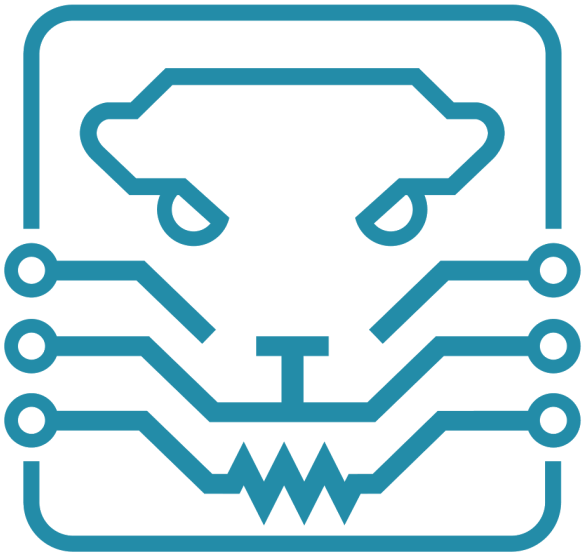


PREDICATE

ACADEMY

Educator Workshop

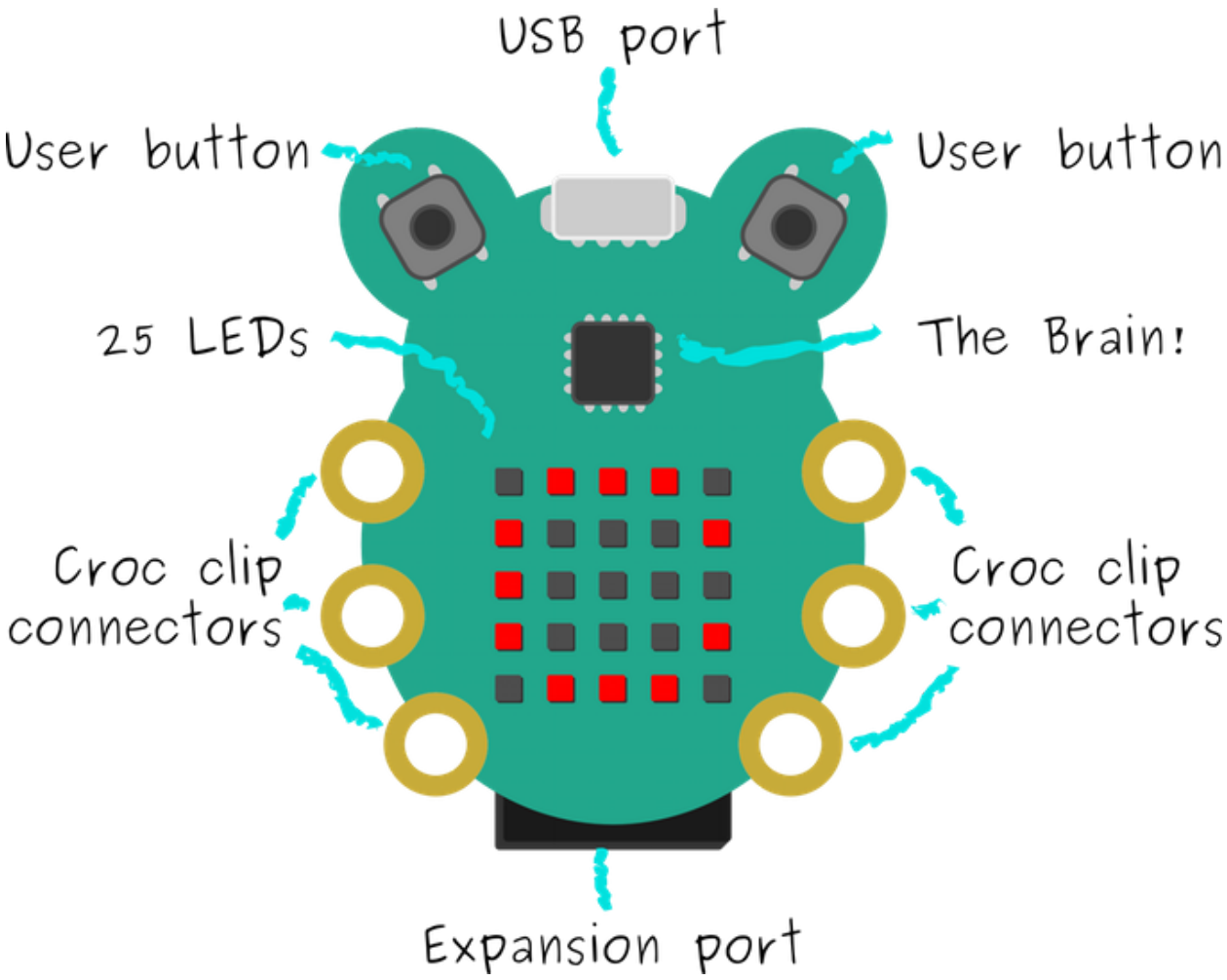
Physical Computing with CodeBugs

CodeBug Introduction

CodeBug is a wearable computer designed to introduce simple programming and electronic concepts to anyone, at any age. Simply put - it is an AMAZING educational device that is designed specifically to bring awareness to the power of writing code. CodeBug can display graphics and text, has touch sensitive inputs and can be powered with a watch battery. The CodeBug is programmable from a website, which features colourful drag and drop blocks, an in-browser emulator and engaging community features. Create your own games, clothes, robots or any other wacky inventions you have in mind!

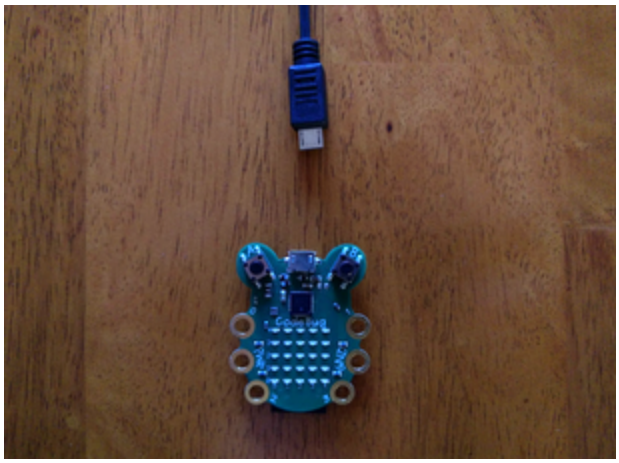
The CodeBug features:

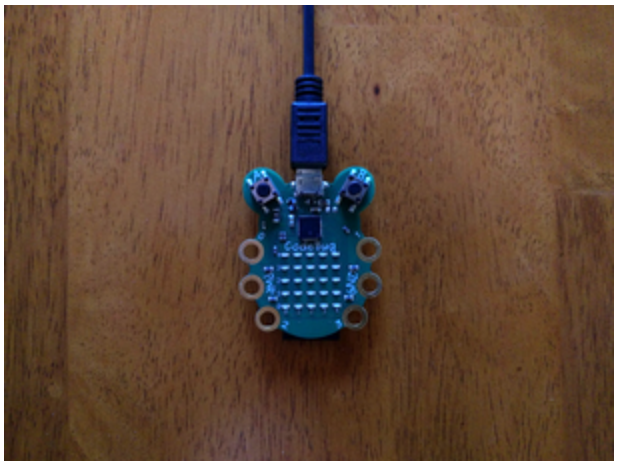
- 2 buttons
- 25 programmable LEDs
- 6 touch sensitive input pads
- CR2032 battery power support

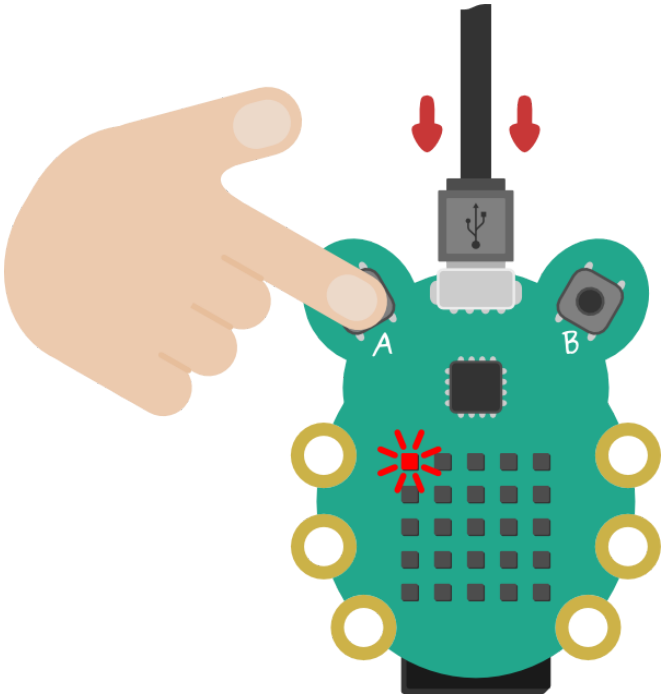


Plugging in the CodeBug

You must connect the CodeBug to a computer before you can start programming it. You can do so with the provided USB cable. **Connect the small end of the USB cable** to the upwards facing port on the CodeBug. The plug can only be inserted in one direction. **CAUTION - the top part of the bug is fragile. We suggest mounting the device to a board for additional stability. Also, this bug conducts electricity. Avoid placing on metal surfaces.**







Press and hold the A button on the CodeBug while connecting the other end of the provided cable to any available USB port.

If you see a **flashing red light on your CodeBug**, then it has successfully been connected.

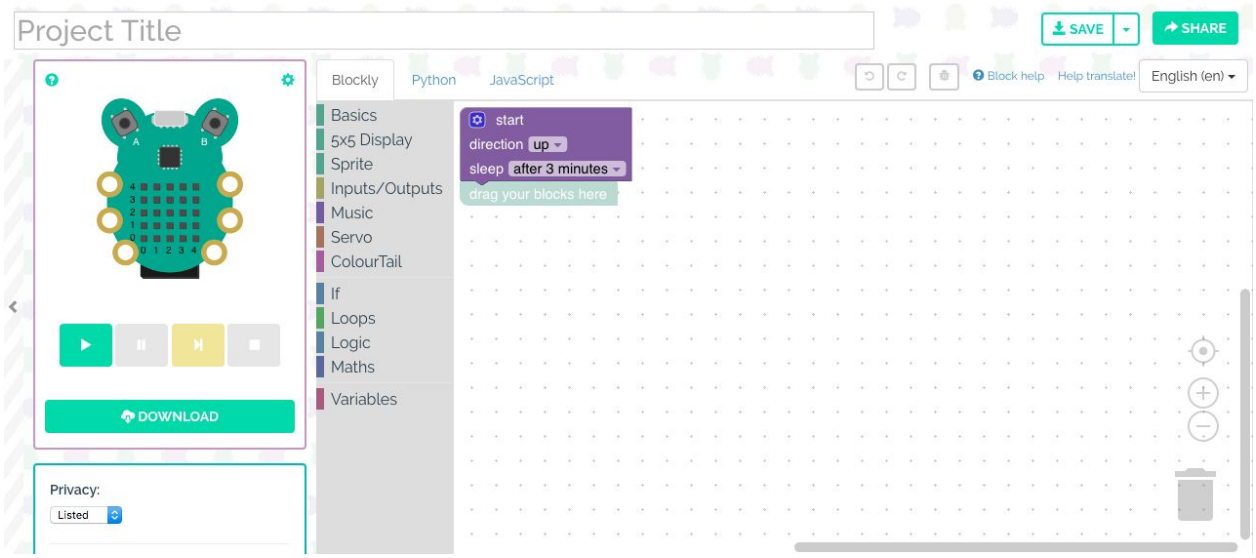
CodeBug Website

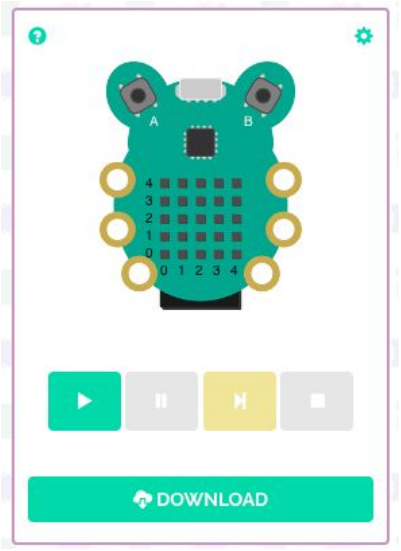
All programming for the CodeBug is done via the official website.

Open http://www.codebug.co.uk in your internet browser.

Click on the "Create" tab located at the top left of the website.

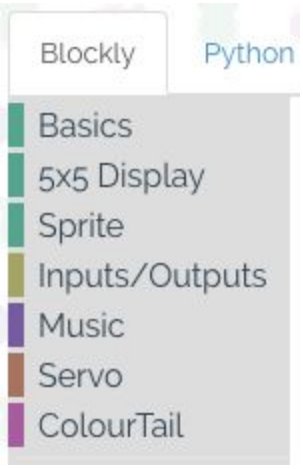
You will now be greeted by a **drag-n-drop block based interface**. This is where all your programming of the CodeBug will be done.





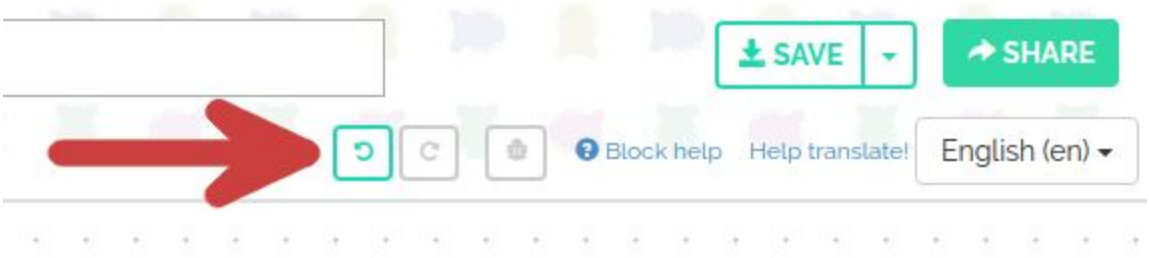
On the left-hand side of the screen, you have the **CodeBug Emulator**. We can use this to test our programs before loading them onto a physical CodeBug.

Just to the right of the emulator, we have the **Block Menus**. These menus contain different coding blocks according to their category.



Lastly, towards the center of your screen is the **Workspace**. This is where we will place all the blocks used in our program.

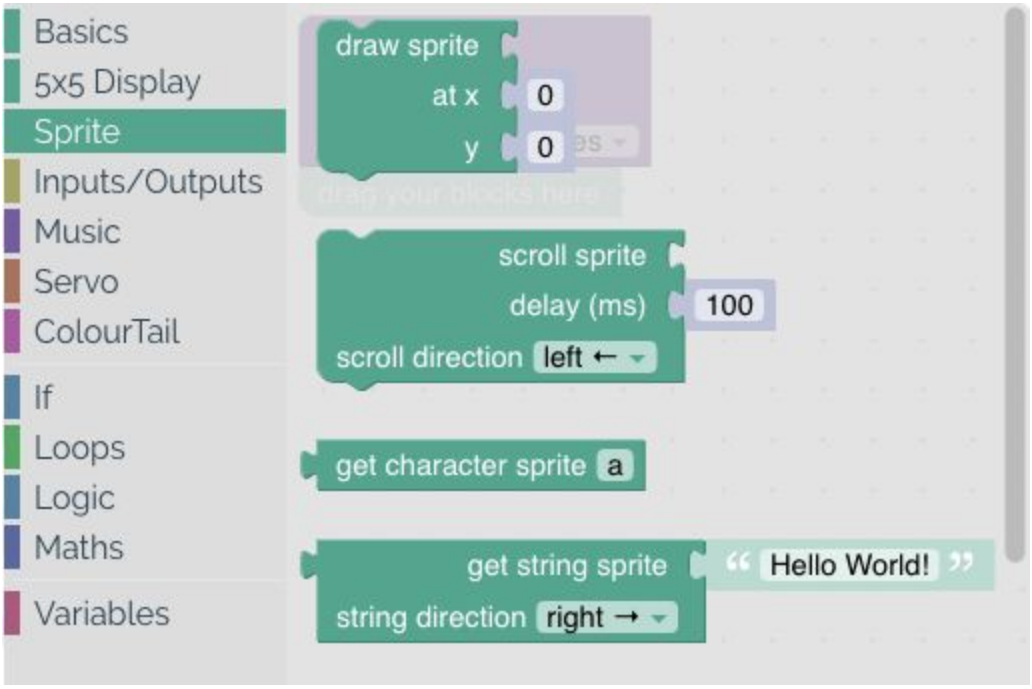
There are very useful **undo and redo buttons** located near the top right of the Workspace. You can use these to reverse any mistakes made when creating your programs.



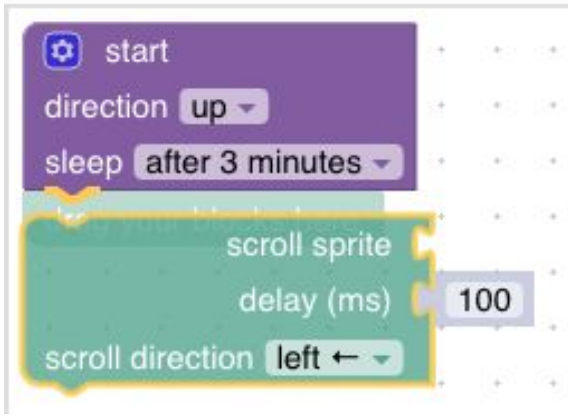
TIP: SAVE frequently and use the undo/redo to clean up any block accidents

Exercise (1) “Hello World” Sprite Control

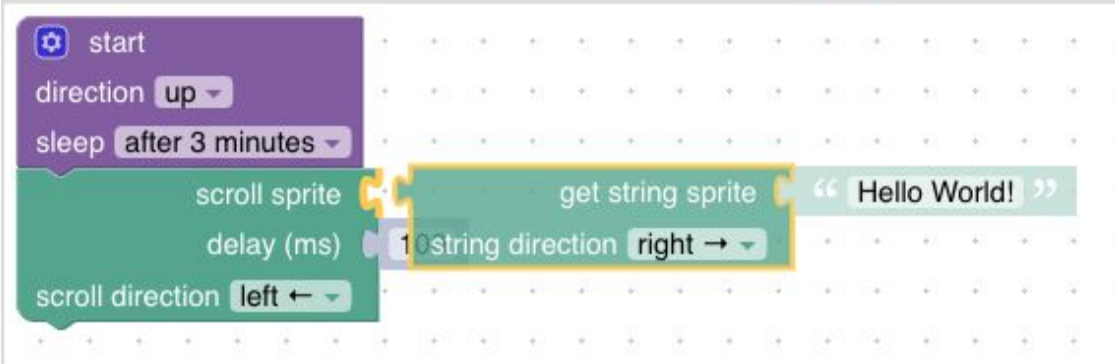
First, **select the “Sprite” Block Menu**. You should then see a list of blocks as seen below.



Drag the “scroll sprite” block to the workplace and snap it under the“start” block. Do this by clicking and holding the “scroll sprite” block and moving your cursor so that the top edge of the block you’re dragging aligns with the bottom edge of the purple “start” block. You should see an orange highlight just before releasing the block.

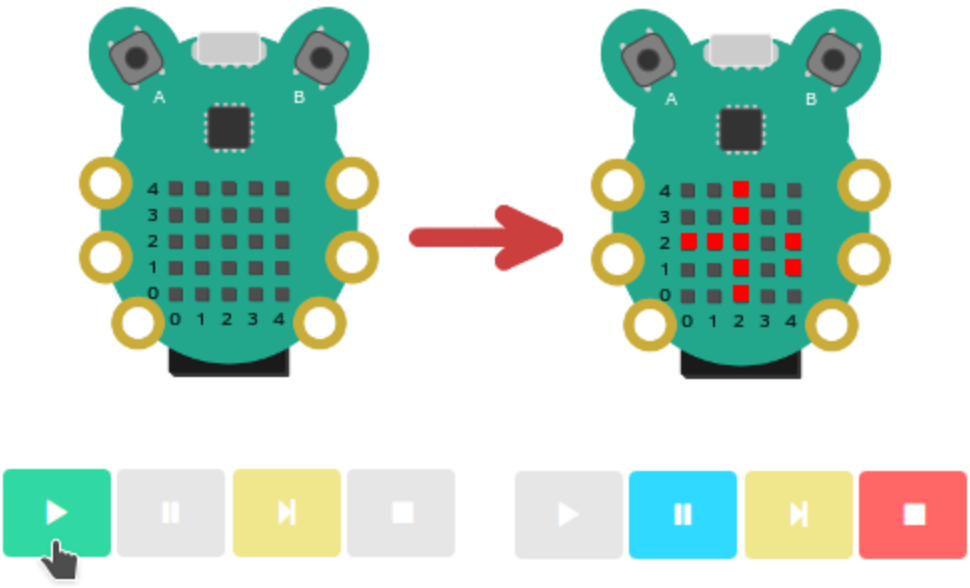


Next, **open the "Sprite" Block Menu again and drag the "get string sprite" block onto your workplace**. Snap this block to the "scroll sprite" block you just created.



