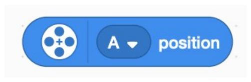
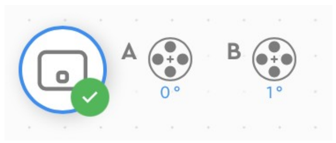
# Descrete Rotary Control

Sometimes you want to select something with a control that you turn. You can use the Lego set to control something, say a light color, brightness, number of lights lit, etc.

The Lego system can sense the rotation angle of the motor. Your task is to convert that angle into a “descrete” control that selects one of several possible options.



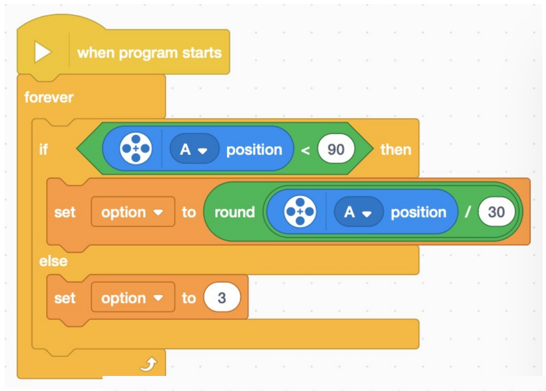
First some basics. The motor reports its position as an angle from 0° to 359° which increases clockwise. The zero position is when the 0 on the motor body is aligned to the 0 on the turning part of the motor.



You can monitor the position of each motor by a display at the upper left corner of your programming window.

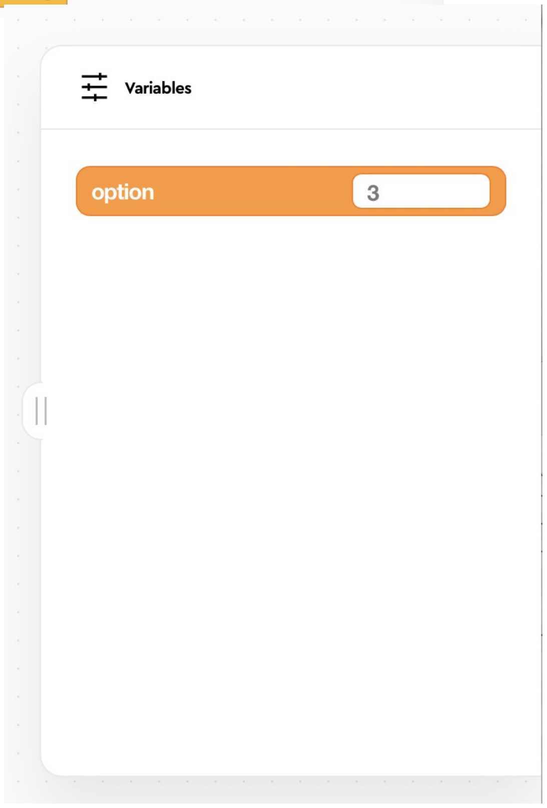
You decide how many options to build. You can start with a small number and make it bigger after you get the basic code working. Decide how far you want to turn the control: just 90° for tighter control or 360° for all the way around. Divide that by the number of options. So 3 options in 90° would be 30° for each option. So you divide the motor position by 30 and then round it to remove the fractional decimal part.

The code would look like:



```
when program starts
forever
    if A position < 90 then
        set option to round A position / 30
    else
        set option to 3
```

So what is going on here.



Variables

option 3

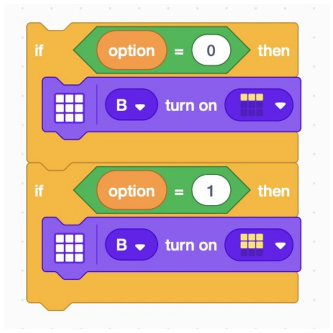
First we **create a variable** called “option”. This allow us to monitor which option is selected with the **Variables** pull out panel on the right side of the programming window. (Look for tab with two vertical bars.) That does not directly show up as code, but it enables the **set** blocks which do show up in the code twice.

The **forever** block is used to check the motor continuously. This lets the option get updated at the control (the motor) is turned.

The **if** block is used to make a choice. In this case we are checking for the maximum allowed angle of 90. If it is less than 90 we use a **set** block set the value of option to the angle divided by 30, otherwise (else) we just use another **set** block set value of option to 3.

You can turn the motor and select 0, 1, 2, or 3. Hey! That is four options, and we only wanted three. The value 3 can either be ignored or treated as one of the other values (usually 0 or 2 in this case). The leaves the “out of bounds” value of 3 to be handled or ignored as a design choice.

So now you know how to write code to select a discrete option. You now can add code using **if...then...** blocks to use the selected option to control something, say the number of LEDs lit up in the LED matrix. Code to do that would  like the following code block. Each **if...then** block handles one option, so you need one block for each option in your design.



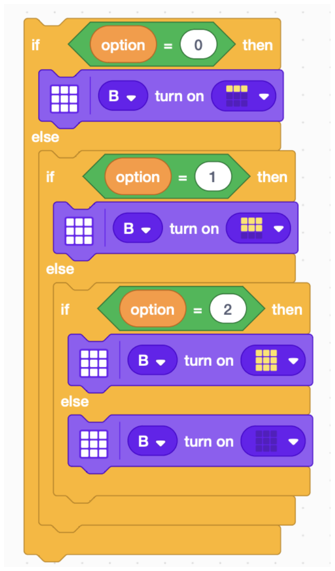
```
if option = 0 then
    B turn on
if option = 1 then
    B turn on
```

See if you can put the two concepts together to use a motor to control the LED matrix display.

- use as many options as you want.
- control the LED matrix patterns and colors as desired.
- control the LED matrix color.
- make the control easier to turn.

## Something slightly different

Another way of expressing the same logic is to use **if...then...else...** block.



```
if option = 0 then
    B turn on
else
    if option = 1 then
        B turn on
    else
        if option = 2 then
            B turn on
        else
            B turn on
```

How is this different than the previous example?

What are its advantages?

What are its disadvantages?

Did you try it or just guess?

## Extra Credit—Two at Once

- Use two motors: one to select a discrete volume and the other to select a sound to play. Tap on the hub to get the sound to play. (Use two start blocks to make them truly independent.)
- Use two motors: one to select a discrete flashing rate and other to change the color of the Power button LED on the hub.
- Use two motors: one to select a set of discrete angles, and other other to turn to that angle.