

Microbit Robotics Advanced Level 3

Lesson 6

Advanced Remote Control

Presented by Advanced Superlogic Team

Today's Topic

1. Doing 2 actions at once
2. Message Chain
3. Advanced Remote Control Programming
4. Game Challenge

Learning Outcome

1. **Able to understand how to perform 2 actions with 1 message**
2. **Able to program message sending with message chain**
3. **Able to decode the message on Superbit**

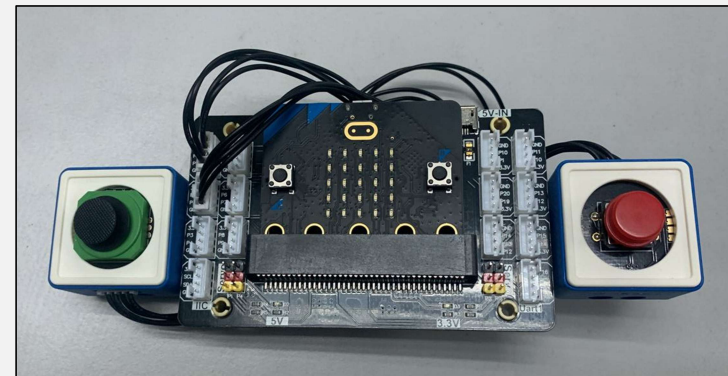
Can your Skip Car move and eject?

Remote Controller Program

```

on start
  radio set group 8
  show number 8

forever
  if Rocker pin P0P1 value Up then
    radio send string "forward"
  else if Rocker pin P0P1 value Down then
    radio send string "backward"
  else if Rocker pin P0P1 value Left then
    radio send string "left"
  else if Rocker pin P0P1 value Right then
    radio send string "right"
  else if Button pin P2P3 value Press then
    radio send string "lift"
  else if Rocker pin P0P1 value NoState then
    radio send string "stop"
  
```



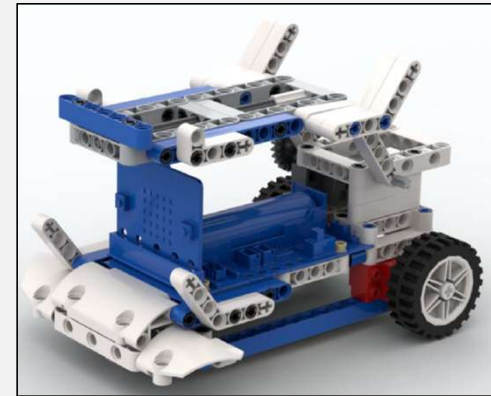
This is your remote controller program, where I didn't put "button release" as lower down the carrier.



Skip Car Program

```
on start
  radio set group 8
  show number 8
  Servo(270°) num 55 pos forward value 0
  Servo(360°) num 51 pos forward value 0
  Servo(360°) num 52 pos forward value 0

on radio received receivedString
  if receivedString = "forward" then
    Servo(360°) num 51 pos forward value 90
    Servo(360°) num 52 pos reverse value 90
  else if receivedString = "backward" then
    Servo(360°) num 51 pos reverse value 90
    Servo(360°) num 52 pos forward value 90
  else if receivedString = "left" then
    Servo(360°) num 51 pos forward value 90
  else if receivedString = "right" then
    Servo(360°) num 51 pos reverse value 90
    Servo(360°) num 52 pos reverse value 90
  else if receivedString = "lift" then
    Servo(270°) num 55 value 90
    pause (ms) 1000
    Servo(270°) num 55 value 0
    pause (ms) 1000
  else if receivedString = "stop" then
    Servo(360°) num 51 pos forward value 0
    Servo(360°) num 52 pos forward value 0
```



With this program, you would find out that your robot sometimes cannot stop when you press the button.



Skip Car Program

```
on start
  radio set group 8
  show number 8
  Servo(270°) num 55 value 0
  Servo(360°) num 51 pos forward value 0
  Servo(360°) num 52 pos forward value 0

on radio received receivedString
  if receivedString = 'forward' then
    Servo(360°) num 51 pos forward value 90
    Servo(360°) num 52 pos reverse value 90
  else if receivedString = 'backward' then
    Servo(360°) num 51 pos reverse value 90
    Servo(360°) num 52 pos forward value 90
  else if receivedString = 'left' then
    Servo(360°) num 51 pos forward value 90
  else if receivedString = 'right' then
    Servo(360°) num 51 pos reverse value 90
    Servo(360°) num 52 pos reverse value 90
  else if receivedString = 'lift' then
    Servo(270°) num 55 value 90
    pause (ms) 1000
    Servo(270°) num 55 value 0
    pause (ms) 1000
  else if receivedString = 'stop' then
    Servo(360°) num 51 pos forward value 0
    Servo(360°) num 52 pos forward value 0
```

As we are using “if then, else if” conditional statement here, it is going to check the condition from top to bottom.

In this case, if “forward” is not received, it will skip the forward action and proceed to check if “backward” is receive. If the “backward” message is received, then it will not proceed to check further.

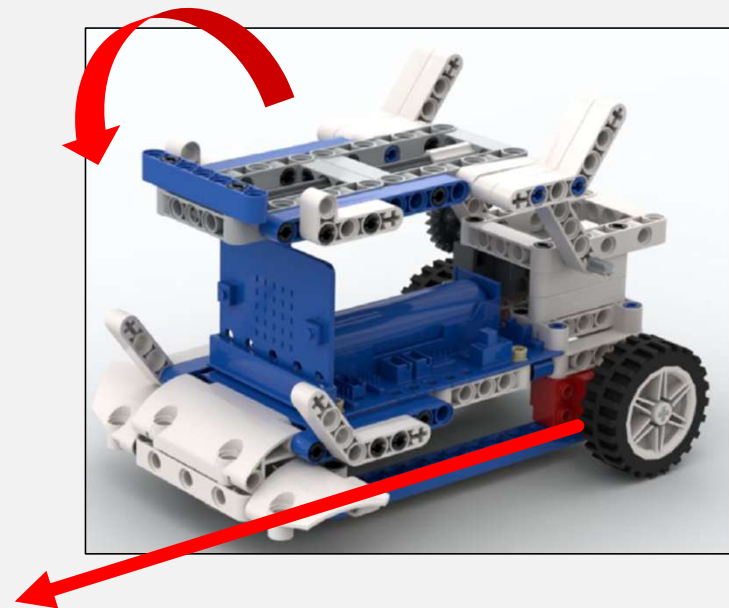
Therefore, in this case, only 1 of the actions can be executed, like first true first execute basis.

Doing 2 Actions at Once

Doing 2 Actions at Once

Since some of our actions might happen at once, for instance, I want to eject my load while I'm moving forward to get some momentum while unloading the trash, instead of just staying there and unload.

Here we need “forward” and “lift” actions to be done together.

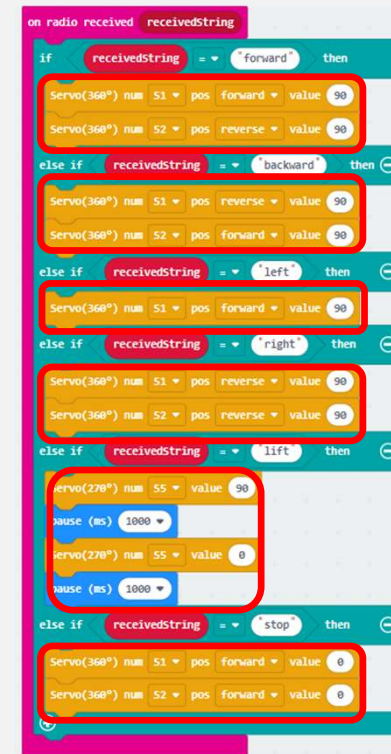


Doing 2 Actions at Once

Unfortunately, we are only allowed to send 1 message every time, so do the receiver.

With just 1 action per message, it's hardly done to make our robot to listen to 2 actions.