You can now **click the green play button beneath the CodeBug Emulator** located at the left side of your screen. You will see the text "Hello World!" scroll across the CodeBug Emulator.

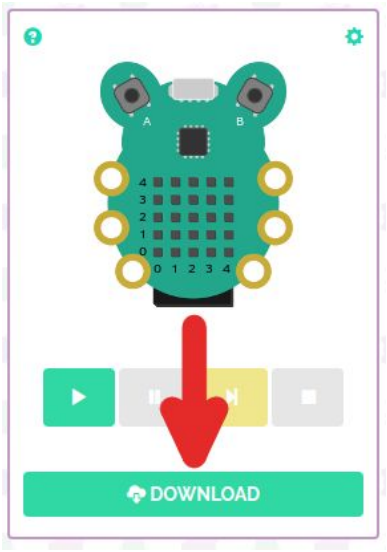
TIP: It's always good to emulate your code to identify any early defects

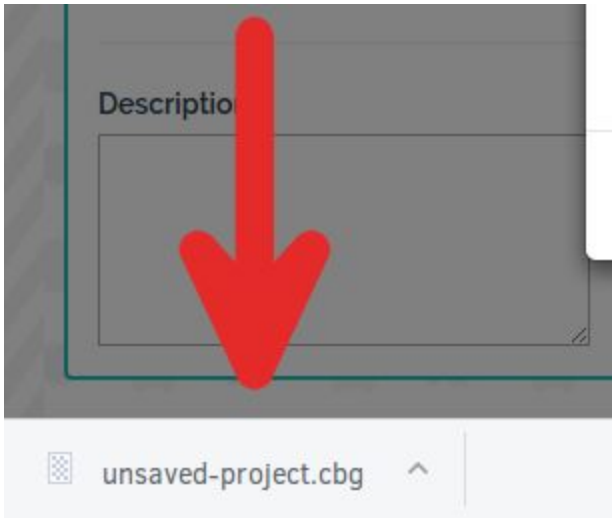


Loading Your Code

If you still see a **blinking red light on your CodeBug**, then it is programming mode and ready for your code to be loaded.

First, **press the green download button** located underneath the CodeBug Emulator.

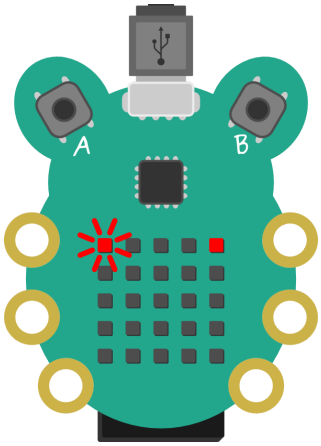


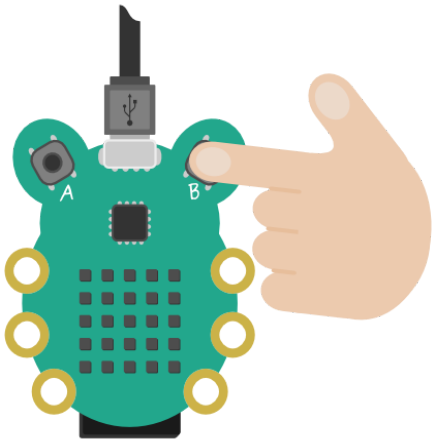


You should notice an **"unsaved-project.cbg" file has been downloaded**. Along with a pop-up inside the CodeBug web interface about loading your CodeBug.

You'll want to **click and drag this file onto your CodeBug device**. Treat this exactly as you would moving a file to a USB flash drive. The procedure varies slightly depending on your operating system. On MacOS you can find the CodeBug device on your desktop or in Finder. Using Windows or Chrome OS, it can be found in the "Folders" application. If you can't find the "unsaved-project.cbg" in your browser, you may find it in your "Downloads" folder.

You should now **see a solid red light**, along with the flashing light on your CodeBug. This means your code was loaded successfully.





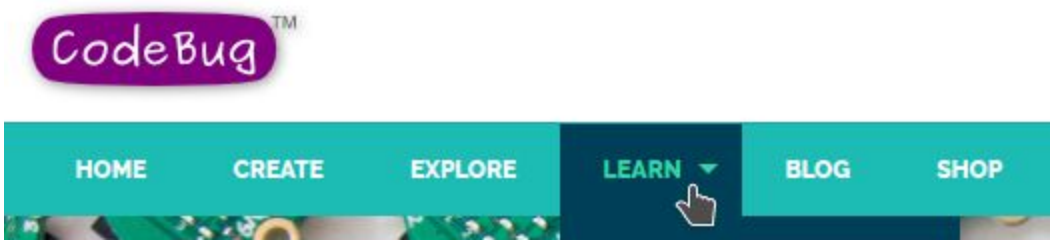
To run your program, **press the B button** on the CodeBug. **TIP: Be sure not to unplug the CodeBug at this stage, as the USB connection is still needed to power the device.**

If you want to make any changes to your code, you'll need to repeat the previous steps in this section.

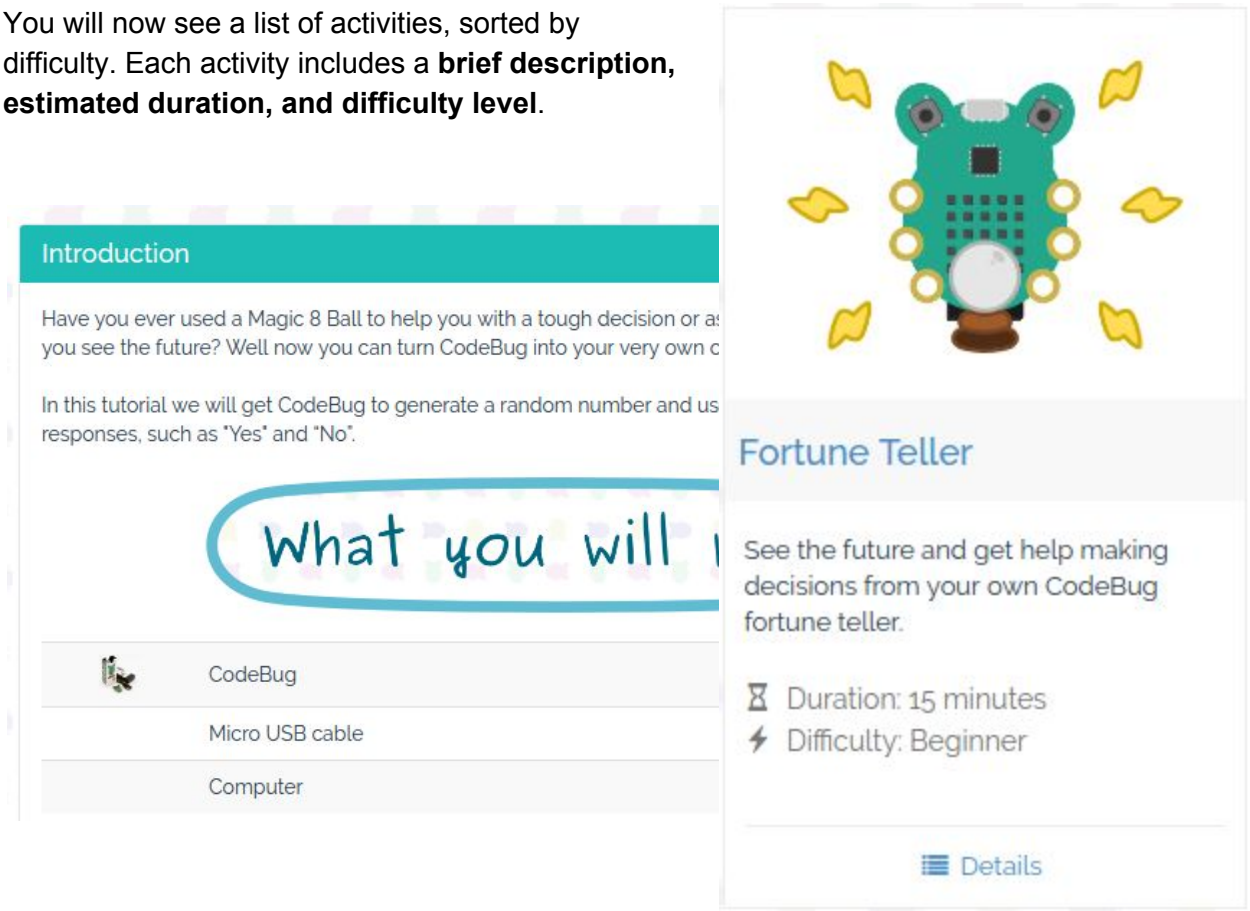
Navigating Self Guided Tutorials and Curriculum

There are many tutorials of varying difficulty on the CodeBug website. To access them, **open** http://www.codebug.co.uk in your internet browser.

Click on "Learn" at the top of the webpage. Alternatively, you can click the "Activities" link that appears when hovering atop "Learn".



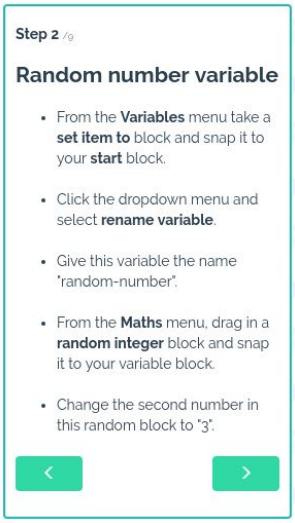
You will now see a list of activities, sorted by difficulty. Each activity includes a **brief description, estimated duration, and difficulty level**.



Click “Details” on any activity to be brought to a page with a more in-depth introduction and tutorial. To read more about the activity, **click the “+” button** located next to right of each section.

To start the coding tutorial, **click the tutorial name**, which is “Random answer” in this case.

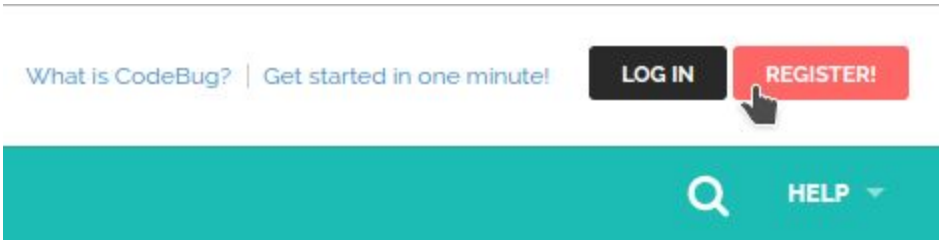
You should now see what looks much like the programming interface we used earlier, but with an added sidebar on the right. **Use the green left and right arrows to navigate the tutorial** after completing each step.



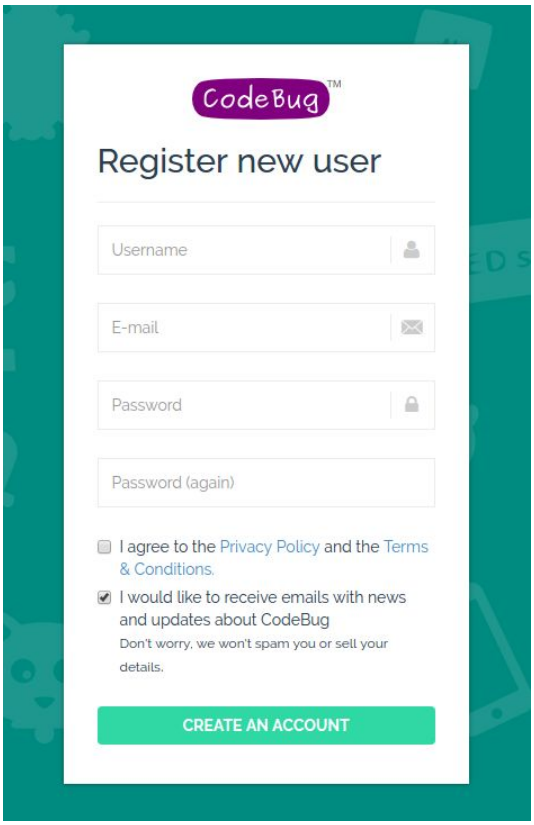
Creating an Account

Creating an account is very useful for **saving and sharing** CodeBug projects. You can create one account for your students to share. Or alternatively, encourage your students to create their own accounts and upload their projects.

To begin, **click the "Register" button** located in the top right of the official CodeBug website.

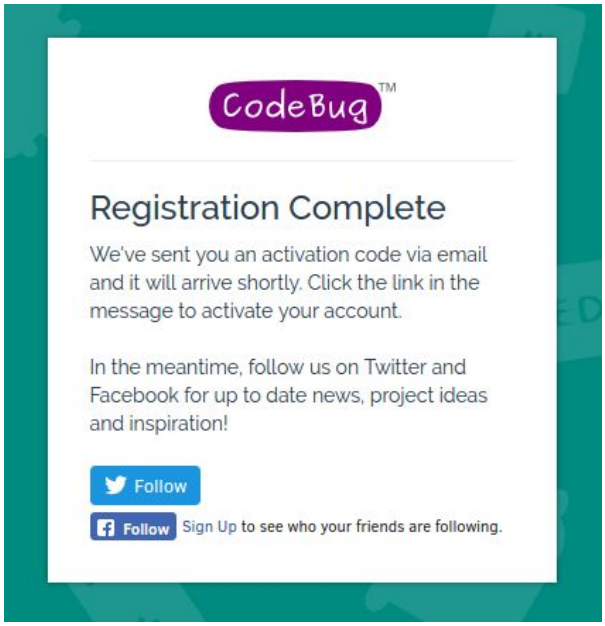


Fill in all the text fields on the Register page. You must read and agree to the "Privacy Policy" and "Terms & Conditions".

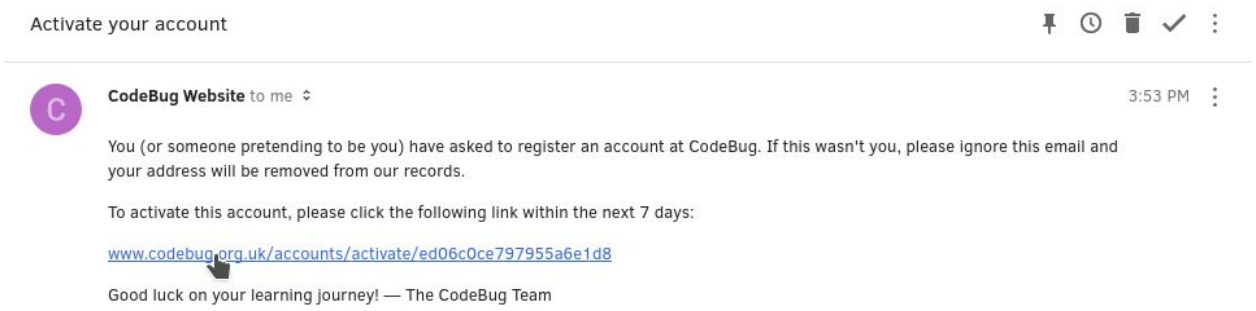


After filling in all required information, **click the green "create an account" button**.

You'll now see a "Registration Complete" page notifying you to confirm your registration via email.

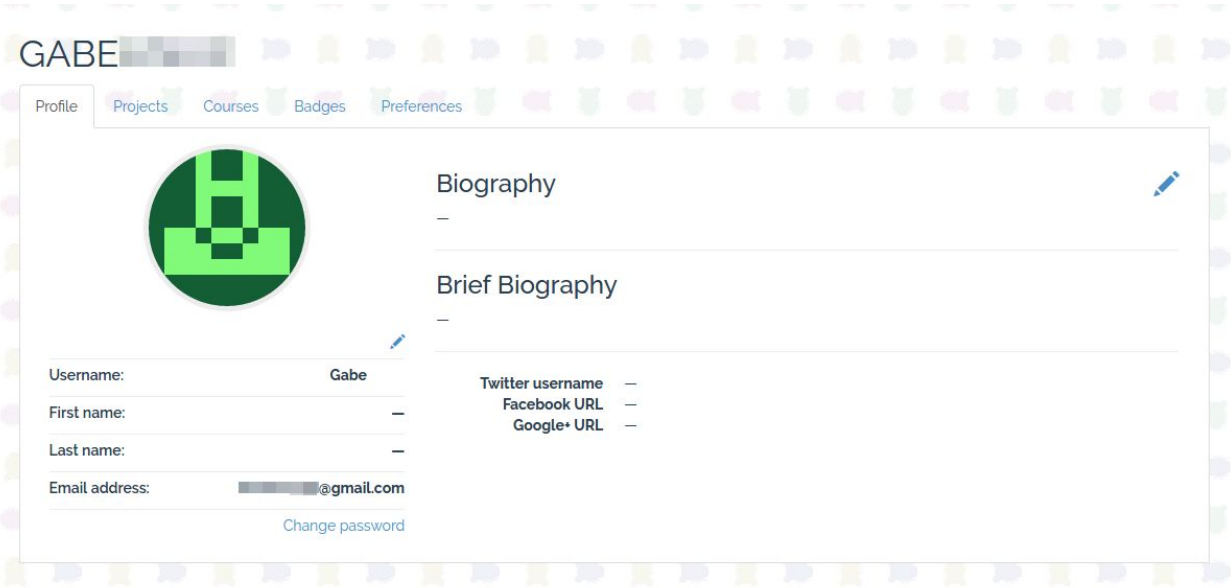


Check your inbox of the email you used to register. You should receive an email from CodeBug with a link to activate your account. **Click the link inside the email** to activate your account.



A page will open up where you have the option of filling in some extra information, such as your name and social accounts. **Either fill in the required information and click "Submit", or press "I'll Do It Later"**.

You'll now be greeted by your public profile page.

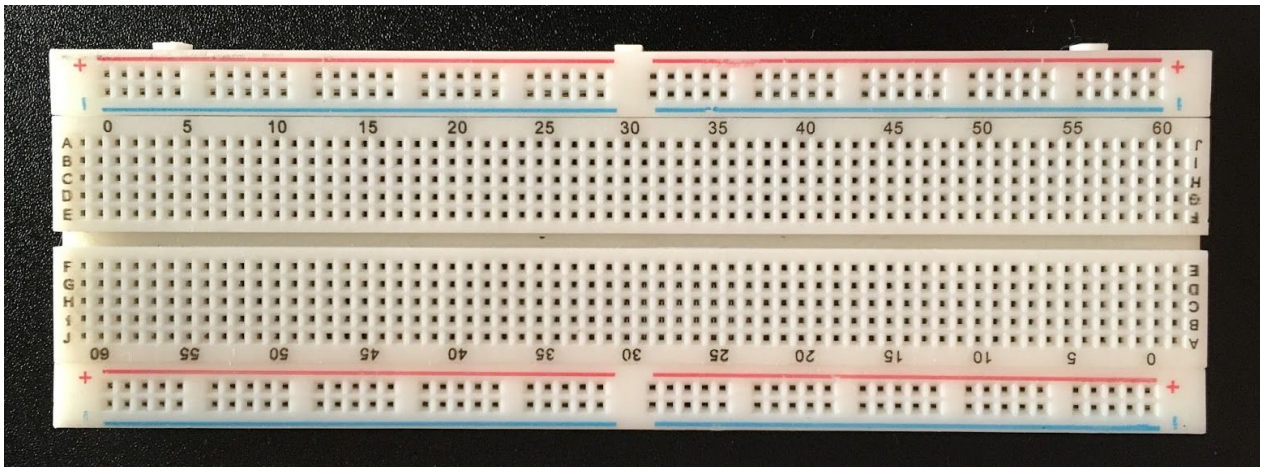


Now that you have a CodeBug account, you may save your projects. **When in "Create", enter a title for your project** in the text box located above the workspace.

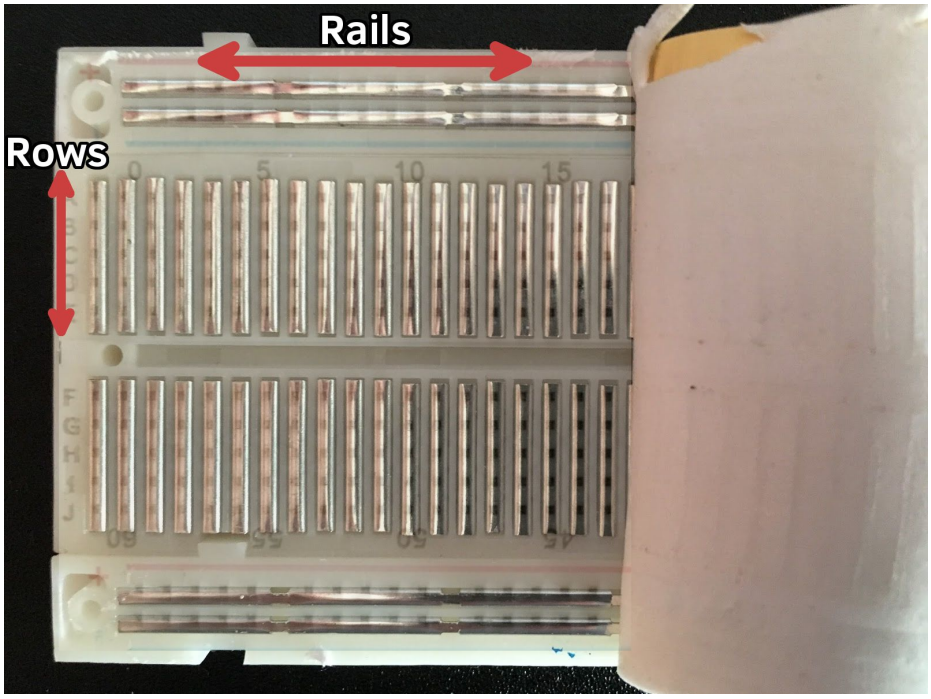
Then **press the "Save" button** located to the right of your project title. You'll want to save your project after making any changes to ensure it stays up-to-date.

