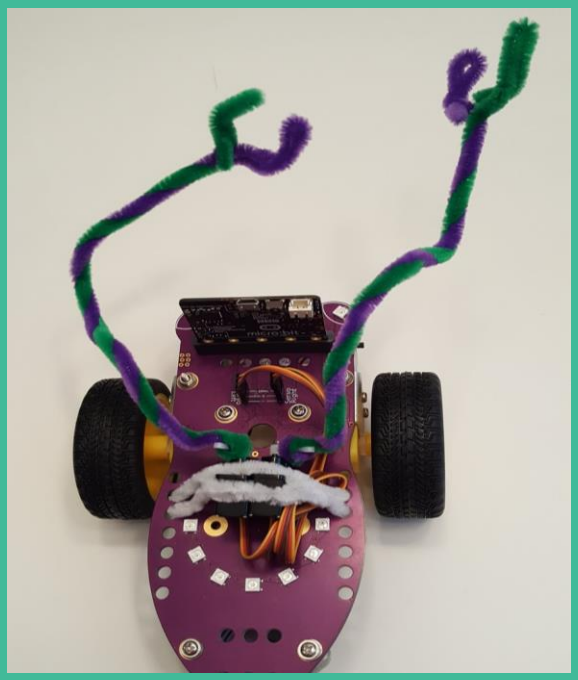
CREATURE CREATION

MISSION 6



Giggle Bot

TABLE OF CONTENTS

> CREATURE CREATION

YOUR ROLE: ANIMATRONICS ENGINEER

ⓘ Animatronics refers to the use of motors and wires to emulate an animal, or an inanimate object to make it lifelike.

YOUR TASK: DESIGN AND BUILD A CREATURE

You are an animatronics engineer for an upcoming movie. Your job is to design and build a new creature to star in the movie. There are two requirements from the director:

- It must react to light. It can seek light, fear light, or just move in some way based on light levels.
- It must have additional appendages that are controlled by servos.

ⓘ An appendage is a body part that extends from the body such as an arm, leg, or tail.

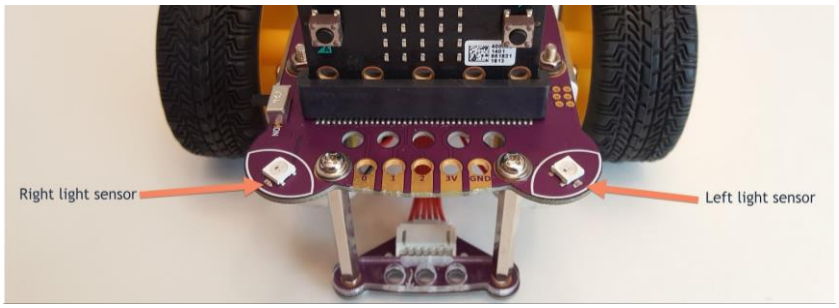
CONSIDERATIONS

- How is your creature going to move?
- Will it be attracted to light? Will it try to avoid light?
- Where will the extra appendages be attached? What purpose will they serve?

MATERIALS

- GiggleBot and good batteries
- micro:bit and provided cable
- Laptop / computer
- Two servos
- Pipe cleaners to attach the servos, alternatively zip ties or hot glue
- Flashlight (a cell phone flashlight works well)
- Optional: craft supplies to design and construct your new creature (Ex. construction paper, pipe cleaners, or cardboard.)

> LEARN: LIGHT SENSORS



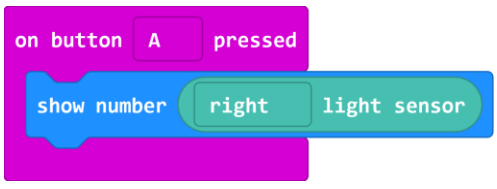
The GiggleBot has two built-in light sensors. They are very small and located right in front of the LED eyes on the GiggleBot. We can utilize light values from them individually (left or right sensor) or tell the GiggleBot to follow light. The light values on the GiggleBot range from 0 (dark) to 1023 (extremely bright).

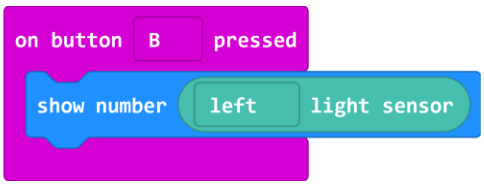
LET'S SEE WHAT THE LIGHT LEVELS ARE IN YOUR ROOM.

We will use the **on button A press** and **on button B press** blocks to initiate the program so that we can display the light levels during different conditions in your room. The right light sensor is on the right side of the GiggleBot (if you are the GiggleBot). If you are looking at the front of the GiggleBot, it is on the left side.

Connect a **light sensor** block (found under **GiggleBot --> more**) inside of a **show number** block (found under **Basic**). Since button **A** is on the same side as the right light sensor, program it to show the right light sensor value. Button **B** is on the same side as the left light sensor, so program it to show the left light sensor value.

Download and transfer the program to your GiggleBot. Move your GiggleBot to different locations in the room to determine the light levels. When you press button **A** or button **B** the light level value for that sensor will scroll across the micro:bit . Write these down so that we can use them later.



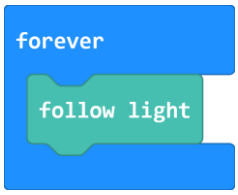


- What are bright values in your room?
- What are dark values in your room?

ⓘ Note: light values over 1000 are rarely reached.

> LEARN: REACTING TO LIGHT

Since the GiggleBot has a light sensor on either side of the front, it can be programmed to follow light.



- Place a **follow light** block inside of a **forever** loop.
- Download and transfer the program to your GiggleBot.
- Find a flashlight and turn on your GiggleBot.
- Use your flashlight to guide the GiggleBot around the room. It should follow the light just like a cat will follow a laser pointer around.

ⓘ Note: a real laser pointer won't work for this project. You need to use a flashlight.

Think about how you could use this block to create a creature.

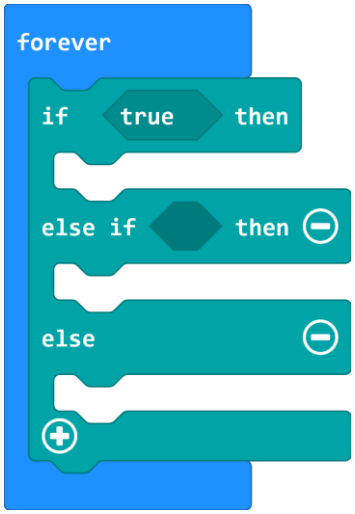
Do you want to make a creature that follows light?

