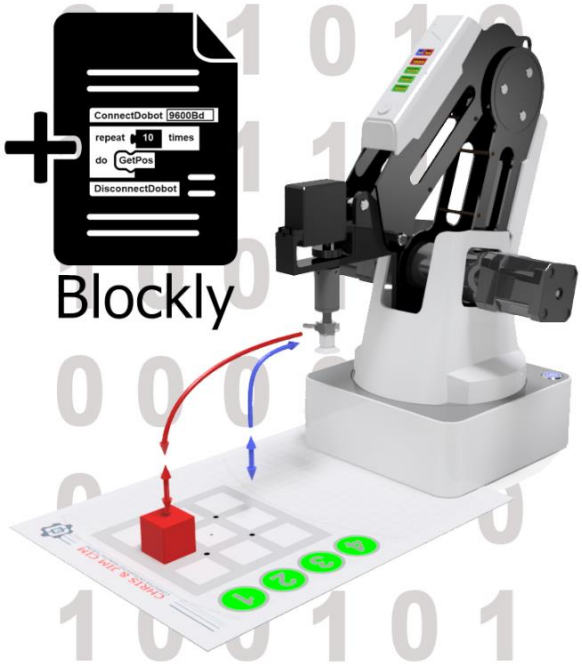
3 Blockly - Pick and Place with Inputs

NAME: _____ *Date:* _____ *Section:* _____

INTRODUCTION

Often it is necessary for a robotic arm to wait for another machine or process to finish before moving with its program. This can be done by adding the ability for a robot to read *input* values.

In this activity you will learn how to add an input to a basic Pick and Place operation in blockly. Through this you will learn how to program the robot to move, turn on it's suction cup, and how to set up *Inputs* in Blockly.



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Forever Loop
- While Loop
- Placeholder
- Multiplexing
- Inputs
-

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cubes or cylinders
- DobotStudio software
- Suction Cup Gripper
- *Dobot Input/Output Guide*

ESSENTIAL QUESTIONS

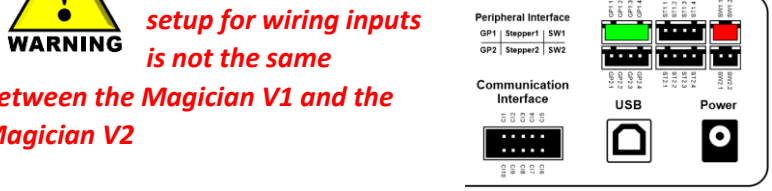
Essential questions answered in this activity include:

- What's the difference between an *input* and an *output* in robotics?
- What are some examples of *digital inputs*?
- How do I code a digital switch as an *input* in blockly?