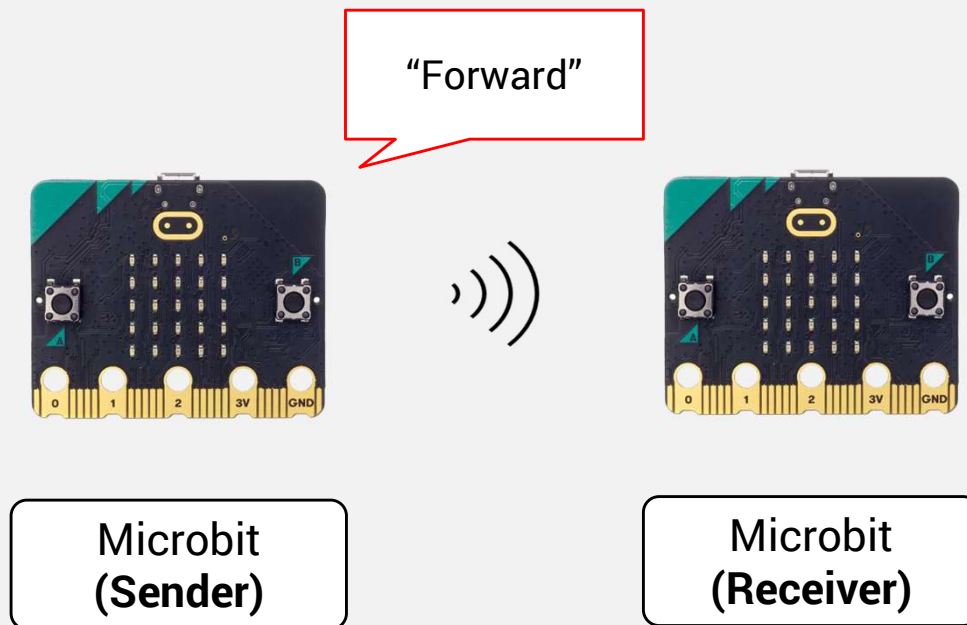
And our robot will only choose 1 action to perform.



```
on radio received receivedString
if receivedString = 'forward' then
  Servo(360°) num 51 pos forward value 90
  Servo(360°) num 52 pos reverse value 90
else if receivedString = 'backward' then
  Servo(360°) num 51 pos reverse value 90
  Servo(360°) num 52 pos forward value 90
else if receivedString = 'left' then
  Servo(360°) num 51 pos forward value 90
else if receivedString = 'right' then
  Servo(360°) num 51 pos reverse value 90
  Servo(360°) num 52 pos reverse value 90
else if receivedString = 'lift' then
  Servo(278°) num 55 value 90
  pause (ms) 1000
  Servo(278°) num 55 value 0
  pause (ms) 1000
else if receivedString = 'stop' then
  Servo(360°) num 51 pos forward value 0
  Servo(360°) num 52 pos forward value 0
```

Message Chain

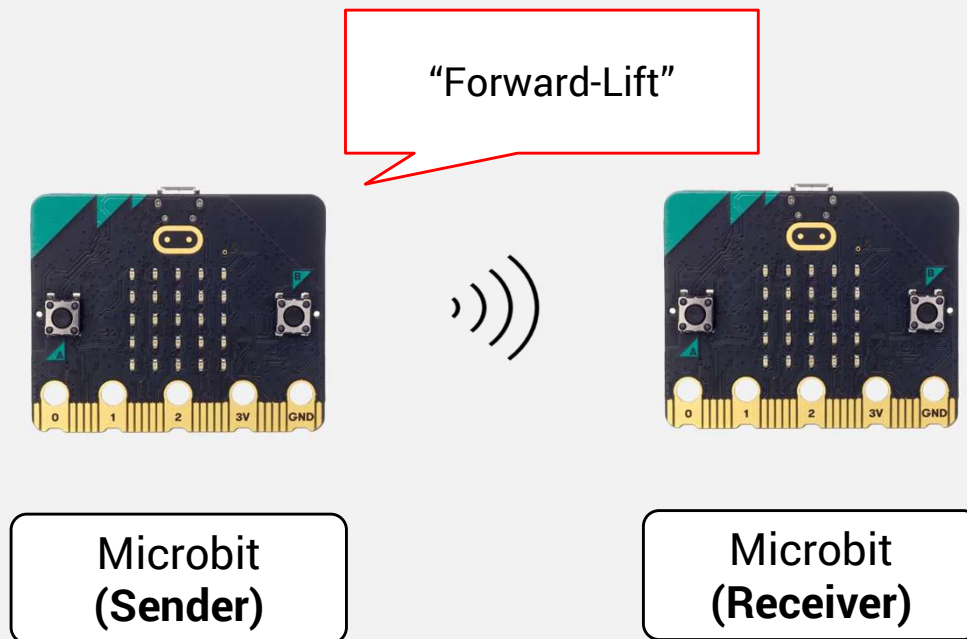
Message Chain



Every time we interact with our rocker module and button, it will trigger microbit (sender) to send a message.

As we are only allowed to send a message each time we interact, to make the superbit to react with 2 actions, we can send a message with the information of 2 actions.

Message Chain



We have 2 separate actions, which is movement and action:

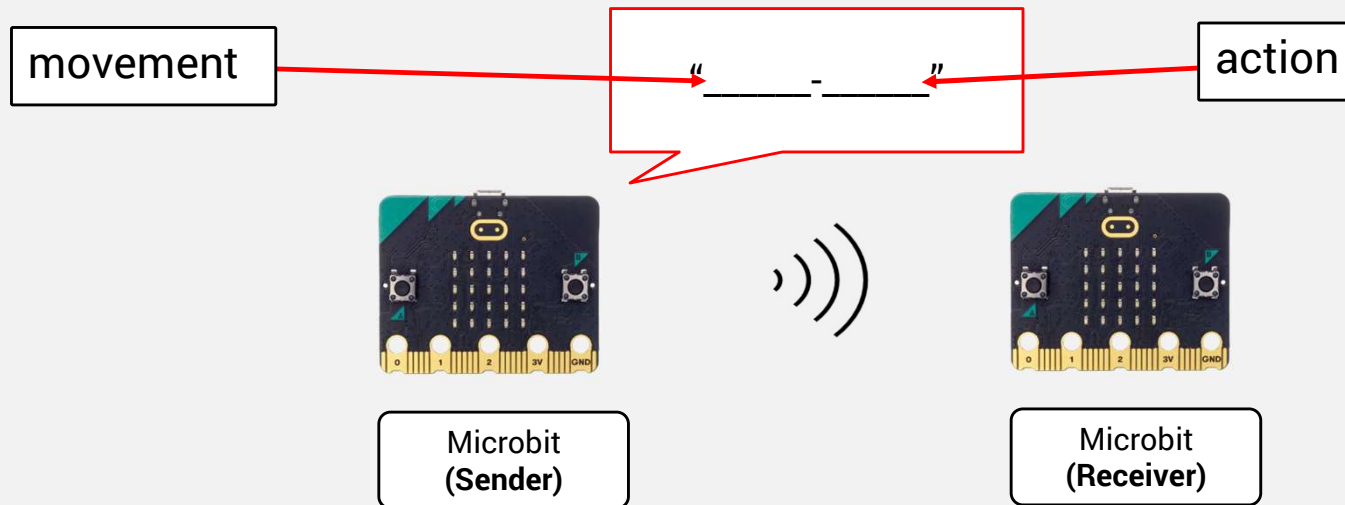
Movement:

1. Forward
2. Backward
3. Left
4. Right
5. Stop

Action:

1. Lift
2. Lower

Message Chain



We can compile the movement and action into a message with the “-” as a separator.

Then we use our receiver to split the message into 2 actions (movement and action).

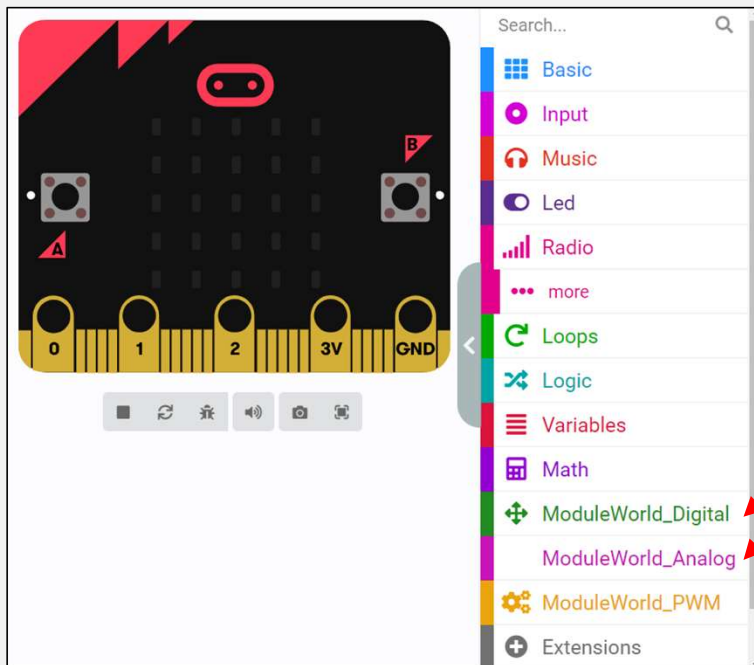
Let's code the message chain for our Remote Controller

First, we need to connect the micro:bit to the computer by USB cable. The computer will pop up a USB flash drive and click on the URL in the USB flash drive: <http://microbit.org/> to enter the programming interface.

And add the **World of Module Extension**.

