

SECURBOT

MISSION 10

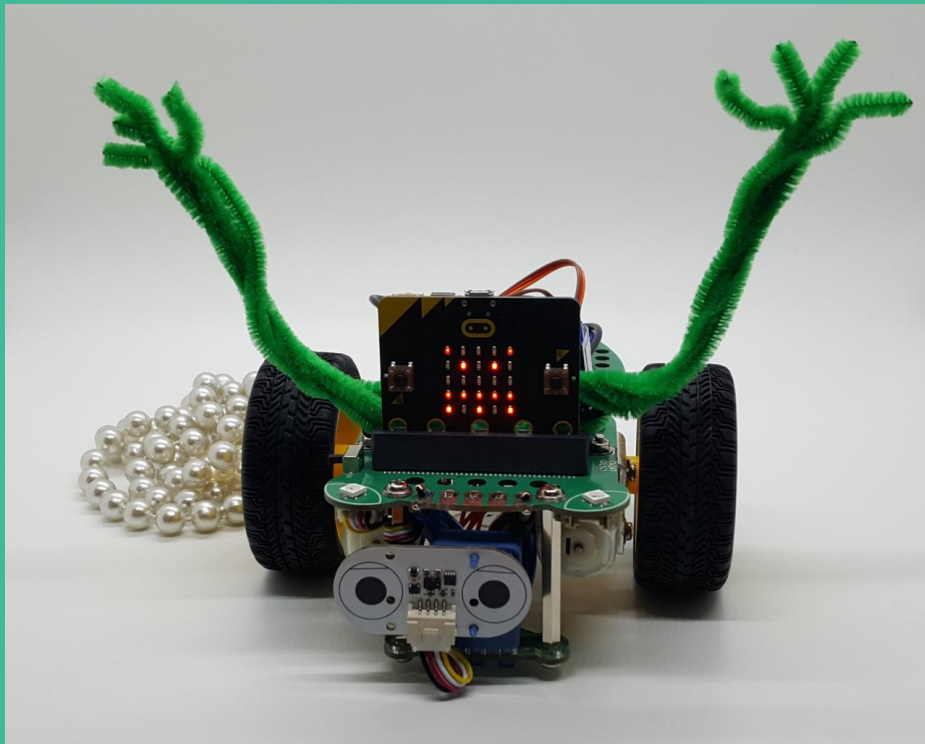


TABLE OF CONTENTS

• SECURE-BOT	1
Your Role: Security Guard.....	1
Your Task: Defend a Treasure.....	1
Considerations.....	1
Materials.....	1
 • LEARN: RADIO.....	2
Program the GiggleBot.....	2
Program the Additional Micro:Bit.....	6
Test It Out.....	7
 • PLAN IT OUT.....	8
Security Robot Features.....	8
Programming.....	8
Build Your Robot Guard.....	10
Build Your Program.....	11
 • ARE YOU STUCK??	12
Work Plan.....	13
Build Your Security Robot	14
Program the GiggleBot.....	14
Program the Additional Micro:Bit.....	24
 • TRY IT OUT!	32
Iterate	32
Challenge	33

> SECURE-BOT

YOUR ROLE: SECURITY GUARD

i A **security guard** is a person employed to protect something specific from a variety of hazards.

YOUR TASK: DEFEND A TREASURE

You have been put in charge of keeping a treasure safe. You must not let anyone get to the treasure. To do this you will create a security robot. If the location of the treasure is compromised, the robot will send a signal to another micro:bit to warn of the intrusion. You will also utilize your skills and materials from previous missions to add other methods for deterring the intruder



CONSIDERATIONS

What will your GiggleBot be in charge of protecting? Your room? A slice of cake in the refrigerator? Your favorite toy hidden in a cabinet? Your clothes in the closet?

MATERIALS

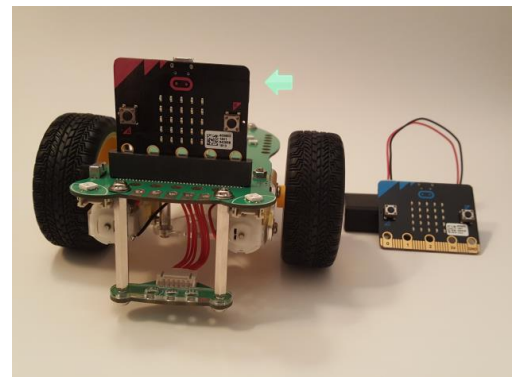
- > GiggleBot and good batteries
- > Micro:bit and provided cable
- > Laptop / computer
- > Distance sensor and sensor mount
- > Servo(s) and some means to secure them
- > Pipe cleaners
- > Scissors
- > Alligator clips
- > Speaker
- > Additional micro:bit
- > Craft supplies or recycleables such as construction paper, cardboard, and tape

> LEARN: RADIO

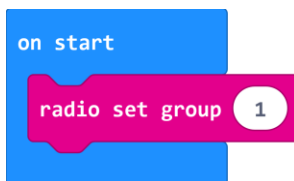
In mission 8, we learned how to use an additional micro:bit to control the GiggleBot using radio signals. In this mission, we will be sending signals from the GiggleBot to the additional micro:bit as well as messages from the additional micro:bit back to the GiggleBot. This will be like an alert or notification on your phone, except it will be on a micro:bit. We will need to program both the GiggleBot and the additional micro:bit. Look for the  icon to identify code that goes on the GiggleBot and for the  icon for code that goes onto the additional micro:bit.

PROGRAM THE GIGGLEBOT


We are going to create a program for the GiggleBot to send a message whenever the light level in the room has changed. For this program we will use the light sensor on the micro:bit that is attached to the GiggleBot. There are also light sensors on the GiggleBot, but we will not use them for now. However, you may want to use them in your own security robot program.



Let's program the GiggleBot to detect if the light level has changed from dark to light. If the light level changes, it will send a message to the additional micro:bit.



- > Start a new project, and name the project "guardian".
- > We need to set the radio group. Put a **radio set group** __ block (found under **Radio**) inside of a **on start** block to set the radio group for your GiggleBot.

 Remember, if there are multiple groups of GiggleBots and additional micro:bits being programmed, each set will need a different radio group number (unless you want several GiggleBots to communicate with one another).

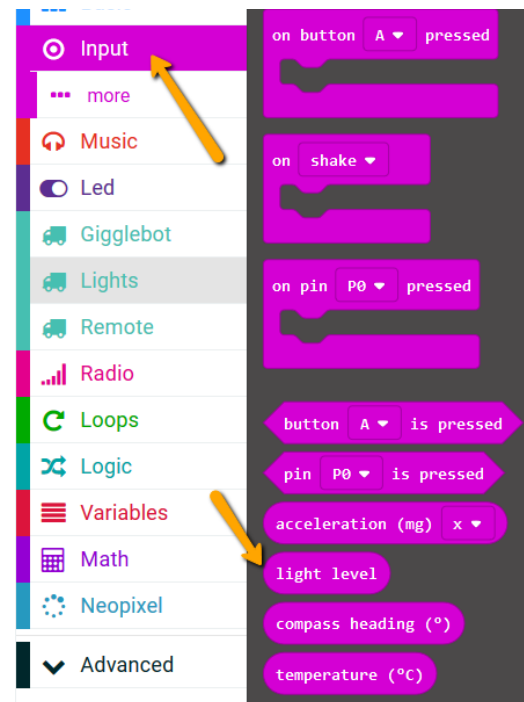


DISPLAY LIGHT LEVEL



Now, move a **forever** block onto your workspace. Then, add a **show number** ___ block to the **forever** loop. Instead of showing just one number, move a **light level** block into the **forever** loop.

You can find the **light level** block under **Input**:



The GiggleBot will display the light value as seen by its own micro:bit. Play with the light level around the GiggleBot: use a blanket to create a shadow, or bring it close to a light source. Verify that the displayed number does change accordingly.

REACT TO CHANGE IN LIGHT LEVEL

Now, we will program the GiggleBot to react to a change in light level.

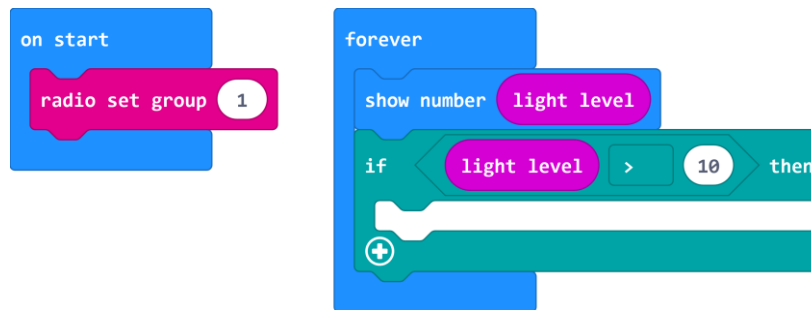
- > Add an **if __ then __** block to the **forever** loop. This block can be found under **Logic**.



Attach a **0 < 0** block to the **if** part of the **if __ then __** block, thus replacing the **true** word. This block can also be found under **Logic**.