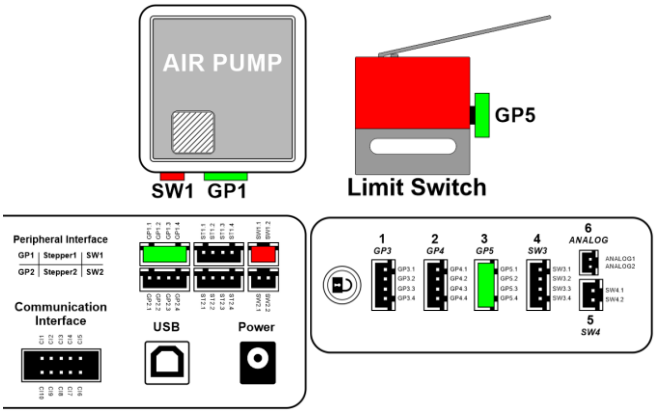
PROCEDURE

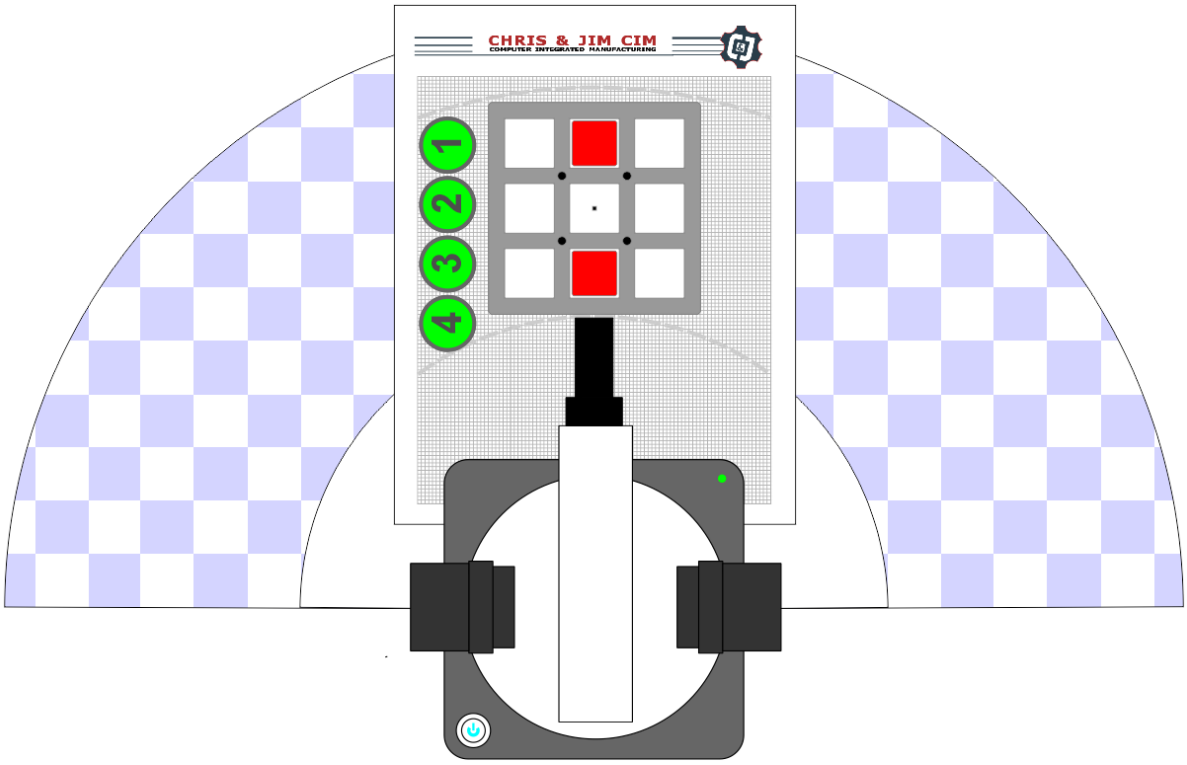
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Print the Blockly - Pick and Place Field Diagram
2. Set up the robot with a suction cup and place a cube in one of the red squares on the field diagram provided
3. Wire the robot such that the LIMIT SWITCH is plugged into port GP5.

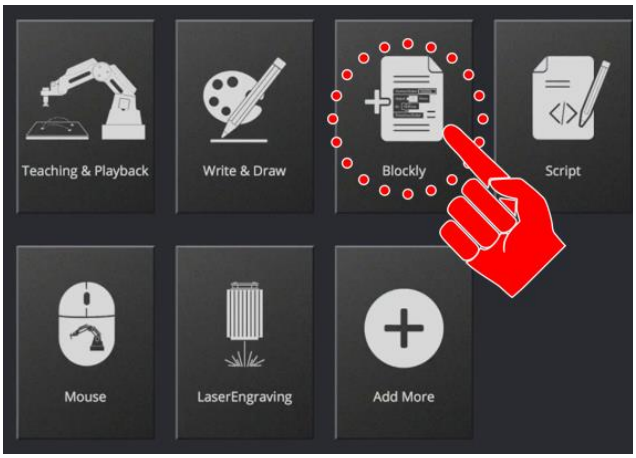


Caution: Please Refer to the Dobot Input/Output Guide for wiring your input. The setup for wiring inputs is not the same between the Magician V1 and the Magician V2





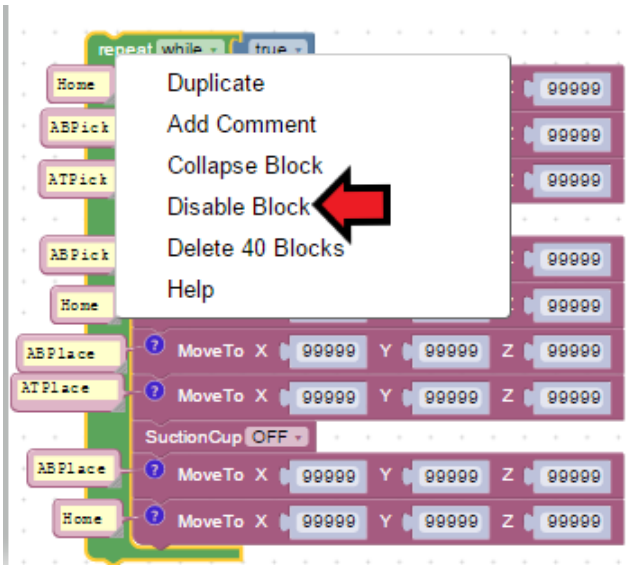
4. Open up Blockly in the software and Open your Blockly - Pick and Place file from the previous activity.



