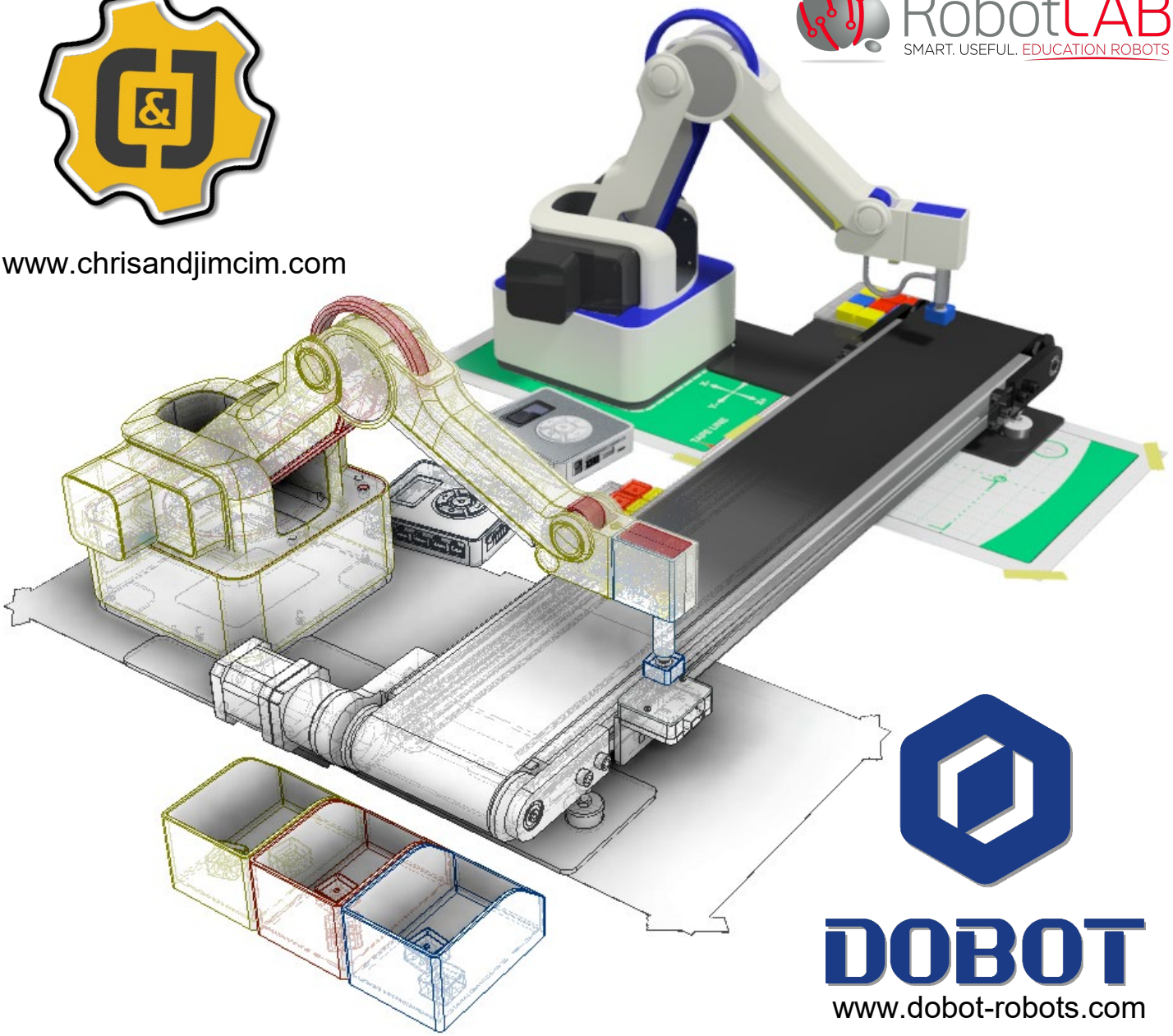
Intro to Robotics

DOBOT MAGICIAN LITE

www.chrisandjimcim.com

RobotLAB
SMART. USEFUL. EDUCATION ROBOTS



DOBOT
www.dobot-robots.com

Jim Hanson & Chris Hurd V1 - 24.1

How to use this curriculum

This curriculum was designed to teach middle school, high school, and college level students the basics of robotics, as used in industry. All concepts taught within can be done most easily with the Dobot Magician, Magician Light, Sensor Kits, & Peripherals.

Chapters with this icon denote that they are designed to be used with **DobotBlock** programing in the **DobotLab** software.

Chapters with this icon denote that they are designed to be used with **Teaching & Playback** programming in the **DobotLab** software.

Chapters with this icon denote that they are designed to be used with **DobotBlock** programing with the in the **DobotLab** software with the **Magician Sensor Kit**.

Table of Contents

1		*Robot Movement*	*6*
		*1.1 T&P - Robot Axis and Movement*	*7*
		*1.2 T&P - Pick & Place Routines*	*21*
		*1.3 Block - Pick & Place Routines*	*31*
		*1.4 T&P - Using Jumps & Loops*	*45*
		*1.5 Block - Pick & Place with Jumps & Loops*	*54*
		*1.6 Block – Loading and Unloading Pallets*	*62*

2	**Robot Communication**		**82**
		2.1 Block - Dobot to Dobot Internal Handshaking	83
		2.2 Block - Magic Box to Magic Box Bluetooth Handshaking	95
		2.3 Block – Pick and Place with the Magician and Magic Box	102

3	**Robotics & The Use of Sensors**		**111**
		3.1 Block - Introduction to Inputs and Outputs (Sensor Kit)	112
		3.2 Block - Sensor Exploration (Sensor Kit)	124
		3.3 Block – Color Sensor with Magic Box (Sensor Kit)	147
		3.4 Block - Sorting with the Color Sensor (Mini Conveyor)	155
		3.5 Block – Servo Feeder (Sensor Kit)	167

4		Robotic Scenarios & the Use of Peripherals	173
		4.1 Block – Start and Stop Conveyor (Mini Conveyor Kit)	174
		4.2 Block - Pick and Place with the Linear Slide Rail (Sliding Rail Kit)	183
		4.3 Block – AI Camera Vision System (Sensor Kit)	194
		4.4 Block - Stack Light with the Magic Box (Sensor Kit)	205
		4.5 Block – Work Cell Design Guided	210
		4.6 Block – Work Cell Design Open Ended	214

5		Robotics Resources and Printable Help Files	218
	ALL	5.1 Key Concepts	219
	ALL	5.2 Peripheral Guide	223
	ALL	5.3 AI Sensor Guide	224
	ALL	5.4 Work Cell Example Guide	225
		5.5 Color Test Template	226
	ALL	5.6 Base Field Diagram	227
	ALL	5.7 Robot Axis and Movement Field Diagram	228
	ALL	5.8 Blank Field Diagram	229
	ALL	5.9 Two Pallet Field Diagram	230

	ALL	*5.10 Dip Tank Field Diagram……*	*231*
	ALL	*5.11 Handshake Field Diagram……*	*232*
		*5.12 Camera Vision Field Diagram……*	*233*
	ALL	*5.13 Part Feeder / Pallet Field Diagram ……*	*234*
	ALL	*5.14 3D Models ……*	*235*
	ALL	*5.15 Robotics Glossary ……*	*238*
		*5.16 Blockly Glossary ……*	*243*
	ALL	*5.17 Educational Standards Addressed in This Curriculum ……*	*250*

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING
MAGICIAN LITE

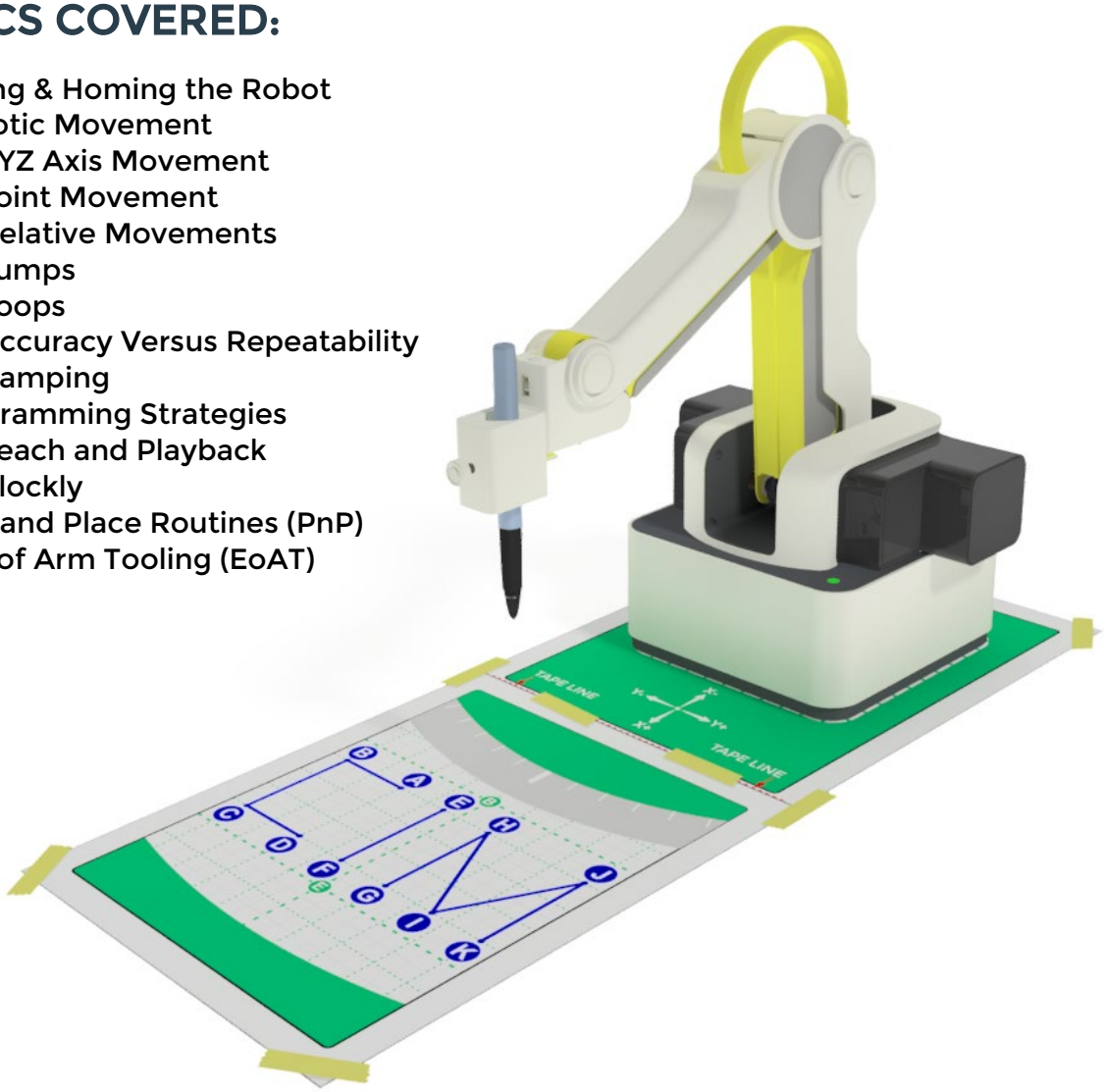
- CHAPTER 1 -
ROBOT MOVEMENT

CHAPTER INTRODUCTION:

Jointed arm robots are useful for many different tasks due to their range of motion and degrees of freedom. This chapter will illustrate how to both move and program a robotic arm in multiple ways.

TOPICS COVERED:

- Wiring & Homing the Robot
- Robotic Movement
  - XYZ Axis Movement
  - Joint Movement
  - Relative Movements
  - Jumps
  - Loops
  - Accuracy Versus Repeatability
  - Ramping
- Programming Strategies
  - Teach and Playback
  - Blockly
- Pick and Place Routines (PnP)
- End of Arm Tooling (EoAT)

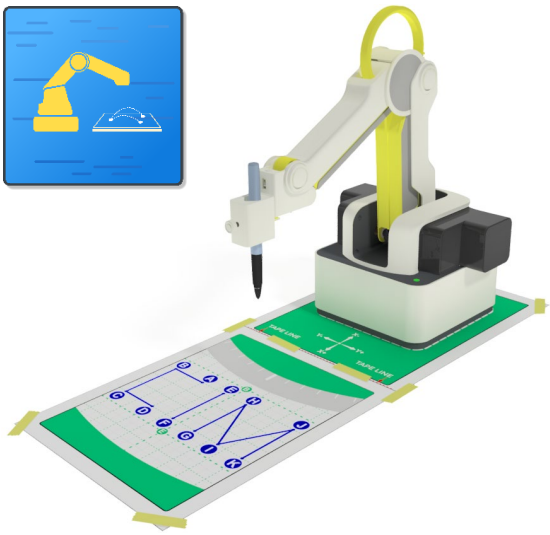


1.1 TP – Robot Axis and Movement

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

INTRODUCTION

Jointed arm robots are useful for many different tasks because of its range of motion and degrees of freedom. In this activity, you will learn how to move a robotic arm in many different ways and write a program to make the robot write the word "CIM" with **ACCURACY** and **REPEATABILITY**. CIM stands for Computer Integrated Manufacturing.



The method of measurement and positioning we will use to do this is called **RELATIVE COORDINATES.** We will have the robot move the pen "relative" to where it was last. We will also use a method of saving points called **TEACHING**. This is where we type in coordinates without having to move the robot arm.

KEY VOCABULARY

- Relative Coordinates
- Joint movement
- Axis movement
- Work envelope
- Ramping
- Accuracy
- Teach
- Linear move
- Home
- Loop
- End Effector
- End of Arm Tooling (EoAT)

EQUIPMENT & SUPPLIES

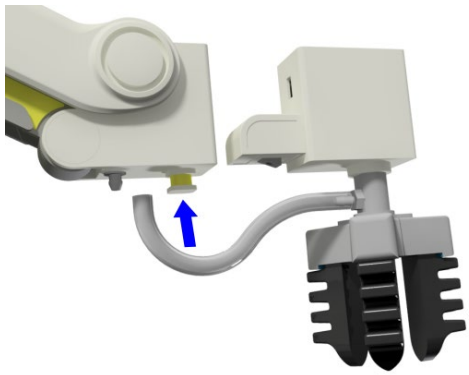
- Dobot Magician Lite
- Printed 2 Part Dobot-Lite Field Diagram (see Resources)
- Pen Holder (end effector bracket)
- Dobot Lab software
- Pen/Marker
- Masking tape

PROCEDURE

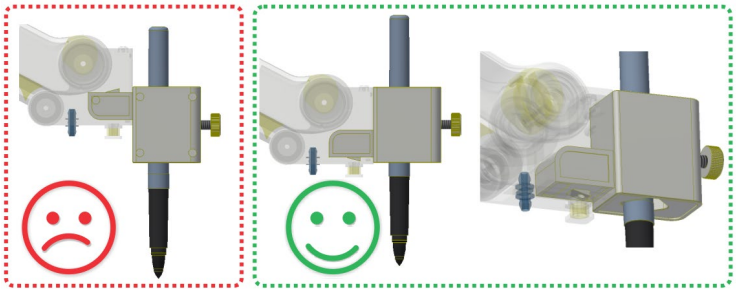
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur.

1. Typical Start Up Procedure

- Disconnect any existing **END of ARM TOOLING (EoAT).**
- Carefully disconnect any existing vacuum tubes.
- Press and hold the release button on the bottom of the arm and pull off the EoAT.

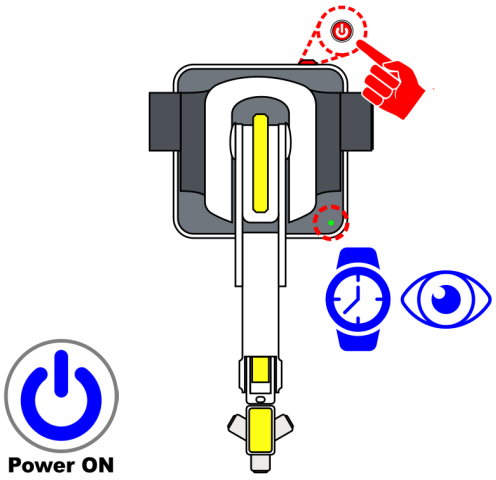


- Attach the Pen holder as the **END EFFECTOR** or **END of ARM TOOLING (EoAT)** on the Dobot.
- Push the pen holder in until it snaps in



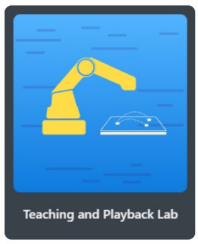
Do not try to attach the pen holder upside down. The locking tab should be on the bottom

- Plug the 120-240VAC power into an outlet.
- Attach the 12VDC 5A barrel plug of the power supply and USB to the Dobot. Plug the USB into the computer and wait for a connection. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)
- Turn on the power to the Dobot (Robot will beep when ready).
- Open DobotLab software.

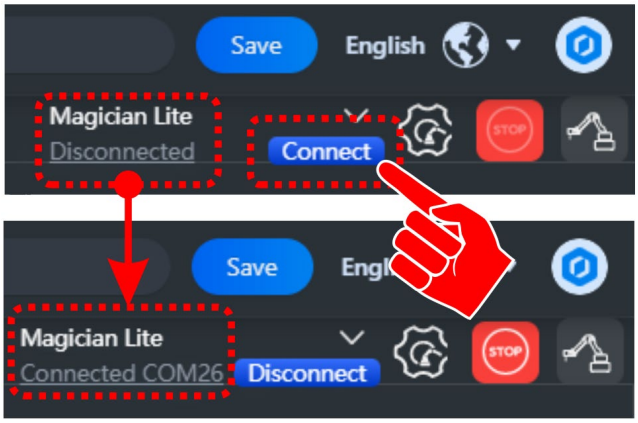


2. Once the Robot's USB is connected to the computer and the robot is powered on, open the DobotLab software

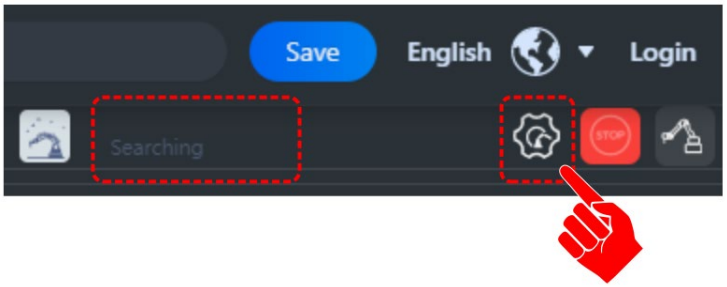
3. Once the software is opened select the Teaching and Playback Lab.



4. Select connect to establish control from the software to the robot. Disconnected should change to Connected and show the COM being used (COM is the USB port being used by the robot)

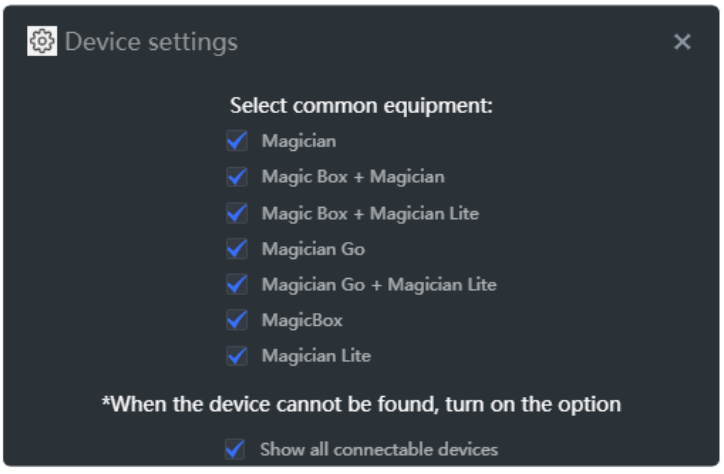


5. NOTE: If no robot is found, the prompt will show as Searching….



SEARCHING… Complete the steps below if no robot is found.

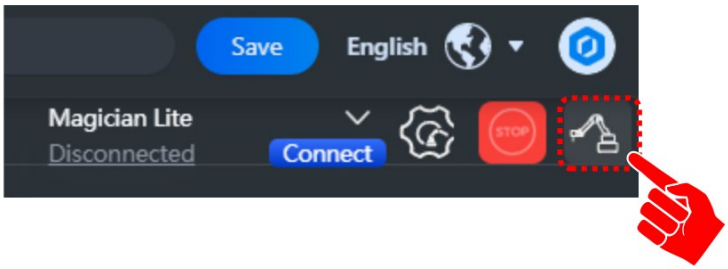
Select the Gear Icon next to the Force Stop Icon.



Make sure the hardware you are connected is checked.

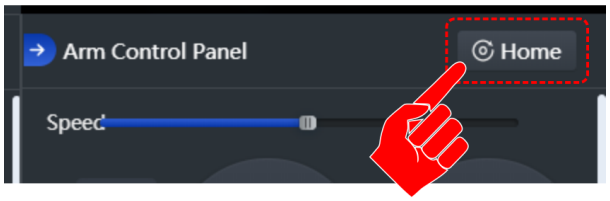
ARM CONTROL PANEL

6. Select the ***ARM CONTROL*** Icon to open the arm control panel.



Before **HOMING** the robot, make sure the robot's **WORK ENVELOPE**, the area in which the robot can reach is clear.

7. Select the **HOME** icon to start the robots homing process. **HOMING** the robot will return the robot to its initial **HOME** position.

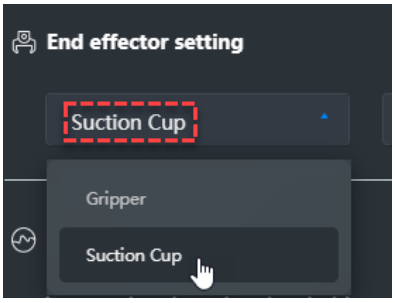


8. Be sure the ***Pen*** is chosen as the Dobot's **END EFFECTOR** or **EoAT** (End of Arm Tooling).

9. Insert a ball point pen or marker into the holder replacing the one that is there and remember to remove the cap.

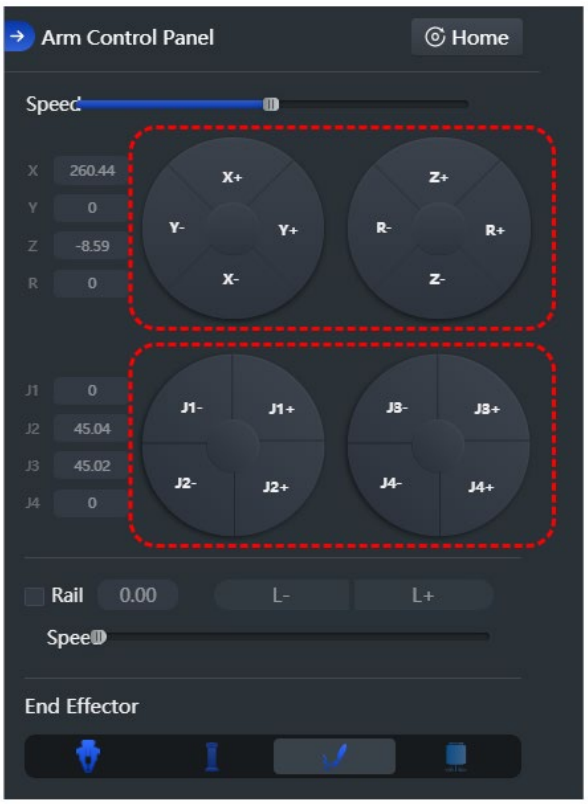
ARM CONTROL PANEL

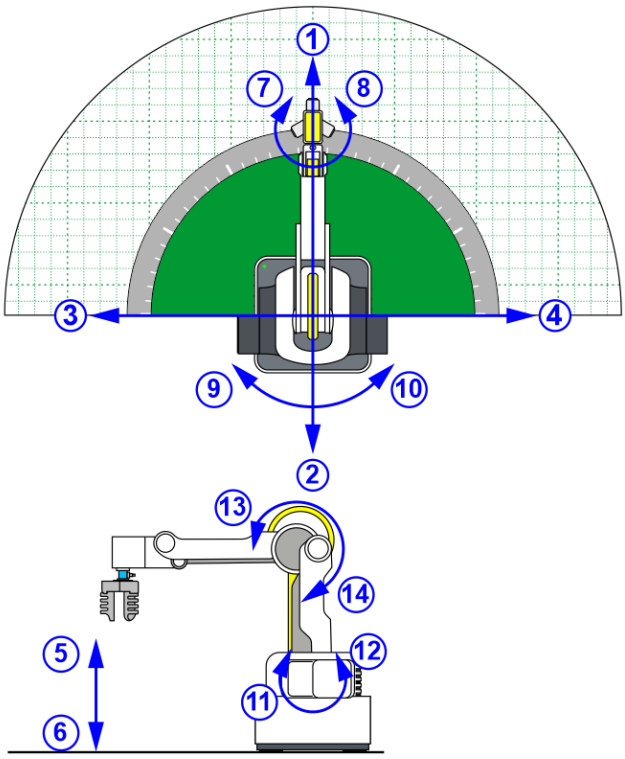
10. In the ***CONTROL PANEL***, set the end effector as the suction cup (the pen is not an option to select).



ARM CONTROL PANEL

11. Use the Axis and Joint buttons to move the robot around. Using the diagram below, identify the **AXIS MOVEMENT** and **JOINT MOVEMENTS** for the robot (1-14). Be sure to label them as +/-. (Press and hold ALL 16 options, what do they do, left or right, up or down, in our out….)





12. Using the information you documented on the previous diagram, write down the correct axis/joint button in the first empty column, and a description of what it does in the second.

		AXIS / JOINT + / -	DESCRIPTION IN / OUT / UP / DN / LEFT / RIGHT
1	Axis		
2	Axis		
3	Axis		
4	Axis		
5	Axis		
6	Axis		
7	Wrist		
8	Wrist		
9	Waist		
10	Waist		
11	Shoulder		
12	Shoulder		
13	Elbow		
14	Elbow		

13. Did the axis button work as you expected… did it move the way you thought it would? Explain:

14. How do the XYZ movements differ from the J movements?

15. **MAIN OBJECTIVE:**
    A. Tape together the two parts of the Template along the **RED TAPE LINE**.

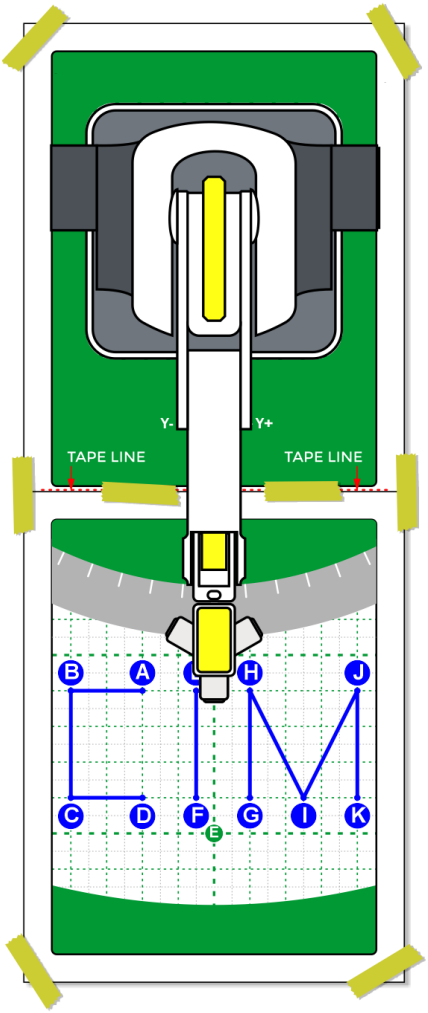
    B. Tape the Template to the table.

    C. Place the Robot on the Template

    D. Handwrite the word “CIM” in pencil on the Dobot Template as shown.

    E. Using the Dobot Lite with the pen end effector, start at point A on your diagram and move the robot from point to point to re-write the word using straight lines. Be sure to pick up the pen between letters.

    F. Send the robot to a home position away from the paper when finished.

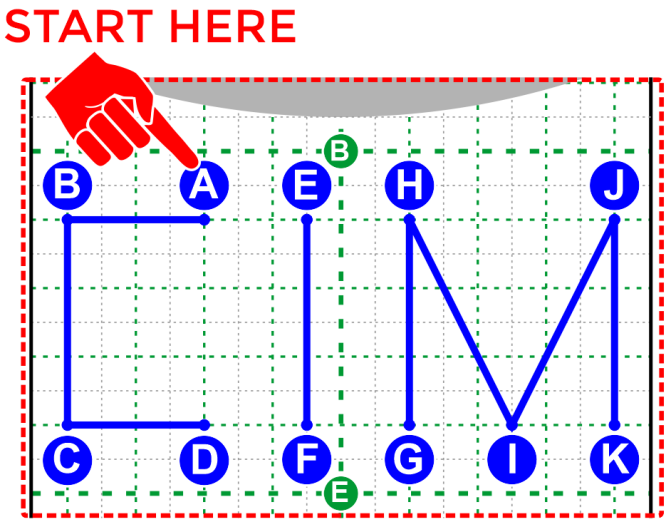


16. What do you think is this the best way to write the letters CIM (order of movement)? Did you take time into consideration? How?

HELPFUL TIPS

When moving in the z-axis, try to put a light pressure on it when you write, or it may tear the paper or break the tip. The pen is NOT spring loaded; You have to have a soft touch or the paper my tear or the pen may be caught on the paper and slide the robot’s base. It may be a good idea to put a piece of cardboard under the paper to give it a softer surface to write on.

17. For this activity, we will use a combination of recorded and taught positions. We will teach the robot the points shown using actual coordinates. The big **GREEN** squares on the paper are equal to about 20mm, this will help us plan our letters.



18. Line Segment lengths are as follows:

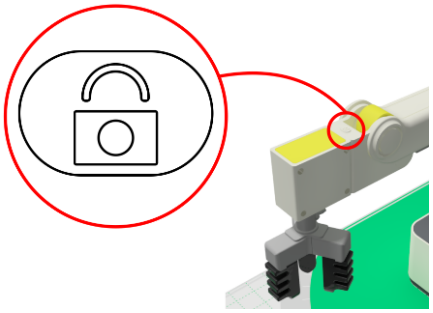
AB = 40mm	BC = 60mm	CD = 40mm
EF = 60mm	GH = 40mm	*HI = 68mm
*IJ = 68mm	JK = 60mm	

19. Make sure you are in the **Teach and Playback** lab.

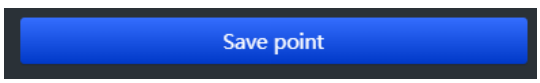
We will use this lab to **RECORD** and **TEACH** the robot the points we want it to go to write the word CIM.

There are two methods to RECORDING a point.

Method 1: **RECORDED** points can be found using the unlock button on the robot's arm. When the unlock button is pressed, the arm becomes free to manually move to a new location. The location is automatically **RECORDED** and added as a step in the program when the unlock is released. Movements in between the steps (while the button is held) will not be recorded.



Method 2: **RECORDED** points can also be found by manually driving the robot to a position from the *Arm Control Panel* and selecting Save Point when the **EoAT** is at the desired location

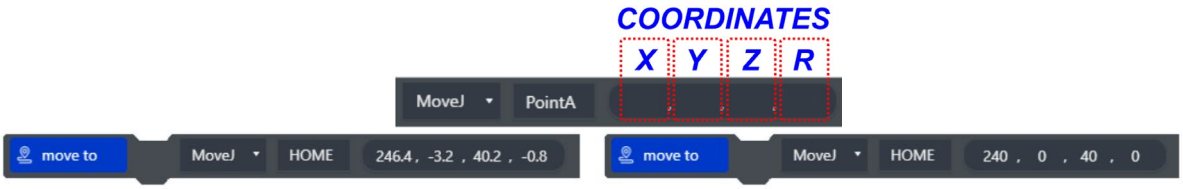


TEACHING points is the process of manually typing in the XYZ coordinates if they are known. We often use this method to **TOUCH UP** points (round them to whole numbers, change them to a known number, or duplicate positions from previous steps that should not have changed).

Always start a program with a safe/clear HOME position. Home is a position that is up and away or clear of any obstacles.

20. Using the XYZ buttons on the *Arm Control Panel*, move the robot to a safe **HOME** position. We can refer to this as "HOME".

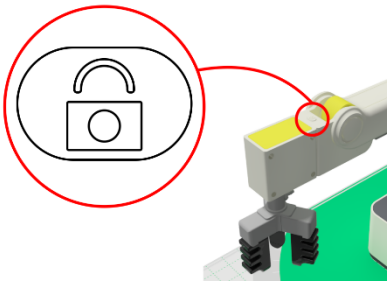
    -Touch up (round) the coordinates as needed. Ignore the last number in the coordinate's window "R", as this stands for "rotation or roll". It is not used with the pen..



    -Record these values on the chart below.

    Using the XYZ buttons on the *Arm Control Panel*, move the robot to a point above **POINT A**. We can refer to this as AB-A (Above Point A). Do not touch the paper yet with the pen! - Record this value on the chart below.

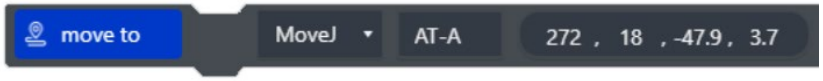
21. You can also press the "Lock" button on the arm and it will **RECORD** the point where the pen is when the button is released. If the point has moved a bit, you can touch up the points by **TEACHING** them in the XYZ coordinates window.



Record the values to create the letter CIM on the chart below.

STEP	POSITIONS	X =	Y =	Z =
1	HOME			
2	AB-A			
3	AT-A			
4	AT-B			
5	AT-C			
6	AT-D			
7	AB-D			
8	AB-E			
9	AT-E			
10	AT-F			
11	AB-F			
12	AB-G			
13	AT-G			
14	AT-H			
15	AT-I			
16	AT-J			
17	AT-K			
18	AB-K			
19	HOME			

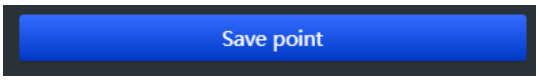
Click in the cell with the "P" for each point to create a descriptive name for each position. See Example Below "AT-A" (EoAT is At Point A).



