Spencer coding - first steps

Introduction

Installation

Thank you for supporting CircuitMess and welcome to the Spencer coding tutorial.

We'll use CircuitBlocks for coding your newly-assembled voice assistant.

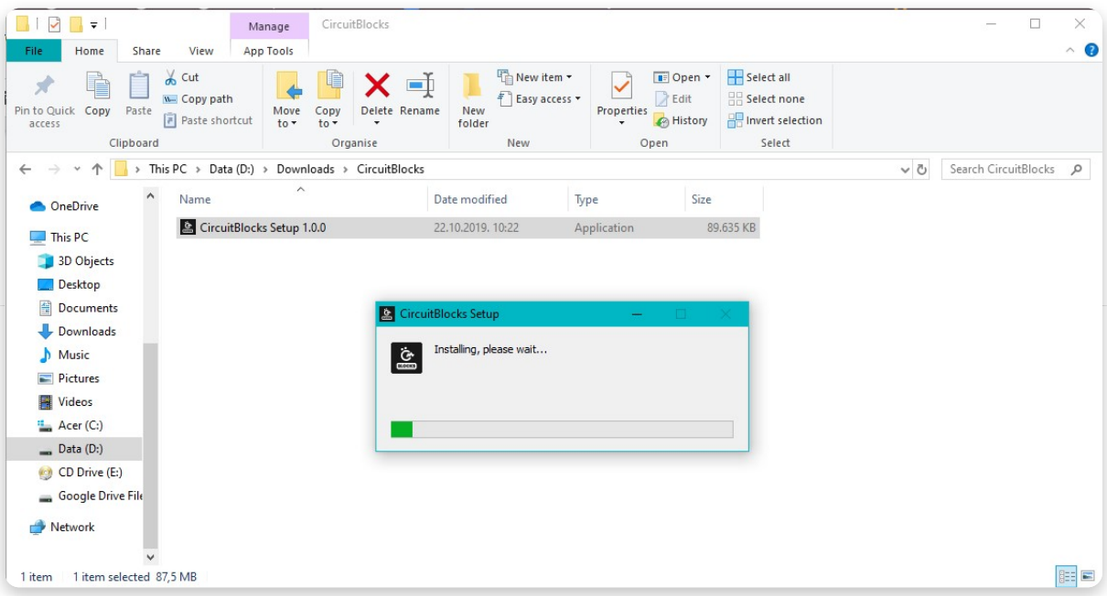
CircuitBlocks is a custom-made coding app that we've designed.
You will code your Spencer in CircuitBlocks' graphical block-based coding interface that will help you make your first steps in the world of physical computing.

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

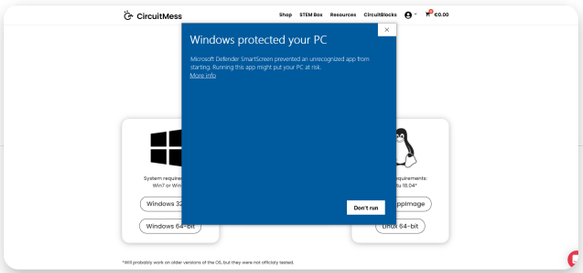


Your PC is not at risk!

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.

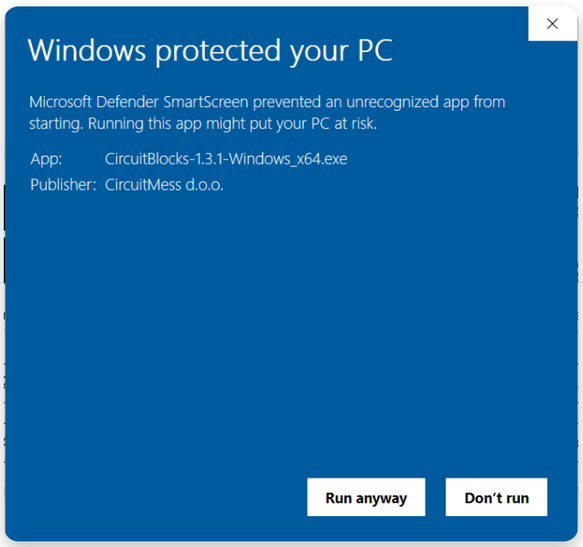
This is the message you might get when trying to install CircuitBlock on your PC. Windows reports a threat despite the program being safe to download and run.

Please proceed with installing by clicking on 'More info' option.



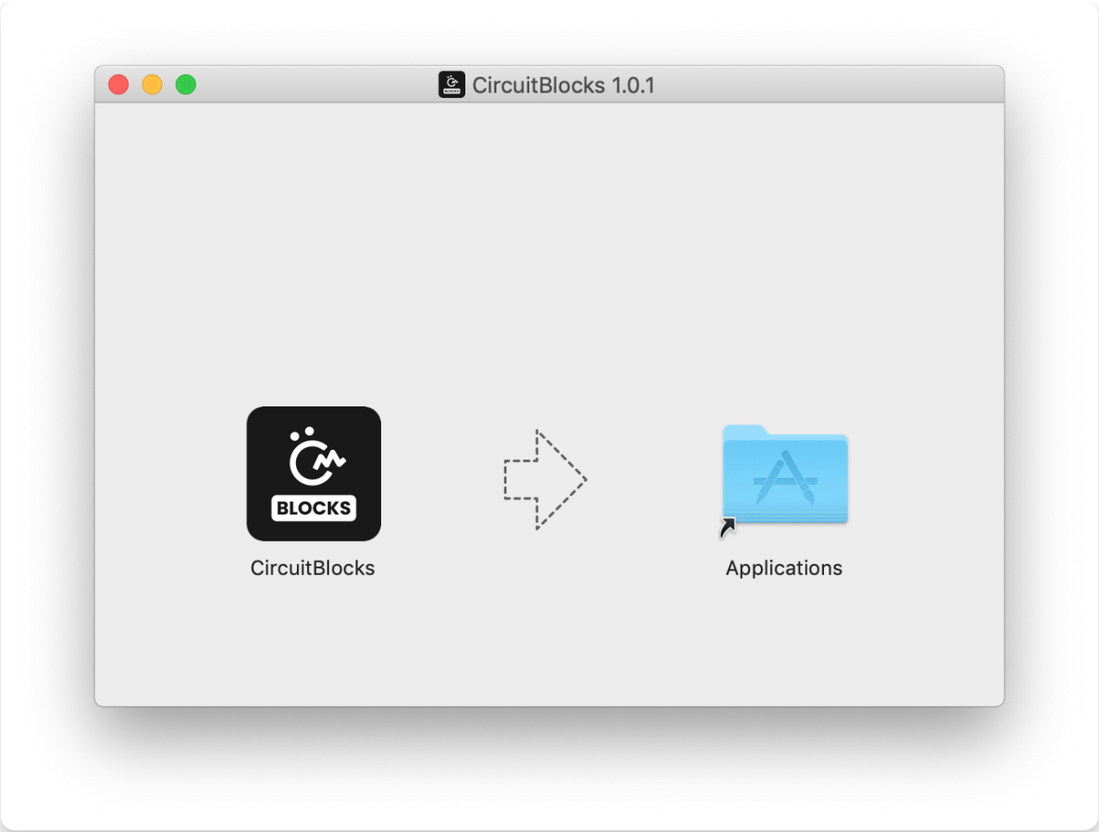
After you click on 'More info' option, an option to 'Run anyway' should appear at the bottom of the window.

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

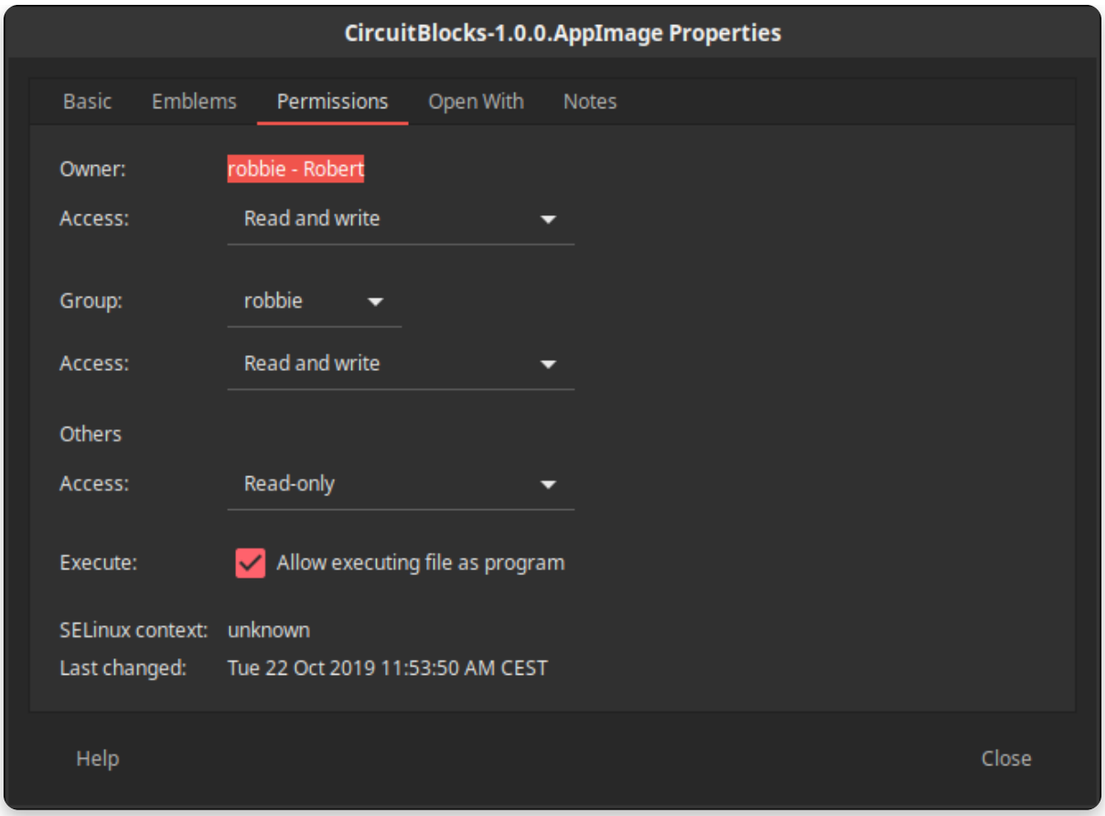
There are two ways of installing CircuitBlocks on Linux

"Regular" installation:

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):

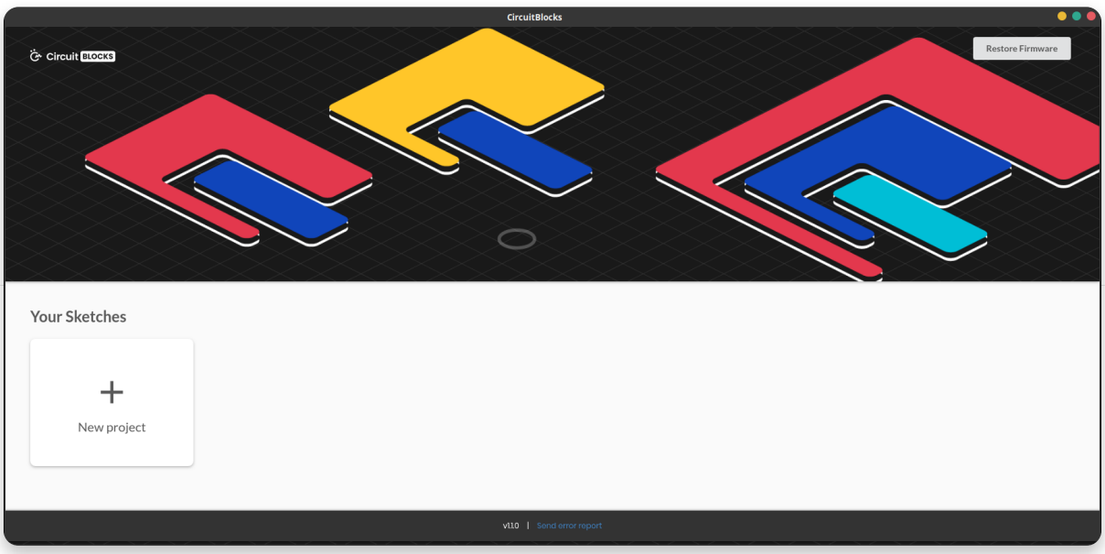
1. **Go to the CircuitBlocks download page**
2. **Press the "Linux AppImage" download button**
3. Right-click on the file and select **'Properties'**
4. Go to '**Permissions**' and tick '**Allow executing file as program**'
5. Double-click the file and the installation will complete automatically



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.

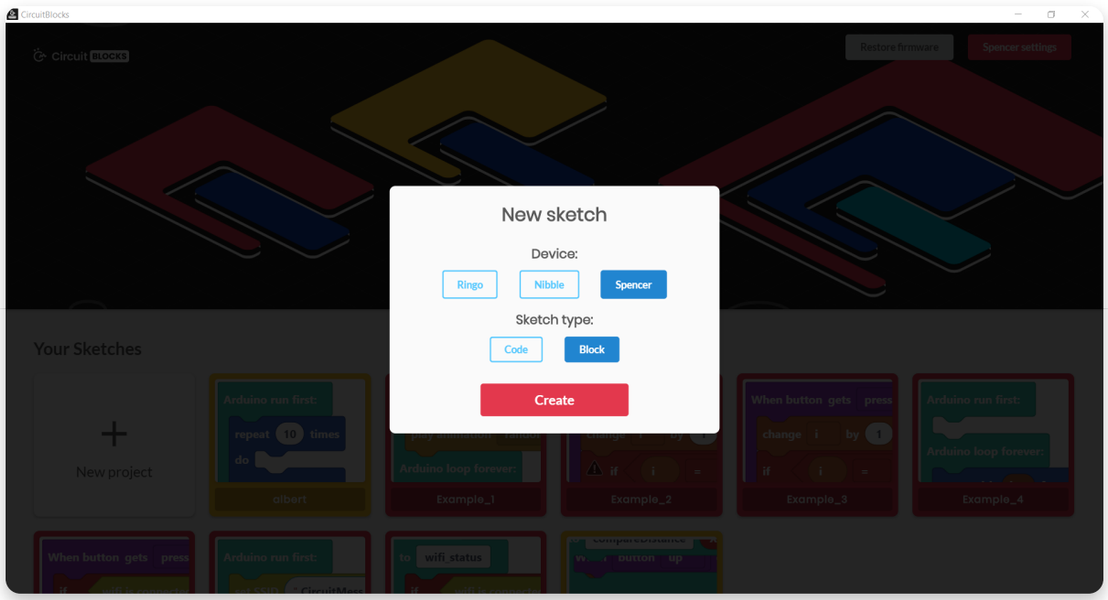
If you encounter any kind of an issue with CircuitBlocks, press the **'Send error report'** at the bottom of the main screen. You'll get an error report number - please reach out to us via contact@circuitmess.com and provide your error report number.

Creating a new project (sketch)

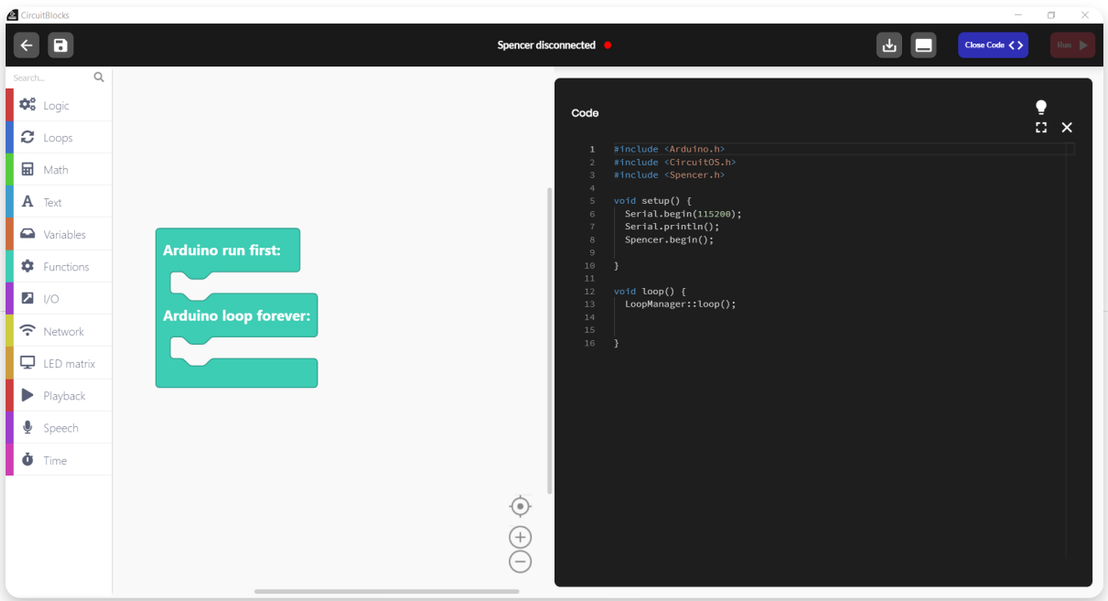
Press on the big "New project" button.

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

For the device, pick **Spencer**.
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.

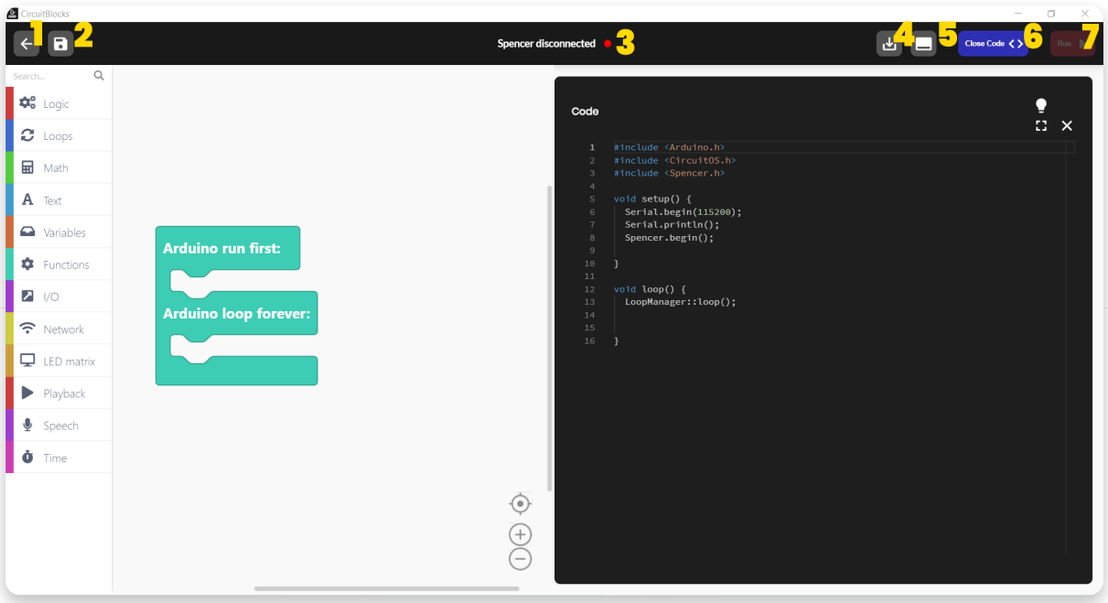
On the right side of the screen, you will see **code written in C++** appear magically by itself when you drag and drop the colorful blocks.ž

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.