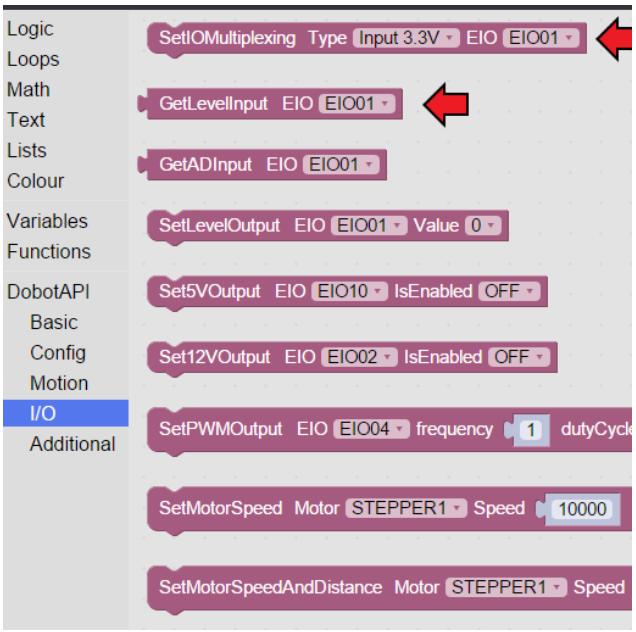
HELPFUL TIPS

When you re-open this program check that the name of the file on top matches the code in the file, if it does not, you may end up overwriting another program

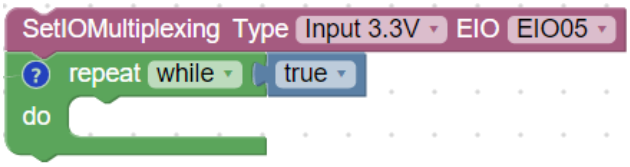
In this activity we are going to add the ability for the program to wait for an input and then have the program run whenever a limit switch is pressed. To do this, you must prepare a limit switch that can work on the Dobot, as done in Activity 5, and plug it into GP5 (PORT3). **Make sure the robot is off before you plug or unplug anything from the robot.** After that, we are going to right click our current while loop and click *Disable Block.* What this does is that it grays out the current selected block and any blocks inside of it, making the program not read it but still having it there incase you want it back later on.



We are now going to setup a program to make sure the dobot is reading the input signal from switch. This is done by having the Dobot print the value of the switch to the *Running Log* on the right of the screen. First though, we must tell the robot that there is an input we are expecting and we do this by using the block ***SetIOMultiplexing***. To see the value we use the *GetLevelInput block*



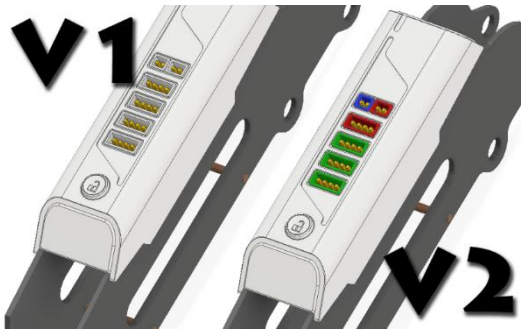
Drag a ***SetIOMultiplexing*** block over and it will be the start of this program. Change the EIO port to EIO05 which is equivalent to GP5 (PORT3), you may use another port but make sure to consult the *Dobot Input/Output Guide* as some ports may damage the Dobot or other equipment. Then create a Forever loop below it.



Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

Be sure to turn off power to the Dobot before continuing!

In order to properly wire your Dobot Magician we will first need to identify which Dobot you have.

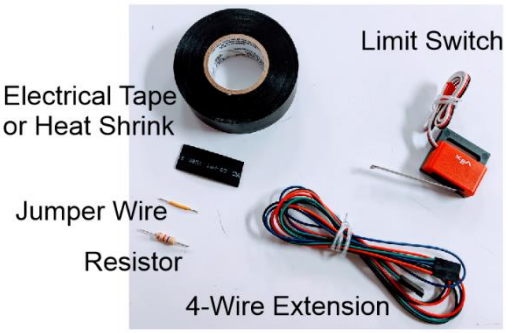


V1 = All white plastic housings for inputs and outputs

V2 = Colored plastic housings for inputs and outputs

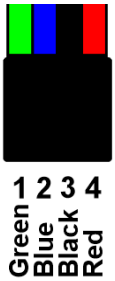
The Dobot Magician V1 needs a **PULL DOWN RESISTOR** installed on the ground side of the servo extension cable to work properly. The pull-down resistor will insure a well-defined logical level in our logic circuit. This resistor will help eliminate possible float values and will pull the input pin to a logical low state in all conditions. The pull-down resistor can have any value from 4.7KΩ to 15KΩ. The larger the resistance, the slower the input pins response will be to voltage in milliseconds.