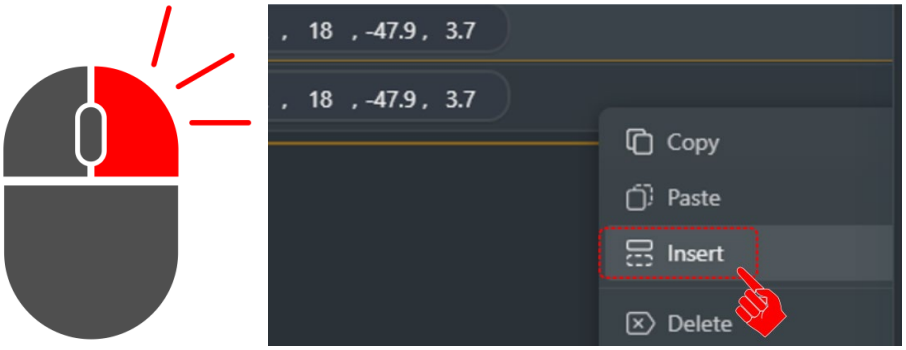
Movements can either be *MoveJ*, *MoveL*, *or Jump*. We will get to *Jump* in later activities.
MoveJ movements always have an arc to them. We use them for movements that can have less accuracy. Example: when traveling from home to an above position or when traveling from an above to another above position.
MoveL movements are linear. We use them when moving down to a point or object, or when moving away from a point or object to ensure movement accuracy. **All of the lines to create the letters for CIM should use *MoveL* movements**

If TEACHING the positions, select the Save Point tool to create 19 empty steps. Using the chart you created, fill in the names of the steps and the XYZ values.

Note: The coordinates automatically used for the Save Point is the current location of the **EoAT**



You may also right click on an existing step and select insert to create an additional step



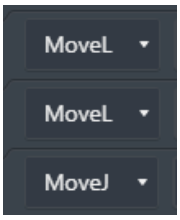
If RECORDING the positions, the positions are automatically created as well as filled in. These positions should be touched up.

Touch Up Example 1: AB (Above) positions should only have a different Z value. The X and Y should be the same.

Touch Up Example 2: All AB (Above) positions should have the same Z value. All AT (At) positions should have the same Z value.

Touch Up Example 3: The positions for AT-A and AT-B should only change on one axis.

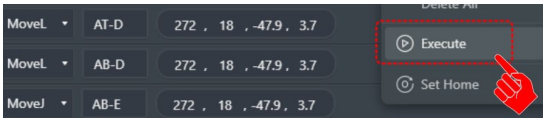
22. Change the Motion Styles to *MOVEJ* and *MOVL* where appropriate.



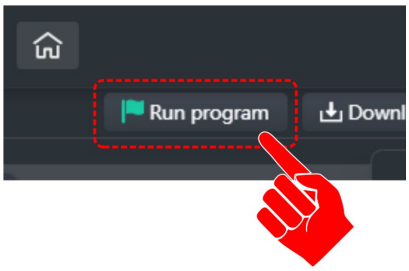
23. It is beneficial to step through (run one line of code at a time) a program to find any errors. Right click on each step one at a time (from the top down) and select EXECUTE. This will run just that step.

    If there are errors in a step, update the XYZ coordinate values until they are correct.

    Repeat this process for each step.



24. Once each individual step is verified, we can try to run the entire program. Select the first line of code, ensure there are no obstructions in the robots path, and select the RUN PROGRAM option at the top of the screen.

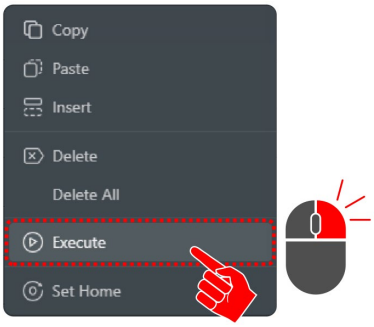


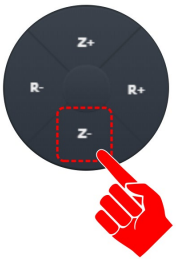
25. Now that we know all the points are correct, we must change the Z value so that it writes on the paper. To do this, follow these steps:

A. Select the first AT line, right click & use ***EXECUTE*** to move the robot to point A.

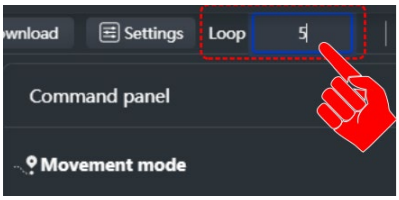
B. Change the Z by increasing its value -2mm at a time and hitting *RunSelected* until the pen just touches the paper. You may also use the Z- from the Arm Control Panel

C. Now change all the Z values where the pen must actually write a line. Leave the AB (Above) value alone when it moves from letter to letter.



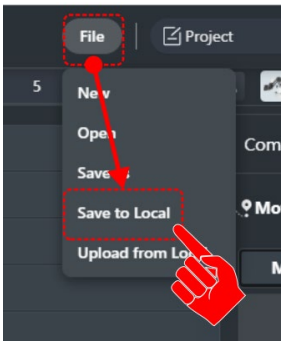


26. Using the *Loop* button, change it to 5 and watch your program **LOOP** five times when you hit the start button.



Save your work.

Select **File** from the top tool bar.
Select **Save to Local** to save your project on to the computer.



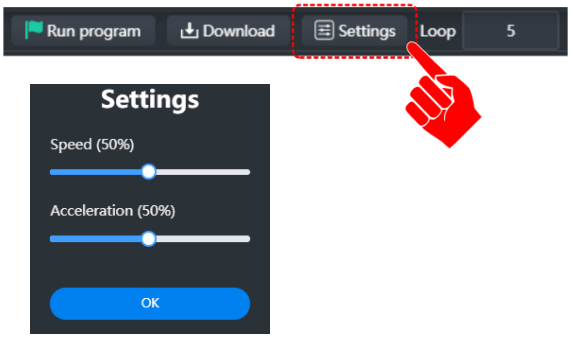
Let's check the **ACCURACY & REPEATABILITY** of our robot. Get another grid portion of the template either replace old one or add it on top of the old one. Be sure to tape it to your work surface. Now run your program again. Check all the lines closely. *How accurate was your robot at reproducing the word CIM five times? Describe it below.*

How accurate and repeatable was your robot and program? Be sure to describe both terms.
Response:

What happens to the **ACCURACY & REPEATABILITY** if you increase speed to 75? Try it and see.

Please do not increase speed above what is specified by your instructor.

To change the speed of your program, select the settings tab at the top of the screen.



Speed Adjustment: What happened when you changed the speed? Be specific.
Response:

CONCLUSION

1. What does it mean to touch up a program?

2. Explain completely the difference between MOVJ and MOVL Motion Styles.

*3. Does speed influence **ACCURACY** or **REPEATIBILITY** of your robot? Describe how.*

4. What would be the effect on the robot's accuracy at higher speeds if you greatly increased the mass of the pen?

5. After completing this activity, how would you define the difference between a robot's accuracy versus its repeatability?

6. Why do we use AB (Above) positions before going to AT (At) Positions? Explain fully.

7. What is the purpose of a HOME position?

GOING BEYOND

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

Teacher
Signature

_____ 1. Teach the robot to make a barcode with the pen.

_____ 2. Teach the robot to write your name.

_____ 3. Teach the robot to write arcs (can you figure Arc movements out?).

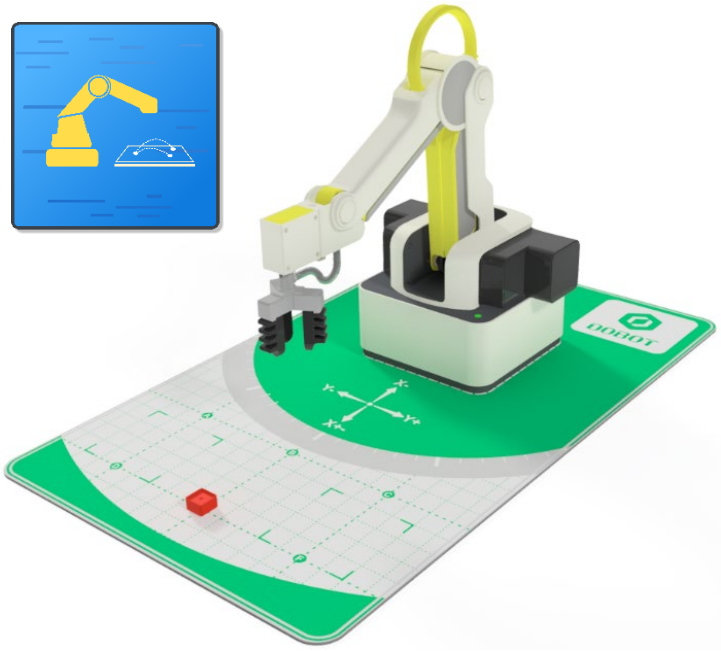
1.2 TP – Pick & Place Routines

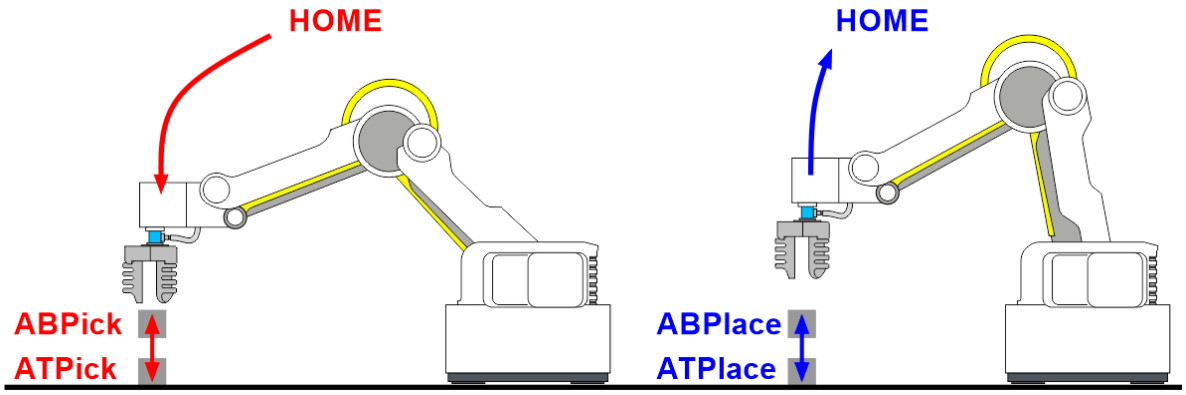
NAME: _____ *Date:* _____ *Section:* _____

INTRODUCTION

Robotic arms are excellent for performing **PICK AND PLACE** operations such as placing small electronic components on circuit boards, as well as large boxes on pallets. A pick and place operation will require at least 5 points:

- Home
- Above the pick point
- At the pick point
- Above the pick point
- Above the place point
- At the drop point
- Above the place point
- Home





As a rule, always go to a position above the pick or place point first so that the robot can accurately and repeatedly place the object straight down in a linear motion, with no friction or interference with the work surface or other objects in the work envelope.

KEY VOCABULARY

- Relative positions
- Palletizing
- Pick and Place
- Home
- Linear movements
- Joint movements
- End effector
- End of Arm Tooling (EoAT)

EQUIPMENT & SUPPLIES

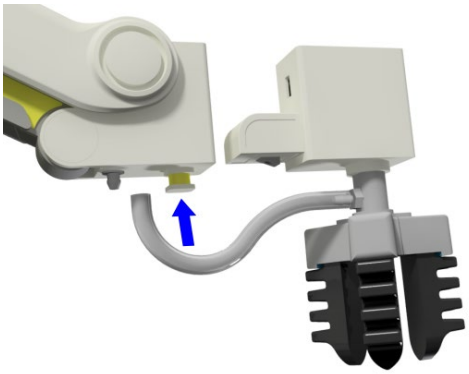
- Dobot Magician Lite
- Dobot Lite Work Mat or Field Diagram
- Small cubes
- Soft Gripper
- DobotLab software

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. **Typical Start Up Procedure**

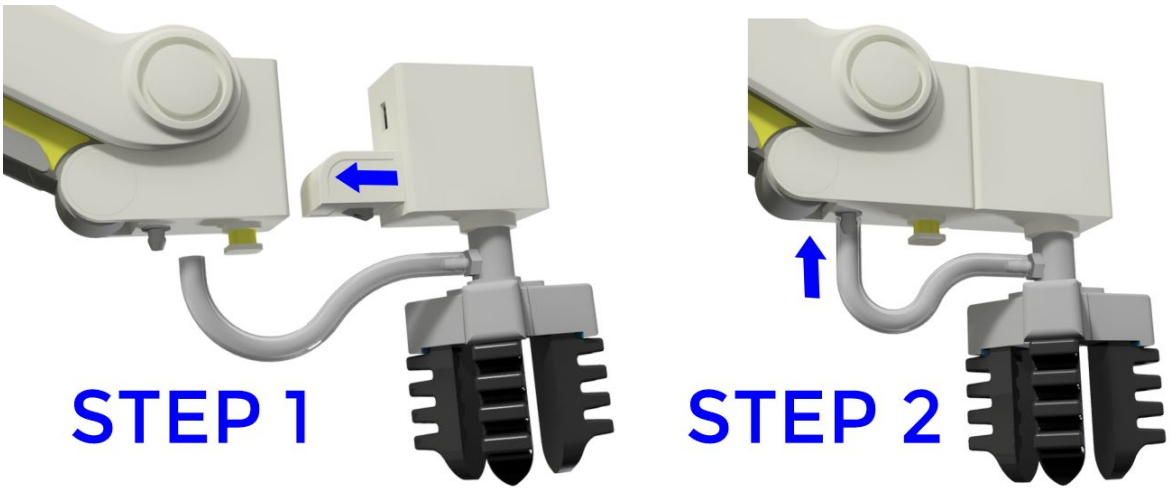
- Disconnect any existing **END of ARM TOOLING (EoAT).**
- Carefully disconnect any existing vacuum tubes.
- Press and hold the release button on the bottom of the arm and pull off the EoAT.



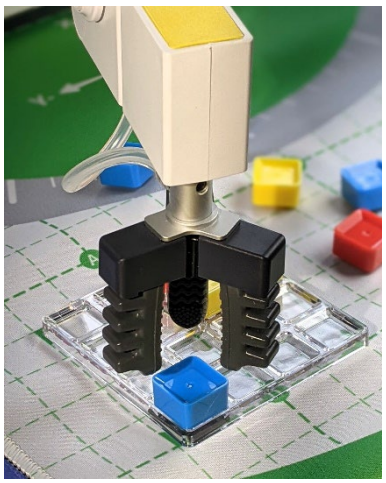
2. Attach the *Soft Gripper* as the **END EFFECTOR** or **END of ARM TOOLING (EoAT)** on the Dobot.
   - STEP 1 - Push the *Soft Gripper* in until it snaps in
   - STEP 2 - Attach the hose to the air nozzle

HELPFUL TIPS

The soft gripper works by inflating or deflating the device with an internal pump in the robot. Blow air into it and it closes; Suck air out of it and it opens! There is also a Disabled mode when the air pump is turned off.



Soft Gripper Modes of Operation

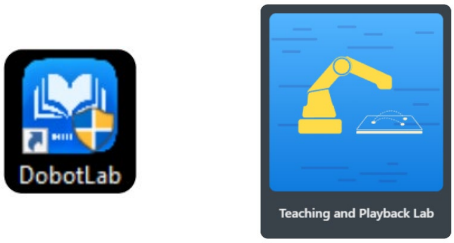


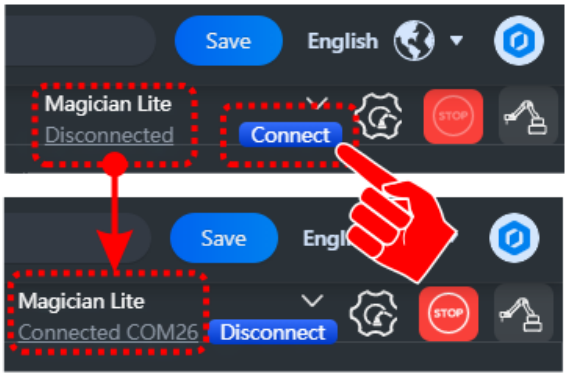
In this mode the vacuum pump inside the robot is off, and the gripper is at rest in a NUETRAL position.

In this mode the vacuum pump inside the robot is on, and the gripper is expanded in the OPEN position.

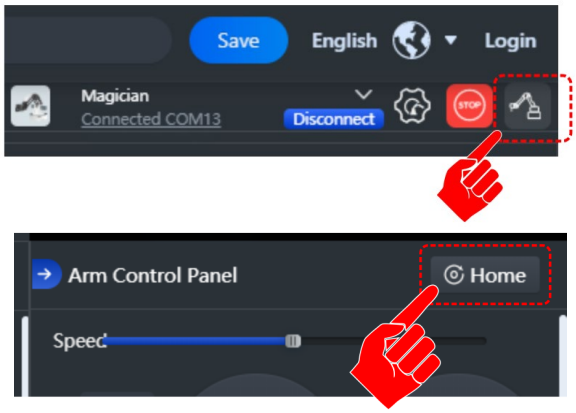
In this mode the vacuum pump inside the robot is on, and the gripper is contracted in the CLOSED position

1. Open DobotLab software, enter the Teach and Playback Lab, and connect the robot.

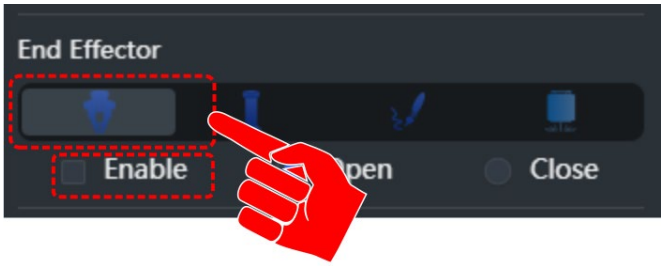




2. Open the *Arm Control* window and *Home* the robot making sure the robot's work envelope is clear.



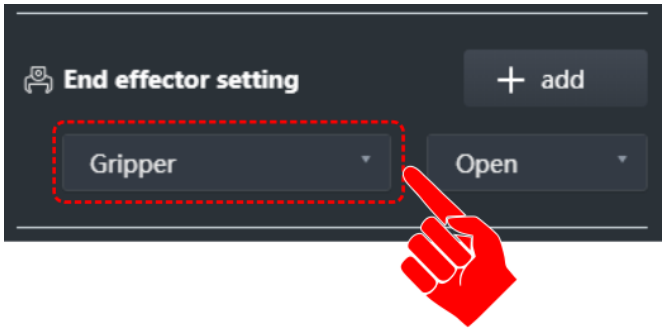
3. Be sure the *Gripper* is chosen as the **END EFFECTOR or EoAT**



HELPFUL TIPS

The **END EFFECTOR** only needs to be ENABLED when the gripper needs to be active. We will activate it later in the activity.

4. The Gripper ALSO needs to be selected as the **END EFFECTOR** setting in the Control Panel in order to record when it is open and closed.

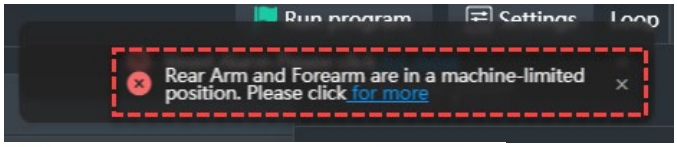


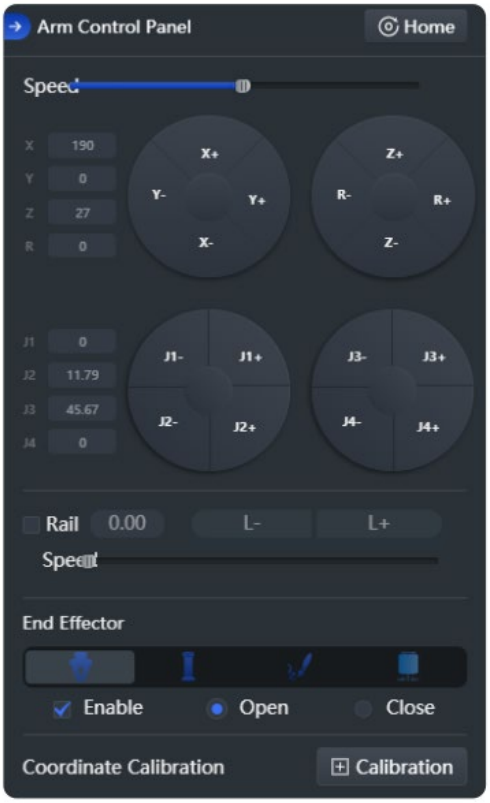
5. Open the *Arm Control Panel* and move the robot around. Using the diagram below, identify the X and Y axis on the robot. Be sure to identify which direction is positive and which is negative (+/-).

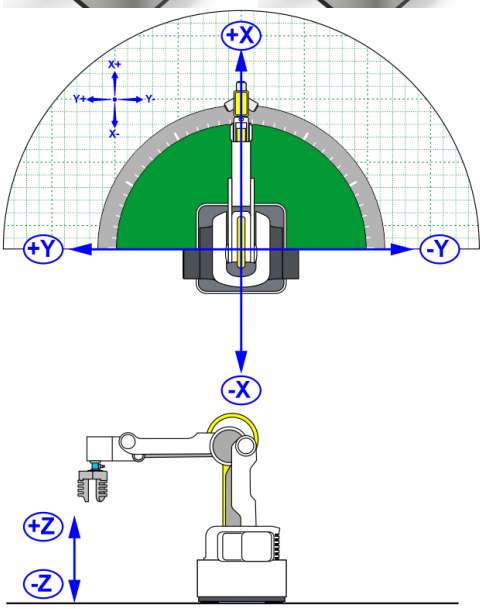
If you move the arm too far in any direction you will get an error (see below), the light on the robot will turn red, and the arm will stop moving! Just hit the opposite axis button until the light turns green again, and the arm starts moving.

LIGHT INDICATORS

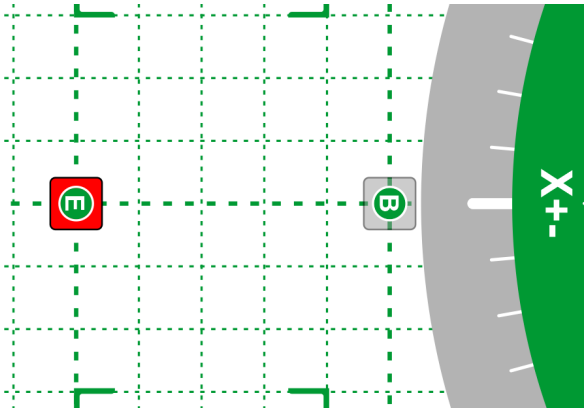
- GREEN = READY
- RED = LIMIT REACHED or IMPACT



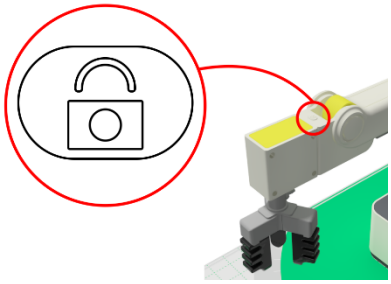


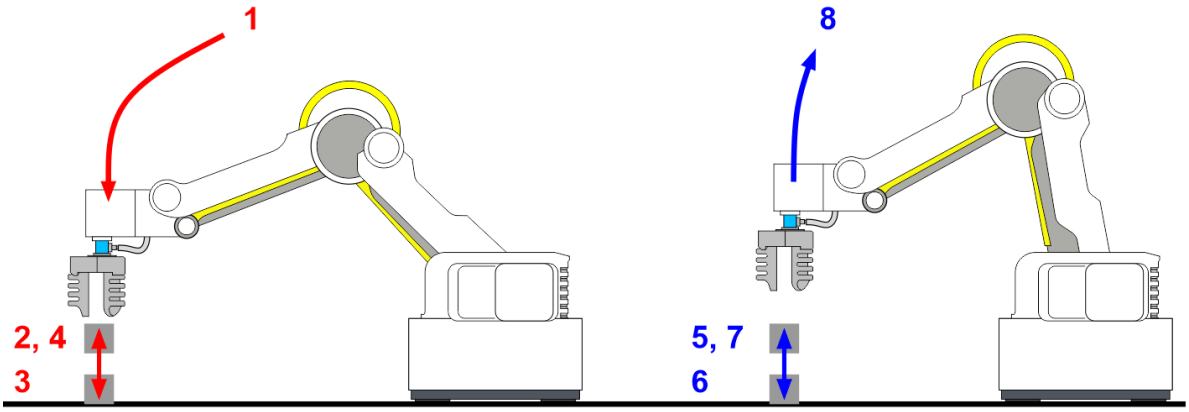


6. Place a cube on letter “E” on the field diagram. The cube will be picked up at letter “E” and placed at letter “B”.



7. Use the *“Lock” button* on the arm to record all the positions necessary to do a pick and place operation in this order:
   - 1 - Home
   - 2 - Above Letter E – (ABPICK)
   - 3 - At Letter E – (ATPICK)
   - 4 - Above Letter E – (ABPICK)
   - 5 - Above Letter B – (ABPLACE)
   - 6 - At Letter B – (ATPLACE)
   - 7 - Above Letter B – (ABPLACE)
   - 8 - Home





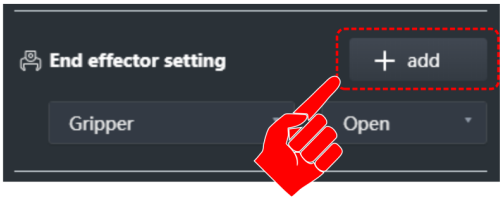
LETTER E – PICK **LETTER B – PLACE**

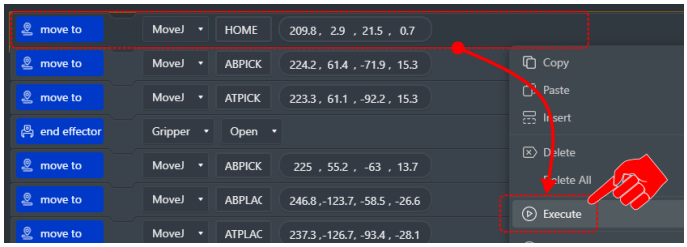
8. Name the positions in the *name column* of the program.

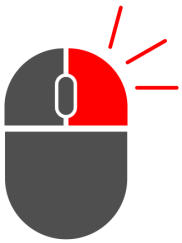
HELPFUL TIPS

Be sure to name the positions something relevant so that others will be able to tell what the positions are. Example: A point named ABPICK means the point above the place where the object is picked up.

9. Next, add the commands to open and close the gripper. Select the add button in the **END EFFECTOR** Settings window. This will add the step to the end of your program. Click and drag the new step into the correct position of your program sequence. Repeat this step to open the gripper when the object needs to be released.

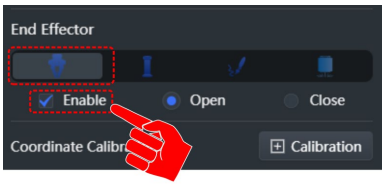




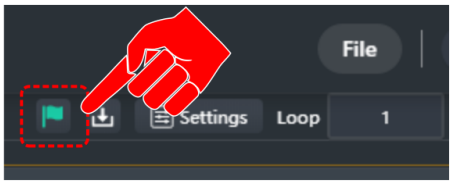


Right click on each step (one at a time), and select Execute. This will allow you to step through the program to find any errors before trying to run the entire program. Make any edits to the sequence as necessary as you go.

NOTE: The gripper will **AUTOMATICALLY** become active (pump will turn on) when an open/close step is excuted. It can also be manually activated and deactivated by selecting or unselecting the Enable options for the **END EFFECTOR** in the Arm Control Window. The pump will switch modes while opening and closing the gripper *(the sound coming from the pump will alternate when the mode is switched).* One mode produces pressure to open the gripper while the other mode produces a vacuum to pull the gripper closed.

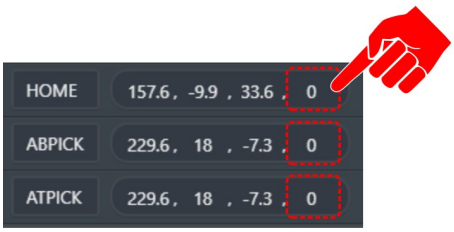


10. Select the "*Run*" button to run your entire sequence and see what happens.



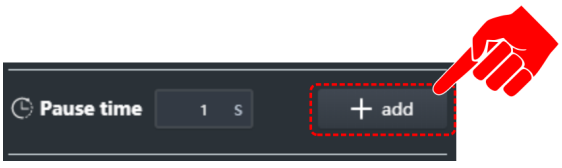
Did it work the first time? If not, what did you have to change to make it work?

11. If you are using cubes, we will want the gripper to remain parallel to the robot. To do this, change all of the Roll Angles to ZERO. This will keep the gripper in the same orientation throughout the program. The gripper's roll angle DOES NOT matter if the objects are cylinders.

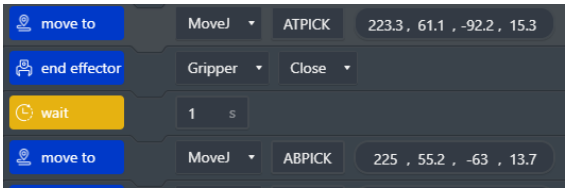


12. Change the position type of step #8 (At Pick) to *Move Linear* (*MOVL*). *Run the program. What changed? What positions should be Linear Movements, and which should/could be Joint Movements? Edit the program to have appropriate movement types for each position in the sequence.*

13. Next we will add Pauses for time in-between step. This will allow the robots EoAT to completely close, or open, before it moves onto the next step in the sequence.



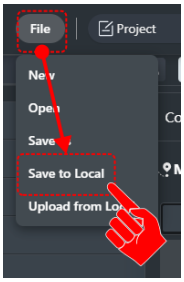
When the Add button is pressed, the pause step will be added to the end of the sequence. Grab the newly created step and drag it below the close gripper step. For this activity, 1 second should be enough.



Add additional Pauses to the program sequence where appropriate. Pauses should be used when trying to ensure the gripper or suction cup has built up full pressure or suction before picking up or letting go of an object.

14. Finally, we will add an additional step at the end of the program to stop or turn off the pump.

15. Save your work.



CONCLUSION

1. *Did this gripper work well to pick up the small cubes? Was it easy? What are some issues you might have with this type of gripper?*

2. *Was your robot very accurate with this gripper? Example: could this gripper be used to stack the cubes? Explain why or why not.*

3. *How can you get the suction to turn on in time to pick up the part, or get it to shut off in time to drop it off correctly? Explain below after you have tried it in the program.*

4. *What happens if you try to pick up something different, like a marble or small sphere? Ask your instructor for some different objects to try. What kind of objects does this gripper work well with?*

GOING BEYOND

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

_____ 1. Have the robot reverse the process and put the cube back in its original place.

_____ 2. Make the process happen multiple times.

_____ 3. Adjust the speeds and acceleration to increase efficiency but not lose accuracy.

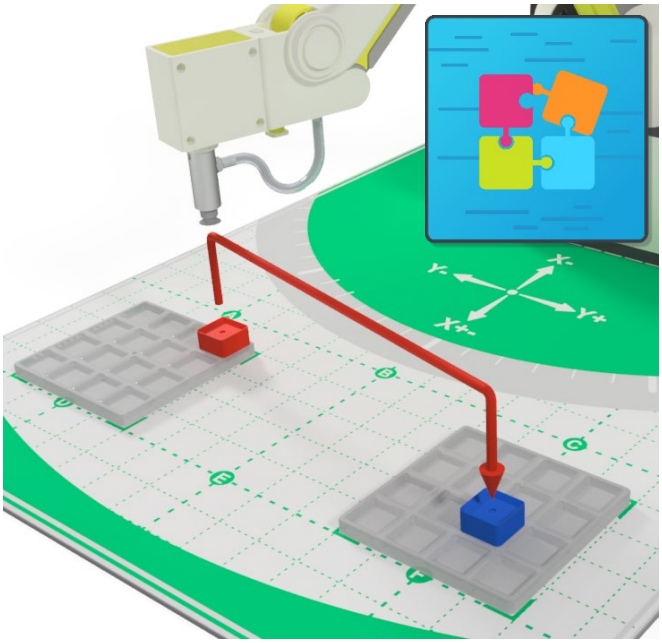
1.3 Block - Pick & Place Routine

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

INTRODUCTION

Robotic arms are excellent for performing pick and place operations such as placing small electronic components on circuit boards, as well as large boxes on pallets. A pick and place operation will require at least 5 points:

1. A home or safe location
2. A position above the object
3. A position at the object
4. A position above the drop off
5. A position at the drop off



In this activity you will learn how to make a basic ***Pick and Place*** operation in blockly. Through this activity you will learn how to program the robot to move and turn on its suction cup 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. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

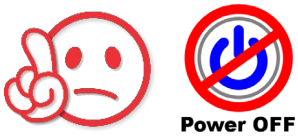
- MoveTo
- Placeholder
- Delay time
- Suction Cup
- Blockly Programming
- Pick and Place

All blockly commands and robotics terms have been put into a separate *glossary* that can be founded in the ***Resources*** chapter of this curriculum.

