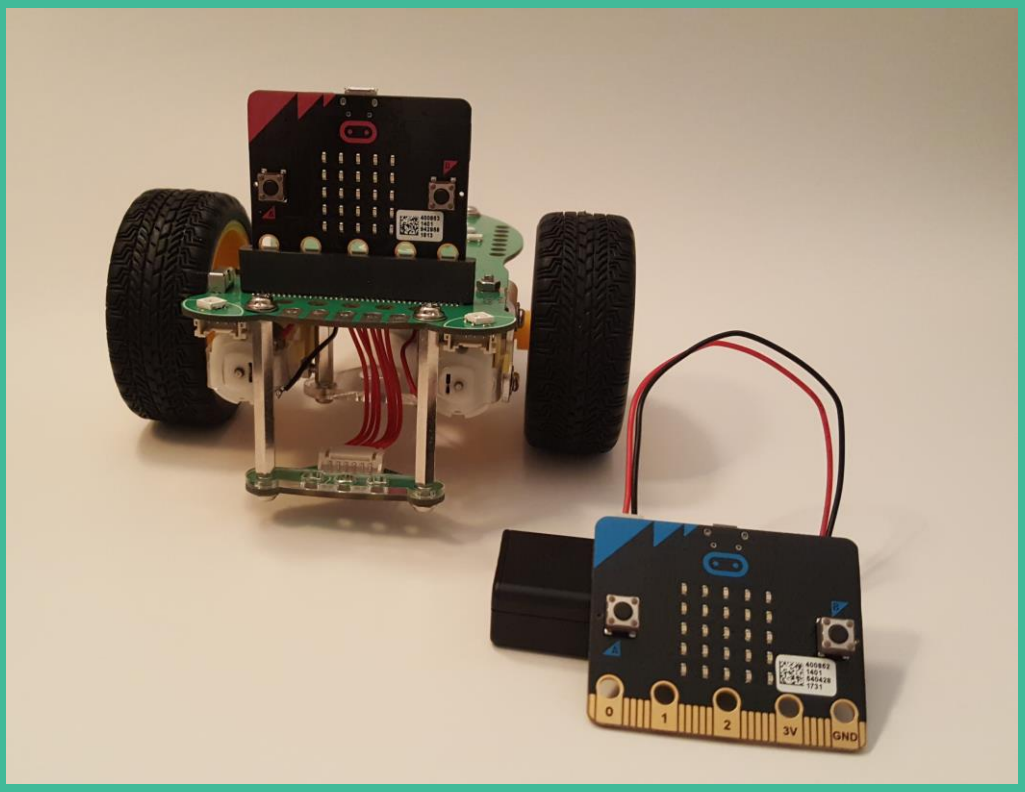
REMOTE CONTROL

MISSION 8



Giggle Bot

TABLE OF CONTENTS

> REMOTE CONTROL

YOUR ROLE: ASSISTIVE TECHNOLOGY ENGINEER

ⓘ Assistive technology (AT) is any item, piece of equipment, software program, or product system that is used to increase, maintain, or improve the functional capabilities of persons with disabilities.

YOUR TASK: CONTROL THE ROBOT FROM AFAR

Program a remote controller for the GiggleBot using a second micro:bit so that you do not need a tethered controller (like the one you made in mission 3). Then, build a holder, glove, or grip for the micro:bit controller for the GiggleBot so that someone with limited fine motor control can drive the GiggleBot around.

ⓘ Fine motor skills - the coordination of small muscles movements in the hands and fingers, with the eyes.

CONSIDERATIONS

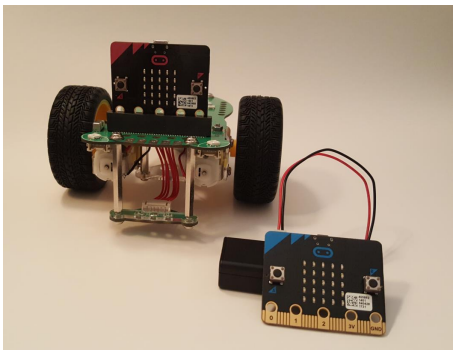
- How can you control the GiggleBot with a second micro:bit?
- What inputs can be used on the extra micro:bit to control the GiggleBot?

MATERIALS

- GiggleBot and good batteries
- micro:bit and provided cable
- Laptop / computer
- Additional micro:bit (not the one in the GiggleBot) and its battery pack
- Craft supplies to build a holder for the additional micro:bit to make it wearable (Ex. construction paper, pipe cleaners, rubber bands, a glove, string, or tape)

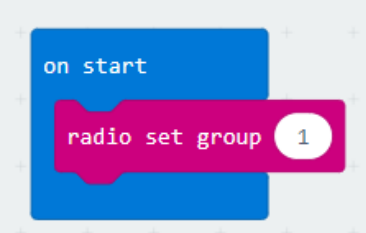
> LEARN: RADIO

This mission is going to be a bit different from the others because we need to program the GiggleBot as well as the additional micro:bit that we will use as a controller.



PROGRAMMING THE HAND-HELD CONTROLLER

SETTING THE RADIO GROUP



> Move an **on start** block to the workspace. Under **Radio**, find the **radio set group __** block and connect it to the **on start** block.

This block sets the group ID for radio communication. The micro:bit can only listen to one group at a time. Later, we will set the group ID for the GiggleBot to the same number so that the two micro:bits can communicate with one another.

ⓘ **Note**: The group number will need to be different for each micro:bit / GiggleBot pair. If you are in a classroom with multiple GiggleBots, each group needs to be assigned a different group number.