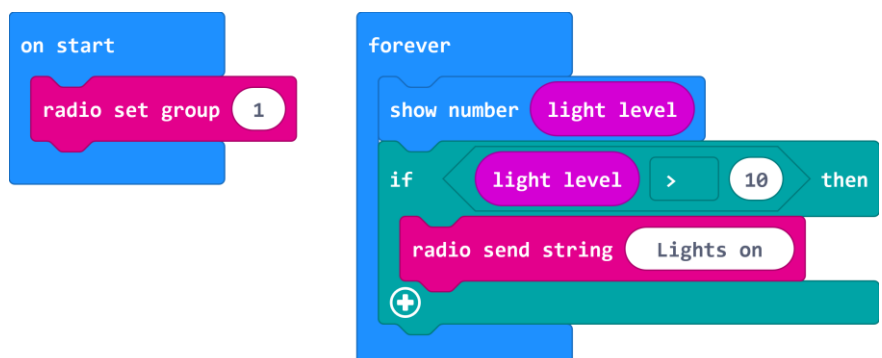
- > In the comparison block, add a **light level** block, replacing the first **0**.
- > Change the comparison sign and value to your desired light level. The micro:bit light sensor reads 0 as complete darkness and 255 as maximum brightness.



Now, you need to decide what message you will send to the additional micro:bit when the light level changes.

- > Attach a **radio send string __** block to the **then __** portion of the **if __ then __** block. Inside of the quotation marks write a message to appear on the additional micro:bit when the light levels around the GiggleBot change.

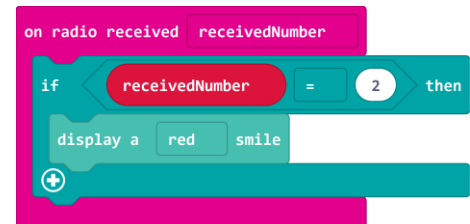
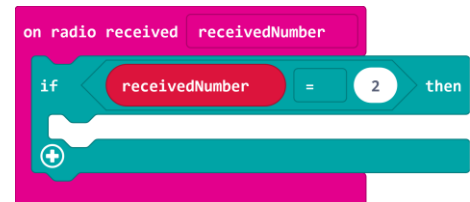
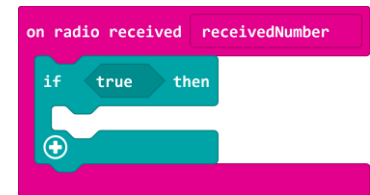
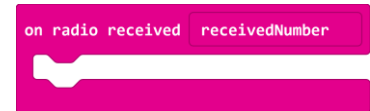
In this example, the additional micro:bit will display the text "Lights on" when the light level around the GiggleBot changes from dark to light.



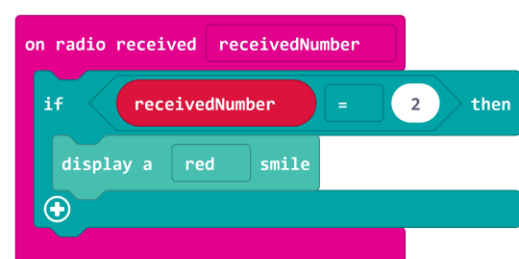
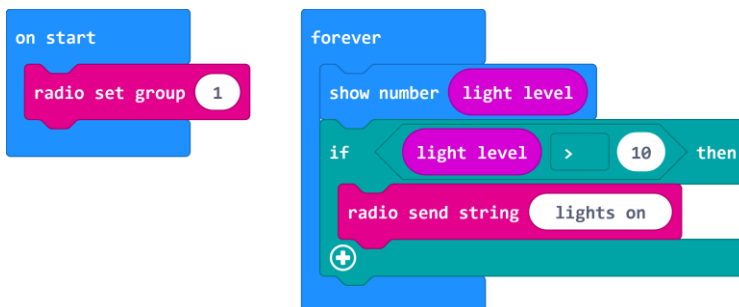
TAKE ORDERS FROM ADDITIONAL MICRO:BIT

Last, we will add on code so that the additional micro:bit can communicate with the GiggleBot as well. For this program the additional micro:bit is going to send a **2** to the GiggleBot when the light level changes from dark to light. You can change this number, but it will need to match the value sent by the additional micro:bit.

- > Move an **on radio received receivedNumber** block to the workspace. This block can be found under **Radio**.
- > Inside this block we will tell the GiggleBot what to do when it receives a number from the additional micro:bit
Put an **if __ then __** block inside the **on radio received receivedNumber** block.
- > Attach a **0 = 0** block to the **if __** portion of the **if __ then __** block.
- > Inside of this block place a **receivedNumber** block (found under **Variables**) in the first place for a value. **receivedNumber** is a variable that will change value based on what number is sent by the additional micro:bit.
- > Now, we will tell the GiggleBot what to do when it receives a **2** from the additional micro:bit. Since turning on lights may mean an intruder has entered the room, let's tell the GiggleBot to display a **red** smile. The blocks to control the lights on the GiggleBot can be found under **Lights**.



Download and transfer this program to the GiggleBot.

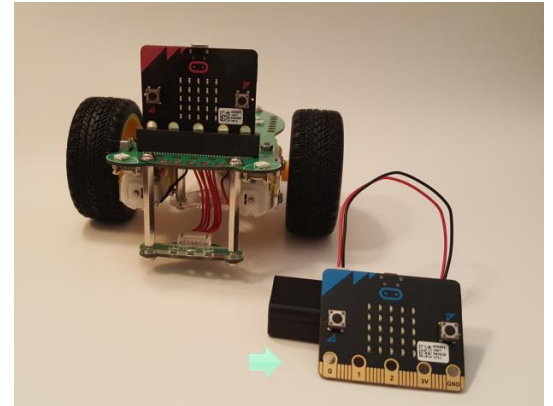


PROGRAM THE ADDITIONAL MICRO:BIT

Now we need to program the additional micro:bit to receive messages from the GiggleBot.

- > Start a new project in Makecode. Name it "alarm".

First, we need to set the radio group to match that of the GiggleBot. Remember, if there are other people programming micro:bits and GiggleBots nearby, each person/group needs to use a different radio group.



on start

radio set group 1

> Put a **radio set group 1** block (found under **Radio**) inside of a **on start** block to set the radio group for your GiggleBot. Change the group if needed.

RESPONDING TO MESSAGES RECEIVED FROM GIGGLEBOT

on radio received **receivedString**

Next, move an **on radio received receivedString** block onto your workspace. Inside this block we will tell the additional micro:bit what to do when it receives a message from the GiggleBot.

For this program we will tell the additional micro:bit to display the message sent by the GiggleBot.

on radio received **receivedString**

show string Hello!

> Move a **show string** ____ block inside of the **on radio received receivedString** block.

Since we want the additional micro:bit to display whatever message is sent by the GiggleBot, we will add a variable block to the portion where the message is written.

on radio received **receivedString**

show string **receivedString**

> Move a **receivedString** (found under **Variables**) to the portion of the **show string** block, replacing the default "Hello!" string.

SENDING MESSAGES TO GIGGLEBOT

Last, we will add to the program to allow the additional micro:bit to communicate back to the GiggleBot.



> Add an **on button A pressed** block to your workspace.

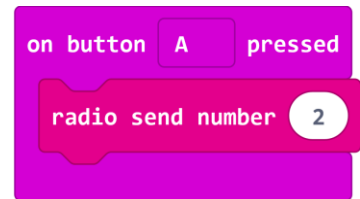
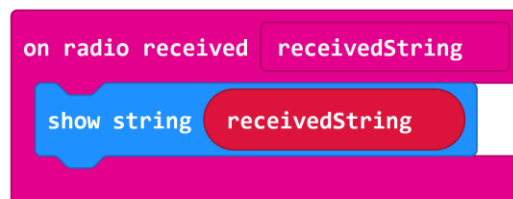
This will allow you to communicate back to the GiggleBot by pressing button **A** on the additional micro:bit.



> Inside of this block attach a **radio send 0** block.

> Change the value inside this block to 2 since that is the value we programmed the GiggleBot to receive.

Download and transfer this project to the additional micro:bit.



TEST IT OUT

Turn off the lights in the room and turn on the GiggleBot and the additional micro:bit.

- > What happens when you turn on the lights?
- > Did the additional micro:bit display a message that the lights were on?
- > What happens when you press the **A** button on the additional micro:bit? Did the GiggleBot display a red smile?

Now that you have learned how to program two micro:bits to communicate with one another, you are ready to create a robotic security guard.!

> PLAN IT OUT

SECURITY ROBOT FEATURES

How will you utilize all of the materials from the missions in your security guard robot? Make a list of the materials and jot down your ideas next to each item.

- 1 Distance sensor
 - 2 Servo(s)
 - 3 Speaker
 - 4 Light sensor (built in)
- > What movements or changes in the environment will the robot be able to detect and react to?
 - Change in light level?
 - An object getting close to it?
 - Being turned upside down?
 - > What will your robot look like?
 - Will it be a dragon with wings that flap?
 - A monster with a large mouth full of teeth?
 - A spy car equipped with defense mechanisms?

PROGRAMMING

Think about what you want the GiggleBot to react to. Write down what the GiggleBot will do in each of the situations. This will help you to program the GiggleBot's reactions as well as the messages it sends to the other micro:bit.