EQUIPMENT & SUPPLIES

- Robot Magician Lite
- Magician Light Work Mat
- Small cubes & pallets
- DobotLab Software or
- Suction Cup Gripper
- Optional: Magician Lite Field Digram

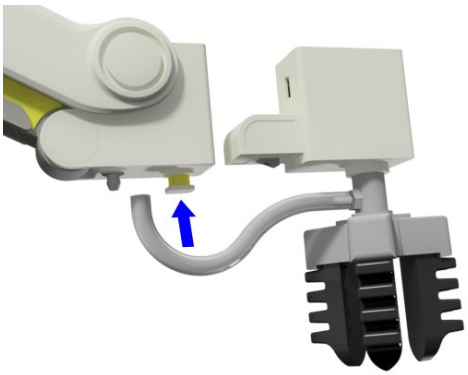
PROCEDURE

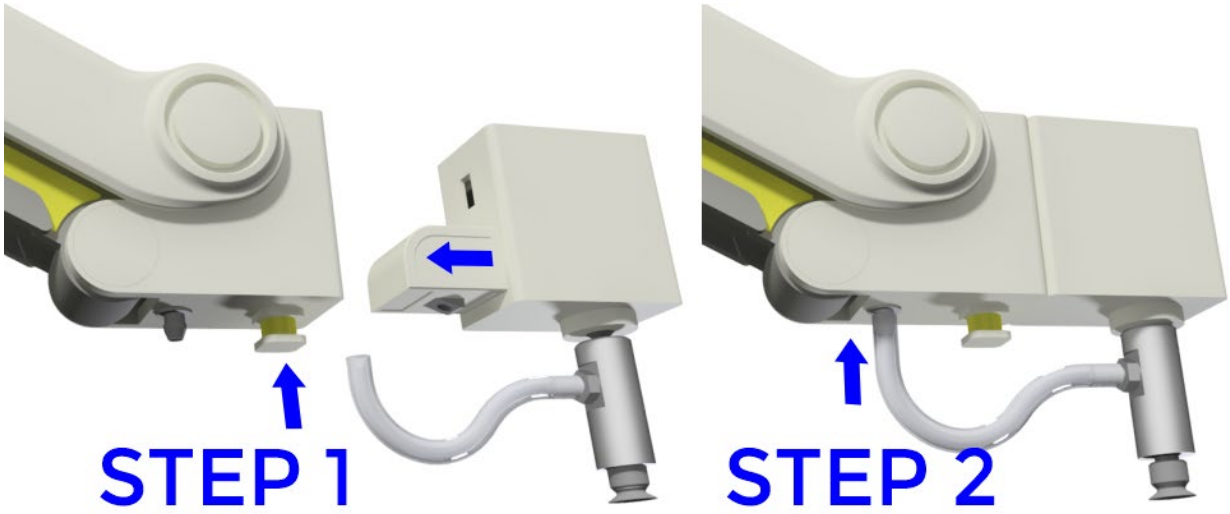


Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

Typical Start Up Procedure

- Disconnect any existing **END of ARM TOOLING (EoAT).**
- Carefully disconnect any existing vacuum tubes.
- Press and hold the release button on the bottom of the arm and pull off the EoAT.





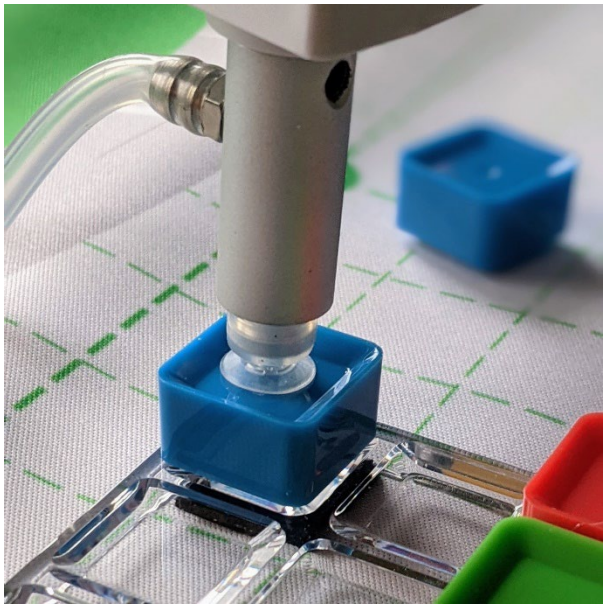
1. Attach the *Suction Cup* as the **END EFFECTOR** or **END of ARM TOOLING (EoAT)** on the Dobot.

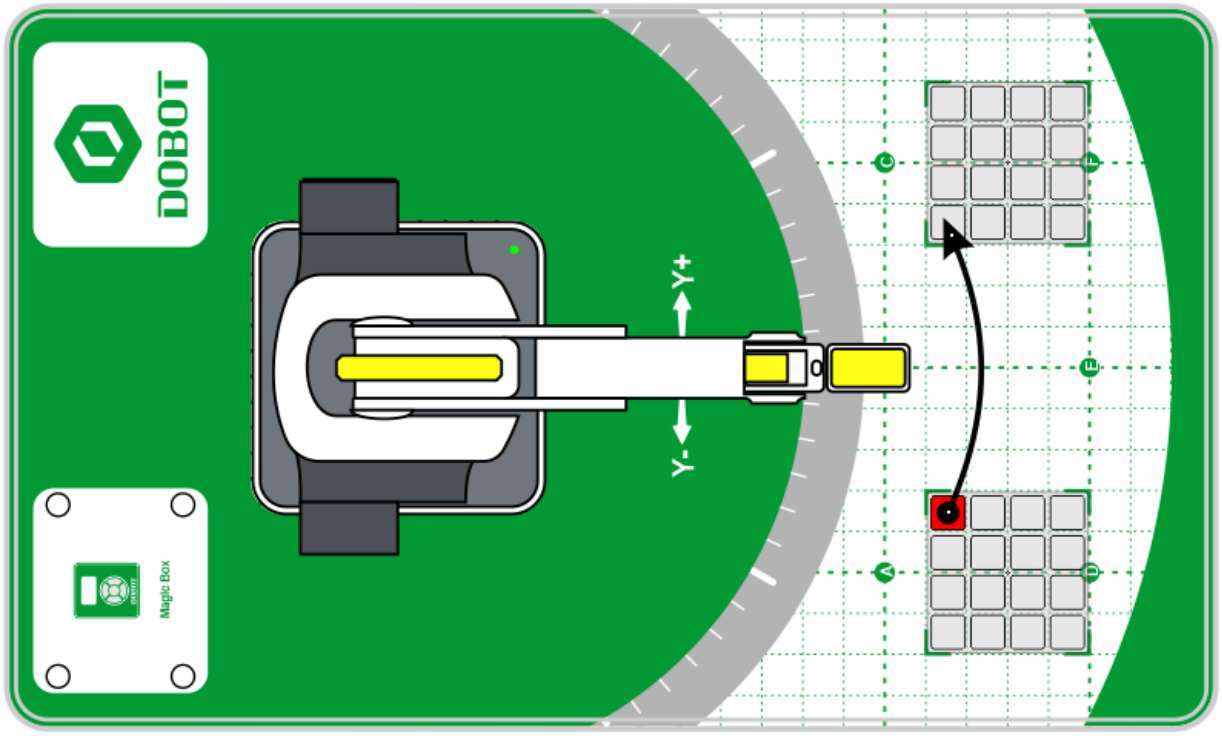
- STEP 1 - Push the *Suction Cup* in until it snaps in place.
- STEP 2 - Attach the hose to the air nozzle.

The suction cup works by creating a vacuum against a flat smooth surface.

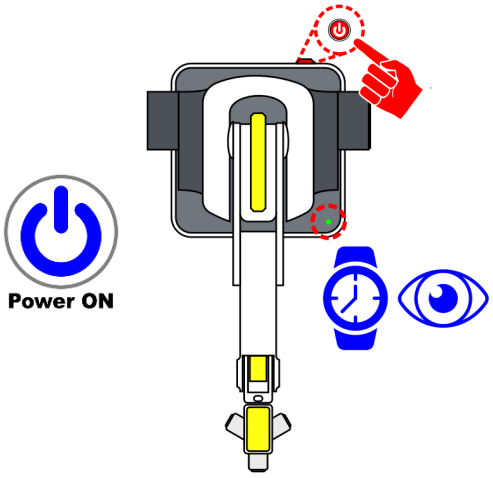
2. Set up the Magician Lite Work Mat or print one of the field diagrams from the ***Resources*** chapter of this curriculum.

The bottom of the suction cup is rubber and acts like a spring. When picking up an object, lower the Z value just until the suction cup has started to compress. Try to center the suction cup on the object. Do NOT force the suction cup down too far!

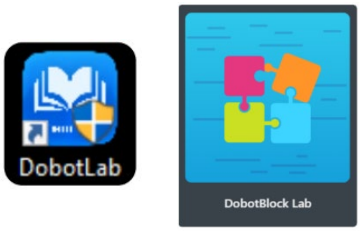




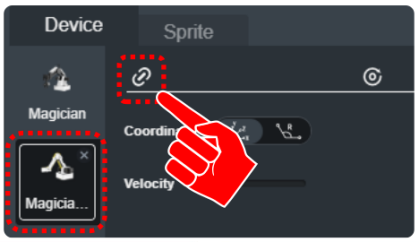
6. Once **ALL** of the wiring is done (sensors connected and robot arm connected), power ON the Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



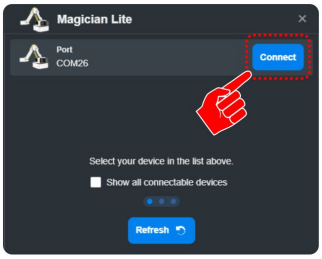
7. Open up DobotBlock Lab in the software and connect to the robot.

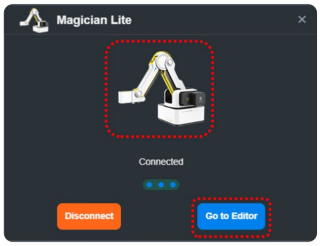


8. Select the connect icon to establish control from the software to the robot. A connection window should pop up.



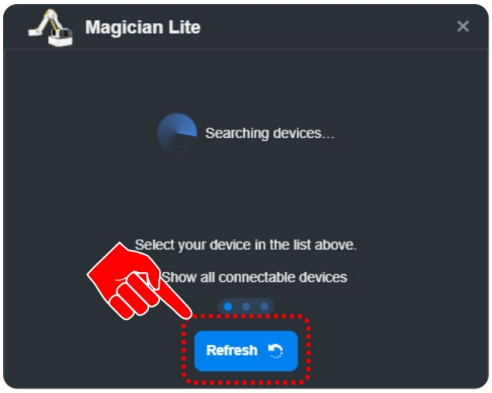
9. Select Connect from the pop-up window. A new window should appear and show that the robot is Connected.





NOTE: if the Magician is not on the Device list, scroll down to select the "Choose a Device" option

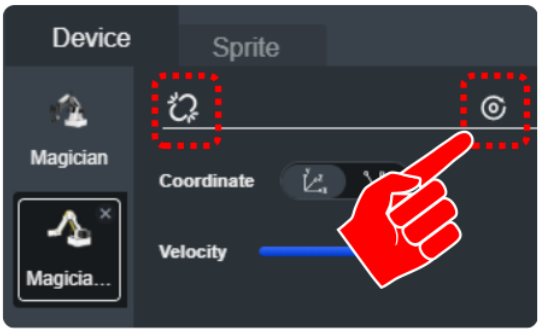
NOTE: If no robot is found, the prompt will show as "Searching Devices". Reconnect the robot and select Refresh



10. Once the robot is connected, we need to home the robot to set its home position.

Before **HOMING** the robot, make sure the robot's **WORK ENVELOPE**, the area in which the robot can reach, is clear.

Select the **HOME** icon to start the robots homing process. **HOMING** the robot will return and set the robot to its initial **HOME** position.



11. Next, choose the Suction Cup as the Dobot's **END EFFECTOR** or **EoAT** (End of Arm Tooling).

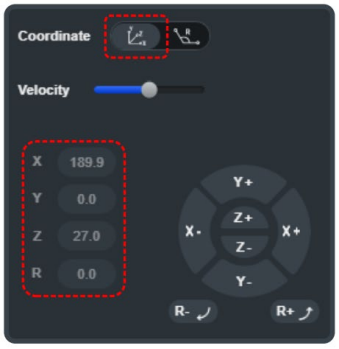
NOTE: You may need to scroll down to see the **EoAT** selections.

12. As with industrial robots, DobotBlock Lab provides the user with two methods for driving the robot manually. These options are XYZ MOTION and JOINT MOTION.

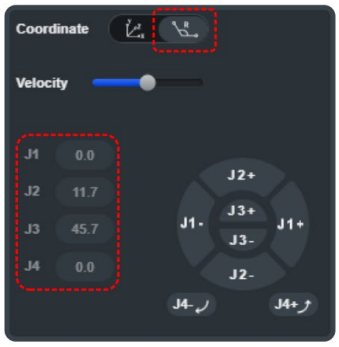
JOINT MOTION - This method of movement allows you to configure one joint at a time by **DEGREES**. All of the other joints stay in the same relationship to the joint being driven. Practice with this type of movement.

XYZ MOTION – XYZ Motion will move on multiple axis at one time by **ABSOLUTE COORDINATES**. The robot's motion is based on the keeping the EoAT in line with the axis chosen. Practice with this type of movement.

XYZ MOTION

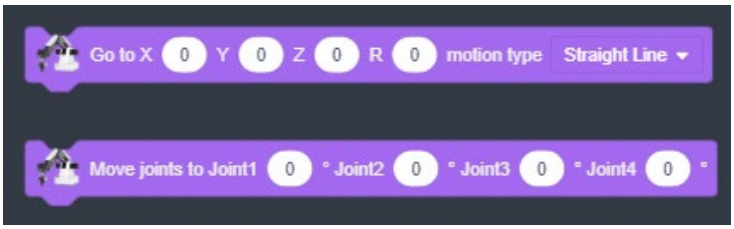


JOINT MOTION



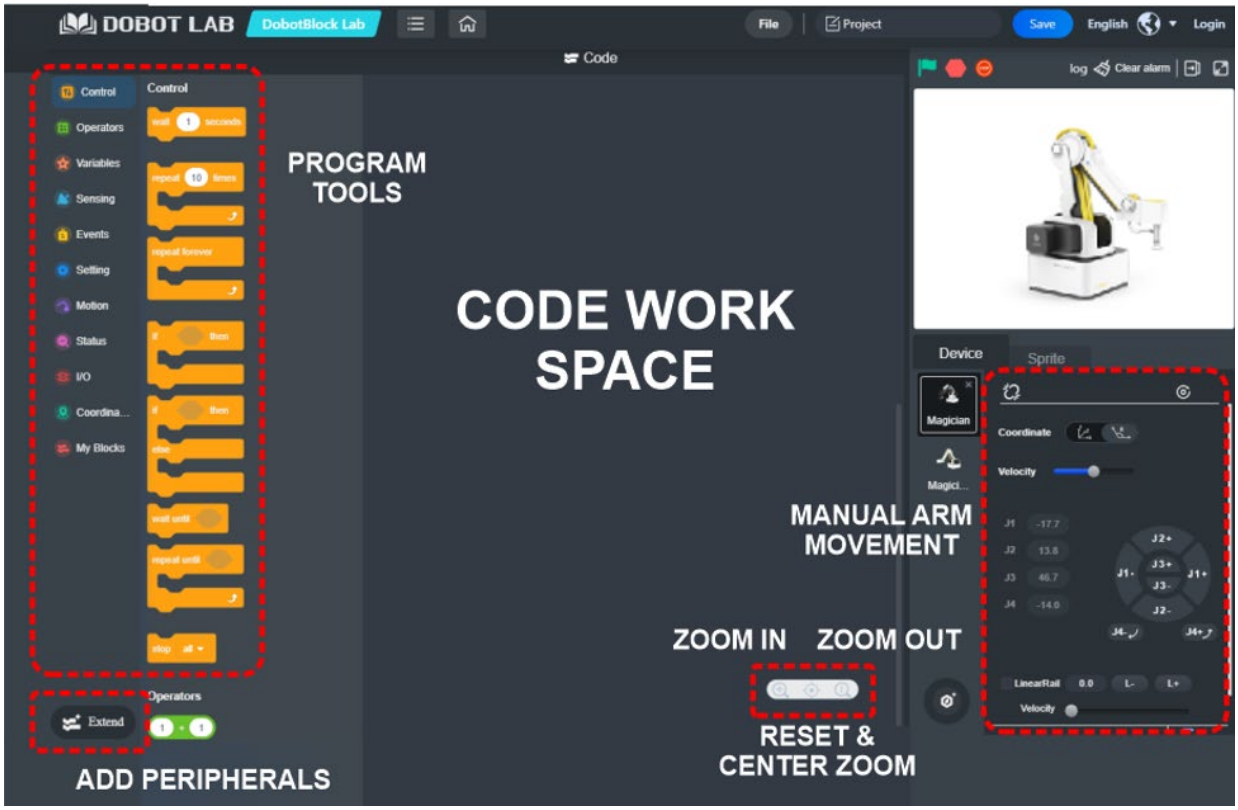
LINEAR MOTION	JOINT MOTION
X – In and Out	J1- Waist
Y – Side to Side	J2 - Shoulder
Z – Up and Down	J3 - Elbow
	J4 – Wrist *(Only aviable when the gripper or suction cup servo is attached)*

The position data can be collected from either method depending on the coordinate motion style used when developing a program.



Shown below is the main menu for the Blockly programming section of DobotBlock Lab. On the left are categories and tools that each block of code is sorted into to. On the right are controls for zooming around the program as you develop it. The scroll wheel on the mouse can also be used to scroll up and down. The scroll wheel or the left mouse button can also be pressed and held to pan throughout the program workspace.

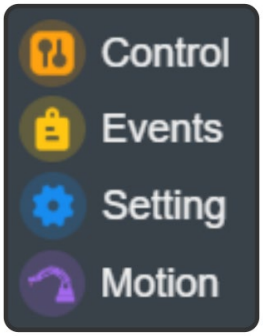
The Crosshair Icon can be used to reset the zoom level back to default and center your code in the middle of the field.



For this activity, we will be using tools from multiple categories. The categories we will be using are:

- Control
- Events
- Settings
- Motion

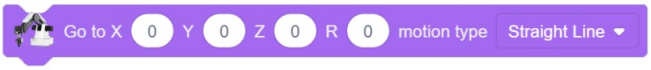
And can be found in the left menu as shown above.

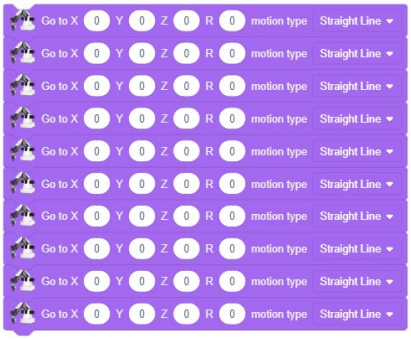


13. Each program will typically start with some setup code (EoAT, Motion Velocity, Inputs, Outputs….)

    For this program we will start by assigning the EoAT. Drag over the Set end effector tool from the **SETTINGS** category. Change the EoAT to the Suction Cup.

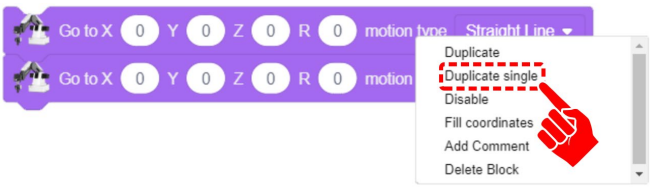
14. Next, drag over multiple "Go to XYZR" from the **MOTION** category (8 steps, one for each movement of a pick and place routine).



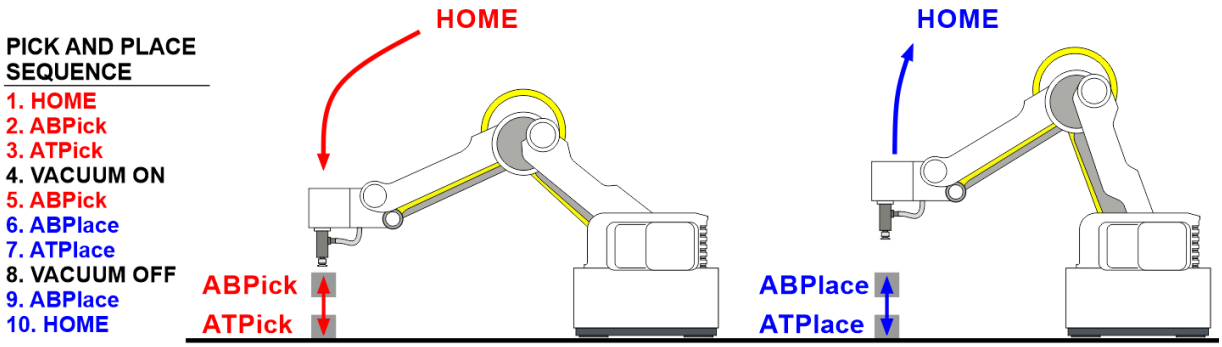


HELPFUL TIPS

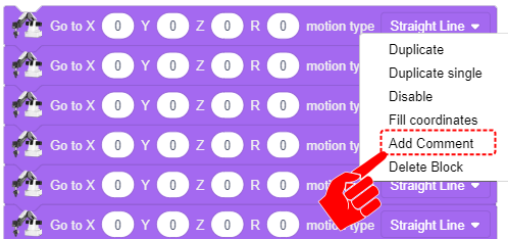
NOTE: you can also right click on an existing step in the program workspace and select DUPLICATE or DUPLICATE SINGLE.



DUPLICATE will create a copy of the current step and all the steps that are below and attached to the current step.

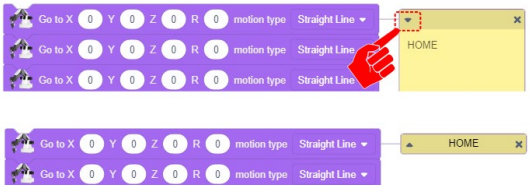


15. We will now add names/comments to each step. Right click on a step and select **ADD COMMENT**.



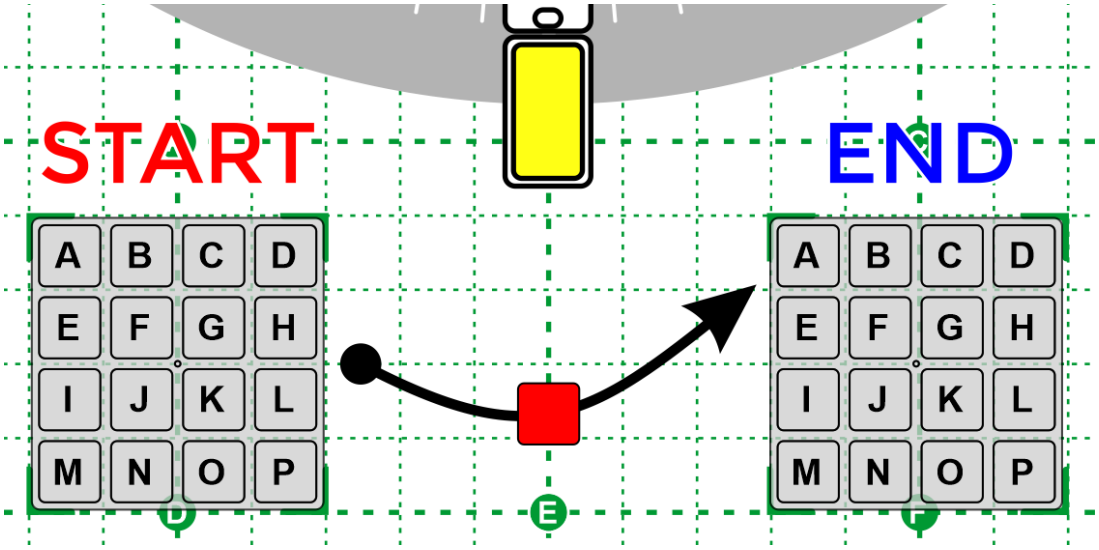
Type in the name of the step using the diagram above as a guide.

When the comment is complete, click the down triangle to collapse the comment.



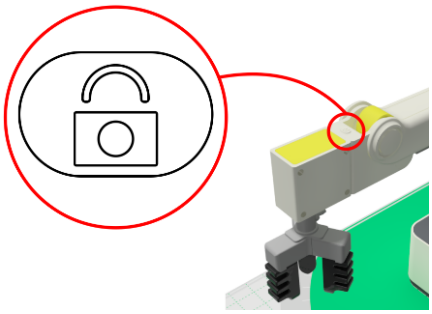
Repeat this process for all 8 movement steps.

16. Have your instructor identify your start and end positions.



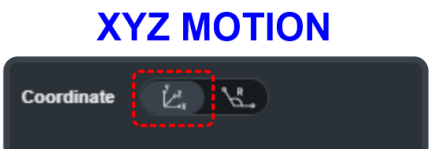
17. Now you will find the XYZR coordinates for each of the positions

    Press and hold the lock button on the arm of the robot. This will release control of the arm to the operator and allow you to move the arm to a new location. Once at the new location, release the lock button to have the arm regain control and hold that position.

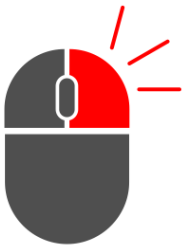


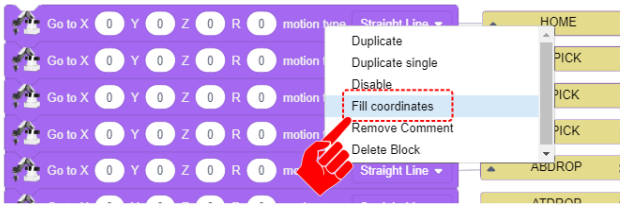
    Find the HOME position first. Using the lock button, move the arm to a new HOME position and release the button.

    Ensure the COORDINATES shown are the XYZR



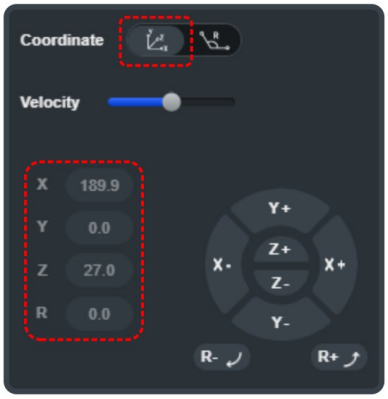
Right click on the 1st step (HOME) and select Fill Coordinates. This will use the robots current position to fill in the values for XYZR





It may be necessary to touch up the steps. For example, the X value of 189.9 can be set to 190.

Repeat this process for the remaining steps.



HELPFUL TIPS

Once the lock button is released, use the manual controls to fine tune the position or type in the desired values (round the numbers).

HELPFUL TIPS

When dragging around blocks of code, if you select the top block, it will allow you to drag around that block and all of the blocks connected below it. If you selected a block of code from the middle of a string of blocks, it will disconnect that block and all of the blocks below it.

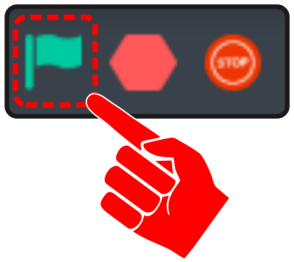
Complete the table below with all of the XYZ coordinates needed.

	X	Y	Z	R
1. Home				
2. Above Pick				
3. At Pick				
5. Above Pick				
6. Above Place				
7. At Place				
9. Above Place				
10. Home				

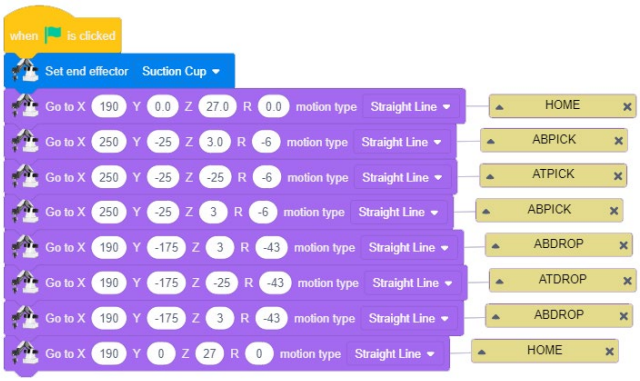
18. At this point we should test the positions we gathered. We will now perform a dry run (only perform movements) to see if our positions will work. To do this, the program needs a line of code that instructs it to run when the go button is selected.

19. Select the Go button (the GREEN flag) to run the program.

    From the EVENTS category, drag over the WHEN GO IS CLICKED tool and attach it to the top of your program.

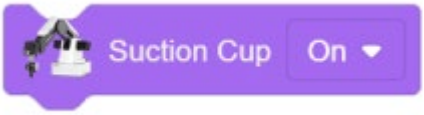


    Make any adjustments to program as needed.



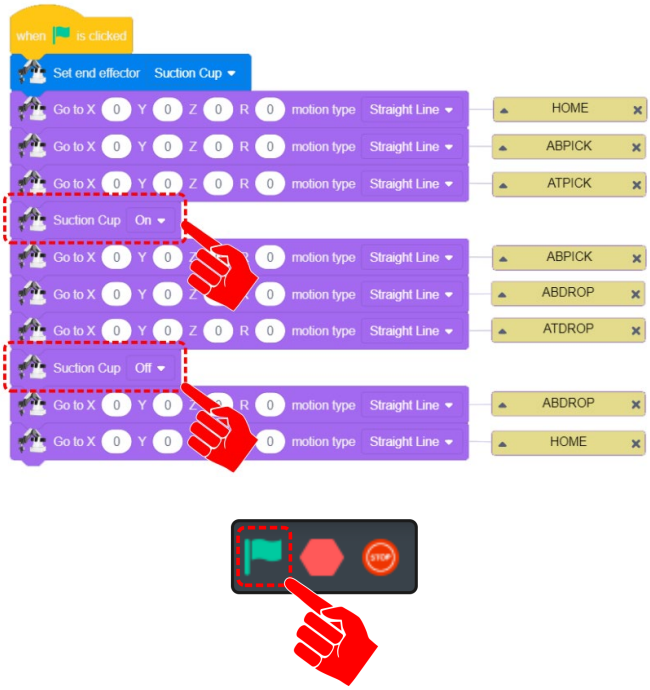
20. We will now add the steps needed to turn on and off the suction cup.

    From the MOTION category, drag over the GRIPPER tool.



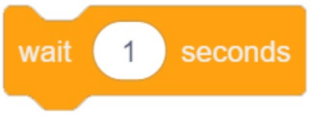
    We will insert this step into the program. As you drag the step over to your program, the program will automatically separate where you drag the tool. We need the GRIPPER to close after we are at the position ATPICK and open after position ATDROP.

Test the program and see if it works as expected.

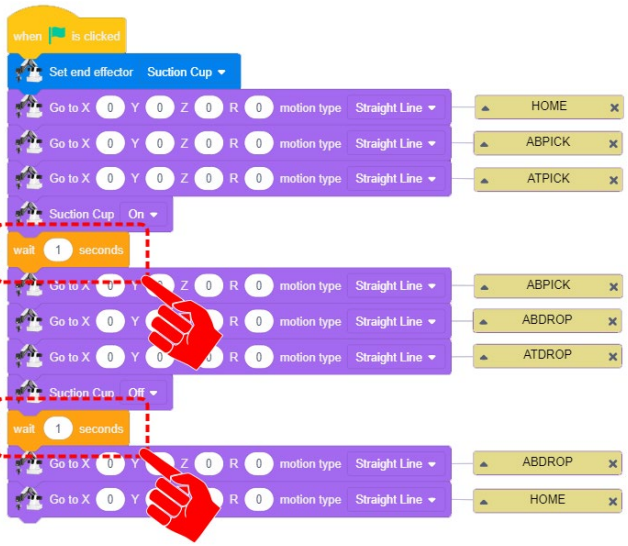


21. Finally, we will add a short pause after we turn on the suction cup and after we turn off the suction cup. This will ensure we have full pressure and have completely released the part before moving to the next step.

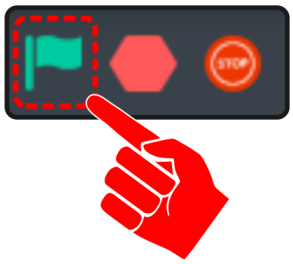
From the **CONTROL** category, drag over the wait/seconds tool.



Place a 1 second **WAIT** after each EoAT operation.



Test the program and see if it works as expected.



When dragging around blocks of code, if you select the top block, it will allow you to drag around that block and all of the blocks connected below it. If you selected a block of code from the middle of a string of blocks, it will disconnect that block and all of the blocks below it.

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

CONCLUSION

1. *What are the five needed positions for a pick and place operation?*

2. *Explain in your own words why it was necessary to add delay times into the program in the space below.*

3. *What is the purpose of the safe positions that are programmed above the object before it is picked up?*

GOING BEYOND

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

_____ 1. Reverse the process so that at the end the robot puts the cube/cylinder back in its original position

_____ 2. Try picking and placing the object on locations that are not directly in front of the robot. What does this change? Can this be corrected in Blockly as it was in Teach and Playback? How?

_____ 3. For the sake of accuracy, change the speed of the arm as it moves from the ABPICK to the ATPICK, and ABDROP and ATDROP to slow it down by half (The codes needed to control speed are in the *SETTINGS* toolbox).

1.4 TP - Using Jumps and Loops

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

INTRODUCTION

A **PICK AND PLACE** style of moving objects around are staples of industrial robots. Another reason to use robots in industry and automation is because of human safety. Robots can work in adverse environments that are dangerous to humans; especially when dealing with chemicals and other toxic substances.

Sometimes the best way to pick up an object is with a suction cup or **VACUUM GRIPPER**. This works especially well with very small objects and provides a less expensive alternative to a **MECHANICAL GRIPPER** or a **SOFT GRIPPER**.

In this activity, you will perform an anodizing operation (which can use harmful chemicals) with a robot arm using the air pump kit and a vacuum gripper.

Do not drop the object in the tanks… wait for the 2 second dip to complete and move on.

Dip Tank Set Up *Use the printed field diagram with the dip tank locations for this activity. 3D printed dip tanks can be added for a more realistic activity. and a Dobot field diagram. In this diagram four tanks are used.*

HELPFUL TIPS

Remember: *We always go to a position above the pick or place point first so that the robot places the object straight down, with no friction or interference. This will increase accuracy!*

KEY VOCABULARY

- TCP
- Loop
- Touch up
- Teach
- Suction
- Vacuum
- Linear movement
- Joint movement

All blockly commands and robotics terms have been put into a separate *glossary* that can be founded in the ***Resources*** chapter of this curriculum.