INPUT OPTION 1 - Direct Wire



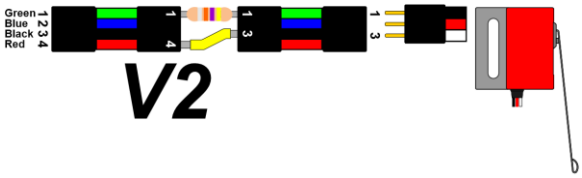
Parts List

2 x 4-Wire Extension Cables
1 x Short Jumper Wire
1 x Resistor 4.7Ω to 15kΩ
1 x Digital Switch
Electrical Tape or Heat Shrink



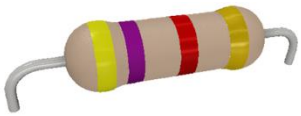
Dobot Magician V1 (All White IO Ports)

Dobot Magician V2 (Colored IO Ports)



Step 1 - Install Jumper Wire and Resistor

*The resistor need to be on the negative side to work as a pull down resistor

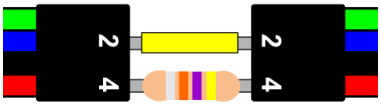


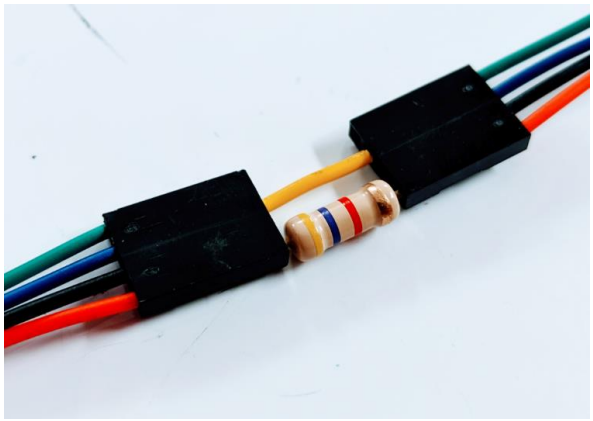
4.7k Ω - GROUND SIDE

YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

Dobot Magician V1 (All White IO Ports)

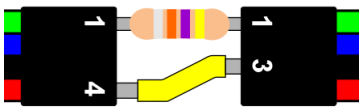
-Pull Down Resistor - 4Red to 4Red
-Jumper Wire – 2Blue to 2Blue

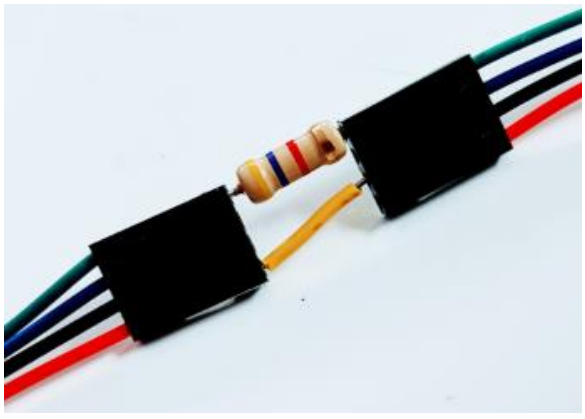




Dobot Magician V2 (Colored IO Ports)

-Pull Down Resistor – 1Green to 1Green
-Jumper Wire – 4Red to 3Black



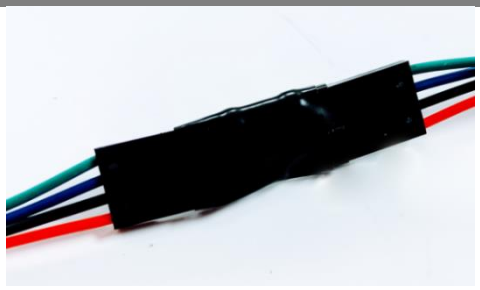


Dobot Magician V1 (All White IO Ports)	*Dobot Magician V2 (Colored IO Ports)*

Step 2 - Secure the Components

-Wrap the resistor and jumper wire with electrical tape or heat shrink to keep them from separating.