Create a chart like the one next page with everything that you want the GiggleBot to do and send alerts about.

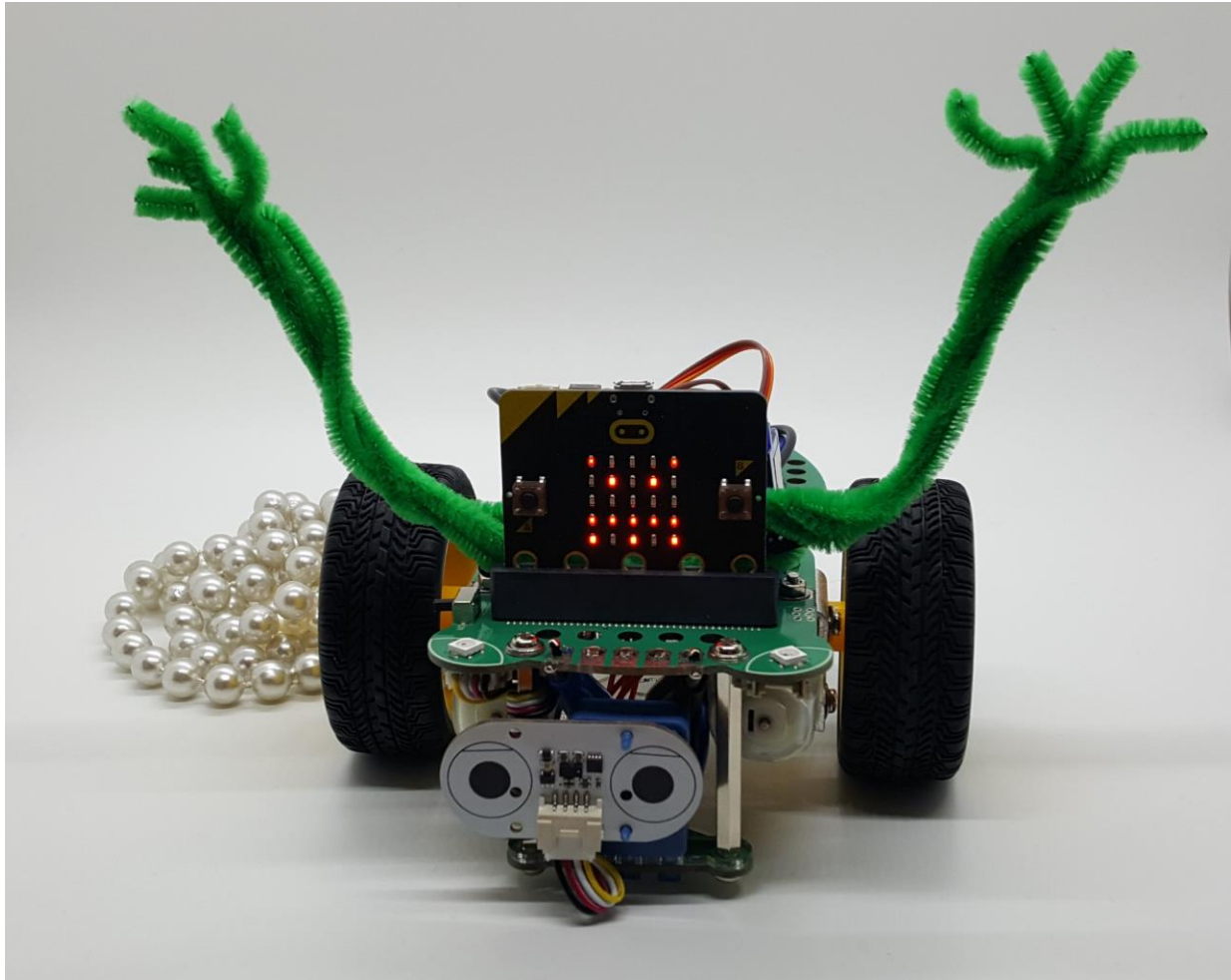
You do not need to incorporate all of the actions in the chart below, but you need to utilize all of the sensors :

- > light sensor
- > distance sensor
- > gyroscope (detects a change in orientation such as being tilted or turned upside down)
- > speaker
- > onboard LEDs

ACTION	GIGGLEBOT REACTION	MESSAGE TO MICRO:BIT	RADIO TRANSMISSION (fill in the number you plan to use for each)
Room lights turned on			radio send number <input type="text"/>
Object detected (distance sensor)			radio send number <input type="text"/>
GiggleBot moved (gyroscope)			radio send number <input type="text"/>
GiggleBot upside down (gyroscope)			radio send number <input type="text"/>
	Speaker sound		radio send number <input type="text"/>
			radio send number <input type="text"/>
			radio send number <input type="text"/>
			radio send number <input type="text"/>
			radio send number <input type="text"/>
			radio send number <input type="text"/>

BUILD YOUR ROBOT GUARD

Attach the distance sensor, speaker, and servo(s) to your GiggleBot. Add any additional decorations to make your robot look like the desired type of security guard.

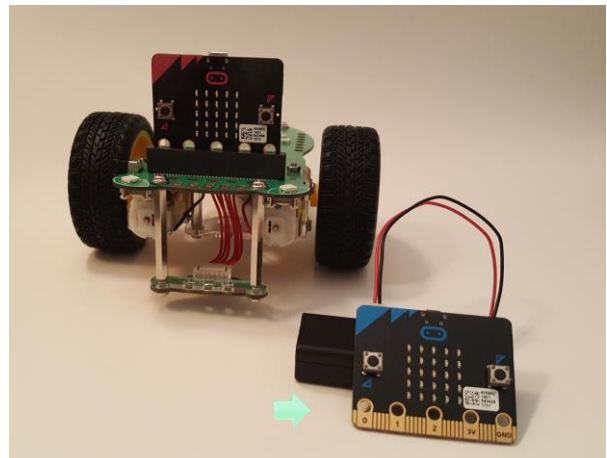
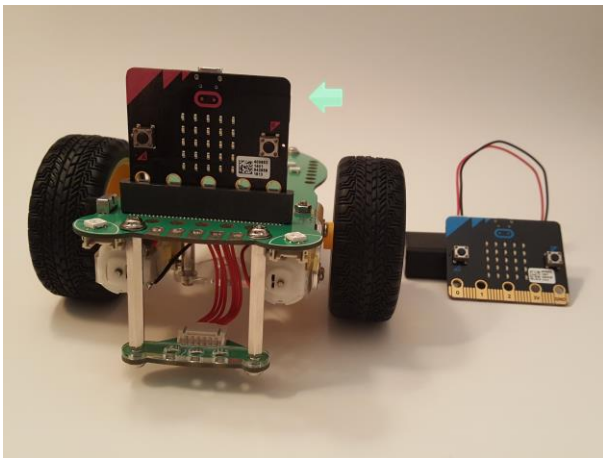


BUILD YOUR PROGRAM

Use your plan from the chart above to program your GiggleBot security guard. Remember that you will need to program the micro:bit on the Gigglebot as well as the additional micro:bit. These are two separate programs.

One program is transferred to the GiggleBot:

The other program is transferred to your additional micro:bit:



Also be sure to plug in the distance sensor to one of the I2C ports and to connect the speaker using alligator clips.

Look back to previous missions if you need help connecting the distance sensor and speaker.

> ARE YOU STUCK??


Let's build together! As this is the most complex project so far, it's been split into smaller parts. Go through each step, one by one. First, attempt it on your own and if you need help, check the following pages.

Work Plan	13
Build Your Security Robot	14
Program the GiggleBot.....	14
Reacting to Changes in its Environment.....	15
A: Turning on the Lights	15
B: Detection of an Object (distance sensor).....	16
C: Detection of GiggleBot Movement ("on shake").....	17
D: Reacting to Being Turned Upside Down (On Logo Down)	18
Receiving Signals from the Additional micro:bit	20
E: Alarm Sounding	20
F: Driving Forward in Attack Mode.....	21
G: Stop the GiggleBot	22
Final GiggleBot Code	23
Program the Additional Micro:Bit	24
Reacting to Signals from the GiggleBot	24
A: Turning on the Lights	25
B: Detection of an Object (distance sensor).....	26
C: Detection of GiggleBot Movement ("on shake").....	27
D: Reacting to Being Turned Upside Down	28
Sending Signals to the GiggleBot.....	29
E: Alarm Souding	29
F: Driving Forward in Attack Mode.....	29
G: Stop the GiggleBot	30
Final Additional Micro:bit Code.....	31

First, we need to decide what the GiggleBot will do in each of the situations below. Here are some ideas for each action and reaction. We will use this plan to build a robotic security guard.

WORK PLAN

	ACTION	GIGGLEBOT REACTION	MESSAGE TO MICRO:BIT	RADIO TRANSMISSION (fill in the number you plan to use for each)
A	Room lights turn on	LED smile turns red	"Lights!"	radio send number 1 
B	Object detected (distance sensor)	Blink orange /white smile	"INTRUDER!"	radio send number 2 
C	GiggleBot moved	LED Sad Face	"Moving!"	radio send number 3 
D	GiggleBot turned upside down	Servos move arms up and down	LED ANGRY FACE	radio send number 4 
E	Additional micro:bit: Button A pressed	Speaker sound		radio send number 5 
F	Additional micro:bit: Button B pressed	GiggleBot drives forward quickly		radio send number 6 
G	Additional micro:bit – Button A + B pressed	GiggleBot stops		radio send number 7 

 For a feature to function fully, you will need to code the feature in both the GiggleBot and the additional micro:bit.