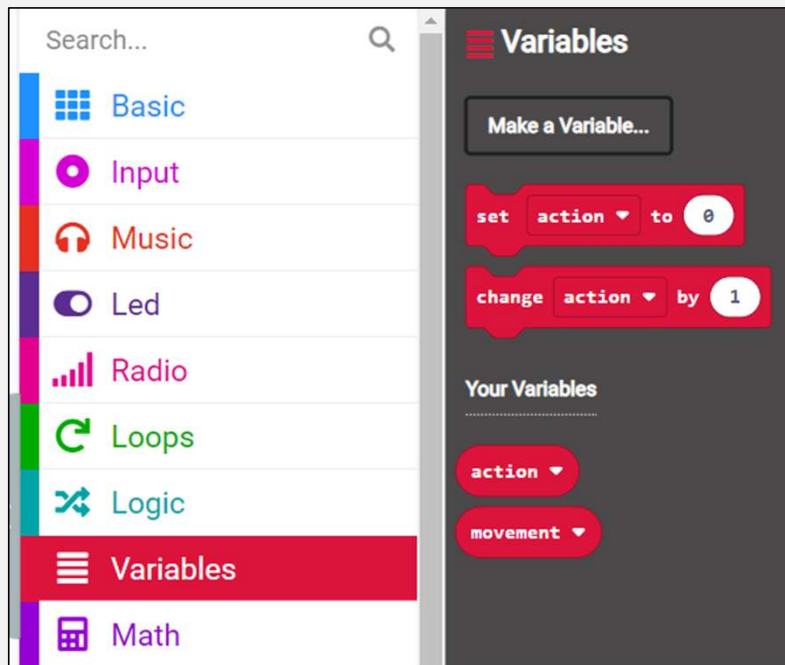
Search [yahboomtechnology/module-world](http://yahboomtechnology.com/module-world) in the Microbit extension.

WOM Extension



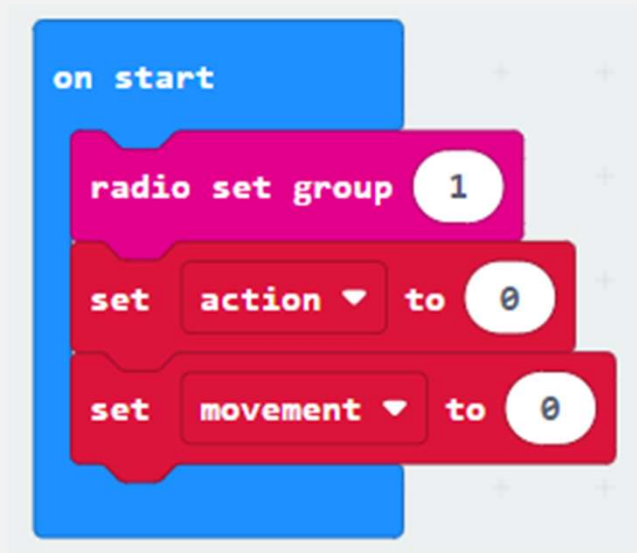
We are going to use the digital blocks for **button** interaction and analog blocks for **rocker module**.

Set Radio Group & Set up Variable



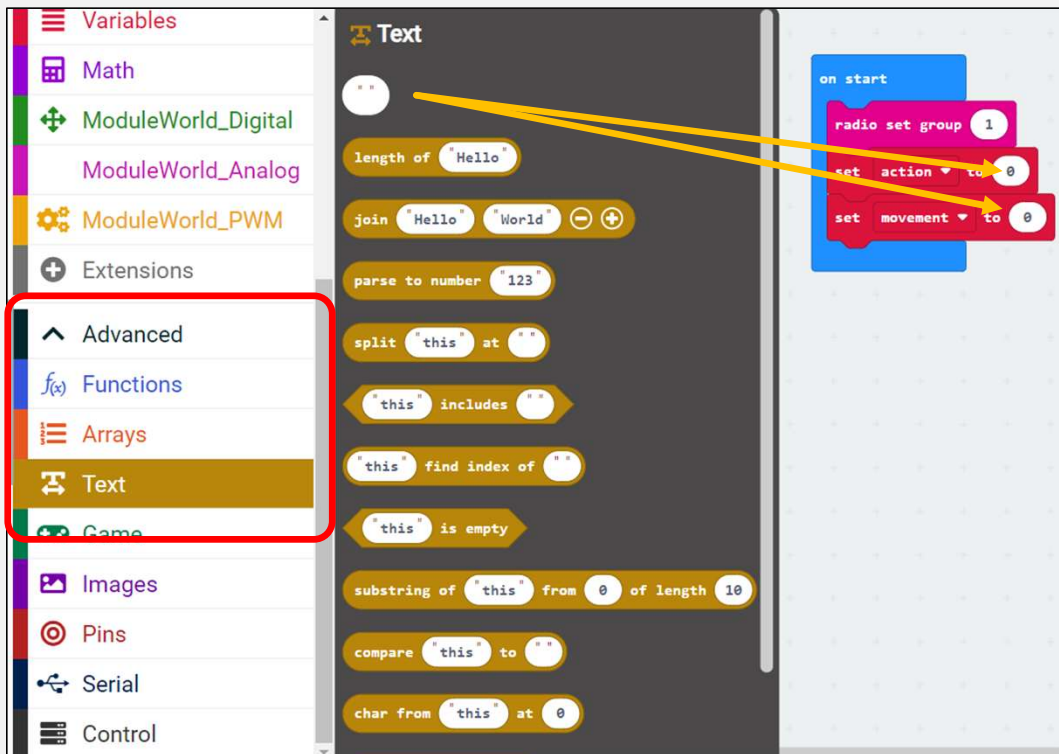
We want to add 2 variables:
action & movement.

On Start – Variable as Text



As we want our action and movement to be sent as text, we need to make it to text (string) variable.

On Start – Variable as Text

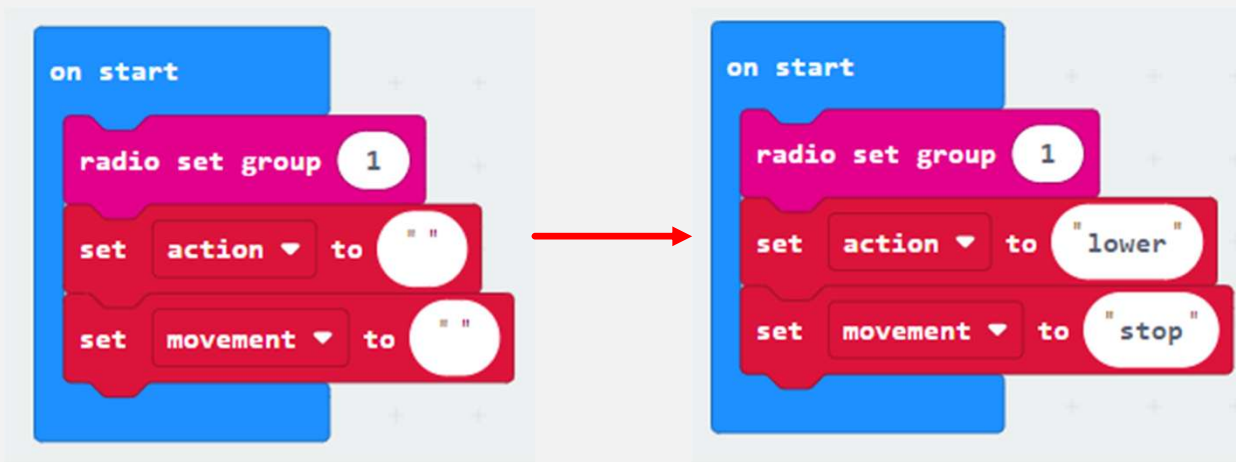


Now navigate to the Advanced block and click the advanced block category.

Select the Text category.

Pick the first block and put it in the action and movement variables.

On Start – Variable as Text



Now you should be able to type word in the variable as we have set the variables as text.

We will set “lower” in action and “stop” in movement as our default.

Forever – Variable changed based on Rocker state

```
forever
  if Rocker pin P0P1 value Up then
    set movement to "forward"
  else if Rocker pin P0P1 value Down then
    set movement to "backward"
  else if Rocker pin P0P1 value Left then
    set movement to "left"
  else if Rocker pin P0P1 value Right then
    set movement to "right"
  else if Rocker pin P0P1 value NoState then
    set movement to "stop"
```

Let's set the movement variable with the rocker condition first.

We have 5 possible action for the movement:

forward
backward
left
right
stop

Forever – Variable changed based on Button

```
forever
  if Rocker pin P0P1 value Up then
    set movement to 'forward'
  else if Rocker pin P0P1 value Down then
    set movement to 'backward'
  else if Rocker pin P0P1 value Left then
    set movement to 'left'
  else if Rocker pin P0P1 value Right then
    set movement to 'right'
  else if Rocker pin P0P1 value NoState then
    set movement to 'stop'
  +
  if Button pin P2P3 value Press then
    set action to 'lift'
  else if Button pin P2P3 value Release then
    set action to 'lower'
```

Now is to add in the button state for action.

We have 2 possible actions:

lift
lower

In this case, the action is to be added with another if else block after the movement.

** Do not add the else if condition after the rocker module. As we want both to be able to happened at the same time.

Forever – Send string through radio

```
forever
  if Rocker pin P0P1 value Up then
    set movement to "forward"
  else if Rocker pin P0P1 value Down then
    set movement to "backward"
  else if Rocker pin P0P1 value Left then
    set movement to "left"
  else if Rocker pin P0P1 value Right then
    set movement to "right"
  else if Rocker pin P0P1 value NoState then
    set movement to "stop"
  +
  if Button pin P2P3 value Press then
    set action to "lift"
  else if Button pin P2P3 value P.false then
    set action to "lower"
  +
  radio send string ""
```

radio send string

In the end of the button state condition, we want to add in the radio send string action.