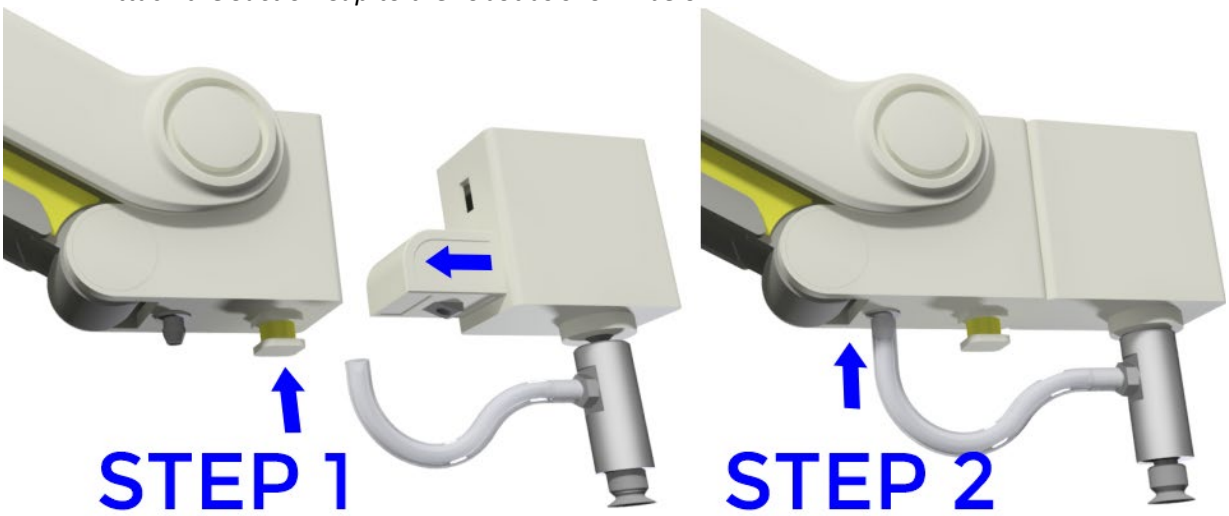
EQUIPMENT & SUPPLIES

- Dobot Magician Lite
- Dobot Dipping Field Diagram
- Small Cubes & Pallets
- Suction Cup Gripper
- DobotLab software
- *Optional: 3D printed dip tanks*
- *(1" PVC Caps or small plastic cups work well)*

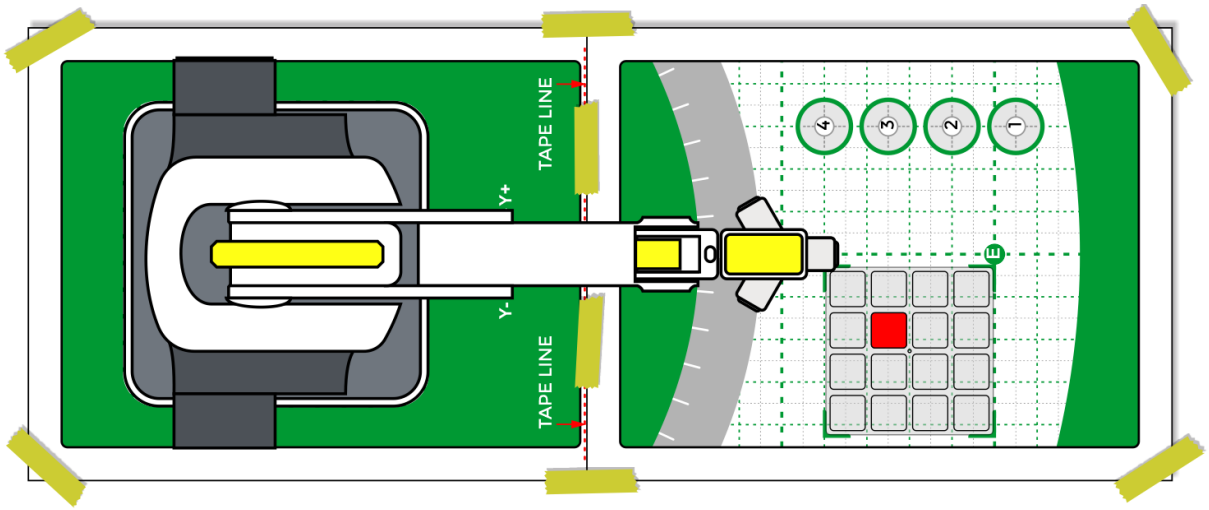
PROCEDURE

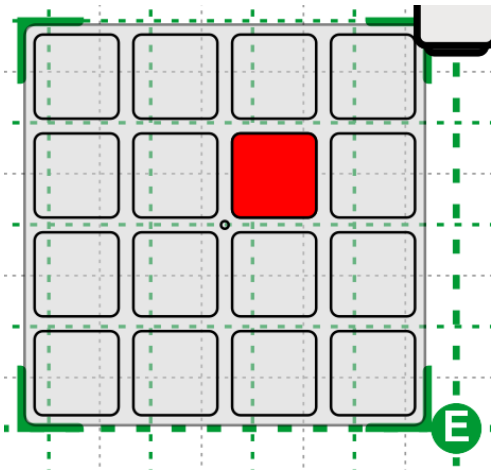
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

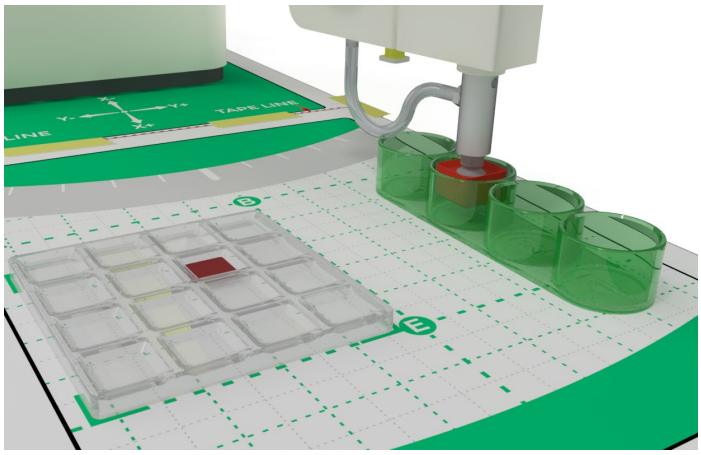
1. Attach the *Suction Cup* to the robot as shown below.



2. Set up the dobot dip tank field diagram as shown below.

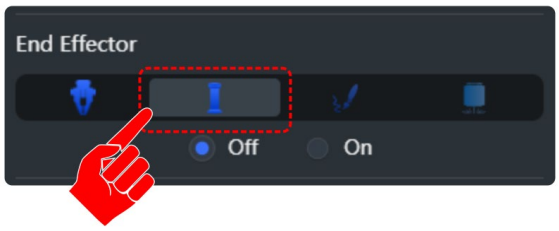




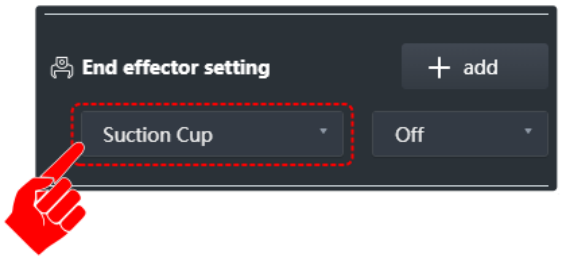


3. Open DobotLab software, open the Teach and Playback Lab, connect and home the robot, and be sure the *Suction Cup* is chosen as the **END OF ARM TOOLING (EoAT)** in both the **ARM CONTROL PANEL** and the **COMMAND CONTROL PANEL.**

Arm Control Panel

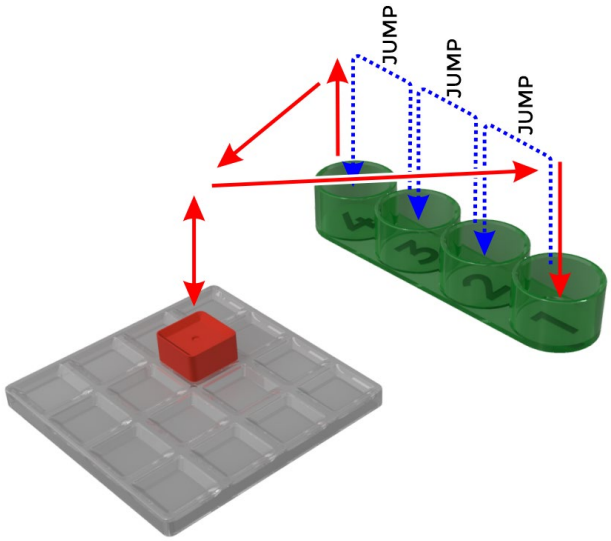


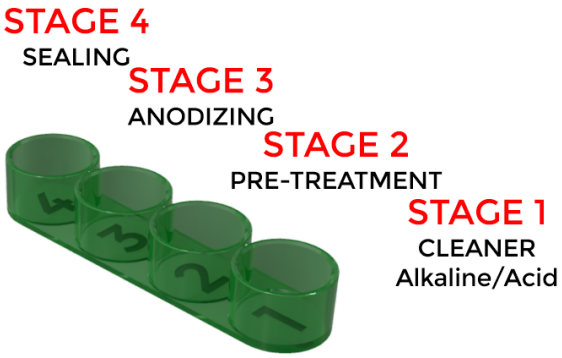
Command Control Panel

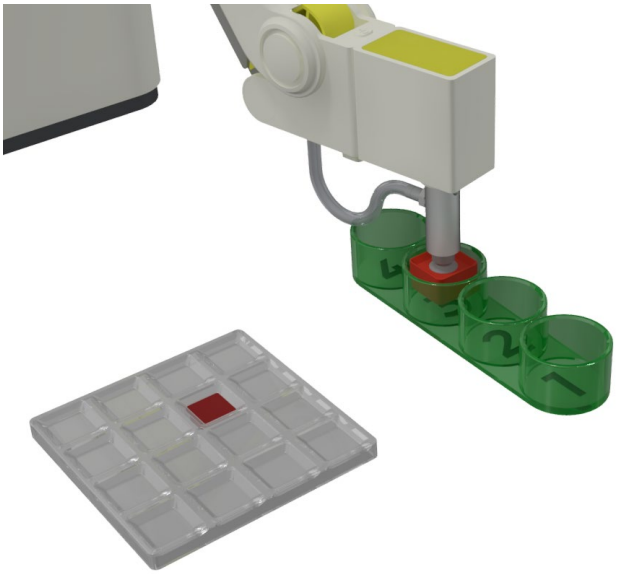


4. Once the robot is setup, create the rough sequence to perform the following actions:

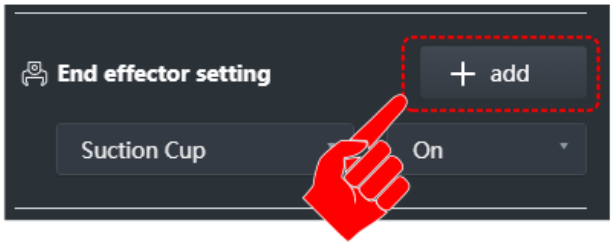
   a. Go home
   b. Pick up the cube from the pallet
   c. Dip it in Tank 1 for 2 seconds
   d. Jump the object and dip it in Tank 2 for 2 seconds
   e. Jump the object and dip it in Tank 3 for 2 seconds
   f. Jump the object and dip it in Tank 4 for 2 seconds
   g. Move the part back to the pallet.
   h. Go home



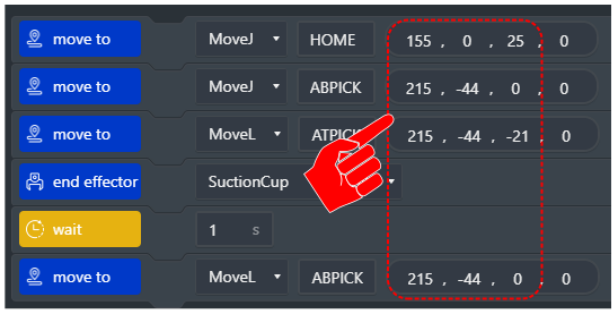




5. Be sure to turn on/off the *Suction Cup* when necessary.
6. Hit the *Play* button to run your program and see what happens. *Did it work the first time? If not, what did you have to change to make it work?*



5. When recording points using the *lock button*, it is not always very accurate. When you look at all the Z values of all the above tank positions, they are probably very different.



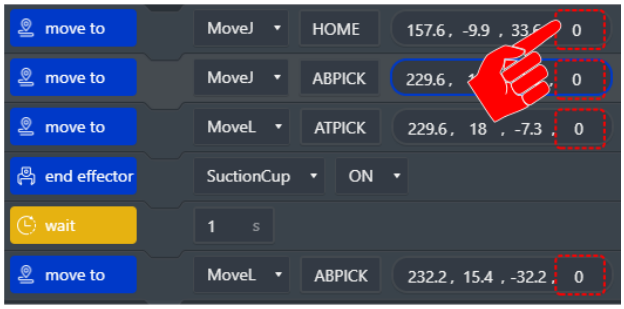
We can make them all the same by clicking on the values and typing in a value that works well; this is called **TEACHING**. When fixing points like this it is also called **TOUCHING UP** points.

Go and **TOUCH UP** all the z values to make the dipping operation happen at the same heights above the tank, and in the tank and have your instructor check it.

HELPFUL TIPS

You can even use this method to touch up the x and y values of the tanks to make sure they are in a straight line.

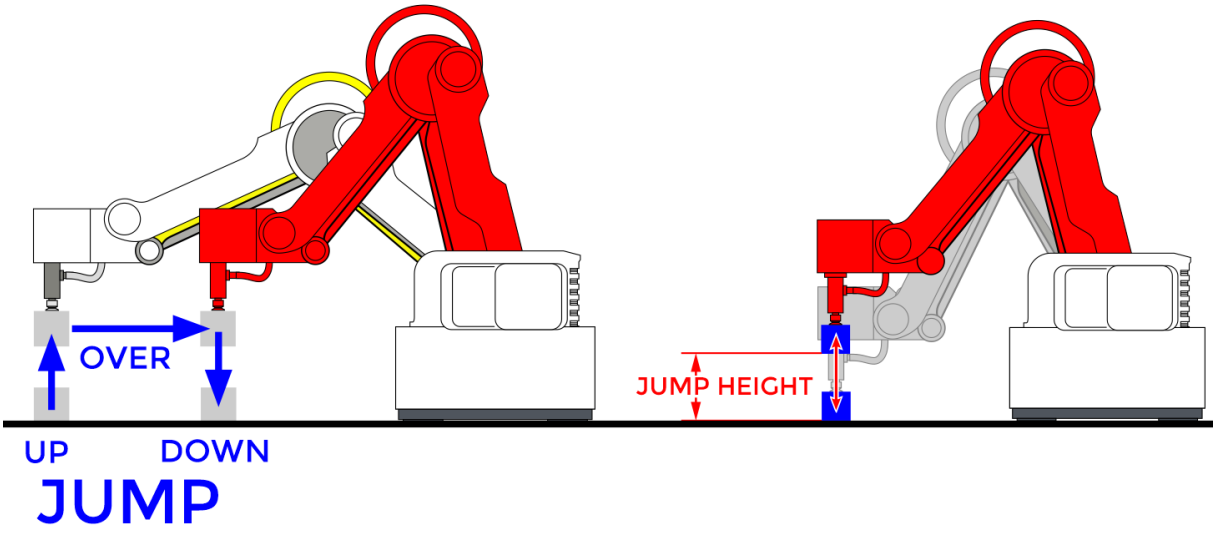
If you are using cubes, change all of the Roll Angles to zero as we did in Activity 2.



6. How many lines of code did it take to write the program using what you have learned before? How many points did you have to teach it?

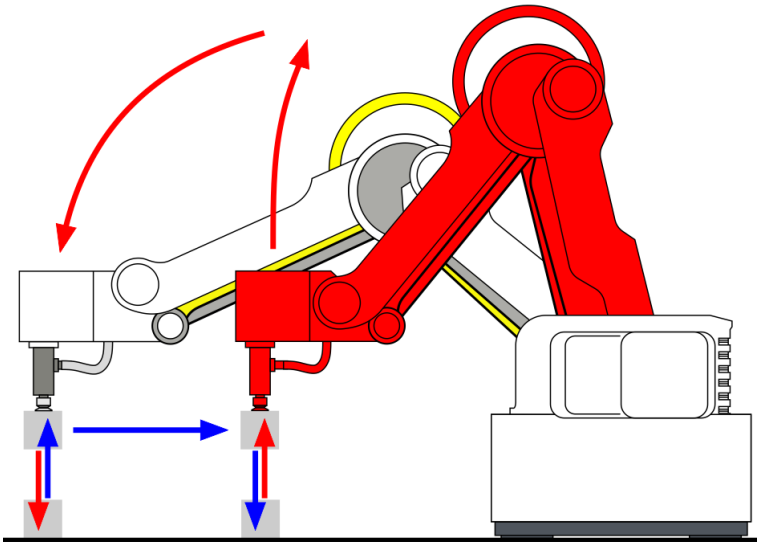
7. Now we are going to use the **JUMP** command to make the program even easier, and much more accurate. Open a new program.

What is a **JUMP**? A JUMP movement combines three steps into one. It combines the raise up, over, and back down to the same Z level at new location. This type of movement simplifies repetitive movements such as a dipping operation or soldering operation. The Z Height is defined in the JUMP parameters in the settings menu.



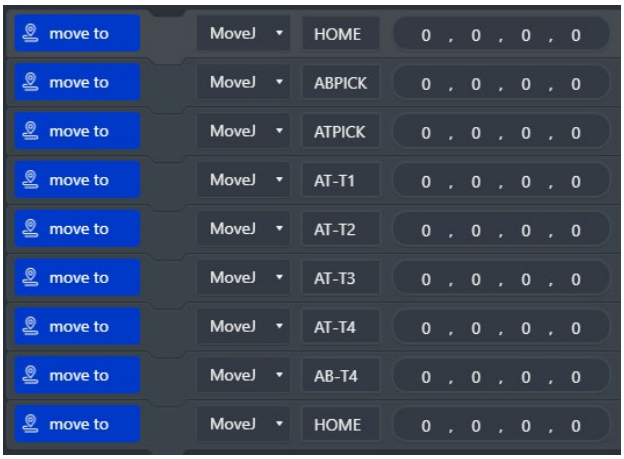
A JUMP movement does not replace the initial ABOVE position or the initial AT position or ABOVE the final position. It is only used to RASIE UP / MOVE TO A DIFFERENT POSITION / RETURN TO Z FROM LAST POSITION.

EXAMPLE:
HOME
ABPick
ATPick
JUMP to ATPlace (Blue)
ABPlace
HOME



8. This time, only record the following points:
Home,
Above Pick and At Pick,
At Tank 1,
At Tank 2,
At Tank 3,
Above Tank 4 and At Tank 4
Home

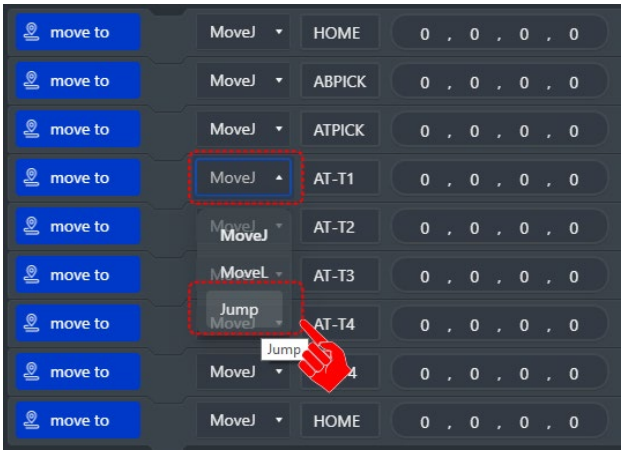
Do not record any above points except for Above Pick and Above Tank 4. Jumps will replace these points

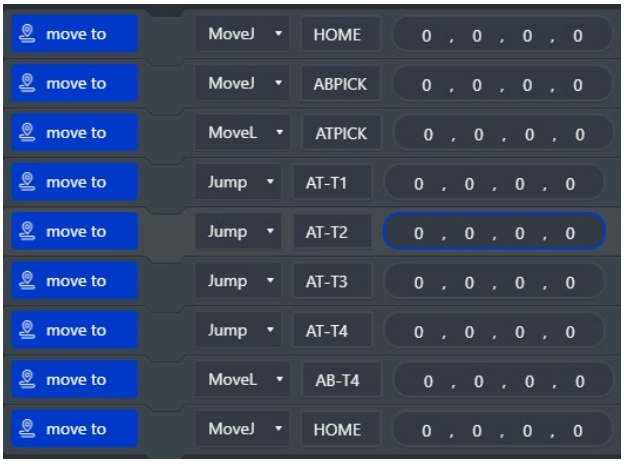


9. Now change the **MotionStyle** of the Tank 1, Tank 2, Tank 3, and Tank 4 to *JUMP*. Double click on the **MotionStyle** box for each point and change it from *MOVJ* to *JUMP.*

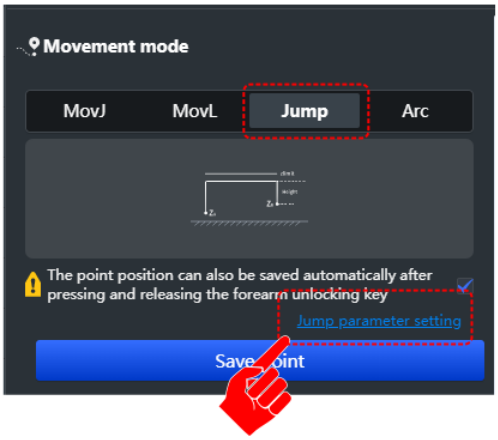
Now play the program.

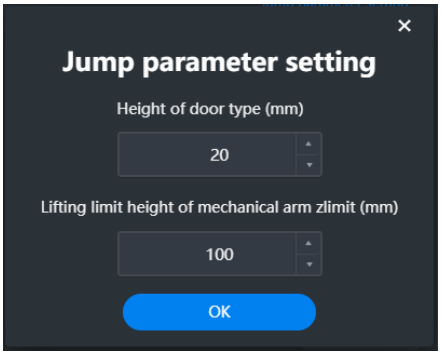
What does this change do to the program?





10. If you ever want to change the height in Z that the *JUMP* actually jumps to, open the *Jump Parameter Settings*. You can edit the *Height of Door* to change the *Jump Height* to whatever you need it to be to clear the tank.



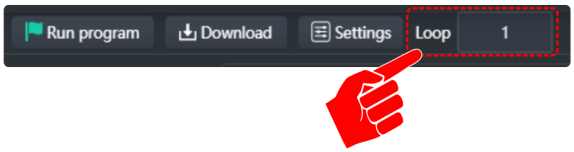


Compare the first dipping program to the second one with JUMPS and explain how it is more efficient.

11. Now add the missing steps to finish the sequence.
    - Move Linear Movements (MoveL)
    - Pause for Time
    - Suction Cup On and Off

12. Next, add the ability to make the robot repeat the operation five times. Do this with a process called **LOOP. LOOPS** will make something happen multiple times so that our program is shorter and more efficient. In the TEACH AND PLAYBACK LAB, the loop function will loop the entire program from start to finish. We do not have the ability to control how much of the program will loop or even if it should loop (a conditional loop).

13. To make your program **LOOP** multiple times, just add the number of times you want it to repeat in the loop window.



14. Be sure to save your work when done.

CONCLUSION

1. *Describe a MOVL move using the Dobot.*

2. *Describe a MOVJ move when using a Dobot.*

3. *What makes a Jump command more efficient to code than a MOVJ or MOVL?*

4. *What kind of manufacturing process is dipping? Explain your answer.*

5. *Anodizing is a manufacturing process. Explain how it works and why we do it.*

GOING BEYOND

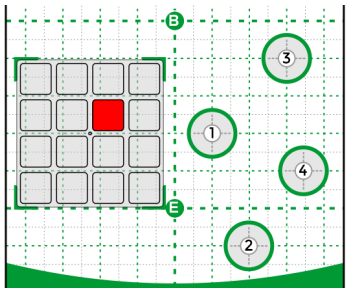
Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

_____ 1. Take away the first tank and change the program to fix the operation using the software.

_____ 2. Add another tank and change the program to fix the operation using the software.

_____ *3.* Raise the tanks using a block of wood provided by your instructor and fix the operation using the software. *What is the most efficient way to do this?*

_____ 4. Randomize the tank patterns for the dipping process

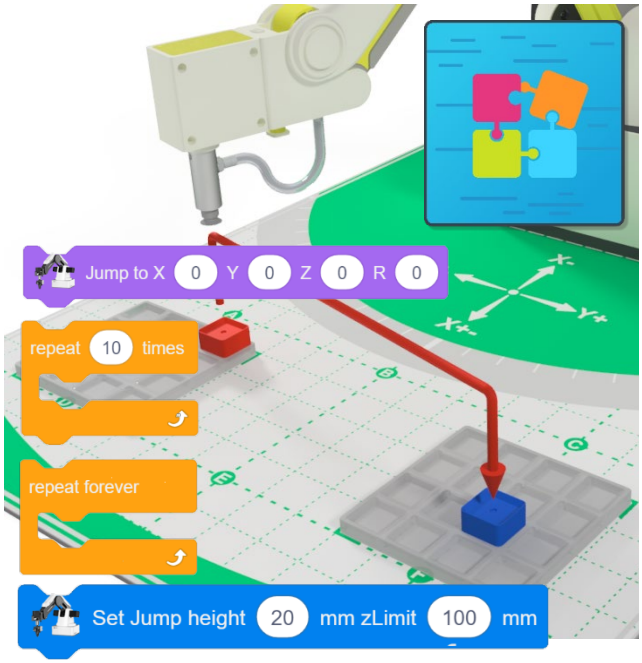


1.5 Block - PnP with Jumps and Loops

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

INTRODUCTION

When programming a robotic arm, it often becomes necessary to repeat operations a set number of times or indefinitely. This can be accomplished by adding different styles of loops to our program. It is also a good programming habit to optimize or reduce your lines of code when appropriate.



In this activity you will learn one way to simplify your code as well as add different styles of loops to a basic Pick and Place operation in DobotLab.

The two main types of loops are:

- Forever loops
- While loops
- Repeat

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
- Repeat
- True
- Jump
- Placeholder
- Condition

HELPFUL TIPS

All blockly commands and robotics terms have been put into a separate *glossary* that can be founded in the ***Resources*** chapter of this curriculum.

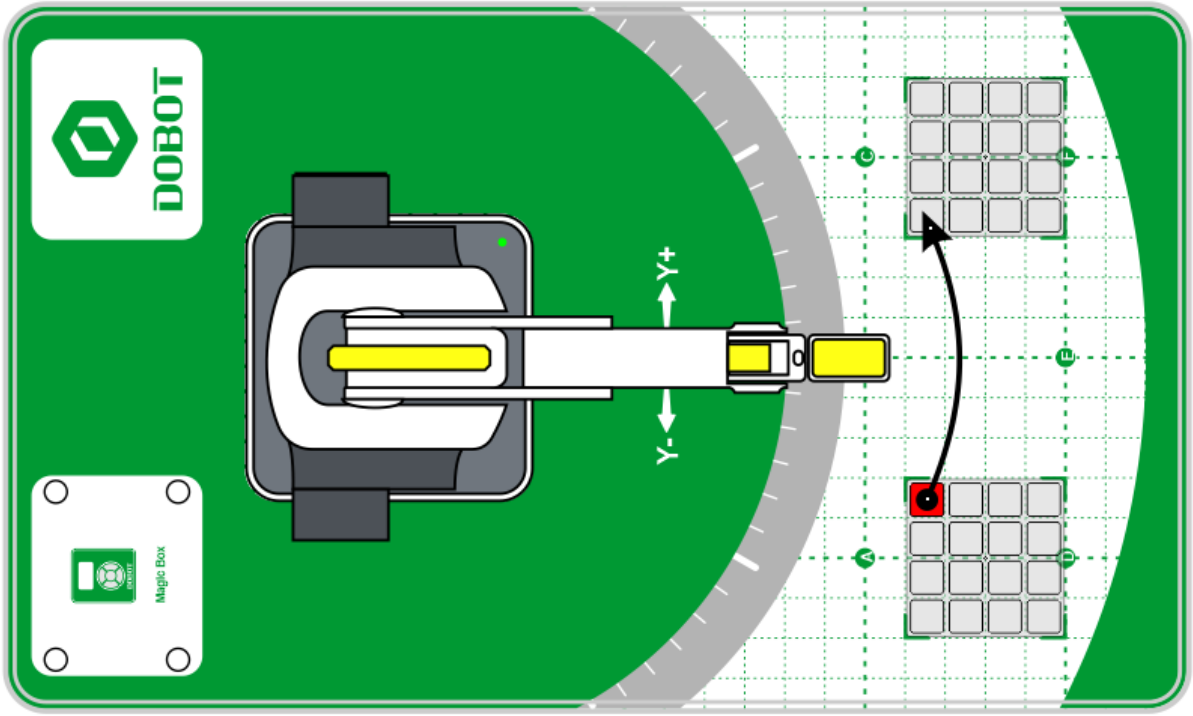
EQUIPMENT & SUPPLIES

- Robot Magician Lite
- Dobot Dip Tank Field Diagram
- Magician Lite Work Mat
- DobotLab software
- Suction Cup Gripper
- Small Cubes & Pallets

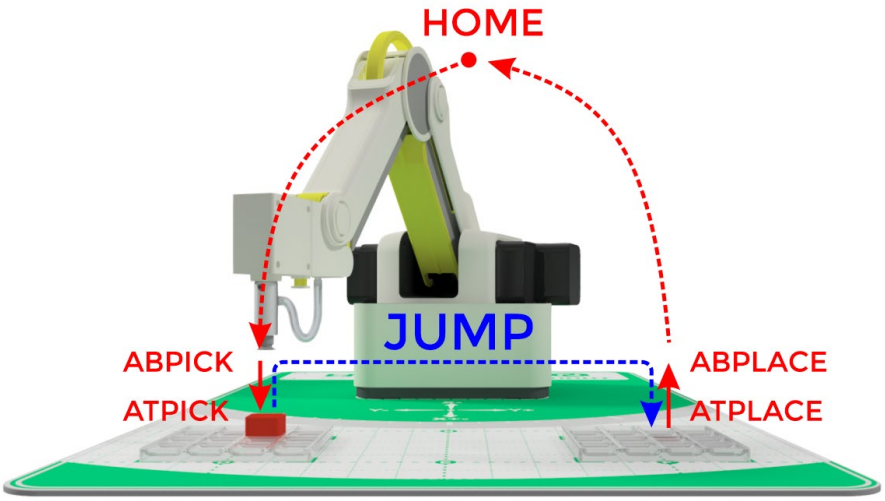
PROCEDURE

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. Set up the Dobot in the same configuration from activity "1.3 Pick and Place."



2. In this activity, use the JUMP command to simplify the original PNP activity from 1.3.

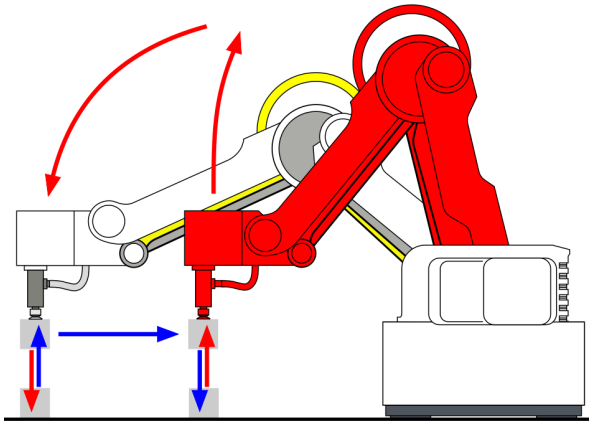


3. Open up the DobotBlock lab in the DobotLab software.

4. Open your Pick and Place Blockly Activity.

In this activity, we will add a ***JUMP*** movement as well as a ***LOOP***. The JUMP command in blockly works the same as it does in Teach and Playback. This will allow us to remove a few lines of code, creating the same operation in less steps.

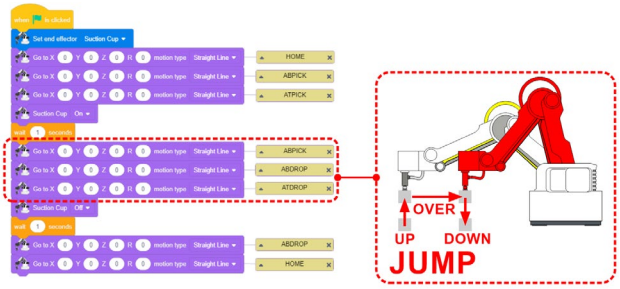
JUMP Movement - A ***JUMP*** movement combines three steps into one. It combines the retract, move over, and back down movements into one step. This type of movement simplifies repetitive movements such as dipping operations or soldering operations. The Z Height for the JUMP movement is defined with a **SETTING** block.



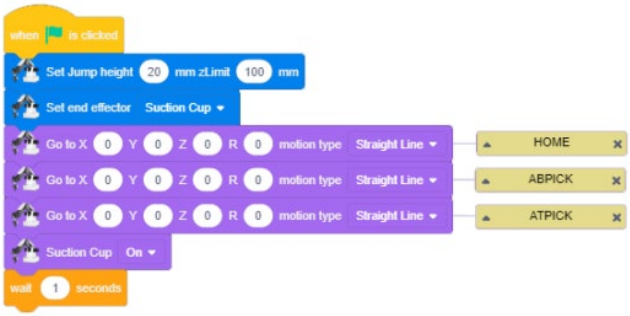
A ***JUMP*** movement does not replace the initial ABPick to ATPick as well as the final ABDrop.

First identify what portions of the last program, if any, can be replaced with a **JumpTo** command.

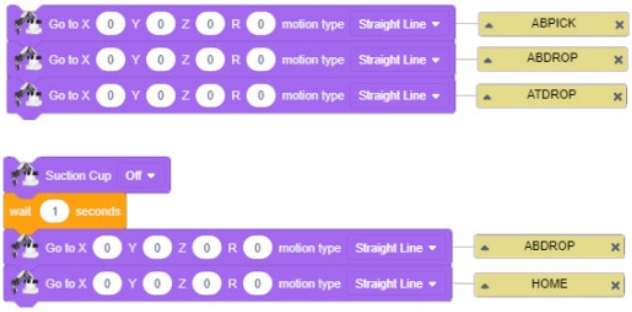
After the suction cup is turned on, there is a group of code that is an UP-OVER-DOWN. We will combine these three steps into one **JUMP**.