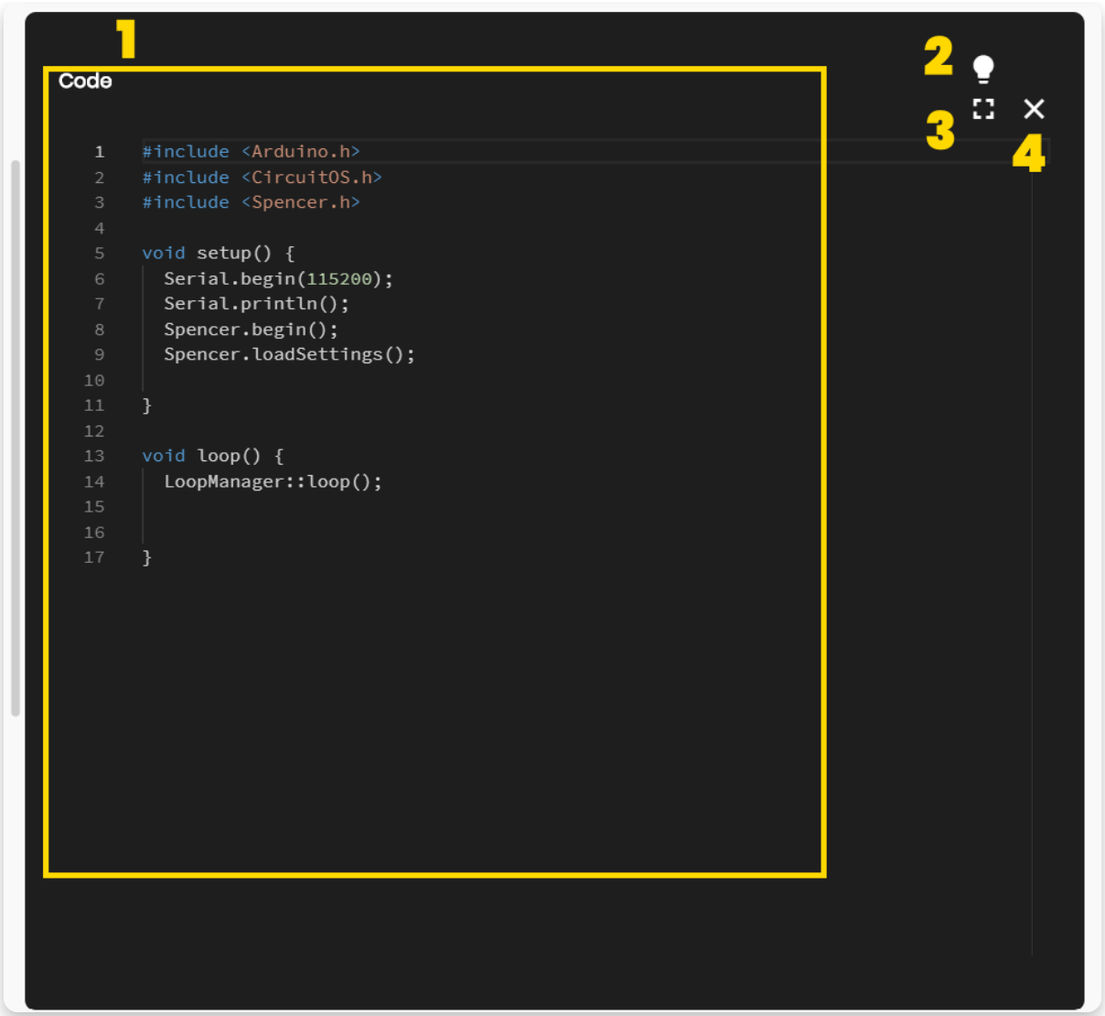
Toolbar



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. **Spencer connection indicator** - the red dot turns green if your Spencer 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 Spencer and your computer. In this window, you will later on be able to see the messages that are being sent from Spencer 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 Spencer understands (beep boop beep boop 1011100101) and send the code to your Spencer via the USB port

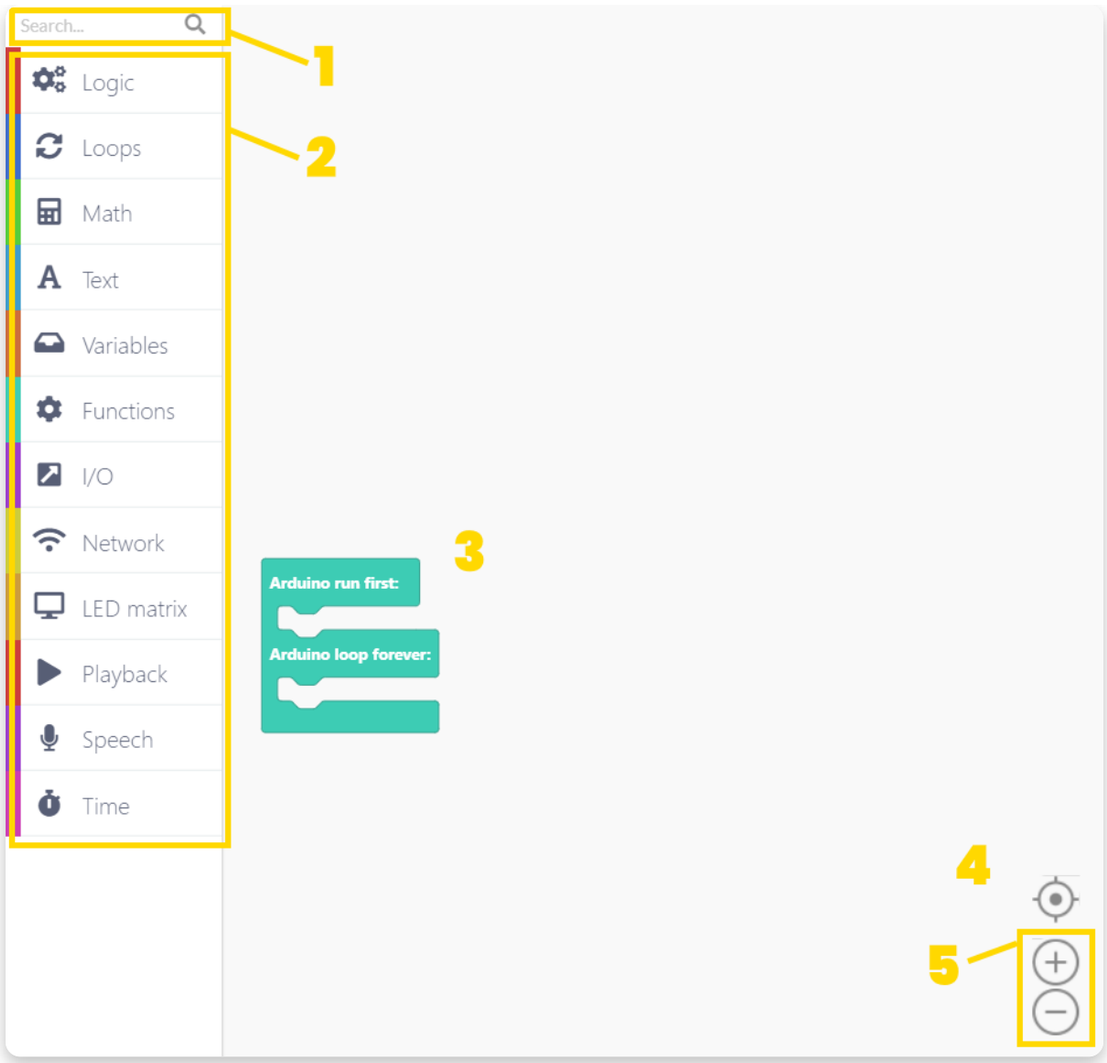
Code window



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

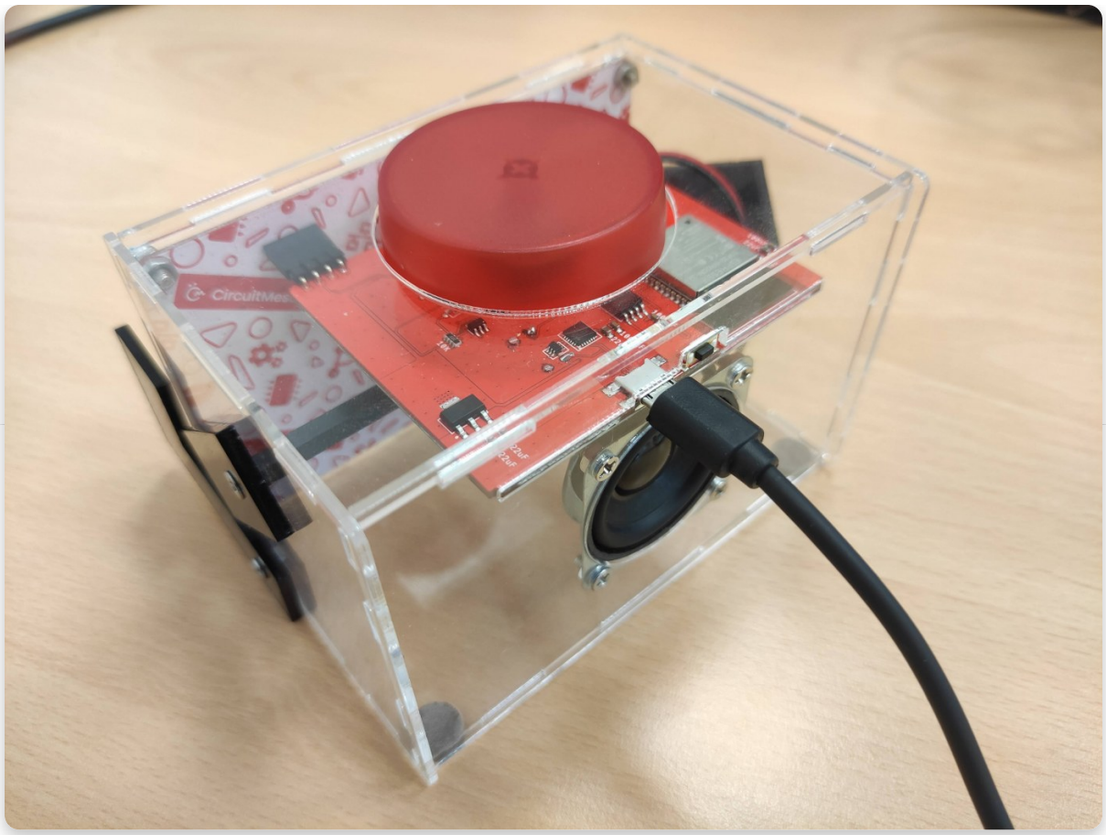
1. **Search bar** – type the name of the component you are looking for here.
2. **Component selector** – the blocks are divided into different categories here. Each category has a specific color designated to it.
3. **Drawing area** – you will drag the blocks from the component selector and drop them 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.

Baby steps

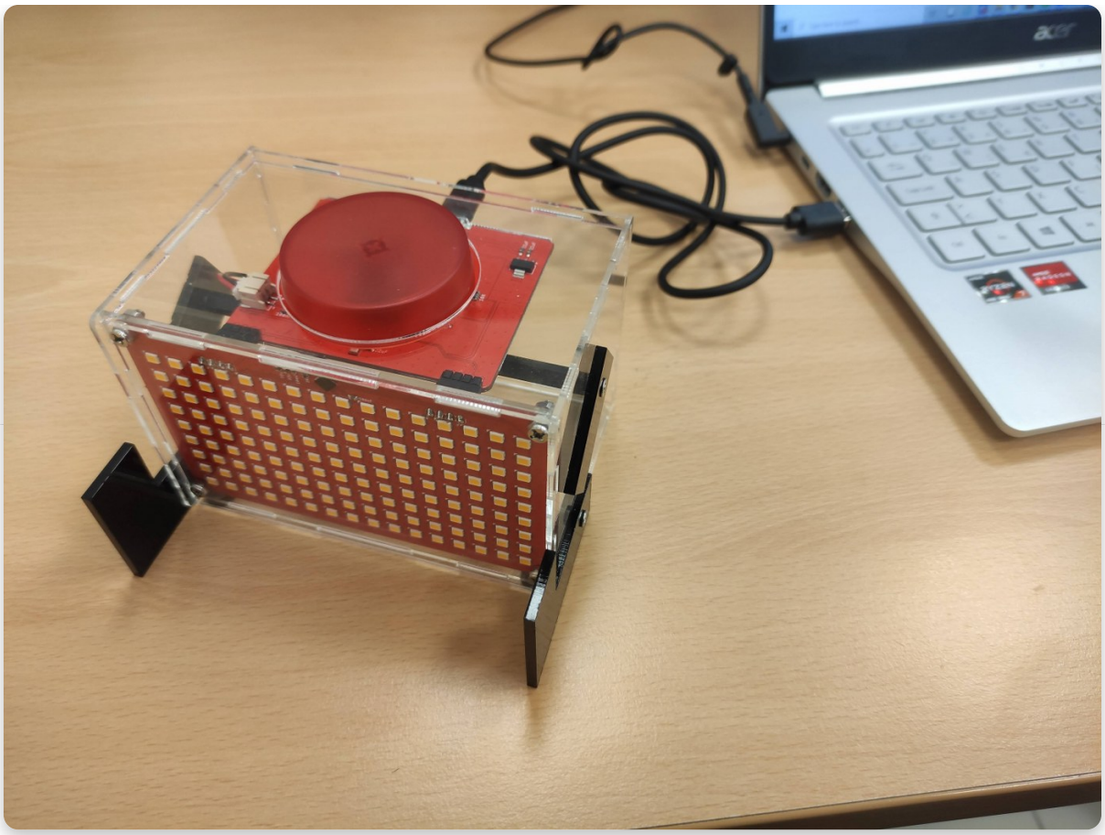
All of the lights

Let's get down to business and code our first app.

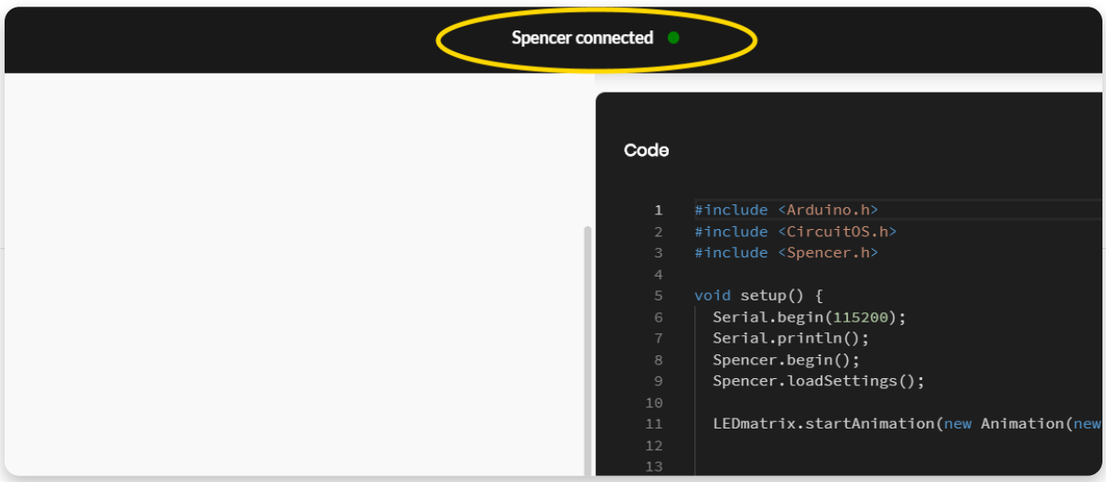
First, you need to connect your Spencer to your computer's USB port.




Caps Lock
Ctrl



CircuitBlocks should now say "Spencer connected".

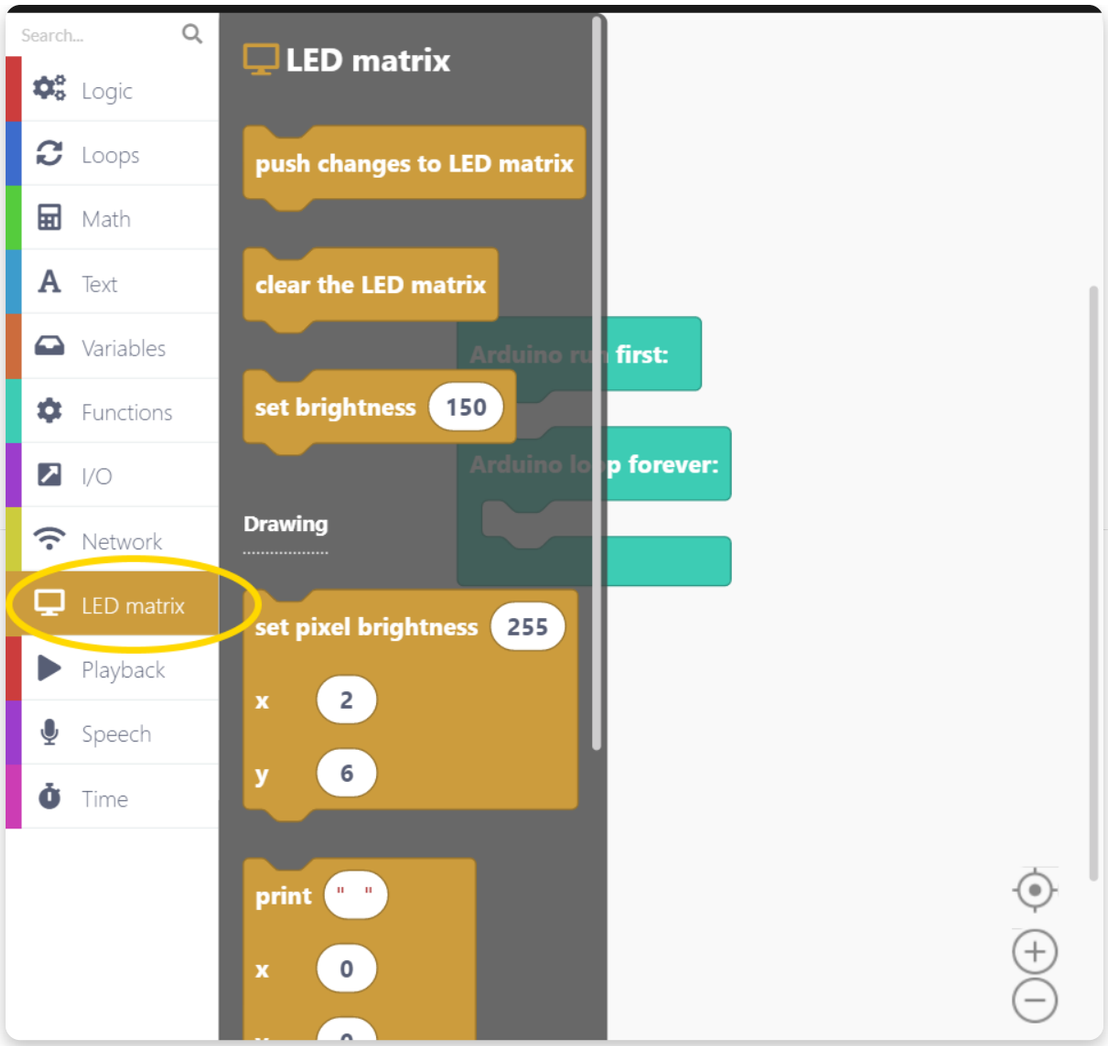


If CircuitBlocks didn't recognize your Spencer, please check if the USB cable is plugged in properly and if you are using a working USB port on your computer.