SENDING MESSAGES

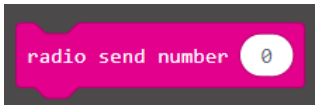
Now, let's program the **A** and **B** buttons to send a message to the GiggleBot to get it to do something.

> Move an **on button A pressed** and an **on button B pressed** block to your workspace.

Determine what you want the GiggleBot to do when those buttons are pressed and program it.

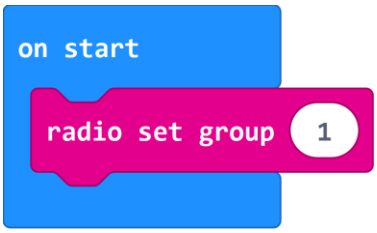
- Do you want the robot to spin around when the **A** button is pressed and stop when the **B** button is pressed?
- Do you want the GiggleBot to do a series of movements when the **A** button is pressed and turn on lights when the **B** button is pressed?

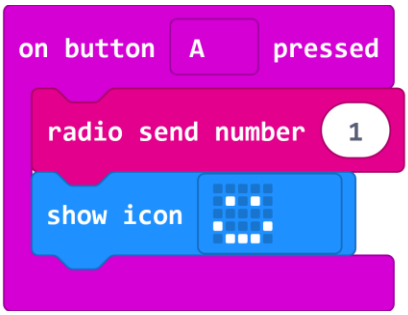
It is up to you!

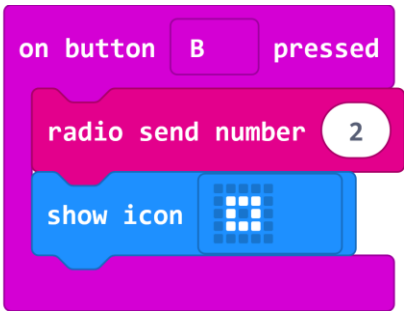


We are not going to program those movements yet, but assign each a number using the **radio send number ___** block. For this example we are also going to display an icon on the additional micro:bit to represent the GiggleBot's movement.

An example is shown here:





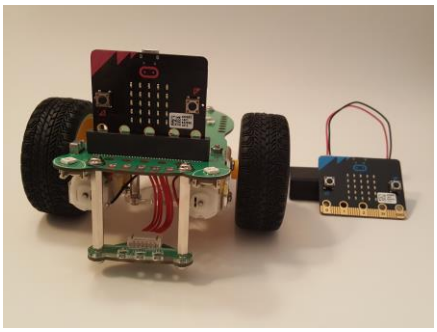


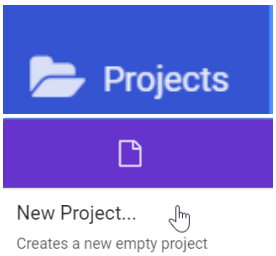
- Be sure that you name this program before you save it. Since we are writing two programs, it is important that you are able to easily find this program later if you want to modify or add onto it.
- Download and transfer your program to your additional micro:bit. This will be used as the controller for the GiggleBot.

When this micro:bit is turned on and the buttons are pressed, it will not do anything YET. Now, we need to program the GiggleBot so the additional micro:bit has something to control.

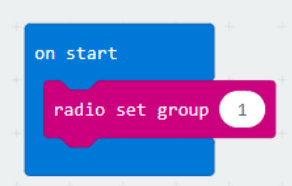
THE GIGGLEBOT AS A REMOTE CONTROLLED ROBOT

Now it's time to move to programming the GiggleBot. Set aisde the other micro:bit for the moment.





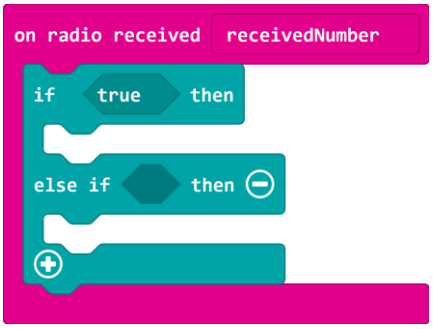
Clear your workspace by either moving all of your blocks to the trashcan or by clicking on the **Projects** icon and then clicking on **New Project** . Now you can write a new program for the GiggleBot.



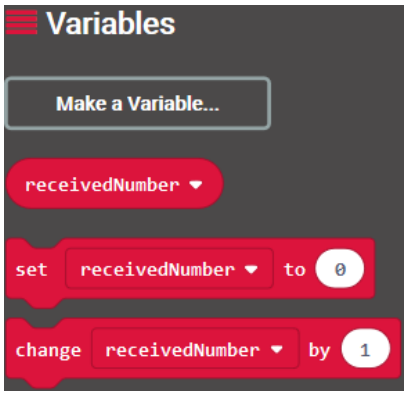
This next program will start out in a similar fashion to the one for the hand-held remote. We need to set the radio group to the same group as the remote. Otherwise, someone else may be able to control your GiggleBot.

Next, we will use a series of **if __ then __** statements to tell the GiggleBot what to do when each number is received from the micro:bit remote.

> Find a **on radio received receivedNumber** block under Radio and move it to your workspace.

