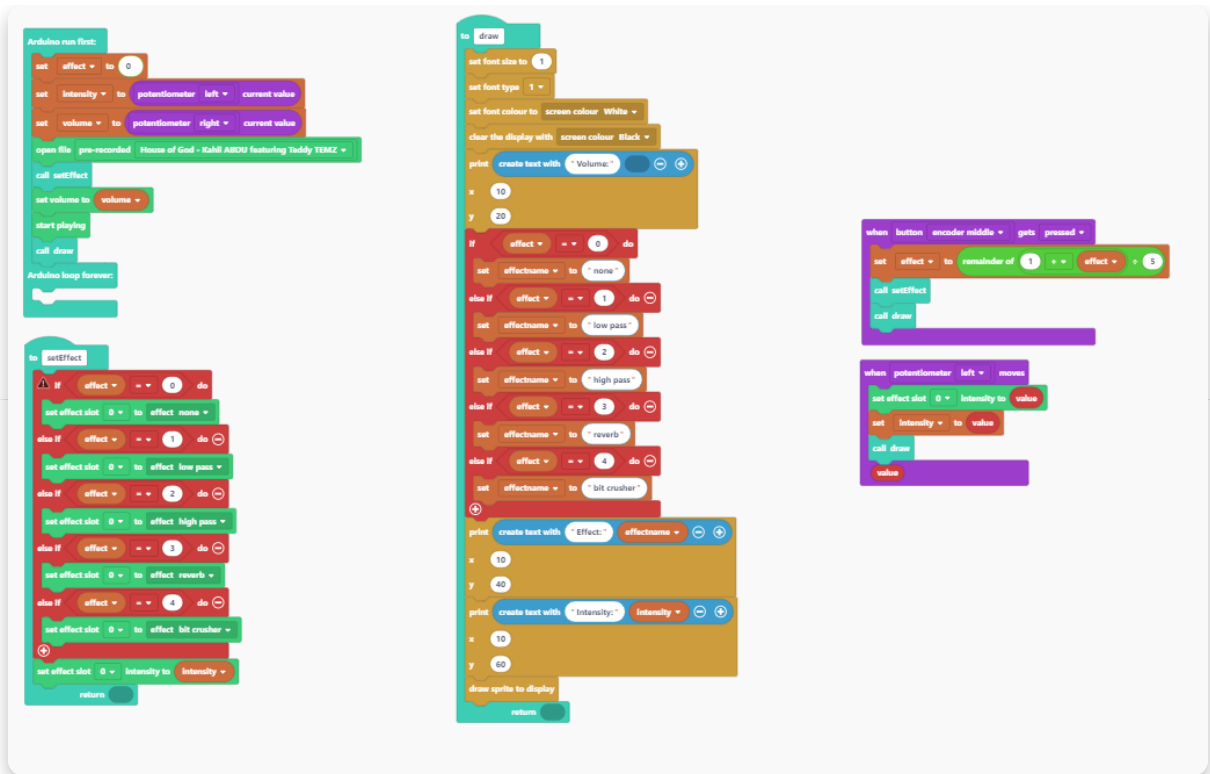


To finalize this block, add the "call draw" function:

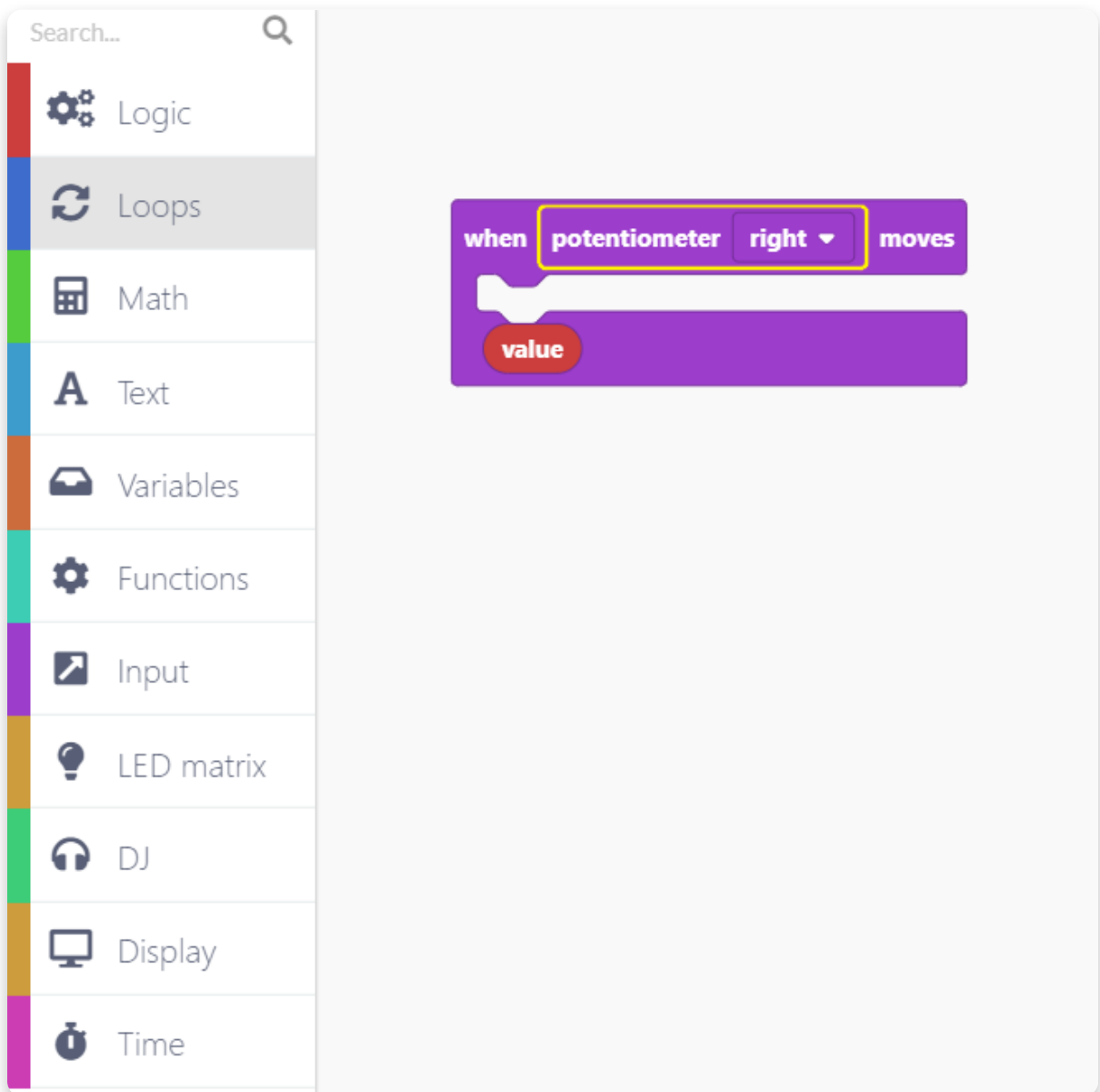


The left potentiometer input block is done!

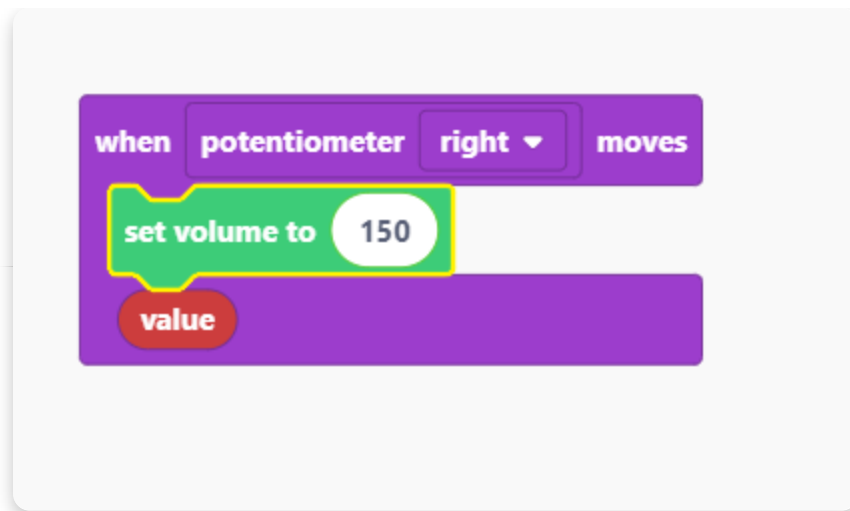
This is what you should have for now until we add only one more block similar to the previous one:



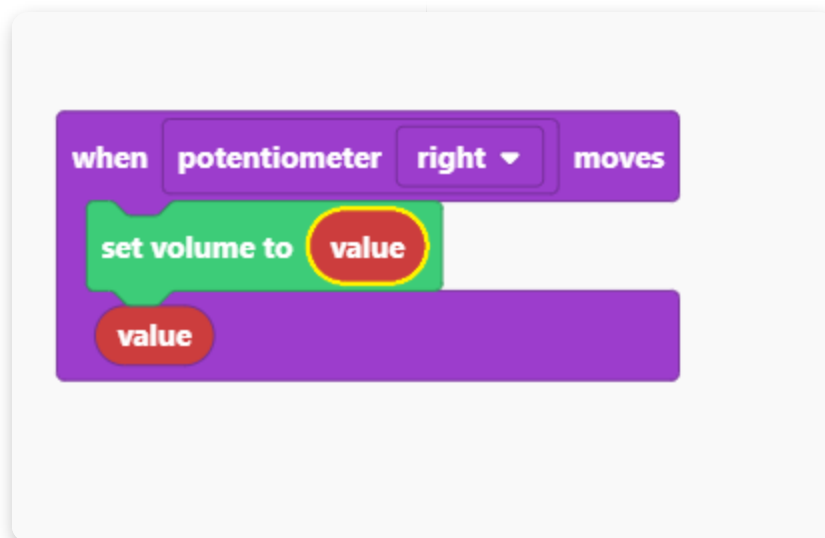
Lastly, let's add one more Input block for the right potentiometer that will regulate the volume:



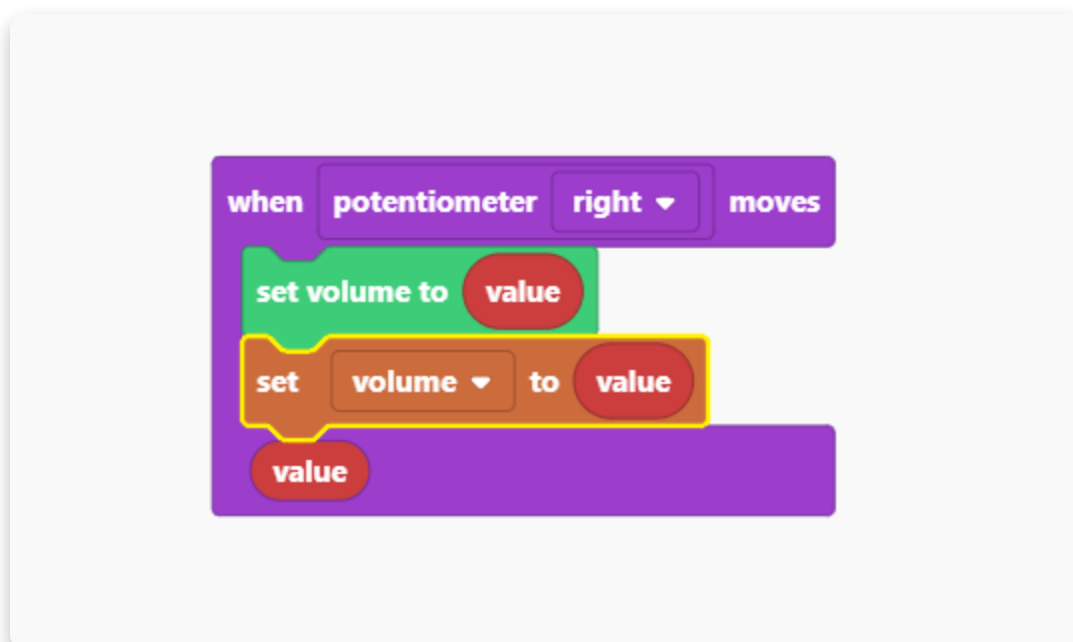
Find the "set volume to" block in the "DJ" section and put it here:



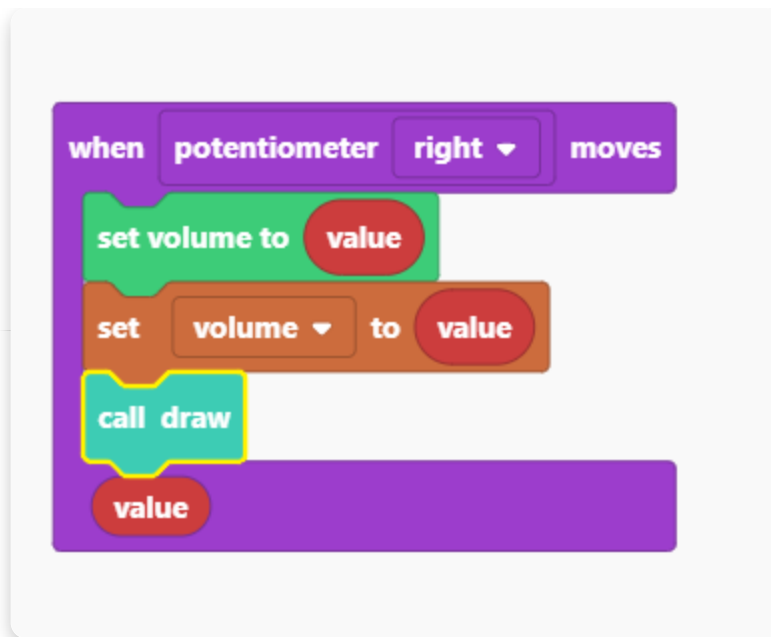
Drag the small value block into the window in the previous block:



Now, find the "set" block in the "Variables" section and insert the value block here:



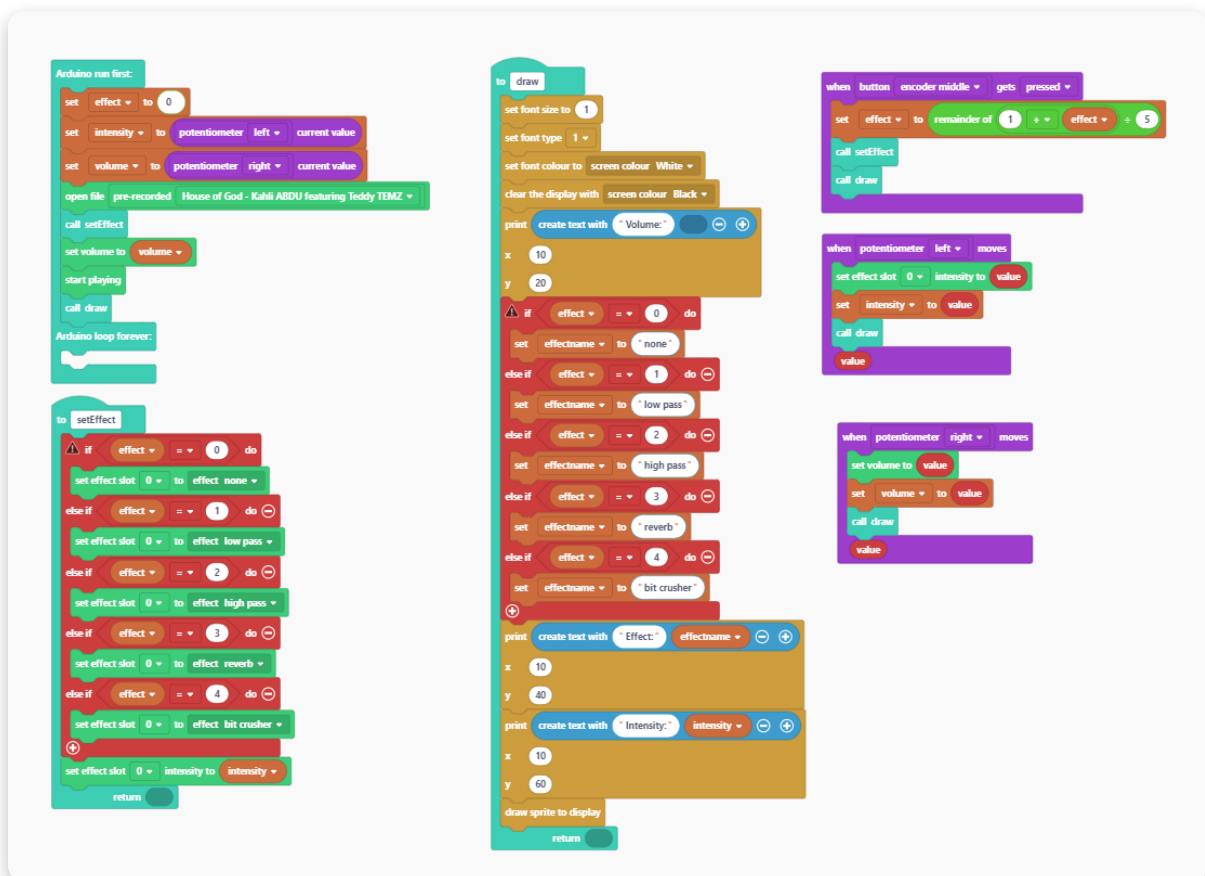
Add the "call draw" function at the end of the block.



This is it!

You should be ready to Run the project now. It might take a few moments to compile, so please be patient.

If the project doesn't work when you try to Run it, go through the sketch one more time.