Send the strings



The screenshot shows the Microbit IDE interface. On the left, a sidebar lists various block categories: Logic, Variables, Math, ModuleWorld_Digital, ModuleWorld_Analog, ModuleWorld_PWM, Extensions, Advanced, Functions, Arrays, Text, Game, Images, and Pins. The 'Text' category is selected and expanded, showing several blocks including 'length of', 'join', 'parse to number', 'split', 'this includes', 'this find index of', 'this is empty', 'substring of', 'compare', and 'char from'. The 'join' block is highlighted with an orange arrow pointing to its use in a 'radio send string' block in the main workspace. The workspace shows a sequence of blocks: an 'else if' block for 'Rocker pin P0P1 value NoState', a 'set movement to stop' block, an 'if' block for 'Button pin P2P3 value Press' with a 'set action to lift' block, an 'else if' block for 'Button pin P2P3 value Realse' with a 'set action to lower' block, and finally a 'radio send string' block containing a 'join' block with 'Hello' and 'World' separated by a hyphen.

Then we want to send a message chain which holds 2 information separated by “-”.

So, we need to join the 2 variables together and make a “-” in between.

We can use “join string” block from the Text block category.

Radio send String



You can click the “+” button to add 1 more column for the string.

Then we will join the movement and the action with the “-” in between.

Drag your “movement” variable to the first block (replace the “Hello”).

Then replace “-” with “World”.

And drag the “action” variable to the last block.



Remote Controller Codes

```
on start
  radio set group 1
  set action to "lower"
  set movement to "stop"

forever
  if Rocker pin P0P1 value Up then
    set movement to "forward"
  else if Rocker pin P0P1 value Down then
    set movement to "backward"
  else if Rocker pin P0P1 value Left then
    set movement to "left"
  else if Rocker pin P0P1 value Right then
    set movement to "right"
  else if Rocker pin P0P1 value NoState then
    set movement to "stop"
  if Button pin P2P3 value Press then
    set action to "lift"
  else if Button pin P2P3 value Release then
    set action to "lower"
  radio send string join movement "-" action
```

Now your Remote Controller coding will be like this.

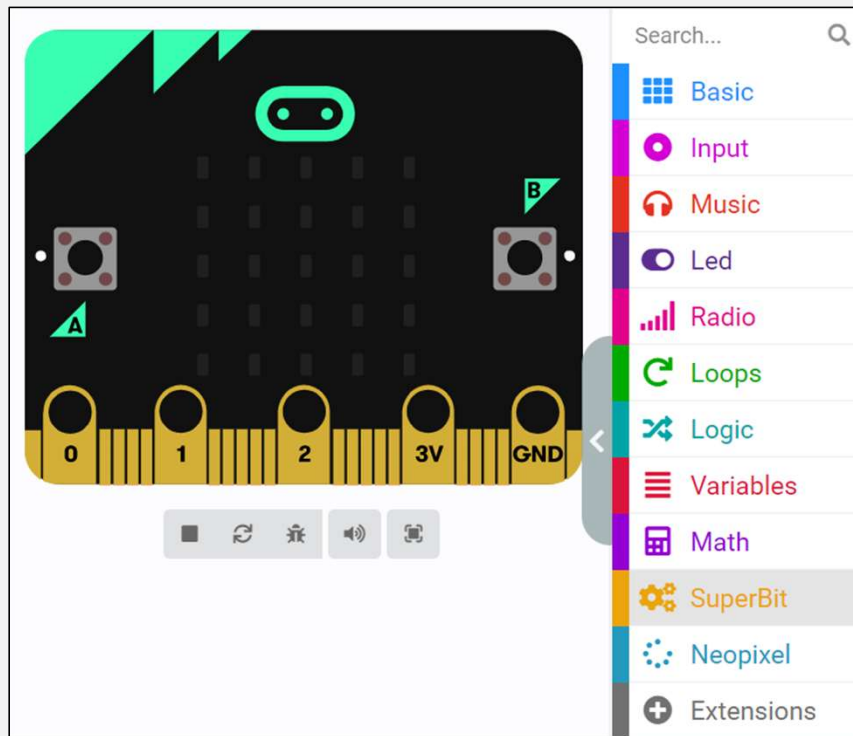
Unpack the Message Chain

Let's code your Skip Car (Required Superbit Extension)

First, we need to connect the micro:bit to the computer by USB cable. The computer will pop up a USB flash drive and click on the URL in the USB flash drive: <http://microbit.org/> to enter the programming interface.

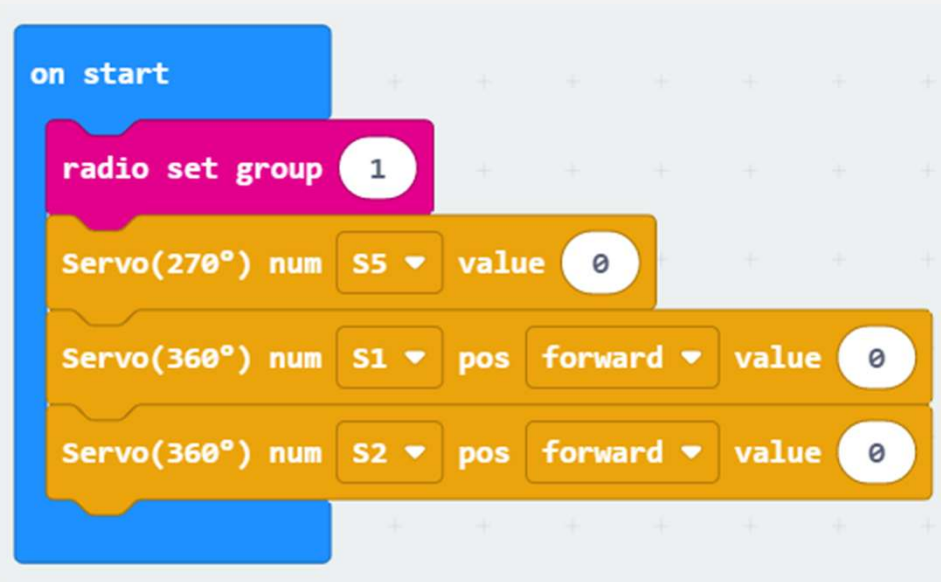
Search [lzty634158/SuperBit](#) in the Microbit extension.

Superbit Extension



After importing the superbite extension, you will see **SuperBit** and **Neopixel** in your coding blocks.

MakeCode Programming

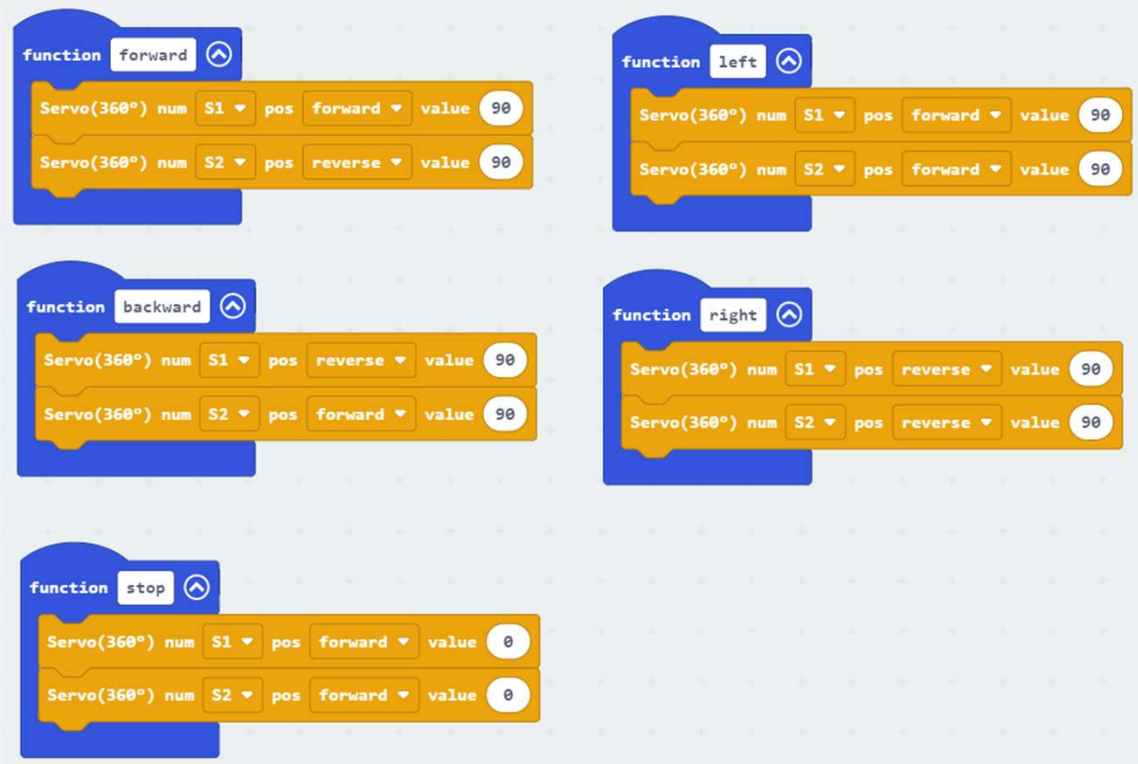


Program our starting state of our servos. And also set the radio group to the group that you set on your remote controller.