Maybe you want to create a creature that fears light.

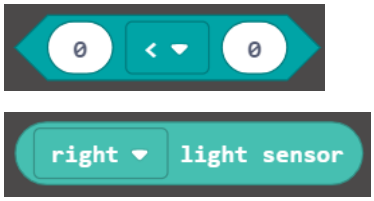
Let's look at what that code could look like.

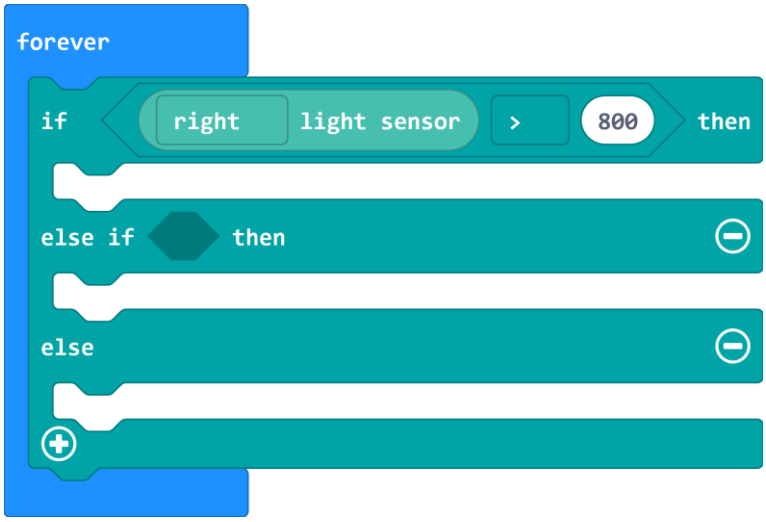
We are going to program the GiggleBot to show a **dislike** for light. We will use an **if ... then ... else ...** block to tell the GiggleBot what to do.

- Place an **if then ... else ...** block (found under **Logic**) inside of a forever loop (found under **Basic**).
- Click on the **+** sign at the bottom of the **if then ... else ...** block to add an **else if ... then ...** section. This allows us to include more than one **if** statement.



- Combine a **0 < 0** block (found under **Logic**) and a **right light sensor** block (found under **GiggleBot -> more**) to create the first **if** statement.
- Test whether the right light sensor value is > **800** (greater than 800). 800 means a particularly bright light.

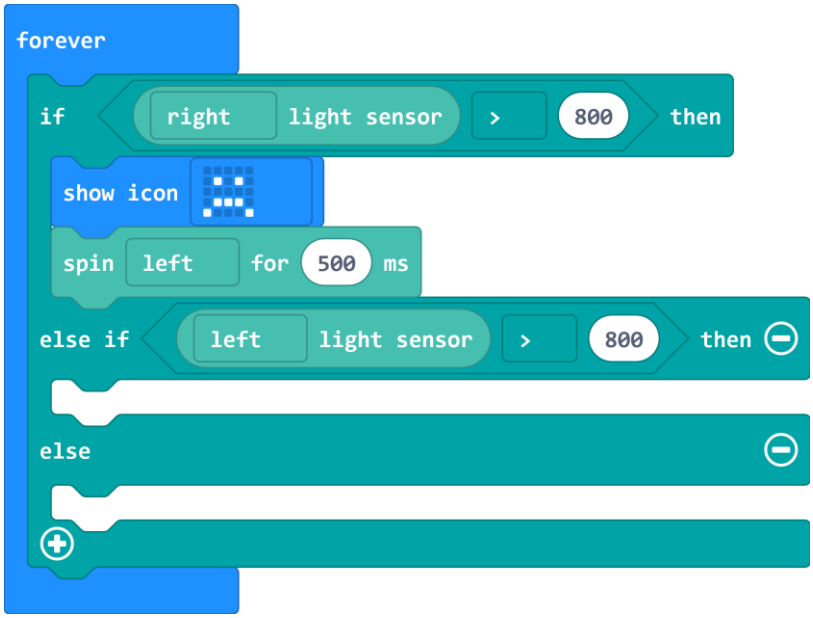




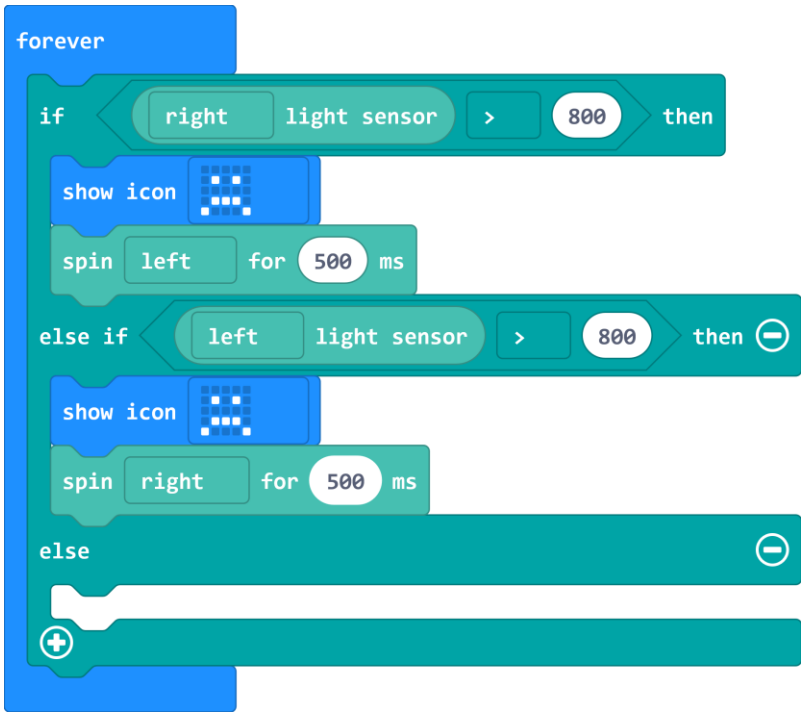
The program above tells the GiggleBot to react if the light sensor value is above 800. You may need to adjust this number based on the light conditions in your room.