Exercise (2) Basic Electronics

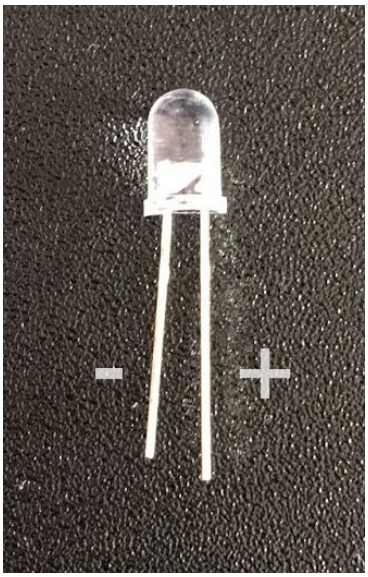
We can use a **Breadboard** to create circuits without having to solder.



The Breadboard has a paper/foam wrapper on the bottom to seal off the innards. If you were to peel off this layer, you would see bits of metal running side to side on the top and bottom, which we call **rails**. You would also see pieces of metal running up and down in each **row**. For reference, here is a picture of the inside superimposed on the breadboard:

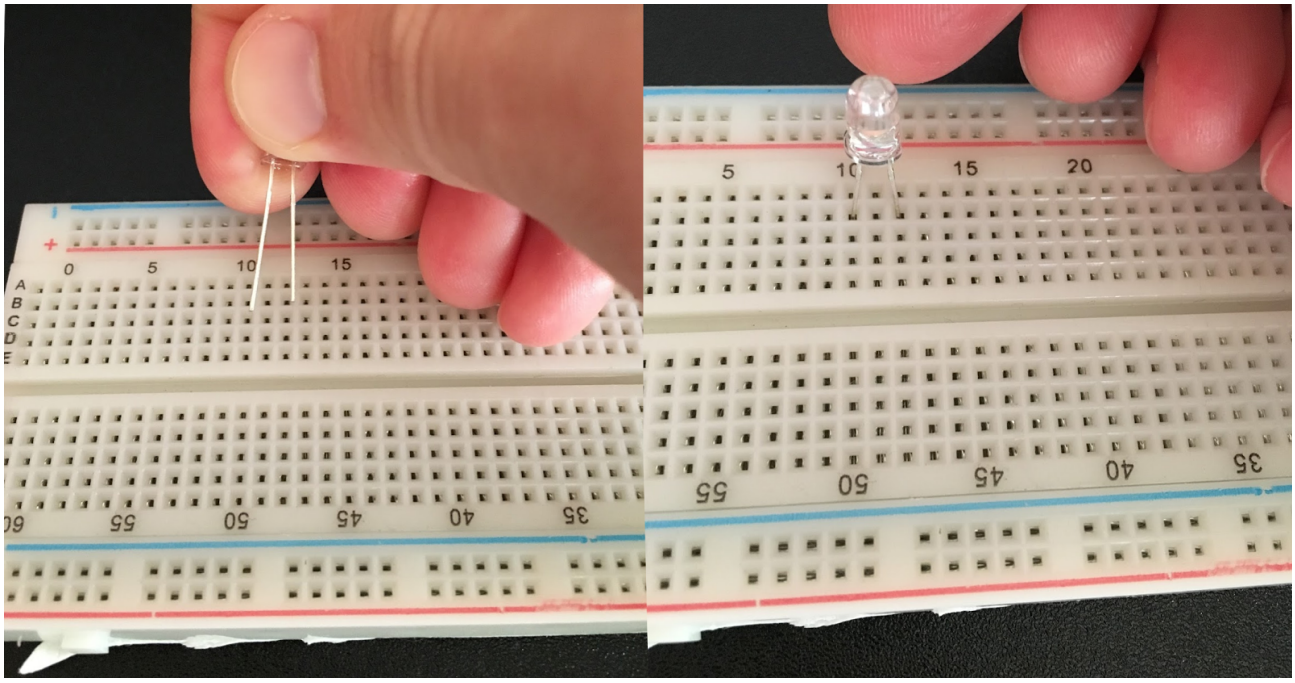


To begin creating a circuit with the breadboard, we'll use an **LED**. LEDs, or light emitting diodes, have two distinct "legs". These two legs, one shorter than the other, can be picture to the right.



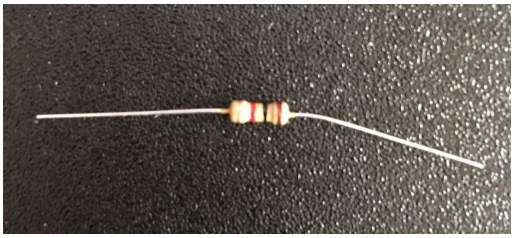
LEDs are unidirectional, meaning they only support electricity traveling in one direction. This direction is denoted by the leg length. The electricity source, or positive (+) direction, must be connected to the longer leg (called the *anode*). Likewise, the ground connection, or negative (-) direction, must be connected to the shorter of the two legs (known as the *cathode*).

Your **LED should be inserted into the breadboard** with each leg going into it's own row. Once the legs are aligned with the holes, you'll have to apply a bit of pressure to get the LED fully situated in the breadboard.

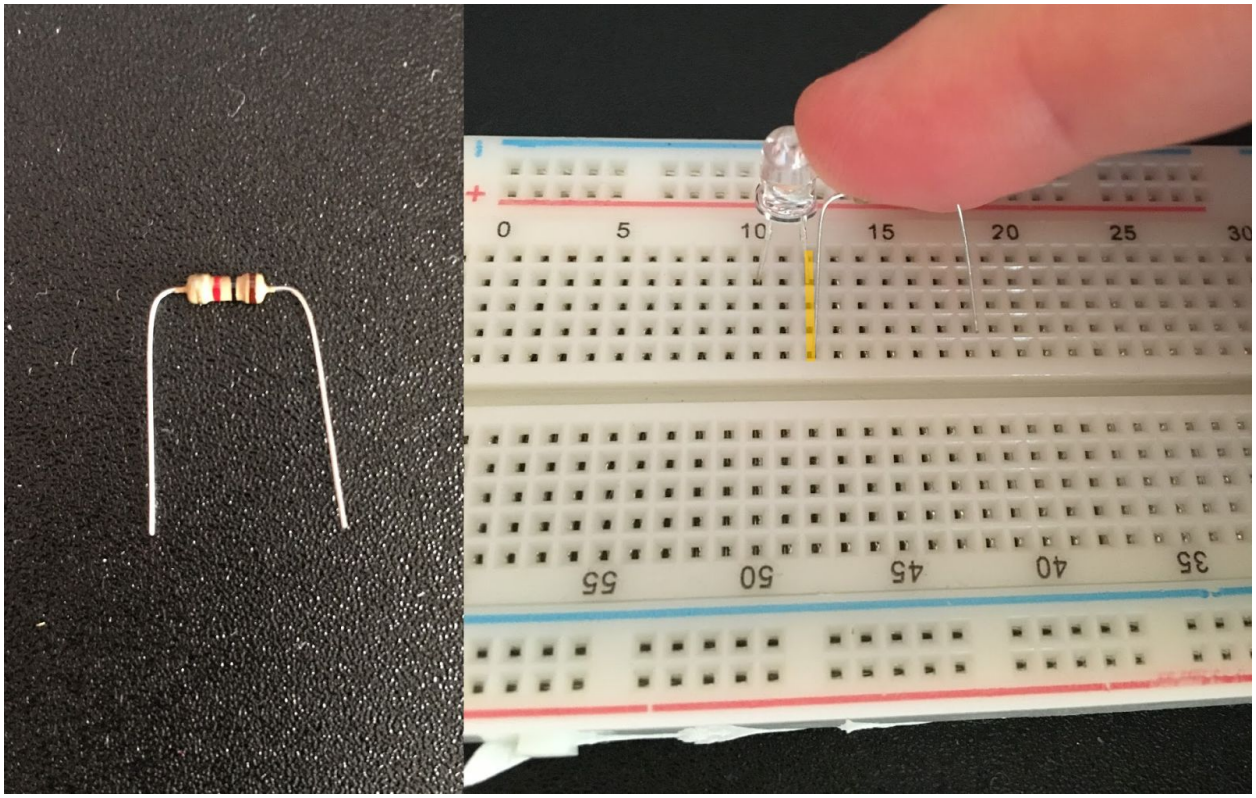


As you can see, we inserted the LED with the **short side to the right**.

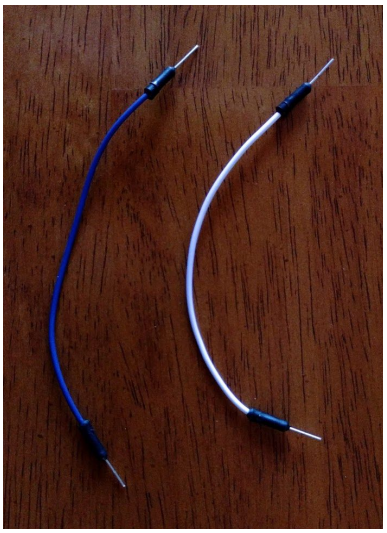
Now, before we can power the LED, we need to reduce the voltage of our circuit. These simple LEDs are very delicate and can only take a certain voltage. To prevent the LED from burning out, we use a **resistor to "take away" some of the voltage**. The resistor accomplishes this by converting some of the voltage to heat. This is what a resistor looks like:



Unlike the LED, our resistor is omnidirectional. This means we don't have to worry about the orientation of our resistor in the breadboard. Before you can insert the resistor into the breadboard, you'll first want to gently bend each leg downward. Then **insert one side of the resistor into the same row as the short side of you**r LED. The other leg may go in a few rows to the right. The highlighted row below is shared between the short side of the LED and one side of the resistor.

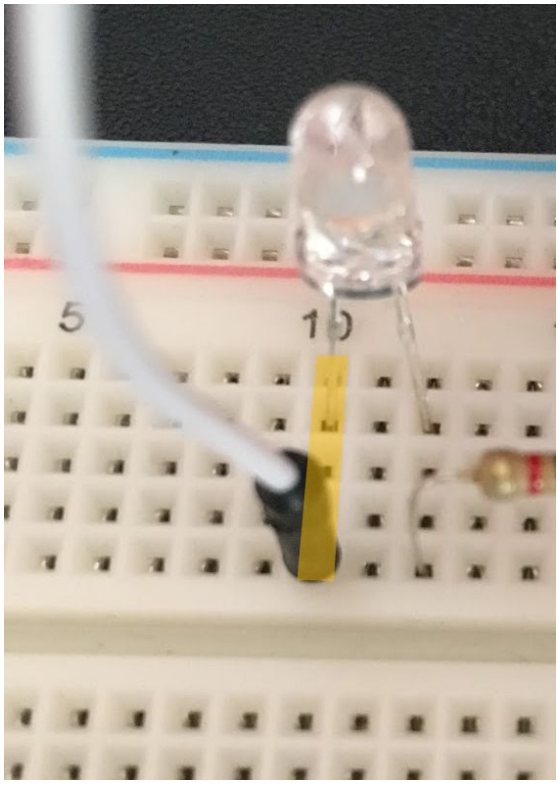


Now that you have some basic components connected to the breadboard, you can complete the circuit by adding a power source and ground connection. We'll use our CodeBug to add these connections.

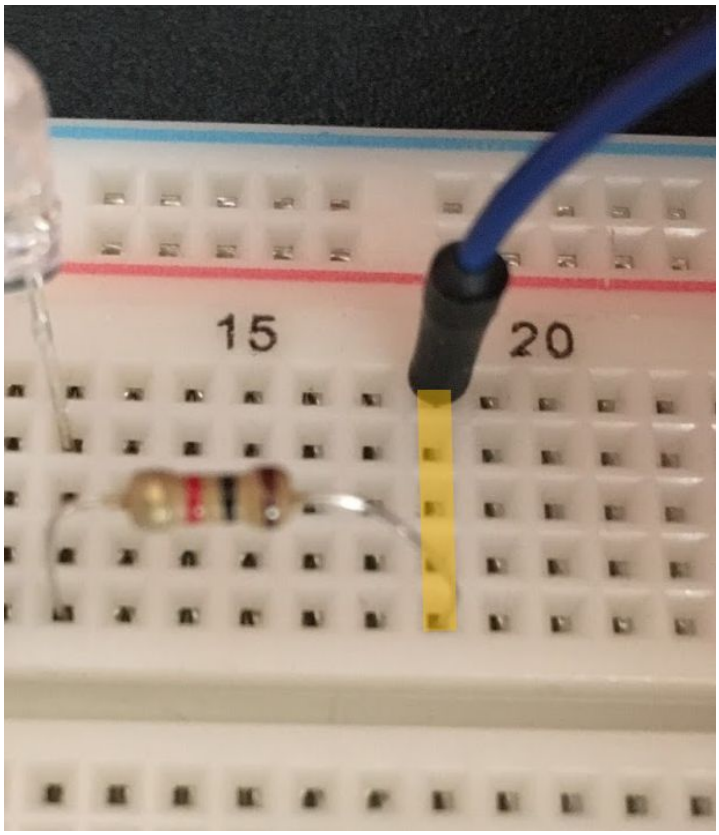


We use **jumper cables** to connect these different parts of our circuit. These are insulated wires with enough metal exposed to insert into our breadboard.

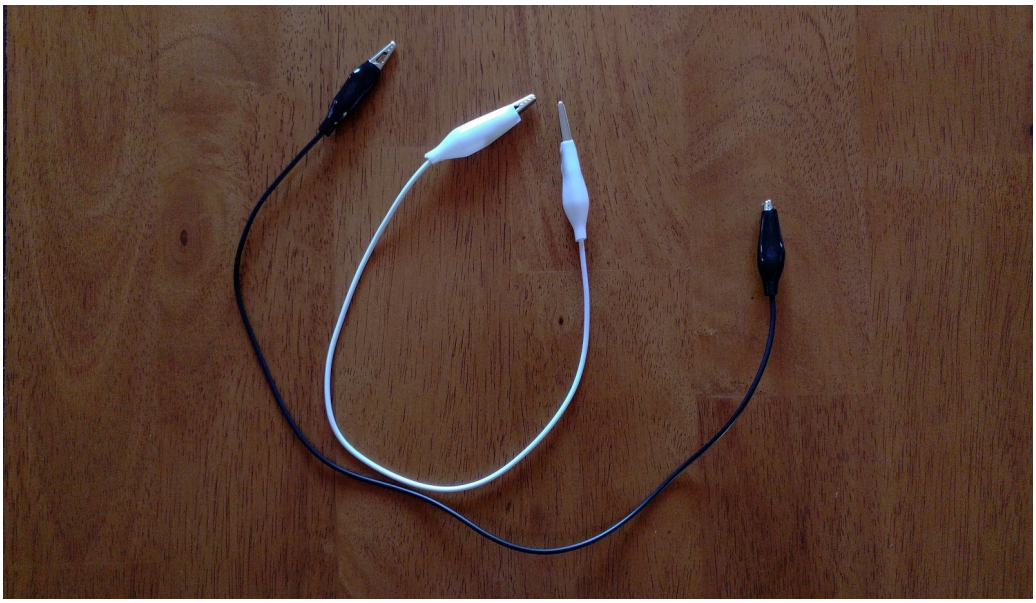
You'll want to **insert one jumper into the same row as the long side of the LED**. You can see this highlighted in the picture to the right.



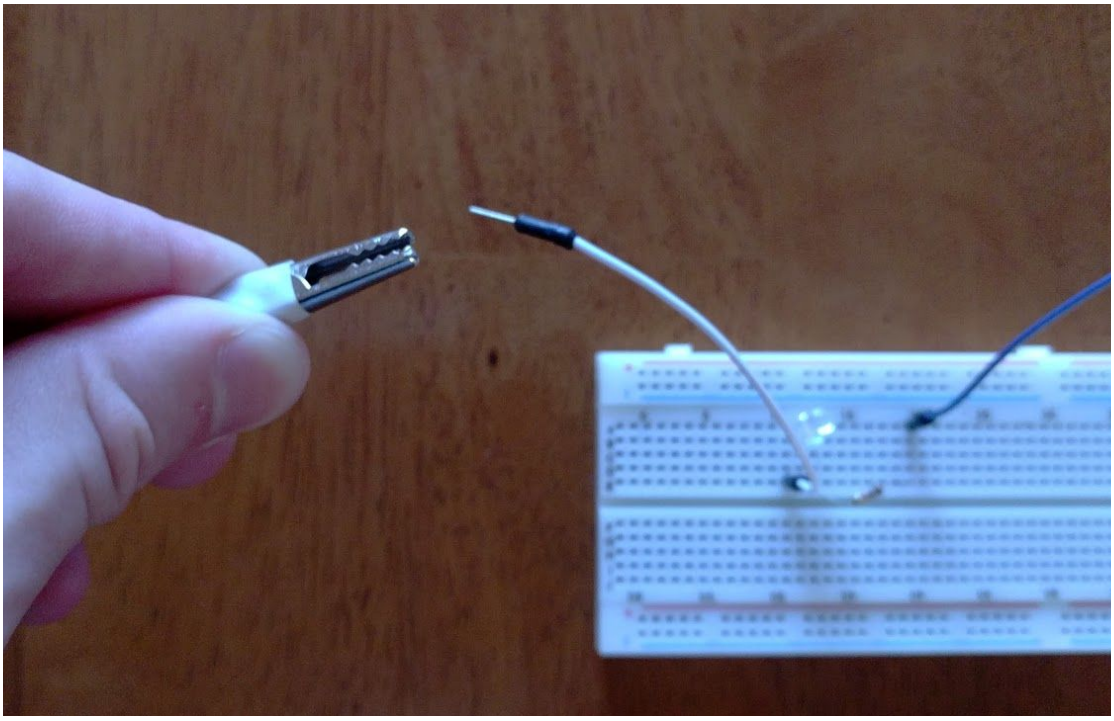
Get another jumper cable and put it in the same row as your resistor. You'll want it to be connected to the leg that is not connected to the LED.



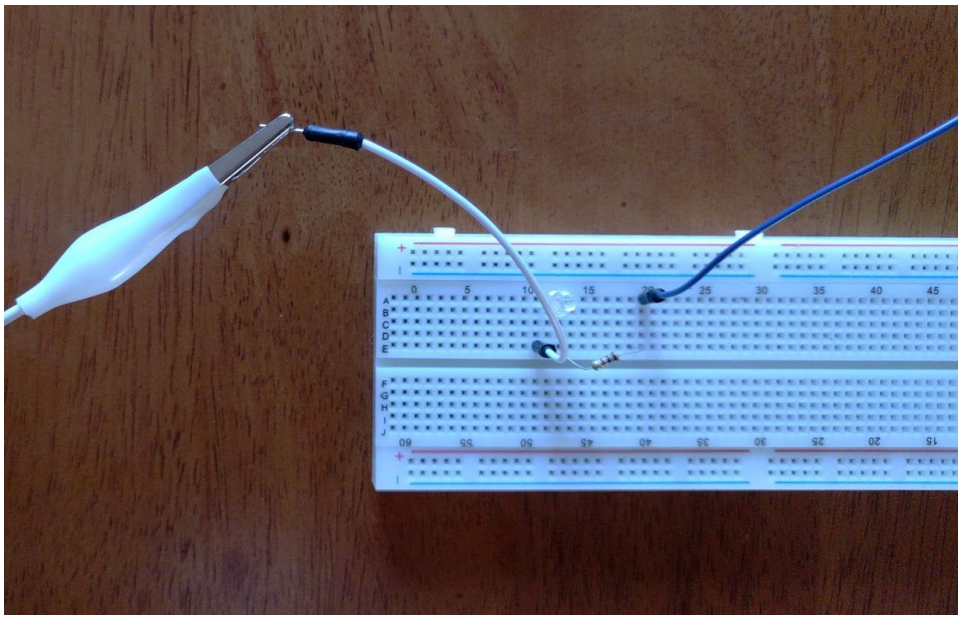
Now that you have jumpers connected to your circuit, you'll need to connect those jumpers to your CodeBug. **We'll use alligator clips to achieve this**.