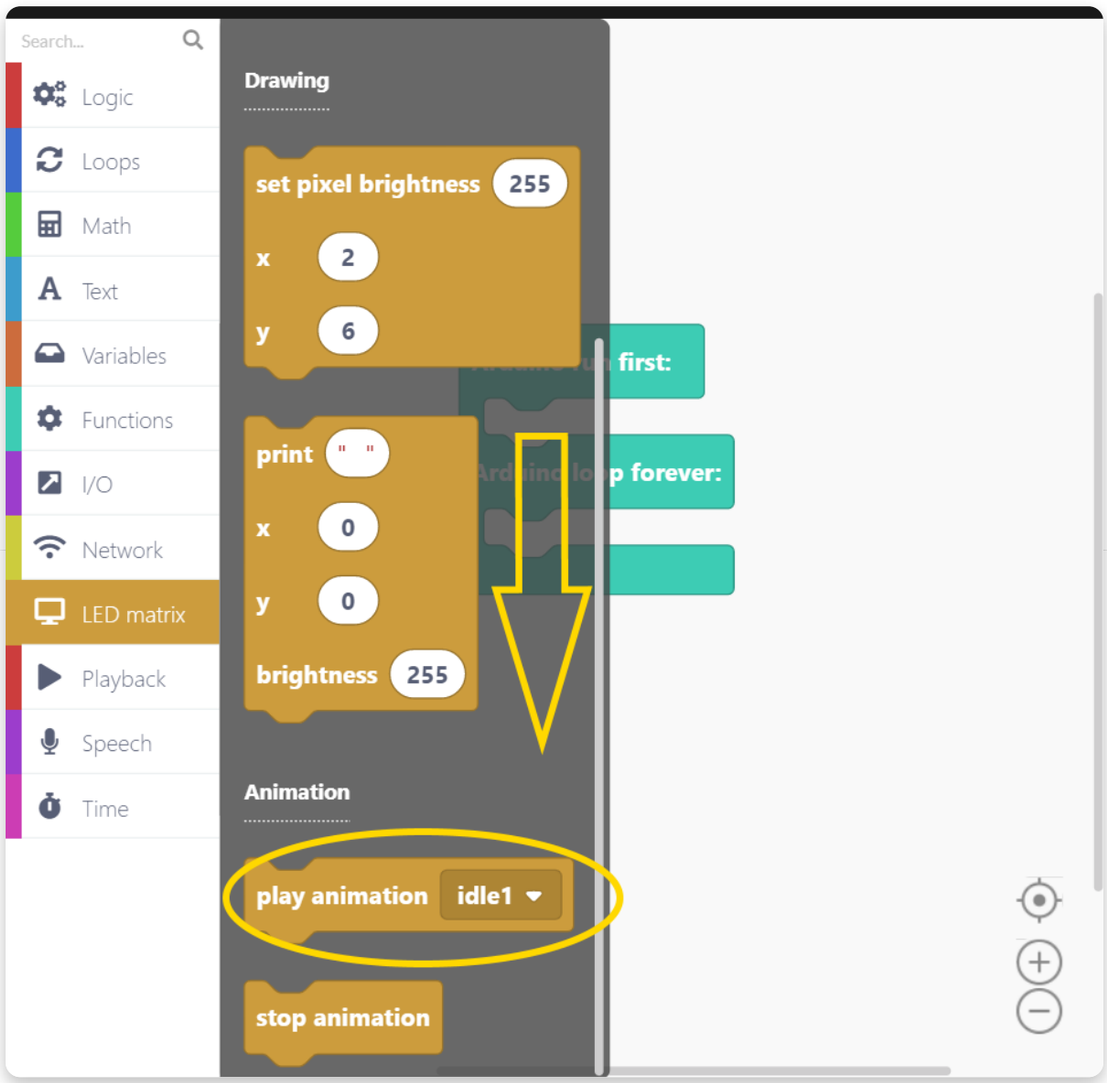
If you still cannot get CircuitBlocks to recognize your Spencer, something possibly went wrong with the driver installation on your computer. Drivers are these little programs that help your computer communicate with Spencer and they sometimes act funny. Reach out to us via email at contact@circuitmess.com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

Let's make Spencer display an animation on its LED display

Find the section called "LED matrix" on the left side of the screen.
This group of blocks has different commands in it that are used for drawing images and on Spencer's LED display (we also call this display the "LED matrix").



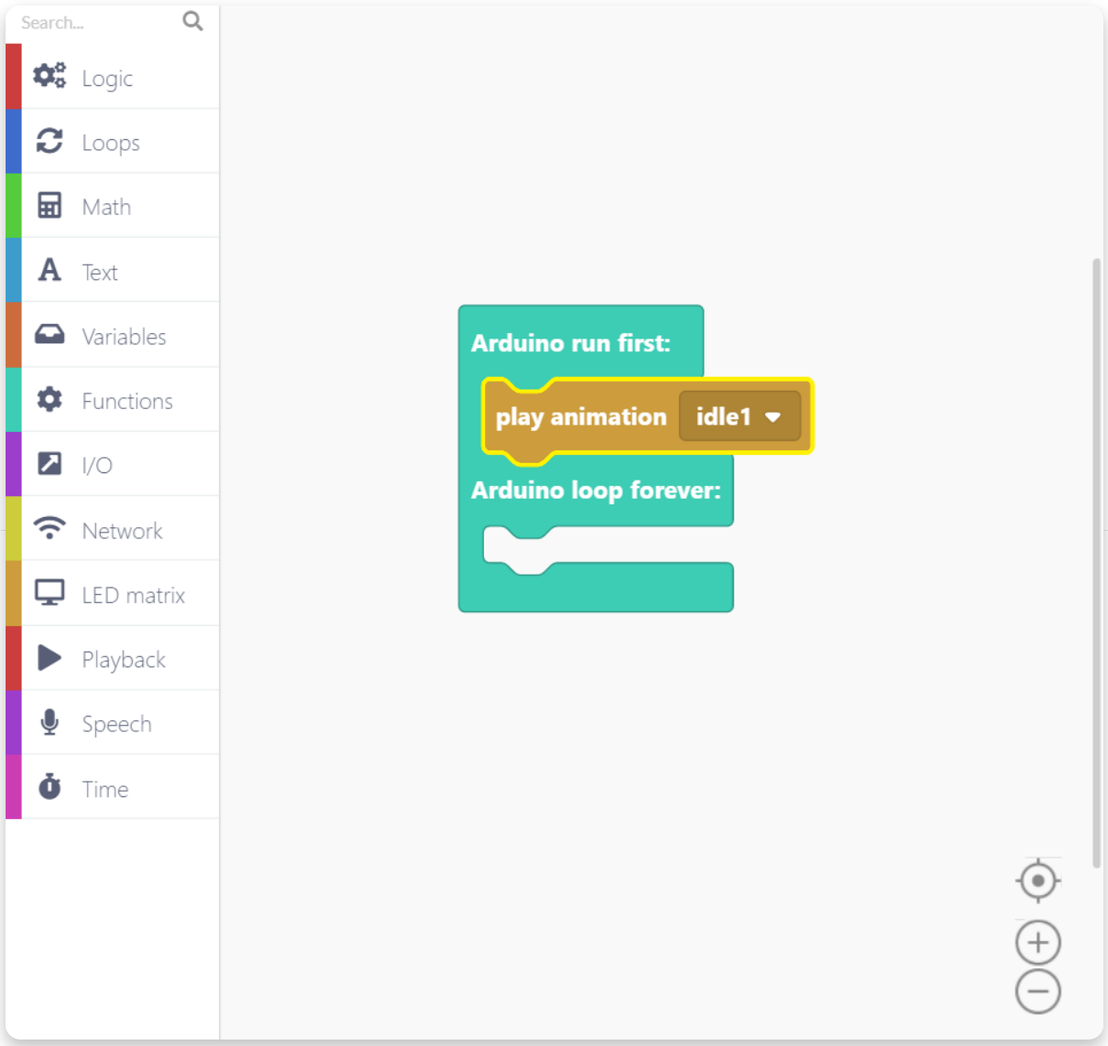
Find the block called "Play animation".



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

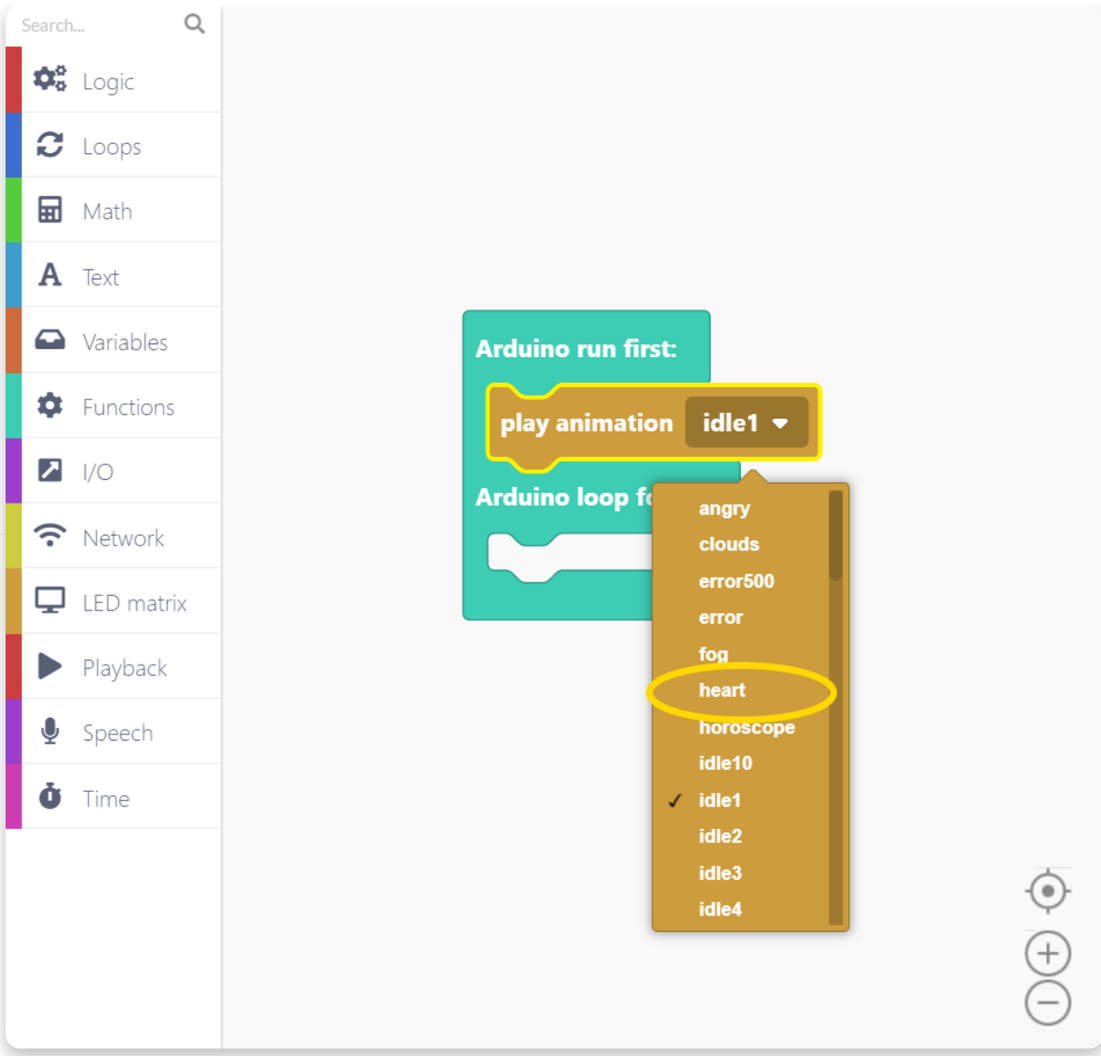
area.
Stick the *Play animation* block under the text saying "Arduino run first".

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



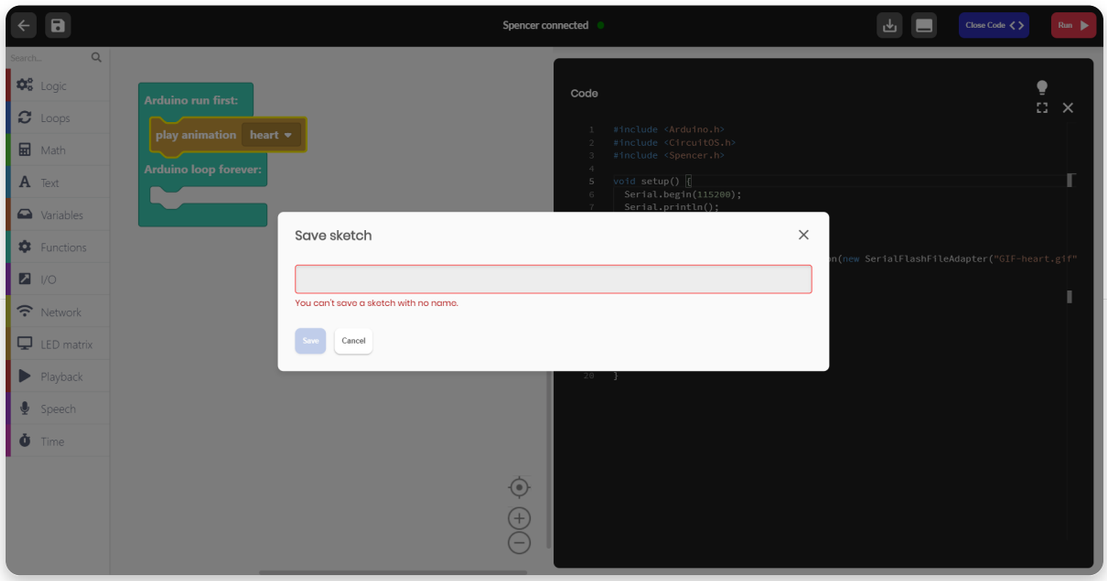
Click on the tiny arrow on the *Play animation* block and a drop-down menu will appear. Here you can find all the GIF animations that are saved on your Spencer's flash memory.
Currently, you cannot add new GIF animations to your Spencer, but we're working on making that easily doable in the next CircuitBlocks update.
On the other hand, you can manipulate the individual pixels of the display, but we'll cover that later on in the tutorial.

Ah yes, back to the tutorial - pick the "heart" animation because I like that one :)

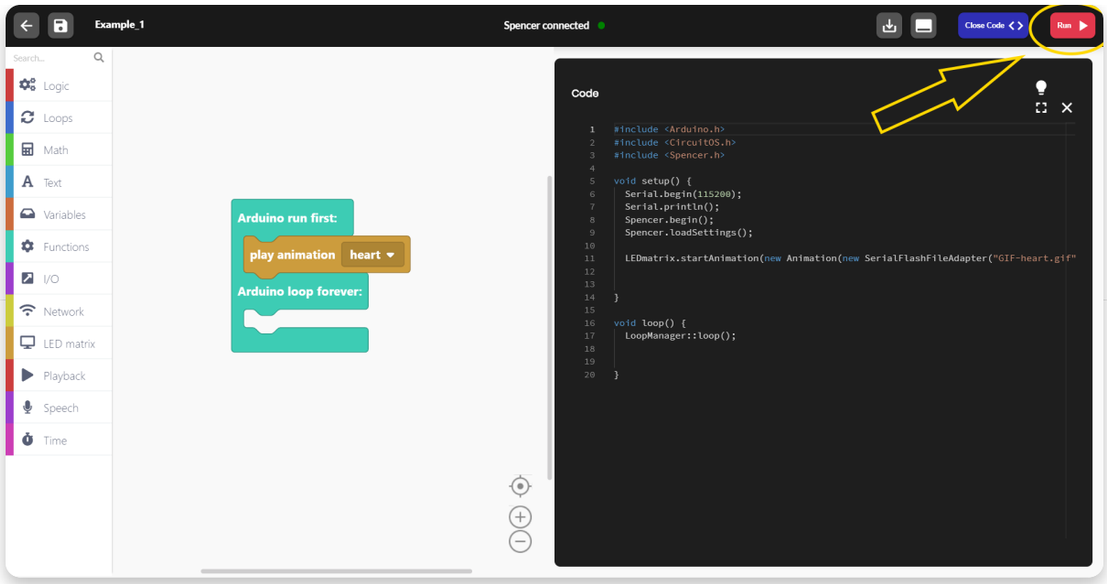


Perfect! Our first ever program is done (that was easy!).
This code that we have written just now is going to play the *heart* animation on your Spencer when it turns on and the animation is going to be played endlessly.

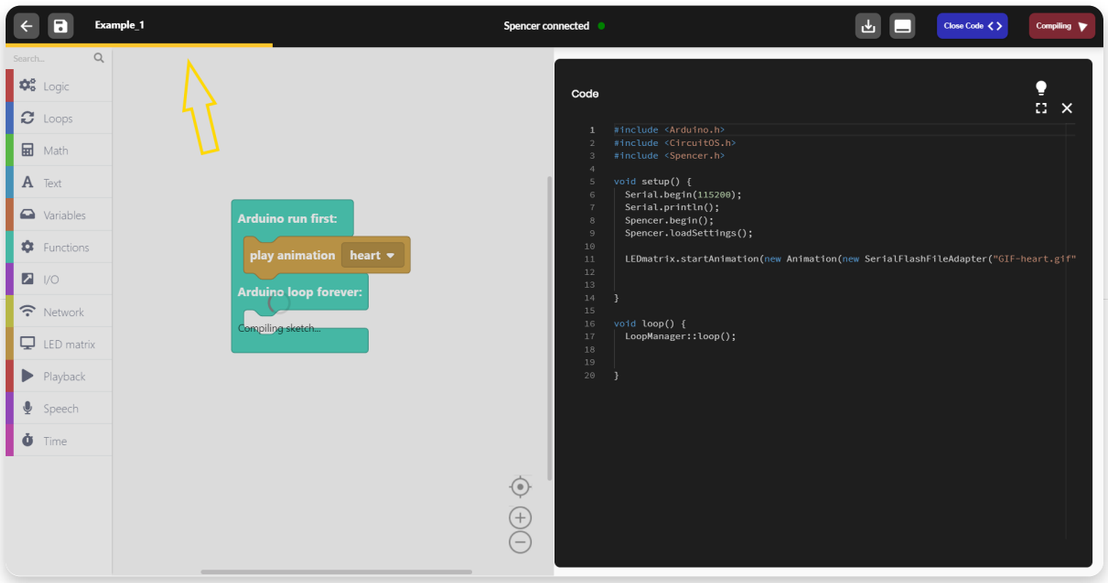
Before going further, press the *Save* button on the top left of the screen and save your program. Name it something creative.



Alrighty, now 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 Spencer, 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 Spencer 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 heart animation playing on your Spencer.

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.