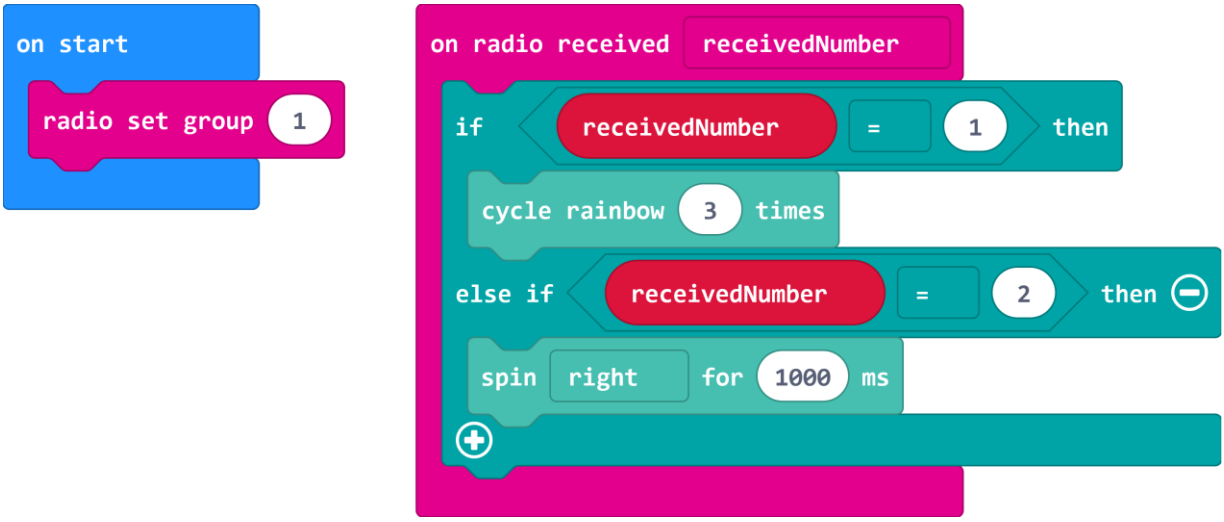


> Build a **if __ then __ else if __ then __** block by using the **+** sign and start filling it with comparison blocks like shown here. See mission 6 for details.



The radio message the GiggleBot will receives will store a number in a variable. Just like in algebra, variables represent a number. In this case it will represent whatever number the additional micro:bit sends. This **receivedNumber** block can be found under Variables. It gets created for you automatically as soon as you bring a **on radio received receivedNumber** block on the workspace.

Build the conditional statements using the comparison blocks and the **receivedNumber** block so it looks like the code below.



- Name this program so it gets saved.
- Download and transfer this file to your GiggleBot micro:bit.

FIRST REMOTE CONTROL RUN

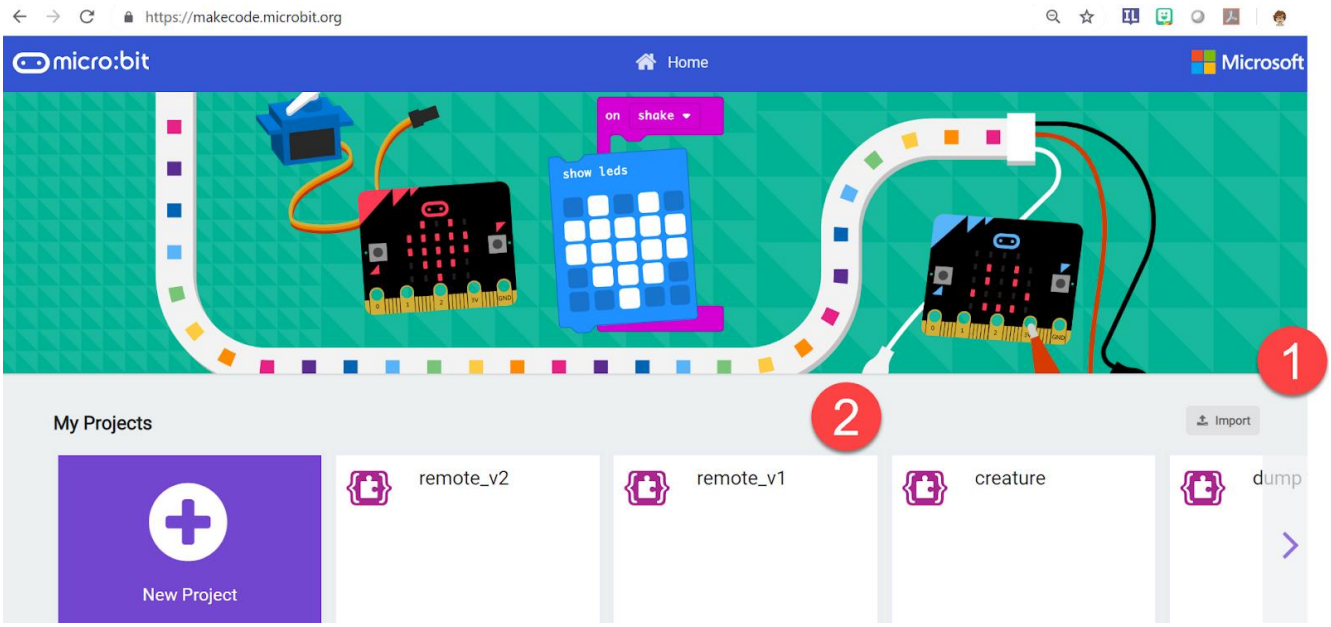
Now, you are ready to control your GiggleBot with another micro:bit!

Turn on both the GiggleBot and the hand-held micro:bit. If you used the example above, when you press the **A** button on the hand-held micro:bit, the GiggleBot will display a rainbow smile and when you press the **B** button on the hand-held micro:bit, the GiggleBot will spin around.