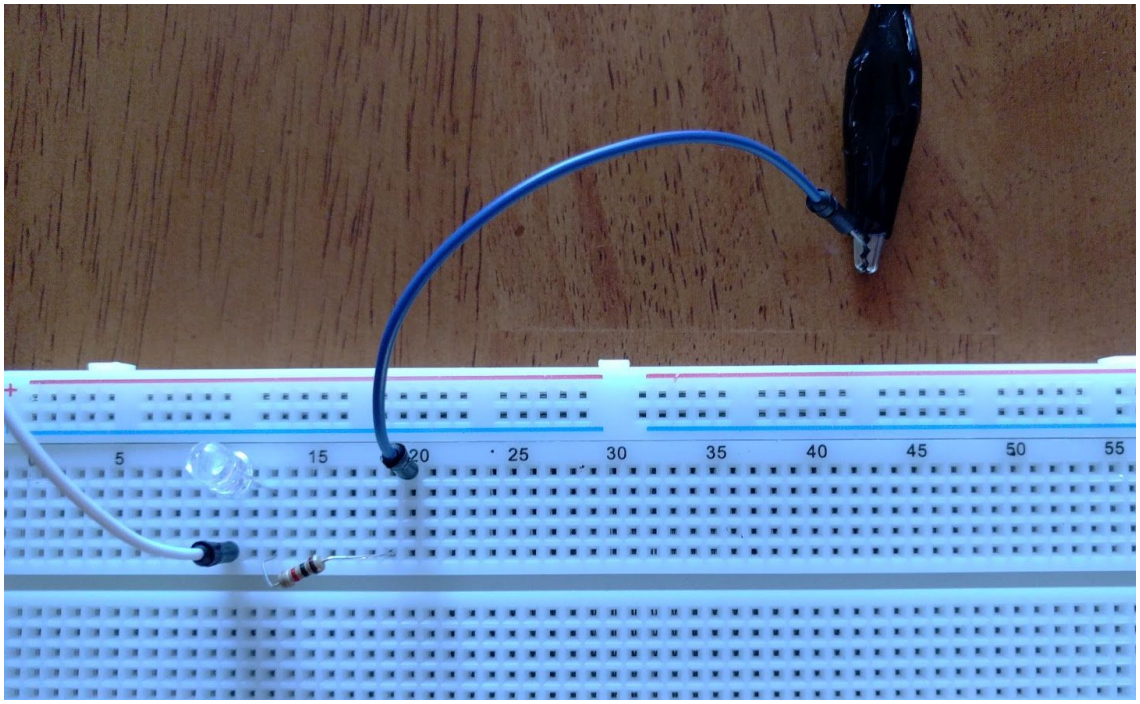
Start with either alligator clip and begin **squeezing below the tip of the alligator clip** to open the teeth.



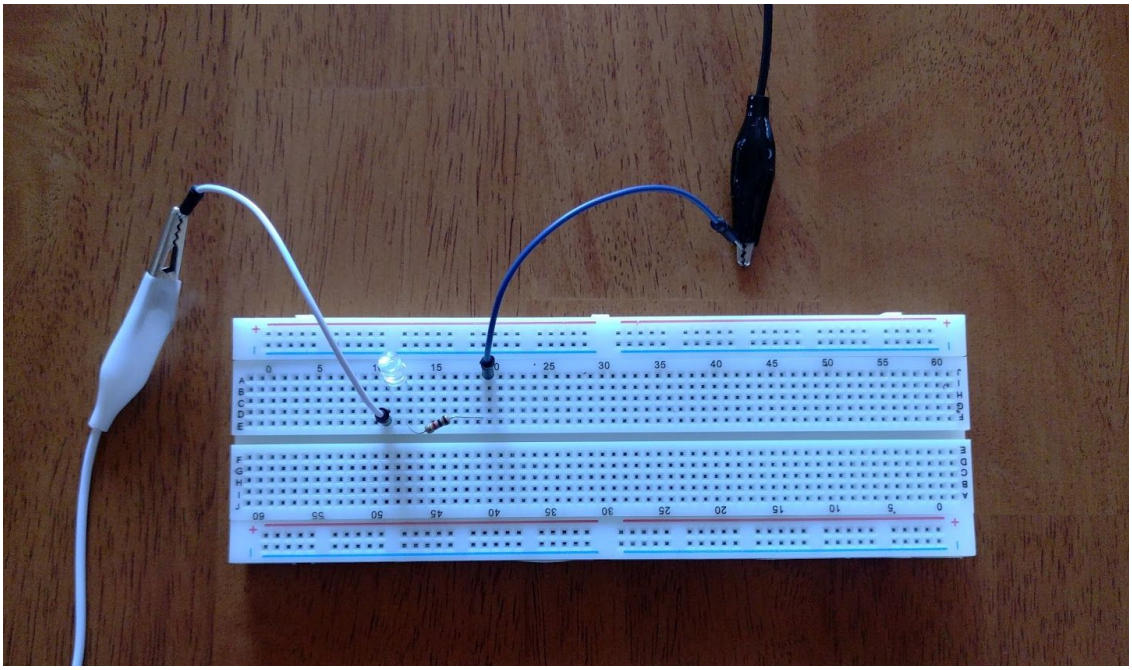
Then **insert the exposed jumper wire into the alligator clip**, and close the teeth by releasing the alligator clip.



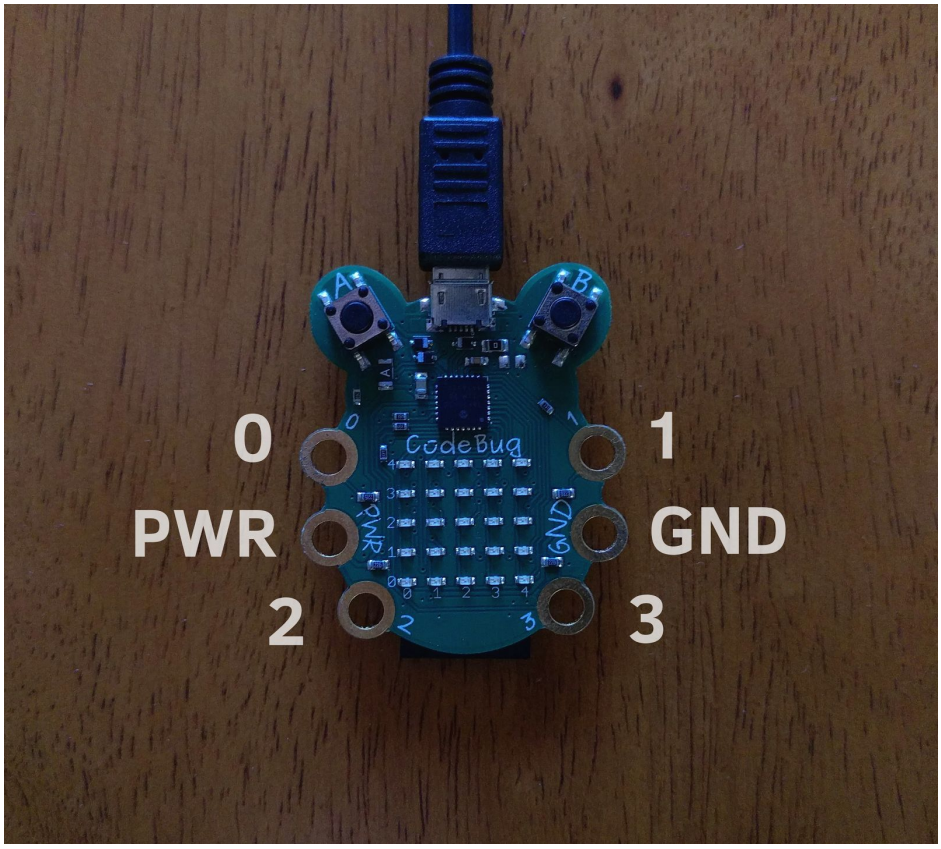
Now, do the same with the other alligator clip and jumper. Open the teeth and **insert the other jumper into your other alligator clip**.



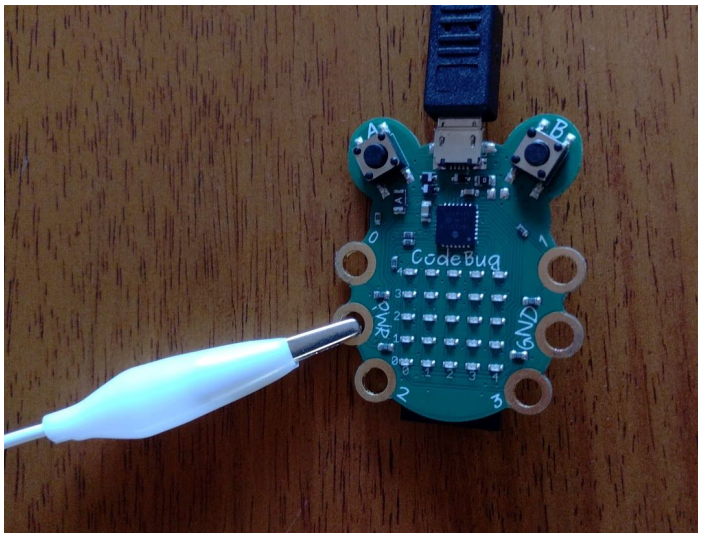
Your circuit should now look something like the following.



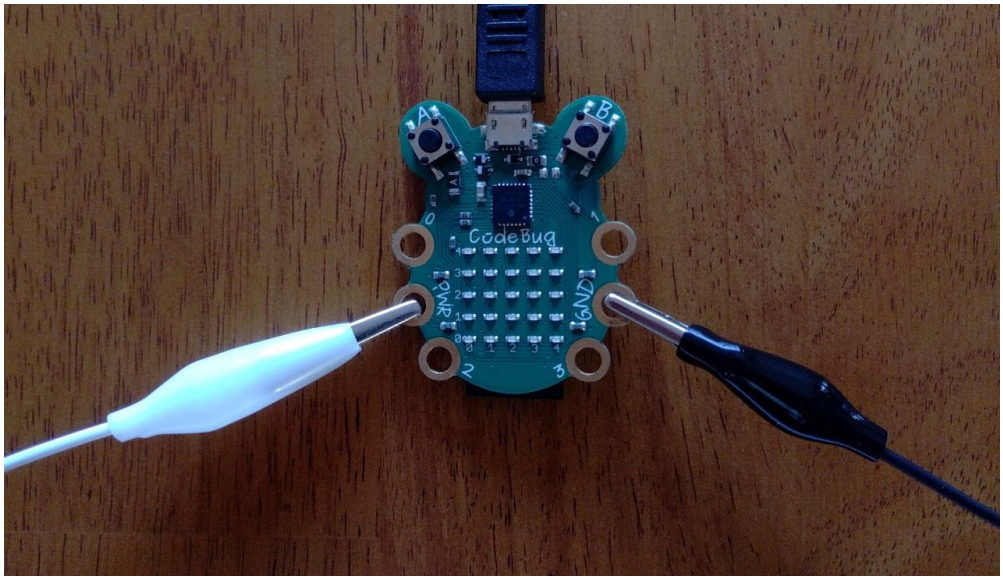
We can now connect the alligator clips to the appropriate legs on our CodeBug. **There are 6 legs on each CodeBug.** 4 input/outputs (these are programmable), 1 power, and 1 ground.



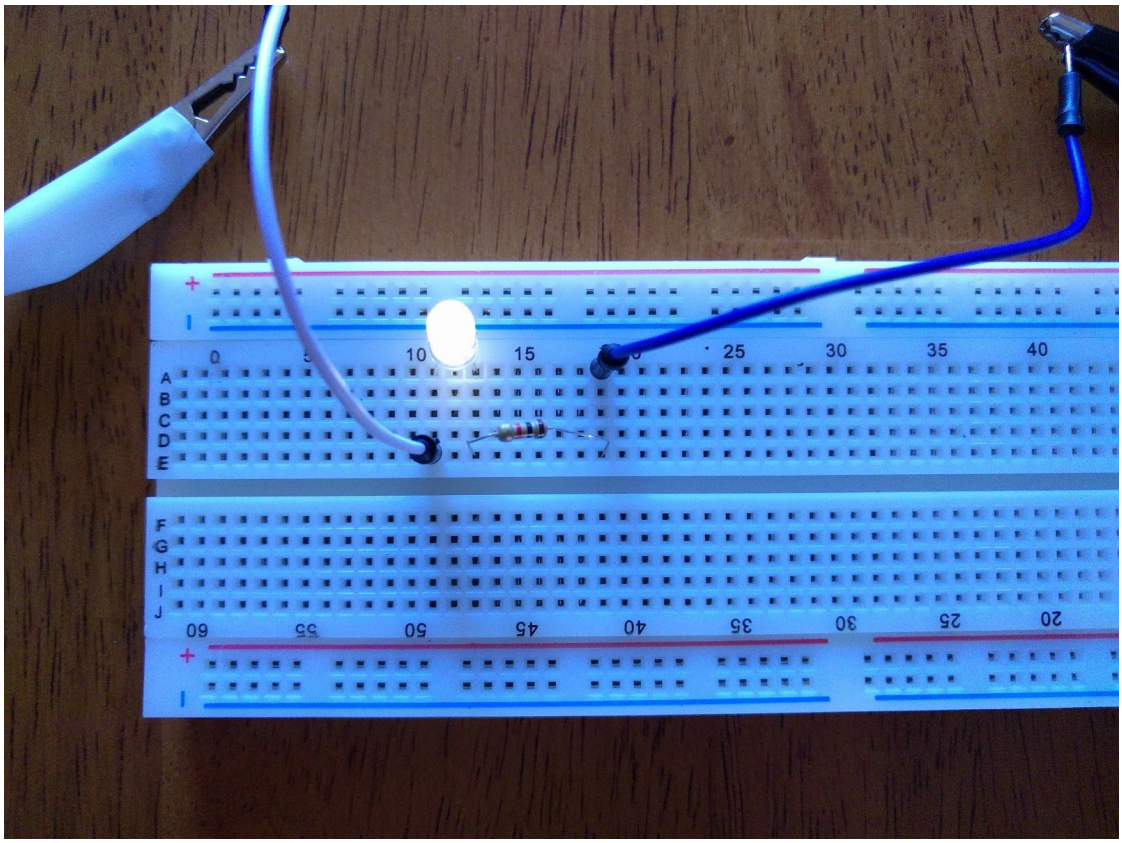
First, **take the alligator clip connected to the anode side of your LED and connect it to the PWR (power) leg of your CodeBug**. If you've followed along, this will be the jumper on the left side of your breadboard.

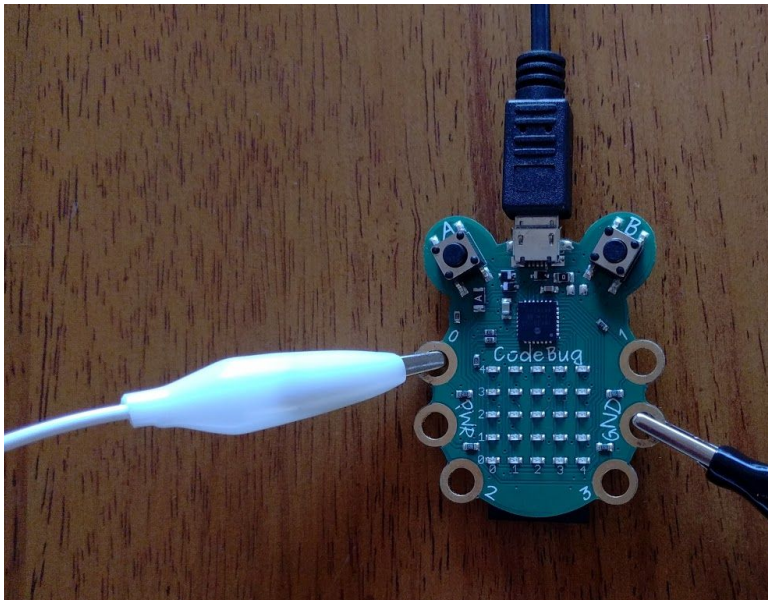


Then, **connect the other alligator clip to the GND (ground) leg of your CodeBug.**



If done correctly, you should now see your LED lighting up.

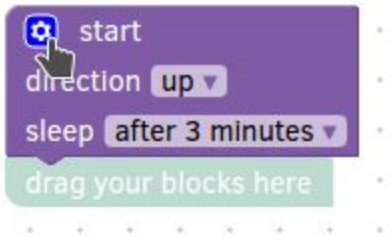




Now that you've confirmed the circuit works, **move your alligator clip from leg "PWR" to leg "0"**. This will allow us to programmatically blink the LED using the CodeBug.

Open the CodeBug website and click on the "Create" tab to access the programming interface.

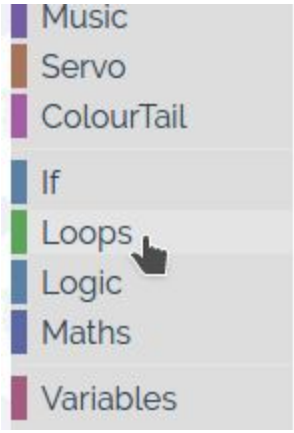
Begin by clicking the blue cog wheel in the upper left corner of the purple start block.

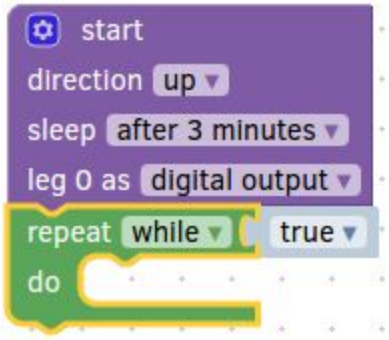


Click to checkmark the "leg 0 input/output" box.

Click the blue cog wheel once more to close the popup.

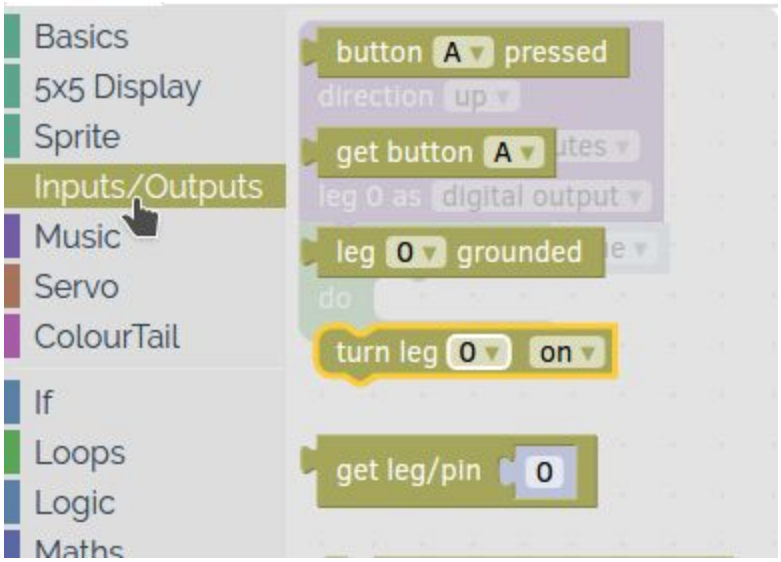
Now, **open the "Loops" Block Menu** by selecting it from the list of menus.

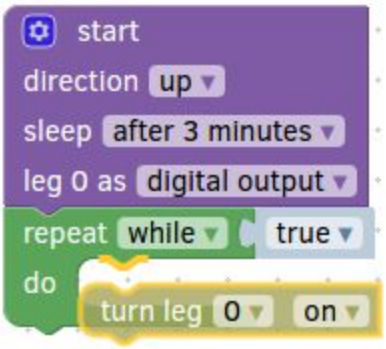




Click and drag the "repeat while true" block, snapping it to your purple start block.

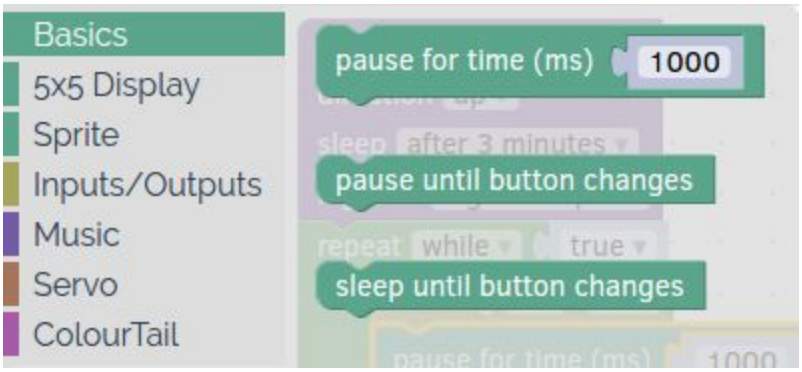
Next, **open the "Input/Outputs" Block Menu.**

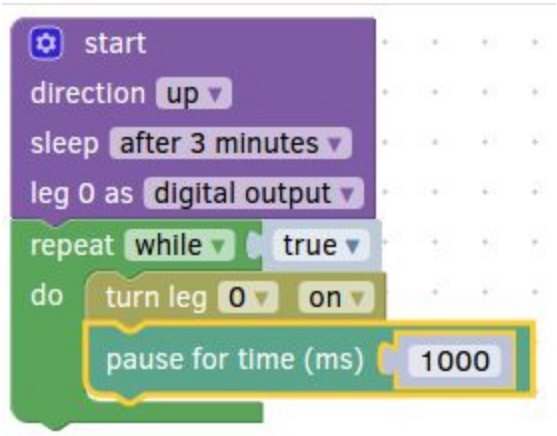




You'll want to **grab the "turn leg 0 on" block and snap it inside your "repeat while true" block.** You'll see an orange wedge appear inside your green "repeat while true" block just before you should release the block.

