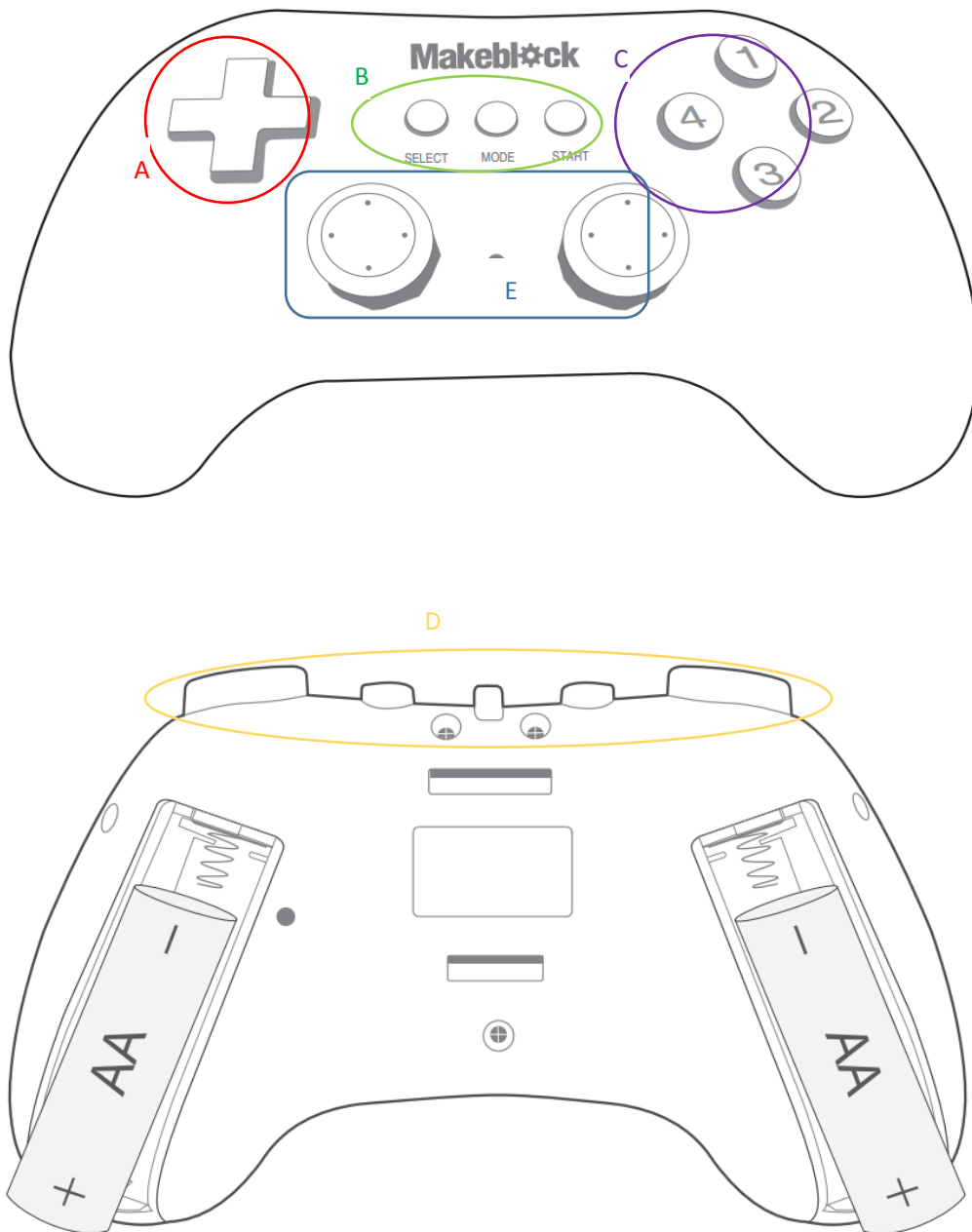


# Guide: 2.4G Wireless Controller

## Learn about the Controller

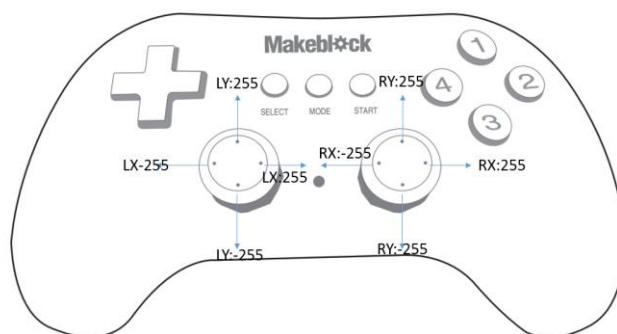
The 2.4G Wireless Controller has 5 main areas (just like A-E below), which can be divided into two different ways of input.