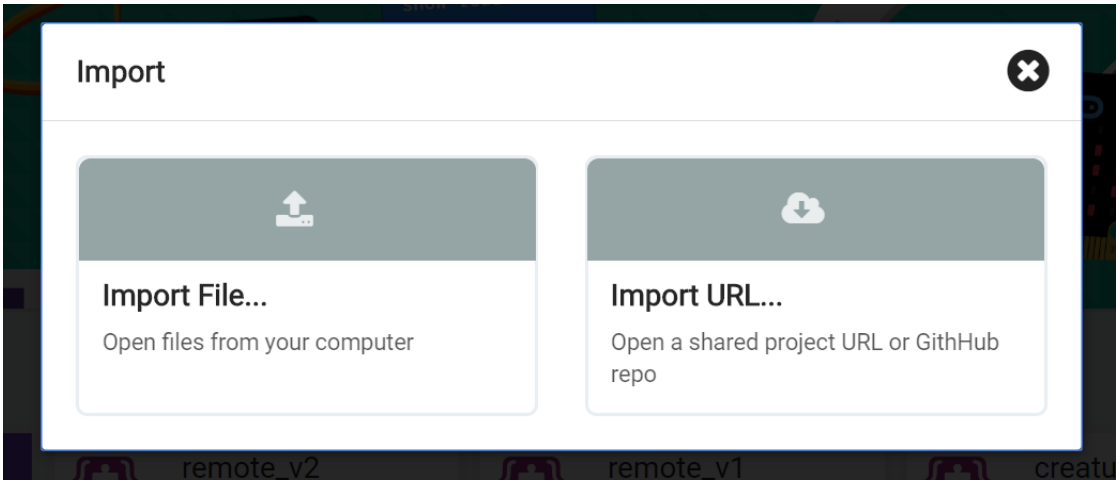
RELOADING A PROGRAM

We will need to go back to the programs we worked on so far.

There are a couple of ways to load a saved program, go back to https://makecode.microbit.org/.

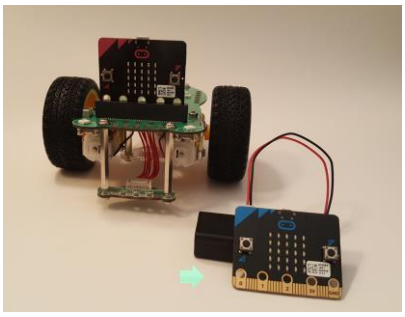


You can import a program by clicking on the *Import* button (1). Then, upload the file from your computer or copy and paste the link for the program.



If your program was made recently, you may be able to just click on the old program (2). Previous programs are listed out under My Projects.

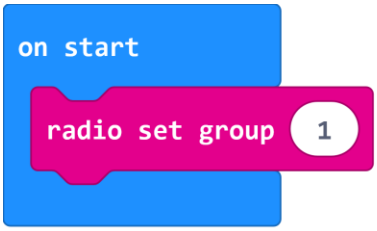
USING MOVEMENT IN THE REMOTE CONTROLLER

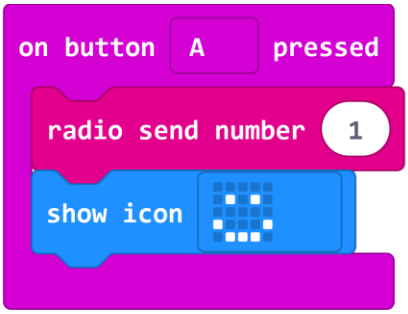


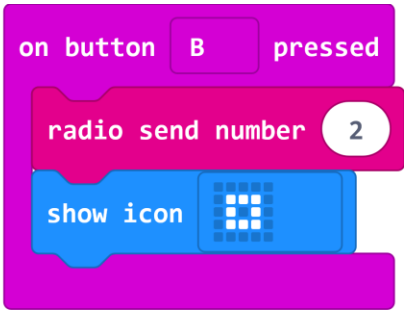
Let's modify these programs to add in more actions. Load or remake the program for the hand-held remote you just created. Set aside the GiggleBot and get the second micro:bit instead.

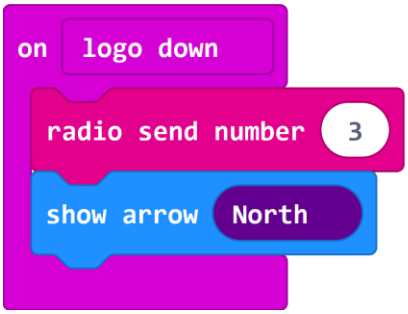
Instead of just using the buttons on the hand-held micro:bit to control the GiggleBot, we can also use the movement of the micro:bit. Think back to mission 2 where we used the position of the GiggleBot to change the lights displayed on the GiggleBot. This program will be similar, however the orientation of the hand-held micro:bit will tell the GiggleBot what to do.

> Add an **on logo down** block in order to send a third command.