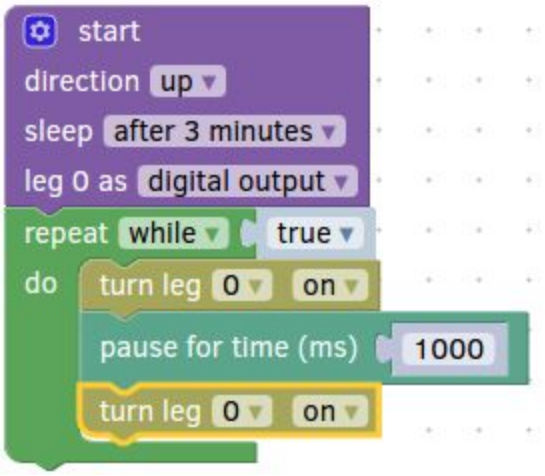
Next, **click the "Basics" Block Menu**.

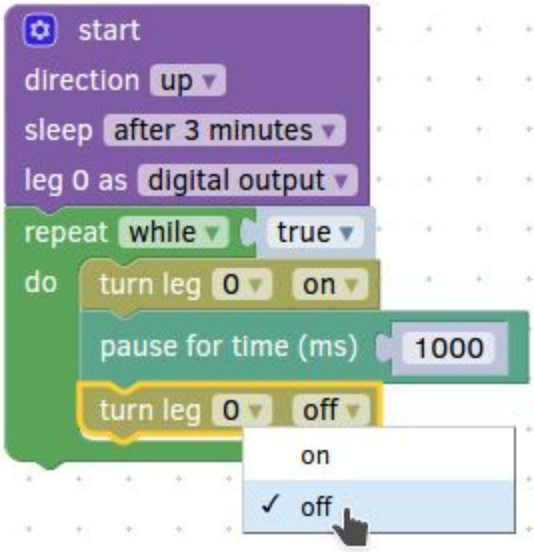




Snap the "pause for time (ms) 1000" block inside your "repeat while true". Place it directly underneath the "turn leg 0 on", while still inside the green "repeat while true".

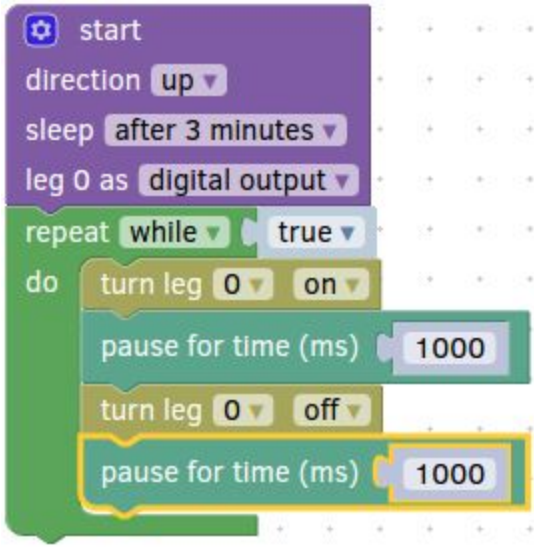
Now, **go back under "Inputs/Outputs" and snap another "turn leg 0 on" into your "repeat while true".** Again, place it underneath the previously placed block.





This time, **click the "on" box inside the "turn leg 0 on" block you just placed and select "off".**

Finally, **open the "Basics" Block menu and snap a new "pause for time (ms) 1000" block into your "repeat while true" block.**



You've just completed the code for your blinking LED program. You can now **load the code onto your CodeBug** by following the steps on page 9.

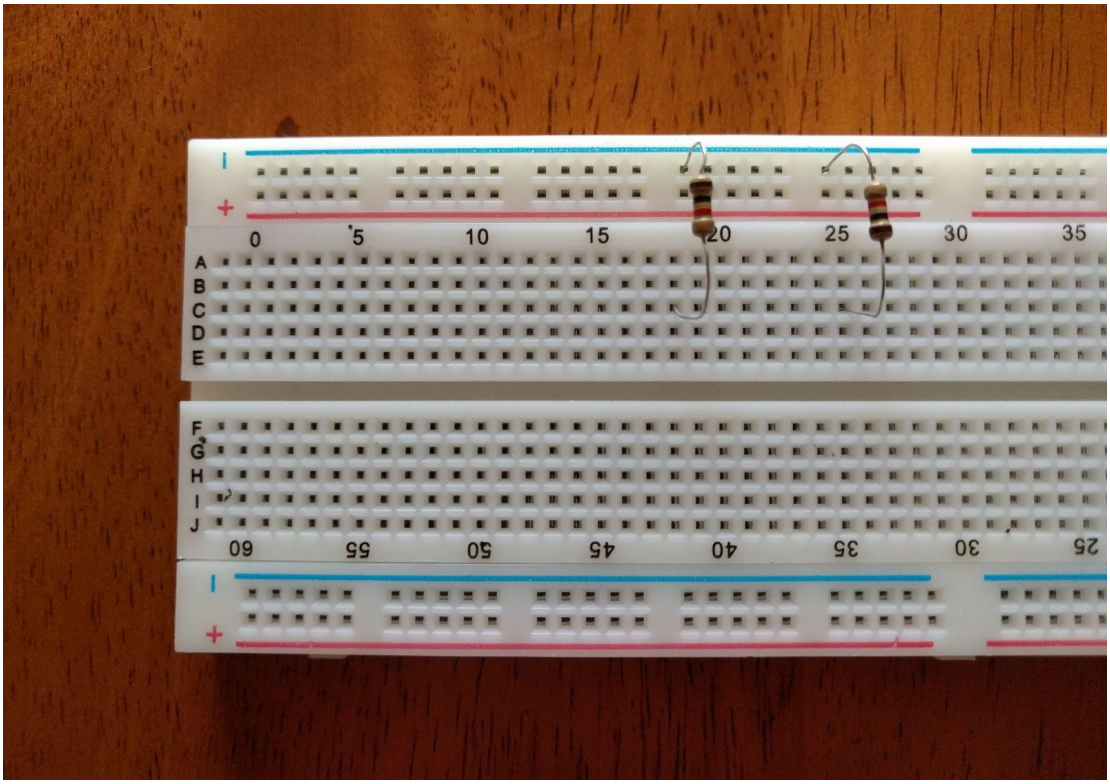
If everything was successful, you will **see your LED blinking**. It will stay on and turn off for 1 second each. You can change the values of your "pause for time" blocks to alter these timings. Remember to re-download the code and load your CodeBug again after making any changes.

Multiple LEDs

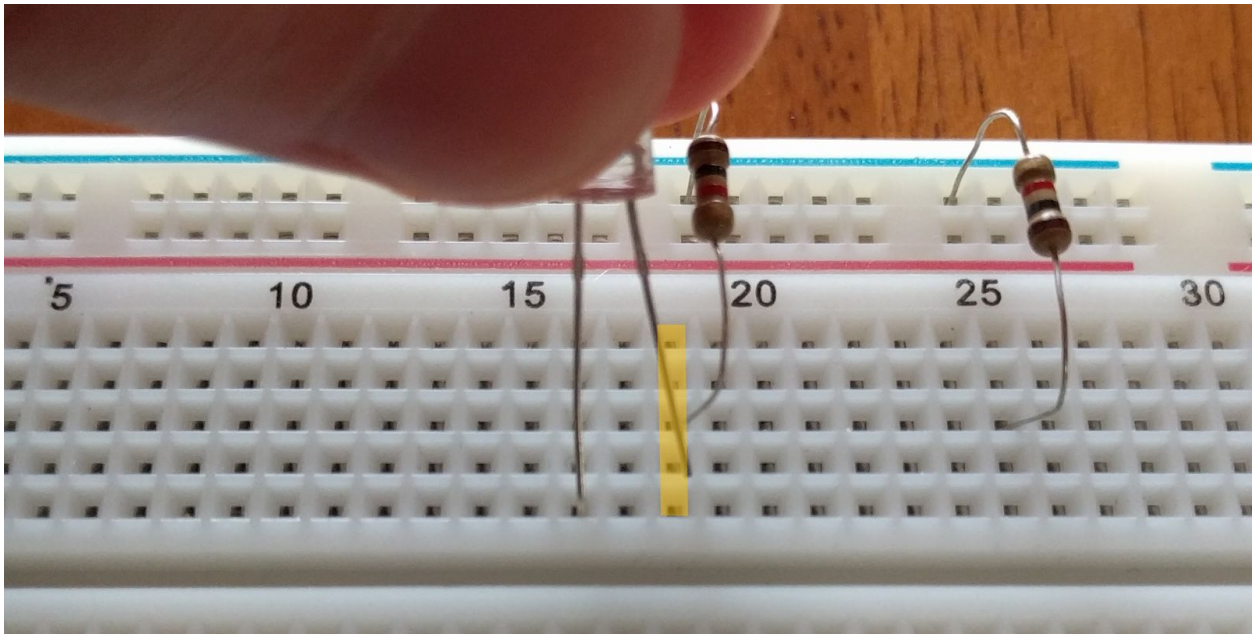
You can power and control multiple LEDs using the CodeBug. Here I'll demonstrate running two, but the process is the same for up to four.

Start by **removing the resistors, LEDs, and jumpers from your breadboard** so that nothing is plugged in.

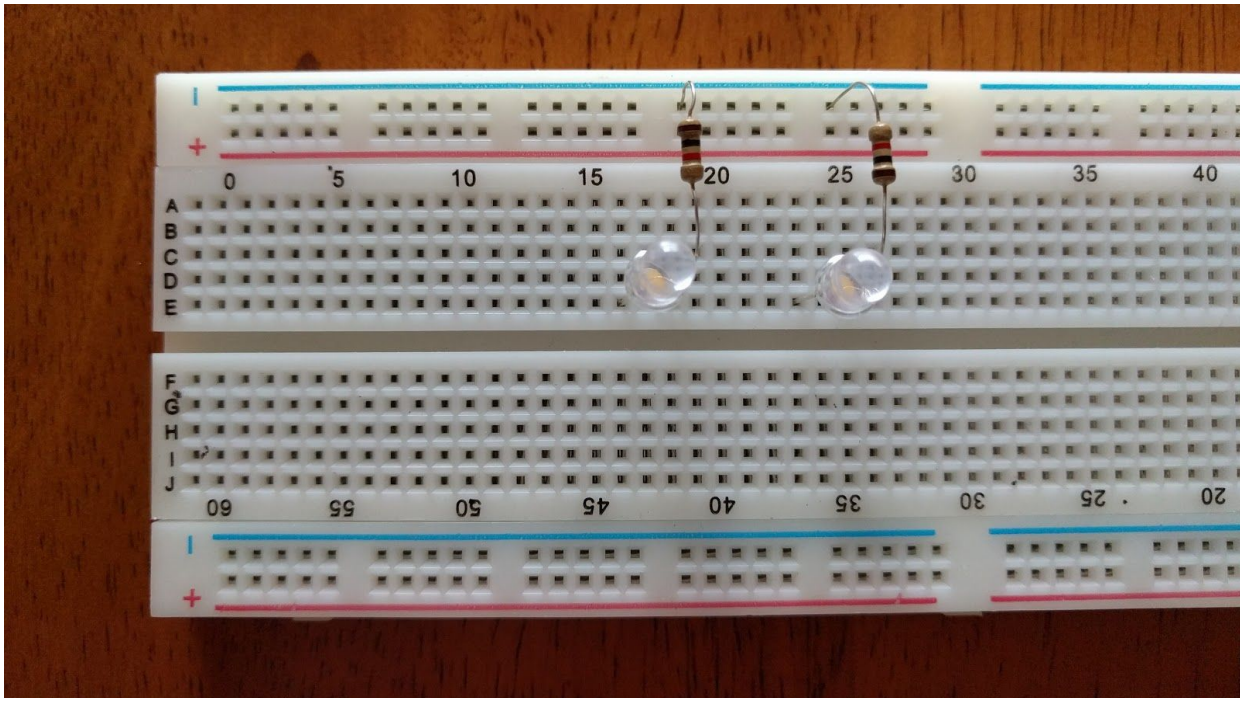
Then, **insert two resistors into your breadboard**. This time, plug one leg into the blue "rail" and the other into the row beneath it. It should look like the following.



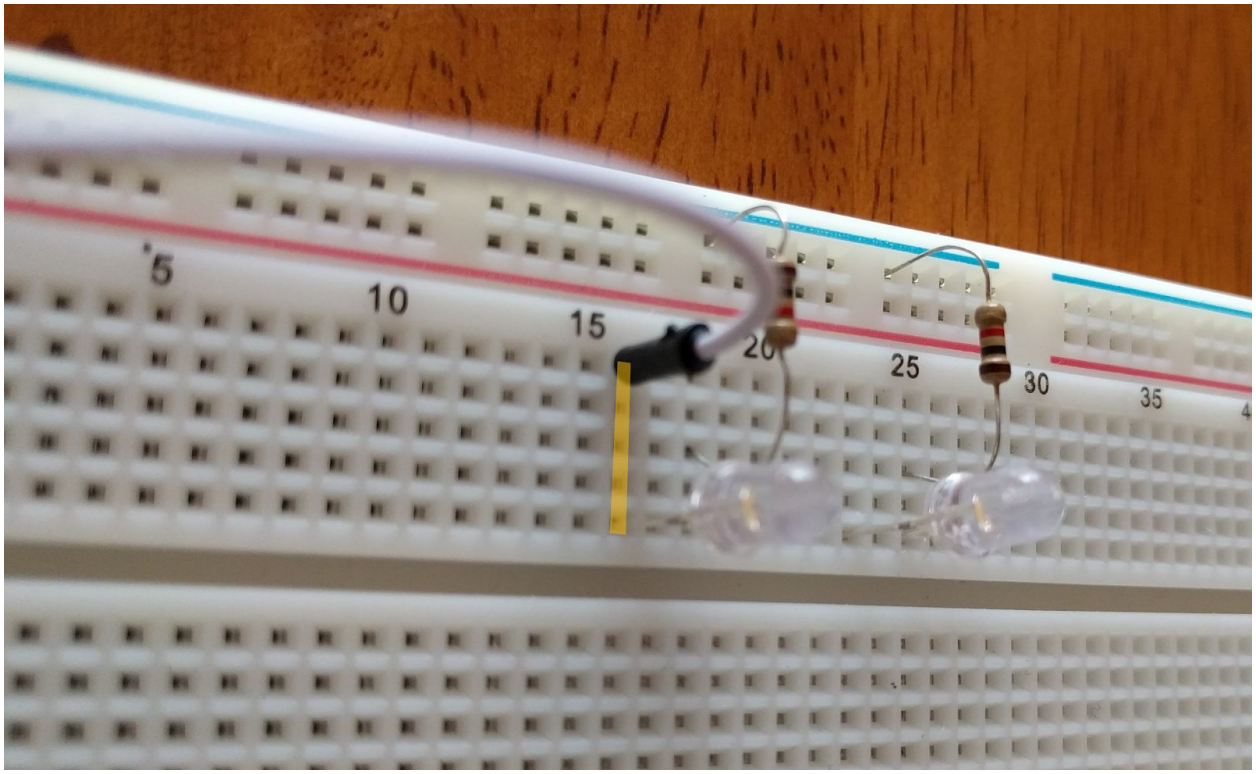
Now, **insert your first LED with the short side in the same row as a resistor, and the long side a couple rows to the left.**



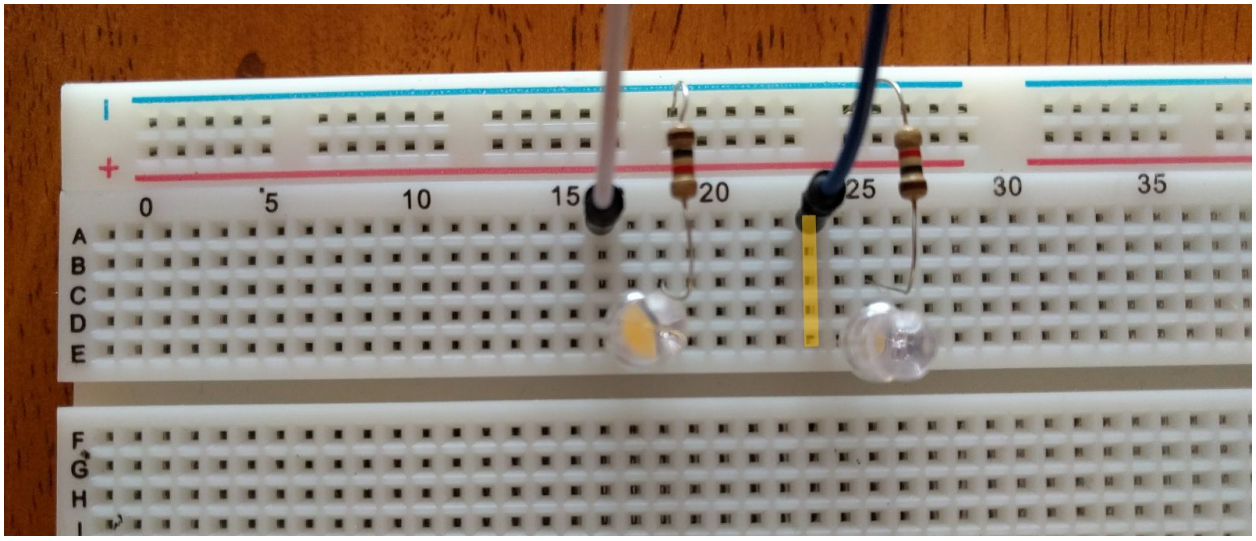
Insert your second LED the same way, with the short leg in the same row as the resistor. When finished, your board should look like this:



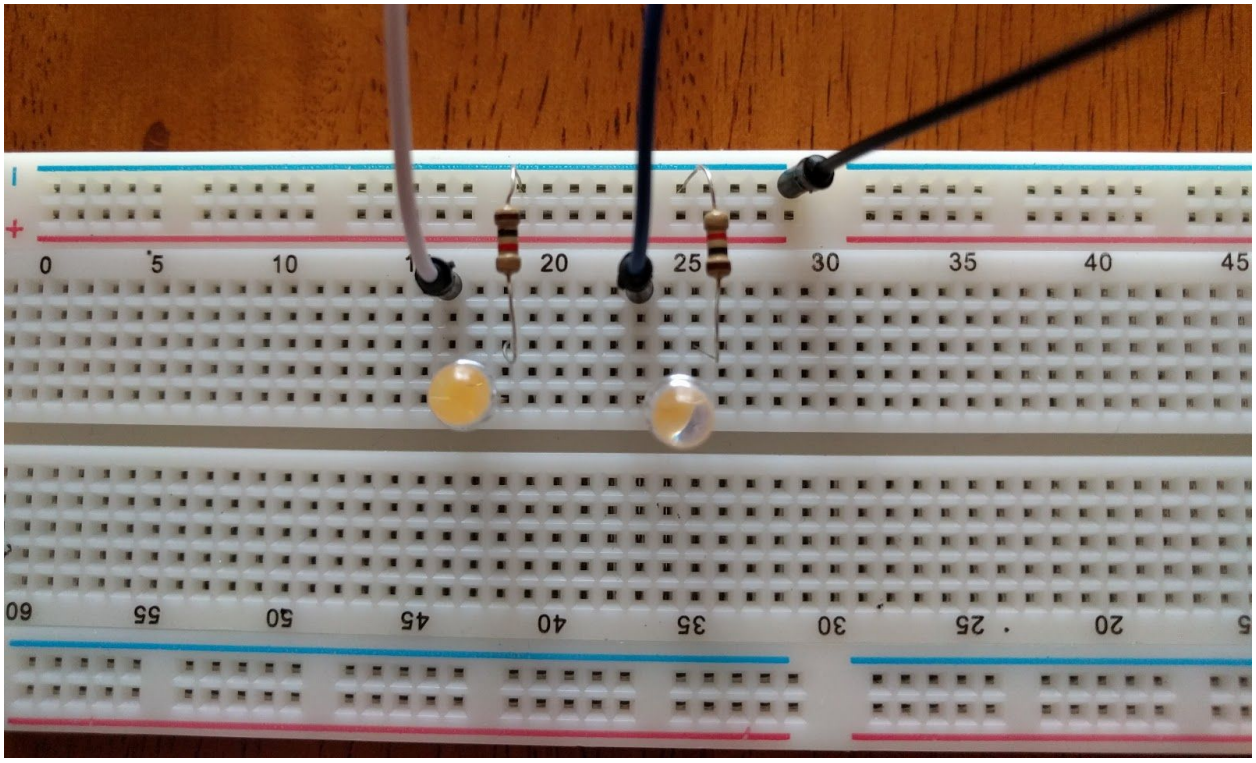
Next, **plug a jumper into the same row as the anode (long side) of one of your LEDs.** In the following picture, you can see the row shared by the jumper and LED is highlighted.