BUILD YOUR SECURITY ROBOT

Gather all of your materials for your GiggleBot security robot. You will need:

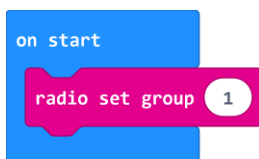
- > GiggleBot + micro:bit
- > Additional micro:bit + battery pack
- > Distance sensor and sensor mount
- > Speaker
- > Alligator clips
- > 2 servos
- > Pipe cleaners to secure the servos and speaker
- > Supplies for decoration (if desired)

Attach your distance sensor, speaker, and servos to your GiggleBot. In the example program below we will program the servos to move arms made of craft supplies. If you are not sure how to connect something, look back to previous missions for step-by-step directions.

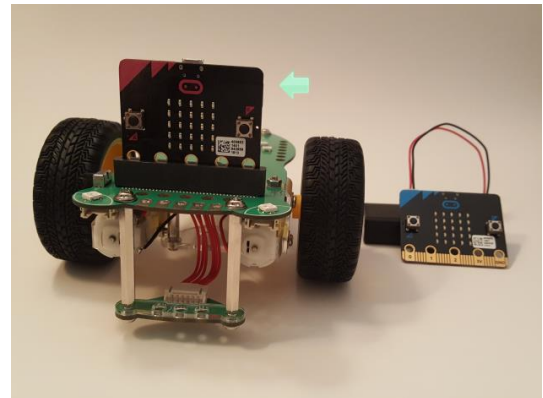
Once your GiggleBot security guard has all of its extra components attached, it is time to program!

PROGRAM THE GIGGLEBOT

We will write the GiggleBot program. Give it a name so you will be able to identify and reload it.



First, we will set the radio group.



 Remember, if there are multiple groups of GiggleBots and additional micro:bits being programmed, each set will need a different radio group number.

- > Put a **radio set group** __ block (found under **Radio**) inside of a **on start** block to set the radio group for your GiggleBot.

Next, we will program the GiggleBot's reactions to different actions or changes. Each of the following sections is to be added to the above **on start** sequence. Your program will grow and grow!



REACTING TO CHANGES IN ITS ENVIRONMENT

Steps A to D deal with the GiggleBot reacting to changes in its environment that signal the presence of an intruder. Messages are going to be sent to the second microbit as alert messages.



A: TURNING ON THE LIGHTS

The first condition we are looking for is a change in the light level. If the lights are turned on, the LED smile will turn red and the GiggleBot will send a message to the additional micro:bit. Add this code to what you already have on your workspace.



> Move a **forever** block onto your workspace. Place an **if __ then __** block inside of the **forever** block.

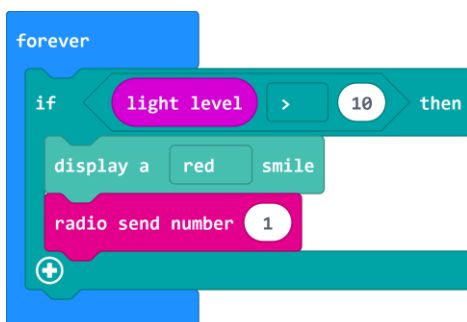
> Attach a **0 < 0** block (found under **Logic**) to the **if __** part of the **if __ then __** block.

> In the comparison block, add a **light level** (found under **Input**) block to the first place for a value. Change the comparison sign and value to your desired light level.

i The micro:bit light sensor reads 0 in complete darkness and 255 as maximum brightness.

We will now attach blocks to the **then __** portion of the **if __ then __** block to tell the GiggleBot what to do when this condition is met. Based on the plan above, we need to :

- 1 connect a **display red smile** block.
- 2 send a message to the additional micro:bit.



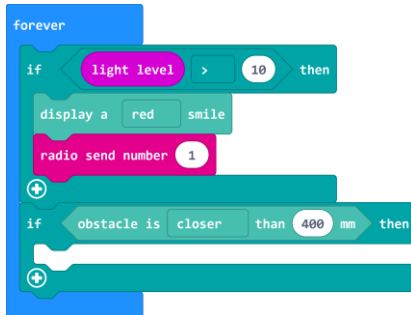
We will send the message by sending a number to the additional micro:bit. Later, we will tell the additional micro:bit what to do when it receives each of the numbers. Connect a **radio send number __** block below the **display red smile** block. Change this number to **1**. Now your GiggleBot will react to the lights being turned on!

Transfer your code to the GiggleBot and test it out!



B: DETECTION OF AN OBJECT (DISTANCE SENSOR)

We now need another **if __ then __** block to program the GiggleBot's reaction to detecting an object using the distance sensor. We will follow the same basic process that we used for the light sensor to program the GiggleBot to react to a change in the distance sensor reading. Add the following to what you already have in your workspace.

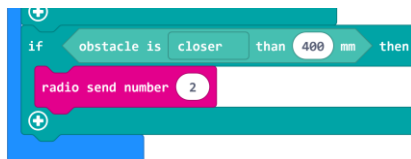


> Place this **if __ then __** block underneath the first **if __ then __** block for the light sensor readings.

Attach an **obstacle is closer than __ mm** block (found under **GiggleBot**) to the **if __** portion of the new **if __ then __** block. Determine what the "normal" value for the distance should be below.

How close should something be able to get to the GiggleBot before it reacts? Your value may be different than the one in the example.

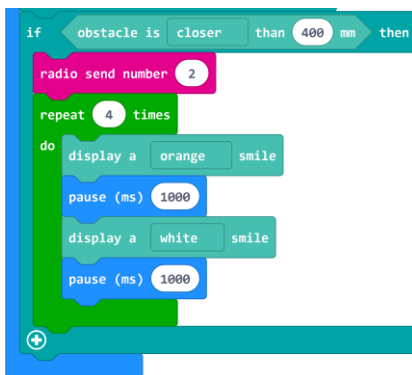
In our plan, the GiggleBot's reaction to a object getting close is to blink an orange smile and send a signal to the additional micro:bit.



> Insert a **radio send number __** block in the **then __** part of the **if __ then __** block.

> Change the number to **2**, per our plan.

Since we want the GiggleBot to blink an orange and white smile, we will need to use a loop.



> Attach a **repeat __ times** block to the **then __** - portion of the **if __ then __** block. You can change the number of times that the orange smile blinks by changing the number in this block.

> Inside of the repeat loop, connect a **display red smile** block and change *red* to *orange*.

> After, this block add a **pause 100 ms** block. Change the value of this block to tell the GiggleBot how long to wait before changing to the next color smile.

> Add another **display red smile** block and change this one to *white*.

> Add one more **pause 100 ms** as well and change the value to match the first one.

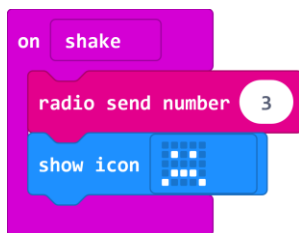
Transfer your code and test your program to ensure the reaction to an obstacle works as intended!

C: DETECTION OF GIGGLEBOT MOVEMENT (“ON SHAKE”)

Next we will program the robot to react to being moved by using the using the accelerometer on the GiggleBot's micro:bit. We add to the code we've already done.

> Move an **on shake** event block to your workspace. This block is found under **Input**.

Inside the **on shake** event block, connect blocks to tell the GiggleBot what to do if it is shaken. Using the plan above, the GiggleBot should first send a radio message with the number **3** and then display a sad face on the GiggleBot micro:bit.



- > Connect a **send radio number** block and change the value to **3**.
- > Connect a **show icon** block to the **on shake** event block.
- > Change the icon to a sad face.

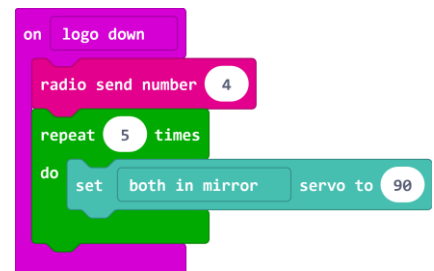
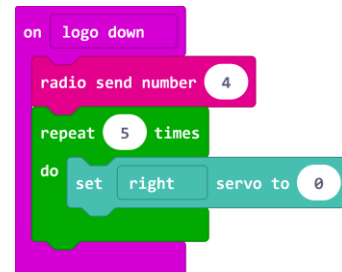
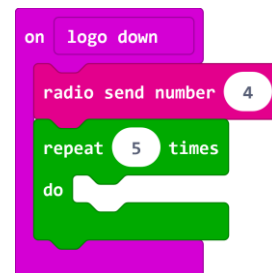
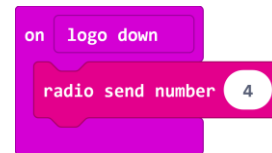
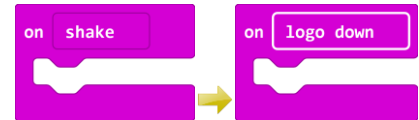
The GiggleBot needs to send a signal to the additional micro:bit that it has been picked up or moved. We will do this by sending another number to the additional micro:bit. Later, we will tell the additional micro:bit what to do when it receives each of the numbers

Your GiggleBot will now react if it is shaken! Transfer your code and test it out!