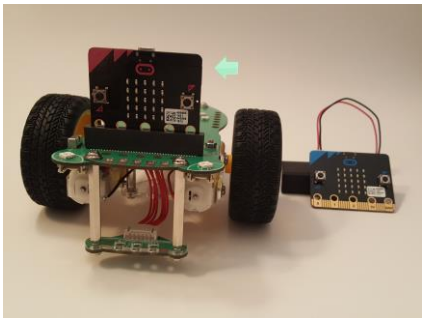




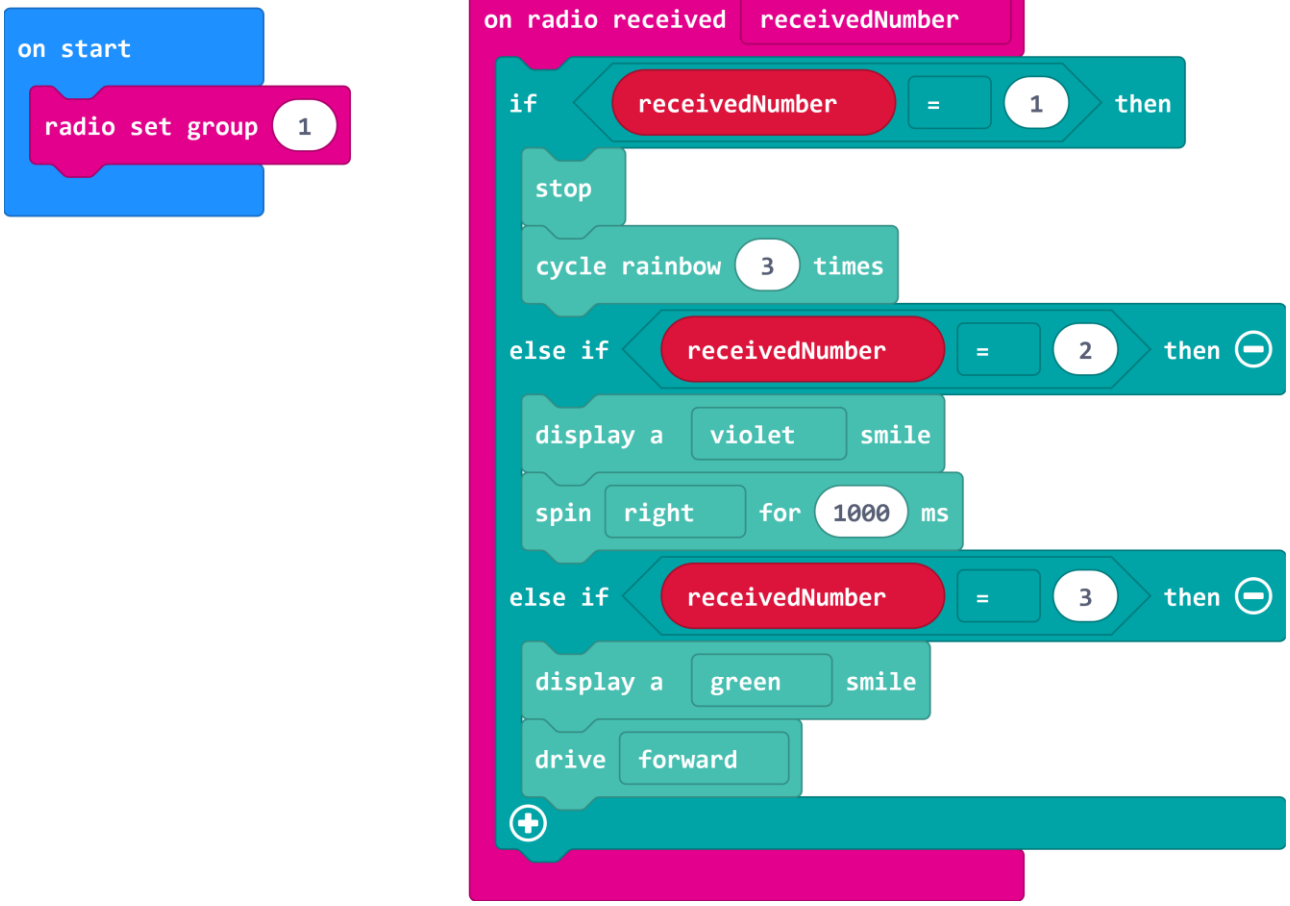


> Download and transfer this program to your additional micro:bit.

RESPONDING TO MOVEMENT ON THE GIGGLEBOT



Next, grab your GiggleBot and reload your program for it so that we can make some modifications to it. We need to add an **else if __ then __** statement for the new movement on the controller. We can also make it more flashy by adding more movement and light commands.



Download and transfer this program to the GiggleBot.

TEST IT OUT!

- Can you make the GiggleBot move forward by moving the additional micro:bit so that the logo is down?
- How do you get the GiggleBot to stop?

> PLAN IT OUT

MICRO:BIT HOLDER

Sketch out a design for the micro:bit holder so that you can control the GiggleBot without holding onto the remote with your hand and fingers. Remember: you will need to have a place for the micro:bit and the battery pack.

- What if you are not able to grip onto something with your fingers? Could you mount the additional micro:bit on a wristband or glove? Could you make an oversized handle for it?
- What if you cannot lift your arms or move your hands? Could you mount the additional micro:bit onto a headband so that the movement of your head would control the GiggleBot?
- What if you could only use your legs or feet, but not your hands and arms? Could you mount the additional micro:bit onto a shoe or sock? Where do you want it to be located so that it is easy to use?

BEHAVIOR

- What movements or actions of the GiggleBot do you want to control?
- Do you want to control the lights? Movement? Both?
- What movements of the extra micro:bit will trigger each GiggleBot motion or action?

Jot down your ideas before you start building and programming.

BUILD YOUR MICRO:BIT HOLDER

Using craft materials or recyclables build a holder for the extra micro:bit so that you do not need to hold it in your hand. Be sure that the micro:bit and its battery pack are secure.

BUILD YOUR PROGRAM

Use what you learned earlier in the mission to program your GiggleBot to be controlled by your extra micro:bit. Remember: you will need to program the GiggleBot as well as the additional micro:bit. These are two separate programs. One program is transferred to the GiggleBot and the other program is transferred to your additional micro:bit.

> ARE YOU STUCK??

Let's build a program together!