ⓘ Remember, the light values on the GiggleBot range from 0 (dark) to 1023 (extremely bright).

Connect blocks to the **then** statement to tell the GiggleBot to show that it does not like the light and to turn away from it. One idea is shown here.



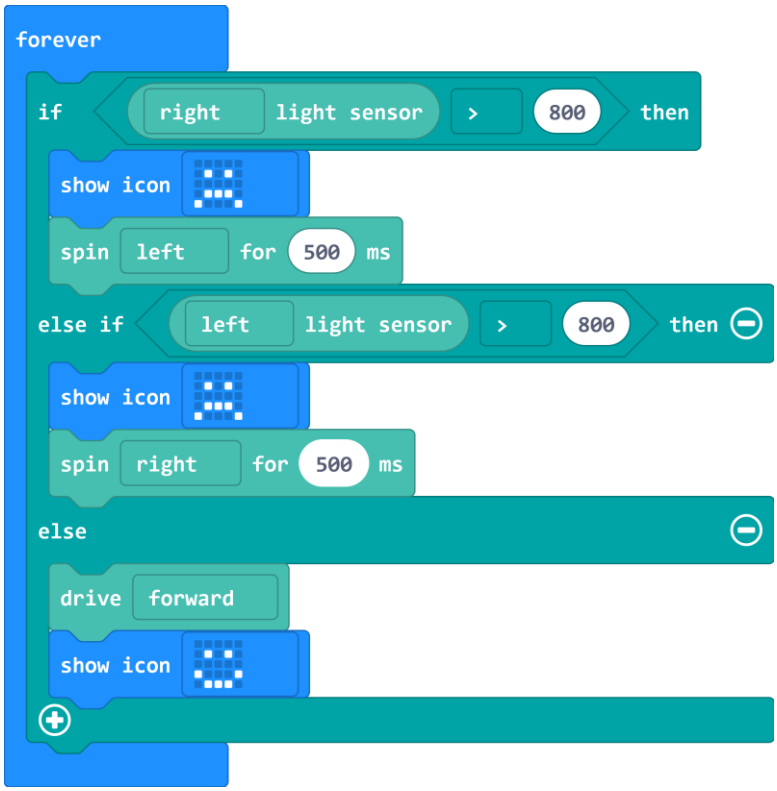
Repeat the same process for the left light sensor using the **else if ... then** section of the block.



Last, we will program the GiggleBot to drive forward when those light conditions are not met.

This means that the GiggleBot should drive forward when it is not particularly bright.

When a high light level (above 800) is sensed, it will turn away from the light.

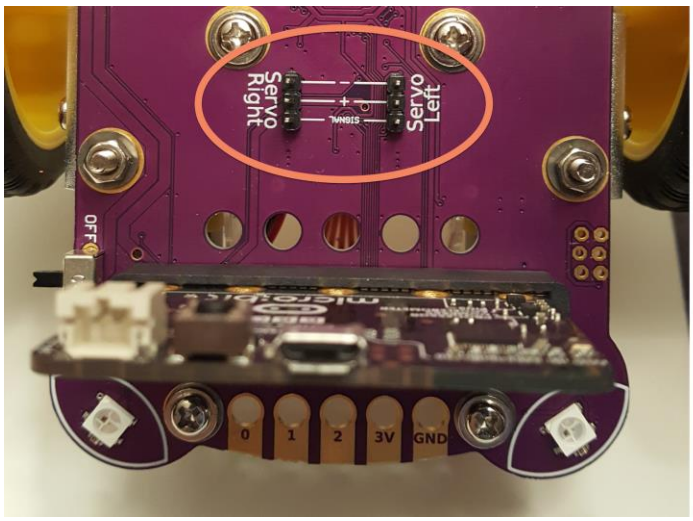


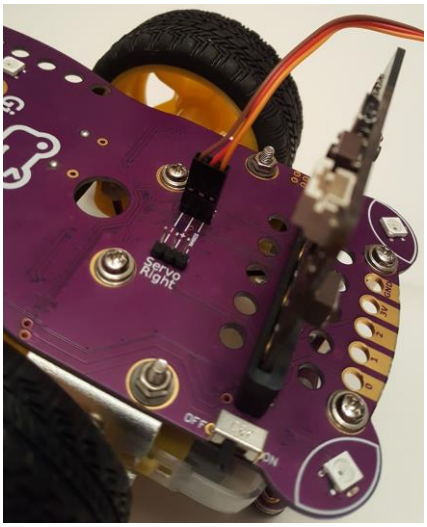
- Download and transfer the program to your GiggleBot.
- Turn off the lights and try it out.
- What happens when you shine a light on the front right side of the GiggleBot?
- Left side?
- In the darkness?

> LEARN: USING TWO SERVOS

In mission 5 you used one servo to create a dump truck. In this mission you will use two servos to create appendages for your creature. These can be anything you like - arms, legs, wings, fingers, or tails.

Connect two servos to your GiggleBot using the pins directly behind the micro:bit. Place the dark wire (brown or black) towards the back of the robot.





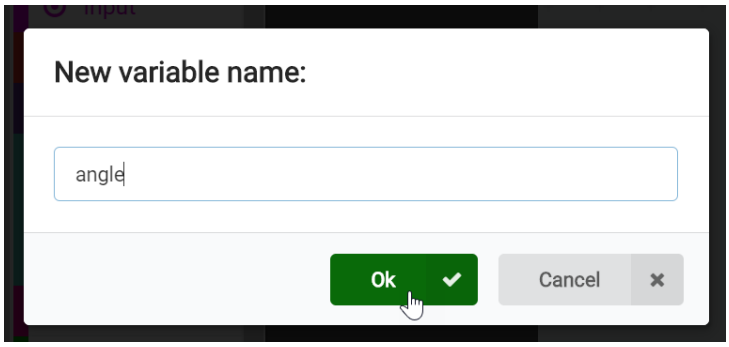
Ensure that the servos are connected properly. If the servos are not connected the correct way, they will not work. The brown wire (or black wire) needs to be towards the rear end of the GiggleBot.

Let's look at what the servos can do. This will help to figure out what motions you will use for your creature. To do this, we will create a program in the next section to see the range of motion for the servos using a variable.