****You can use your last lesson's program.**

****We just need to add the actions for received signals from the remote controller.**

Make all movement functions

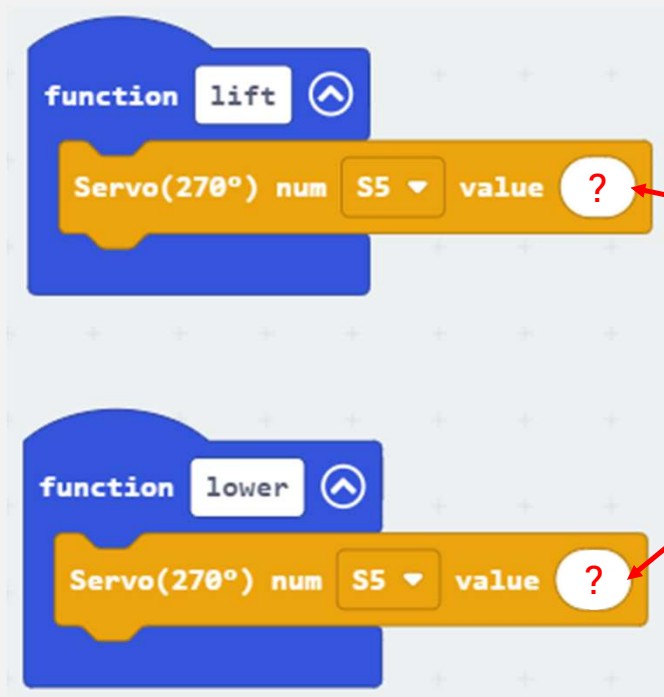


The image displays five Scratch function blocks for controlling two servos (S1 and S2) at 360 degrees. Each function block is a blue container with a title and an up arrow icon. The 'forward' block contains two orange blocks: 'Servo(360°) num S1 pos forward value 90' and 'Servo(360°) num S2 pos reverse value 90'. The 'backward' block contains two orange blocks: 'Servo(360°) num S1 pos reverse value 90' and 'Servo(360°) num S2 pos forward value 90'. The 'left' block contains two orange blocks: 'Servo(360°) num S1 pos forward value 90' and 'Servo(360°) num S2 pos forward value 90'. The 'right' block contains two orange blocks: 'Servo(360°) num S1 pos reverse value 90' and 'Servo(360°) num S2 pos reverse value 90'. The 'stop' block contains two orange blocks: 'Servo(360°) num S1 pos forward value 0' and 'Servo(360°) num S2 pos forward value 0'.

Use the function method to create all movement functions, including:

forward
backward
left
right
stop

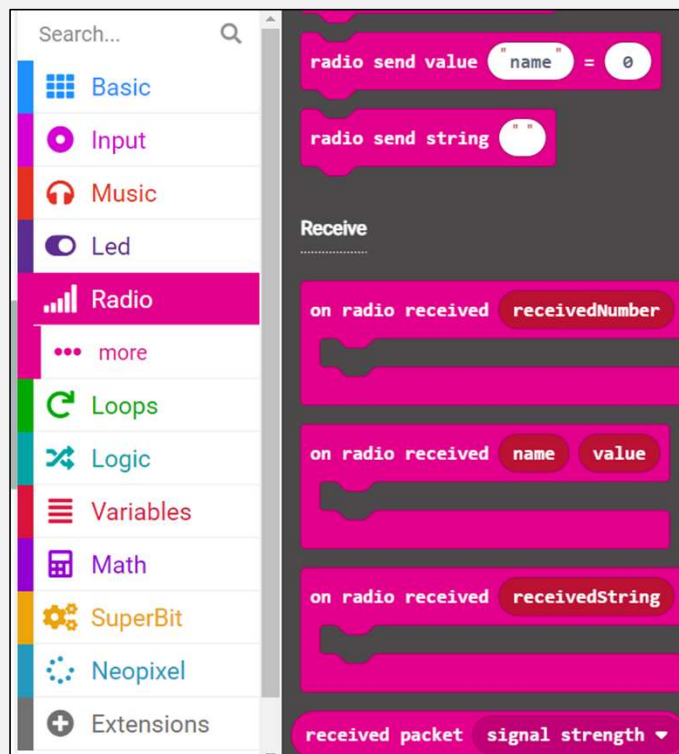
Make functions for your actions



Add in the lift and lower actions for your skip car.

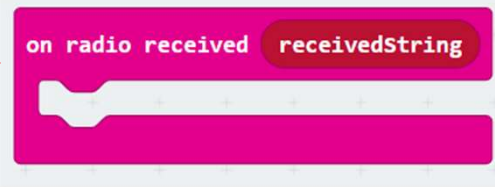
Here you need to test the value for your skip car, you can trial and error to see which values are best for your S5.

Code your Skip Car's actions



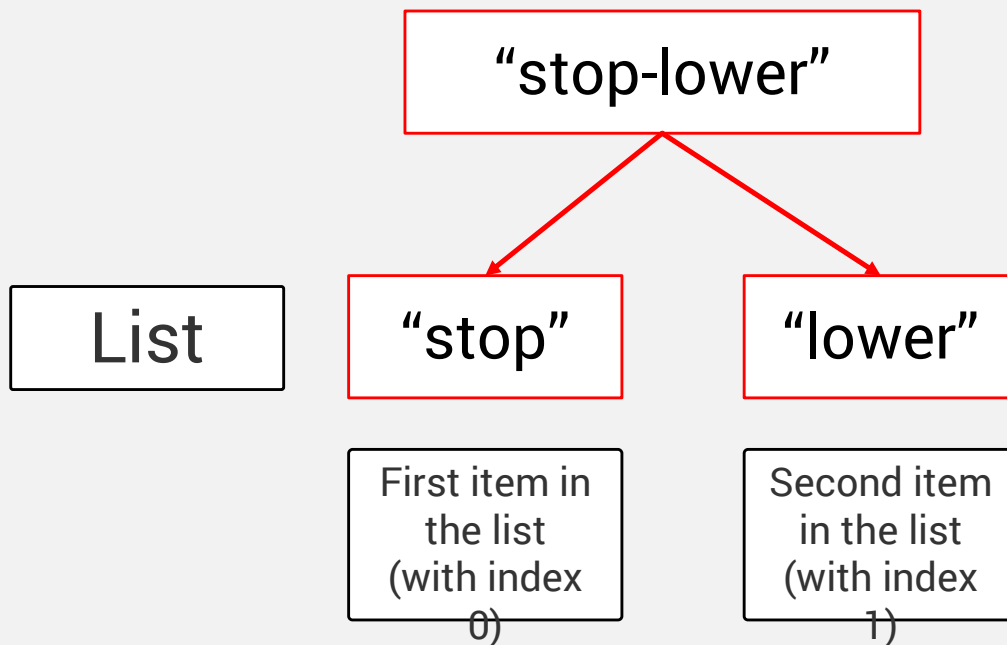
The screenshot shows the Scratch code editor interface. On the left is a sidebar with a search bar and a list of categories: Basic, Input, Music, Led, Radio (highlighted), more, Loops, Logic, Variables, Math, SuperBit, Neopixel, and Extensions. The main workspace contains several code blocks: a 'radio send value' block with 'name' set to '0', a 'radio send string' block, and a 'Receive' section with three 'on radio received' blocks. The first block has 'receivedNumber' as the variable. The second block has 'name' and 'value' as inputs. The third block has 'receivedString' as the input. A red arrow points from the 'receivedString' block in the workspace to a larger, highlighted version of the same block on the right.

Let's take the block "on radio received **'receivedString'**" for the actions when received the signals from other microbit.