5. Isolate the section of code to be replaced and pull it off to the side (we still need the ATDROP coordinates).

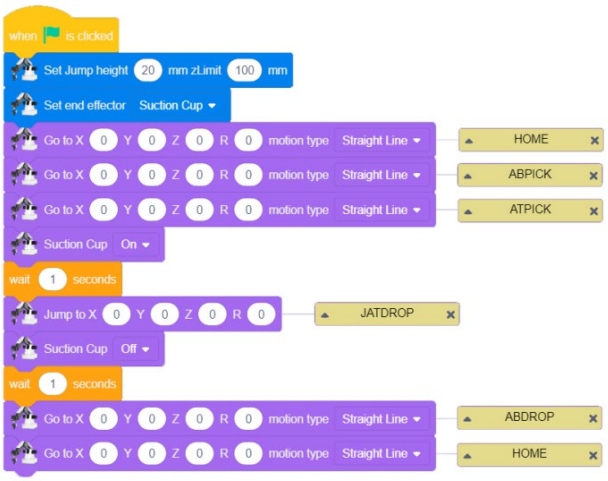


Does the Jump command make writing a program any easier?

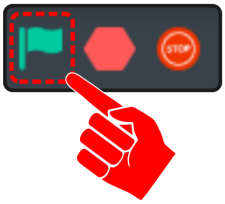


6. From the MOTION category, drag over a **JUMP TO XYZR**. Fill in the coordinates with the values for **ATDROP** (the final point we would like to **JUMP** to) and add a comment the block. Reassemble the code with the new **JUMP** line.

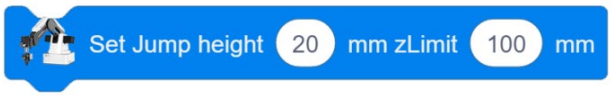
That definitely makes your program shorter and more efficient.



7. Run your code to ensure proper operation.



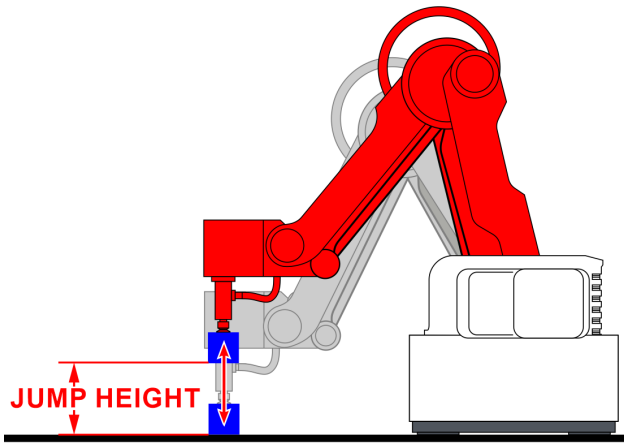
8. Next, we want to add a command that will allow us to control the **JUMP** Z Height. Drag over a **SET JUMP HEIGHT** from the **SETTING** category.



The JUMP height is a relative coordinate, not an absolute coordinate. We want the jump height to be the distance we want the JUMP to travel up for its current location.

Example:
ABPICK Z = 2.0 ABSOLUTE
ATPICK Z = -33 ABSOLUTE

JUMP HEIGHT = (33+2) = 35



HELPFUL TIPS

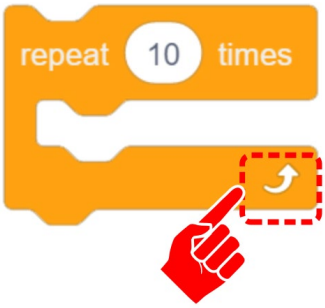
Since this is a setting command line, we will add it to the top of our program where we assigned the EoAT.

HELPFUL TIPS

This step can be added anywhere in the program, as long as it is placed before the jump command is used.

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

9. Next, we are now going to make the program loop, or repeat, a certain number of times. To do this we will be using the **REPEAT** block from the CONTROL category. and the *while* loop, both can be found in the *Loops* section on the left.

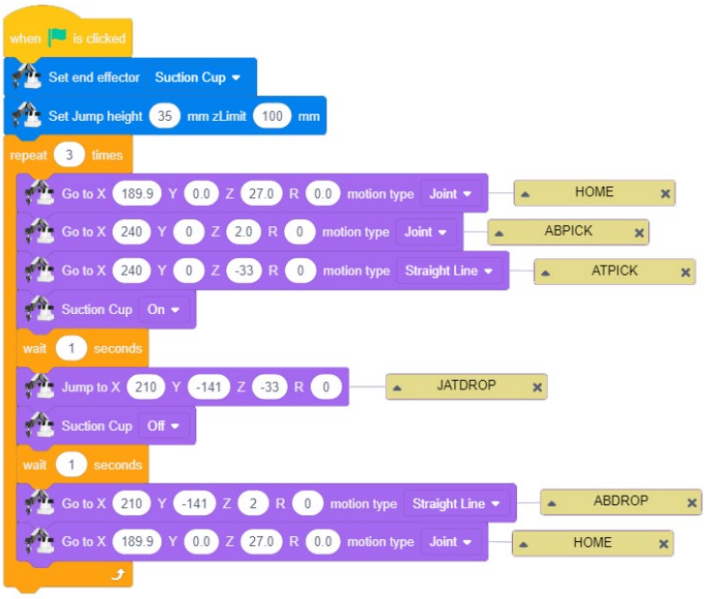


HELPFUL TIPS

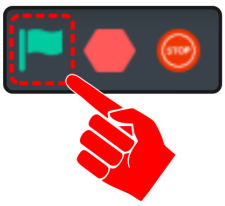
The white arrow at the bottom of a C Block *(Referred to as a "C" block because of its shape)* indicates that the original condition will be rechecked to see if it is still true at the end of each loop. If it is, the loop will run again. In this example, our condition is the number of times to REPEAT. The REPEAT will loop until the condition is no longer true.

Drag only the operations we wish to repeat into this "C" block. We will leave the rest of the program outside of the loop.

Edit the REPEAT block to make this section run 3 times

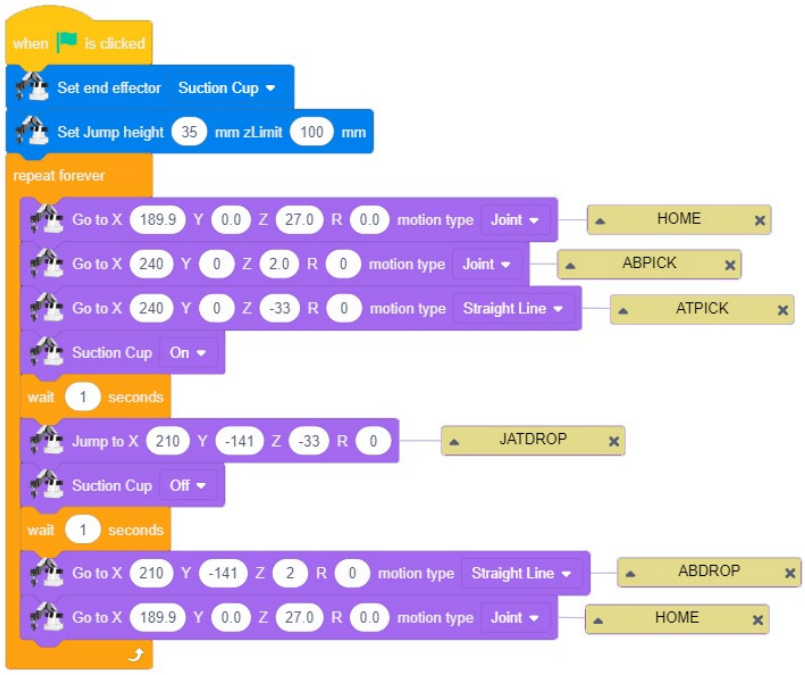


10. Run your code to ensure proper operation.



Often, we need a section of code to run **FOREVER** instead of a set number of times. TO do this, we will use a **FOREVER** block. Drag over a **FOREVER** block from the **CONTROL** category. Replace the **REPEAT** block with the **FOREVER** block and test your program.

Run your code to ensure proper operation.



HELPFUL TIPS

*If we select the **Repeat** Loop to be deleted before the program section is pulled out of the loop, it will delete the loop and everything it contains.*

Notice that once the program drops off the block and returns to its safe position, it goes back to the start of the program to try to pick up another block and that this will continue repeating indefinitely until you hit the stop button, or the robot is turned off.

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

CONCLUSION

1. *What is the difference between a forever loop and a repeat loop?*

2. *Can a loop be placed inside another loop? Give an example of how you might use this when programming in Blockly.*

3. *Describe in your own words how a JumpTo command works. Why is it good programming practice to use JumpTo's?*

GOING BEYOND

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

_____ 1. Nesting Loops

Produce 2 different pick and place routines

Routine 1 will repeat three times

Then routine 2 will repeat twice

This process will loop routine 1 and then 2 forever

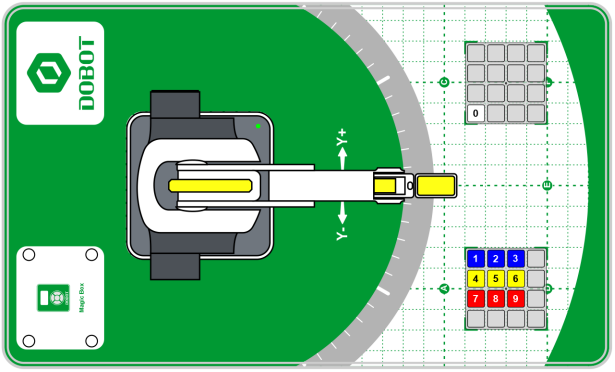
_____ 2. Produce a repeat loop for a dipping operation. Use the JumpTo when appropriate

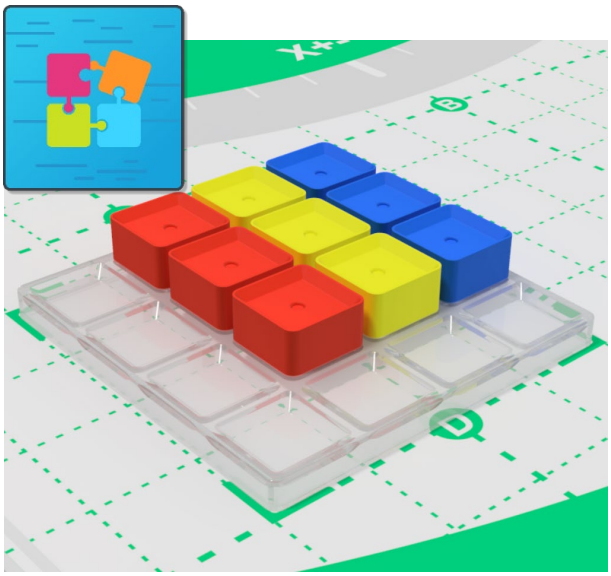
1.6 Block - Loading & Unloading Pallets

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

INTRODUCTION

Pallets come in many different sizes throughout industry and are used to stack, store and transport goods in trains, ships and trucks. Many companies will **Palletize** containers in order to efficiently store products and materials, so it is important for you to know how they work and how to program one yourself.





In this activity, you will simulate removing boxes from a pallet. This process is called ***De-palletization.***

KEY VOCABULARY

- Forever Loop
- Repeat Loop
- Matrix
- Suction Cup
- Delaytime
- Function
- User-Defined Function
- Inputs
- Variable

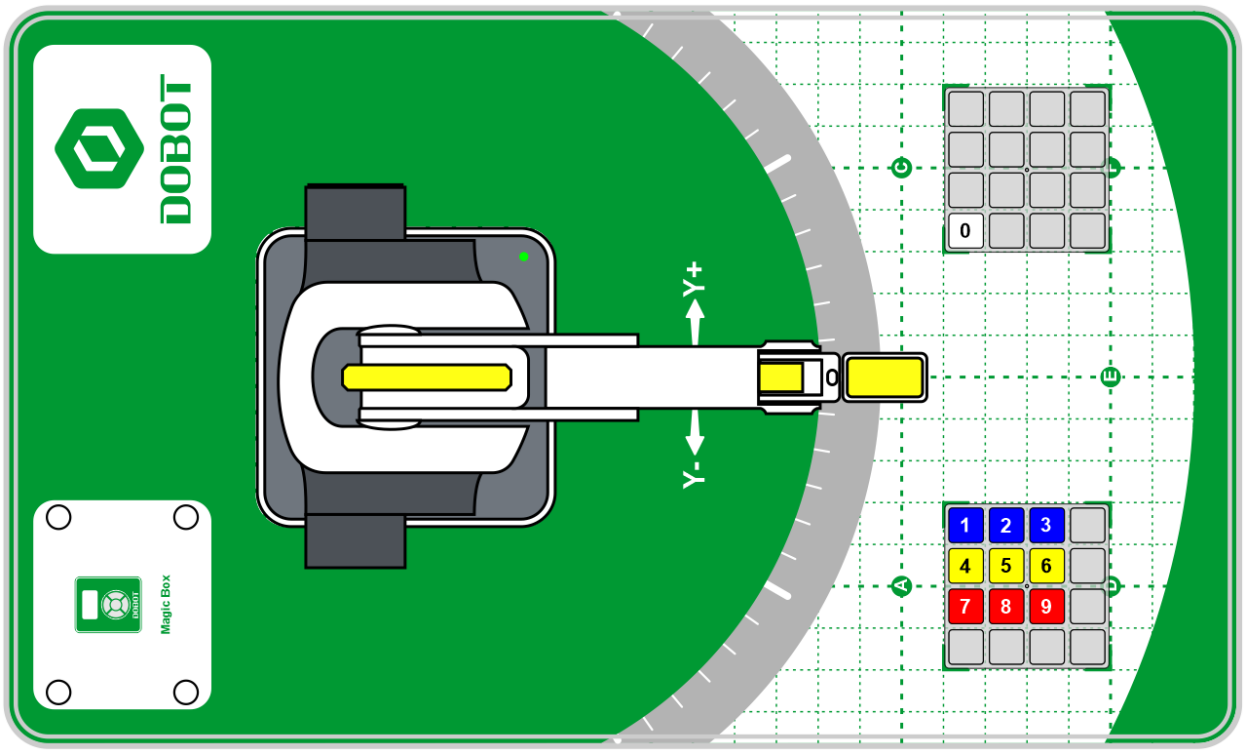
EQUIPMENT & SUPPLIES

- Robot Magician Lite
- Magician Lite Work Mat
- Small Cubes & Pallets
- DobotLab software
- Suction Cup Gripper

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. Set up the robot with the Magician Lite work mat with the pallets and cubes as shown below.
2. Set up the robot with a suction cup and place a cube on all of the empty squares on the field diagram provided. *(Cube color does not matter for this activity)*



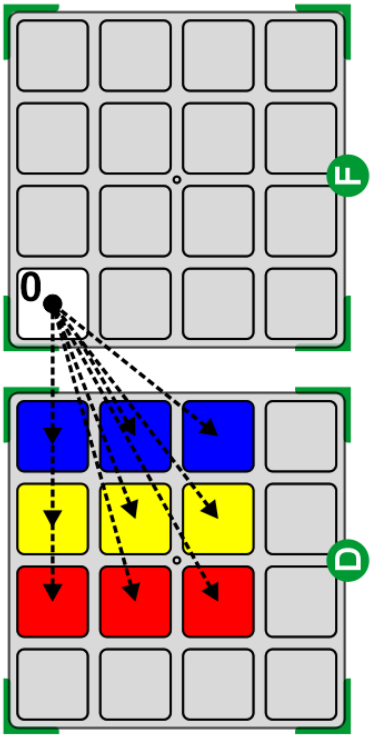
1. Open up DobotBlock Lab in the software and connect the robot.

Programming the robot to pick up each cube from a matrix of cubes and drop them all off at a common location is not necessarily hard to do, just tedious, time consuming, and numerous lines of code that are repetitive motions. This same process can be done in reverse, such as when you move parts from a feeder and place them on a pallet. This is called *palletization*.

From this activity we can learn how to condense and simplify our program when we have things in common or repeated.

In this exercise we have the following constants:

- Above Z locations are the same
- At Z locations are the same
- Place X and Y locations are the same
- They are all EVENLY spaced in a regular matrix.

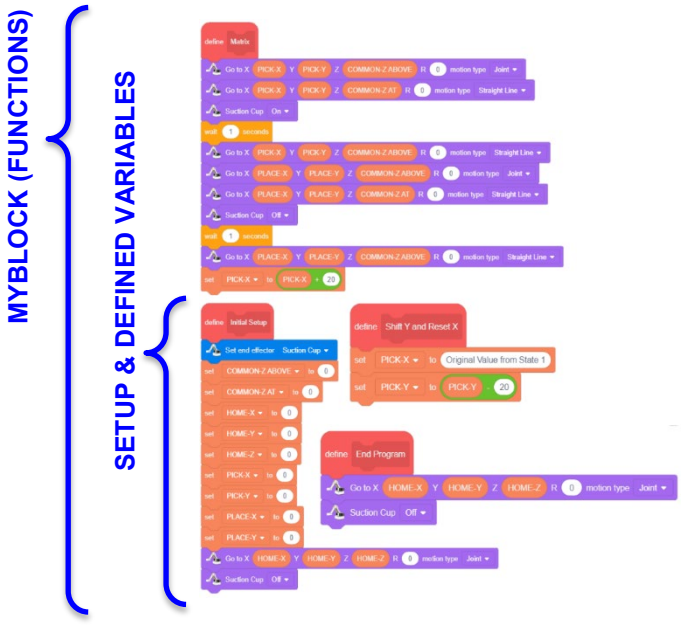


This is an example of palletizing where a robot takes boxes or parts from a part feeder, a conveyor belt, or a single pick up position and places them in a grid-like fashion on a pallet.

The only thing that keeps changing is called a variable. In this case it is the X and Y coordinates of the cubes on the pallet.

We can use these common factors to simplify our operations and our code. We will do this with user defined *variables* and *MyBlocks*.

MyBlocks are called *Functions or Voids* in programming language. These are just a named section of a program that performs a task that will be repeated over and over again in our program. This is a routine that will make a long program shorter.

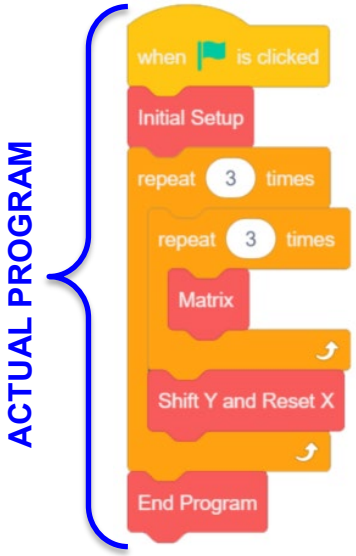


An example of a program that uses MyBlocks/ Functions

My Blocks are several lines of code that are needed to complete a single task or operation. They allow a programmer to group actions together.

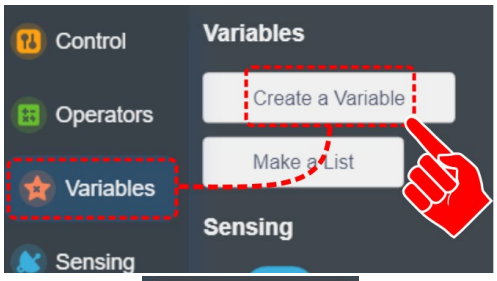
We use **My Blocks** when:

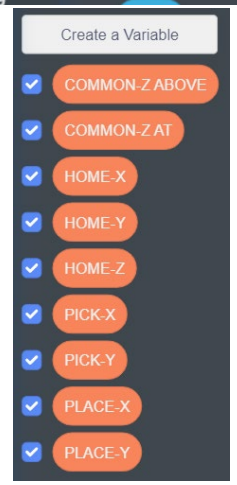
- **Used Multiple Times:** My Blocks can be called to run multiple times throughout a program.
- **Simplifying Programs:** Allows repeated chunks of code to be simplified into a single line of code throughout the main program.
- **Simplifies Editing:** Code that repeats itself multiple times throughout a program now only needs to be edited once.



The actual program after using MyBlocks!

We will start our program by defining a set of variables. Think of *Variables* as a group of lockers where information can be stored. The names of the lockers can be made by you. Each time the locker's name is used in the program, the name is replaced in the program by the value stored inside it. Variables can also be updated in a lot of different ways.





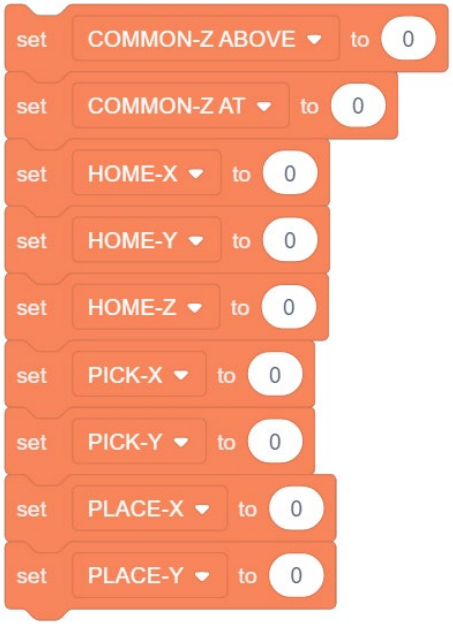
Start by creating the list of variables below

- COMMON-Z ABOVE
- COMMON-Z AT
- HOME-X
- HOME-Y
- HOME-Z
- PICK-X
- PICK-Y
- PLACE-X
- PLACE-Y

Next, from the *Variables* section, drag over a *Set Variable* block into the programming field.

Either duplicate this block or drag over a total of 9 Set Variable blocks.

Change the variable names by clicking the down arrow on the Set Variable block



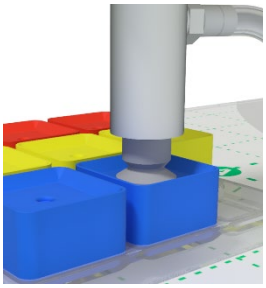
The next step is to start finding each of our starting values.

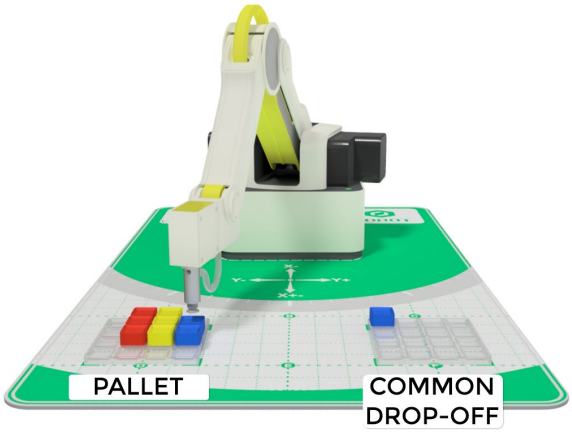
Using either the Jog Controls, or the Lock Button find the values for the table below.

It makes it easier to use the Lock Button to get close to the desired position and then use the Jog Controls for smaller movements.

DE-PALLETIZING SEQUENCE

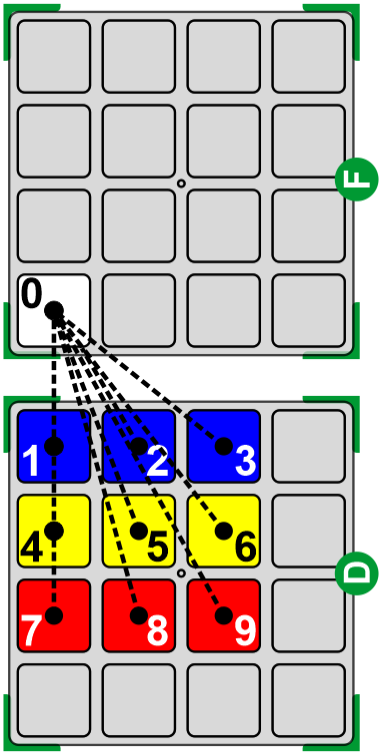
- *Round all values to the nearest whole number*
- *Align the center of the vacuum gripper to the center of the cube*



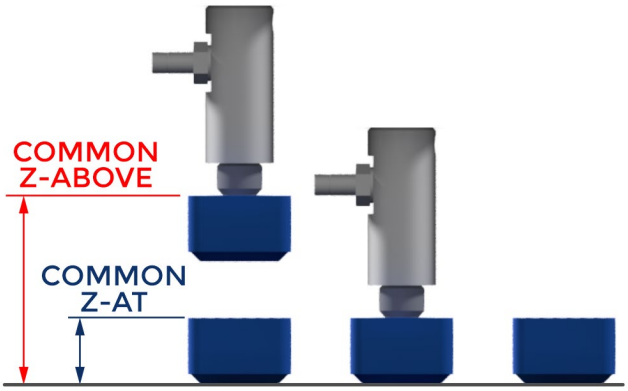


- *Write the values in the table below or in your notebook for future reference.*
- ***HOME-X, HOME-Y, & HOME-Z:*** *Find a comfortable Home position that can be used for the project.*
- ***PICK-X & PICK-Y:*** *Use the center of Cube '1', from the diagram on the right, as the X and Y location for the Pick Location.*
- ***PLACE-X & PLACE-Y:*** *Use the location of Cube '0' for the common Drop-Off point.*
- ***COMMON-Z AT:*** *Find a common Z value that can be used to PICK UP all of the blocks.*
- ***COMMON-Z ABOVE:*** *Find a common Z value that can be used to raise up and carry all of the cubes. Ensure this position is high enough to clear the other blocks*

COMMON-Z AT	
COMMON-Z ABOVE	
HOME-X	
HOME-Y	
HOME-Z	
PICK-X	
PICK-Y	
PLACE-X	
PLACE-Y	

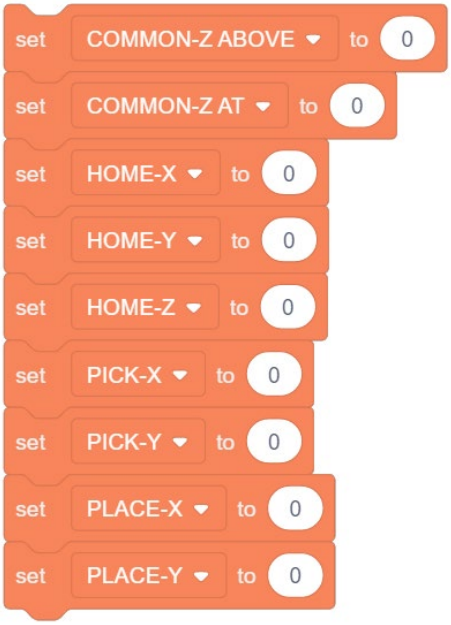


DE-PALLETIZING SEQUENCE



Part 1: Setting Variables

Start by recording the **COMMON-Z ABOVE** position in the number window of the block we created



HELPFUL TIPS

If you select Rename Variable, it will Rename that variable and everywhere it has been used in the program

The order in which we set the ***variables*** does not matter in this program.

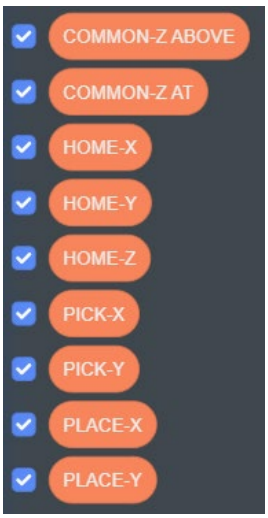
It is good practice to start each program by sending the robot to a HOME position, turning off or opening the EoAT (End of Arm Tooling) and turning off any DIGITAL OUTPUTS.

Drag over a GO TO (JOINT) and a SUCTION CUP (OFF)

HELPFUL TIPS

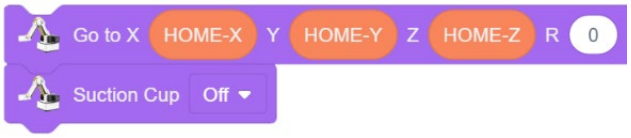
Remember that these tools come from the Motion section.

We will now use some of the user defined *variables* that we just created.



From the Variable section, you should see that all of the variables that we have created are now options to use.

Drag over your ***HOME Variables*** (*not the* ***set HOME Variables****)* and drag them into their corresponding positions for the ***GO TO*** step. Leave the Roll Value as "0".



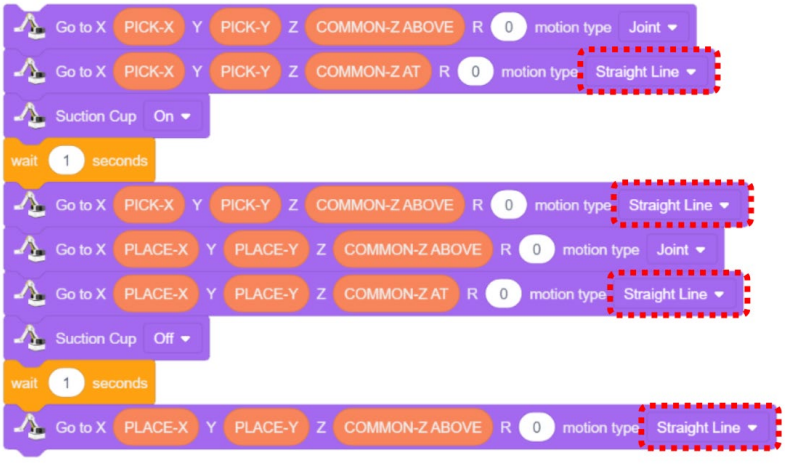
HELPFUL TIPS

Do not use the HOME Block. This block runs the homing operation.

HELPFUL TIPS

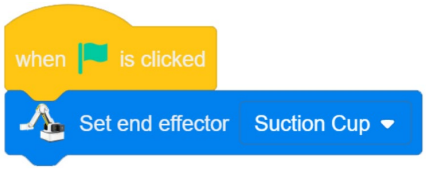
Assigning variables helps when coordinates in a program need to be changed multiple times or is going to be used as a template for multiple programs. They can be altered once at the beginning of the program and that alteration will update everywhere they were used throughout the program. It also makes reading the program easier as the user now sees words in place of number values.

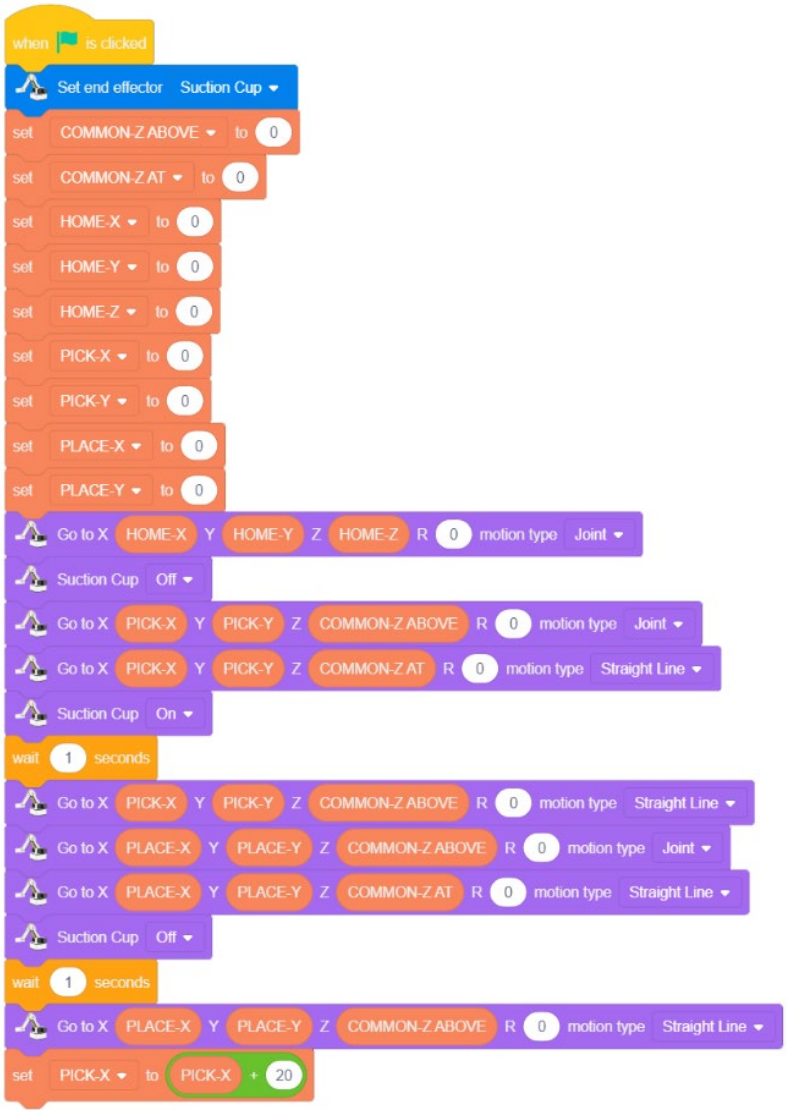
Next, add all the steps needed to go get the first block '0' from the first assigned pick location and move it to the place (BLUE) location.



Link everything, we now have as one program and add to the top of the program:

EVENTS: WHEN STARTED
&
SETTINGS: SET END EFFECTOR (EoAT)





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

Remember: if you hit a limit with the robot at any time, or you hear a clicking or a grinding noise, it's always a good idea to home the robot again. Also, if the robot does not return to the same position, just re-home it.

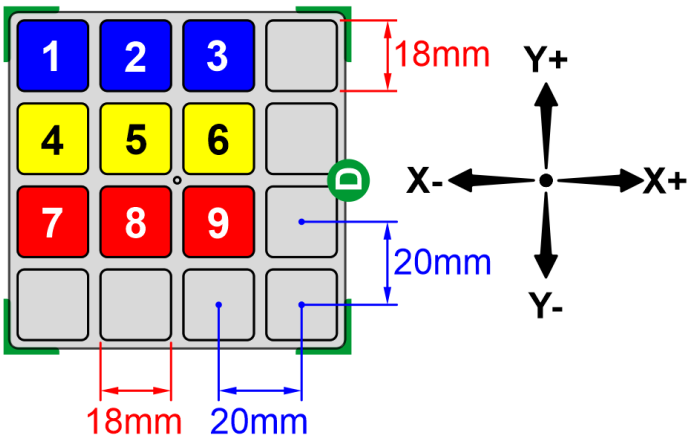
If your set up did not work correctly the first time, what did you have to do to make it work?