The buttons in A/B/C/D areas can only input bool numbers, which means they can only tell if they are triggered or not. These buttons are controlled by the block below.

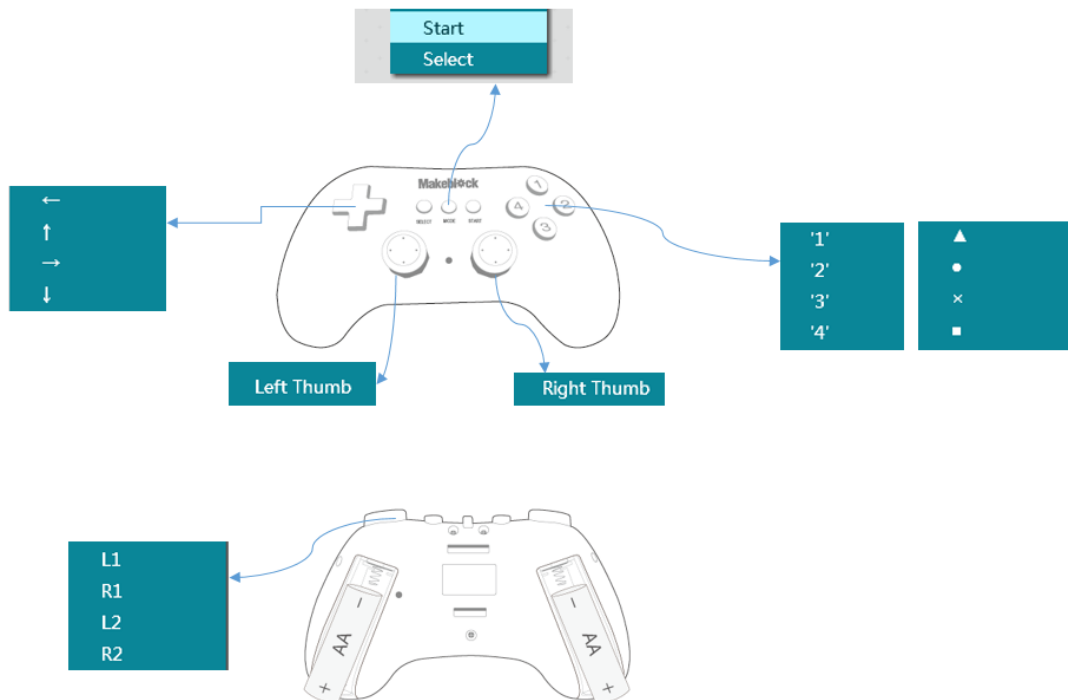


As for the thumbs in area E, they can input analog value from -255 to 255 (step by 1). The reporter block for thumbs is in the picture below.



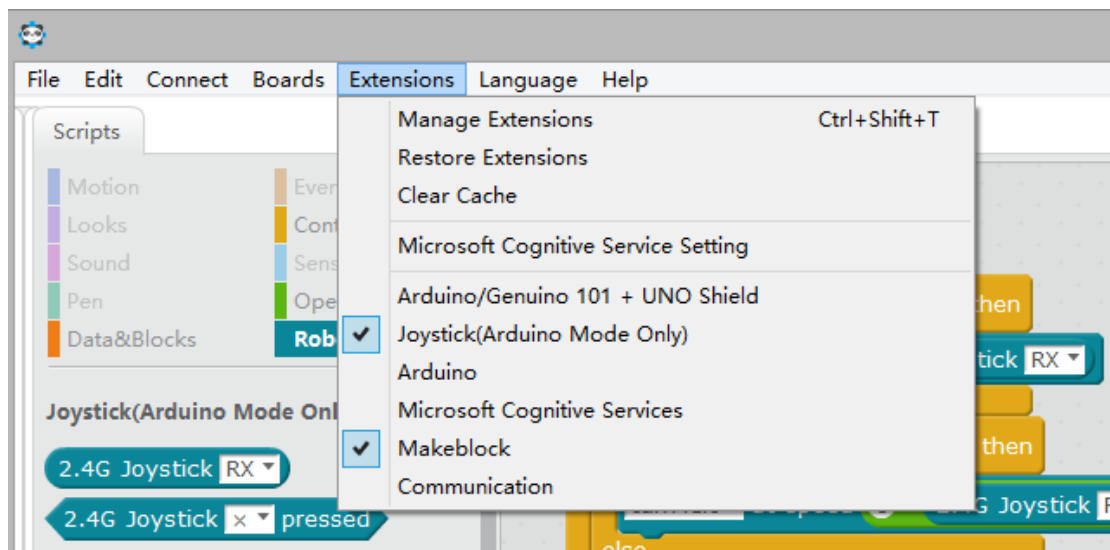
RX/RY respond to the right thumb, LX/LY is for the left thumb. X stand for X-axis and Y stand for Y-axis, which means if you move the right thumb to its left, RX turns out to be -255.