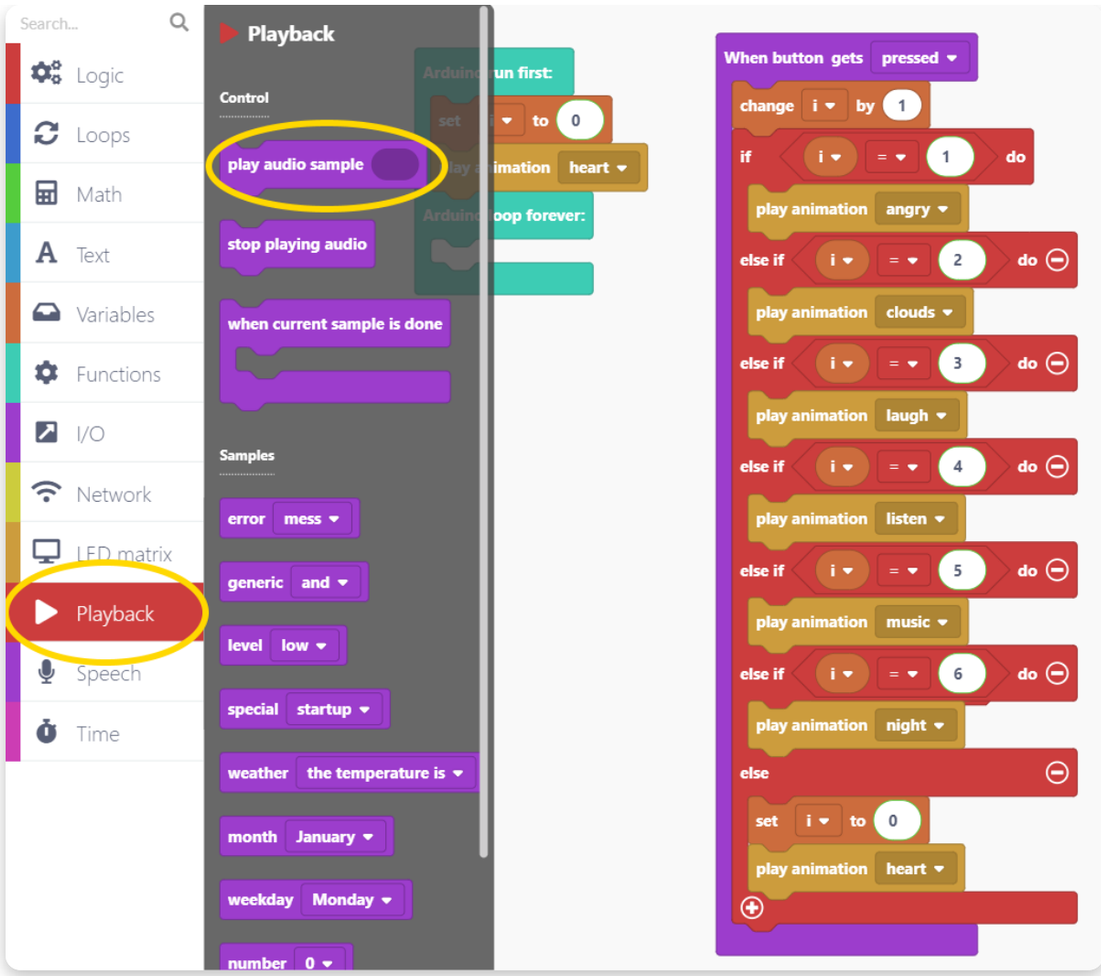
Loading...

Sound effects ahoy

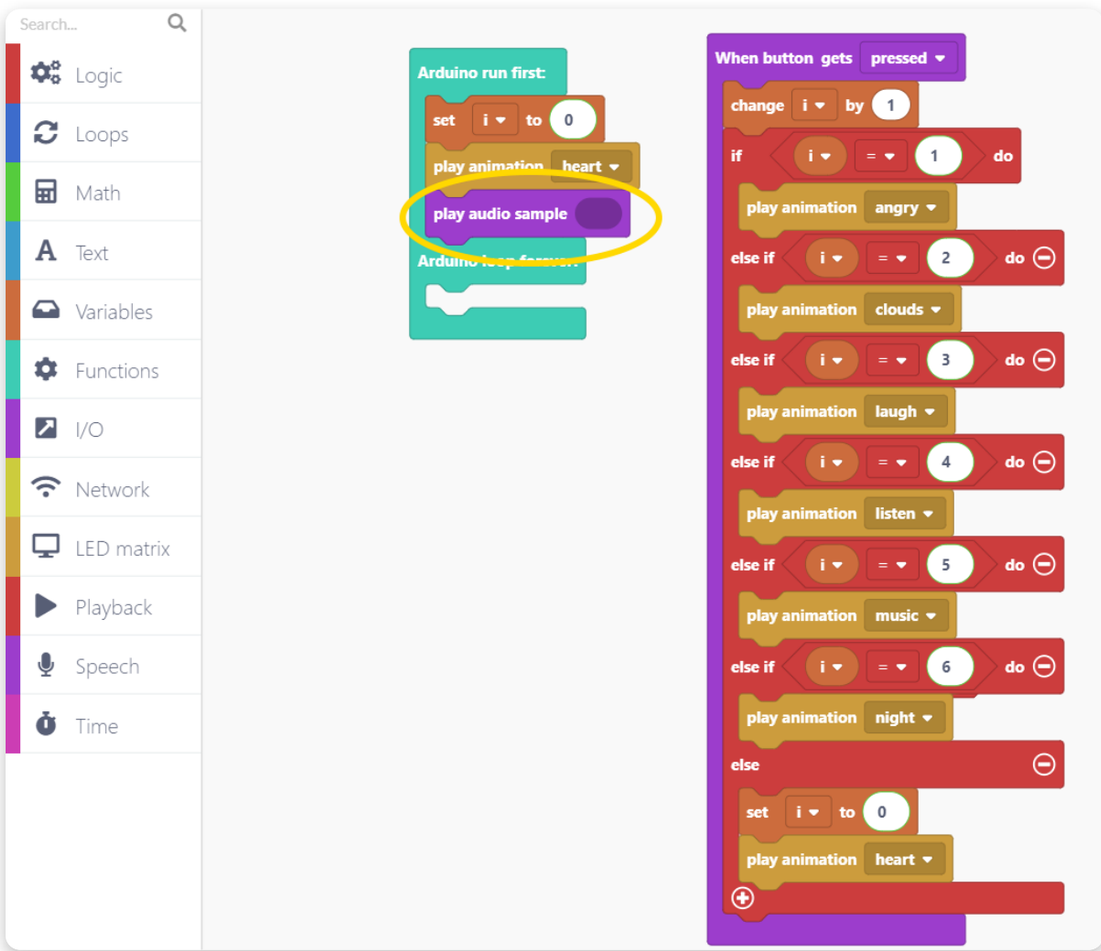
Spencer can talk and make funny noises as well. It comes preloaded with a bunch of sound effects and dialogues.

You can generate your own voice dialogues too, but first, let's play some of the preloaded sound samples.

The section called "Playback" has everything sound-related in it. Let's start by dragging and dropping the "Play audio sample" block onto te drawing area.



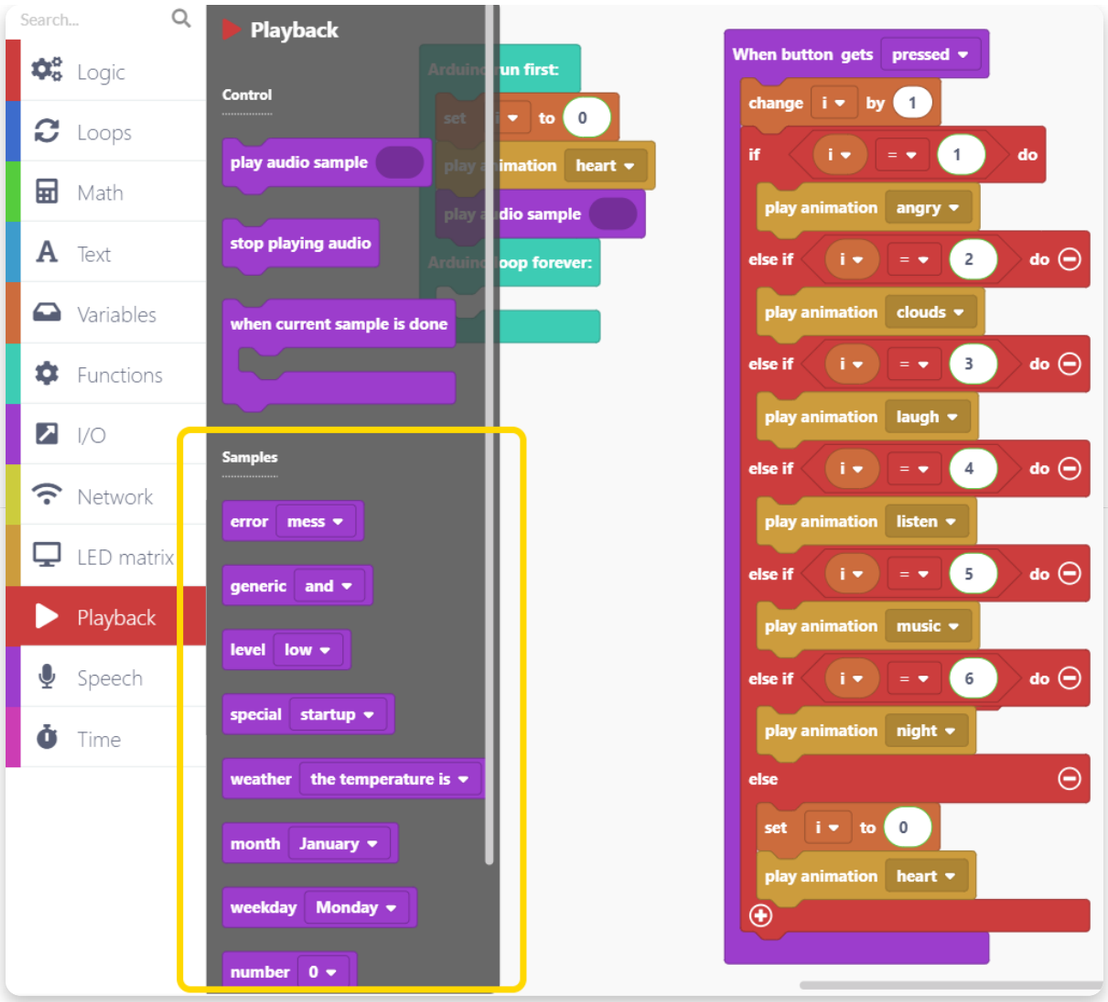
Place the block into the **Arduino run first** branch so that the sound effect gets played as soon as your program starts.



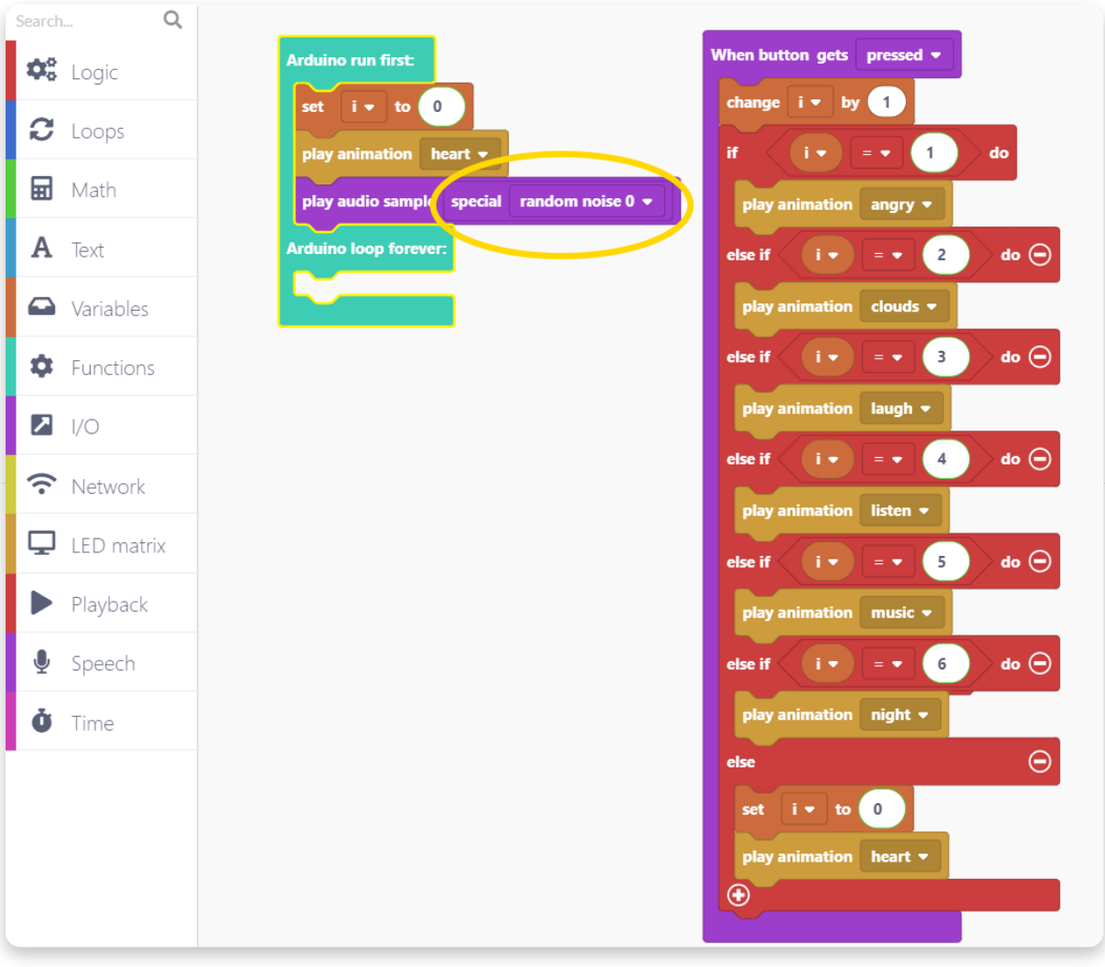
Next, we need to choose the sound effect that is going to be played.

There is a whole section of different samples you can choose and explore from.

I'll take the sample block named "Special".

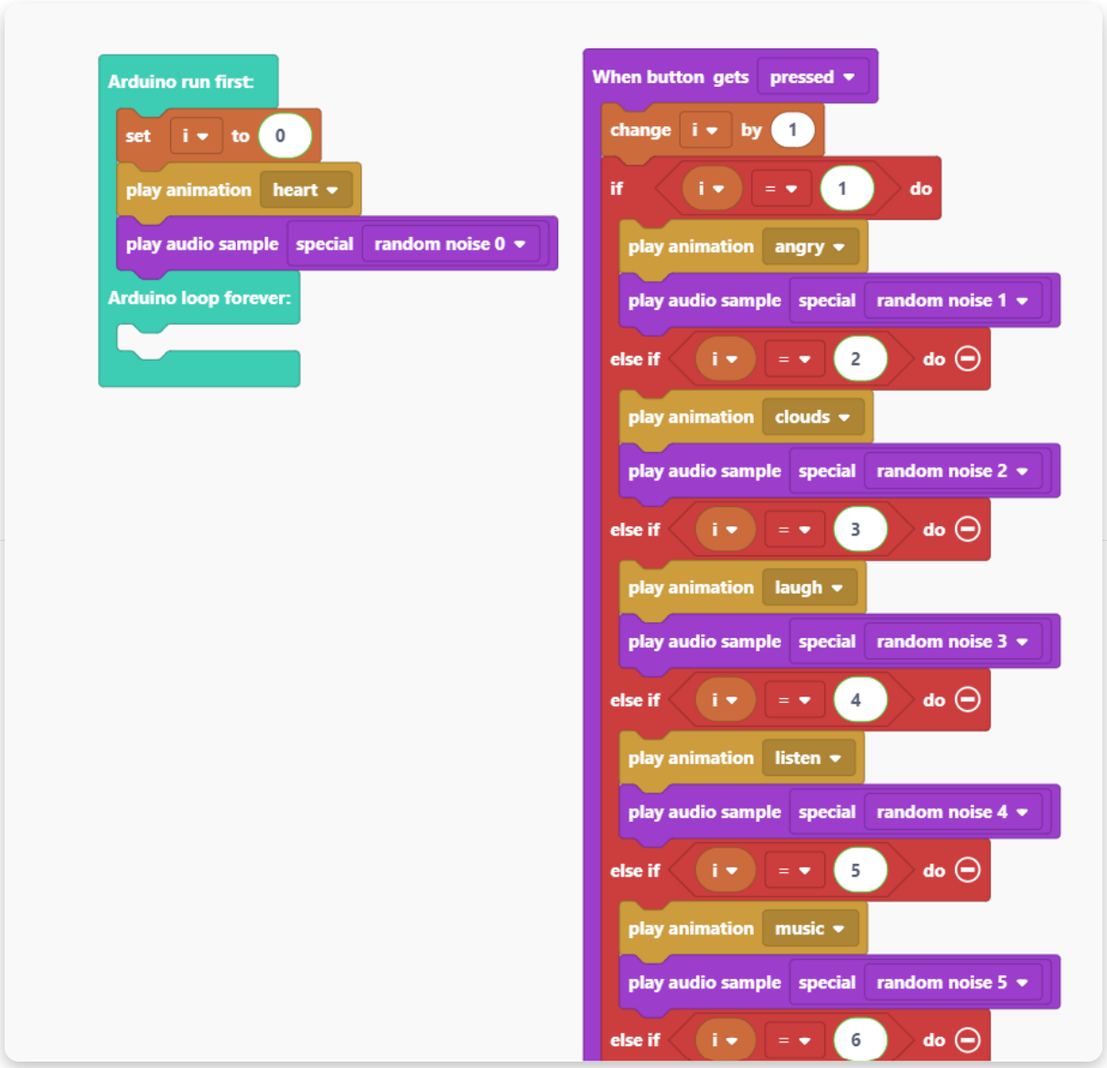


You need to place this sample block into the "Play audio sample" block. You can choose the different audio samples present in the "Special" category of audio samples by pressing on the drop-down menu.



If you Run this code and upload it to your Spencer, it will play the selected sound effect as soon as your program starts.

If we want to add a bit more *oomph* to our code, we can make Spencer play a different audio sample every time you press its button. Your code should look like this if you want to do so:



Remember to save your code!

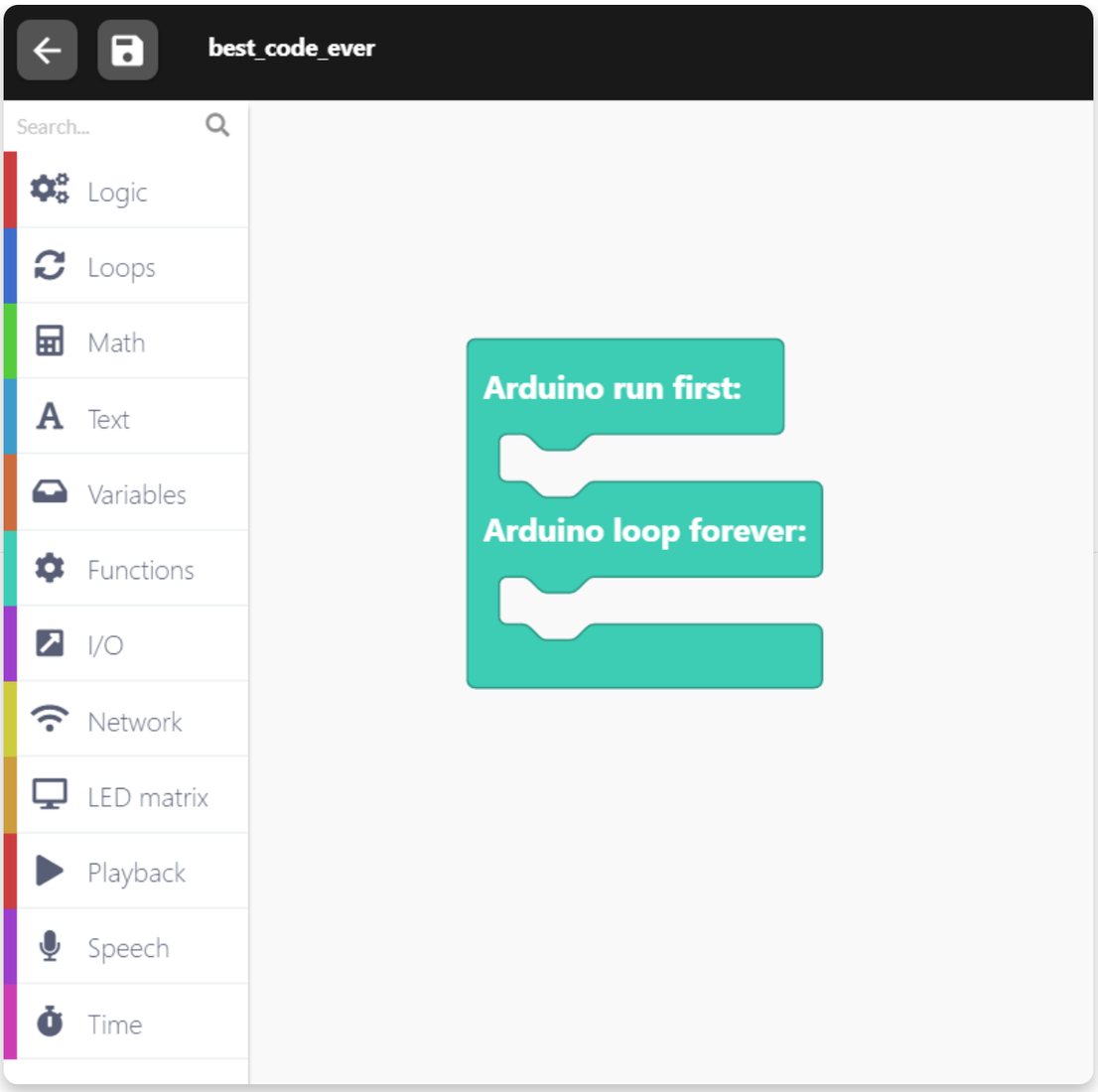
This was fun. Let's see what else we can do.