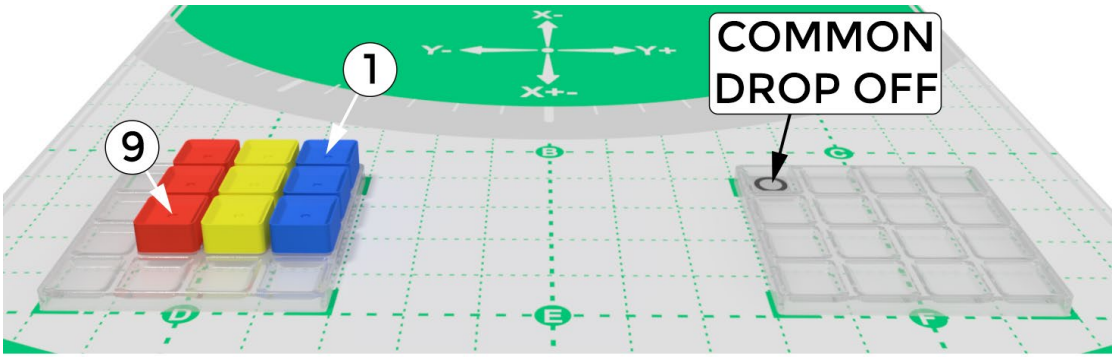
ALL NEW - THE TRICKY PART!!!

Now it's time for the second cube.

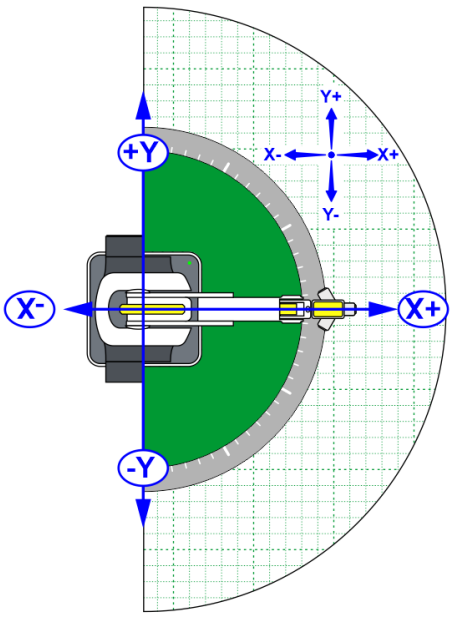
The only difference in the programming for this cube would be the X Position.

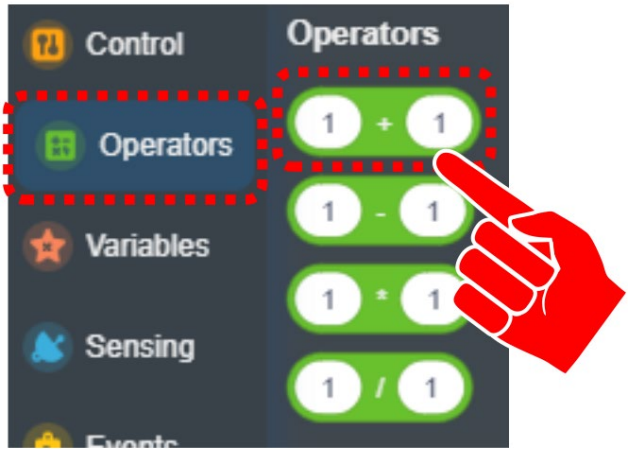
From the diagram shown, the blocks are 20mm apart (from center to center) and the block values are moving *away from* the robot. Moving away from the robot is shown mathematically by adding 20mm from the current X position value for block number 2 and again for block number 3.

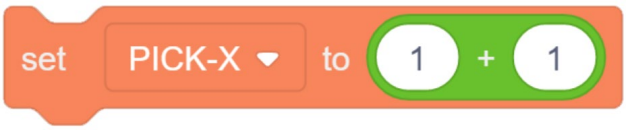




In order to adjust the X position, choose the ADDITION block from the OPERATORS section, and attach it to the ***SetVariable*** block.







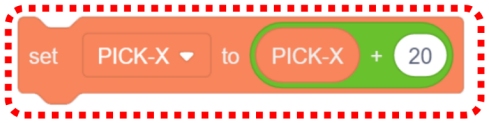
In order to add 20mm from the current PICK-X value and assign that value to the new PICK-X value, we will create a computation block that replaces the first value with PICK-X and the second value with 20.

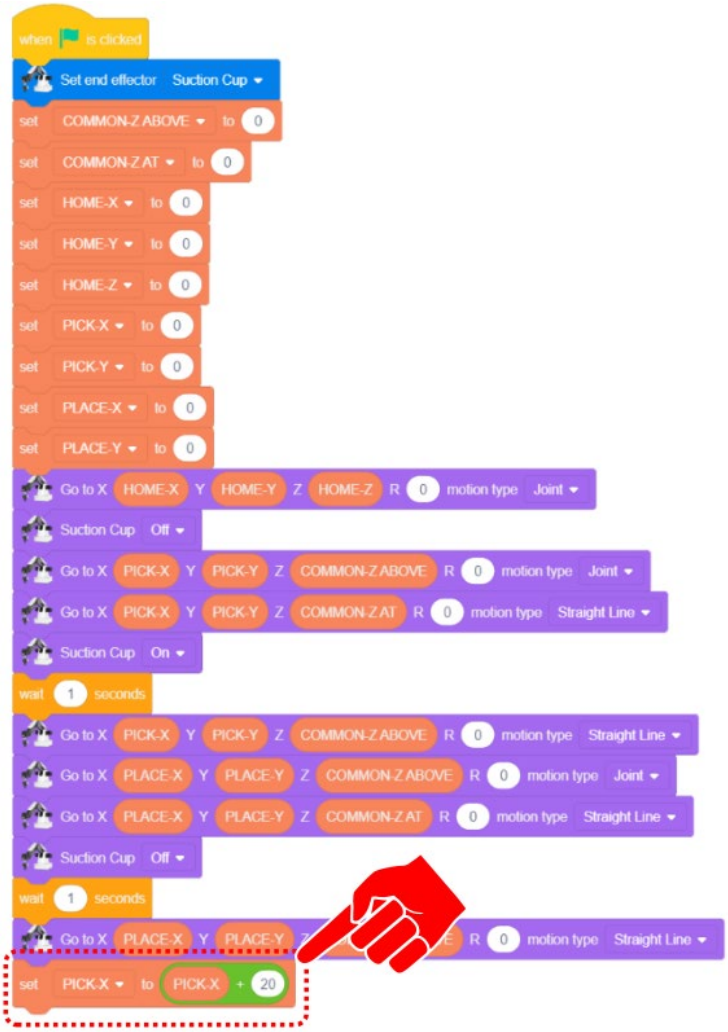
Current X value equals itself minus a constant of 20 (Positive due to the direction of motion)

Mathematically: PICK-X = PICK-X + 20

Grab the PICK-X Variable from the Variables section or duplicate it from the main program

Add this block line to the bottom of your program





At this point, the program is already starting to get long. The next step will be to divide the program up into separate thoughts called STATES.

STATE 1 – SETUP

The STATE 1 will assign the variables and send the robot to its start location (with the suction cup off).

STATE 2 – PALLETIZE STATE

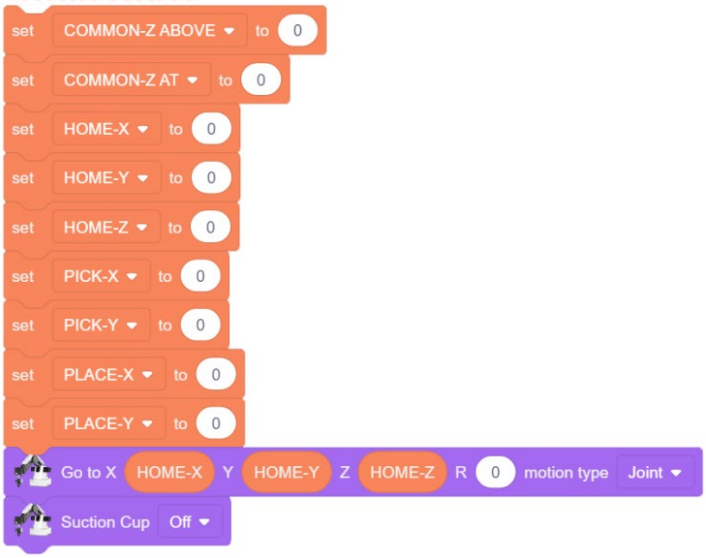
Now that the **PICK-X** value has been adjusted, the program needs to repeat the entire process of picking up the second and third cubes from row 1 and dropping them off at the place location.

This section will be STATE 2

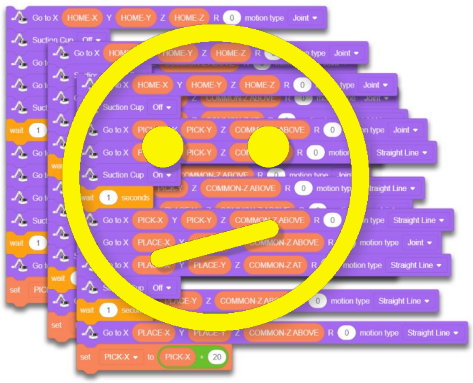
Since there are three rows, we would need three of these states (one for each block)

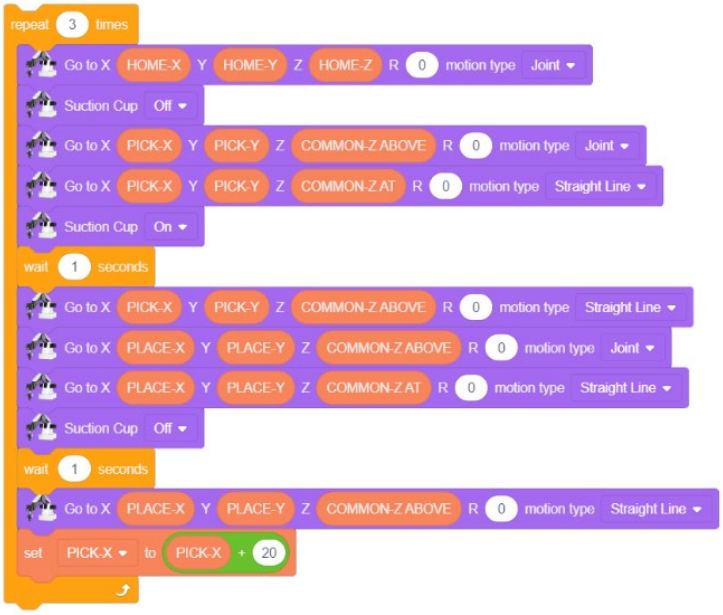
Instead of re-creating all of STATE 2 two more times, we place STATE 2 inside a REPEAT LOOP. Drag over a REPEAT LOOP from the Control section. Grab the 1st *Go To X* and drop it inside the REPEAT LOOP.

STATE 1
SETUP STATE
- Assign Variables
- Home Robot

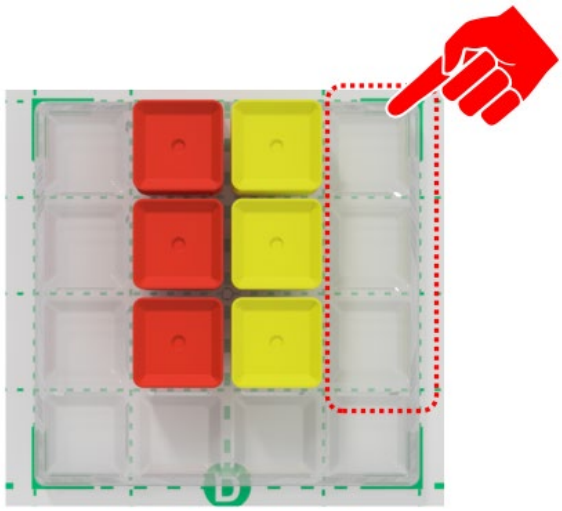


STATE 2
PALLETIZE STATE
- Pick Block
- Place Block
- Reassign Variables





The robot now has the ability to empty one full row (Block 1, 2, & 3) along the X.



Now is the fun part.

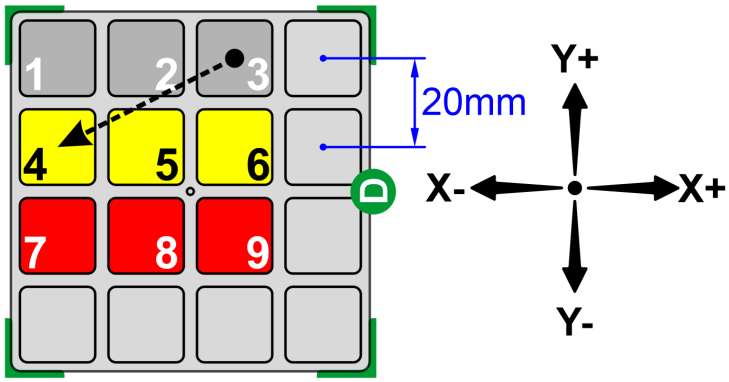
At the end of the 3 times loop, the program needs to SHIFT over to the next column (Y Axis) and reset the PICK-X to its initial value. This will be STATE 3

STATE 3

SHIFT STATE

- Shift Column Variable

- Reassign X Variable



Drag over 2 set Variable blocks

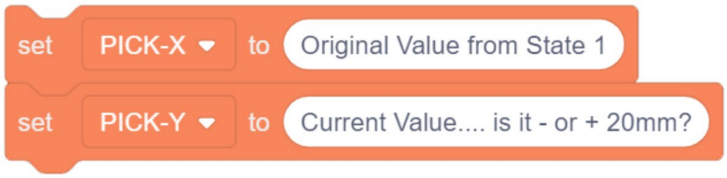
The 1st one will set PICK-X back to its original value. Get the value from State 1

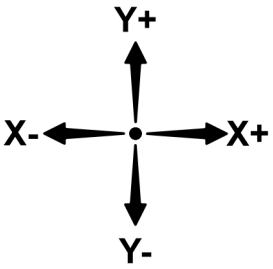
The 2nd one will need some math….

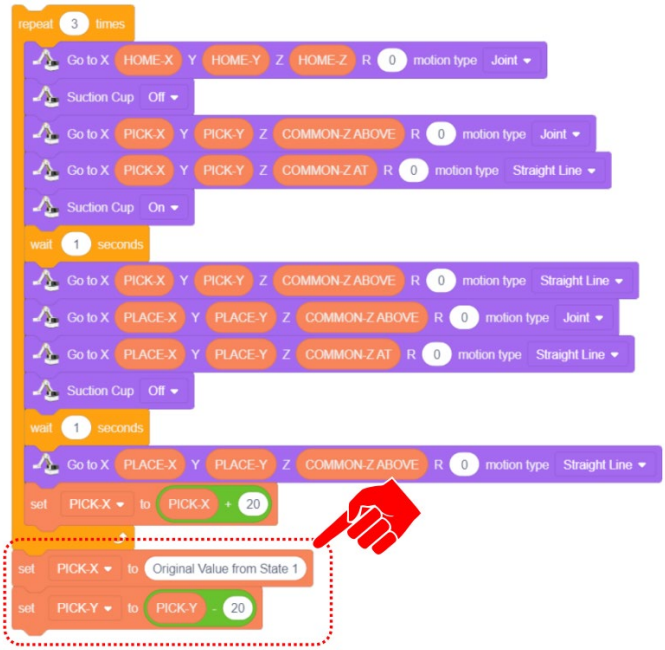
The column needs to shift in the negative direction on the Y-axis. Drag over a subtraction operator and place the PICK-Y variable inside it.

The program needs to subtract 20mm from the current value to shift to the correct location

Add these blocks to the bottom of STATE 2







EITHER STRATEGY WILL WORK!!

HELPFUL TIPS

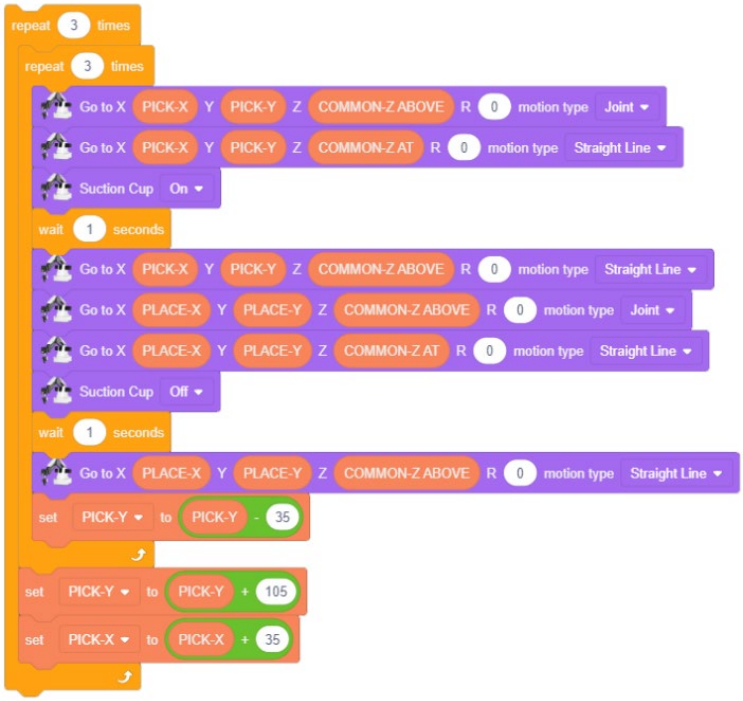
… There is also so cool math that could be done for resetting the X-Value that would not use a new constant value. The program could reverse the math… we took off added 20mm twice… so… (20+20=40)… PICK-X = PICK-X - 40?... Oh, Do not forget… the 20 was added at the end of each loop… so we accidently took off "x3" 20 increments… so (20+20+20=60) PICK-X - 60… This sets it back to what it was.

Once the variables are updated. The program needs to re run STATE 2 for the next two rows.

We could then duplicate the entire process two more times…

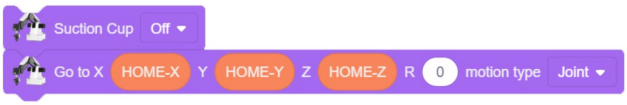
However, using this method, the program becomes very long, and we would have to do a lot of scrolling to move around in our program. Editing or updating the program can also be tedious and leaves a lot of room for errors.

There must be a better way to keep the program simpler, shorter, and easier to edit… if we notice that we are duplicating the exact same program three times… can we just REPEAT the REPEAT LOOP 3 times? Drag over a second REPEAT LOOP and drag all of the entire original REPEAT LOOP inside of it (a nested loop, or a loop inside of another loop).



STATE 4
END PROGRAM
- Home Robot

Duplicate and add one final SuctionCup Off and a final Go To Home to end the code.



Add this to the program, then try to run it.

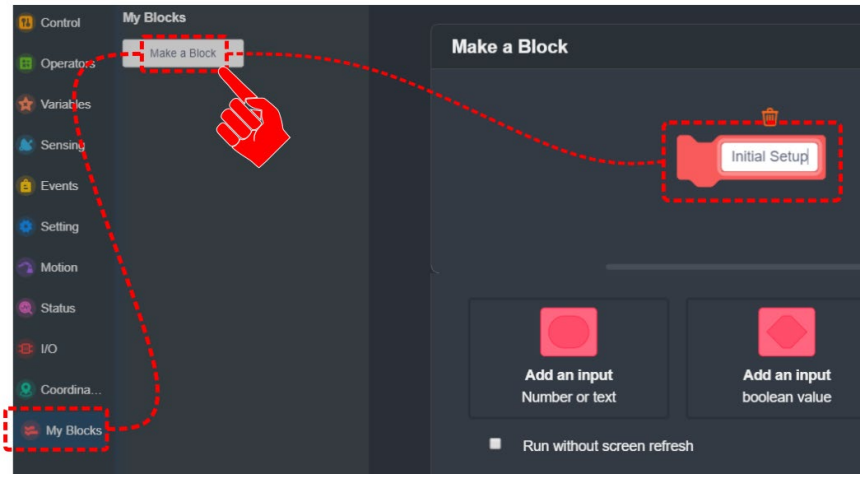
The program still seems very long even though variables and loops were used to make it shorter and simpler.

The next step is to clean up the program even further with a block of code called a function or void. Blockly programs refer to this code as creating MY BLOCKS. A *function* is a named section of a program that performs a task or a group of tasks. It can also be considered a procedure or a routine and can greatly simplify otherwise complicated programs. In other programming languages they might be called subroutines.

When a program calls MY BLOCKS to run, the program pauses, runs the MY BLOCK, and then returns to the program where it left off.

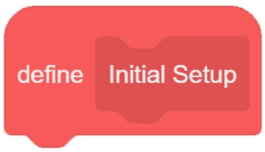
Click on “Make a Block” from the My Blocks toolbar.

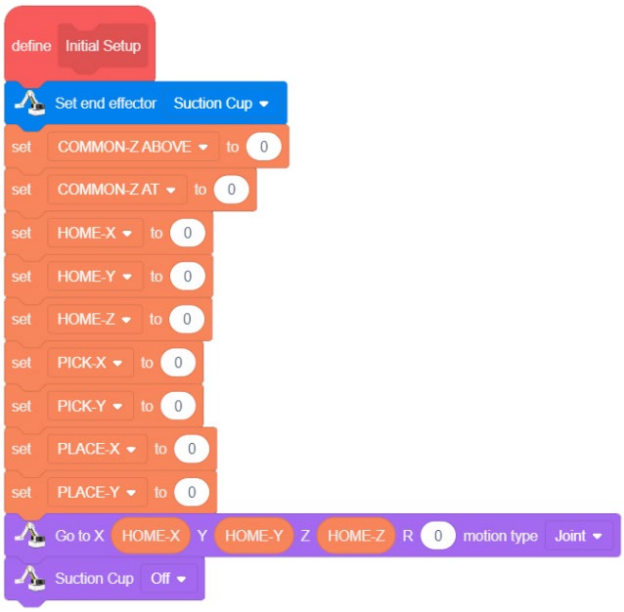
Name the first MY BLOCK Initial Setup



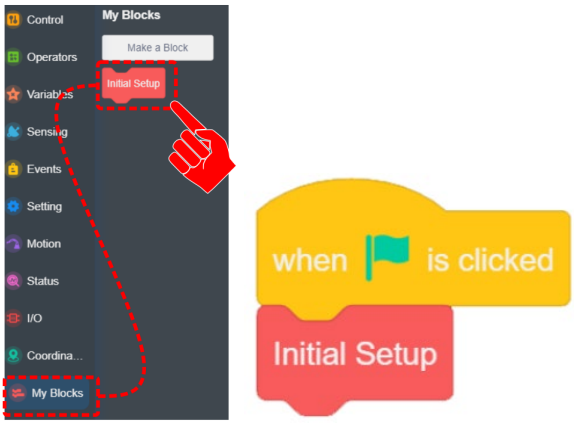
Once the new MY BLOCK is named, a new header will appear in the programing area.

Drag all of the setup code over and attach it to our new header.

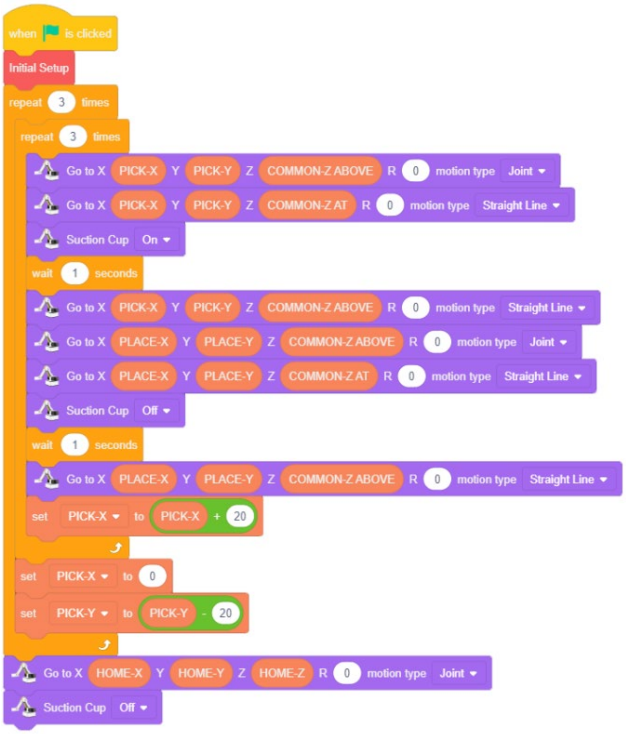




Now that the MY BLOCK is defined and the tasks are added to it, we need to add the ability for the main program to call the MY BLOCK to run. Drag the new block from the “My Blocks” tool bar over to the program area and attach it to the WHEN STARTED



Link the new header back on to the remaining portion of the program and run it… just to make sure everything is still running correctly.



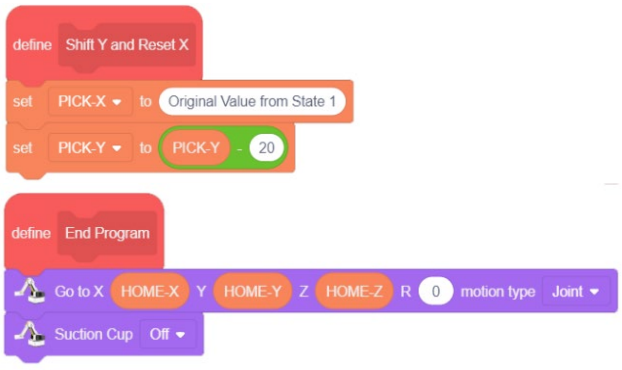
Can we simplify this even further? Are there any other chunks of code that we can make a My Block out of?

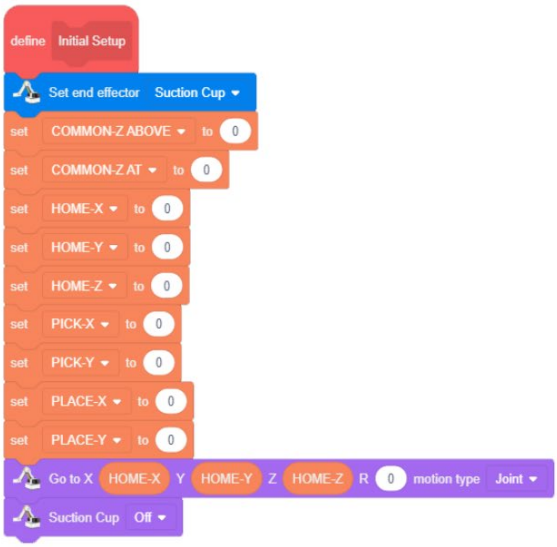
To clean-up the program even more, lets break it up into a few more groups and create additional my blocks

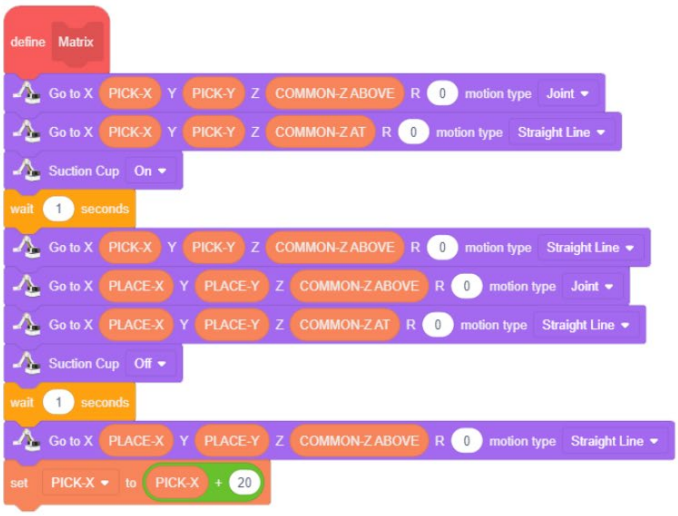
- Initial Setup – Any setup coding, Initial Variable values, sending the robot to our defined home and turning off the EoAT and digital outputs.
- Matrix – Matrix program for one full row
- Shift Y and Reset X – Move over to the next column and reset the X to the initial value
- End Program – ensure the EoAT is off and send the robot back to home

Drag each section of the program over to their new headers

MY BLOCKS just float in the programming environment unattached to the main program. They are essentially separate mini programs to be called by the main program.







Install the new MY BLOCKS into the program and test it.

Developing MY BLOCKS (Functions/Voids) not only produce groups of code that can be used multiple times but can also help a programmer organize their program into mini programs or separate thoughts.

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. *In your own words, define a variable.*

2. *In your own words, define a MY BLOCK (Function/VOID).*

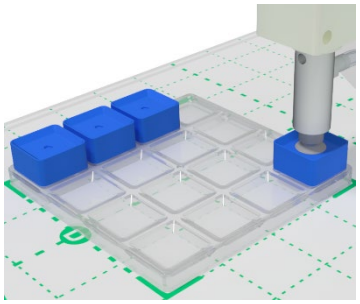
3. *Explain what would have to be done to palletize two layers using bullet points or a step-by-step list below.*

GOING BEYOND

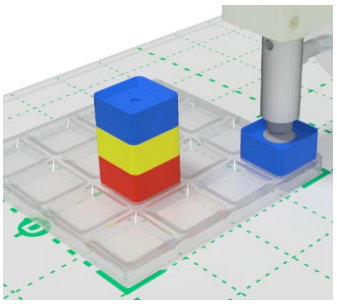
Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

_____ 1. Use a switch as an input to make the robot wait until you move the block from the Place position and hit the switch.

_____ 2. Move a row of three cubes to another row (Matrix to Matrix)



_____ 3. Develop a program that will run the entire process in reverse. Take a block from a common location and distribute them into a 3x3 matrix. This process is referred to as palletizing.



_____ 4. Take the cubes from a common location and stack them in a column vertically using variables for the height.

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING
MAGICIAN LITE

- CHAPTER 2 -
ROBOTIC COMMUNICATIONS

CHAPTER INTRODUCTION:

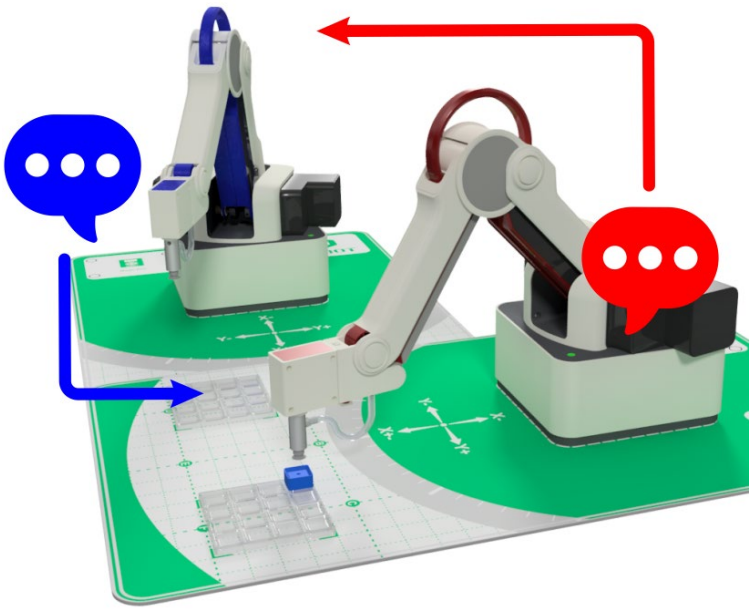
Robotic arms in industry need to communicate with other robots, as well as with other machines, within a work cell, or factory. This form of communication is called handshaking and can be done safely between different machines, devices, and robots. Handshaking is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

Example:

- Robot places a part in a machine to be processed.
- The robot tells the machine the part is ready and the machine can safely start by sending a signal.
- The machine returns a signal to the robot when that the part is done and the robot can retrieve the part and send it on its way to the next process.

TOPICS COVERED:

- Optical Isolators
- Handshaking with a Rely Module
  - Robot to Robot
  - Microcontroller to Microcontroller (VEX V5)
  - Robot to Microcontroller
- Internal Handshaking
- Wireless Bluetooth



2.1 Block - Robot to Robot Communication: Handshaking

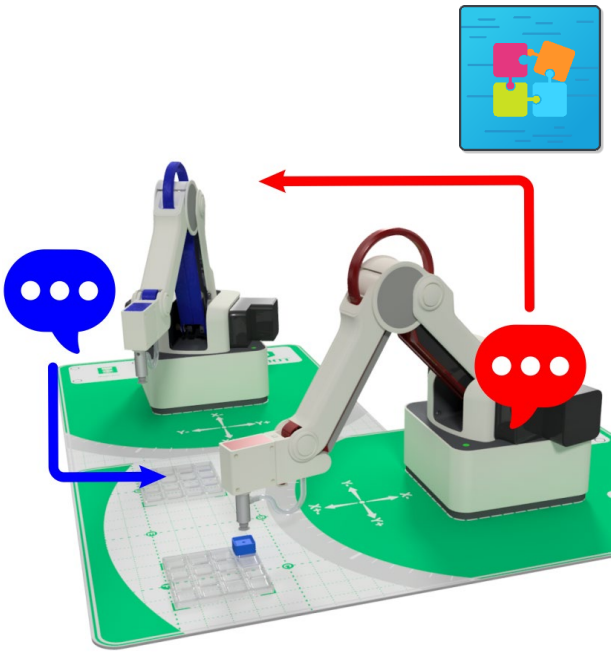
NAME: _____ *Date:* _____ *Section:* _____

INTRODUCTION

Robotic arms need to communicate with other robots in a work cell, or factory. This is called ***HANDSHAKING*** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or "ons" and "offs".

With DobotBlock Lab, we can safely have the robots communicate with no external wiring or other hardware!

Two or more robots will still be communicating with YES and NO signals called **messages**, this time it will be internally controlled by the computer and software. This too is an industry standard for equipment to be able to communicate at the lowest level (1s and 0s).



DobotBlock Lab software can create internal robot to robot communcation/handshaking.