> LEARN: VARIABLES

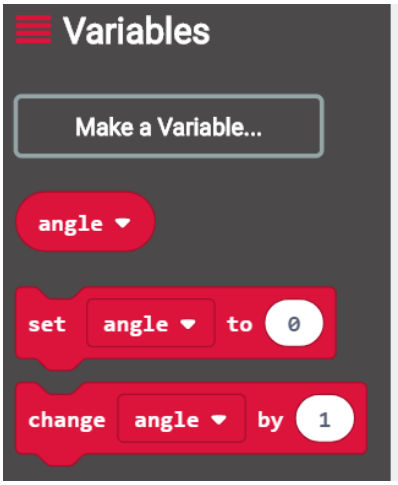
A new type of block that we will use is a variable block. *Variables* are used to store information. Just like in algebra, variables represent a number. In this case it will represent whatever number we want the servo to be set to.

Under **Variables**, click on "make a variable".

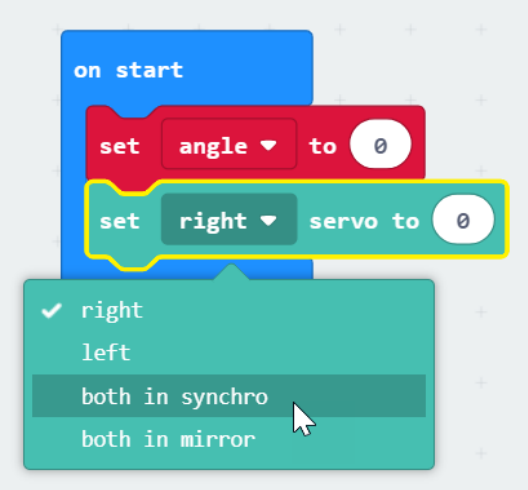


You can name your variable anything you want. Since it is going to represent the angle that the servo is set to, it is named "angle" in the example program.

Once you have created your new variable and click **ok**, more blocks will appear for you to use.



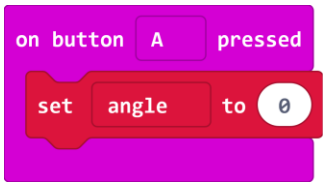
Move an **on start** block to the workspace.



- Inside of this block, connect a **set angle to 0** block (found under **Variables**).
- Connect a **set right servo to 0** block (found under **GiggleBot**).
- Use the dropdown menu to change the block to say **both in synchro**.

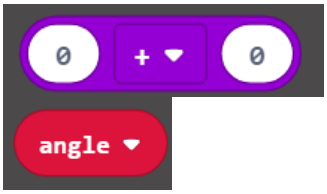
The servos will both be set to **0** each time the GiggleBot is turned on.

Next, we will program the GiggleBot to move the servo arm 45 degrees each time button A is pressed.

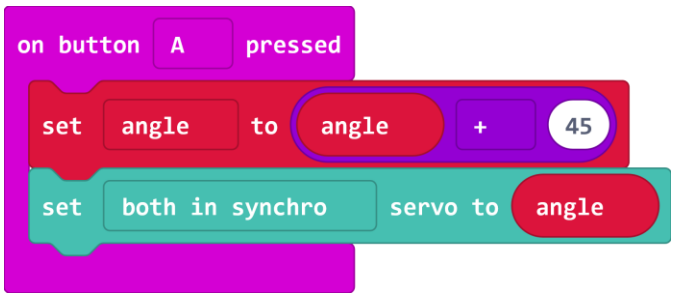


- Move a block **set angle to 0** block inside of a **on button A pressed** block.

Instead of just setting the variable to a constant value as we did before, we will set it to 45 degrees more than the previous value.



- Combine an addition block (**0 + 0**) block (found under **Math**) and an **angle** block (found under **Variables**).
- Change the second value in the addition block to the number of degrees you want the angle to change each time the button is pressed.
- Last, connect a servo block, set it to **both in synchro**.
- Again, instead of using a constant value to set the servo to, use the **angle** variable so that the servo will move to the new variable value each time button **A** is pressed.

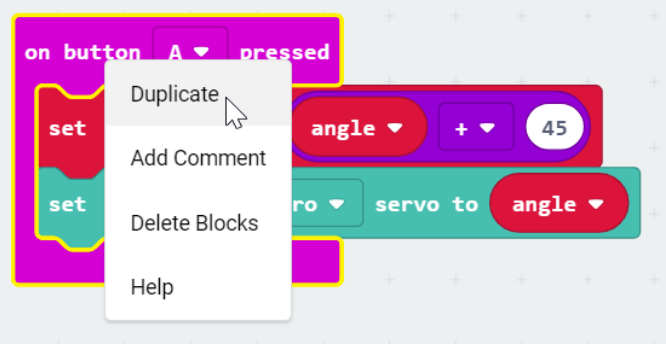


What will that behavior look like?

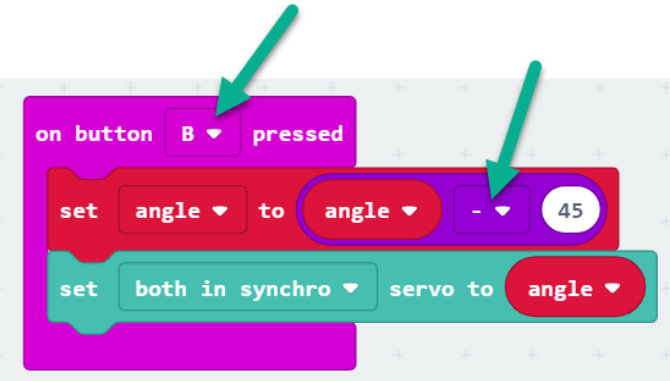
1. GiggleBot turned on → both servos move to 0 degrees
2. Button **A** pressed once → both servos move to 45 degrees
3. Button **A** pressed twice → both servos move to 90 degrees (45 +45)
4. Button **A** pressed three times → both servos move to 135 degrees (90 + 45)

WHAT IF YOU WANT THE SERVOS TO MOVE BACK THE OTHER WAY?

Right click on the on button A pressed block and duplicate that part of the code.