-Lightly tug on the assembly to ensure connection is stable



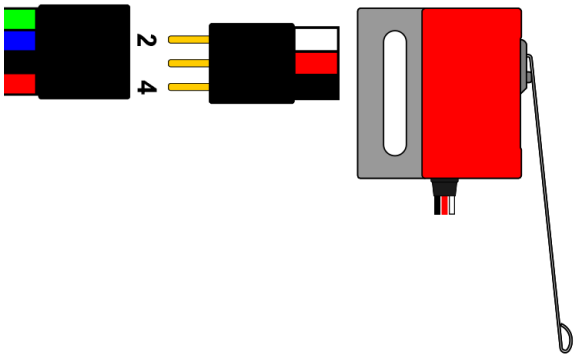
Step 3 - Install Digital Switch

**Red wire is NOT active when using a VEX limit switch*

Dobot Magician V1 (All White IO Ports)	*Dobot Magician V2 (Colored IO Ports)*

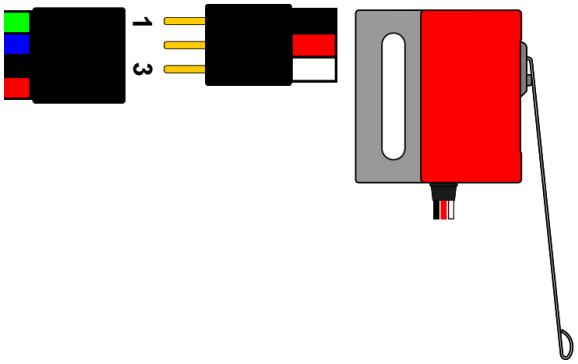
-4Red to Limit Switch Black

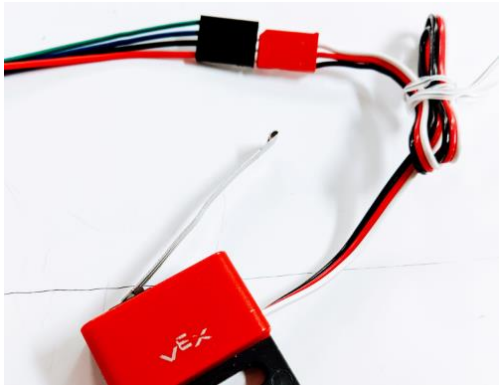
-2Blue to Limit Switch Red

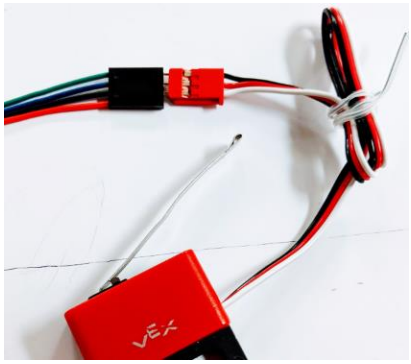


-4Red to Limit Switch Black

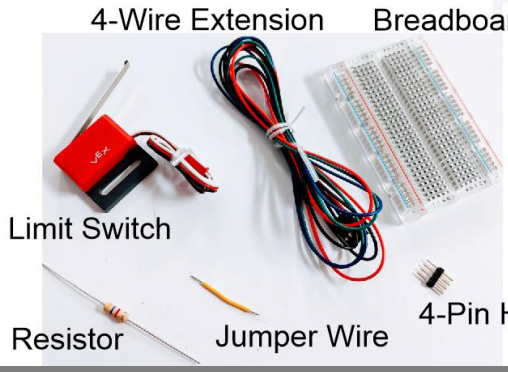
-2Blue to Limit Switch Red





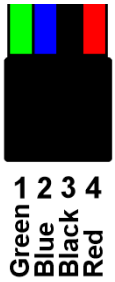


INPUT OPTION 2 - Breadboard

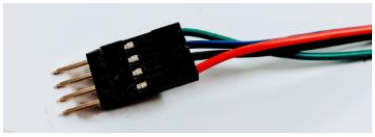


Parts List

1 x 4-Wire Extension Cables
1 x Jumper Wire
1 x Resistor 4.7Ω to 15kΩ
1 x Digital Switch
1 x Breadboard
1 x 4-Pin Male Header

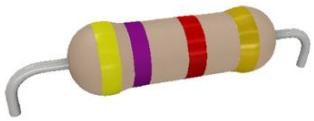


Step 1 – Install the 4-Pin Male Header
-Insert the Header pins into one end of the 4 wire extension cable. This will allow the 4-wire cable to be attached to the bread board.



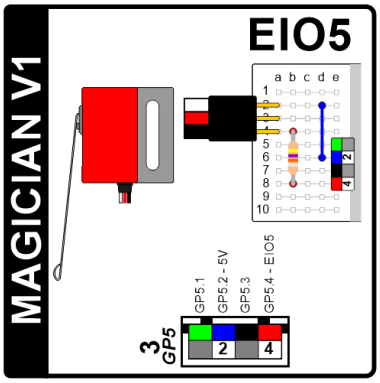
Step 2 - Install Jumper Wire, Resistor, and Extension Cable
-Insert the resistor and jumper wire into the breadboard as shown in the illustrations below.