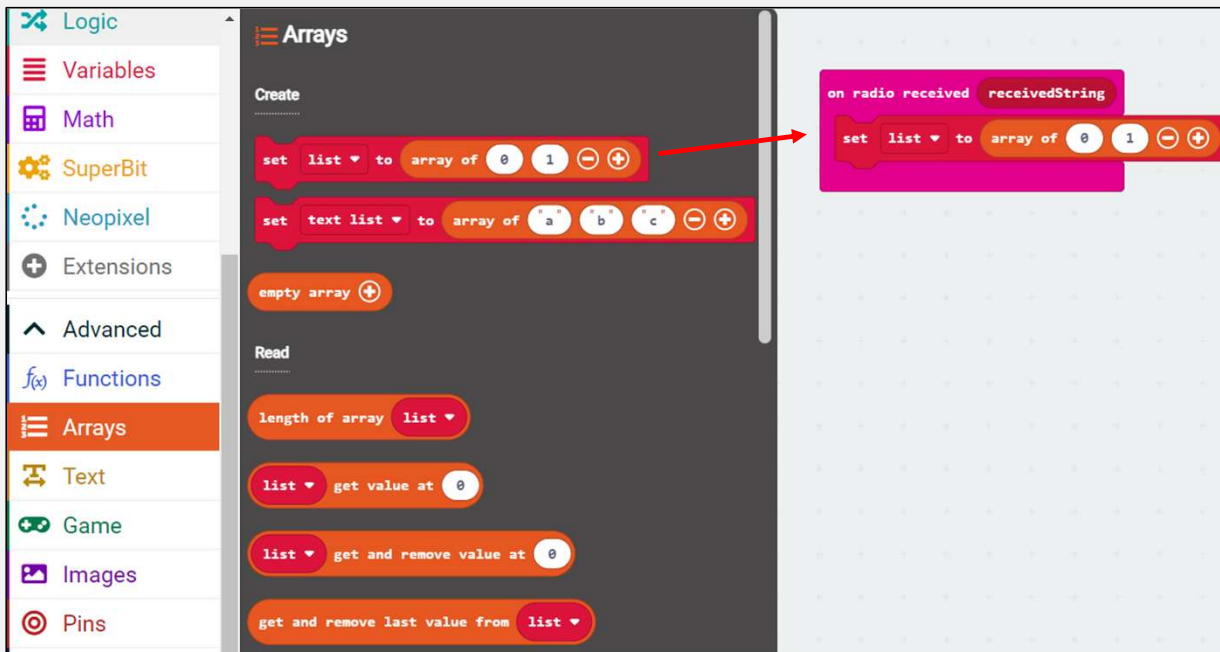
A single 'on radio received' block with 'receivedString' selected as the variable to receive the signal. The block is highlighted with a light blue background and a red arrow points to it from the main workspace.

Unpack the message to list



Now we want to split the message into 2 values and make it in a list. We will base on the "-" as the separator to split the message. This process is called unpacking.

Unpack Message



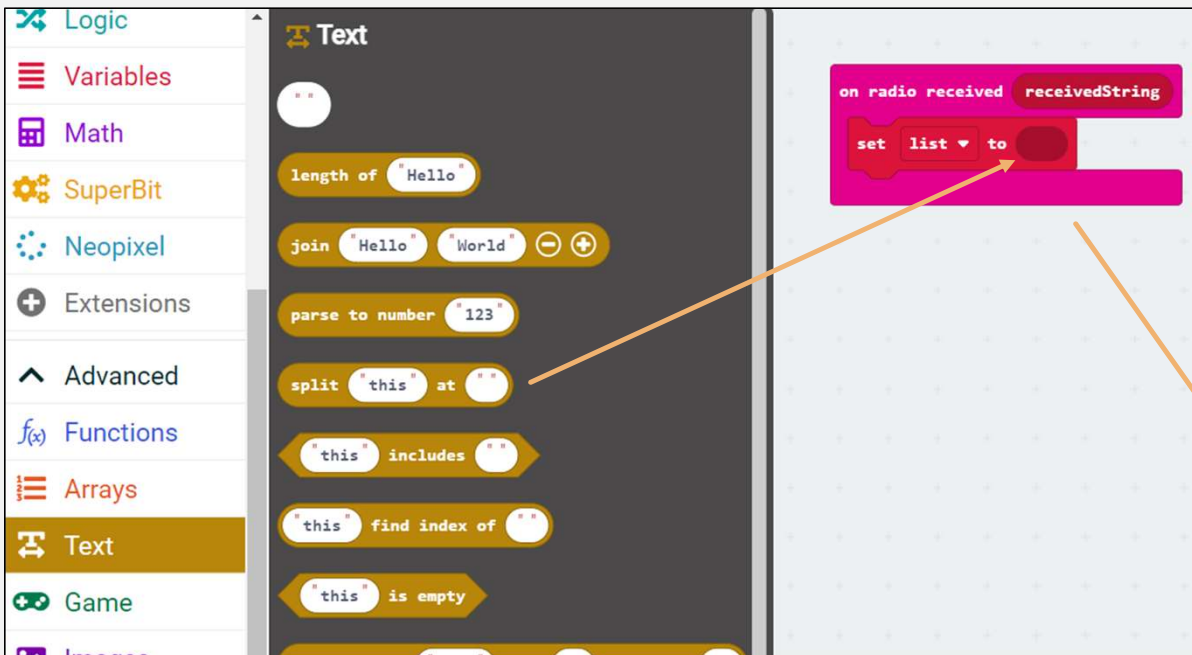
The screenshot shows the Microbit code editor interface. On the left, a sidebar lists various categories: Logic, Variables, Math, SuperBit, Neopixel, Extensions, Advanced, Functions, Arrays (highlighted), Text, Game, Images, and Pins. The 'Arrays' category is expanded, showing several blocks: 'set list to array of 0 1', 'set text list to array of 'a' 'b' 'c'', 'empty array', 'length of array list', 'list get value at 0', 'list get and remove value at 0', and 'get and remove last value from list'. In the workspace, an 'on radio received receivedString' block is connected to a 'set list to array of 0 1' block. A red arrow points from the 'set list to array of 0 1' block in the 'Arrays' category to the 'set list to array of 0 1' block in the workspace.

To unpack the message, we need to know where to put the message.

As we are separating with the “-”, we will store it temporary into a list.

You can find the set list in the Arrays block category.

Unpack Message

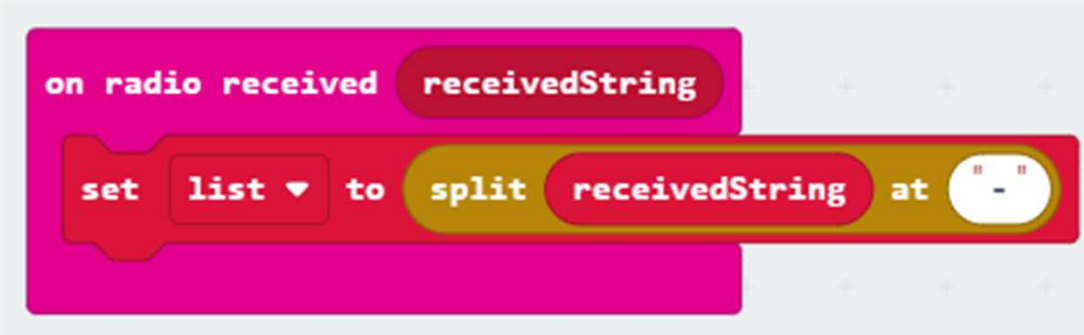


Remove the array in the list, then go to “Text” block category.

Find the split “this” at “” block and put into the set list block.



Unpack Message



We want to split the receivedString at "-", arrange your block as shown in the left.

Now your string should have 2 values in it with index 0 and 1.

Setup the variables for movement and action

```
on start
  radio set group 1
  Servo(270°) num S5 value 0
  Servo(360°) num S1 pos forward value 0
  Servo(360°) num S2 pos forward value 0
  set action to ""
  set movement to ""
```

Now we need to create variable for action and movement.

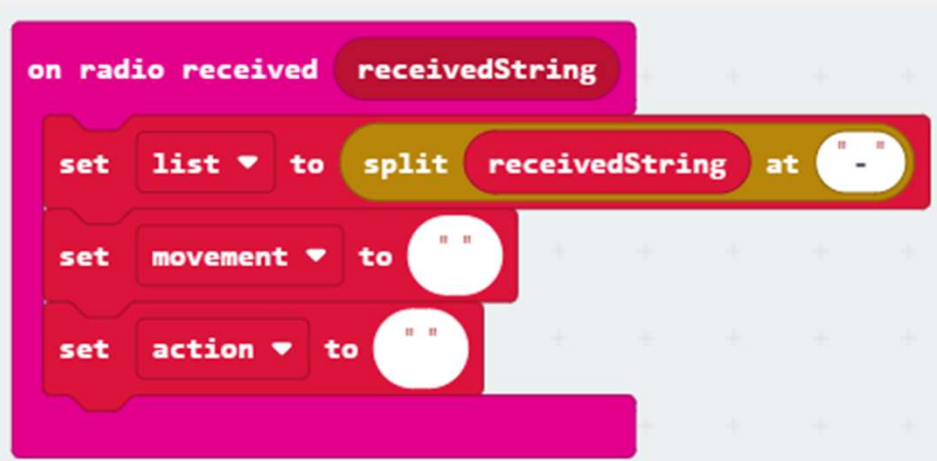
Do it as same as how you did with the remote controller.