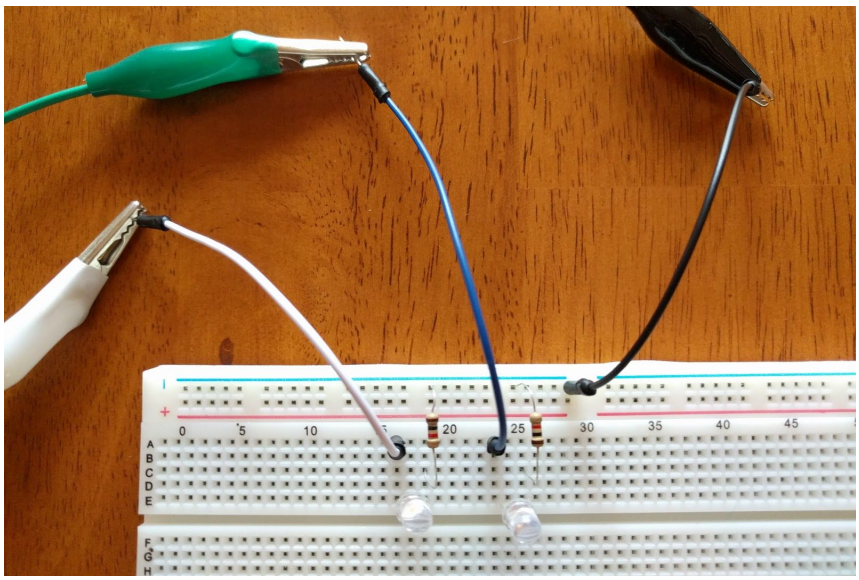
Do the same with your other jumper and LED, inserting it into the same row as the anode of the second LED. Again, the shared row is highlighted below.



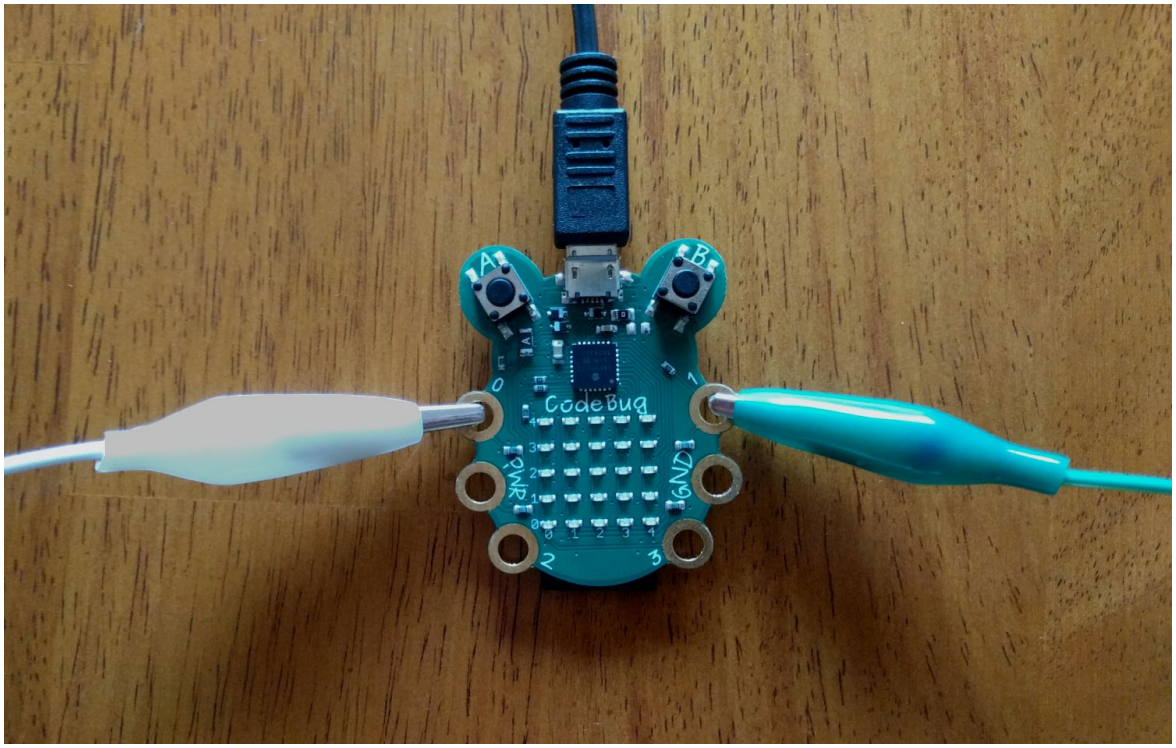
We still need to ground this circuit. Instead of grounding each resistor/LED, we can use the blue “rail” to ground all the components in our circuit. Do this by **inserting another jumper into the top-most blue “rail” on your breadboard.** This should be the same rail where your resistors are plugged in.



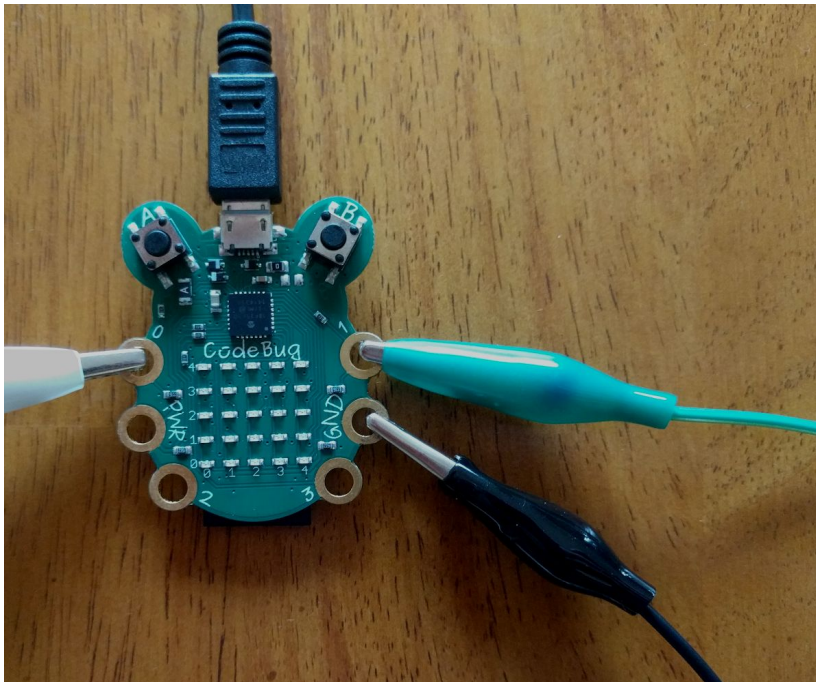
Now that your jumpers are connected, we'll once again use alligator clips to connect the CodeBug. **Connect the end of each jumper to an alligator clip** as pictured to the right.

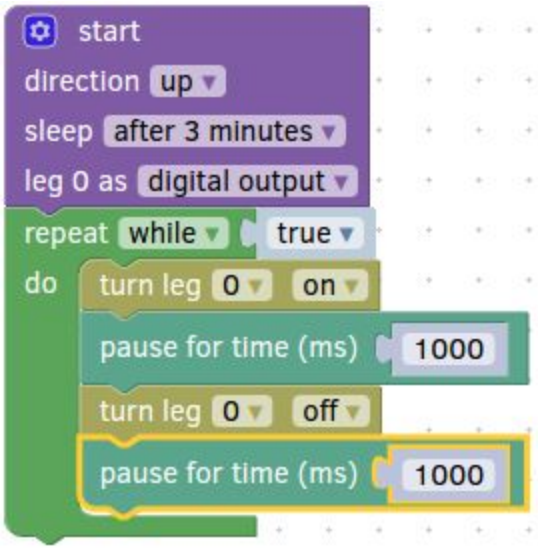


Connect the alligator clips coming from your LEDs to legs “0” and “1” on your CodeBug respectively.



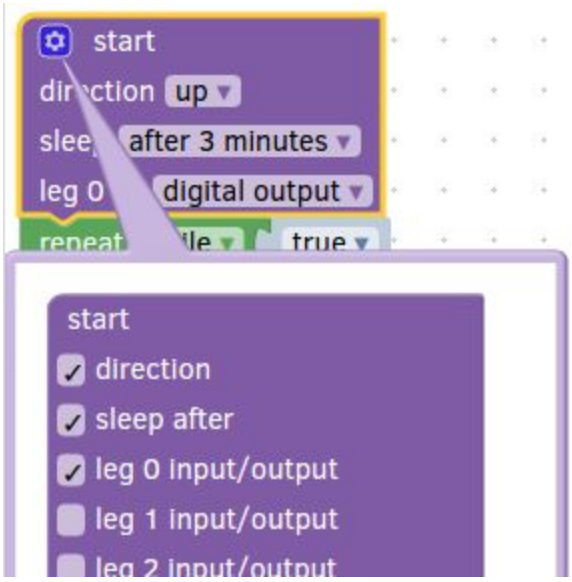
Then, **plug the alligator clip connected to the blue “rail” on your breadboard into leg “GND” on the CodeBug**. In our case it would be the black clip as pictured to the right.

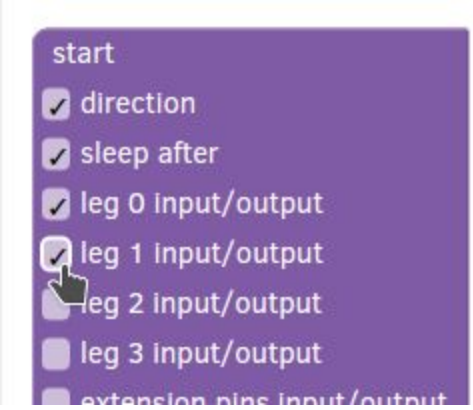




Your CodeBug and breadboard are now setup for powering two LEDs. **Head back to your code on the CodeBug website**. This is how we left our code.

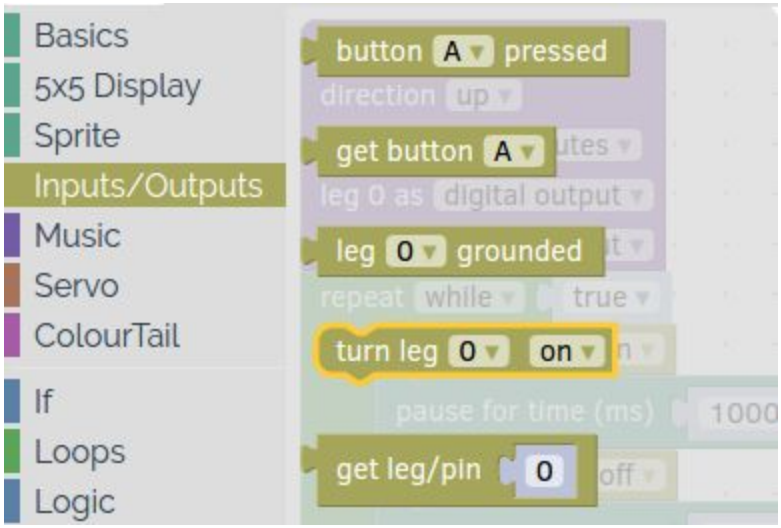
Powering a second LED using the CodeBug is very similar to powering only one. We'll start by **clicking the blue cog in our purple start block**.

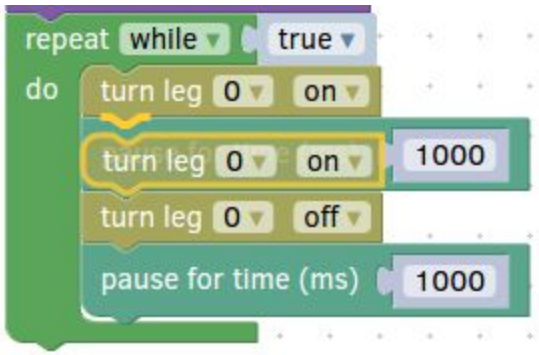




Now you'll want to **checkmark "leg 1 input/output",** then click the blue cog again to close the popup.

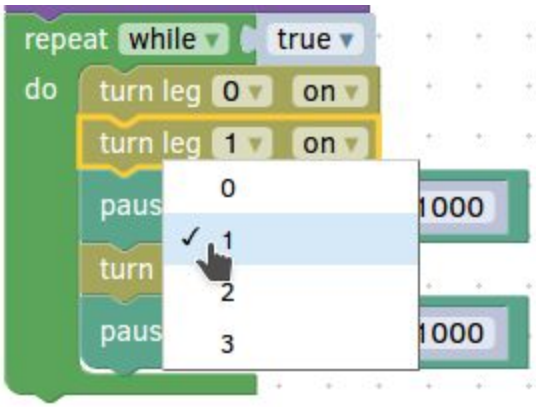
Open the "Inputs/Outputs" Block Menu and grab a "turn leg 0 on" block. Don't snap it to your existing code right away.

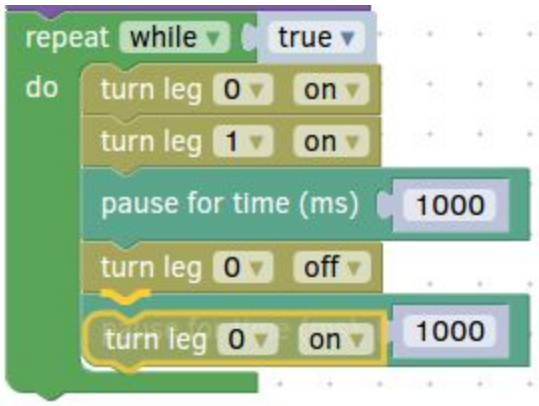




Snap the block directly under the first "turn leg 0 on" in your code. Do this by aligning the top edge of your block to the bottom edge of the existing "turn leg 0 on" block. You'll see an orange wedge appear, signalling where the block will be inserted.

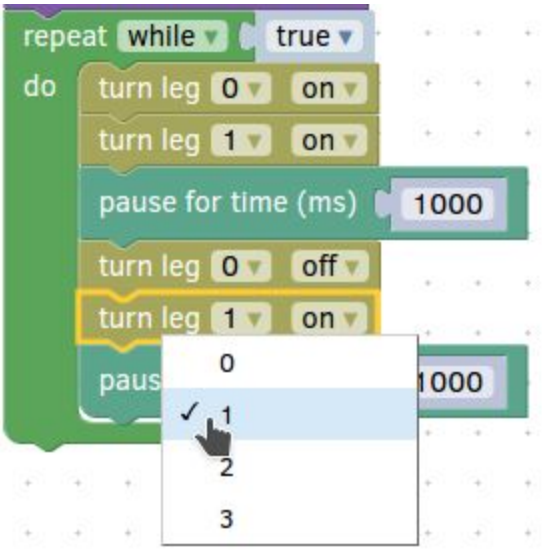
Now, **click the "0" in the block you just snapped and change it to a "1".**

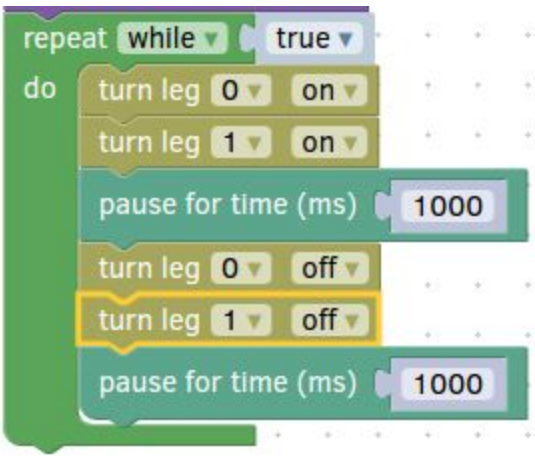




Go back under "Inputs/Outputs" and get another "turn leg 0 on" block. This time, snap it directly under the last "turn leg 0 off" in your code.

Click the "0" on the block you just inserted and change it to "1".

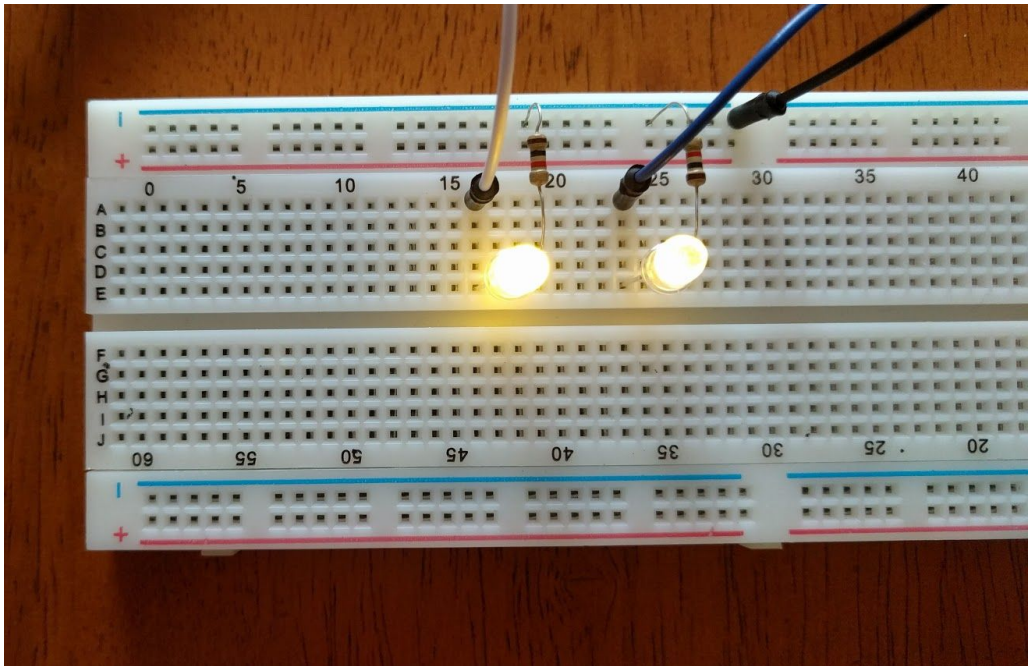




Finally, **click the "on" button of the same block and change it to "off".**

Go ahead and **load this code onto your CodeBug**. Refer back to “Loading Your Code” on page 9 for instructions on loading your CodeBug.

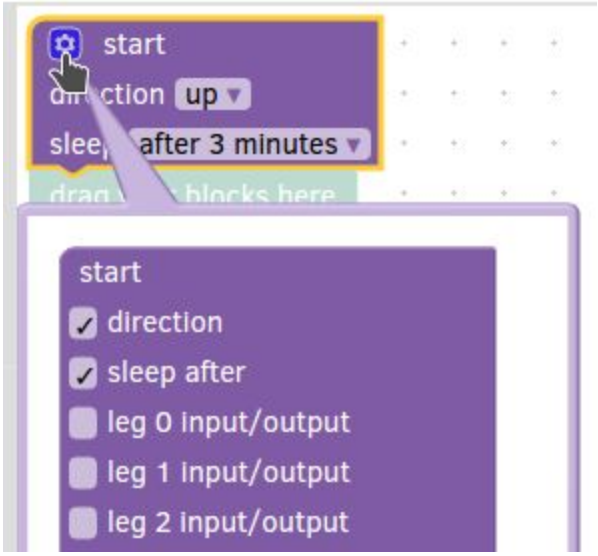
If done correctly, **you should now see both of your LEDs blinking** on and off for 1 second each. Again, you can change these timings by adjusting the “pause for time” blocks in your code.

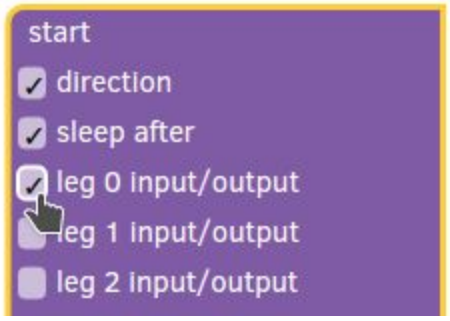


Exercise (3) Does It Conduct?

First, **open a new Workspace by clicking the "Create" button** located at the top of the CodeBug website.

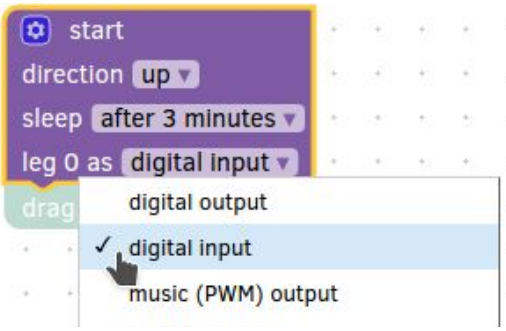
To begin creating the "Does it Conduct?" game, **click the blue cog** in the top left of your purple start block. You should then see a pop-up with several checkboxes.



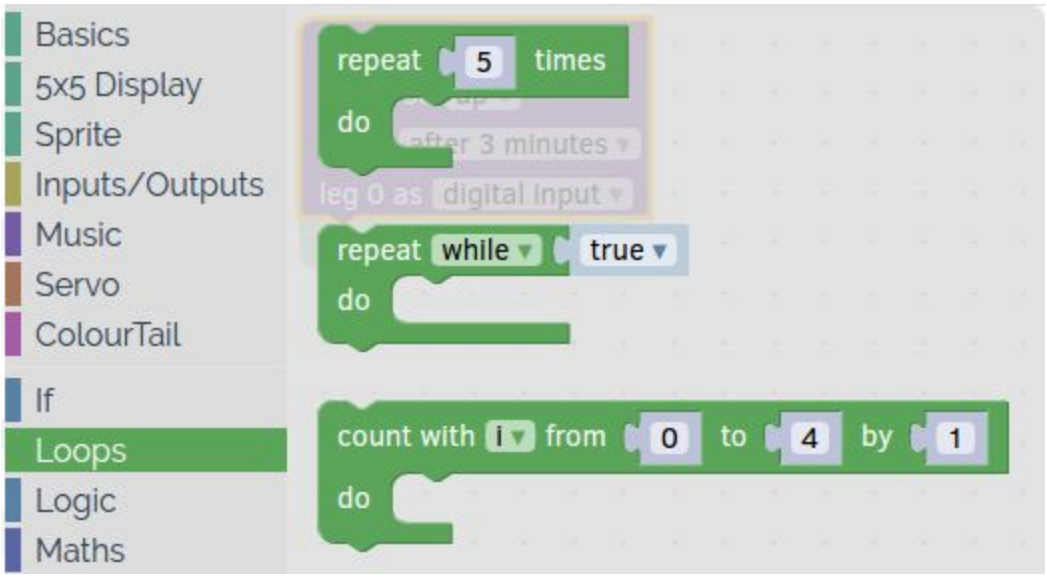


Checkmark the "leg 0 input/output" box. Then press the blue cog again to close the pop-up.