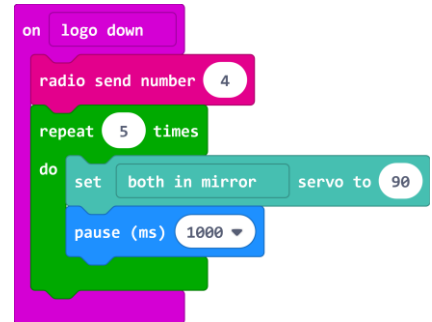
D: REACTING TO BEING TURNED UPSIDE DOWN (ON LOGO DOWN)

According to our plan, the GiggleBot's reaction to being upside down is waving its arms. The GiggleBot will wave the servo-controlled arms 5 times. You can control each servo individually, in mirror, or in synchro.

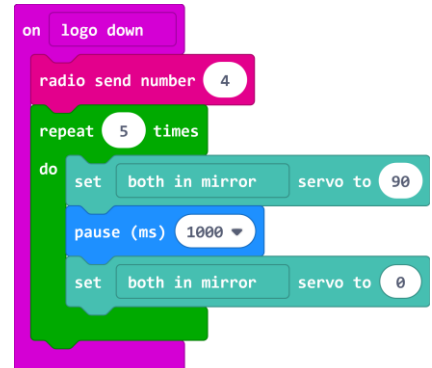
- > Move another **on shake** block to the workspace.
- > Using the drop-down menu change this to **on logo down**
- > The GiggleBot needs to send a signal to the additional micro:bit. We will do this by sending the number **4**. Later, we will tell the additional micro:bit what to do when it receives each of the numbers.
- > Connect a **repeat 4 times** block to the **on logo down** block.
- > Change the repeat value to your desired number of repetitions.
- > Inside of the **repeat** block, connect a **set right servo to 0** block to the **on logo down** block
- > Change **right** to **both in mirror**. Then, change the value to tell the Gigglebot to raise its arms



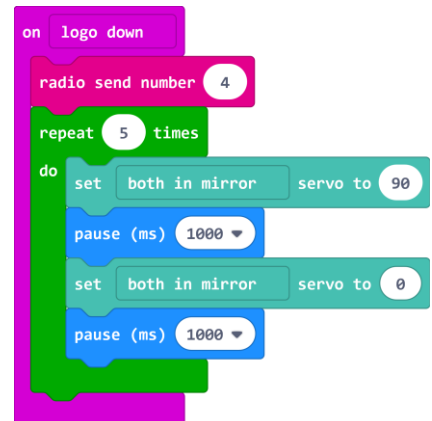
- > Add a **pause 100 ms** block next.
- > Change the value of the pause to tell the GiggleBot how long to wait before moving the servos again. The lower the number, the faster the servos will move back and forth.



- > Now, connect another **set right servo to 0** block.
- > Use the drop-down menu to change **right** to **both in mirror**. You may prefer **both in synchro**.



- > Add one last **pause 100 ms** block.
- > Change this value to tell the GiggleBot how long to wait before moving the servos again



Your GiggleBot will now wave its arms if it is flipped over!

Transfer your code and test it out!



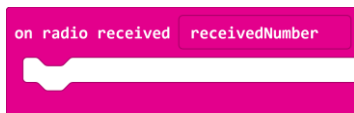
RECEIVING SIGNALS FROM THE ADDITIONAL MICRO:BIT

We are done with sending messages to the additional micro:bit to let it know what is happening to our guardian. Now we will program the GiggleBot to react to signals sent by the additional micro:bit. That way, we can remote control the GiggleBot and better defend our treasure.

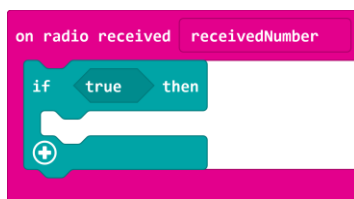


E: ALARM SOUNDING

According to our plan, if button **A** is pressed on the additional micro:bit, the speaker attached to the GiggleBot will make a sound.

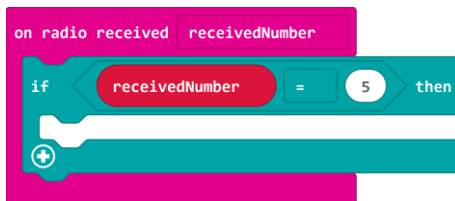


> Move an **on radio received receivedNumber** block into your workspace. Inside of this block we will tell the GiggleBot what to do whenever the additional micro:bit sends a number



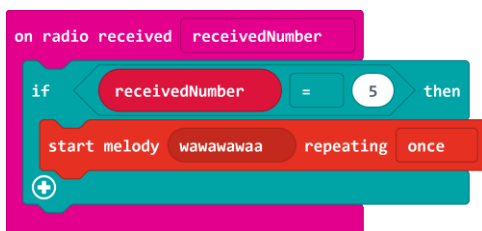
> Connect an **if true then** inside of the **on radio received** event block.

According to our plan, the GiggleBot will react to button presses on the additional micro:bit via a number sent by the additional micro:bit. The number 5 corresponded to the GiggleBot playing a sound through the speaker.



> Attach a **0 = 0** block (found under **Logic**) to the **if** portion of the **if then** block, replacing the **true** section.

> Inside of this block place a **receivedNumber** block (found under **Variables**) over the first **0**. Change the second value in the **0 = 0** block to **5** since that is what was assigned to this action in the plan above.



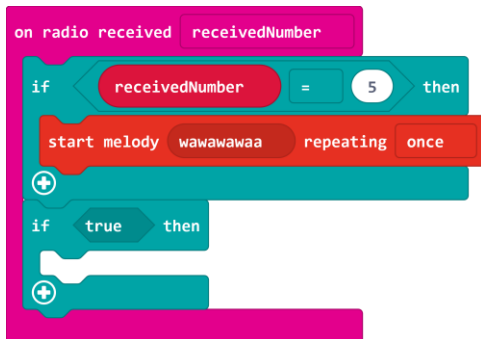
There are lots of options for what sound or music you can play when the GiggleBot receives this signal from the additional micro:bit. Click on **Music** to see all of the options.

Select the **start melody dadadum repeating once** block and then choose which melody to play using the drop-down menu.

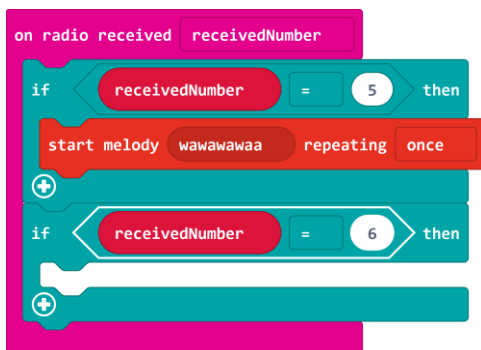


F: DRIVING FORWARD IN ATTACK MODE

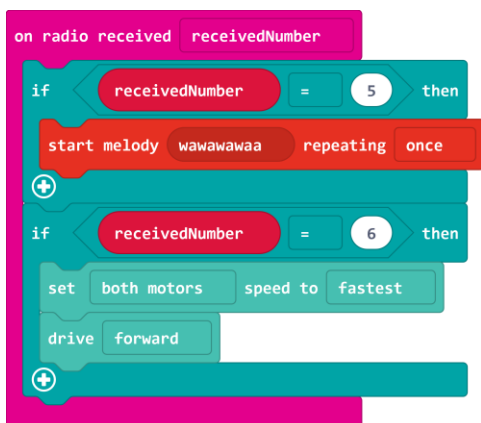
When the additional micro:bit sends the signal to attack (the radio message with number **6**), the GiggleBot will drive forward as quickly as it can.



- > Connect another **if true then** __ block inside of the **on radio received** block.



- > Attach a **0 = 0** block to the **if** __ portion of the new **if __ then** __ block.
- > Inside of this block place a **receivedNumber** block (found under **Variables**) in the first place for a value. This is a variable that will change value based on what number is sent by the additional micro:bit.
- > Change the second value in the **0 = 0** block to **6** since that is what was assigned to this action in the plan above.

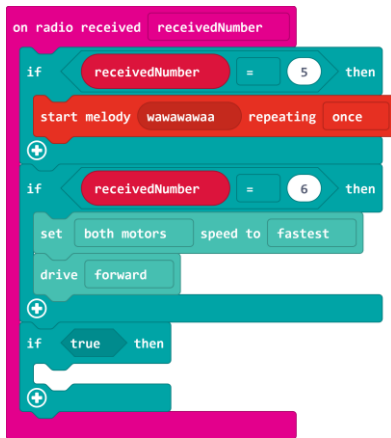


- > Attach movement blocks to the **then** __- portion of the **if __ then** __ block.
- > Connect a **set both motors speed to slowest** block.
- > Change **slowest** to **fastest** using the dropdown menu.
- > Underneath the set speed block, attach a **drive forward** block. This tells the GiggleBot to drive forward forever (or until it is told to stop).