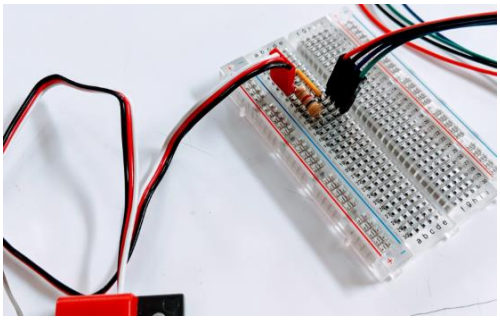
**Note that the 4-wire cable is not in the same orientation for both setups.*



Dobot Magician V1 (All White IO Ports)

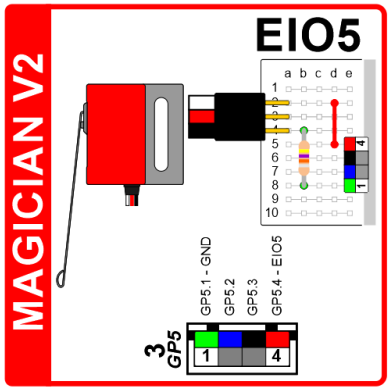
-4Red to Limit Switch Black
-2Blue to Limit Switch White

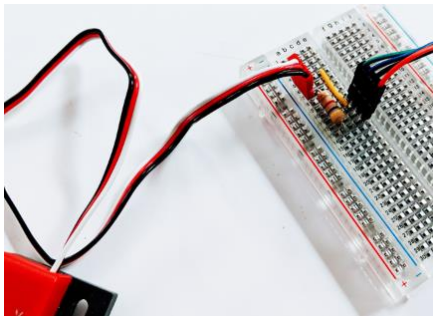




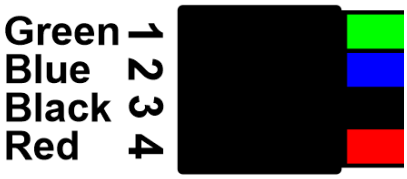
Dobot Magician V2 (Colored IO Ports)

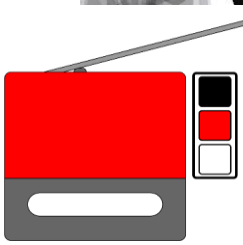
-4Red to Limit Switch Black
-2Blue to Limit Switch Red