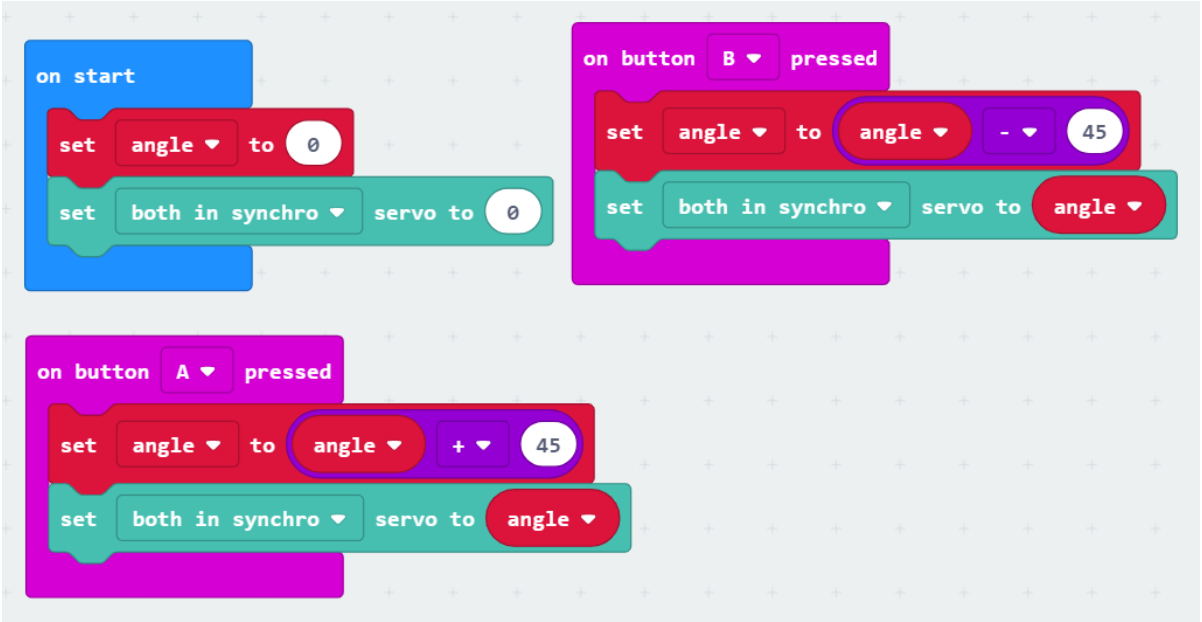
In the duplicated code, change it to read **on button B pressed**. Then, change the set angle to *subtract* 45 degrees.



- Each time you press the **A** button the servos move **+45** degrees in synchro, going one way.
- Each time you press the **B** button the servos move **-45** degrees in synchro. going the other way.

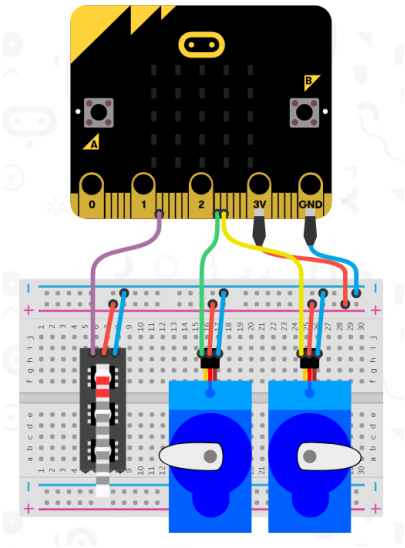
They can now go back and forth!

Here's what the code looks like so far:



Download and transfer the program to your GiggleBot to see how the servos move together.

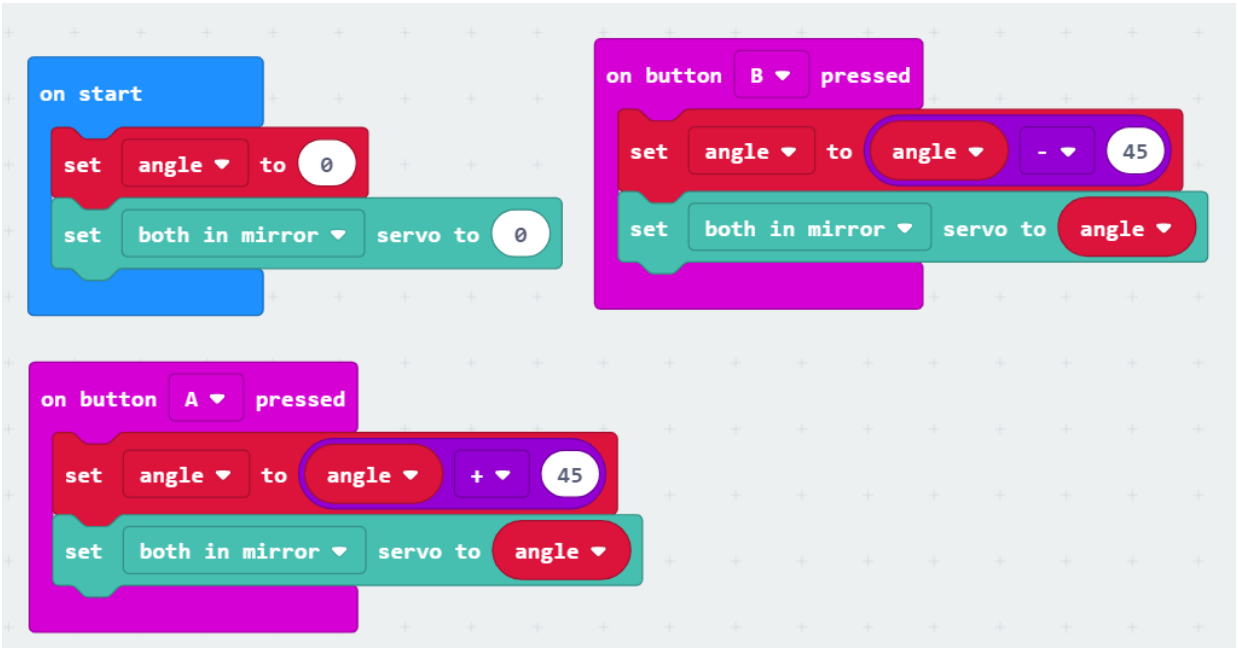
TRY OUT THE MIRROR OPTION



Use the dropdown menu on the servo blocks to say **in mirror**. Take a look at the micro:bit emulator on the MakeCode webpage. You will see that the servos shown are pointing in opposite directions since we set it to **in mirror**. When the servos were set to move **in synchro** they moved in the exact same direction.