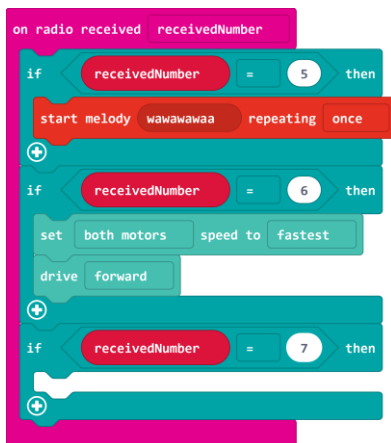


G: STOP THE GIGGLEBOT

Almost done! Since the GiggleBot is going forward for an unspecified amount of time, we need to be able to stop it.

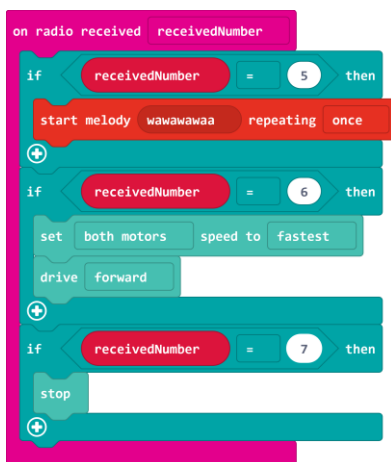


- Add one last **if true then** __ block inside of the **on radio received** block.



In the plan, the number **7** meant stopping the GiggleBot.

- Connect a **0 = 0** block to the **if** portion of the new **if __ then __** block, replacing the **true** placeholder.
- Inside of this block place a **receivedNumber** block (found under **Variables**) in the first place for a value, like we did twice already.
- Change the second value in the **0 = 0** block to **7** according to the plan above.

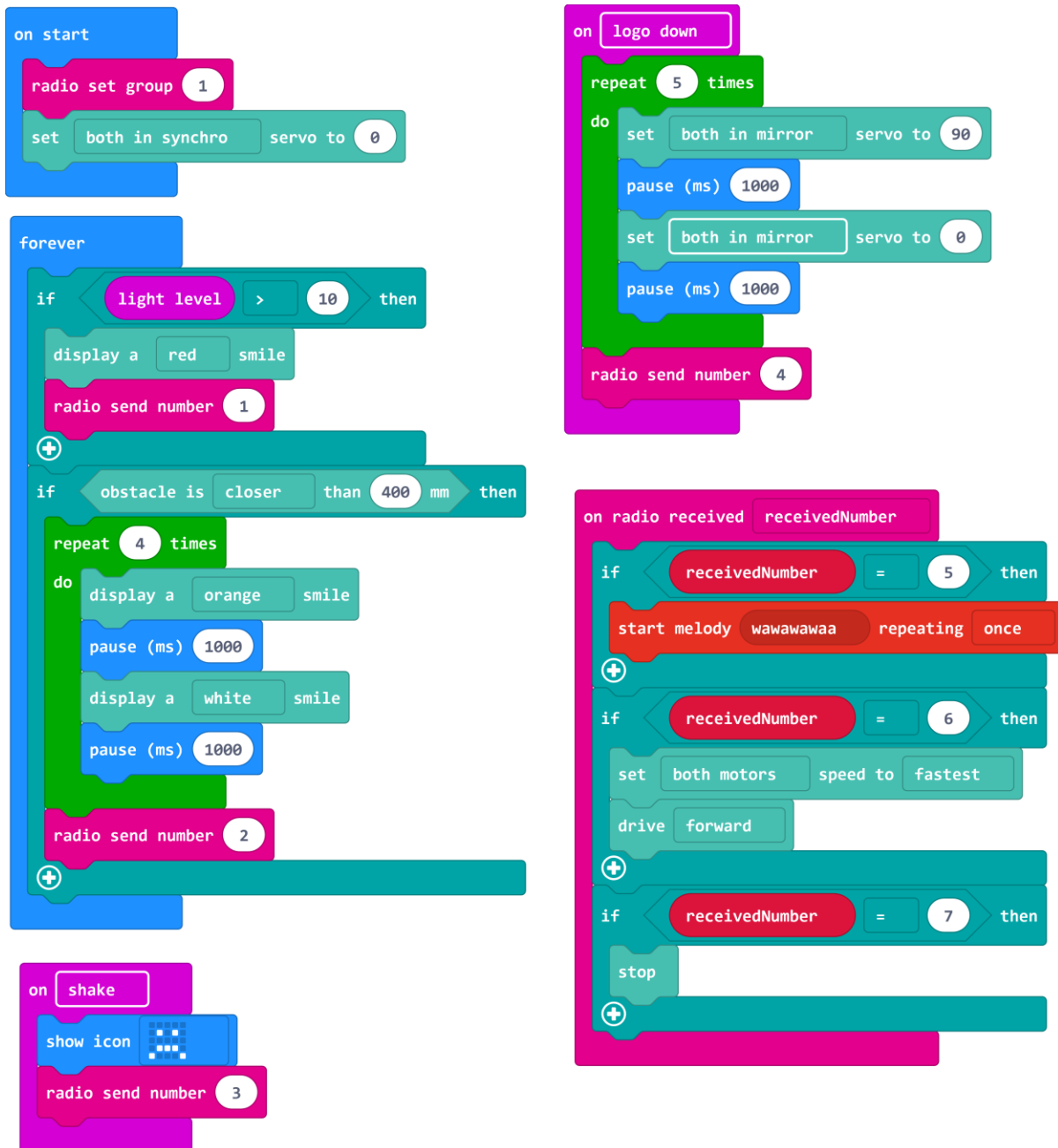


- Finally, connect a **stop** block to the **then** __ portion of the **if __ then __** block.



FINAL GIGGLEBOT CODE

Whew! That was a lot of programming! Here it is in all its glory!



```

on start
  radio set group 1
  set both in synchro servo to 0

forever
  if light level > 10 then
    display a red smile
    radio send number 1
  +
  if obstacle is closer than 400 mm then
    repeat 4 times
      do
        display a orange smile
        pause (ms) 1000
        display a white smile
        pause (ms) 1000
    radio send number 2
  +

on shake
  show icon [GiggleBot icon]
  radio send number 3

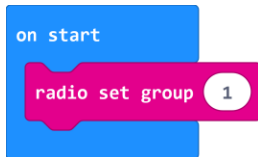
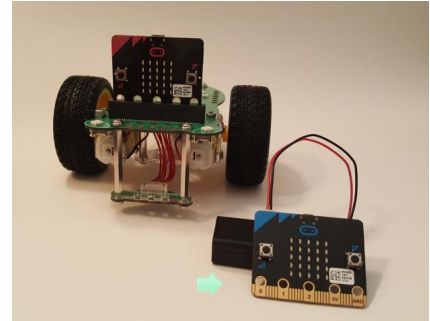
on logo down
  repeat 5 times
    do
      set both in mirror servo to 90
      pause (ms) 1000
      set both in mirror servo to 0
      pause (ms) 1000
  radio send number 4

on radio received receivedNumber
  if receivedNumber = 5 then
    start melody wawawawaa repeating once
  +
  if receivedNumber = 6 then
    set both motors speed to fastest
    drive forward
  +
  if receivedNumber = 7 then
    stop
  +
  
```

- > Give a name to your program if you haven't done so already.
- > Download and transfer this program to your GiggleBot.
- > Clear the workspace on MakeCode to write a new program for the additional micro:bit.

PROGRAM THE ADDITIONAL MICRO:BIT

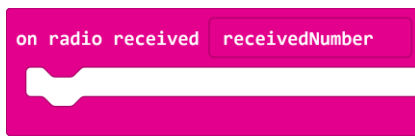
We will start this program the same way we started the program for the GiggleBot. First, we will set the radio group. Make sure this radio group matches the GiggleBot's radio group.



- > Put a **radio set group** __ block (found under **Radio**) inside of a **on start** event block to set the radio group for your GiggleBot.

REACTING TO SIGNALS FROM THE GIGGLEBOT

Keep the **on start** sequence on your workspace and add the following sequences to the workspace.



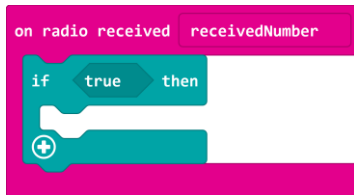
- > Move an **on radio received receivedNumber** block into your workspace.

Inside of this block we will tell the GiggleBot what to do whenever the GiggleBot sends a number to it using many **if** __ **then** __ blocks.

We will refer to our plan in order to keep track of what number means which action. Keep your plan at hand.

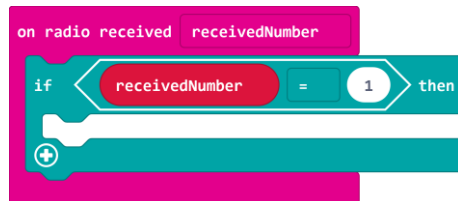
A: TURNING ON THE LIGHTS

When the GiggleBot detects light, it sends out a radio message with the number **1**. The additional micro:bit needs to react when it receives that message, and let us know it has received it.



