
4tronix bit:bot control over radio

This is the fifth part of our tutorial, and assumes that you've already completed Part 1 through to Part 4. You'll also need a second micro:bit to act as the remote control for your Bitbot. We'll be reading the angle of tilt from a second micro:bit, and sending it to the micro:bit in the Bitbot over radio. More tilt is translated to the Bitbot motor speeds - this gives better control than using fixed speeds, and will allow more accurate positioning for pushing or grabbing objects later.

Since we will be programming two micro:bits, it may be useful to open <https://makecode.microbit.org/> in two separate tabs, so you can switch between both sets of code.

In this tutorial, you'll learn how to:

1. Read the accelerometer to get pitch (tilting forward and back) and roll (tilting side to side) values.
2. Send two different values over radio, and translating these into movements in our Bitbot robot.
3. Map angles of tilt in degrees, to the required range of values for motor speeds.

You will need:

- Two (2) micro:bits - one for the Bit:bot and one to act as a remote control.
- A battery pack or similar to power the controller micro:bit.

And the finished sending (controller) micro:bit code is available at:

<https://makecode.microbit.org/6csexYCTufvC>

You can find the finished receiving Bitbot code for this project at:

<https://makecode.microbit.org/Rra98JccCMER>

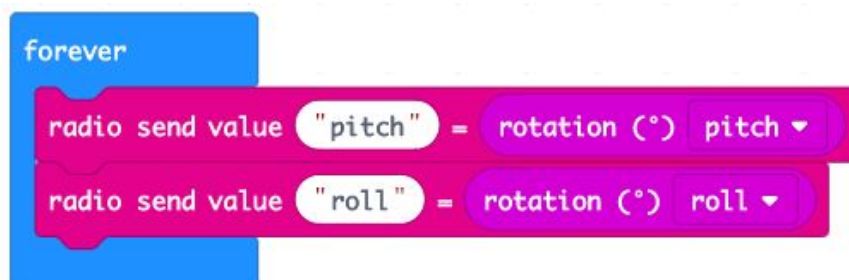
Send the tilt values over radio [controller micro:bit]

- ❑ First we need to set the radio channel we want to use on our controller. We can have up to 255 different channels, so you could easily set this same project up on two separate Bit:bots for competing against each other e.g. pushing objects around etc.



- ❑ Now we need to continuously send the tilt data over radio. We can separate the tilt data into pitch (forward/backwards) and roll (side to side). In order to do this, we'll use the **radio send value** block with name and value pairs, so that we can recognise the values when they are received by the Bitbot.

Hint: to type "pitch" and "roll" do a search for "text" and you'll find a bubble that will let you add text instead of a number.

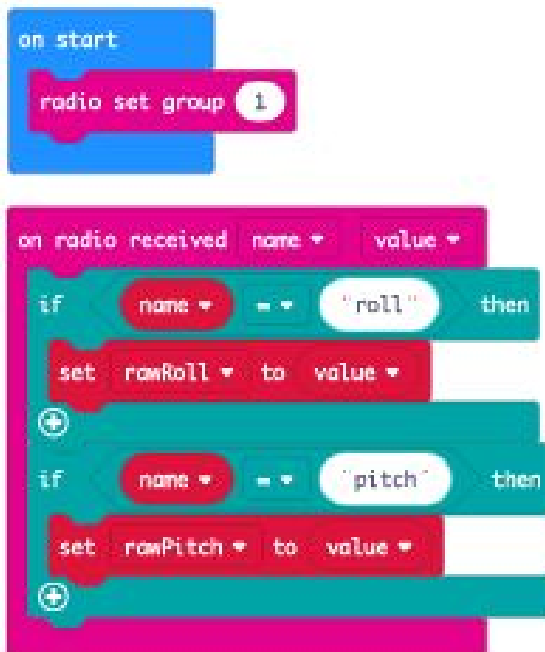


- ❑ Write this code to your controller micro:bit and put it aside. Now we are ready to program the micro:bit that sits in the Bit:bot robot.