You should now see "leg 0 as digital output" appear in your start block. **Click "digital output" and change it to "digital input".**

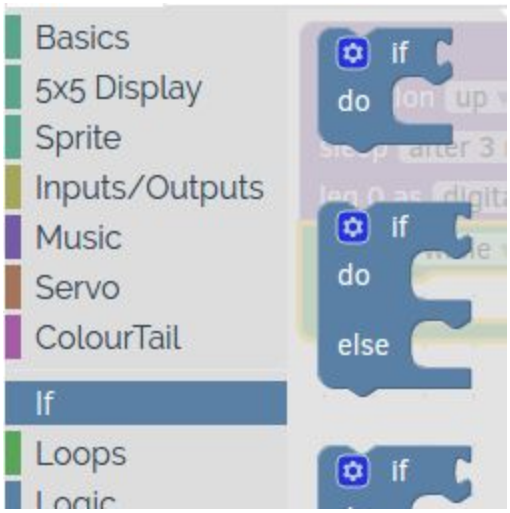


Now, **open the "Loops" Block Menu** by clicking it.

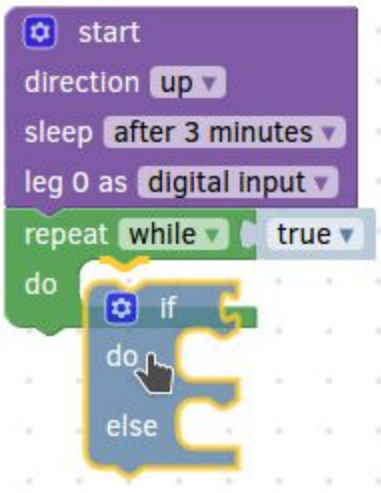


Click and drag the "repeat while true" block, snapping it underneath your start block.

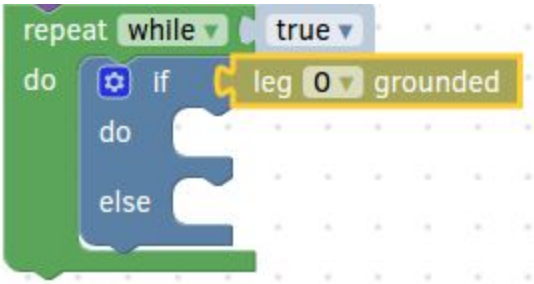
Next, **click on "If" to open it's Block menu.**

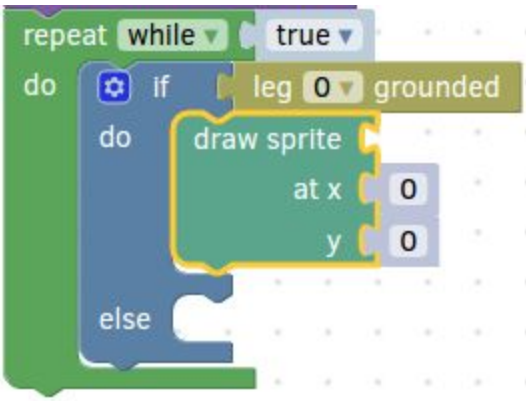


Snap the “if do else” block to the inside of your “repeat while true” block. To accomplish this, align the inside edge of the green “repeat while true” to the top edge of your “if do else” block. You should see an orange wedge inside your “repeat while true” just before releasing.



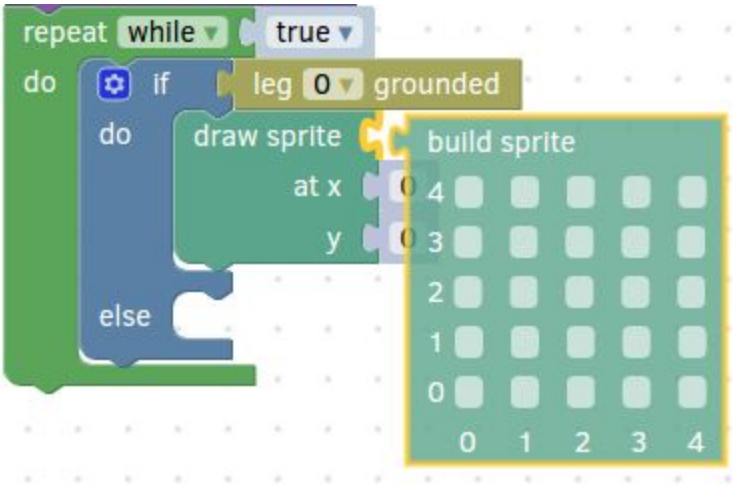
Open the “Inputs/Outputs” Block Menu, grab a “leg 0 grounded”, and snap it to your blue “if do else” block. Snap it to the “if” portion of your block.



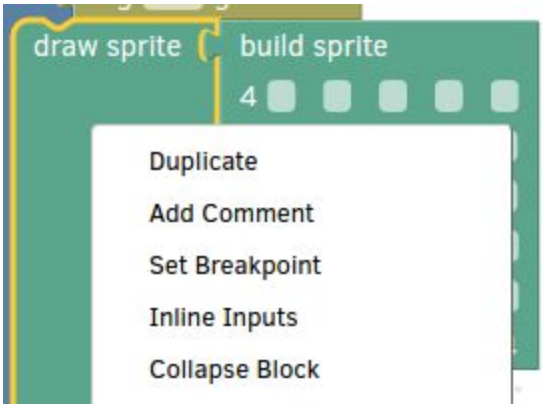


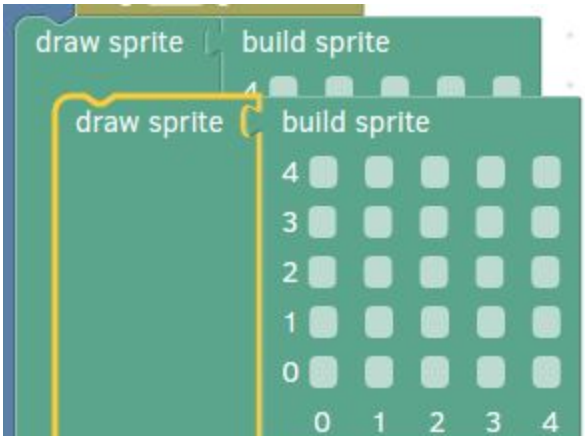
Open the “Sprite” Block Menu once again and **snap a “draw sprite” block inside your “if” block**. Place it inside the “do” portion. This process is the same as putting the “if block” in your “repeat while true”.

Now, **grab a "build sprite" block from the "Sprite" Block Menu and snap it to your newly created "draw sprite" block**. Remember to align the notches in the blocks to snap into place.



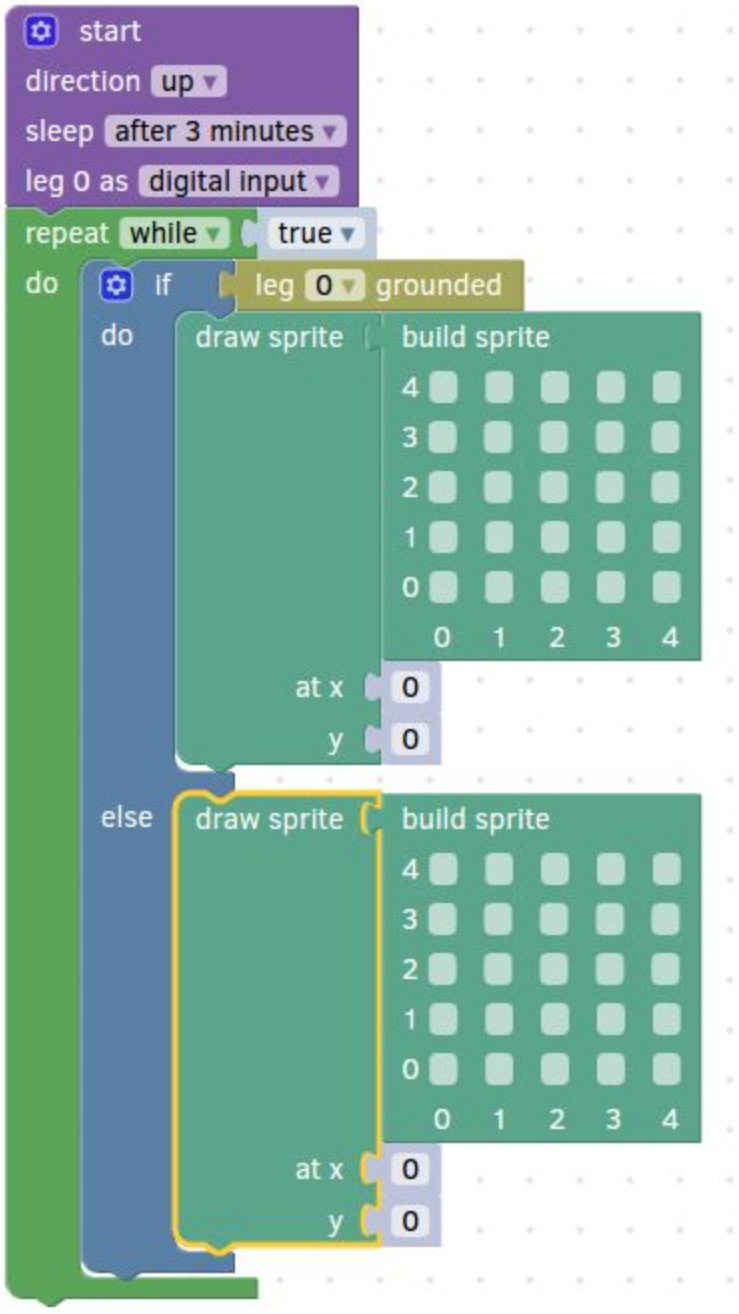
We're going to put these same two blocks in the "else" portion, but we can make this faster by duplicating. **Right click on the "draw sprite"** that is now connected to a "build sprite". You should see a context menu appear.





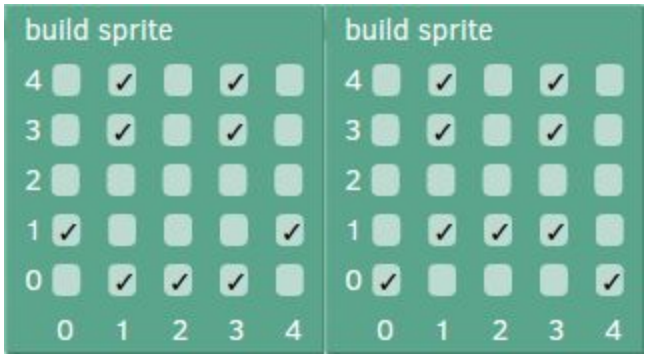
Click on "Duplicate". You should now see another "draw sprite" and "build sprite" appear.

Drag this new “draw sprite” into the “else” portion of your blue “if do else” block. Make sure to click and drag on “draw sprite”, so that it moves both the “draw sprite” and “build sprite” blocks. Your code should now look like this:



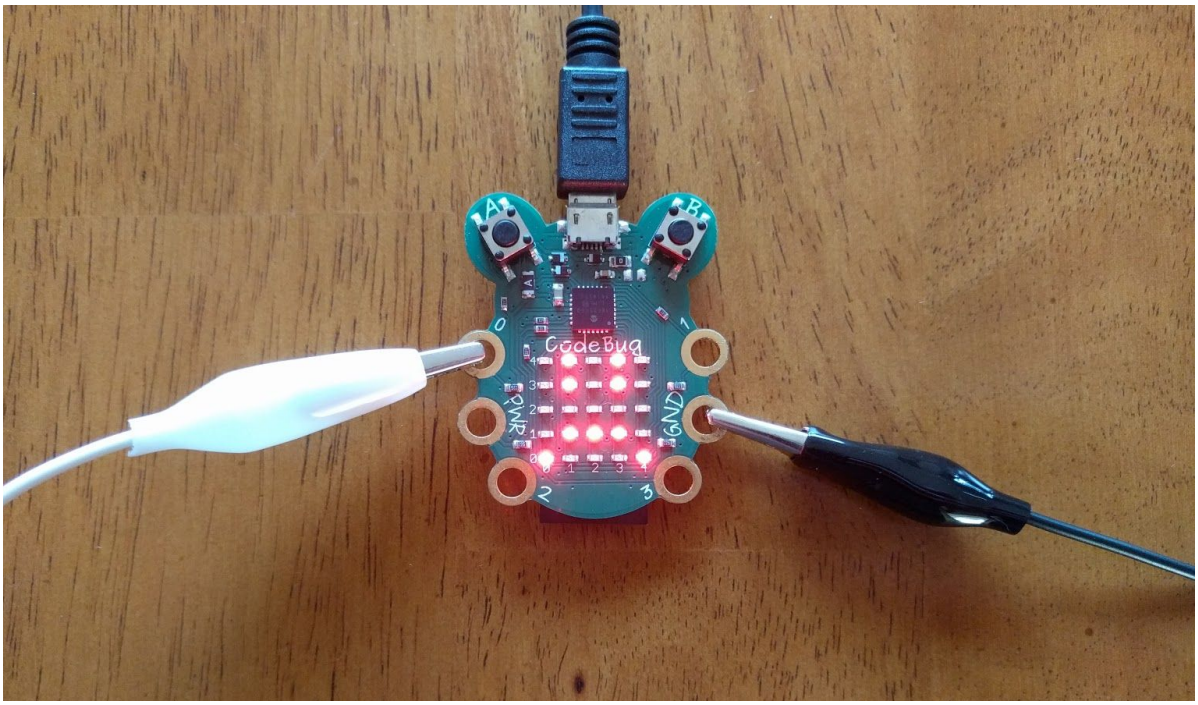
Check the boxes on your first "build sprite" to create a happy face. **Then create a sad face with your second "draw sprite".** The faces should look something like these:

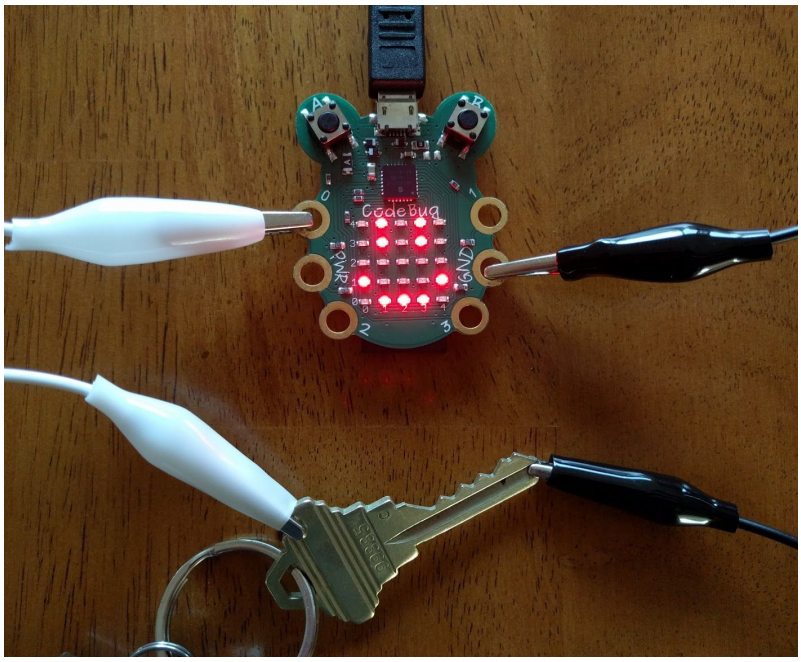
HAPPY **SAD**



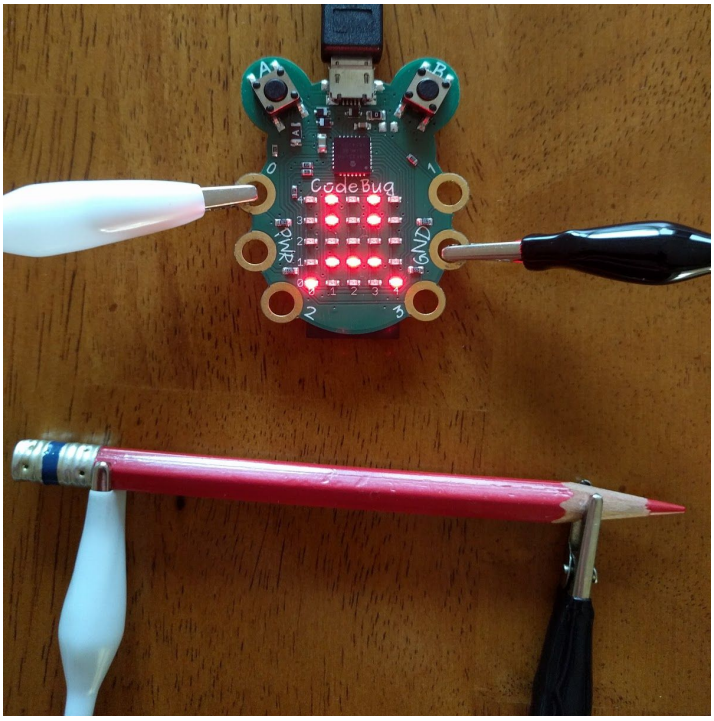
Your program is complete! **Follow the steps from "Loading your code" on page 9 to download and program your CodeBug**. You may need to unplug and plug your CodeBug back in if you do not see a red flashing light.

To test if an object or surface conducts electricity, you'll have to connect two alligator clips to your CodeBug. **Connect one clip to leg "0" and the other to "GND".**





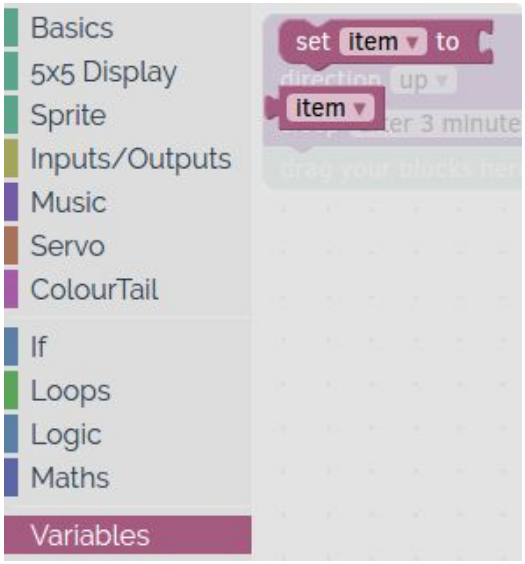
Now try connecting the alligator clips to different objects to test their conductivity. **Connect it to something that does conduct**, like metal keys, **and you'll see a happy face**.



If you connect it to something that does not conduct electricity, say a wooden pencil, **you'll see a sad face appear**.

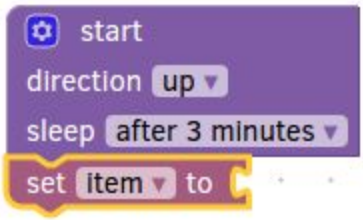
Exercise (4) Closest To 9 Game

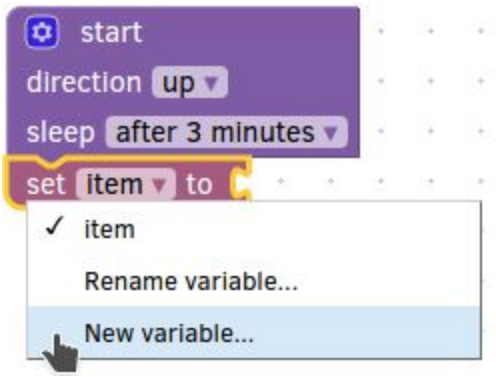
We're going to create a new program, so be sure to **create a new CodeBug project**.



We're going to start under the "Variables" Block Menu, located at the bottom of the menu list. **Open the "Variables" menu and select the "set item to" block**.

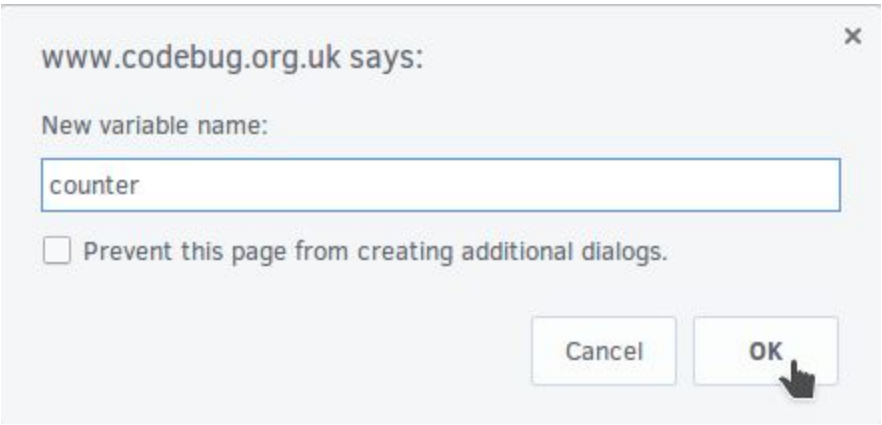
Snap the "set item to" block underneath the purple start block.