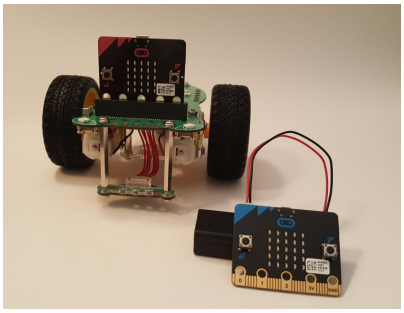
We are going to make a remote that you can use to drive around the GiggleBot just by moving another micro:bit. It will be like having super powers!

The plan:

- Screen up - stop
- Logo up - drive backward
- Logo down- drive forward
- Tilt left - spin left
- Tilt right - spin right

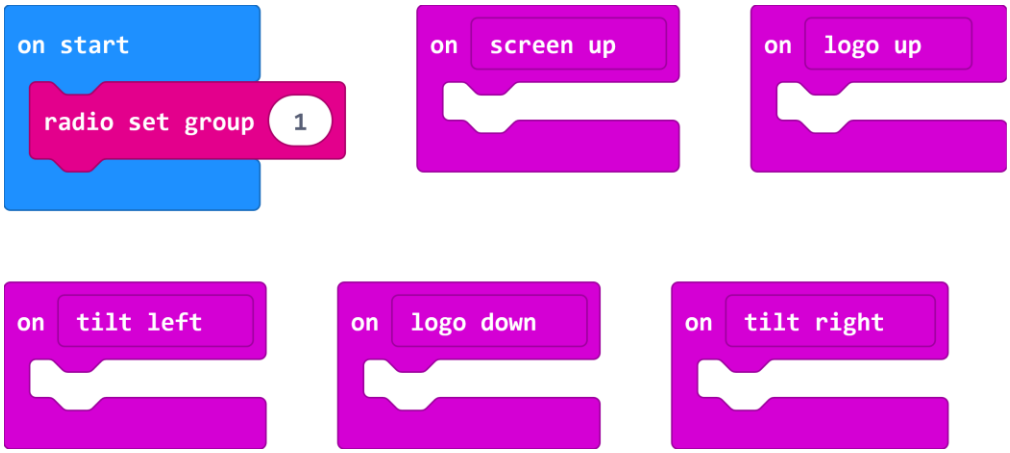
We need to write two programs, just like in the Learn section. One program is for the extra micro:bit and the other is for the GiggleBot.

REMOTE CONTROLLER PROGRAM:

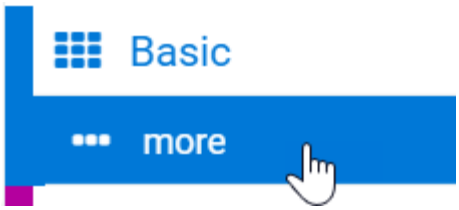


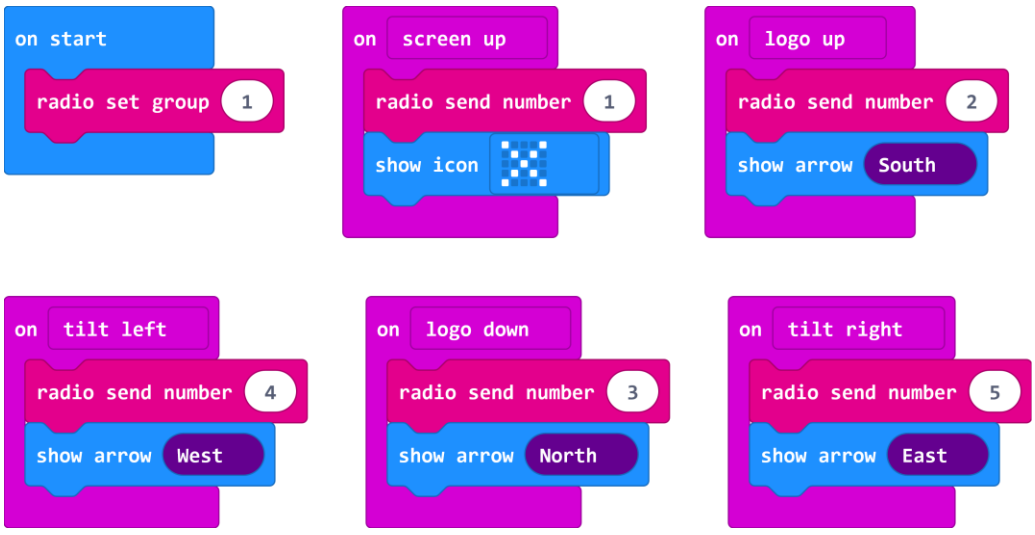
We will begin this program by setting the group and the radio transmission power. Remember, the group needs to match the group you set for the GiggleBot. If there are multiple GiggleBots in the room ensure that each one has a different group number, unless you want to control all of the robots at once!

> Move in an event handler for each movement of the additional micro:bit. These blocks can be found under **Input**.



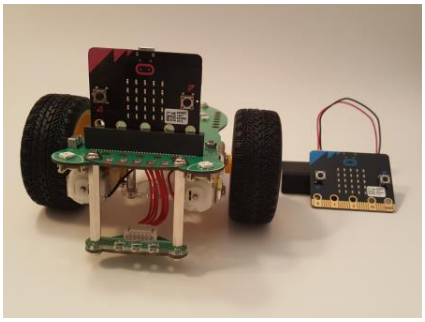
When each movement of the additional micro:bit occurs we will program it to send a number via the radio to the GiggleBot to tell it what to do. Using **show icon** blocks or **show arrow** blocks, program the remote-controller micro:bit to display an image that represents the GiggleBot's movement. The show arrow blocks can be found under **Basic** → **more**.