Big boy/girl coding

Let's make Spencer talk

Let's try something more advanced now - we're going to make Spencer generate and "say" a unique piece of dialogue every time we smack his big red button.

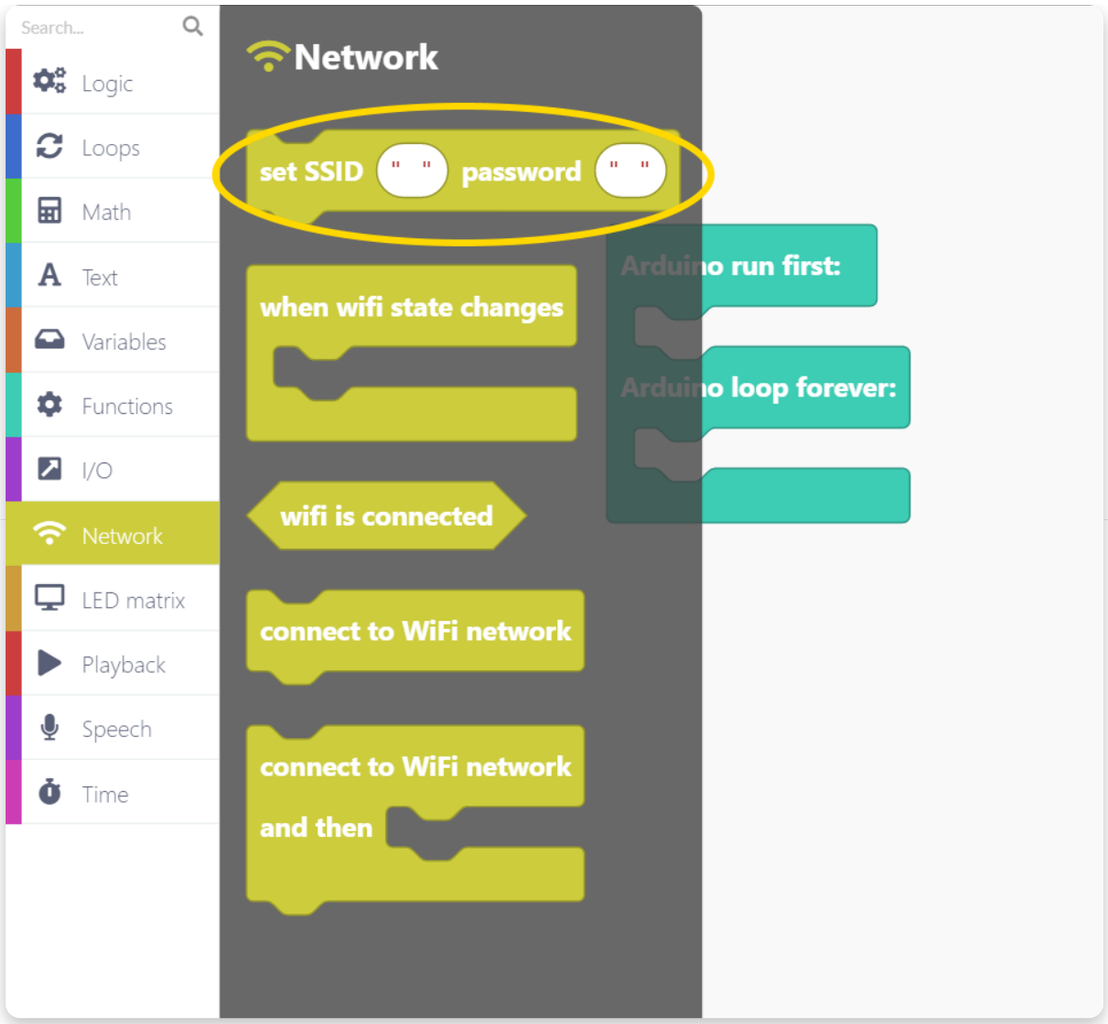
Start fresh by creating a new sketch. Save it right away and name it something snazzy (I've named mine "best_code_ever").



Next, we need to connect Spencer to a WiFi network.

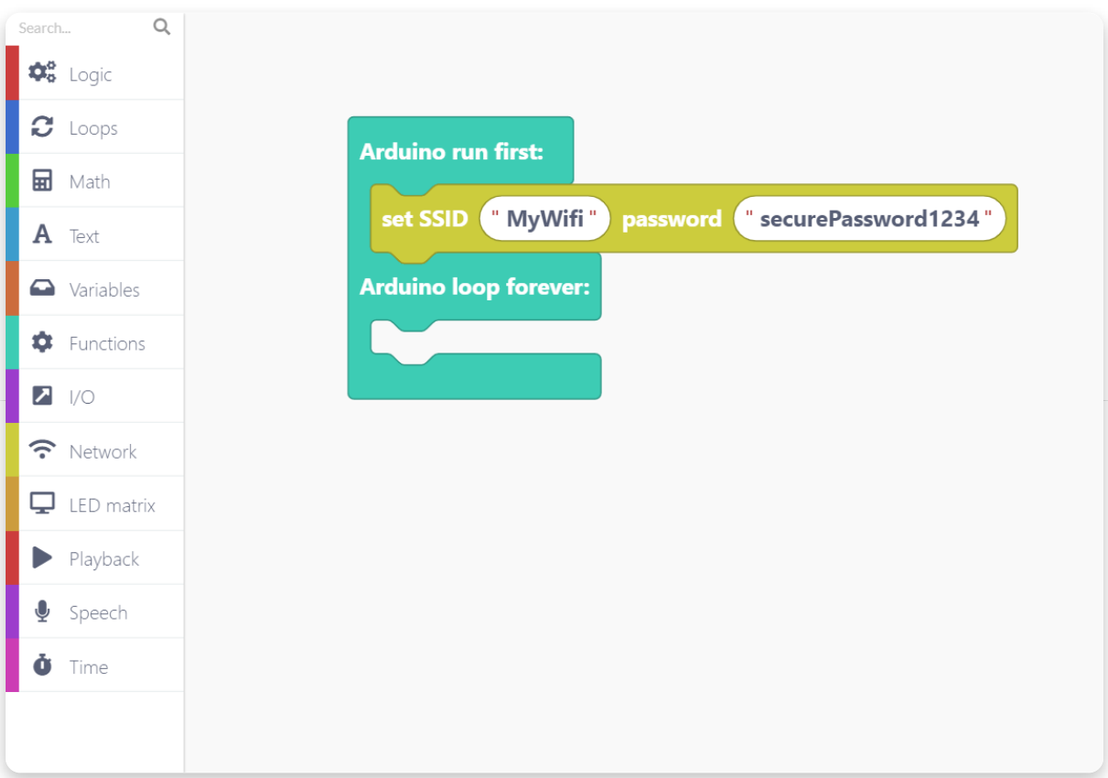
Spencer needs internet access to "talk". The way this works is that Spencer connects to our server and sends the text it wants to say. The server then generates an audio clip using a TTS (Text To Speech) algorithm and sends it back to your Spencer.

For connecting to the WiFi network, you'll need the "Set SSID and password" block located under the "Network" category.

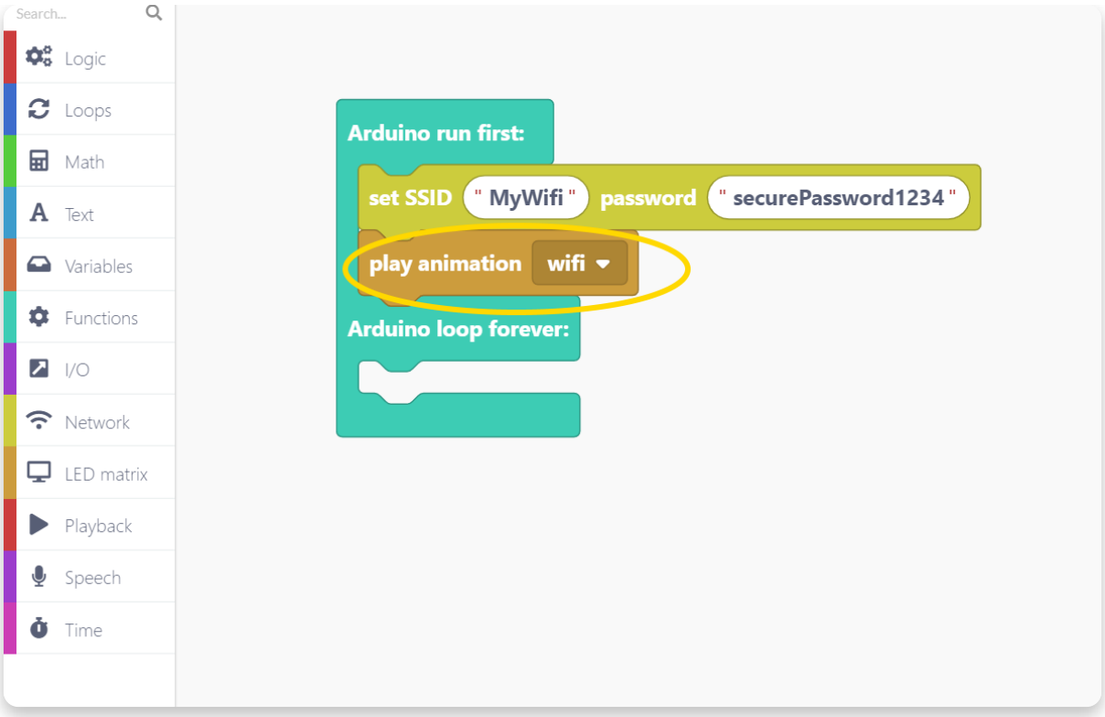


Drag the block on the drawing area and enter your WiFi network's name and password.

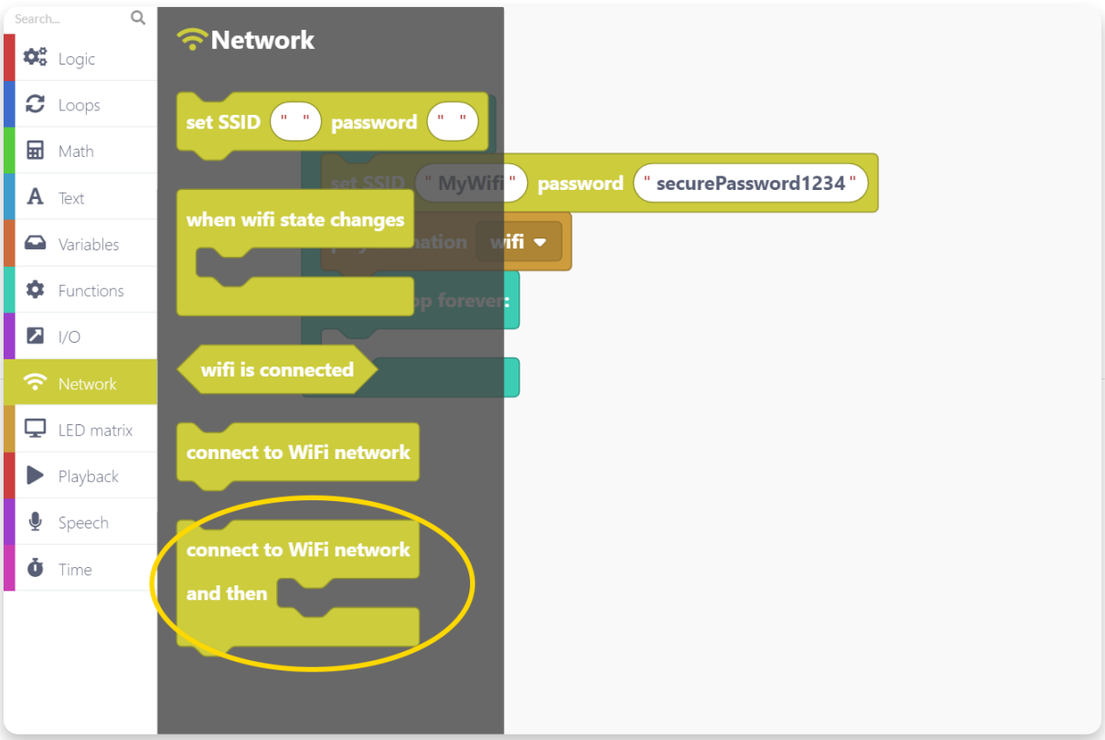
SSID is the name of your network.



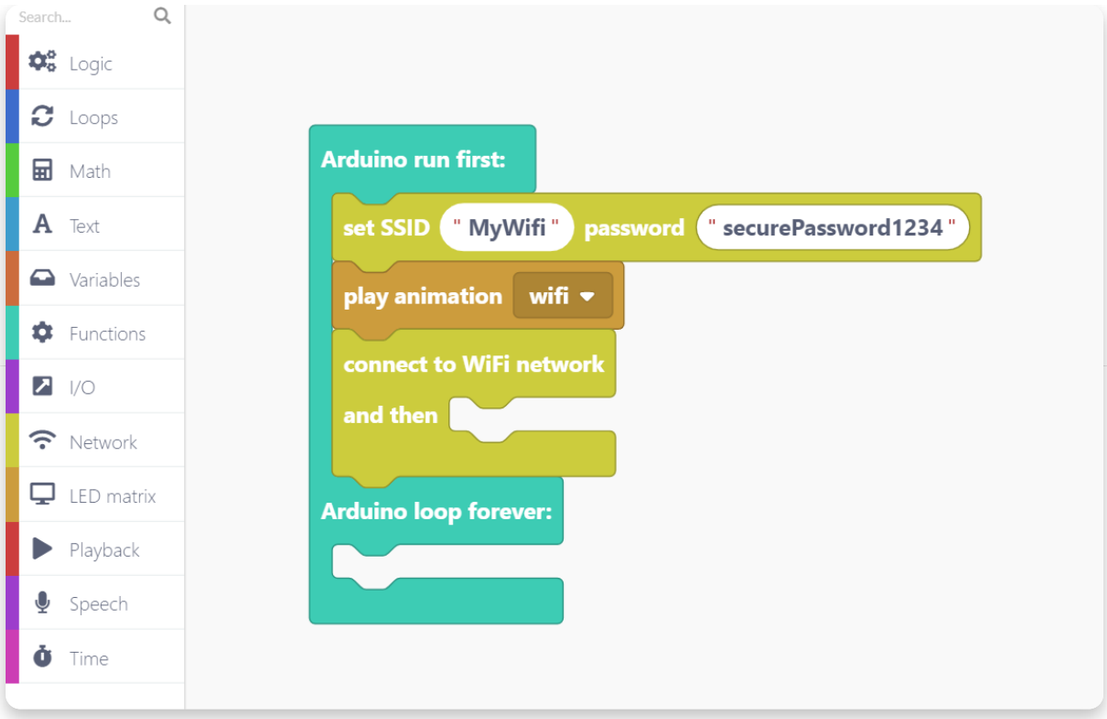
When we set the login details for the network, let's make Spencer show the "WiFi" animation. We can do this by adding this block over here:



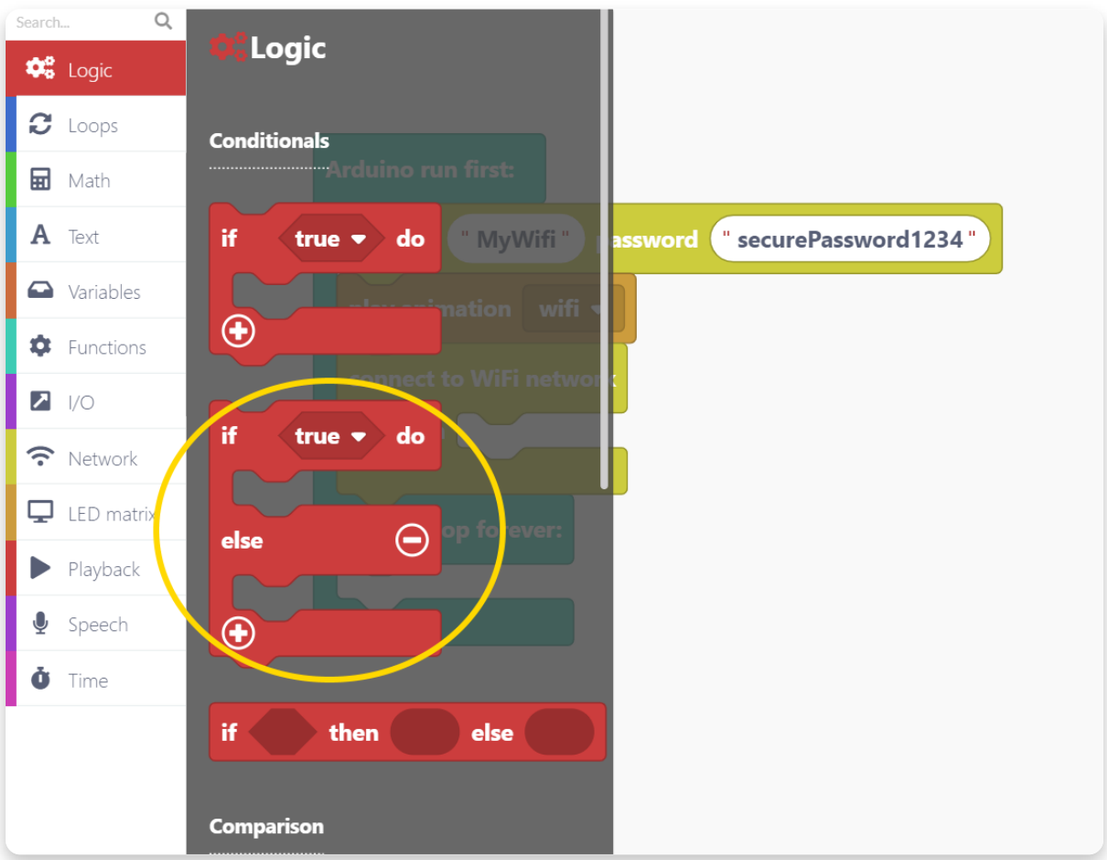
Now you need to drag and drop the "Connect to WiFi network" block. This block initiates a connection routine.