Download and transfer this new program to your GiggleBot to see how the servos move when they are programmed in mirror.

Here's the code using the **mirror** option:



ⓘ Think about how you could use the servos "in synchro" or "in mirror" to make appendages for your creature.

> PLAN IT OUT

Remember, your creature needs to react to light AND have appendages (legs, arms, wings) that move using servos. If you need help with the servos, look back at the Learn section in mission 5.

- Will your creature seek light? Fear light? Love light?
- Will your creature have arms? Legs? Wings? A tail?
- How will the appendages move? Will they move vertically? Horizontally? Diagonally?
- When will the appendages move? When it is happy? Scared? Excited?

Jot down your ideas about what the creature will look like and how it will behave before beginning your build.

BUILD YOUR CREATURE

Use craft materials or recycleables to build your creature. Be sure to secure the servos tightly onto the GiggleBot using pipe cleaners, hot glue, or zip ties, so that only the arm of the servo moves around, not the entire assembly.

ⓘ Check to make sure that none of the decorations on your GiggleBot are blocking the light sensors or the wheels.

BUILD YOUR PROGRAM

Write your program for the new creature. Remember there are two requirements:

- It must react to light. It can seek light, fear light, or just move in some way based on light levels.
- It must have additional appendages that are controlled by servos.

Once you are finished, download and transfer the program to your GiggleBot.

> ARE YOU STUCK??

Let's build together!

BUILD THE APPENDAGES

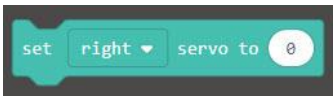
For this creature we are going to add two arms using pipe cleaners. One way to make arms is to twist two pipe cleaners together and then bend the ends to make claws. Two pipe cleaners will help the arms stand up better on their own. Use colors to give your creature some character.

Attach the pipe cleaners to the servos by wrapping the pipe cleaners tightly around the servo arm.

Next, secure the servos to the body of the GiggleBot with another pipe cleaner or two. You may prefer to use zip ties or hot glue. Make sure that they are attached tightly. Your GiggleBot now has arms that it can wave around!

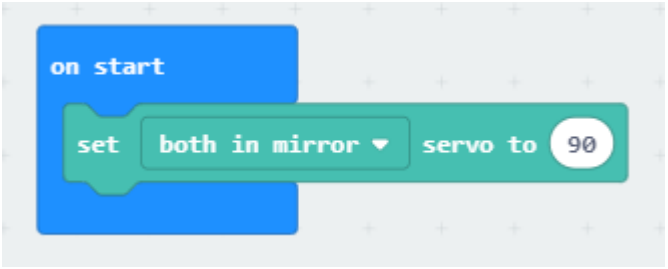
WRITE THE PROGRAM

We need to program the arms to move around as well as tell the GiggleBot to react to light.



To start the program, put a **set right servo to 0** block inside of an **on start** block. Use the dropdown menu to change **right** to **both in mirror**.

In the example below the servos are set to 90. This will tell the servos to move the arms so that that they point straight out of either side of the GiggleBot when it is turned on. You may need to adjust the value for your GiggleBot based on the arrangement of your servos.



Next, we will program how the GiggleBot moves around.

- In a forever block, put a **set both motors speed to slowest** and a **follow light** block.
- Set the speed to your desired speed. Do you want the creature to move quickly? Slowly? A medium speed?

Now we will program the GiggleBot to move its arms in addition to following a light source. Let's tell the GiggleBot to raise its right arm if the light sensor on the right side is above a certain value and to raise its left arm if the light sensor on the left side is above a certain value.

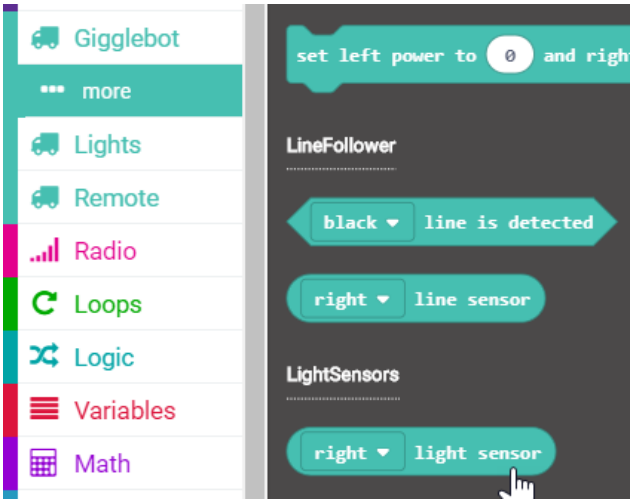
To do this, add an **if true then ... else ...** block to the forever loop.

Click on the + symbol on the **if true then ... else ...** block to add **else if ... then ...** to the block. This will allow you to have two **if** statements.

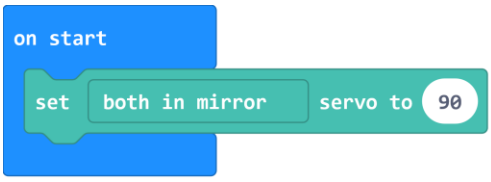
Next, we will tell the GiggleBot what to do based on the light level.

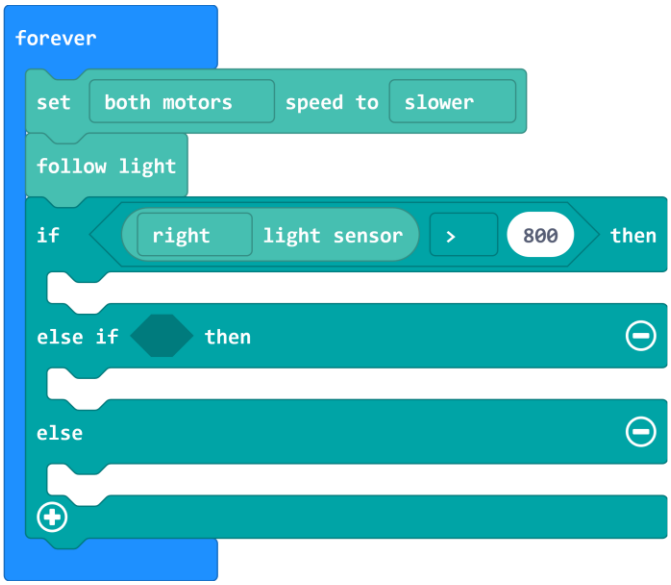
- For the first **if** statement, replace the **true** statement with a comparison block (a **0 < 0** block), found under **Logic**. We will use this block to tell the GiggleBot what conditions it is looking for.