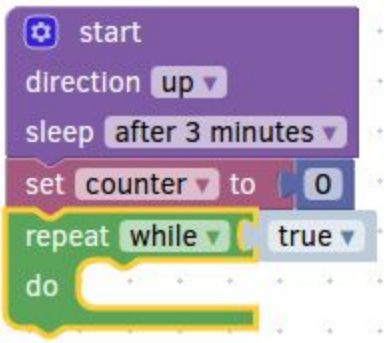
Click on "item" in your new block and select "New variable..."

Enter “counter” in the pop-up and press OK.



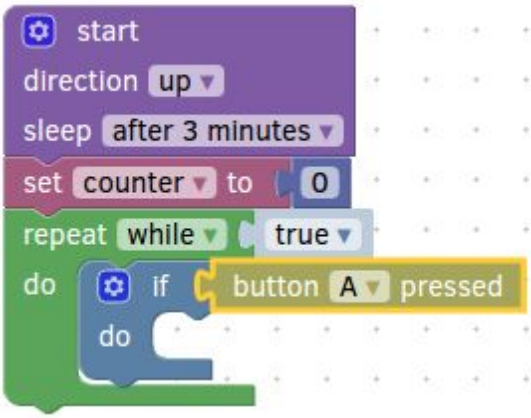
Get the “0” block from the “Maths” Menu and snap it to your “set counter to” block.

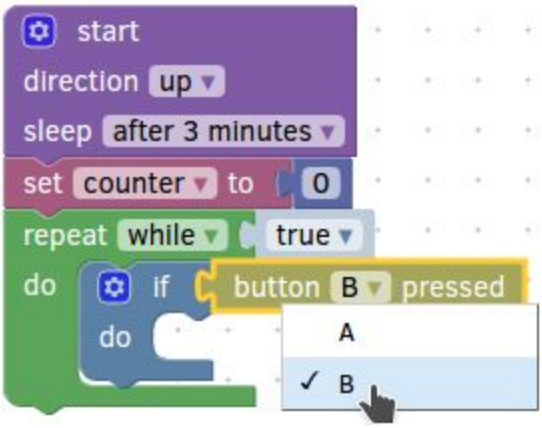
Next, **snap a “repeat while true” from the “Loops” Menu to your “set counter to” block**.



Insert an “if do” block from the “if” Menu into your “repeat while true”.

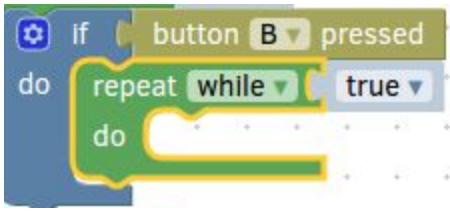
From the “Inputs/Outputs” menu, attach a “button A pressed” to your “if do” block.





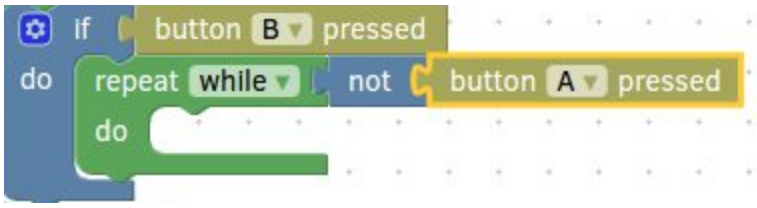
Click the “A” in “button A pressed” and change it to “B”.

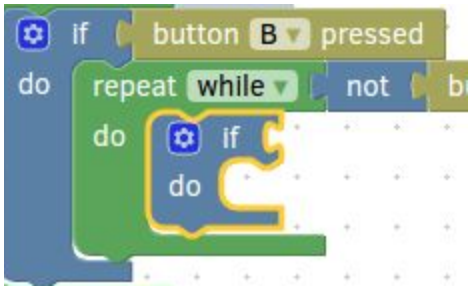
Get a “repeat while true” from “Loops” and attach it inside of your “if do” block.



From “Logic”, get a “not” block and snap it overtop the “true” in your “repeat while true” block.

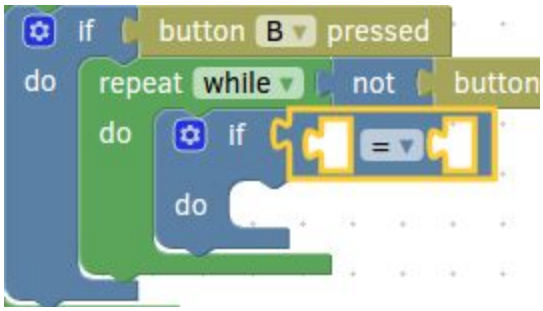
Get a “button A pressed” from “Inputs/Outputs” and snap it to the end of your “not” block.

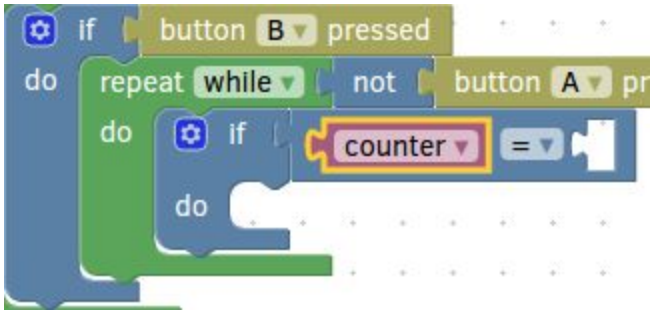




Pull out an “if do” from the “If” Menu and snap it inside your new “repeat while” block.

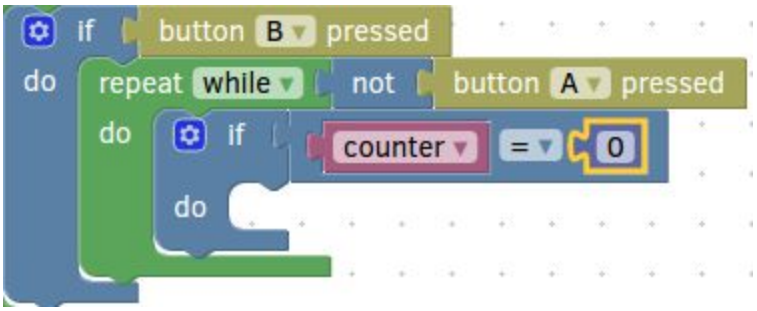
From the “Logic” menu, snap the “ = ” block to your new “if do”.

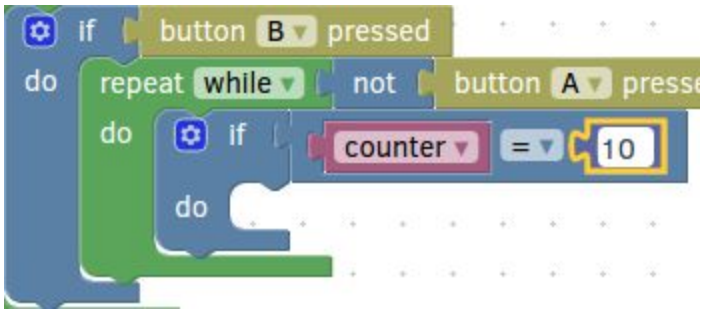




Inside of the first gap created by the “ = ” block, insert a “counter” block from the “Variables” Menu.

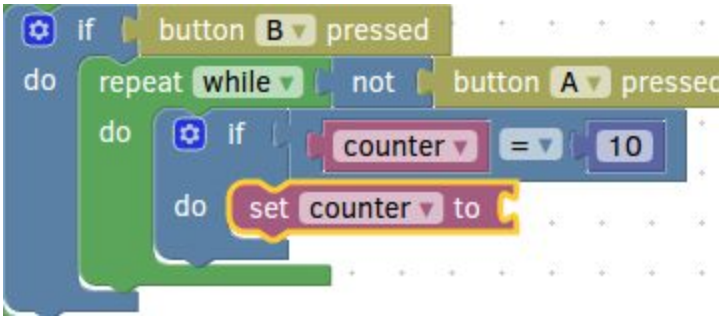
In the second gap, snap a “0” from the “Maths” menu.

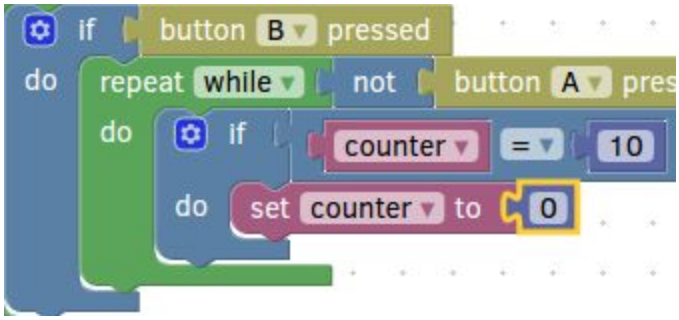




Click the “0” you just created and type in “10” to replace it.

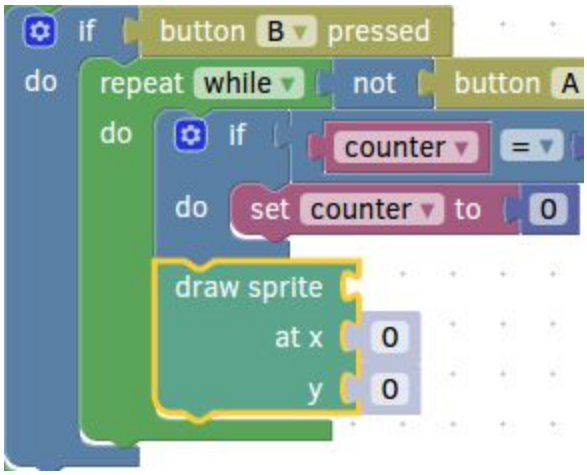
Get a “set counter to” block from “Variables” and snap it inside of your “if do” block.



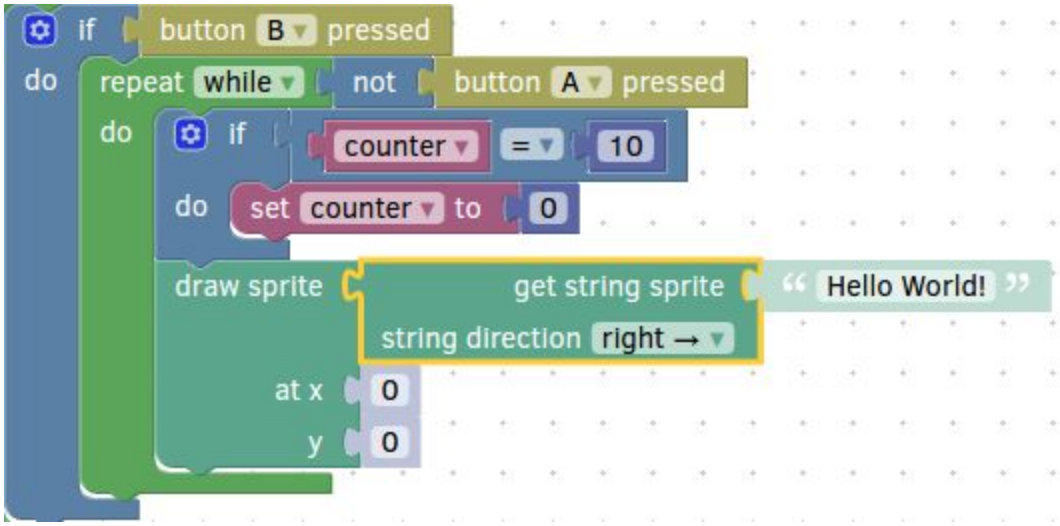


Attach a “0” from “Maths” to the end of your “set counter to” block.

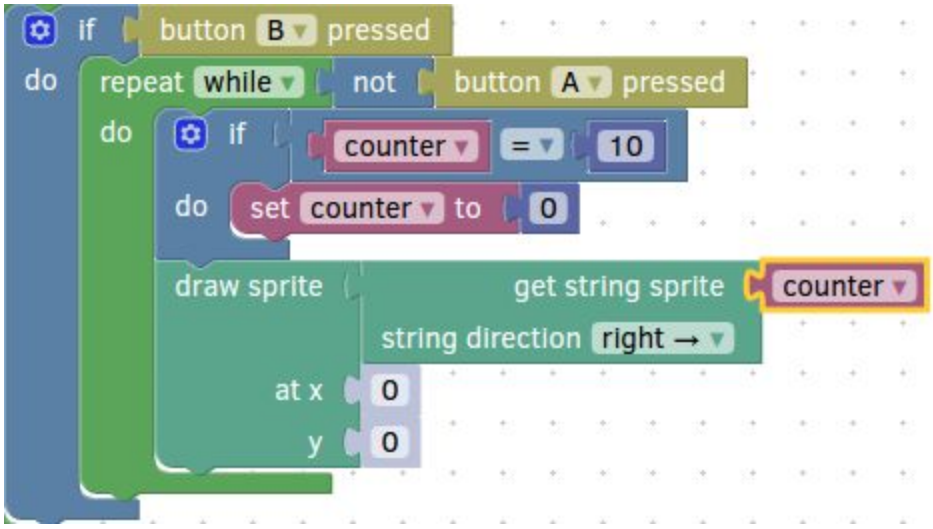
Next, **pull out a “draw sprite” from “Sprites” and attach it directly under (not in) your “if do” block.**



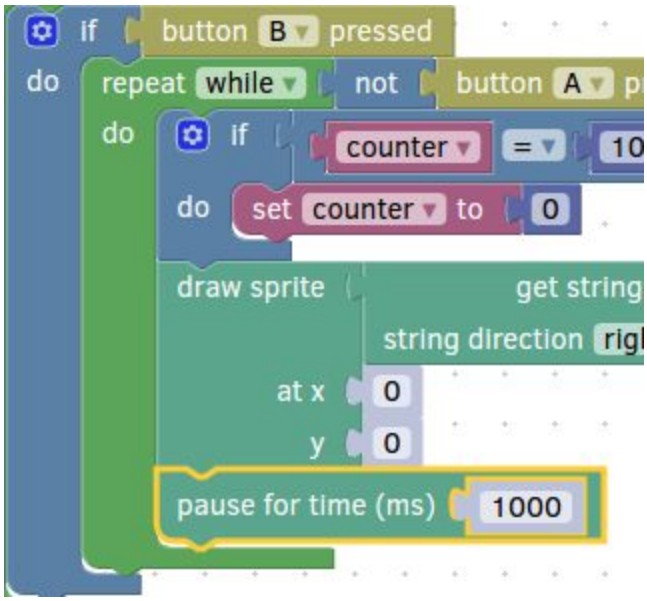
Also from “Sprites”, snap a “get string sprite” to the “draw sprite” you just created.



Grab a “counter” block from “Variables” and snap it overtop the “Hello World!” at the end of your “get string sprite”.

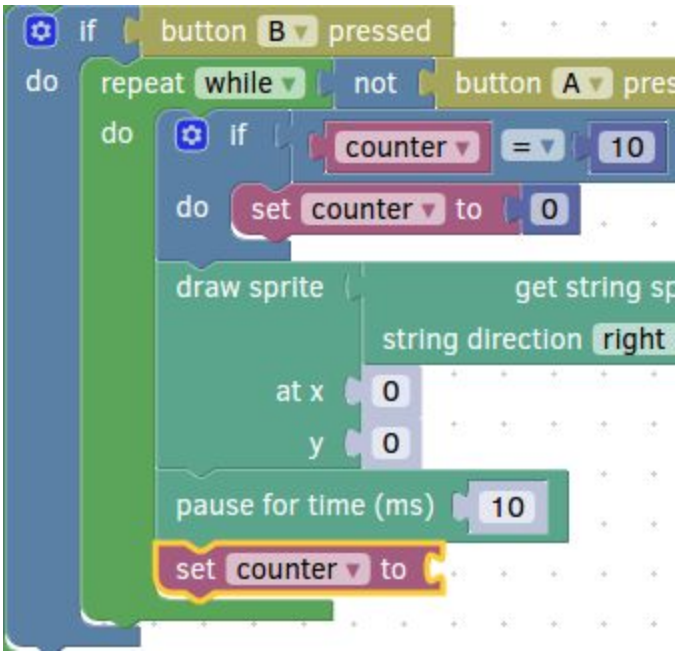


Under “Basics”, grab a “pause for time (ms) 1000” block and snap it beneath your “draw sprite” block.



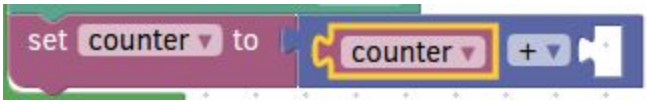
Click the “1000” in your “pause for time (ms) 1000” and change it to a “10”.

Get a “set counter to” from “Variables” and insert it beneath the “pause for time (ms) 10” you just created.



From “Maths”, get a “ + “ and attach it to your “set counter to”.

Snap a “counter” block from “Variables” into the first gap created by the “ + “.

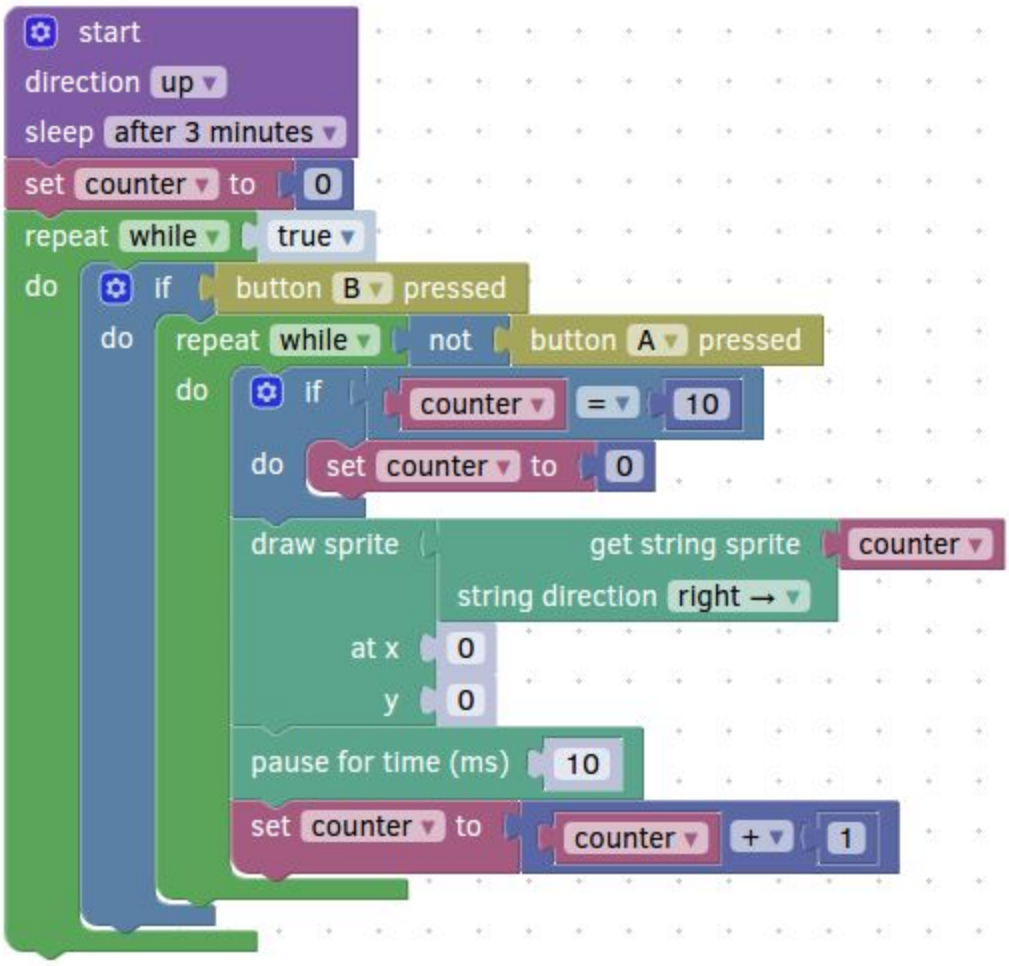


Finally, **insert a “1” block from “Maths” into the second gap in the “ + “ block.**

To play this game, press the **B** button to start/reset and button **A** to stop the counter. Your goal is to get number 9.

You can test this code using the on-screen CodeBug Emulator. If it works correctly, then **refer back to the “Loading your CodeBug” section on page 9** to see your program in action on your CodeBug.

The complete code for this exercise is below for reference.


start
direction up
sleep after 3 minutes
set counter to 0
repeat while true
do
if button B pressed
do
repeat while not button A pressed
do
if counter = 10
do set counter to 0
draw sprite
get string sprite counter
string direction right →
at x 0
y 0
pause for time (ms) 10
set counter to counter + 1