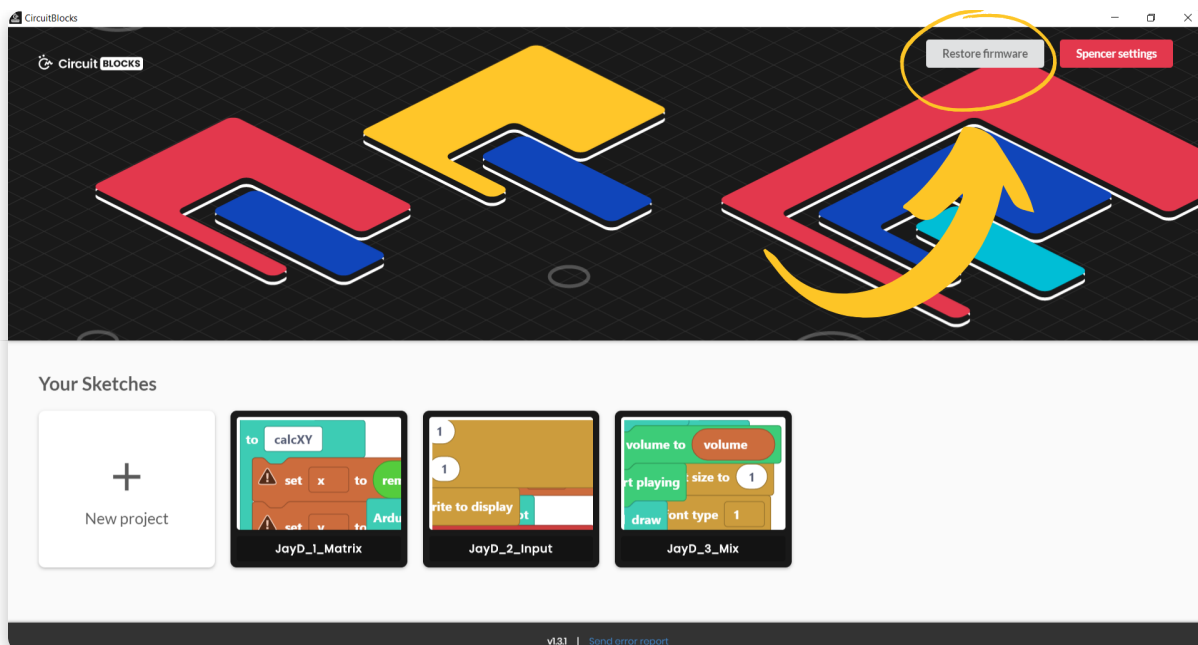


Restoring the firmware

Restoring Jay-D's firmware

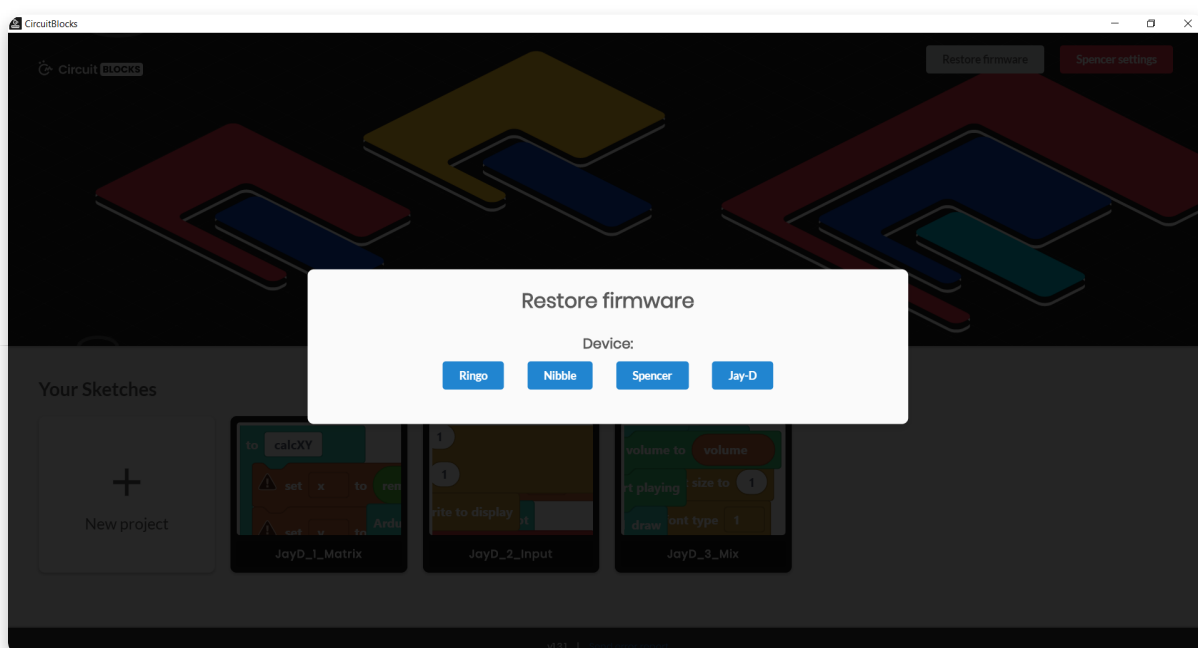
Once you're done coding and just want your Jay-D to be "normal" again, you need to restore his base firmware.

This is quite simple, just connect your Jay-D 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 Jay-D, of course.



Wait for a few seconds, and your Jay-D will be back and running like usual.

You need to do this whenever you're done coding your Jay-D if you want him to revert to his initial out-of-the-box functionality.

Get creative

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

We hope you're as excited as we are about Jay-D'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 Jay-D's lights, display, or remixes 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 and we'll help as soon as we can.

Thank you and keep making!