Setup the variables for movement and action

```
on start
  radio set group 1
  Servo(270°) num S5 value 0
  Servo(360°) num S1 pos forward value 0
  Servo(360°) num S2 pos forward value 0
  set action to "lower"
  set movement to "stop"
```

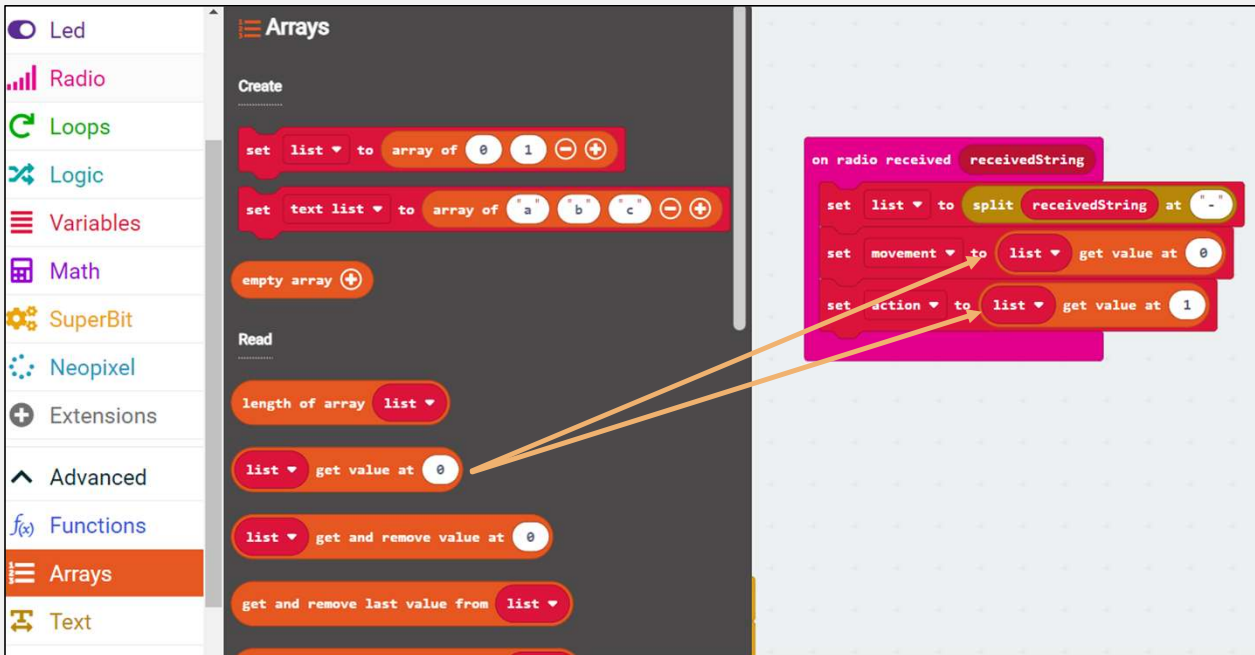
And we will give a default value with "lower" in action, and "stop" in movement variables.

Split the list and change the variables



Now we have 2 values in the list. We want to assign them to the action and movement accordingly.

Split the list and change the variables



The screenshot shows the Microbit IDE interface. On the left, a sidebar lists various components like Led, Radio, Loops, Logic, Variables, Math, SuperBit, Neopixel, Extensions, Advanced, Functions, Arrays, and Text. The main workspace is titled 'Arrays' and contains several blocks. A 'Create' section has two 'set list to array of' blocks: one for an empty array and another for an array containing 'a', 'b', and 'c'. A 'Read' section has several 'list' blocks: 'length of array', 'get value at', 'get and remove value at', and 'get and remove last value from'. On the right, a code block is shown with the following logic: 'on radio received receivedString' triggers a 'set list to split receivedString at ""' block. This is followed by 'set movement to list get value at 0' and 'set action to list get value at 1'. Two orange arrows point from the 'list get value at 0' block in the code to the 'list get value at 0' block in the 'Read' section of the IDE.

In the Arrays block, we can read the value from the list using the “list” get value at 0 (index).

The index 0 consist of the information of movement, and the index 1 consist of the information of action.

Let's assign both to the variables accordingly.

Add in the conditions for response

```
on radio received receivedString
  set list to split receivedString at "-"
  set movement to list get value at 0
  set action to list get value at 1
  if "" = "forward" then
  else if "" = "backward" then
  else if "" = "left" then
  else if "" = "right" then
  else if "" = "stop" then
```

Now you can compare the information in movement for the movement functions.

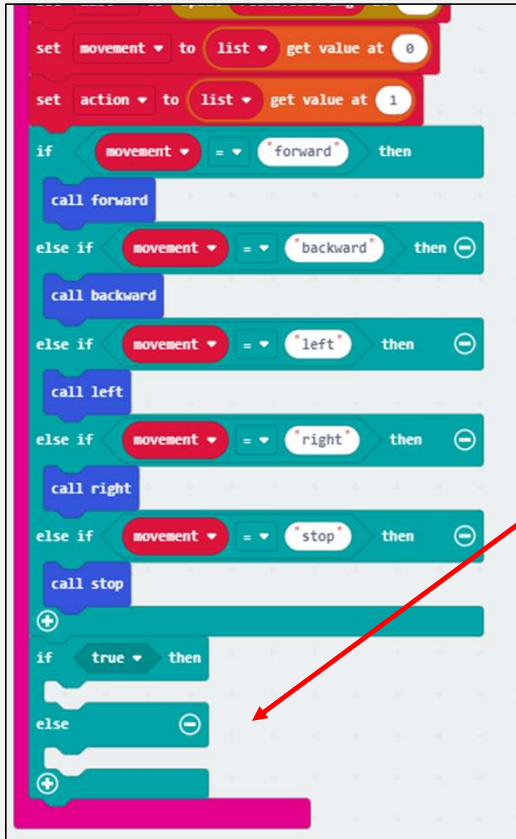
You need to use string comparison operator (" " = " ") to complete this coding.

Movement functions

```
on radio received receivedString
  set list to split receivedString at ""
  set movement to list get value at 0
  set action to list get value at 1
  if movement = "forward" then
    call forward
  else if movement = "backward" then
    call backward
  else if movement = "left" then
    call left
  else if movement = "right" then
    call right
  else if movement = "stop" then
    call stop
```

Your movement conditions are set, it should look like the left.

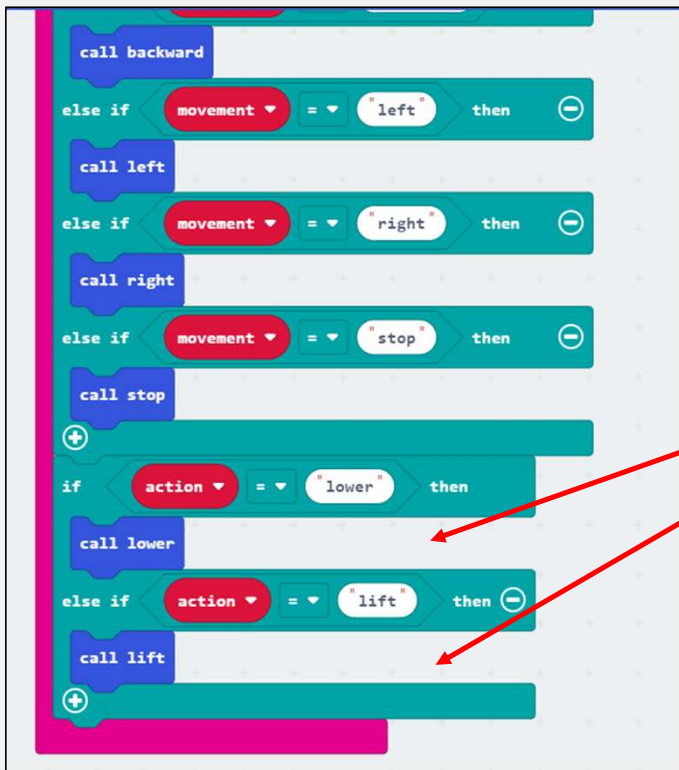
Action functions



```
set movement to list get value at 0
set action to list get value at 1
if movement = forward then
  call forward
else if movement = backward then
  call backward
else if movement = left then
  call left
else if movement = right then
  call right
else if movement = stop then
  call stop
+
if true then
  else
  +
```

Drag another if else statement for your action function.

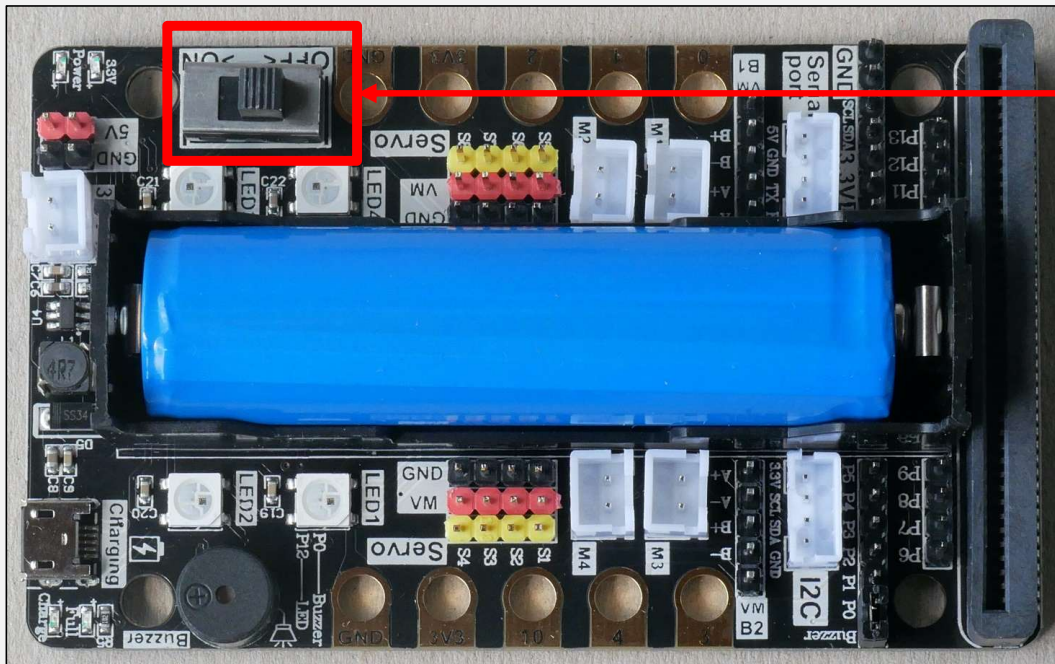
Action functions



```
call backward
else if movement = left then
  call left
else if movement = right then
  call right
else if movement = stop then
  call stop
+
if action = lower then
  call lower
else if action = lift then
  call lift
+
```

Add in the condition for both lower and lift after the movement conditions.