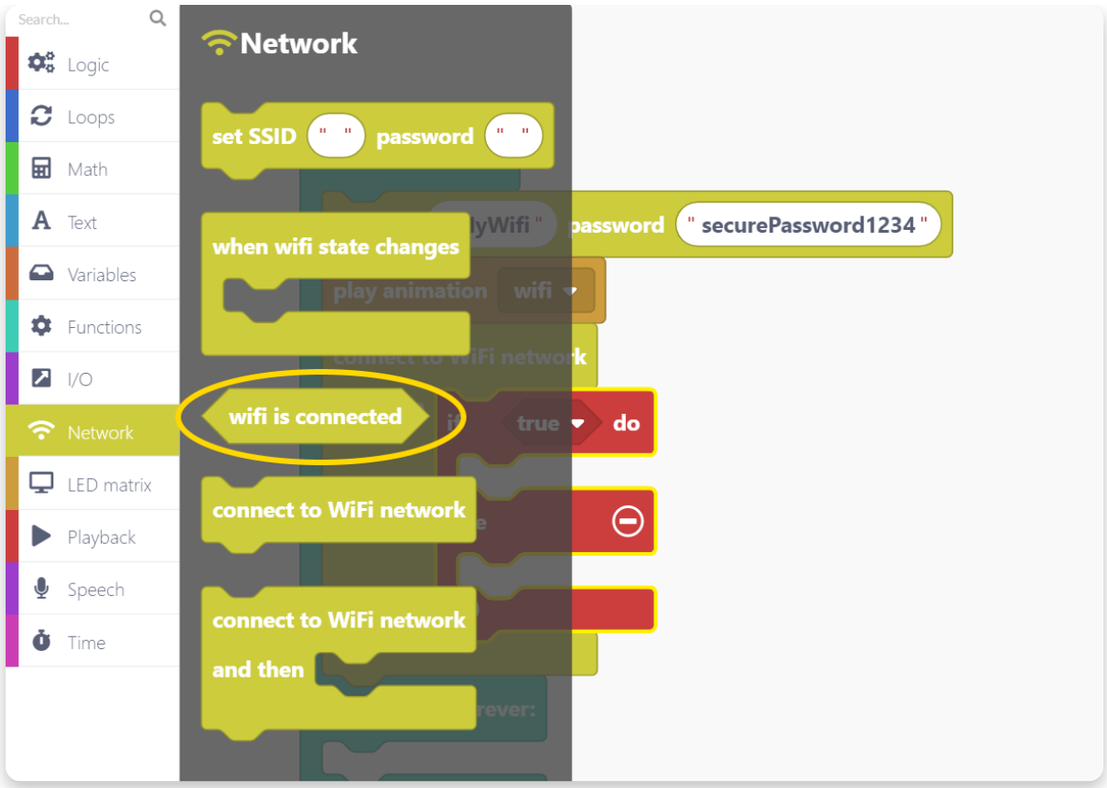
Place the block here:



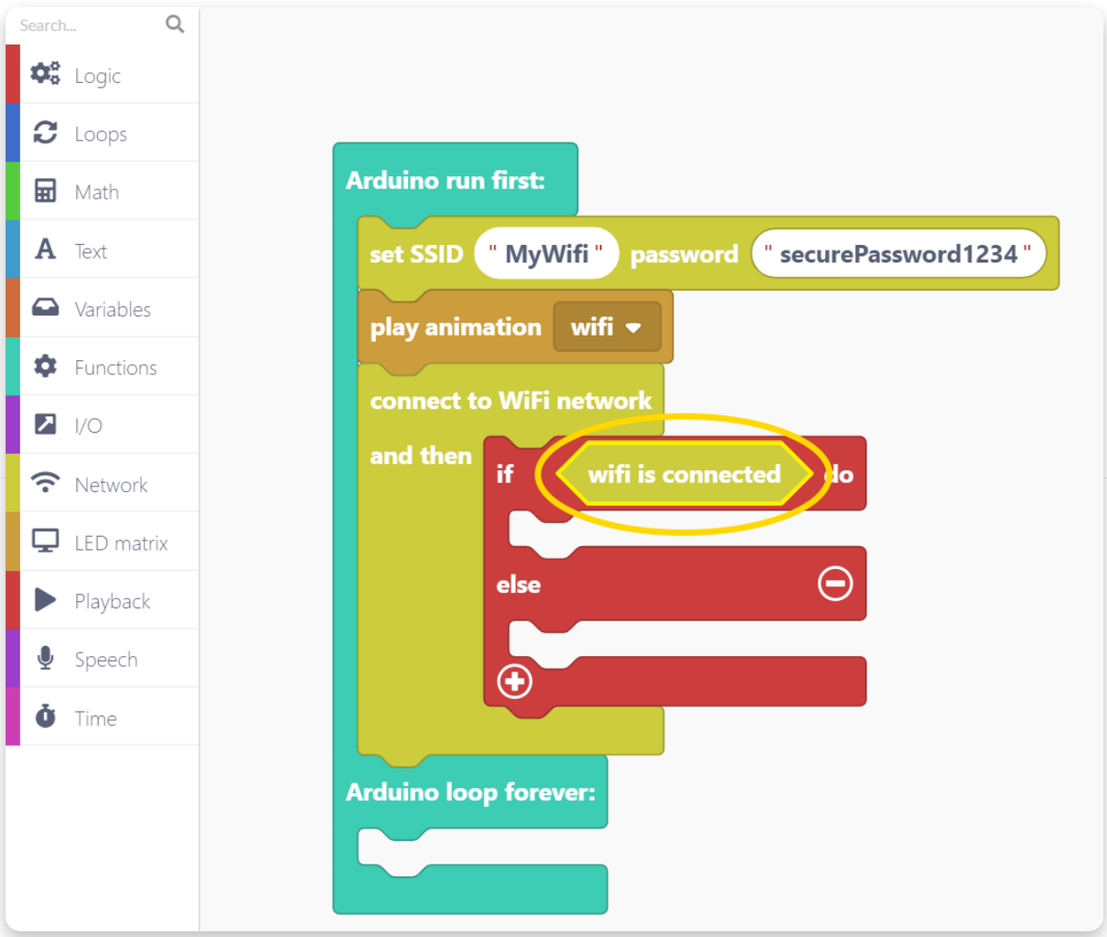
Now we'll need the logic IF-THEN-ELSE block. We'll use this block to check whether the WiFi network connection was established successfully.



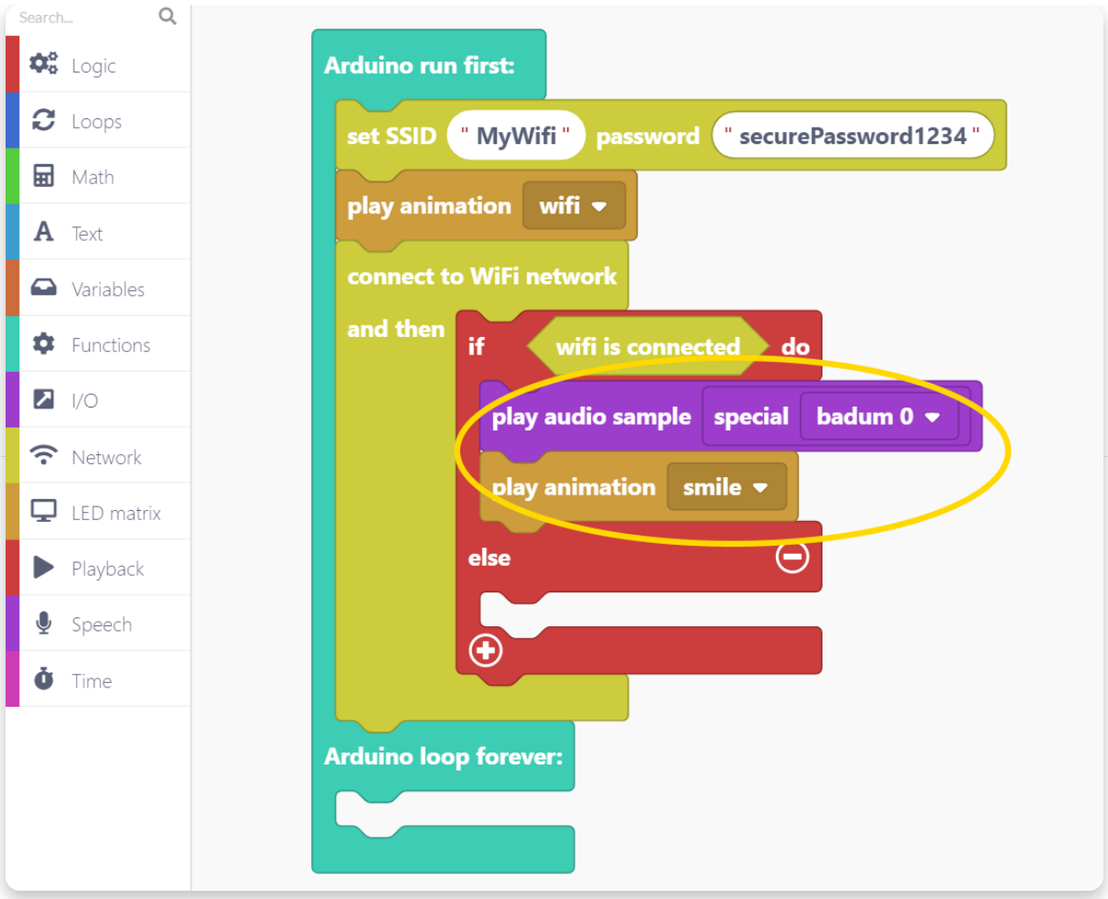
Take the "wifi is connected" condition from the "Network" section.



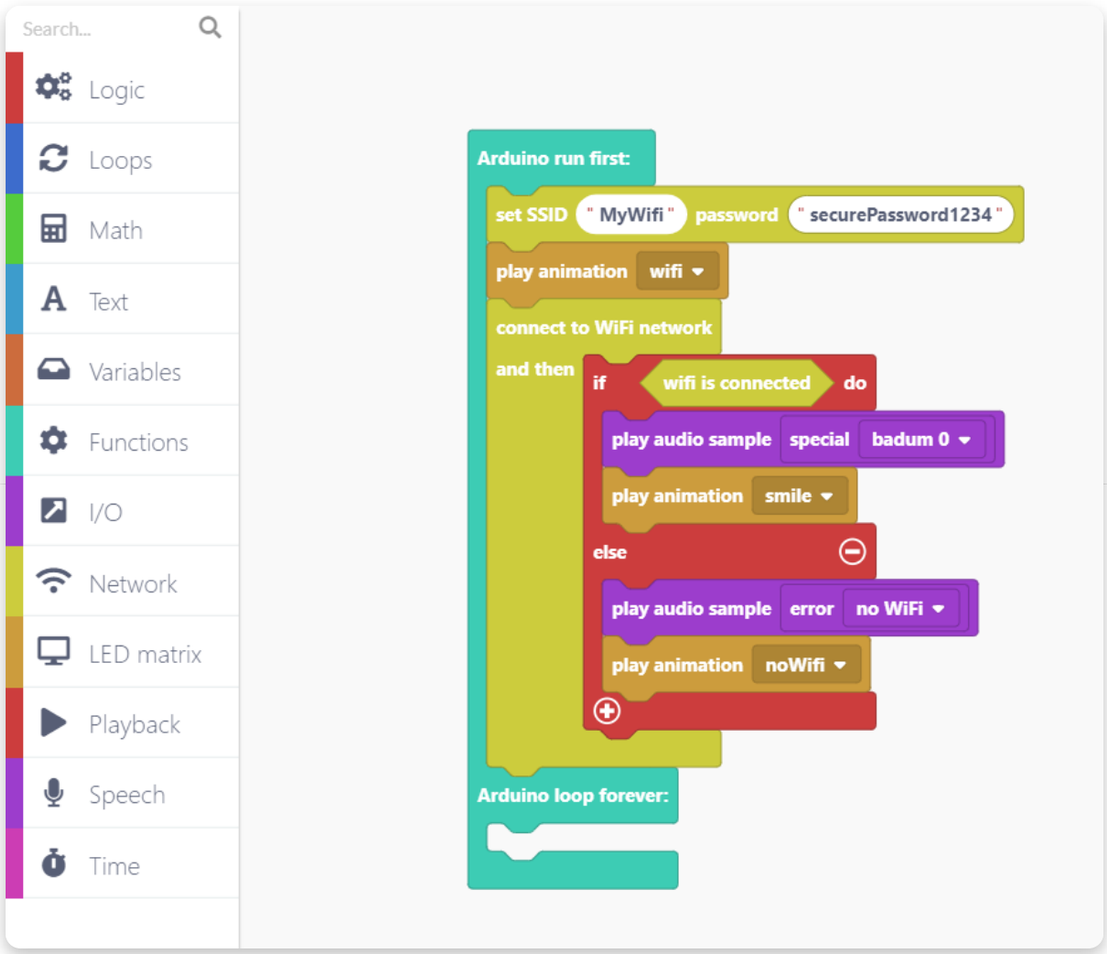
Place it into the logic IF-THEN-ELSE block as a condition like this:



Let's play a specific sound sample and animation if the wifi connection is successfully established.
I've chosen the smiling animation and a sound effect named "badum 0" because it sounded funny:

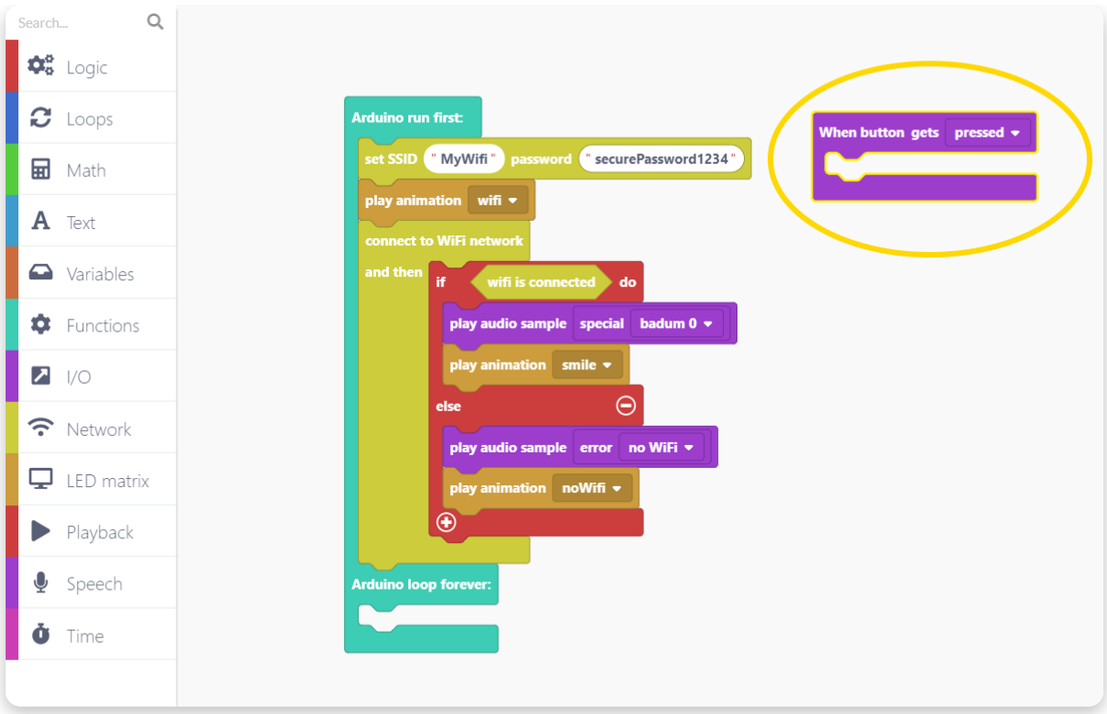


If the WiFi internet connection is not successfully established, let's play the animation and audio sample called "no WiFi".

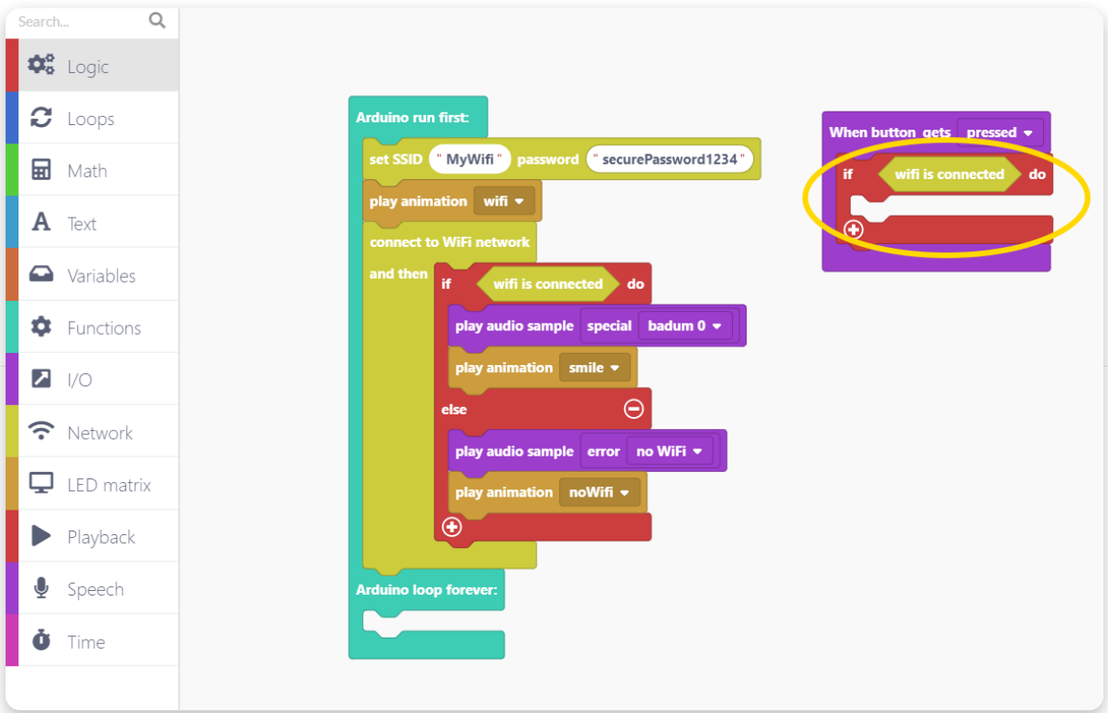


This program will proceed if Spencer successfully connects to the network. If not, Spencer will tell you that it's having trouble connecting to the WiFi network.