> Connect an **if true then** block inside of the **on radio received receivedNumber** event block.

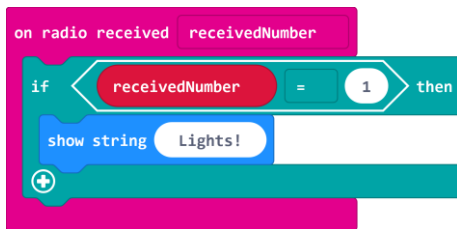
In the plan, the GiggleBot sends a **1** when it detects a change in light level (dark to light).



> Attach a **0 = 0** block to the **if** ___ portion of the **if** ___ **then** ___ block.


> Place a **receivedNumber** block in the first place for a value and change the second value to **1**.

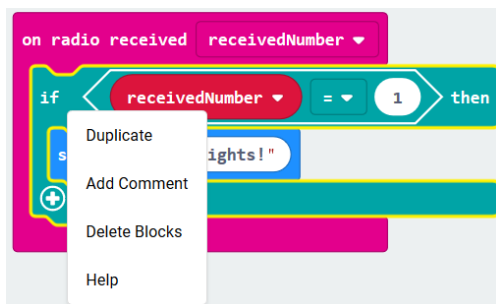
> The number **1** corresponds to the light level changing from dark to light in the GiggleBot's environment



> Attach a **show string** ___ (found under Basic)

> Change the string to "Lights!" or whatever message you want it to display.

 And here's a neat little trick that will simplify our work as we are going to need an if-then statement for each of the sent numbers

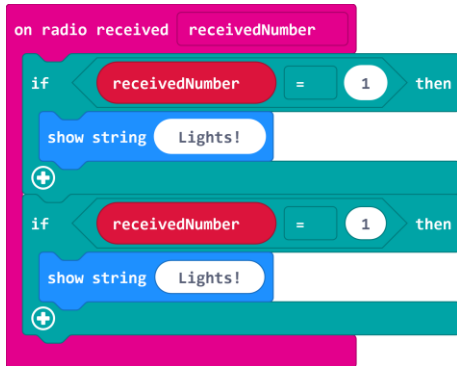


> To expedite this process, right-click on the **if** ___ **then** ___ block and select **Duplicate**.

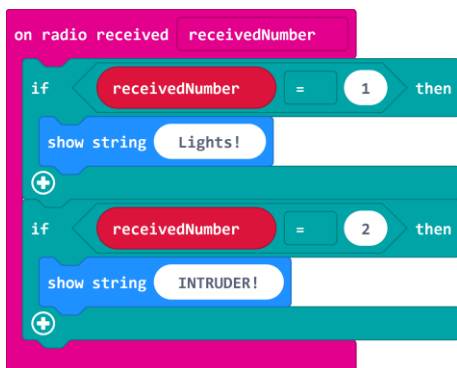
> Connect the duplicated blocks to the previous **if** ___ **then** ___ block.

B: DETECTION OF AN OBJECT (DISTANCE SENSOR)

According to the plan, when the GiggleBot detects an object it sends out a radio message with the number **2**. The additional micro:bit has to let us know what is happening. We need to code it to react to the number **2**.



> Duplicate the **if __ then __** block as shown on the previous page, if you haven't done so already.



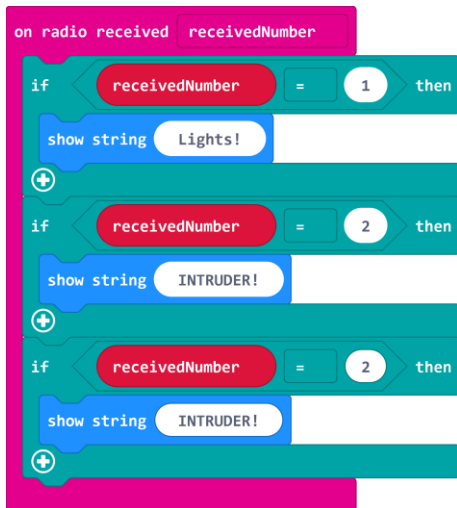
> Change the second **if** statement so that it checks for the number **2**

> Change the **show string __** statement to say "INTRUDER!"

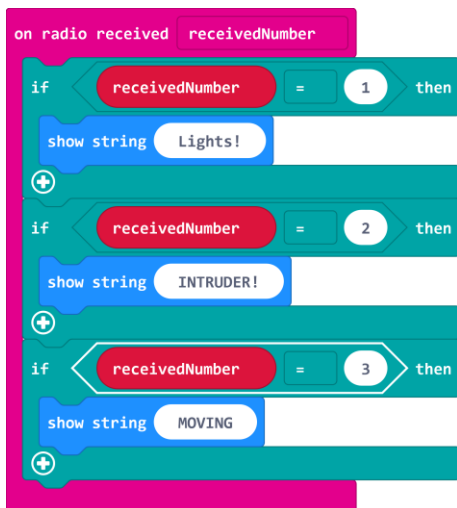
This corresponds to the detection of an object near the GiggleBot.

C: DETECTION OF GIGGLEBOT MOVEMENT (“ON SHAKE”)

Based on the plan, when the GiggleBot detects it is being shaken, it sends out a message with the number **3**. The additional micro:bit needs to react to the number **3** and let us know what is happening with the GiggleBot.



- > Right click on the **if-then** block once again, duplicate it.
- > Attach it to the second **if-then** block.

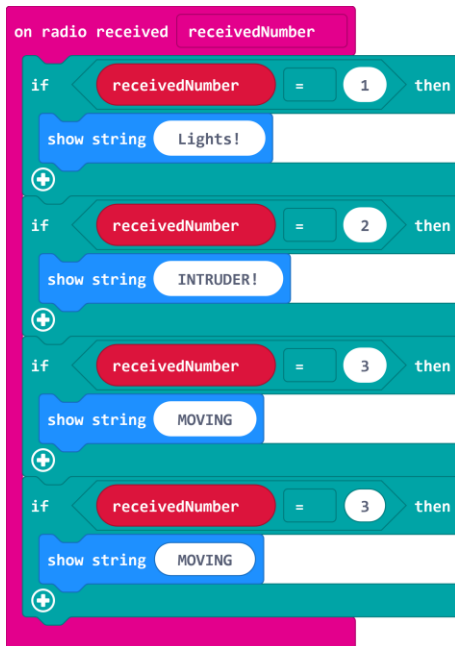


- > Change **receivedNumber = 2** to **receivedNumber = 3**
- > Change the word to **MOVING**.

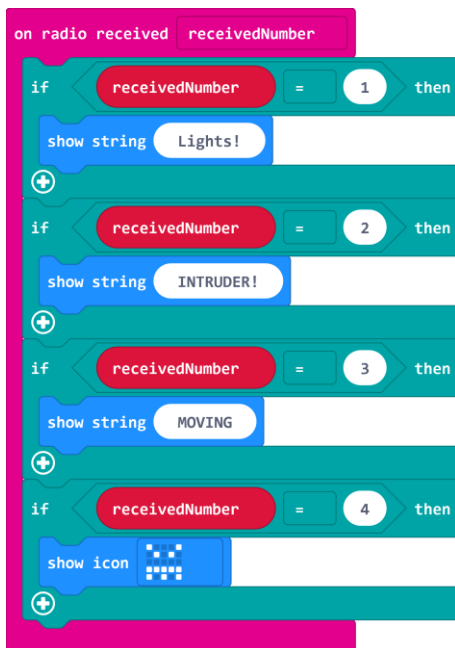
This corresponds to the GiggleBot being shaken.

D: REACTING TO BEING TURNED UPSIDE DOWN

When the GiggleBot gets turned upside down, it sends out a radio message with the number **4**. We need to code the additional micro:bit to display a message whenever the GiggleBot finds itself upside down.



- > Right click on the **if-then** block once again, duplicate it.
- > Attach it below the third **if-then** block.



- > Change the number **3** to a **4**. This is what the GiggleBot sends when it's turned upside down.
- > Get rid of the **show string** block
- > Insert a **show icon** block (found under **Basic**)
- > Change the icon to the **angry face** icon.

SENDING SIGNALS TO THE GIGGLEBOT

We have finished handling the messages sent by the GiggleBot. Next, we will use some of the inputs on the additional micro:bit to send signals back to the GiggleBot.

E: ALARM SOUNDING

We programmed the GiggleBot to play a sound on the speaker already when it receives a radio message with the number 5. Now we need to program the additional micro:bit to send that message. Keep the code you've already done in your workspace and add the following.



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



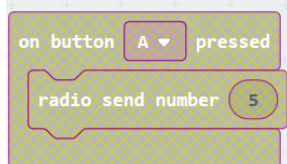
The only other block we need to put inside this block is one to send the number 5 to the GiggleBot

- > Move a radio send number __ block
- > Change the number to **5**.

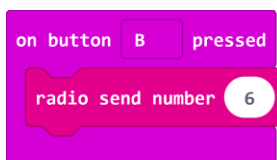
F: DRIVING FORWARD IN ATTACK MODE



> Right-click on the block and duplicate it, just like we did earlier.