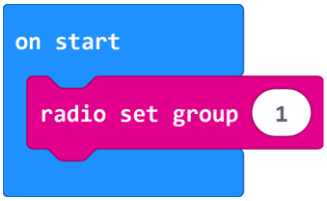
> Download and transfer the program to your additional micro:bit.

GIGGLEBOT PROGRAM:

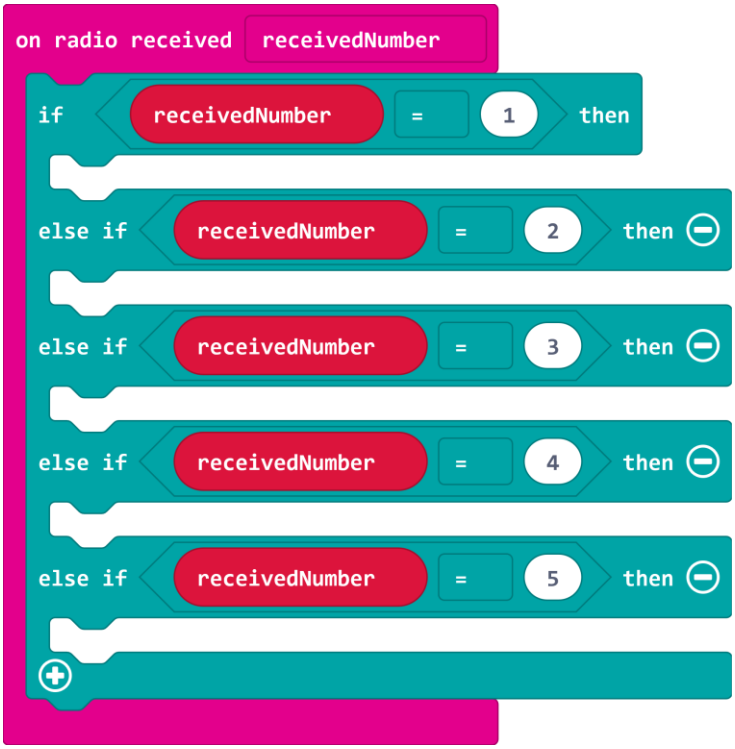


Clear your workspace so that we can program the GiggleBot. Also fetch the GiggleBot itself.

Start this program as we did in the Learn section. First, set the radio group to the same group as the remote. Remember, the micro:bits will only be able to communicate with one another if the radio is set to the same group.



Since we want the GiggleBot to move based on five different movements of the additional micro:bit, we will need to have five **if** statements. After each **if __ then __** statement (or **else if __ then __**), we will test for the number that will be sent by the micro:bit remote controller using a the **receivedNumber** variable block. Our **on radio received receivedNumber** will now look like the following:



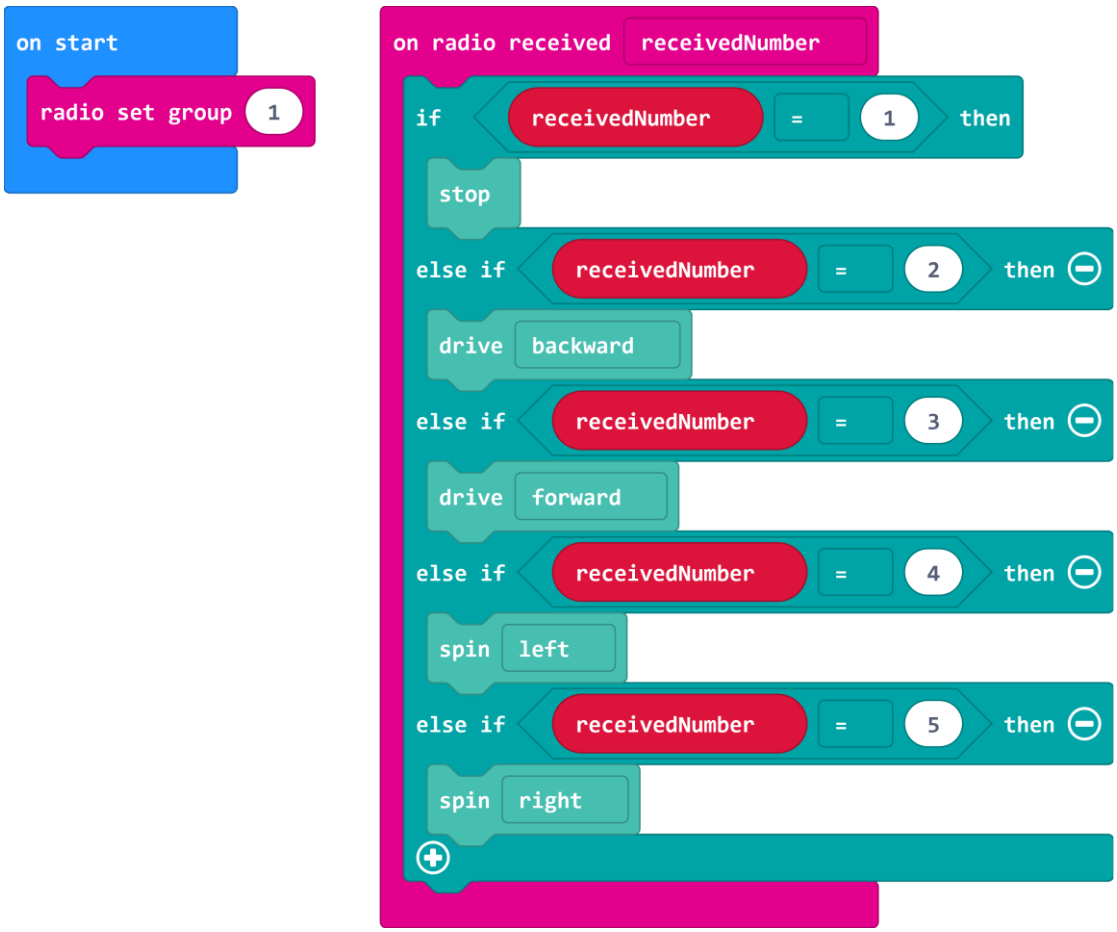
Each movement of the GiggleBot needs to match what we wrote in the remote controller program. For example, in the remote controller program:

1. We programmed the micro:bit to send the number **1** to tell the GiggleBot to **stop**.
2. We programmed the micro:bit to send the number **2** to tell the GiggleBot to **drive backward**.
3. We programmed the micro:bit to send the number **3** to tell the GiggleBot to **drive forward**.