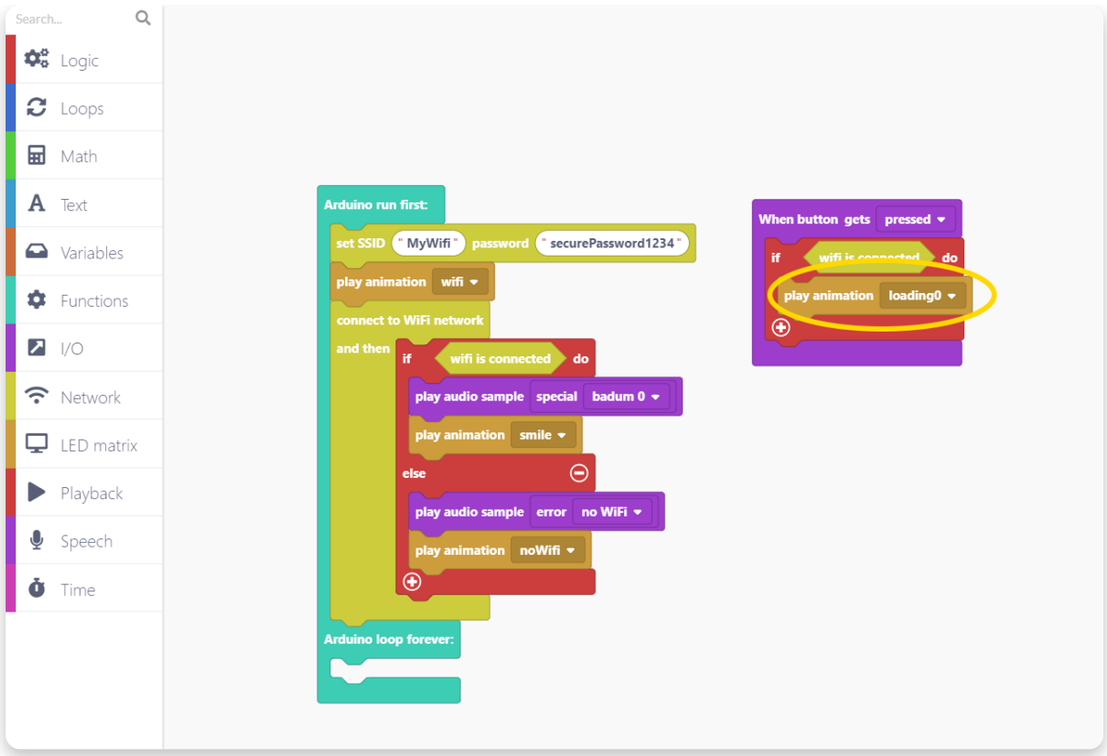
Now let's add the "When button gets pressed" event block:



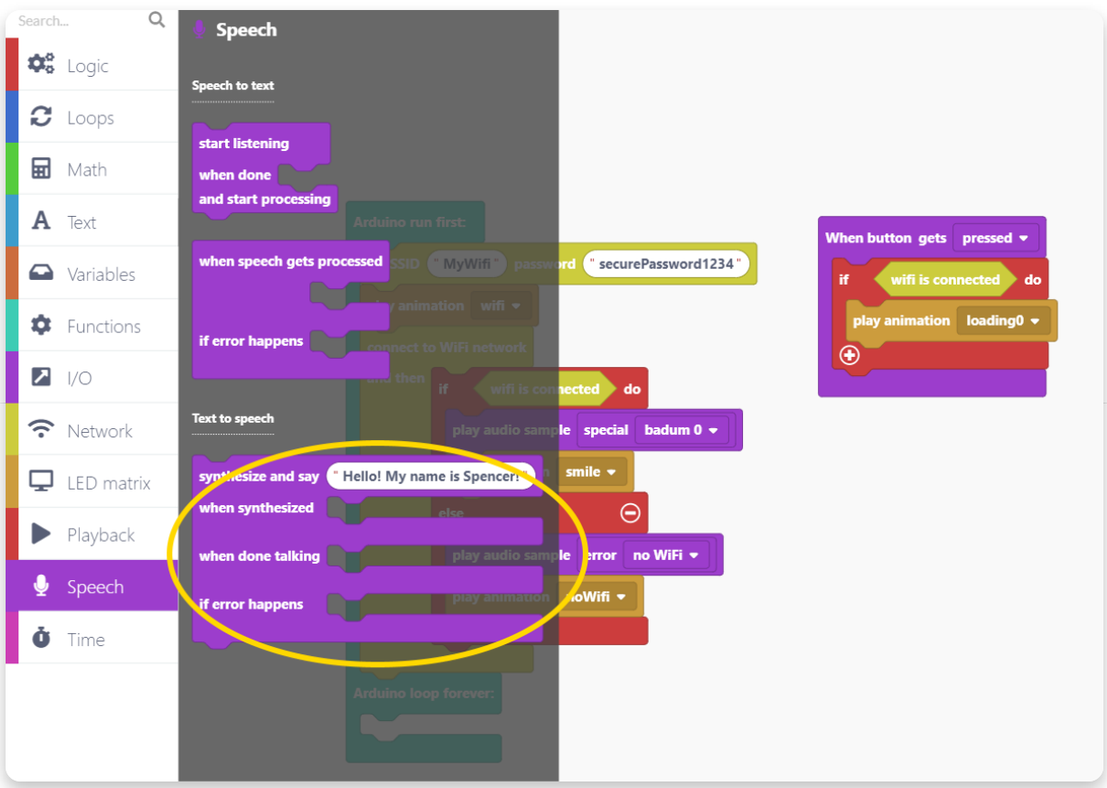
Check whether WiFi is connected every time the button gets pressed. Do it with the same IF-THEN logic block and put "wifi is connected" as its condition:



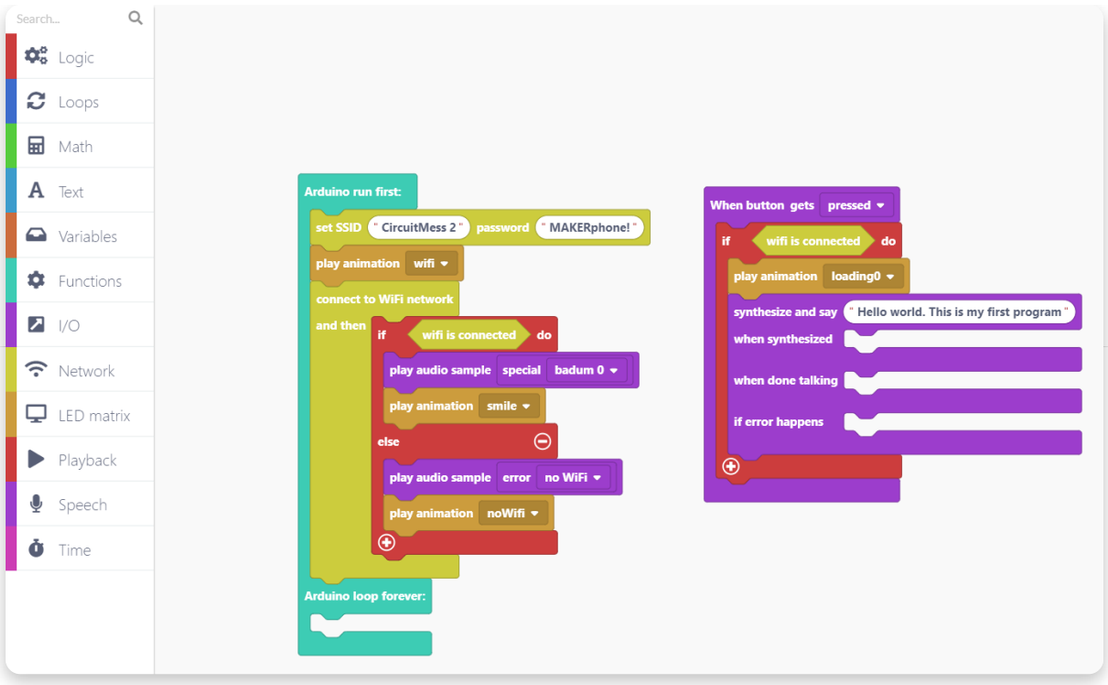
Spencer will need a short time to generate the audio sample that we want. Let's play a loading animation while the sample is generating:



For the next step, open the category called "Speech" and find the "synthesize and say" block.



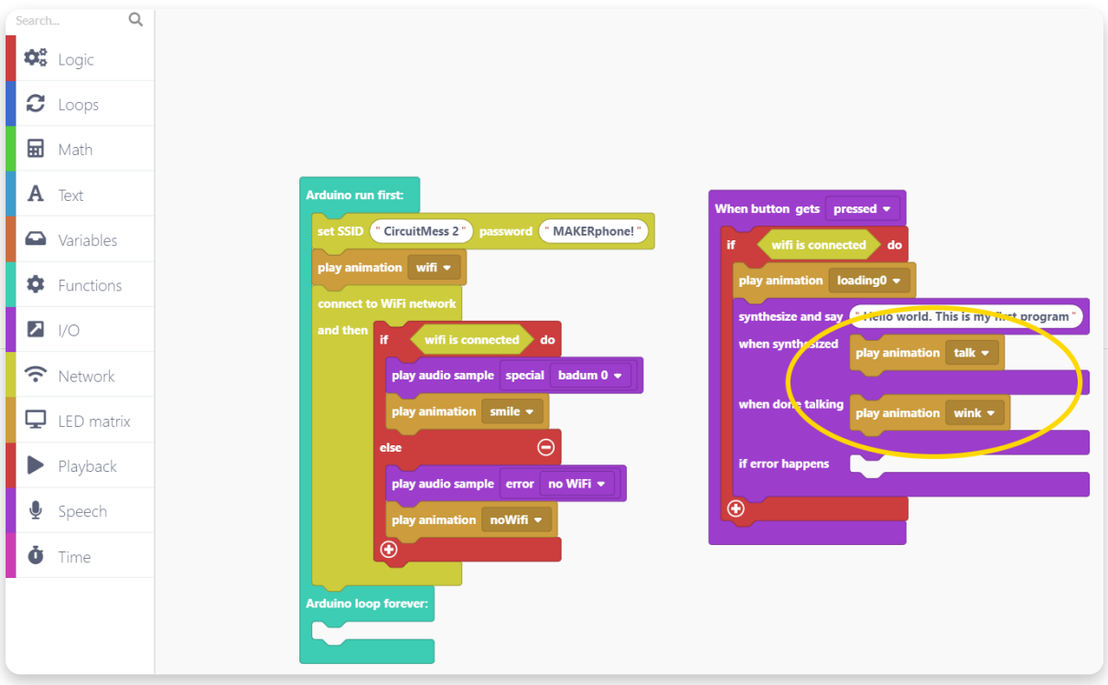
Place the block here and type in the sentence you want Spencer to say. I've typed in "Hellow world. This is my first program".



When the dialogue voice sample gets generated (synthesized), it will be automatically played.

When that happens, let's make spencer's face look like he's talking by playing the animation named "talk".

When Spencer finishes talking, I'm going to play the animation called "wink".



Remember to save your hard work by pressing the save button on the top left.

Run the code and upload it to your Spencer. Every time you press Spencer's button, it should read aloud the text you have written into the "Synthesize and say" block.

If your Spencer is having trouble connecting to the Wi-Fi network, check the network name and password and try turning your spencer off and on again.

> **Remember, Spencer only works with 2.4GHz (IEEE 802.11 b/g/n) WiFi access points.**
> **5GHz (IEEE 802.11 ac) networks, unfortunately, aren't supported.**

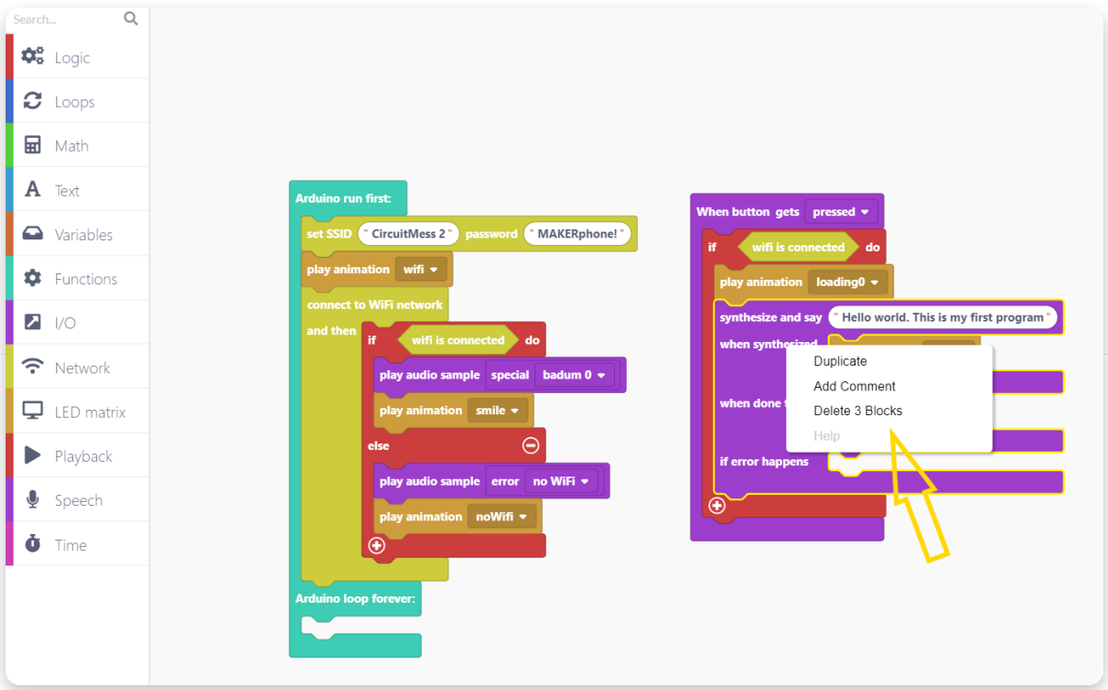
If you're having trouble making your Spencer talk, reach out to us via contact@circuitmess.com, and we'll help.

Make Spencer listen and repeat

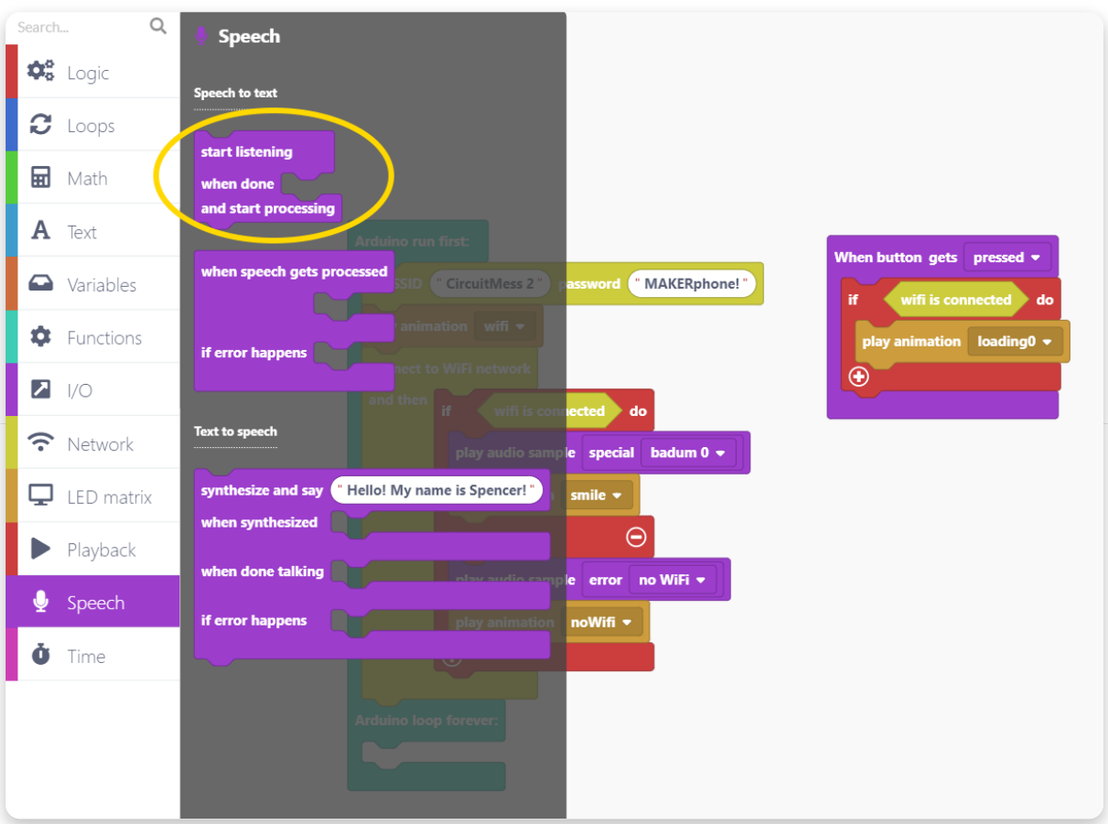
For the ultimate coding challenge, let's make Spencer listen to what we say and repeat that same sentence with his funny robotic voice.

We can just continue with the code you have made in the previous chapter for this one.

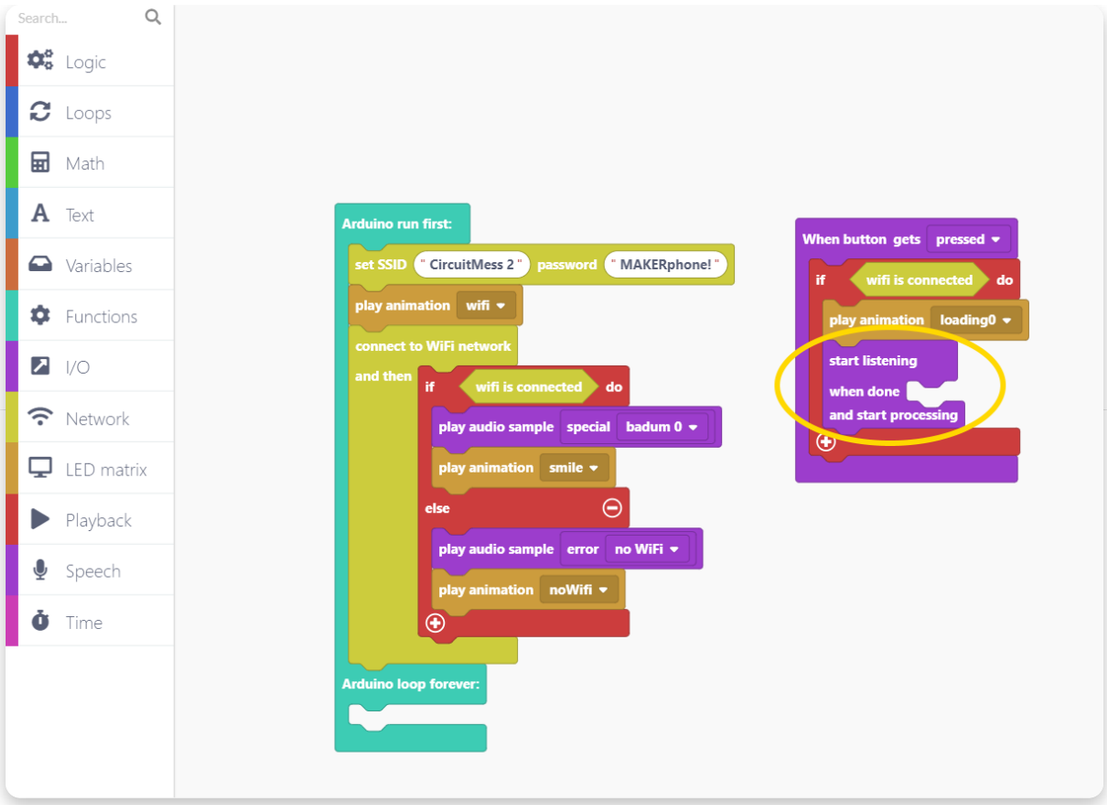
Firstly, delete the "synthesize and say" block by right-clicking on it and pressing the delete option.



So now we need the "start listening" block.