- Move a **right light sensor** block into the first slot in the comparison block. This block is located under **GiggleBot → More**.



- Change the inequality to **greater than** (or **>**).
- Change the second value to a bright value in your room. The value in the example is 800, but it may be different for your environment. Refer back to your test values in the Learn section.



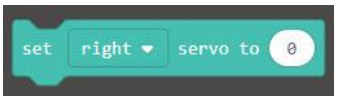


At this point, we need to tell the GiggleBot what do do if that condition is met.

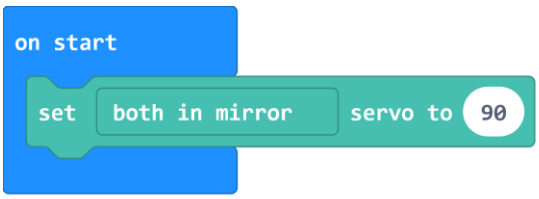
- Do you want some of the onboard lights to turn on?
- Do you want it to spin around?
- Do you want it to raise its arms?

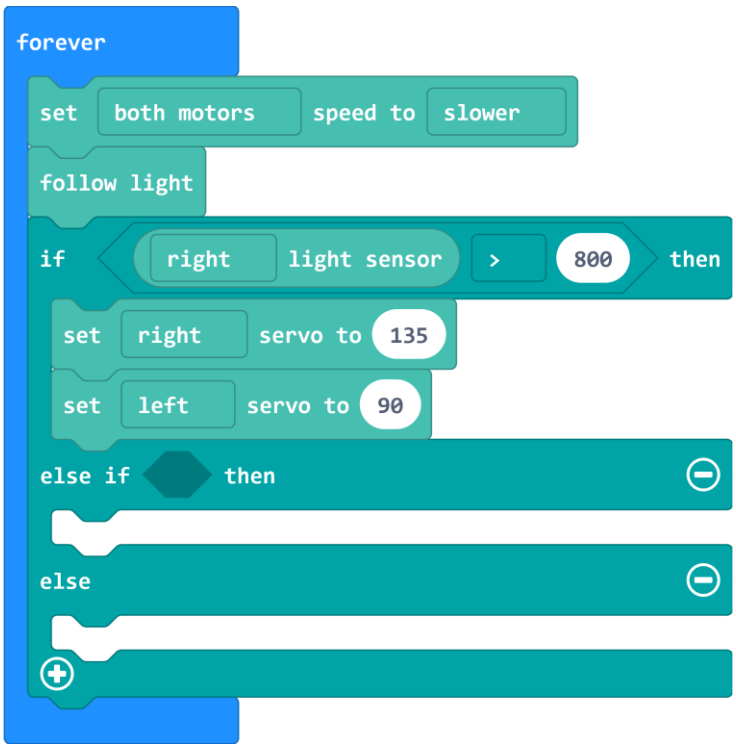
For this example we will tell the GiggleBot to raise its right arm if the light level is above 800. You may choose to have your GiggleBot react differently.

- Connect a **set right servo to 0** block to the body of the first **if** statement and change the value to tell the GiggleBot to raise its arm. Your value may differ from the example.



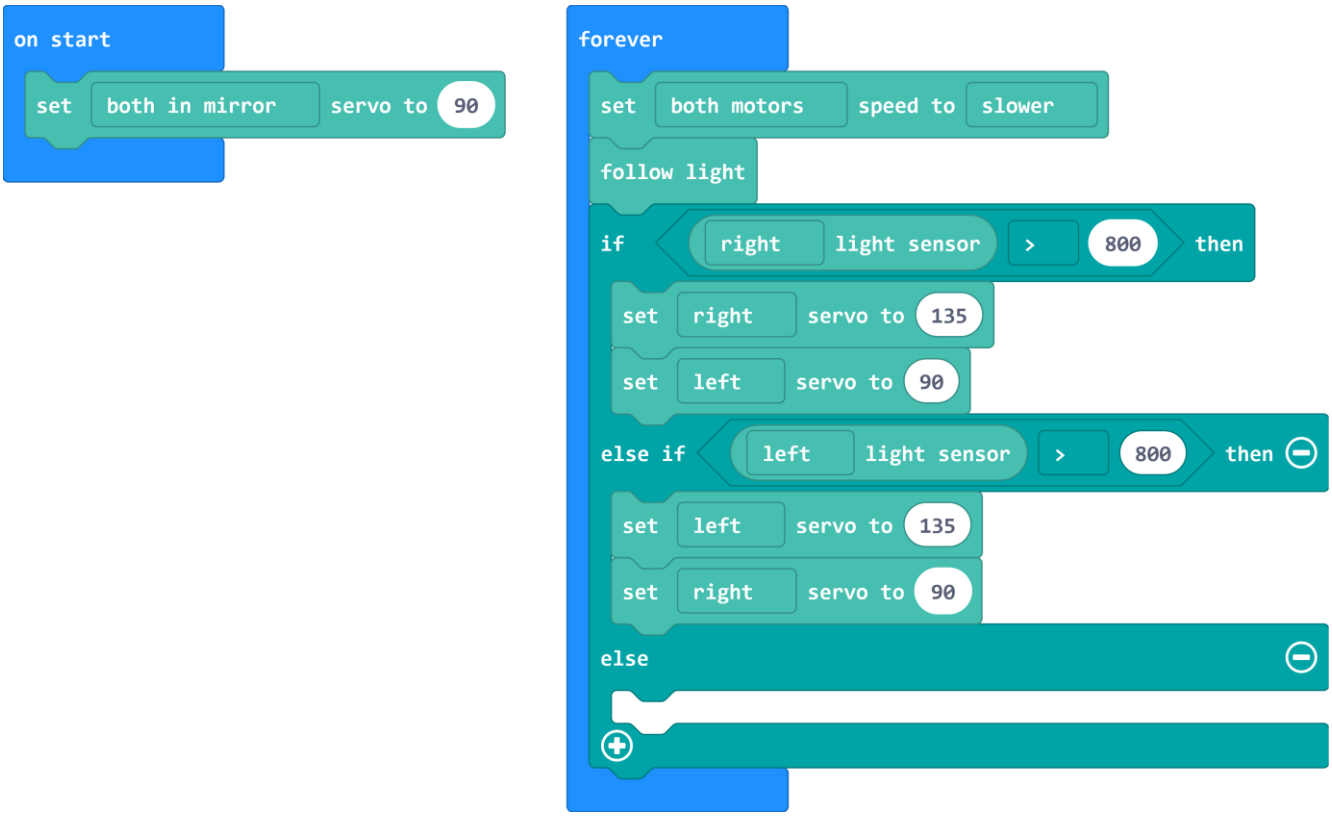
- Add another **set right servo to 0** block and change it to *left* servo. Change the value in this block so that the arm goes to a neutral position. In the example, the same value is used as the **on start** value.