Receive the tilt data values [Bit:bot micro:bit]

- Again, we'll set our radio group to match the radio channel used to send data from our controller micro:bit. Instead of sending this time, we'll put our code in the **on radio received** block and use an **if..then** block to decide which data is 'pitch' and which is 'roll'.

You'll also need to create two variables called 'rawRoll' and 'rawPitch' - we'll store the received data in these.

A screenshot of Scratch code blocks. The first block is a blue 'on start' block containing a pink 'radio set group' block with the value '1'. The second block is a pink 'on radio received' block with 'name' and 'value' dropdowns. It contains two 'if' blocks: the first checks if 'name' is 'roll' and sets 'rawRoll' to 'value'; the second checks if 'name' is 'pitch' and sets 'rawPitch' to 'value'. There are also two empty teal blocks with plus signs below the second 'if' block.

```
on start
  radio set group 1

on radio received name value
  if name == "roll" then
    set rawRoll to value
  if name == "pitch" then
    set rawPitch to value
```

Convert 'roll' data into left and right motor speeds [Bit:bot micro:bit]

- ❑ The roll or pitch data is an angle, so we need to check whether the controller micro:bit is tilted:

- right (roll greater than 10 degrees)
- left (roll less than -10 degrees)

If the answer is left, the value is going to be negative and this would make the motor go the wrong way, so we multiply it by -1.

When we write this code into a **forever** block, it looks like this:



You'll need to create two more variables: **rightMotor**, and **leftMotor** for the above code.

- ❑ These variables are used to hold the speed we want each motor to run at. But before we send these to the motors, we should check if the micro:bit controller is not tilted left or right. Although this should be 0 degrees, we're human so we'll say between -10 and 10 degree just to make the controls a bit easier.

If the controller is not tilted left or right, then we'll use the pitch (forward/backward) to get the speed and direction to more in a straight line.

To check whether the roll value is between -10 or 10, we need another **if..then** block with the condition shown below (put it in the **forever** loop, just below the other two **if** blocks):



- ❑ To move forwards in a straight line, we just set both **motorRight** and **motorLeft** values to the **rawPitch** value. Remember, the **rawPitch** value is the angle that the controller micro:bit is tilted forwards or back (we multiply by -1 because it makes more sense that tilting the micro:bit controller forward makes the Bitbot go forwards).



Mapping the tilt angle to motor speeds [Bit:bot micro:bit]

- ❑ We are going to use a **map** block to convert the **motorLeft** and **motorRight** to motor speeds. For this, we'll use the **map** block when setting the **motorRight** and **motorLeft** variables. You'll see that we are converting a value in the range 0-180 degrees into a useful value for our motor speeds between 0-1024.
- ❑ To set the motors to run at the mapped **motorLeft** and **motorRight** speeds, we use the drive motor at..speed blocks. The code below shows these blocks, which go in our **forever** loop, but below and not inside any **if..then** blocks:



Although these values are what we found to work for us, you may want to play with the numbers to get a more or less responsive robot. You can also add additional commands to your controller, such as pressing the A and B buttons to operate the LEDs.

We'll leave these to you as an additional challenge!

The complete code listings

The complete code for the **controller micro:bit** is as follows:

```

on start
  radio set group 1
  show icon [robot]

forever
  radio send value "pitch" = rotation (°) pitch
  radio send value "roll" = rotation (°) roll
  
```

The complete code for the **receiving Bitbot micro:bit** is as follows:

```

on start
  radio set group 1

on radio received name value
  if name = "roll" then
    set rawRoll to value
  if name = "pitch" then
    set rawPitch to value

forever
  if rawRoll < -10 then
    set rightMotor to rawRoll x -1
    set leftMotor to 0
  if rawRoll > 10 then
    set leftMotor to rawRoll
    set rightMotor to 0
  if -10 < rawRoll and rawRoll < 10 then
    set leftMotor to rawPitch x -1
    set rightMotor to rawPitch x -1

  map leftMotor
    from low 0
    from high 180
    to low 0
    to high 1024
  drive left motor(s) at speed

  map rightMotor
    from low 0
    from high 180
    to low 0
    to high 1024
  drive right motor(s) at speed
  
```