KEY VOCABULARY

- Input
- Output
- Signal
- Network Handshaking

EQUIPMENT & SUPPLIES

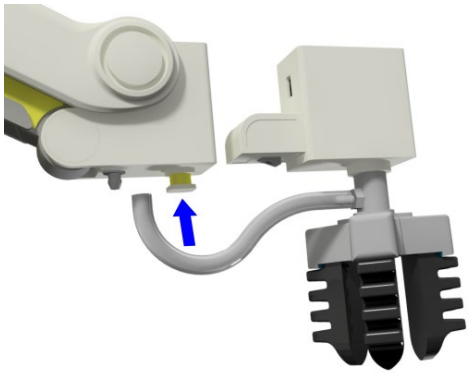
- 2 Magician Lite Robots
- 2 Magician Lite Work Mats
- Small cubes & pallets
- DobotLab software
- Or printed templates from the resource section

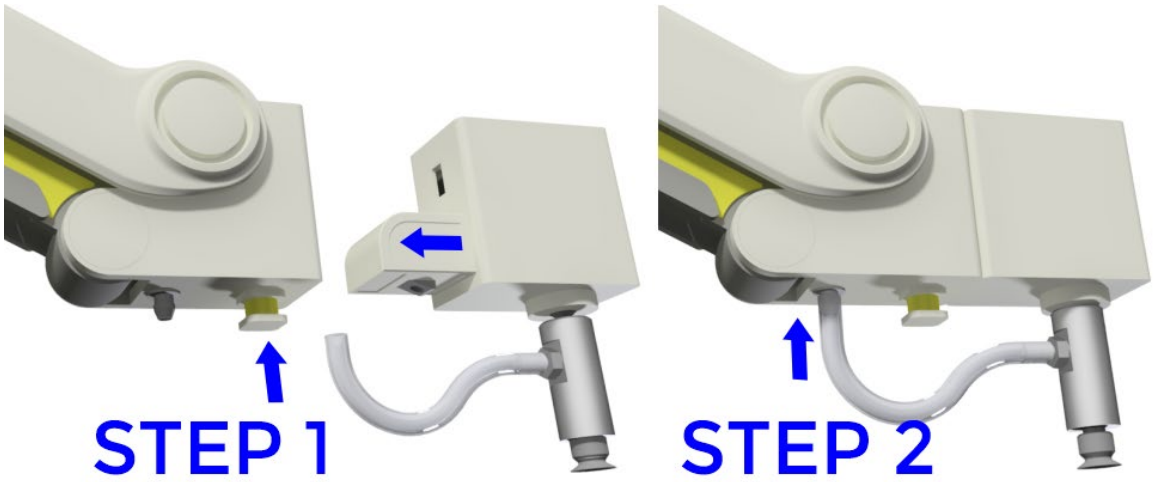
SECTION 1 – SETUP BOTH ROBOTS

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. **Typical Start Up Procedure**

- Disconnect any existing **END of ARM TOOLING (EoAT).**
- Carefully disconnect any existing vacuum tubes.
- Press and hold the release button on the bottom of the arm and pull off the EoAT.



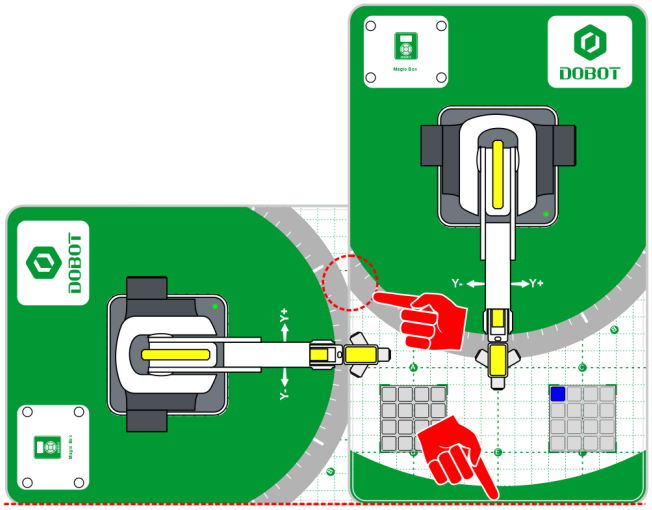


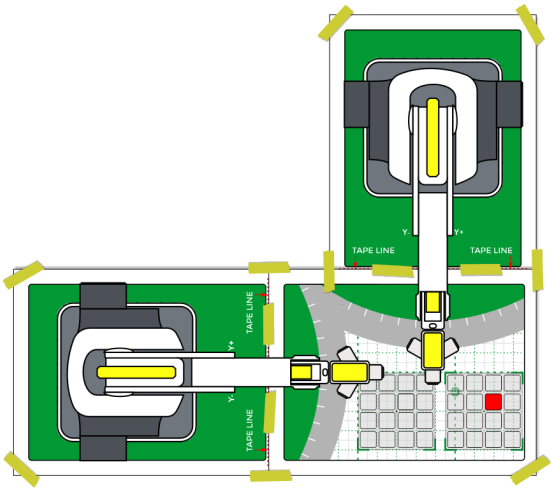
2. Attach the *Suction Cup* as the **END EFFECTOR** or **END of ARM TOOLING (EoAT)** on both robots.

- STEP 1 - Push the *Suction Cup* in until it snaps in place.
- STEP 2 - Attach the hose to the air nozzle.

3. Set up the robots on the Magician Lite Work Mats as shown below.

Included Robot Mats	Printed Field Templates (from resource section)
Line the edges up along the red dotted line. Overlap the mats where the grey arcs intersect	Line up the work area page with the other two pages. Tape them together. Tape the to the table if needed

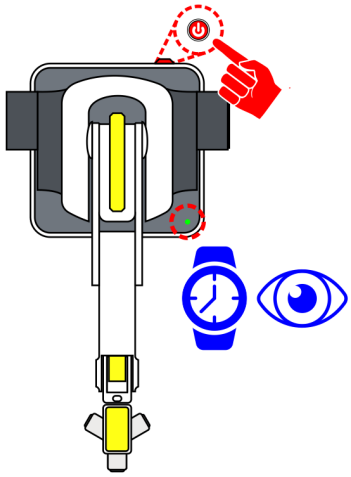




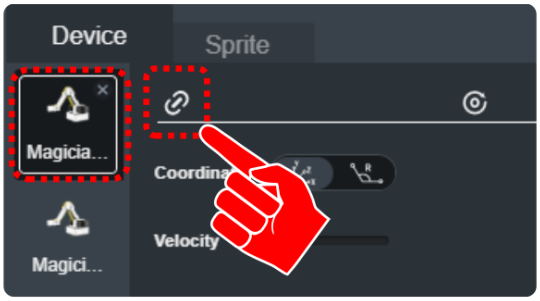
SECTION 2 – CONNECTING ROBOTS to the COMPUTER & DOBOTLAB

1. Connect both robots to the computer using their included USB cables
2. Open up DobotBlock Lab in the software.

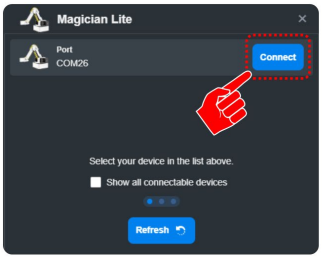
3. Once **ALL** of the wiring is done (Power cables, and robot arm connected), power ON both robots. **WATCH** and **WAIT** for the robots to completely power on. (GREEN INDICATOR)

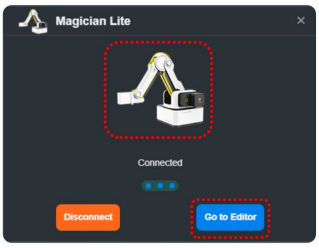


4. Select the connect icon to establish control from the software to ROBOT 1. A connection window should pop up.



5. Select Connect from the pop-up window. A new window should appear and show that the robot is Connected.





6. HOME ROBOT 1 to ensure communication.

HELPFUL TIPS

Two robots can be plugged into the same computer at the same time!

7. Select choose a device and add a second Magician Lite.

   Connect to this device and home it just as we have done with previous activities.

   As you click back and forth between the magician devices (icons), you will see a different programming page, one for each. When you save the program, you will save both programs at the same time.

SKILL BUILDER 1 – BROADCAST - INTERNAL HANDSHAKING BETWEEN ROBOTS

For this skill builder, test the ability to send a signal from one robot (ROBOT1) to another robot (ROBOT2) as long as they are plugged into the same computer and running off the same file

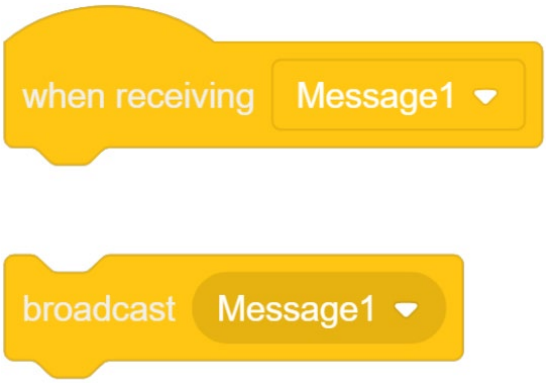
The process of sending a signal from one robot to another robot can be done with messages.

1. Drag over WHEN RECEIVING and BROADCAST black from the EVENTS toolbox.

   The WHEN RECEIVING works the same as the WHEN STARTED, but instead of selecting the START FLAG to be clicked, the program waits for a message to be received.

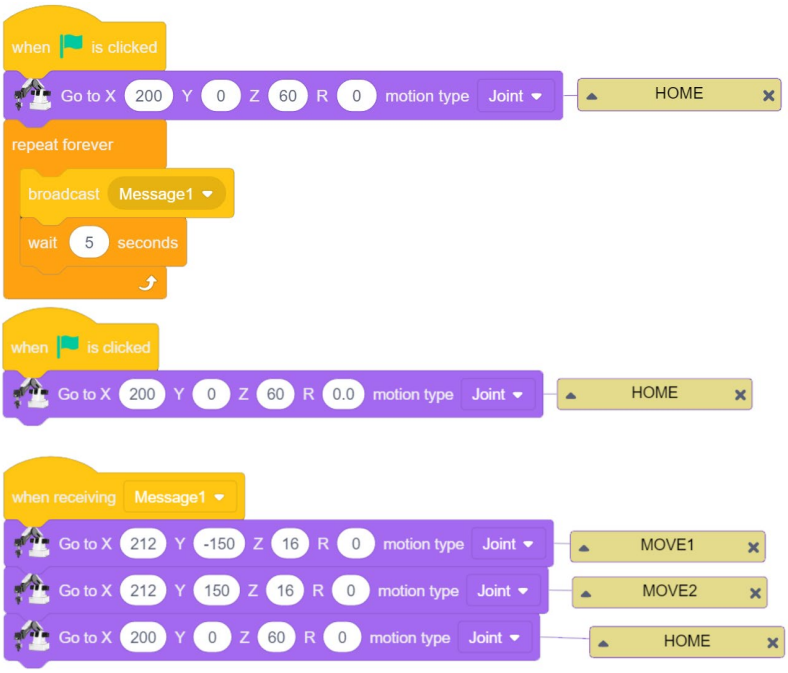
   

   The BROACAST block is the tool that initiates the message. Both blocks also have the option to create new messages in their drop downs.

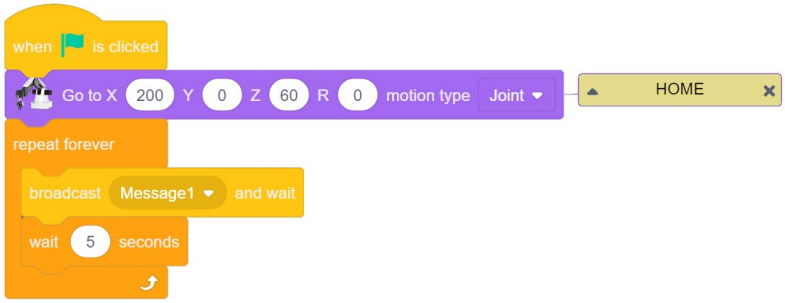


1. Create a program similar to the one on the right for **ROBOT 1**
   a) WHEN STARTED – send **ROBOT1** HOME
   b) BROACAST MESSAGE
   c) WAIT 5 seconds and repeat
2. Create a program similar to the one on the right for **ROBOT 2**

   a) WHEN STARTED – send **ROBOT2** HOME
   b) WAIT for a message
   c) Once RECEIVED, move **ROBOT2** (move the robot to one or two new positions and then back to home. **ROBOT2** needs to be back at the home position in less than the 5 seconds to receive the next message)

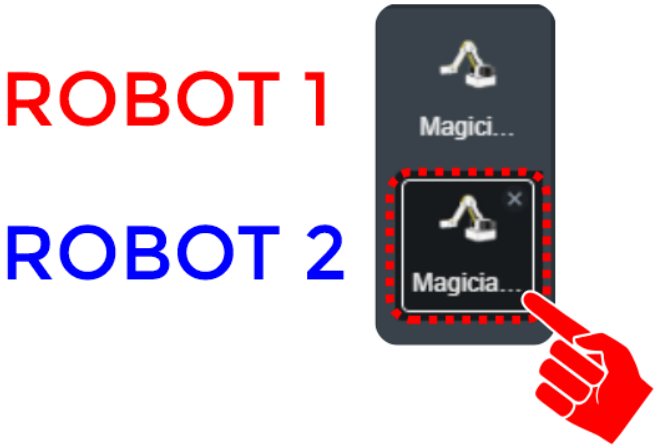


The program created is an OPEN LOOP system. **ROBOT1** does not know **ROBOT2** has received the signal.

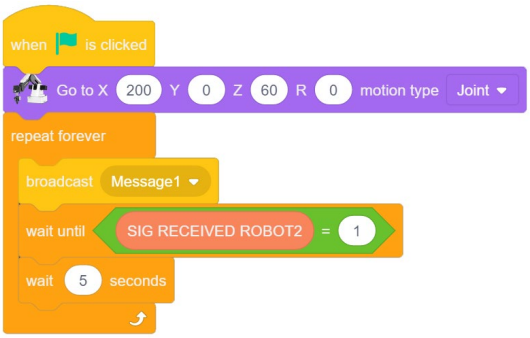


3. Change the BROADCAST MESSASE to BROADCAST and WAIT. **ROBOT1** will now send the signal and wait until the broadcast is received by **ROBOT2**

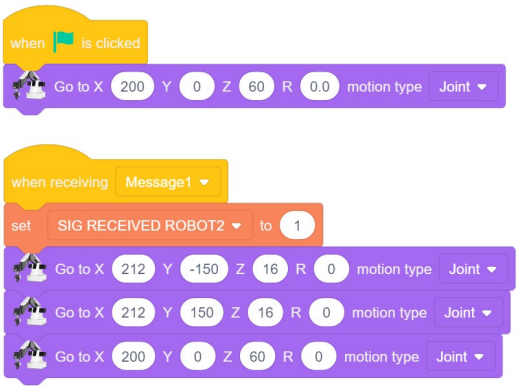
Recreate the CLOSED LOOP process using VARIABLES



ROBOT 1



ROBOT 2



Create a VARIABLE called **SIG RECEIVED ROBOT2** and alter the two programs.

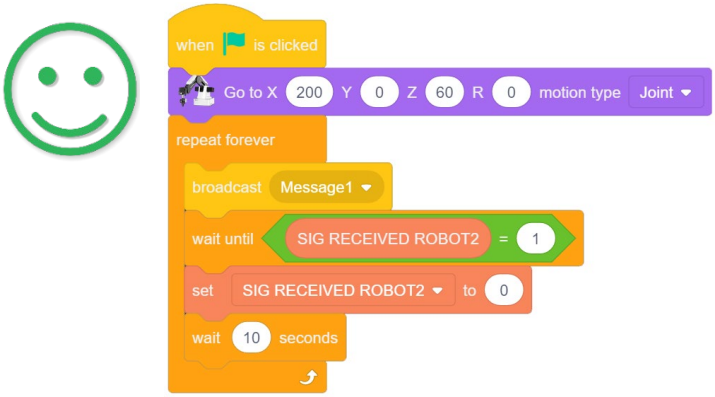
Run the program.

Is the program really working correctly? It may look like it is working, but once the variable is set to TRUE… it never changes. The program is now skipping the WAIT UNTIL.

4. Alter **ROBOT1**'s program to change the variable back to 0 once it is received.

5. Change the time to 10 seconds to make sure it is working correctly.

   Run the program.



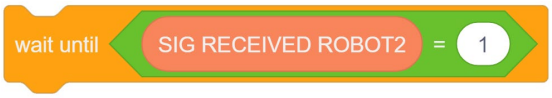
These tools are great for one way communication, but the trick is how to wait for a message in the middle of a program rather than just using it to start the ENTIRE PROCESS… Is there a better way to do this?

…Notice that there is only one list of variables for the SPRITE

…NOTICE that variables are not only GLOBAL for one robot's program but for all of them!! (each robot can SET AND USE the same variables)

If the program can use WHEN MESSAGE RECEIVED blocks to control variables… we can then handshake in the middle of a program.
The program can use the variable in the middle of the program to tell the robot when to proceed. *(each time the message is received, set a variable to tell the program the message has been received)*

The WAIT UNTIL with the SIG RECEIVED block is a great way to make this happen.



If this is the case… can the program JUST USE VARIABLES instead of broadcasting messages?

YES!!!! We can skip using MESSAGES for this program. BROADCAST MESSAGE has its place in programing, especially when it can be used to replace a WHEN STARTED for a second program.

SKILL BUILDER 2 – VARIABLES ONLY - INTERNAL HANDSHAKING BETWEEN ROBOTS

For the remainder of this activity, the BROADCAST MESSAGES will NOT be used.

6. Create an additional variable
    a) SEND SIG **ROBOT1**
    b) SIG RECEIVED **ROBOT2**

ROBOT1 will use SET variable SEND SIG to TRUE(1) to start **ROBOT2**.

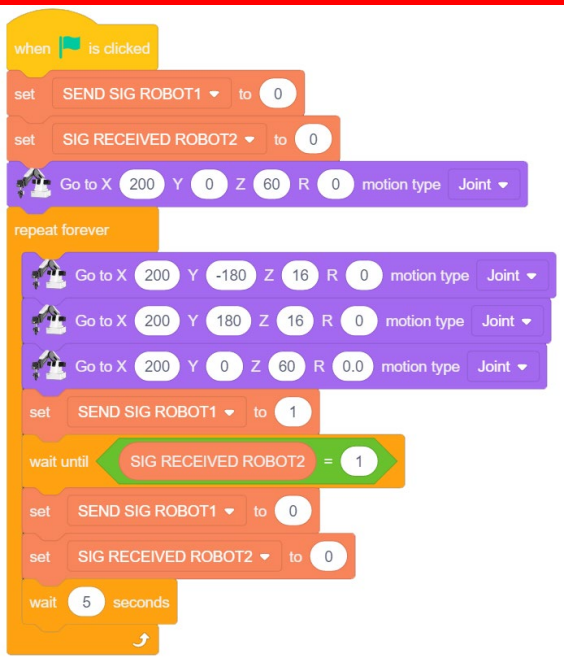
ROBOT2 will use SET variable SIG RECEIVED to TRUE (1) to let **ROBOT1** know the signal has been processed.

The variables will be SET to FALSE(0) to reset them for the next loop

For this activity, integers will be used rather than strings (words), however, the blockly program can use both.

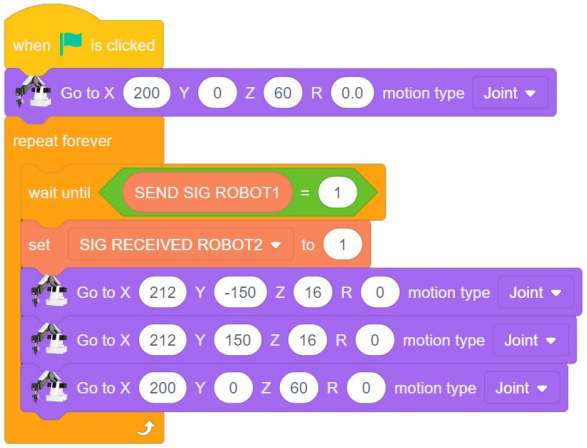
OPERATION – ROBOT 1 (CONTROL ROBOT)

a. Initialize variables to ZERO (only one robot needs to do this, they share the same variables.
b. GO TO HOME
   START THE FOREVER LOOP
c. Move the robot to location “A” any location
d. Move the robot to location “B” any location
e. GO TO HOME
f. SEND the signal to start **ROBOT2**
g. WAIT until **ROBOT2** has received the signal
h. Turn off the SEND signal and the RECEIVE signal
i. Wait 5 seconds
j. Loop the process



OPERATION – ROBOT 2

a. GO TO HOME
   START THE FOREVER LOOP
b. SEND a signal to **ROBOT1** that the sequence has been started **(ROBOT1)** will reset the variables)
c. Move the robot to location “A” any location
d. Move the robot to location “B” any location
e. GO TO HOME
f. Loop the Process



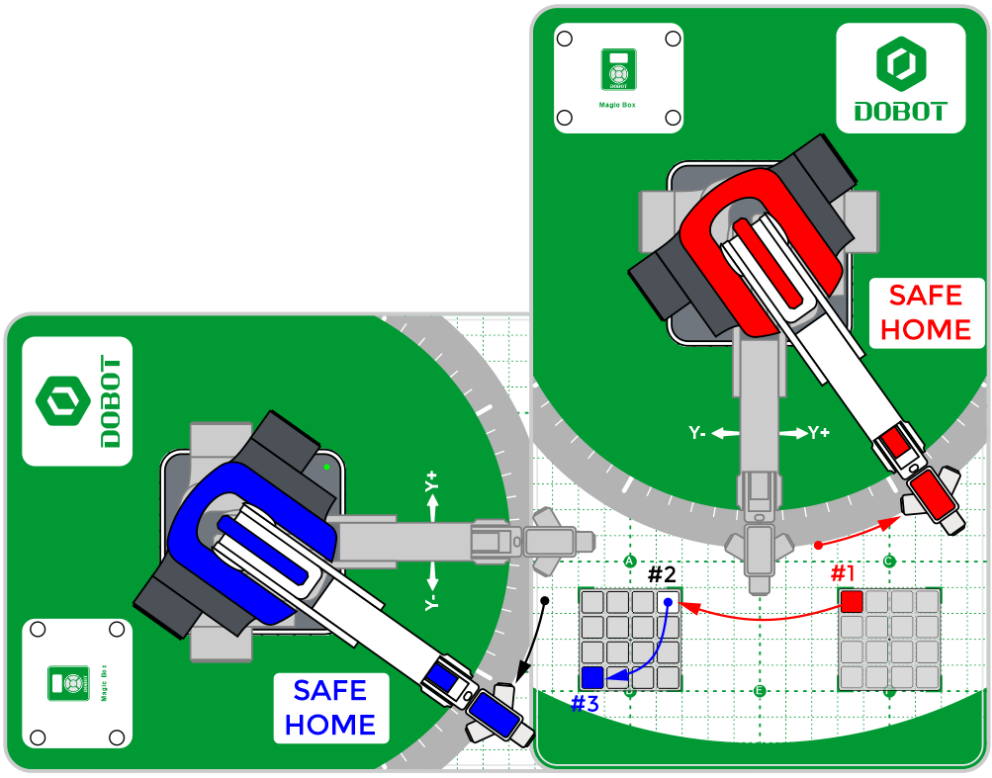
Both programs will start with the same WHEN CLICKED

CHALLENGE

SECTION 3 – PASS the BLOCK

HOME both robots before placing them on the templates. NOTICE there is an overlap in their work envelopes. The robots can run into each other during the homing operation.

Order of operations



ROBOT 1 -

- Move – SAFE HOME
- Move – Cube from Position **#1** to **#2**
- Move – SAFE HOME
- Send START SIGNAL to **ROBOT2**
- WAIT for START SIGNAL from **ROBOT2**
- Turn OFF Signal from **ROBOT2**
- **LOOP

ROBOT 2 -

- Move – SAFE HOME
- WAIT for START SIGNAL from **ROBOT1**
- Turn OFF Signal from **ROBOT1**
- Move Cube from Position **#2** to **#3**
- Move – SAFE HOME
- Send START SIGNAL to **ROBOT1**
- **LOOP

Give each robot a **SAFE HOME** position. A position that is out of the way of the other robot, so that they never run into one another.

Fun Fact: If they did run into each other, the robot are intelligent enough to know they have ran into something along their path and will stop immediately so that there is minimal damage to the equipment, and everyone working around them is a lot safer!

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

CONCLUSION

1. *Explain why you would ever want to make two robots communicate with each other. What are the benefits? Use at least three bullet points.*

2. *What is the advantage to using variables in this program? Use at least three bullet points.*
   -
   -
   -

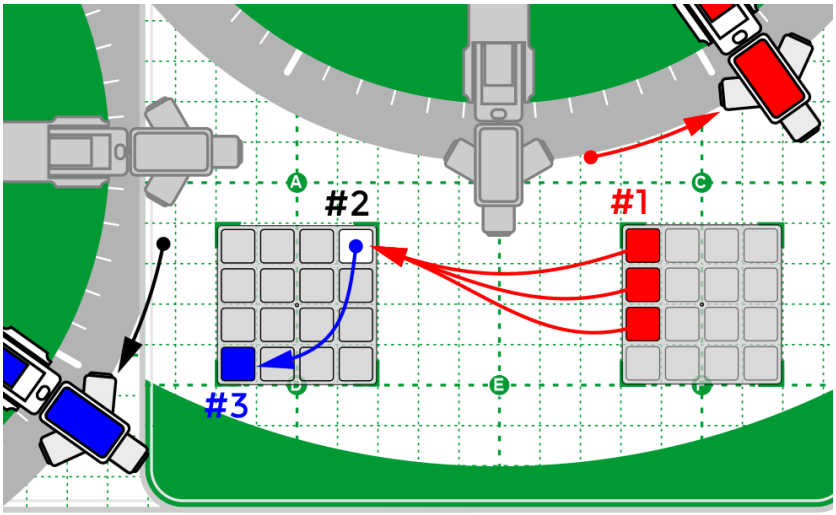
3. *Look up the word "Cobot" or "Collaborative Robot" on the internet and find a definition that you understand. Is the Magician Lite, the robot that you are using in this activity a Cobot? Explain why it is in your own words.*

GOING BEYOND

_____ 1. Use two robots where:

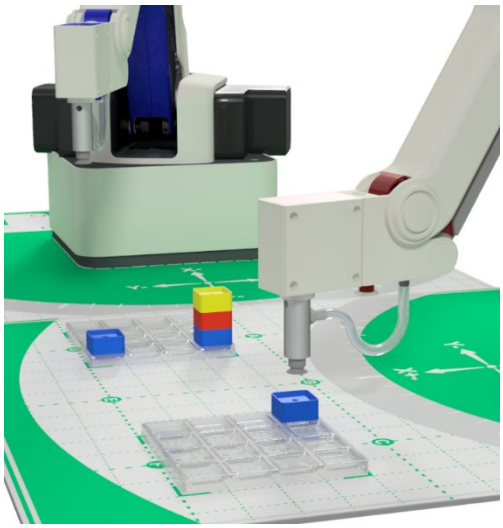
a. Robot 1 picks up the first block on Pallet 1 in a row of three.
b. Robot 1 places it on the corner of pallet 2
c. Robot 2 then moves it to another part of pallet 2

Use what you learned in previous activities and make robot 1 move to the other to spots in the row automatically, and make robot 2 stack the three Blocks on pallet 2.



_____ 2. Use the two robots to perform a stacking operation of multiple blocks.

a. Robot 1 picks and places a block from the first pallet to the second.
b. Robot 1 goes home and sends a signal to robot 2.
c. Robot 2 waits for the signal, then picks up a different block on the pallet.
d. Robot 2 stacks the block on the other one.
e. Robot 2 goes home.
f. Repeat this multiple times.
g. How many can you get them to stack successful



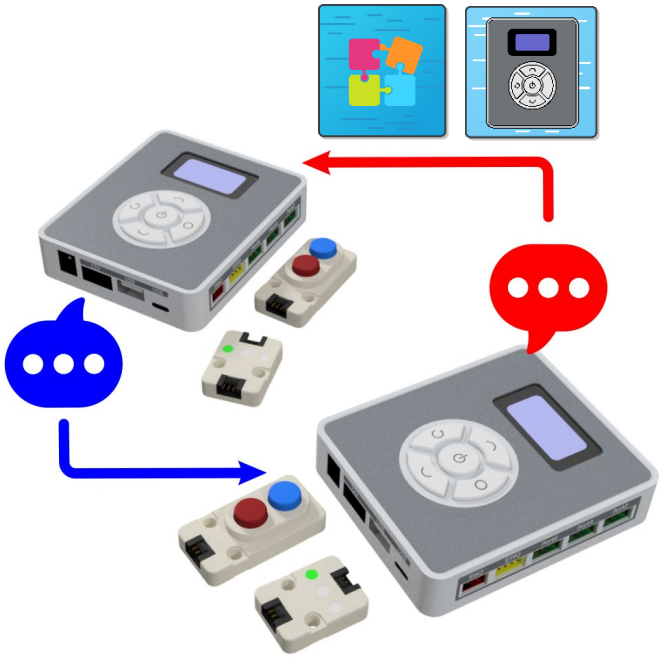
2.2 Block - Magic Box to Magic Box Bluetooth Handshaking

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

INTRODUCTION

These activities have shown you that you can **handshake** without hardware and do it through software with **variables** and **broadcasting**. There is still yet one more way to make two Dobot's communicate with one another. This is through **Bluetooth**. Bluetooth is a short-range wireless communication protocol that will allow two devices, such as robots to communicate wirelessly across a room.

In this activity, signals will be sent wirelessly between two **magic boxes**. The **2-Button Input** module will be the input and the **LED module** will be the output.



KEY VOCABULARY

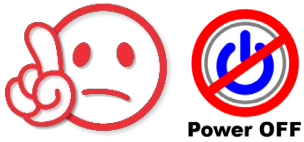
- Input
- Output
- Broadcast
- Magic Box
- LED Module
- Signal
- Handshake
- Variables
- Bluetooth
- 2-Button input

EQUIPMENT & SUPPLIES

- 2 Dobot Magic Boxes
- Digital - 2-Button Input Module
- LED Module
- 2 Separate Computers
- DobotLab software
- 2x 12V Power Cable and Supply
- 4x Common Sensor Cables
- 2x USB C to A Cable

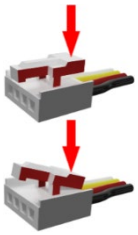
PROCEDURE

SECTION 1 – SETUP BOTH ROBOTS



Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS FROM THE MAGIC BOX!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

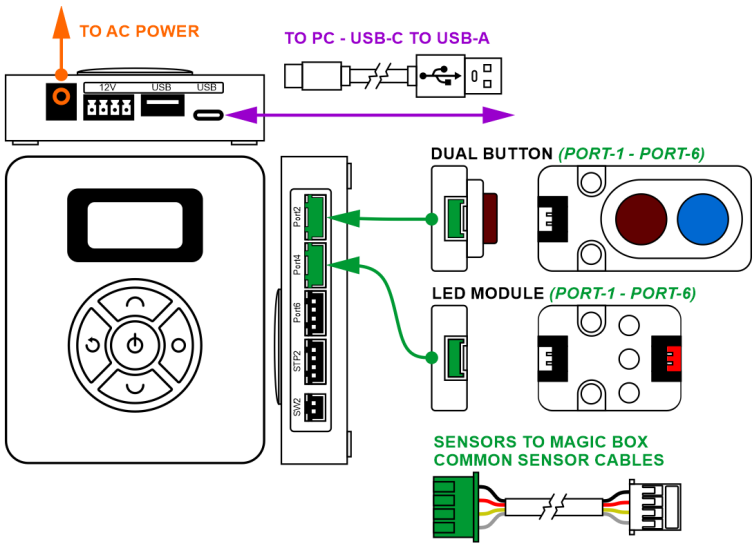


SECTION 2 – WIRING the MAGIC BOX AND SENSORS – 1st SETUP

SETUP #1

1. For this activity, you will need:
   - 2x Magic Box
   - 2x USB-A to USB-C Cable
   - 2x AC/DC Power Adapter (12V)
   - 4x Common Sensor Cables
   - 1x Dual Button – ***PORT 2***
   - 1x LED Module – ***PORT 4***

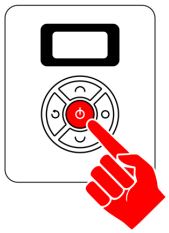
Wire the Magic Box as shown below



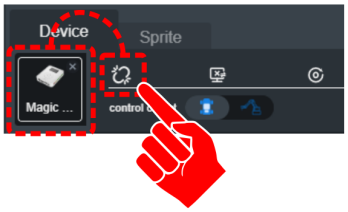
SECTION 3 – CONNECTING MAGIC BOXES to the COMPUTER & DOBOTLAB

1. Connect both Magic Boxes to separate computers using their included USB cables
2. Open up DobotBlock Lab in the downloaded software on your computer.

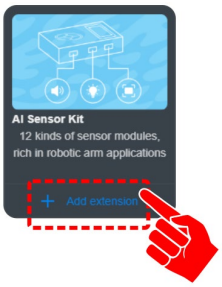
3. Once all of the wiring is done (sensors connected), power ON the Magic Box.



4. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).



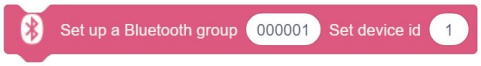
5. Click on "+ Add extension" for the AI SENSOR KIT.



SKILL BUILDER 1 – SEND A SIGNAL THROUGH BLUETOOTH

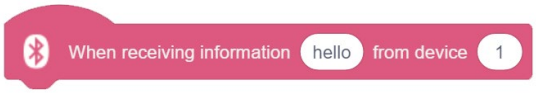
The setup for sensing a signal between Magic Boxes is done with a few tools.

SET UP BLUETOOTH GROUP is the block that allows the program to send signals on a unique Bluetooth network. Each device must run this block at the beginning of the program.

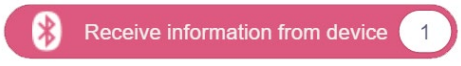


A unique group (Street Name) should be chosen for each group of Magic Boxes that are to communicate (6 digit number. (000001 – 999999)

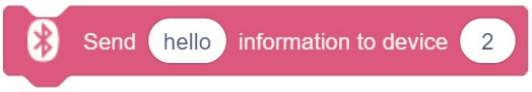
Each device is then given a specific device ID (House Number). The number can be any number from 0-255.



WHEN RECEIVING INFORMATION – Work similar to BROADCASTING MESSAGES. This block can replace the WHEN STARTED FLAG event header to start an operation each time a signal is received.