## Correspondence



## Program the Controller Buttons

First of all, we need to connect to the main board with mBlock. Then we should choose **Joystick (Arduino Mode Only)** in the Extensions.

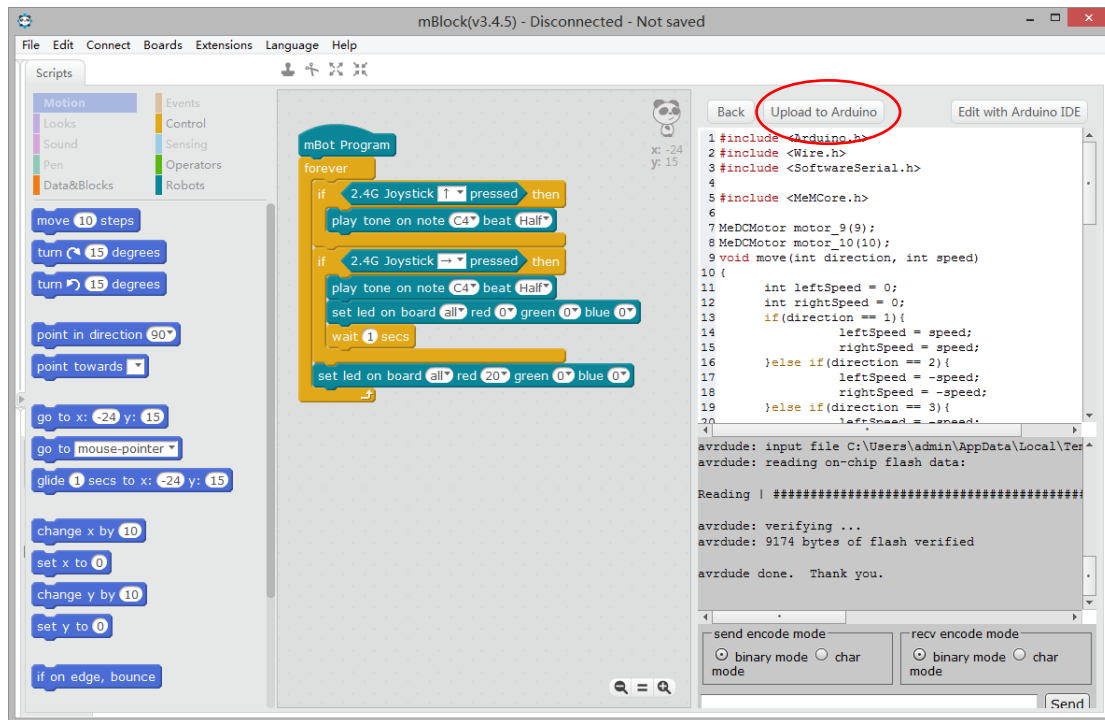


Now we can use mBlock to program the Controller buttons.