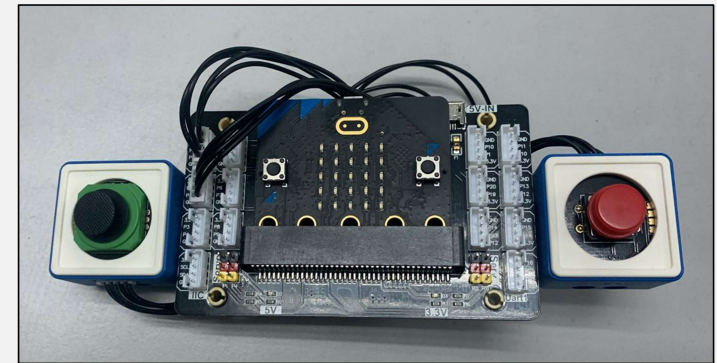
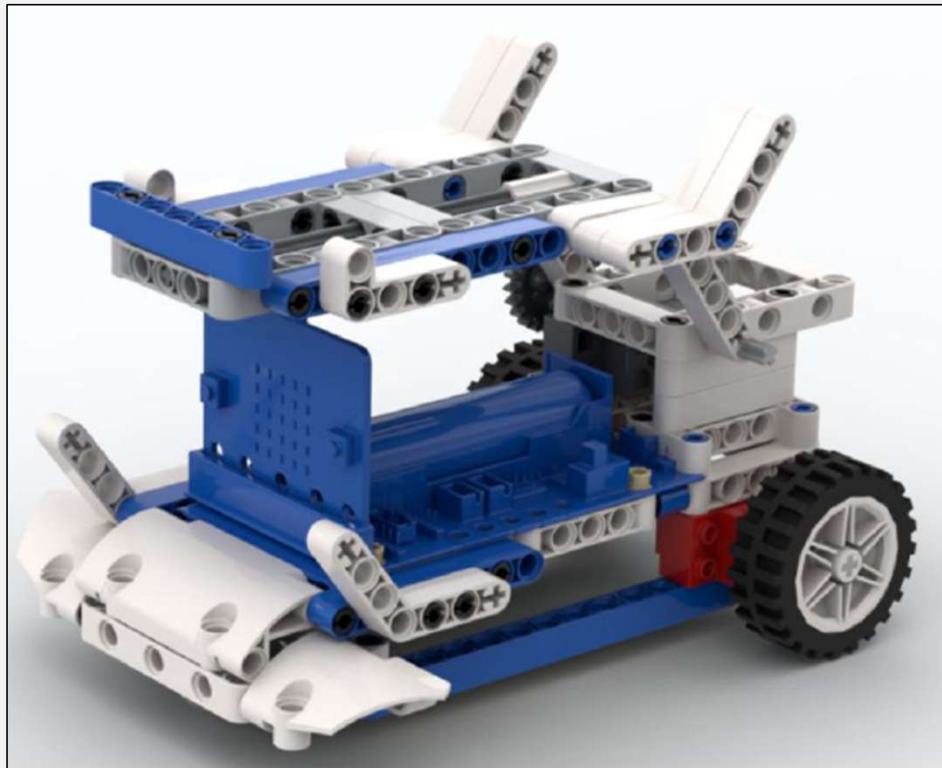
MakeCode Programming



After the program is downloaded, turn on the SuperBit with the power switch to turn on your **Skip Car**.



Phenomenon



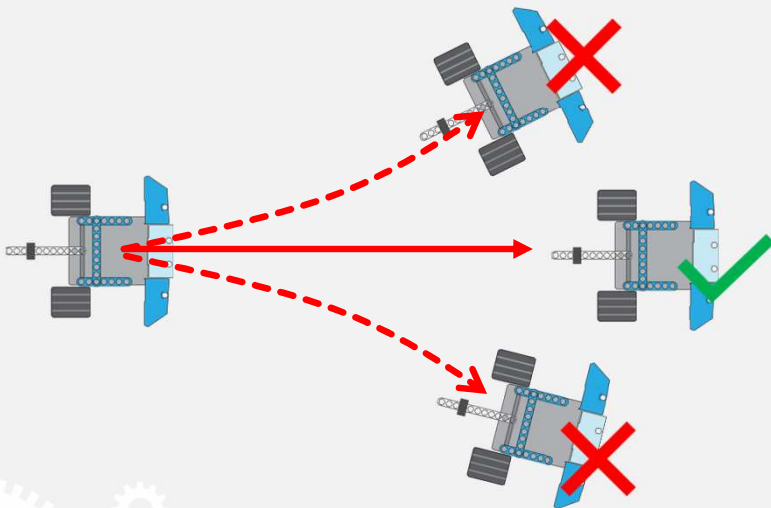
Connect your remote controller to your laptop to power on and try to interact with the Rocker module and the button module.

30 Points

CHALLENGE

for : Lesson 6

L6 – Challenge 1

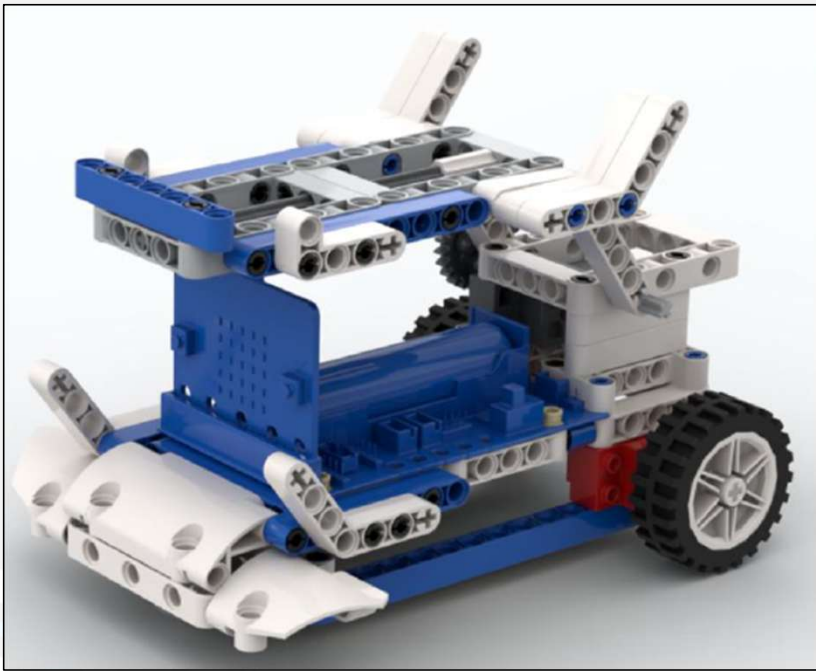


Program your Proficient Carrier to have full movement functions:

1. Move Forward (Tune it to be straight)
2. Turn Right
3. Turn Left
4. Move Backward (Tune it to be straight)

30 Points

L6 – Challenge 2

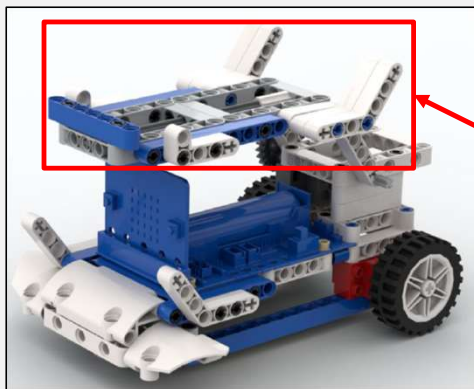


Tune your lift and lower functions for your skip car.

Don't make your S5 servo 270° overturn, it will make your code hang and stop functioning.

30 Points

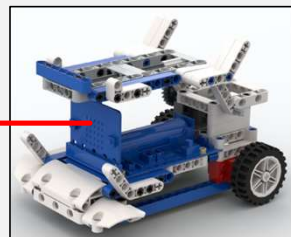
L6 – Challenge 3



Modify your skip car's handle to make sure the bricks carrier won't fall from it.

And then carrier some bricks and send the bricks to the destination, then eject the bricks to the destination.

Repeat this process to send 3 piles of bricks to the destination.



30 Points