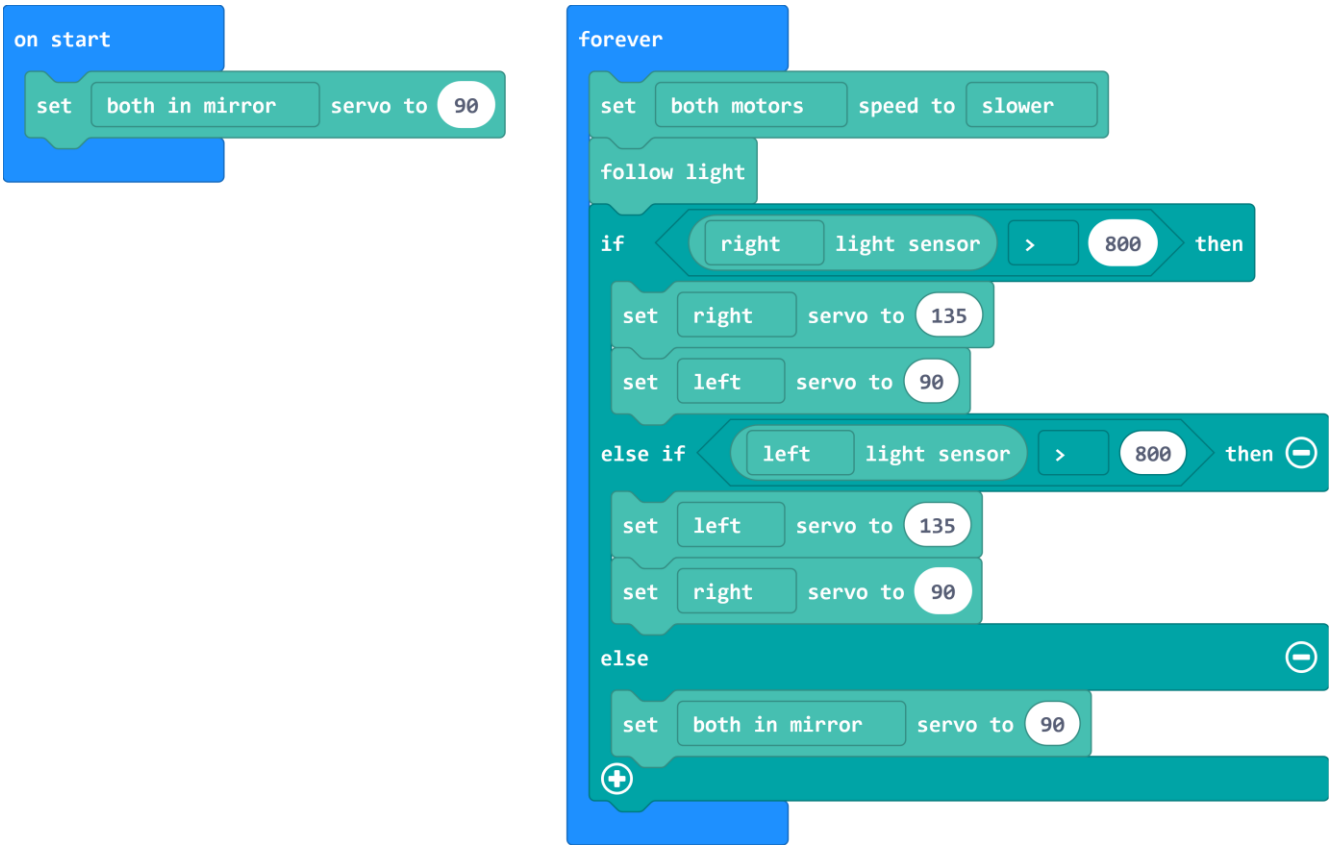
Complete the process for the **left** light sensor.

- Connect these blocks to the **else if ... then ...** body.
- Reverse the arm motion.



Last, tell the GiggleBot what to do if neither condition is met. In the example below, the GiggleBot will move its arms to a neutral position.

This is the complete program so far:



- Name, download and transfer the program to your GiggleBot.
- Use a flashlight to change the light levels for each light sensor and see your GiggleBot creature react.

> TRY IT OUT

Did your GiggleBot creature move and behave exactly as you had anticipated?

Do you see any areas that you would like to modify or improve upon?

In engineering,

- we try a solution
- think about what needs to change
- and we iterate

As you modify and revise your program, save each new program with a version number – you never know when you might want to take another look at an older version (for example: Creature_V1 where "V" stands for version).

EXTENSION

Program the GiggleBot to follow light WITHOUT using the **follow light** block.

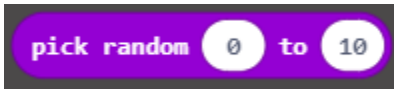
ⓘ Hint: Use a series of **if ... then ....** blocks.

WHAT IF YOUR CREATURE DOES NOT LIKE LIGHT?

How could you modify your program to make the GiggleBot react to lower light levels?

- Could you display a face on the micro:bit to show how the GiggleBot is feeling?
- Could you program the GiggleBot to drive around while reacting to light?

Investigate what the **pick random** block (found under **Math**) does. How could you use this with your programs? One idea for using the block is to program the GiggleBot to move forward and turn for random amounts of time.



- Make sure your previous work is named.
- Change the name and then download and transfer the program below onto your GiggleBot. Modify it to work with your GiggleBot.