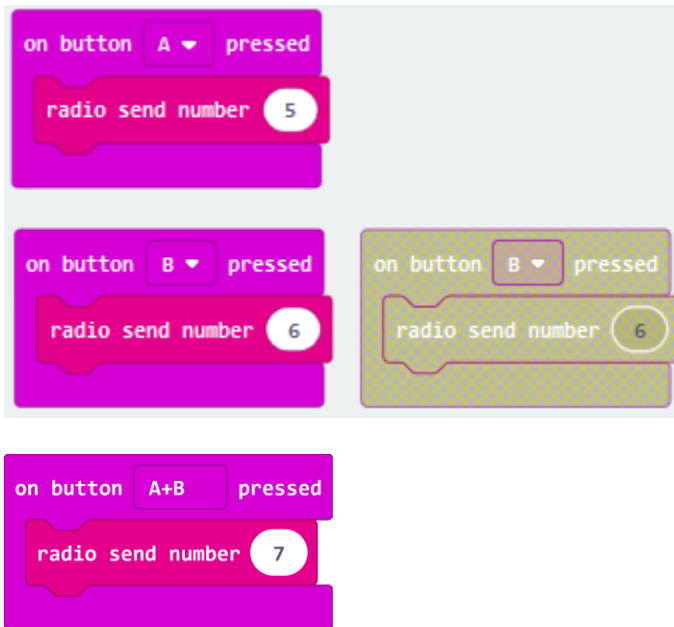
You will notice that this block shows up grayed out. That is because we have already programmed the additional micro:bit to do something when button **A** is pressed.



- > Use the drop-down menu to change the button to **B**
- > Change the number being sent to **6**. This triggers the code on the GiggleBot to drive forward.

G: STOP THE GIGGLEBOT

Pressing both buttons (A+B) will send a radio message with the number 7, which we have already coded on the GiggleBot side. The GiggleBot will stop when it receives this message.

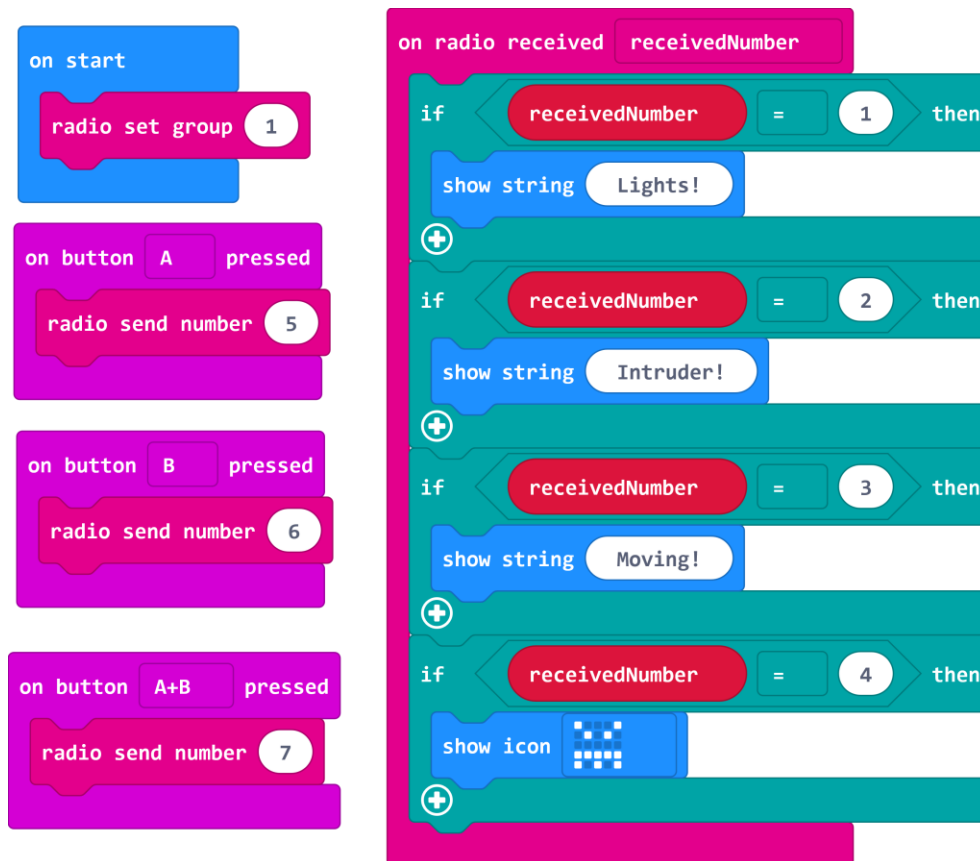


> Right-click on either of the **on button pressed** blocks and duplicate it once again.

- > Change it to **on button A + B pressed**.
- > Change the number to **7**.

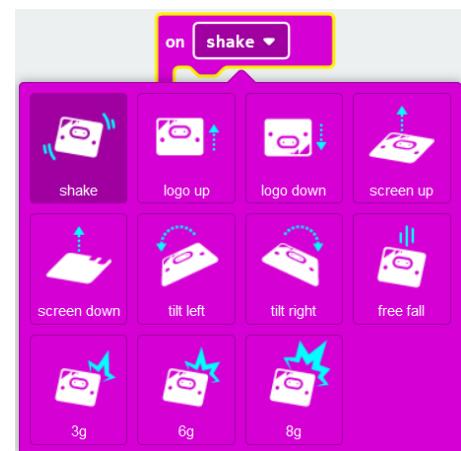
FINAL ADDITIONAL MICRO:BIT CODE

Here is the whole program for the additional micro:bit! Pretty impressive! High Fives all around!



Transfer it to the additional micro:bit and test it out with the GiggleBot!

You can add even more inputs to send messages back to the GiggleBot. Experiment with using other input blocks for the additional micro:bit such as **on shake**, **logo up** (normal position for the GiggleBot), **screen up**, **screen down**, **tilt left**, and **tilt right**.



❗ Remember: If you add more inputs and signals sent back to the GiggleBot, you will need to go back and modify the GiggleBot program to include those received numbers as well.

> TRY IT OUT!

After all of that hard work it is time to do some testing! Place your GiggleBot security guard in a dark place, away from objects. Turn on the GiggleBot and connect the battery pack to your additional micro:bit. You are now ready to test all of the GiggleBot's new security features!

- > Turn on the lights
- > Get close to it
- > Shake it
- > Turn it over
- > Press the buttons on the additional micro:bit (no need to mash the buttons!)
- > Try it all!

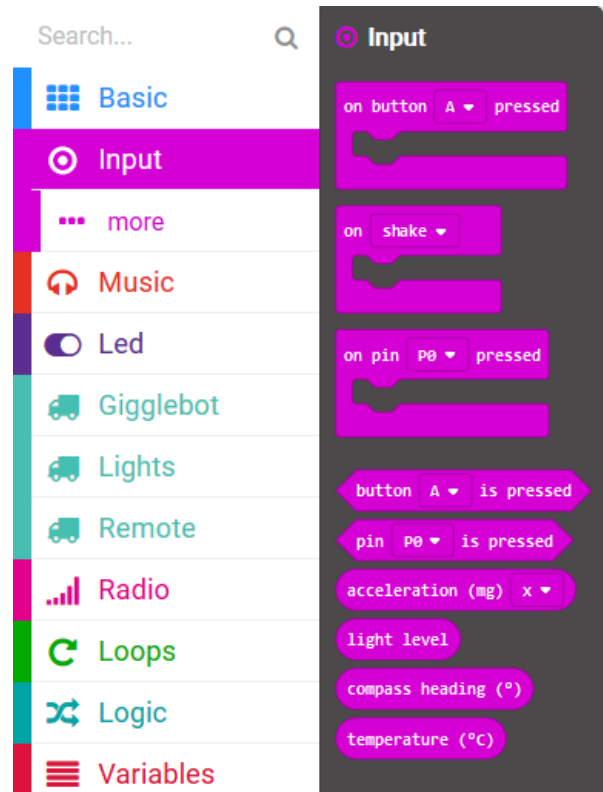
ITERATE

- > Did your GiggleBot perform well as a security guard?
 - What 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.
- > What other security features do you want your GiggleBot to have?
 - Think about all of the things you learned about through the previous missions.
 - Could you use more of the sensors on the GiggleBot and micro:bit?

Did you use the light sensors on the GiggleBot or just the light sensor on the micro:bit?

Look at the Input blocks. Did you use all of the inputs?

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: ***Guard_V1*** and ***Remote_V1*** where “V” stands for version).



CHALLENGE

Now that you have designed and built a guardian robot to protect your valuables, it is time to figure out how to make it even better. First, you must find the weaknesses.

- > Set up your guardian robot and try to defeat the security measures.
- > Can you move the object you are protecting WITHOUT the GiggleBot guardian noticing?
- > Can you turn on lights WITHOUT the GiggleBot guardian noticing?
- > Can you move the GiggleBot without it sending a warning to the additional micro:bit?

Now that you have found the weaknesses, edit your program so that it **cannot** be defeated again.



Copyright Dexter Industries 2019. All rights reserved. Reproduction and distribution of the Mission without written permission of Dexter Industries is prohibited. GiggleBot is a registered Trademark of Dexter Industries.

Contact dexterred@dexterindustries.com for permissions and questions.