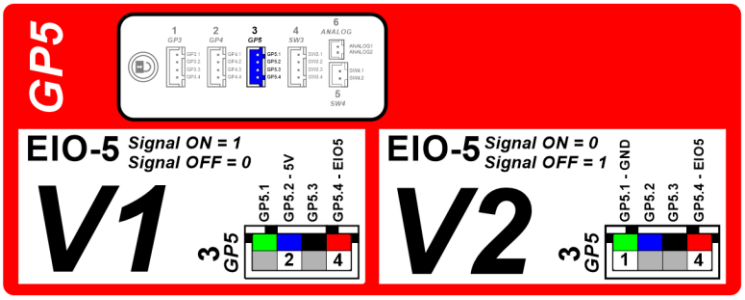




Attach the 4-Wire Extension cable to the Robot Arm

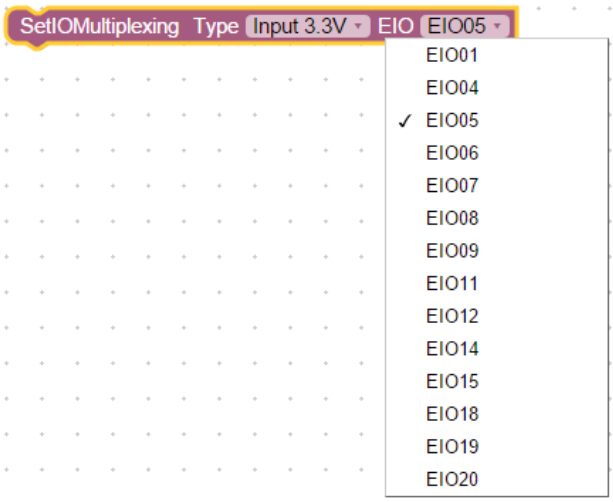


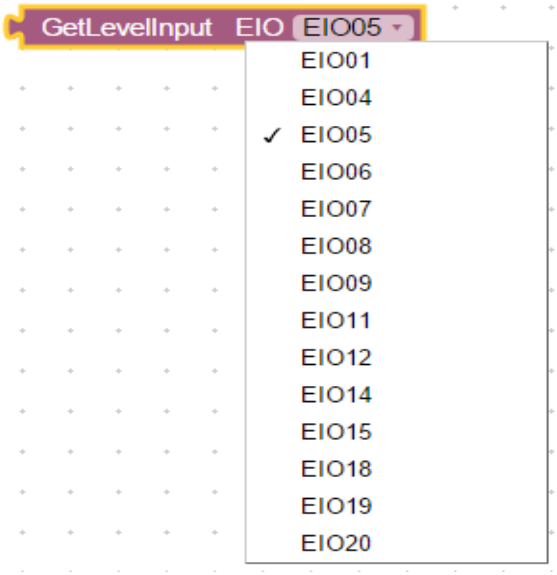




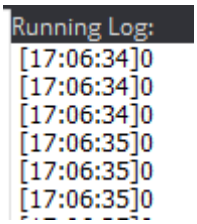
Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

Next we are going to have the program print the status of the switch. To do this we are going to need the ***print*** block, this can be found in the *text* section near the bottom. Drag the ***print*** block into the Forever loop and then drag the ***GetLevelInput*** block, that was indicated before, and drag it into the right side of the ***print*** block. Make sure to change it to the correct port in order for it to work.

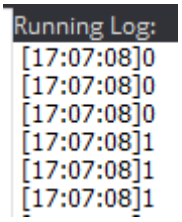




Now run the program. In the running log you should see it printing 0's or 1's while the switch is not pressed and it should swap printing to either a 1 or a 0 while it is pressed.



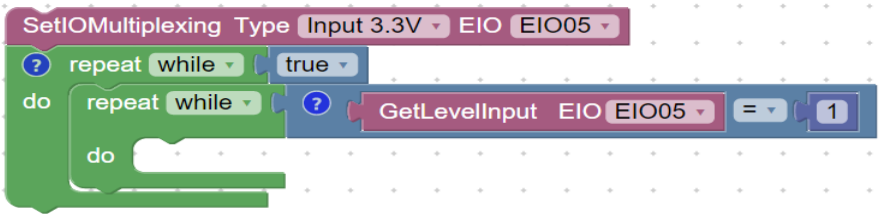
N.O. - If you see 0's that change to 1's, your switch or the Dobot is wired N.O. - Normally Open. The input will read Low or False until the switch is pressed.
N.C. - If you see 1's that change to 0's, your switch or the Dobot is wired N.C. - Normally Closed. The input will read high or true until the switch is pressed.



If this is not the case, you may be plugged into the wrong port, have the wrong port selected for the IO setup, or have the wiring for the switch incorrect. Please refer to the Dobot Input/Output Guide. for your setup and try again. ***Remember to turn the power off to the robot if you switch any wires.***

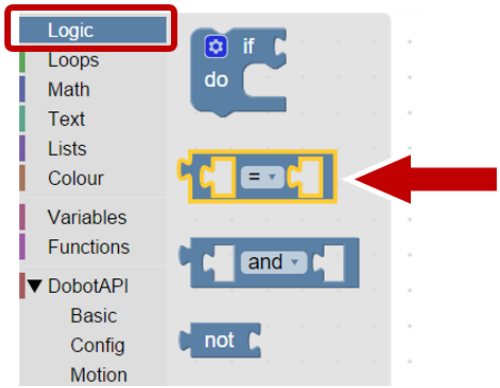
If your setup did not work correctly the first time, what did you have to do to make it work?

Now that we are able to see the switch, get rid of the ***Print*** block. We are now going to make a while loop that will run as long as the switch is pressed. Drag another ***while*** loop into the *Forever loop* but instead of dragging in a *True* block as we did previously, we are going to go to the *Logic* section and drag over a ***[blank] = [blank]*** block. This block is how we check if an item is equal, greater, lesser, or not equal to another item. In the left side of the block, drag in the *GetLevelInput* block that we just used. We then are going to go to the *Math* Section and grab a ***number*** block, and drag that into the right side of the block and change it into a 1.



Create the *Nested Loop* as seen above (Steps provided below). A *nested loop* is a loop within another loop. We will place the robot's pick and place movement commands inside the second loop that is only called to start if the robot sees a change in the input value on input EIO05. Refer to your testing to whether the robot should start the movement on a 1 or a 0.