## Programming Examples

Example a:

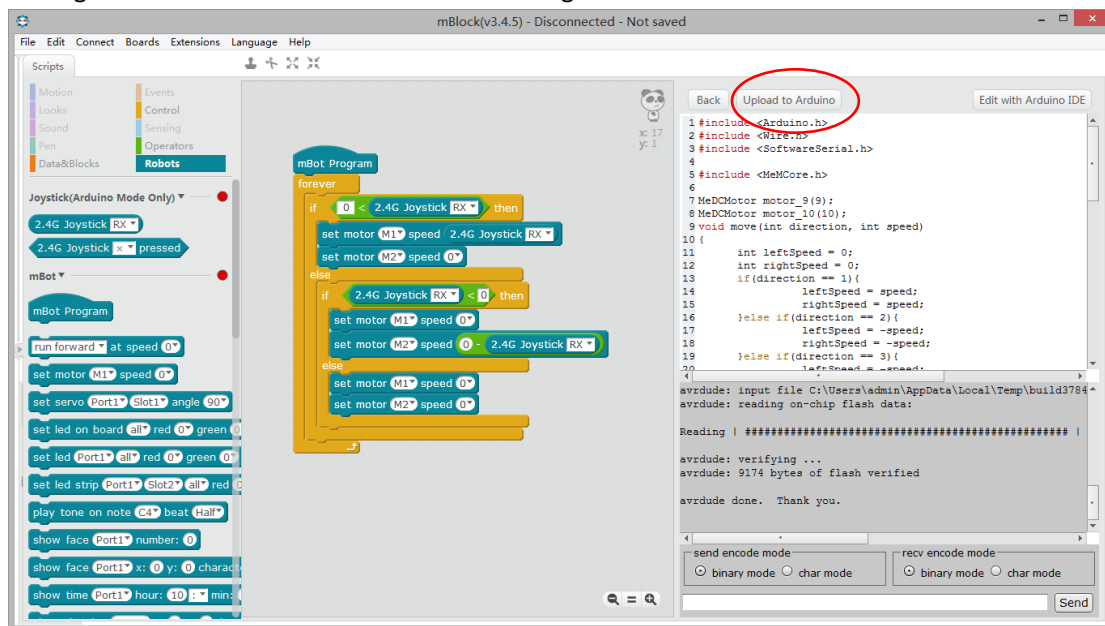
Use buttons of the Controller to play sound.



The screenshot shows the mBlock IDE interface. The central workspace contains a script for an mBot program. It starts with a 'forever' loop. Inside the loop, there are two 'if' blocks. The first 'if' block checks if the '2.4G Joystick 1' button is pressed; if true, it plays a tone on note C4 with a half-beat duration. The second 'if' block checks if the '2.4G Joystick 2' button is pressed; if true, it plays a tone on note C4 with a half-beat duration, sets all LEDs on the board to red (0), green (0), and blue (0), waits for 1 second, and then sets all LEDs to red (20), green (0), and blue (0). The right-hand pane shows the compiled C++ code, with the 'Upload to Arduino' button circled in red. The code includes headers for Arduino, Wire, SoftwareSerial, and MeMCore, and defines two MeDCMotor objects. A 'move' function is defined to handle different directions (1, 2, 3) by setting left and right motor speeds. The terminal output shows the upload process: 'avrduide: input file C:\Users\admin\AppData\Local\Temp\build3784', 'avrduide: reading on-chip flash data:', 'Reading | #####', 'avrduide: verifying ...', 'avrduide: 9174 bytes of flash verified', and 'avrduide done. Thank you.'

Example b:

Use right thumb to control mBot to turn left or right.



The screenshot shows the mBlock IDE interface. The central workspace contains a script for an mBot program. It starts with a 'forever' loop. Inside the loop, there are three 'if' blocks. The first 'if' block checks if '0 < 2.4G Joystick RX'; if true, it sets motor M1 speed to 2.4G Joystick RX and motor M2 speed to 0. The second 'if' block checks if '2.4G Joystick RX < 0'; if true, it sets motor M1 speed to 0 and motor M2 speed to -2.4G Joystick RX. The third 'if' block checks if '2.4G Joystick RX < 0'; if true, it sets motor M1 speed to 0 and motor M2 speed to 0. The right-hand pane shows the compiled C++ code, with the 'Upload to Arduino' button circled in red. The code includes headers for Arduino, Wire, SoftwareSerial, and MeMCore, and defines two MeDCMotor objects. A 'move' function is defined to handle different directions (1, 2, 3) by setting left and right motor speeds. The terminal output shows the upload process: 'avrduide: input file C:\Users\admin\AppData\Local\Temp\build3784', 'avrduide: reading on-chip flash data:', 'Reading | #####', 'avrduide: verifying ...', 'avrduide: 9174 bytes of flash verified', and 'avrduide done. Thank you.'

Please remember to click 'Upload to Arduino'! And now we can use Controller to control mBot!