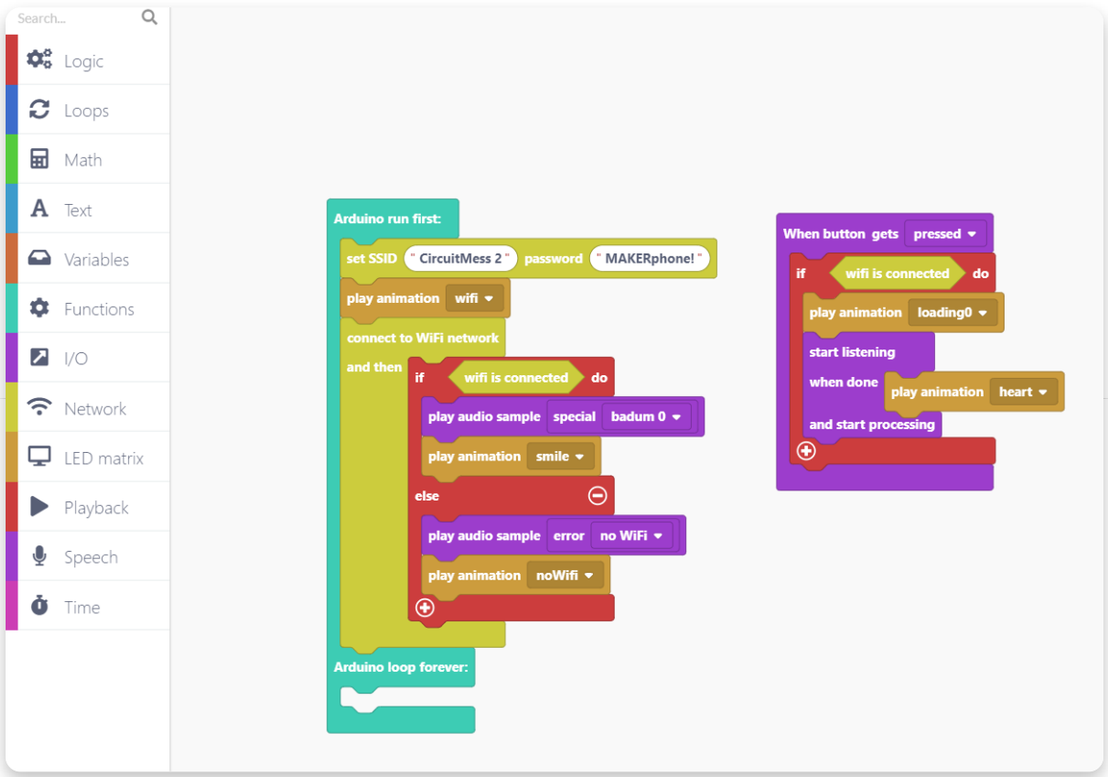


Place it here:

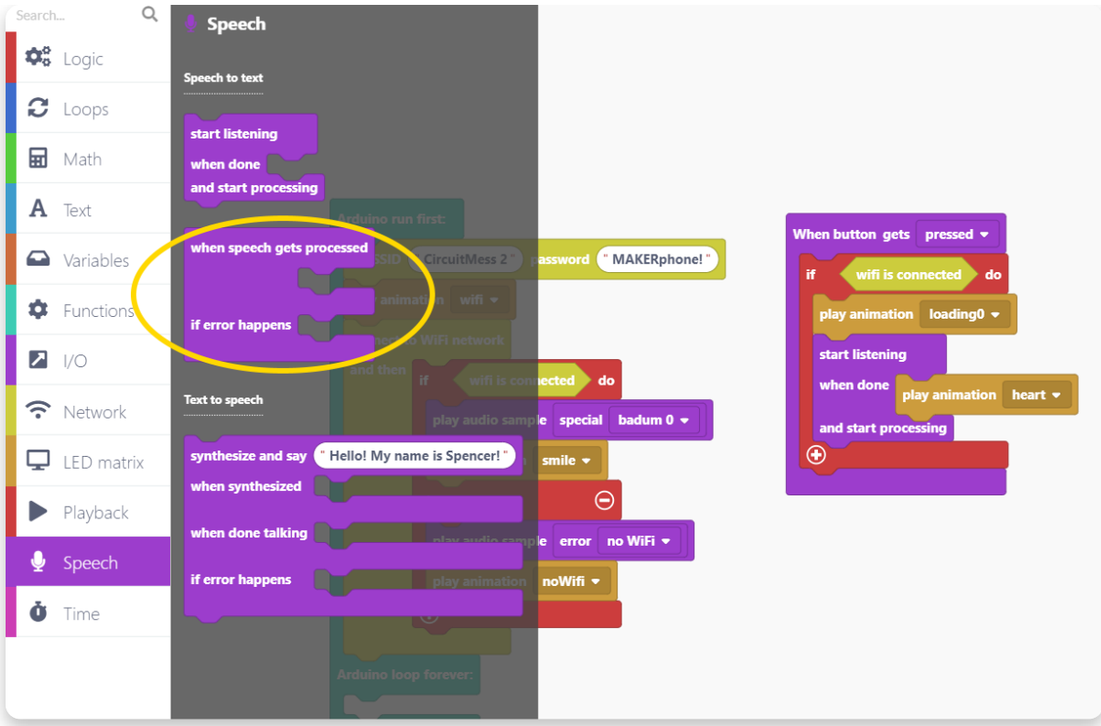


With this code, every time you press its red button, Spencer is going to check if WiFi is connected, play the loading animation, and start listening to what you say.

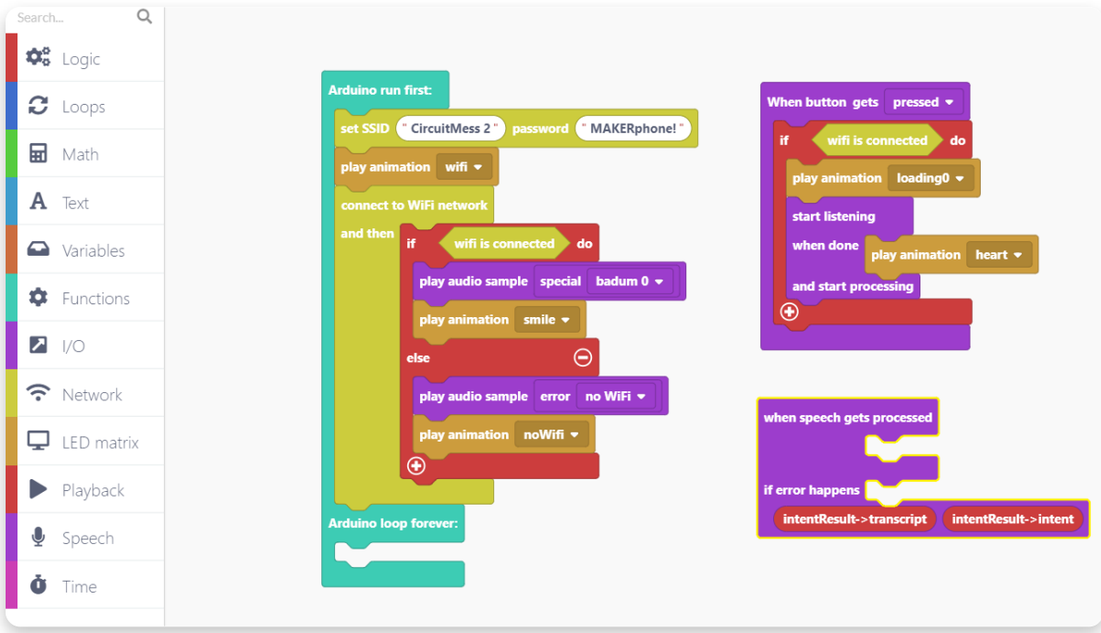
When Spencer is done listening, let's play an animation so that we know it's now processing our voice command.



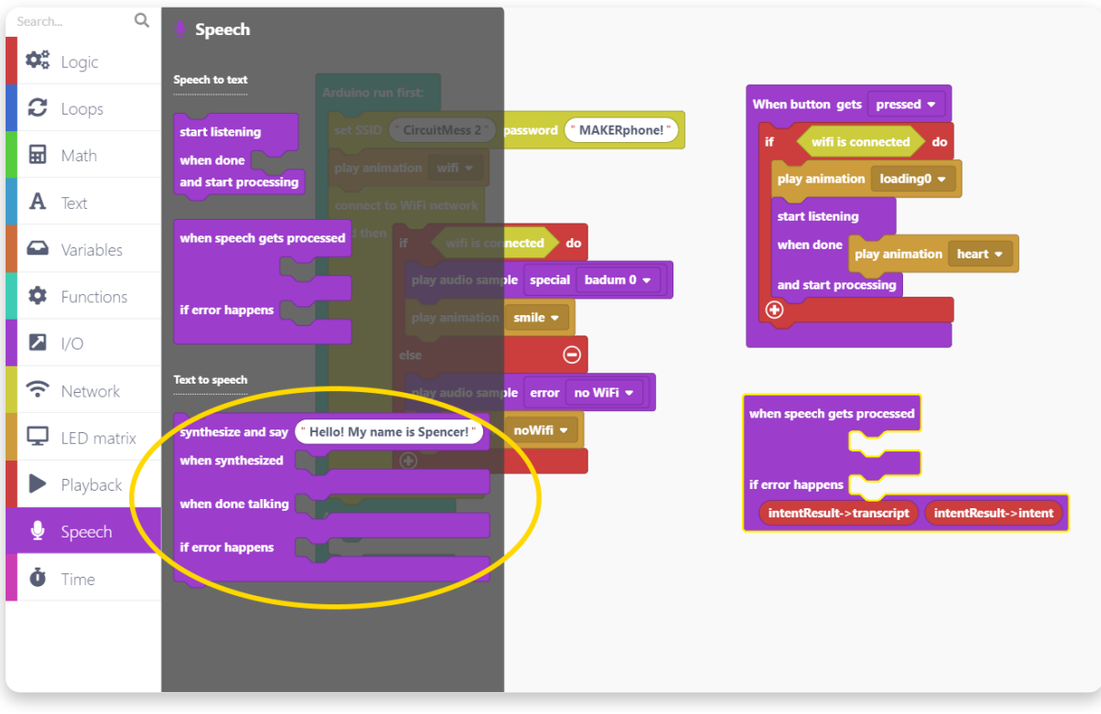
Now find the "when speech gets processed" event block. This block will be triggered once Spencer processes your voice command and receives a transcript from the server.



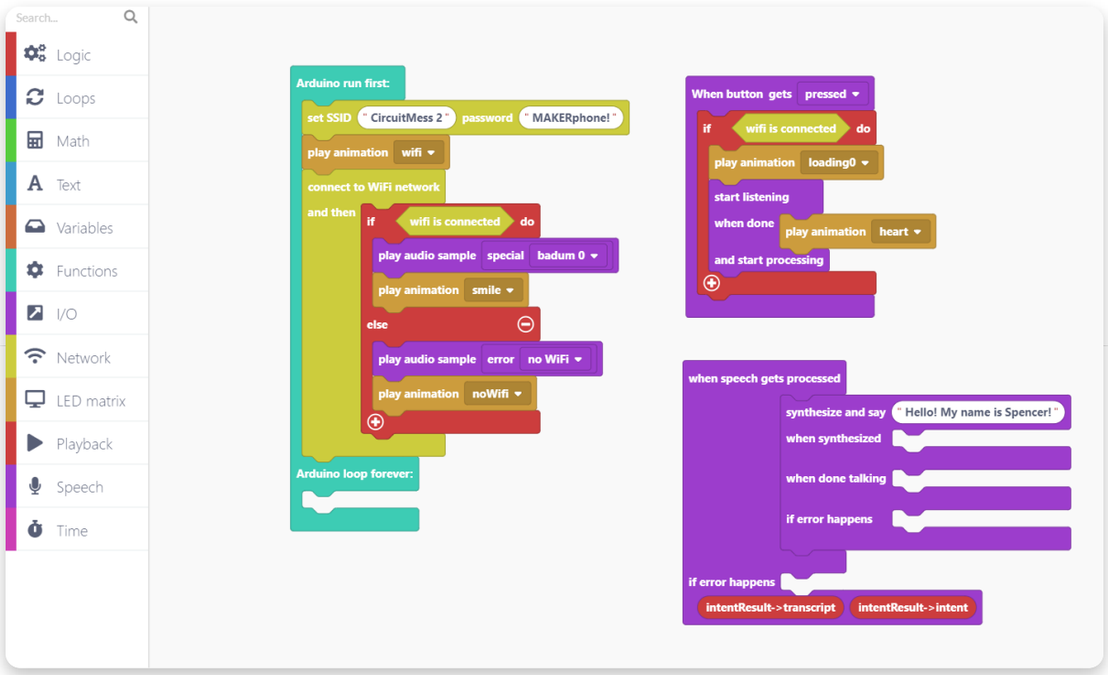
Drag and drop it onto the drawing area:



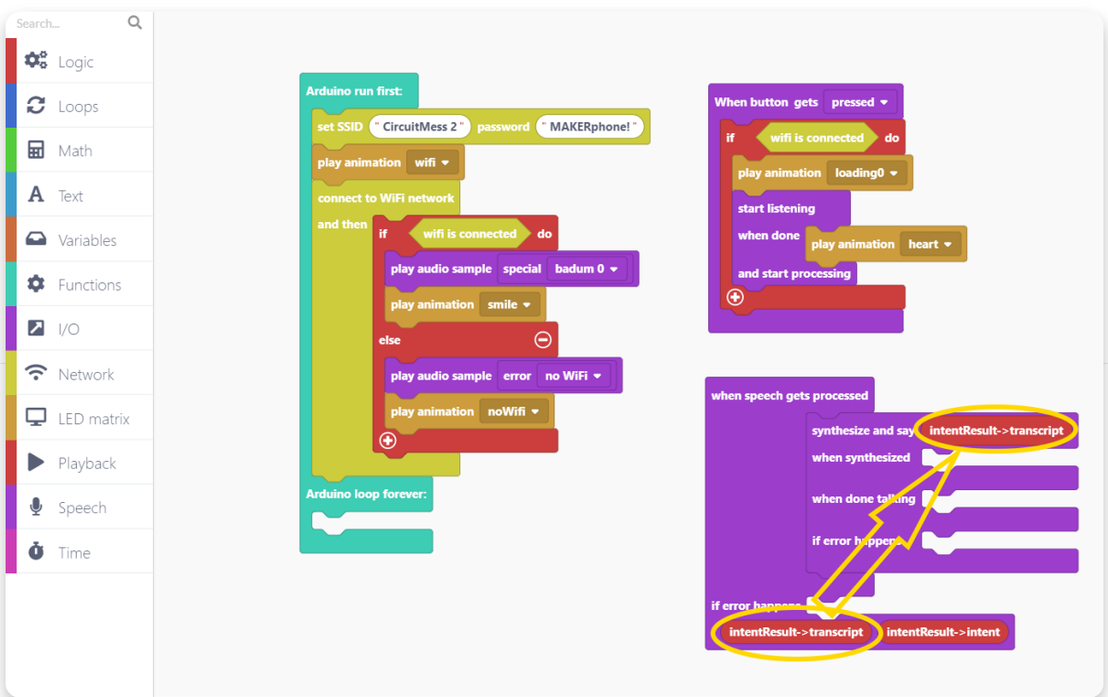
Since we want Spencer to repeat what you've said to him, let's find the "Synthesize and say" block:



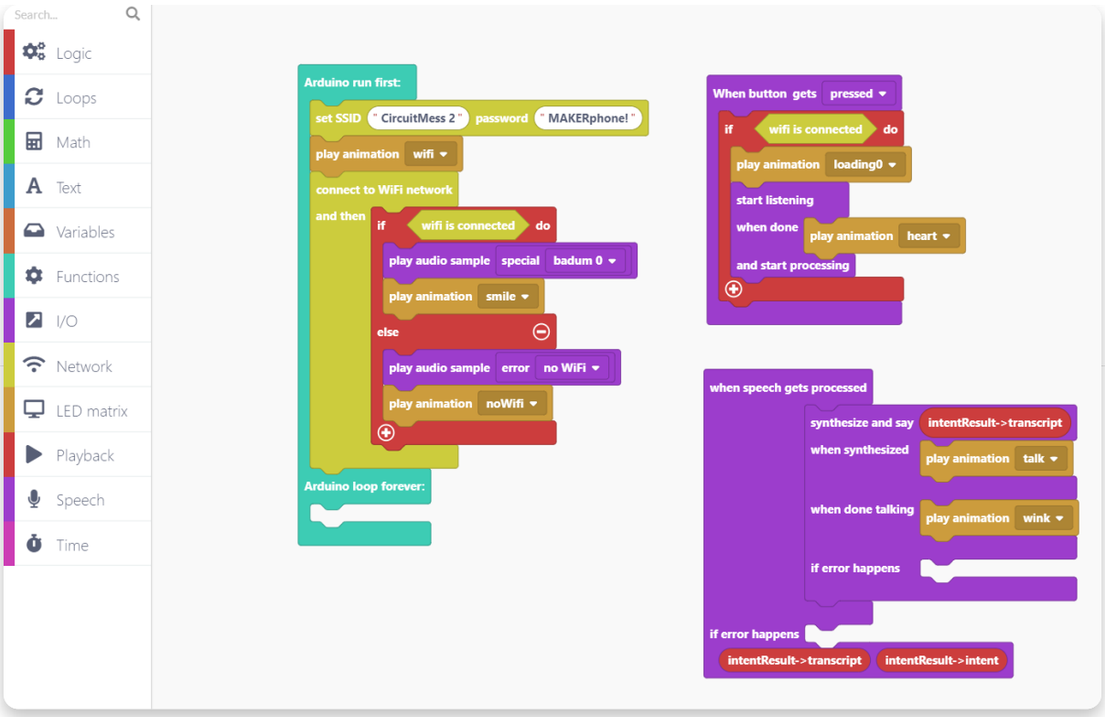
Place it here since we want to make Spencer repeat your words as soon as your voice gets processed:



We want Spencer to say the exact thing that you have said. That's why you need to take the red "intentResult->transcript" value block and drop it into the "synthesize and say" block.



Finally, let's play a talking animation when Spencer's voice sample gets generated and make him wink once he's done talking.



Awesome, your code is done now.

Save it and run it on your Spencer.

Smack Spencer's head and tell him something (i.e., "Hello Spencer, you are my friend"). Spencer will process your voice command and repeat it with his robotic voice.

Restore Spencer's base firmware

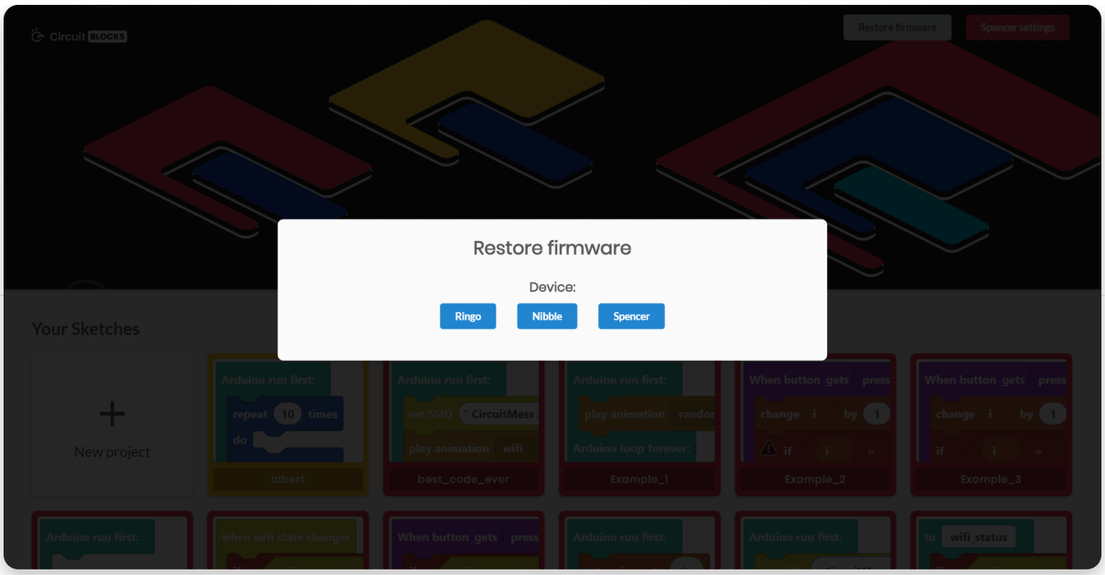
Restore Spencer's firmware

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

This is quite simple, just connect your Spencer 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 Spencer, of course.



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

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

Loading...