If these do not match up, your controller will not work as you planned.

Add the appropriate movement blocks to each of the **then __** statements.

Here is what the whole GiggleBot program looks like:



- Download and transfer the program to your GiggleBot.
- Turn on the GiggleBot.
- Power the additional micro:bit.
- Test out your programs.

> TRY IT OUT

- Were you able to control the GiggleBot 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 iterate.

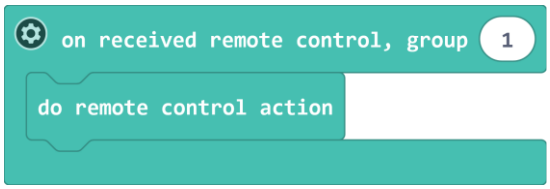
As you modify and revise your program, save each new program with a version number – you never know when you might want to take another look at an older version (for example: **Remote_V1** where "**V**" stands for version).

EXTENSION

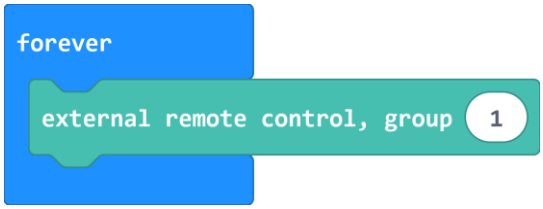
READY-MADE BLOCKS.

There are GiggleBot-specific remote control blocks under **Remote**.

Program the GiggleBot to receive signals from the additional micro:bit using the premade blocks. You only need two blocks!! Download and transfer this program to your GiggleBot.



Next, program the additional micro:bit to act as a remote for the GiggleBot. Download and transfer this program to the additional micro:bit.

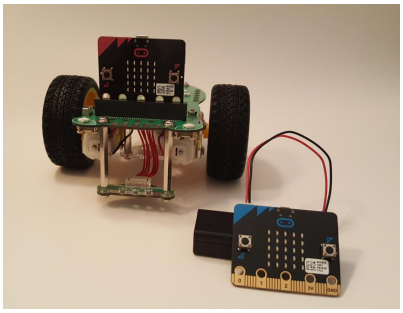


- Turn on both the GiggleBot and the additional micro:bit.
- Use the movement of the additional micro:bit to control the GiggleBot.

How does this differ from the programs you made previously to control the GiggleBot remotely? What can you do now that you could not do before?

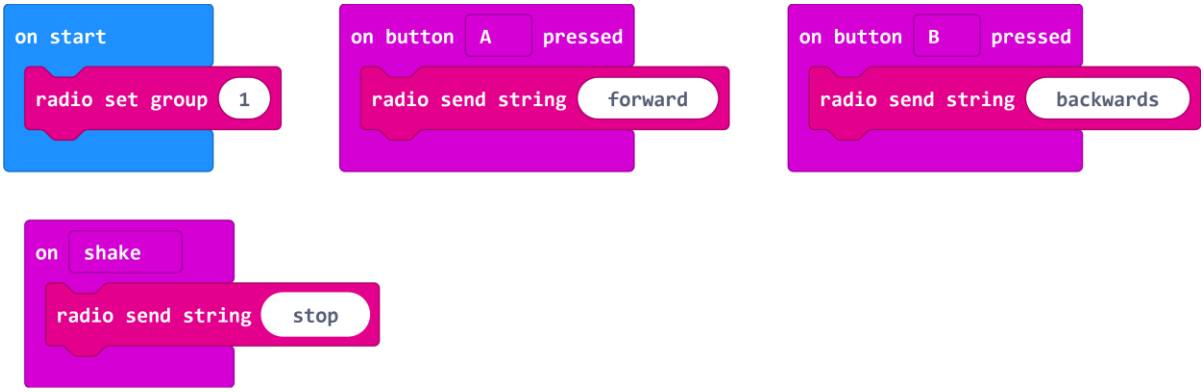
RADIO MESSAGES WITH WORDS

It is possible to pass word messages instead of being limited to numbers as per the following code.



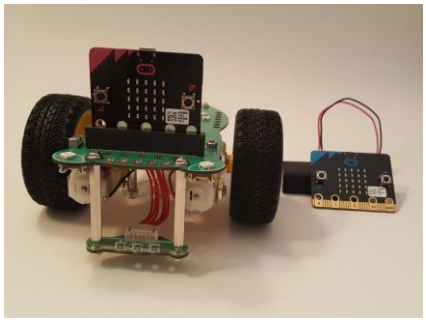
SENDING FROM THE REMOTE CONTROL:

- What advantages do you see with using strings intead of numbers?
- What disadvantages can you think of?



RECEIVING ON THE GIGGLEBOT:

Notice the quotes around the code words. To get those quotes you will need to use a block from **Advanced > Text**.



It is usually easier for a computer to deal with numbers and for a human to deal with words.