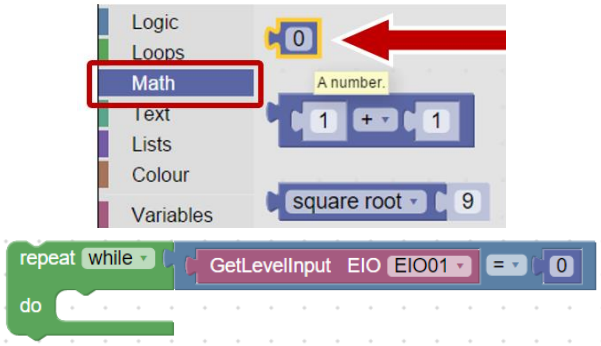
Attach a ***ReturnTrue*** from the *Logic* section. This condition will return a true value if both inputs are equal as the condition for the second while loop



Insert a ***GetLevelInput*** from the *DobotAPI* section and add that as the first condition for the ***ReturnTrue*** statement. Change the Input value to EIO05.

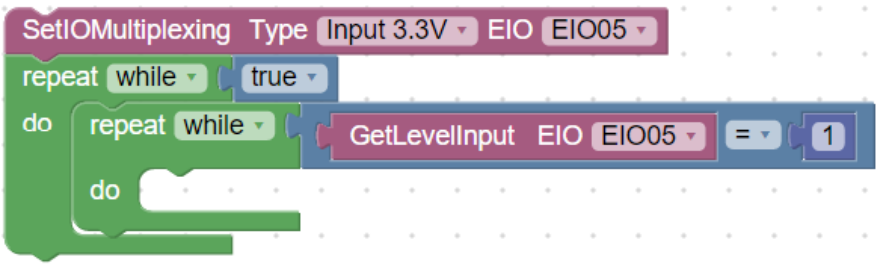
Insert an ***AddNumber*** condition as the second part of the ***ReturnTrue*** statement.

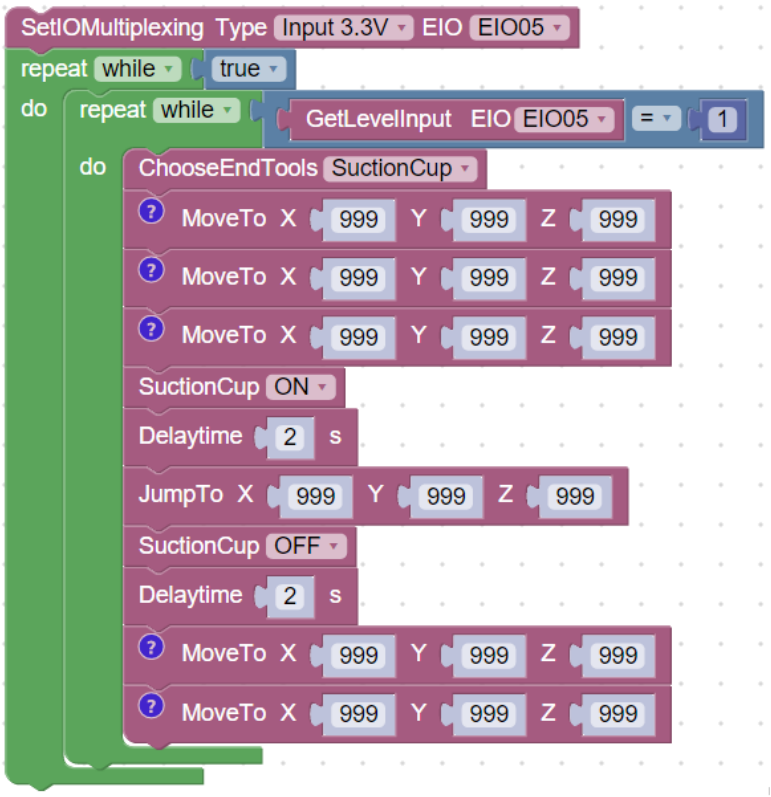
Change the value of the ***AddNumber*** to the value that is needed to start your loop (0 or 1)



Once a loop has started, it will continue to finish ALL of the steps in the loop even if the condition that started the loop becomes false. The condition to start the loop is only evaluated at the beginning of each loop

Now that the *nested loop* is setup, we want to re-enable our previous code for the pick and place routine. To do this, right click on the loop again and click *enable*. Once it is enabled, drag the movement code from the old loop into the new loop and then delete or disable the old loop. In the end it should like this but with real values instead of placeholders.





Once the program is written, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?

CONCLUSION

1. *What would happen if the loop detecting the switch wasn't in a forever loop? Try it and describe what happens. (Hint: Make sure to be holding the switch when you start the program)*

2. What happens to the Pick and Place process when the switch is released (The loop condition *becomes false)?*

3. *In your own words, define a nested loop.*

4. *In your own words, define a condition.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

_____ 1. Break the pick and place activity into multiple sections that will wait for the limit switch before performing each step.