RECEIVE INFO FROM DEVICE – Can be set to a variable. Both numerical and string values can be received.



SEND INFO TO DEVICE – Block used to send numeric and string values to a specific device in the group.

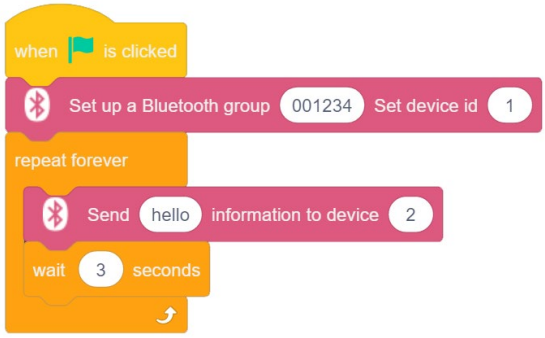
The Bluetooth group # must be 6 digits long. For example: if you want to use the Bluetooth group # 1234, as in this activity, you must type in the leading two 0's: 001234!

DEVICE 1

1. Drag over a SETUP BLUETOOTH and a SEND BLUETOOTH from the WIRELESS category in the block toolbox.

2. Create the program to the right on one computer.

   a) Set up a unique group number for both devices.
   b) This device will be DEVICE ID 1
   c) Send the STRING HELLO to DEVICE 2 in a loop every three seconds.



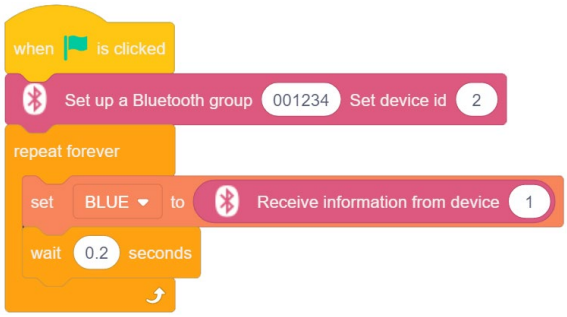
DEVICE 2

3. On the second computer, create a new variable (BLUE)
4. Drag over a SETUP BLUETOOTH and a RECEIVE INFO from the WIRELESS category in the block toolbox.

Does it matter what you name the variable? Is there a better name? Try it!

5. Create the program to the right on one computer.

   a) Use the same group number form the 1st device.
   b)
   c) This device will be DEVICE ID 2.
   d) Set the VARIABLE to the information RECEIVED from device 1.



Run both programs. The STRING hello should briefly appear and disappear in the BLUE variables box.

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

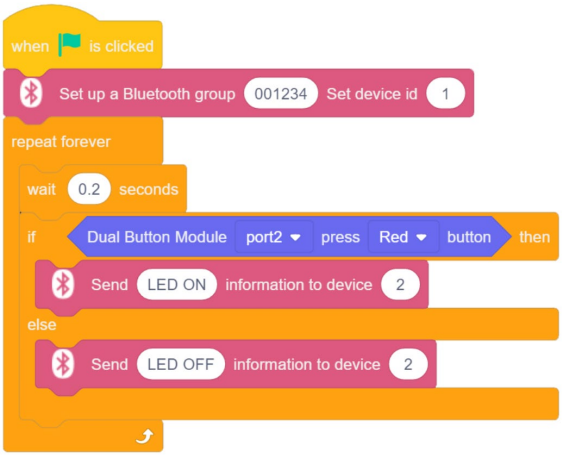
SKILL BUILDER 2 – CONTROL A DEVICE THROUGH BLUETOOTH

Now that a signal can be both sent and received through Bluetooth, use this signal to control one device from another. The following program will use a button press (DUAL BUTTON) from Magic Box 1 to turn ON and OFF the LED MODULE from Magic Box 2.

DEVICE 1

1. Drag over a SETUP BLUETOOTH and a SEND BLUETOOTH from the WIRELESS category in the block toolbox.

2. Create the program to the right on one computer.
   a) Keep the same GROUP and ID numbers
   b) Drag over an IF/ELSE Statements from the Control toolbox
   c) IF BUTTON RED is pressed, Send the STRING LED ON else LED OFF

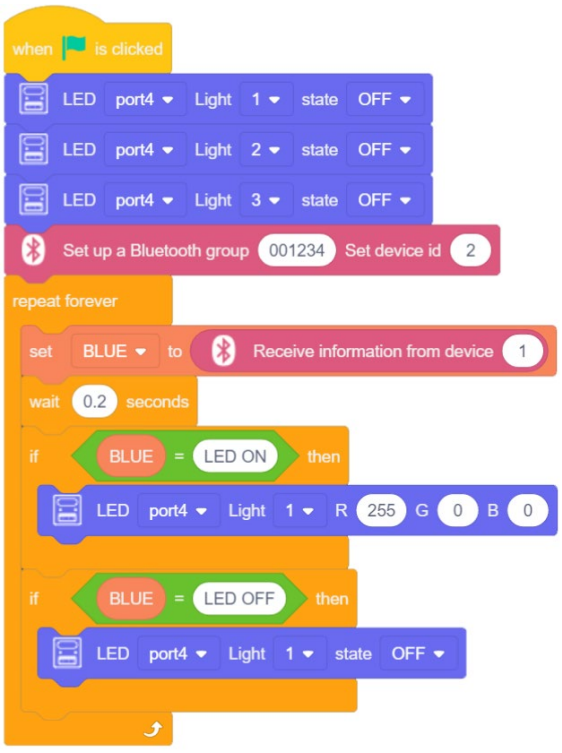


.

DEVICE 2

3. Drag over a SETUP BLUETOOTH and a RECEIVE BLUETOOTH from the WIRELESS category in the block toolbox.

4. Create the program to the right on computer 2.
   a) Start the program by turning off all of the LEDs
   b) Keep the same GROUP and ID numbers
   c) Set the variable BLUE to the info received from device 1
   d) Drag over two IF Statements from the Control toolbox
   e) IF variable BLUE = LED ON, turn ON LED 1 (RED)
   f) IF variable BLUE = LED OFF, turn OFF LED 1



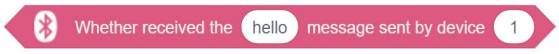
Run both programs.

NOTE: There will be a small delay in the LED turning on and OFF. Be patient, the delay may be 1 to 2 seconds.

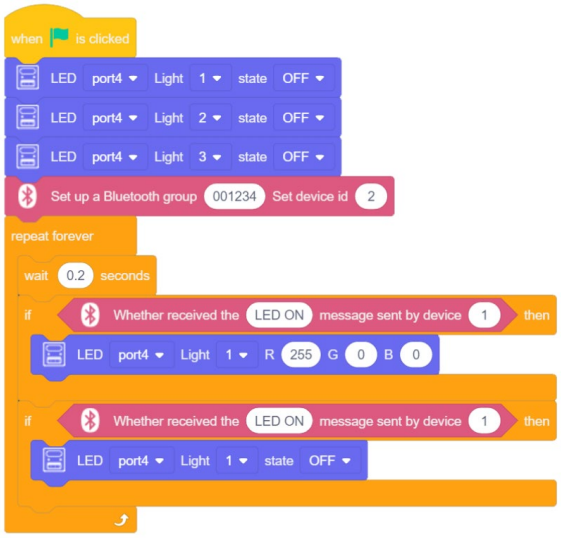
Why do you think there is a delay?

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

The program can be simplified further be using the WHETHER RECEIVED instead of SETTING a variable to the value received



5. Create the program to the right on computer 2.
   a) Keep the same GROUP and ID numbers
   b) The Variable (BLUE) can be removed completely from the program
   c) Drag over a WHETHER RECEIVED and place one in each IF Statement
   d) IF variable BLUE = LED ON, turn ON LED 1 (RED)
   e) IF variable BLUE = LED OFF, turn OFF LED 1
   f) Run both Programs.



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

CHALLENGE

SECTION 4 – PASS the LED SIGNALS (HOT POTATO)

Create a program that will allow BOTH Magic Boxes to cross communicate

Program both Magic Boxes with the same program

1. The BLUE and RED DUAL BUTTONS control LED 1 and LED 2 from the opposite Magic Boxes
2. The RED BUTTON turns LED 1 RED
3. The BLUE BUTTON turns LED 2 BLUE

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

CONCLUSION

1. *List two advantages to wireless communication between devices.*

   A.

   B.

2. *List two disadvantages to wireless communication between devices. Did you experience any?*

   A.

   B.

3. *How could wireless communication in an industrial, or factory type setting?*

GOING BEYOND

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

_____ 1. Use My Blocks to make the challenge above simpler.

_____ 2. Make one LED come on when the other Magic Boxes button is pressed, and another come on when it is not.

_____ 3. Make the challenge above work with two different LED's back and forth, on both magic boxes.

2.3 Block - Pick & Place with the Magic Box

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

INTRODUCTION

Adding sensors to a robot's programming increases its ability to be aware of various conditions in its environment. It also allows the programmer to build in the functions that allow humans to better interact with the robot.

This activity has two main goals. The first goal is to teach the programmer how to safely wire the Magic Box to communicate data between the robot and the magic box.

The second goal is to provide examples on how to program this communication to help provide sensor-based feedback to the robot.

Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

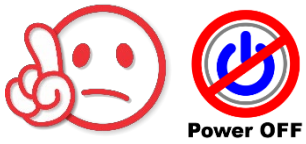
- Magic Box
- 2-Button Input Module

EQUIPMENT & SUPPLIES

- Magician Light
- Dobot Magic Box
- Digital - 2-Button Input Module
- Additional Sensor (Your Choice)
- Magician Lite Work Mat or printed template from *Resources*
- Magician Robot Arm w/ Suction Cup
- 12V Power Cable and Supply
- Common Sensor Cables
- USB C to A Cable
- DobotLab Software

PROCEDURE

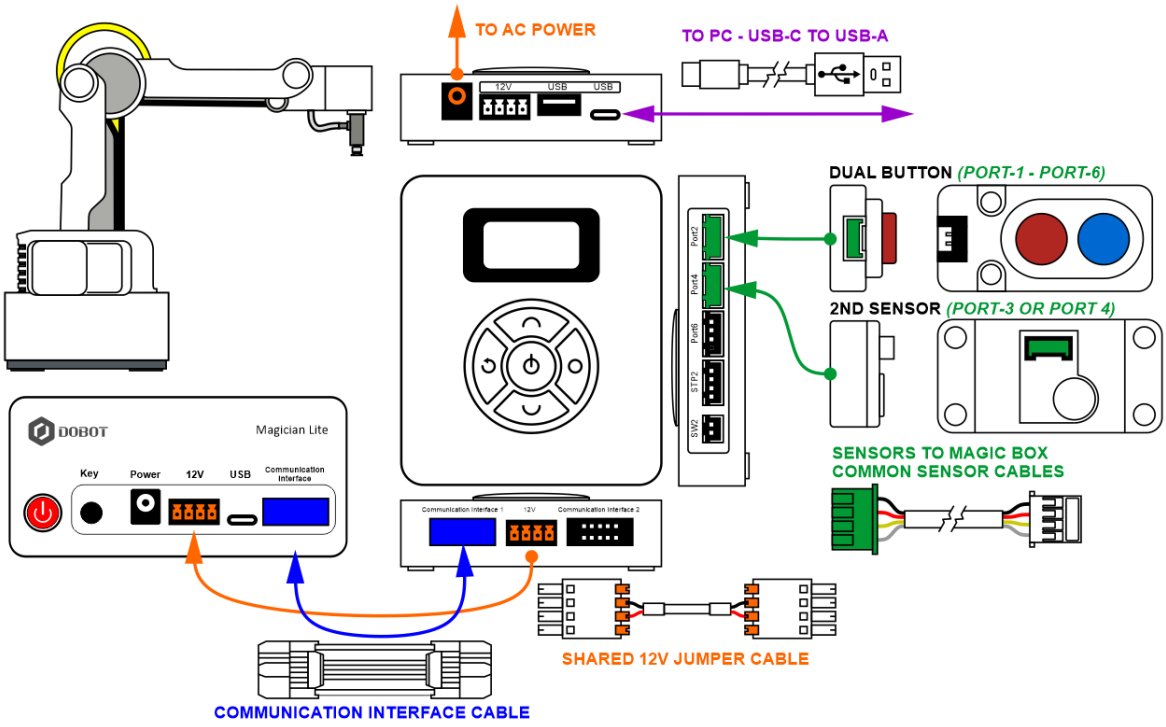
SECTION 1 – WIRING the MAGIC BOX, SENSORS, & MAGICIAN ROBOT



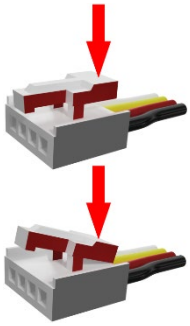
Caution: NEVER wire anything to the Magic Box or Robot while it has power on. ALWAYS turn them off before making connections or damage to the Controllers could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
   - 1x Magic Box
   - 1x USB-A to USB-C Cable
   - 1x AC/DC Power Adapter (12V)
   - 2x Common Sensor Cables
   - 1x Dual Button Sensor – ***PORT 2***
   - 1x Additional Sensor – ***ANY AVAILABLE CORRESPONDING PORT***

2. Wire the Magic Box as shown below.



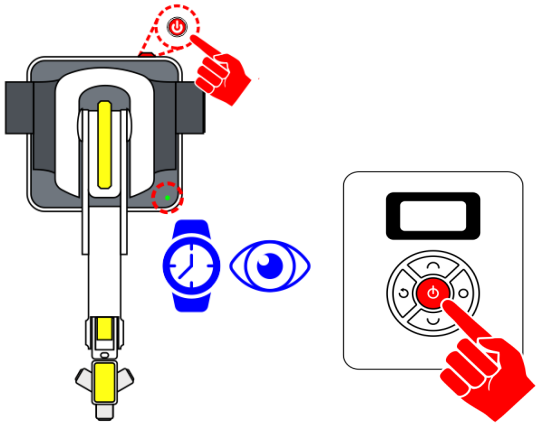
Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



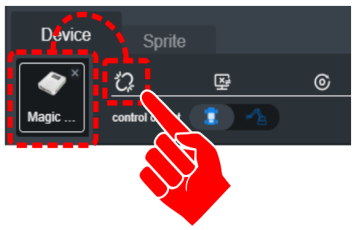
SECTION 2 – CONNECTING the MAGIC BOX & ROBOT to DOBOTLAB

1. Open up DobotBlock Lab in the software.

2. Once **ALL** of the wiring is done (sensors connected and robot arm connected), power ON the Magic Box and Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)

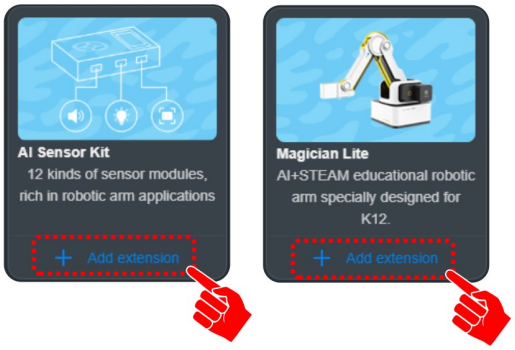


3. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).

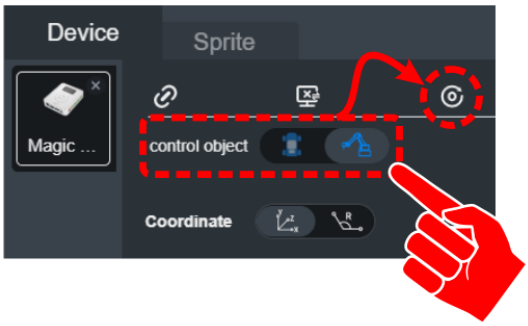


WAIT…. THERE IS NO NEED TO CONNECT TO THE ROBOT? For this activity, you will use the magic box to control the robot (communication cable and added extension)

4. Click on “+ Add extension” for the AI SENSOR KIT & MAGICIAN LITE.

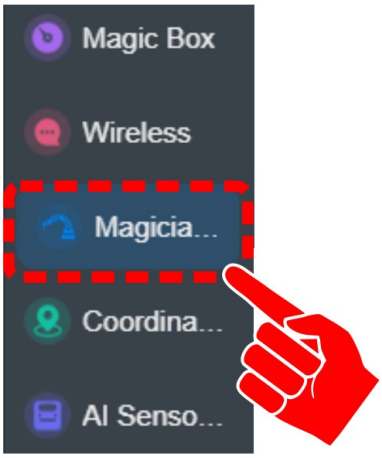


5. Once the EXTENSIONS are added, slide the control object from the Magician GO to the Magician Lite Robot and press the Home button. The robot should move to its home position.



Ensure the robot’s work envelop is clear!

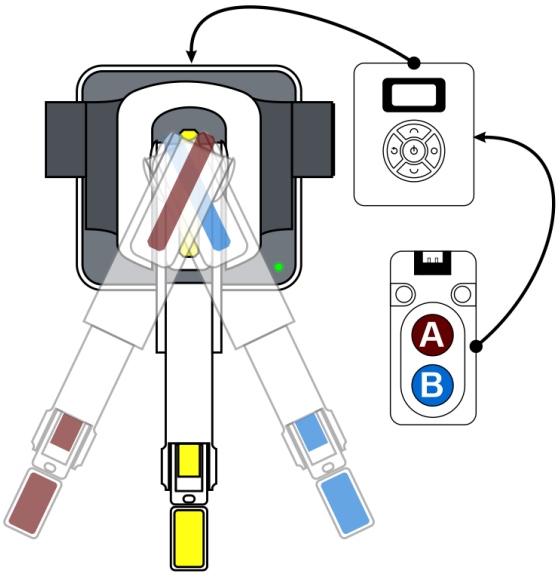
Notice that after adding the Magician Extension, a new category has been added to the tool bar.



SECTION 3 – BASIC PROGRAMMING BETWEEN THE MAGIC BOX & ROBOT ARM

SKILL BUILDER – DUAL BUTTON AND MAGICIAN ARM

This activity will use the DUAL BUTTON input sensor to send the robot to different positions.



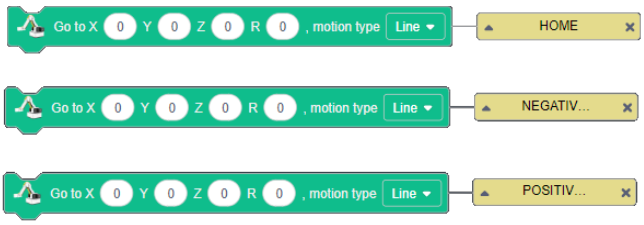
a) When no button is pressed, the robot will go to or remain at the home position.

b) When Button "A" is pressed, the robot will go to a position in the negative direction.

c) When Button "B" is pressed, the robot will go to a position in the positive direction.

d) The robot will stay at a position if the button is held

1. Drag over three GO TO POSITIONS from the NEW category.



2. Find the coordinates and fill them in.

HELPFUL TIPS

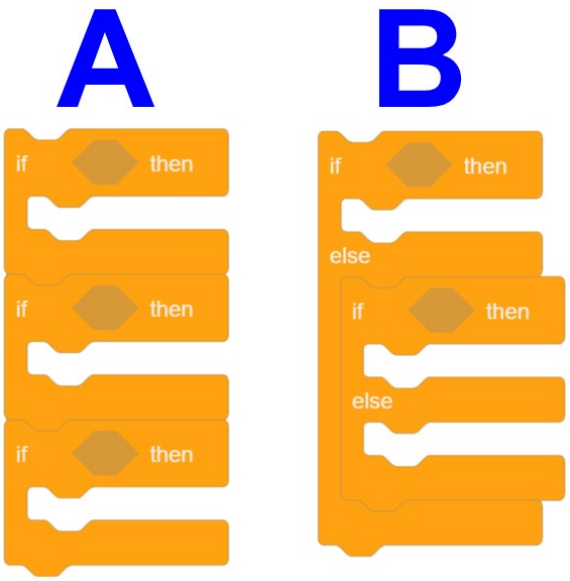
Either drive the robot to the desired positions or use the Unlock Arm Button. Either type in the position numbers or use the fill coordinates option

The robot needs to be able to make a choice of which position it should be at.

There will be three possible locations for the robot to choose from: HOME, NAGATIVE, or POSITIVE.

While there are several ways to program this process, this activity will look at two of them.

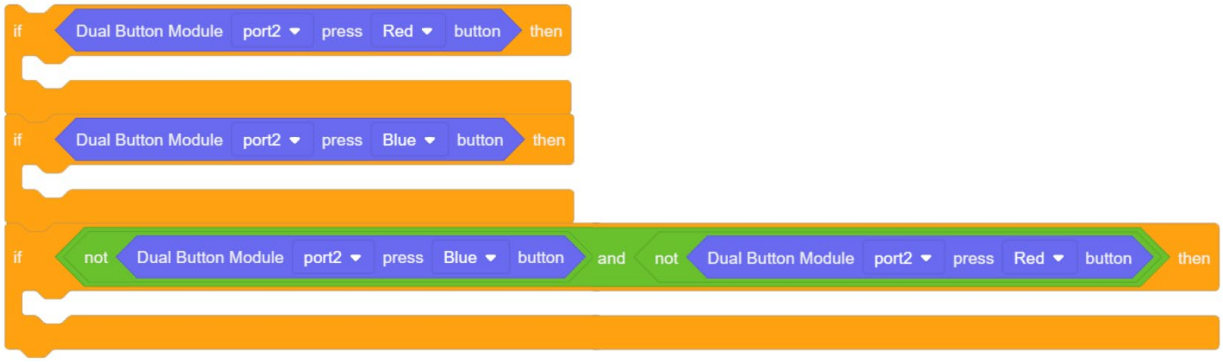
a) A group of individual IF Statements
OR
b) A nested set of IF Statements for this process.



OPTION A - A group of individual IF Statements

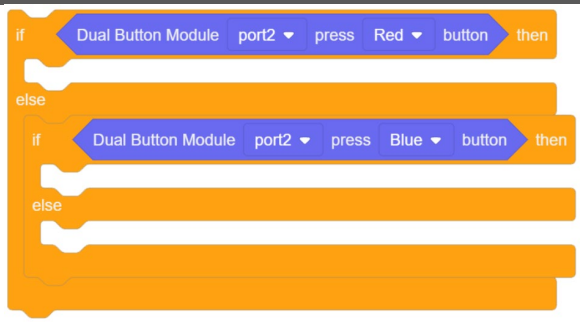
The condition of the DUAL BUTTON will be placed inside the first two IF Statement. The third IF Statement will instruct the robot on what to do IF the buttons are not pressed

Option A provides the programmer the opportunity to build a complex computation for the third condition. The condition should only be met if the RED button and the BLUE button are NOT pressed.



OPTION B - A NESTED SET OF IF STATEMENTS

a) The first condition of the DUAL BUTTON (RED) will be placed inside the IF Statement.
b) The second condition of the DUAL BUTTON (BLUE) will be placed inside the ELSE IF Statement.
c) The third condition false into the final ELSE condition (IF neither condition is TRUE, do this).

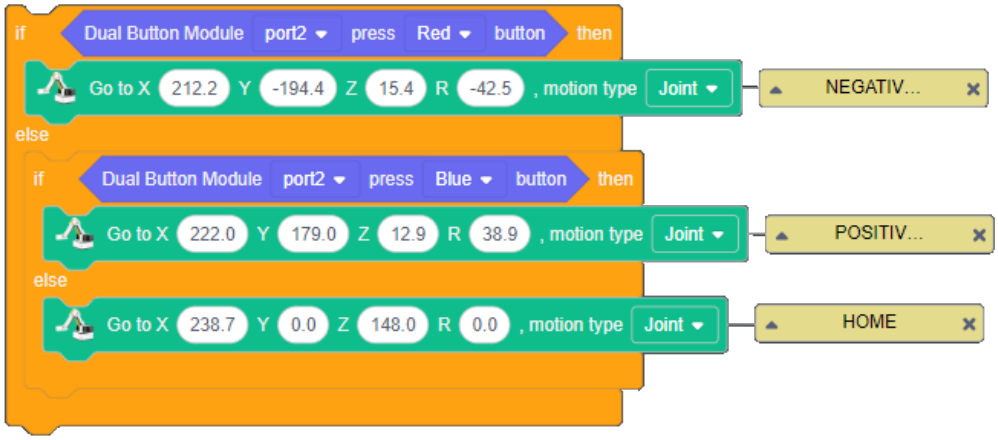


This option simplifies the code and places the choices in a heirarchy (order of importance) that also takes cares of the fourth possible condition… BOTH the RED button and BLUE button are pressed at the SAME time. The RED condition will override the BLUE condition is this format.

ADD IN THE MOVEMENT COMMANDS

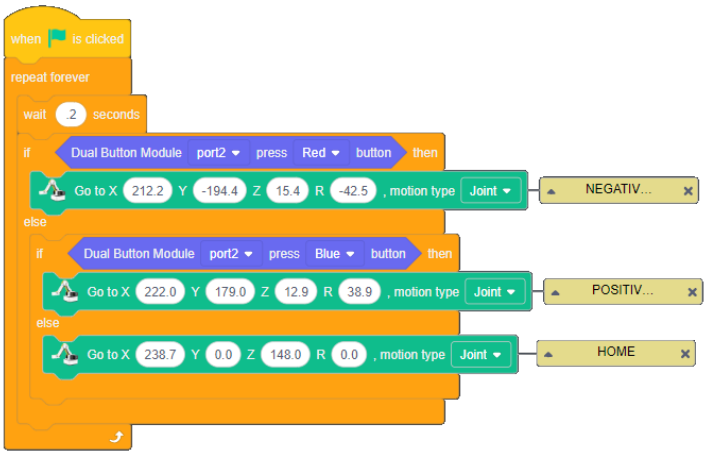
3. Regardless of which option is used, the GO TO POSITION blocks can now be added to each of the Statements.

EXAMPLE:



ADD ADDITIONAL CODE

4. Add additional code to the program to complete it for testing.



a) When Started – Start the process.
b) Forever Loop –Loop/Rewind and repeat the process.
c) Small Micro Wait – Small wait to slow the process of looking for the sensor values.
d) If Nest - Look for conditions, Run conditions if TRUE or Run ELSE.

5. Once the program is completed, run it and see if it works correctly.

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

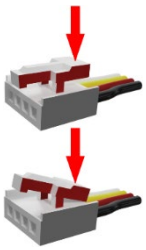
CHALLENGE

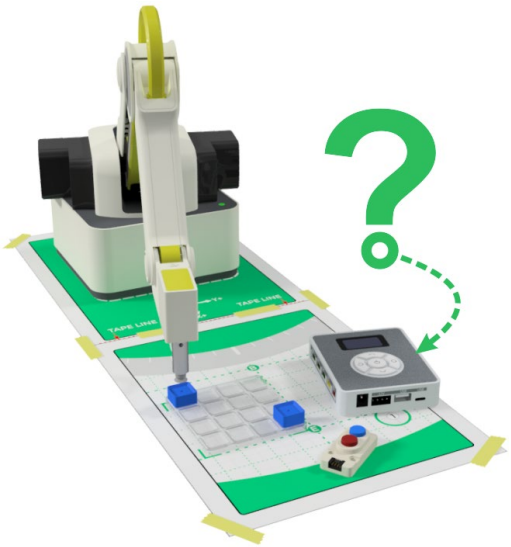
SECTION 4 – USE A DIFFERENT SENSOR(S) TO CONTROL THE ROBOT'S MOVEMENT THROUGH A PICK AND PLACE ROUTINE

For this challenge, choose a different sensor from the kit that you have practiced with the will fit this challenge.

REMINDER: If the sensor is not already attached to the Magic BOX or a sensor is being removed, make sure it is powered off before installing the new sensor to keep from damaging the controller.

If a sensor is bong uninstalled or removed, make sure the tab is completely pressed to keep from damaging the sensor.



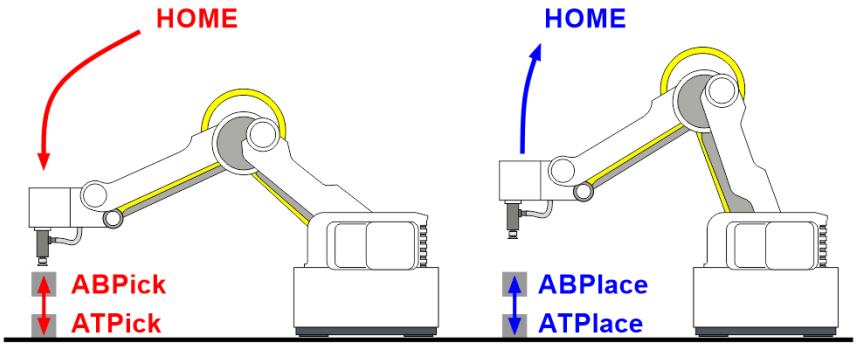


Pick and Place Challenge.

- *The robot will start at a home position and wait for an input signal before moving to the ABOVE PICK location.*
- *The robot will wait at the ABOVE PICK location until a singal (different signal?) is received to pick up the block and jump it over to the AT PLACE location.*
- *The robot will sit at the ABOVE PLACE location until another signal (different signal?) tells it to move home.*
- *The process should then be looped and ready for another loop*

PICK AND PLACE SEQUENCE

1. HOME
2. WAIT FOR CONDITION 1
3. ABPick
4. WAIT FOR CONDITION 2
5. ATPick
6. VACUUM ON
7. JUMP to ATPlace
8. VACUUM OFF
9. ABPlace
10. WAIT FOR CONDITION 3
11. HOME



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

CONCLUSION

1. *Give a reason why the Dual Button module would be useful in industry and explain your answer.*

2. *What sensor did you choose to start the challenge activity? Justify your choice with at least three bullet points.*

3. *What other sensor might be a good choice? Why?*

GOING BEYOND

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

_____ 1. Use the JOYSTICK to control the robot's movements through a pick and place operation. Use a digital input to turn on and off the vacuum gripper.

_____ 2. Use the Dual-button module to complete the following task:

   a. Place a block to be picked up

   b. If the block is red, have the operator press the red button and drop it into a bin for red blocks.

   c. If the block is blue, have the operator press the blue button and drop it in the blue block bin.

   d. If the block is any other color, drop it in a garbage bin by using a different input.

   e. Repeat this forever.

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING
MAGICIAN LITE

- CHAPTER 3 -
ROBOTICS & SENSORS (I/Os)

CHAPTER INTRODUCTION:

Imagine a robot who could sense the temperature of a part that it is working with or one that could tell the color of different objects. Imagine if a robot could tell that the object it was picking up was fragile, so it would pick it up gently. This is not science fiction and happens in factories around the world everyday by equipping robots and machinery in industry with sensors.

There is a wide array of sensors used for many different things and you will get a taste of how to use some sensors with your robotic arm. Some of the sensors used in this chapter include:

- PIR Sensor
- Sound Sensor
- Gesture Sensor
- Humiture sensor
- Potentiometer Sensor
- Color Sensor
- Light Sensor
- Photoelectric switch
- Joystick
- Microservo
- Limit Switch
- LED Module

TOPICS COVERED:

- Inputs Vs Outputs
- Analog Vs Digital Signals
- Sensor Applications

3.1 Block - Intro to I/O– Digital vs Analog

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

INTRODUCTION –

Sensors are all around us helping control various forms of automation in our daily lives. They range from simple proximity sensors that activate automatic doors to the light sensors in your car that automatically turn on and off your headlights. These sensors provide the needed feedback that help control various devices.

Sensors in an industrial environment work in the same manner. They can inform an operator when a machine is low on coolant or help add intelligence to a robotic arm that allows workers to safely work side by side with it in the same environment. This activity will explore how inputs and outputs work and how they may be programmed.

KEY VOCABULARY

- Sensors
- Inputs
- Outputs
- Magic Box
- Digital
- Analog
- Normally Open (NO)
- Normally Closed (NC)

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Analog – Potentiometer Module
- LED Module
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable

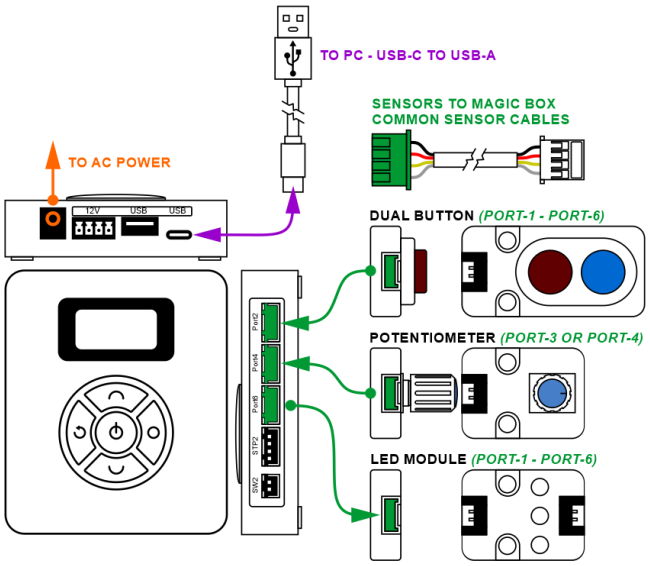
PROCEDURE

SECTION 1 – WIRING the MAGIC BOX AND SENSORS

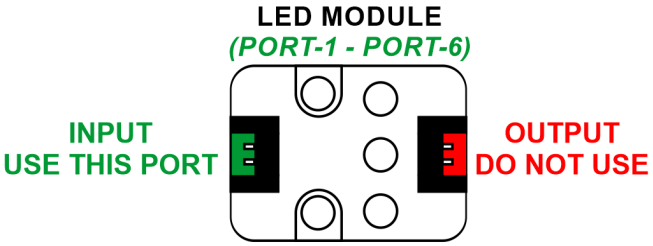
Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
   1x Magic Box
   1x USB-A to USB-C Cable
   1x AC/DC Power Adapter (12V)
   3x Common Sensor Cables
   1x Dual Button Sensor – ***PORT 2***
   1x Potentiometer Sensor – ***PORT 4***
   1x LED Module – ***PORT 6***

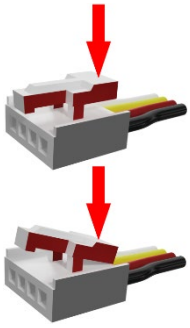
2. Wire the Magic Box with sensors as shown.



3. The LED has two ports, one is an input, and one is an output used to connect multiple devices together. *Be sure to plug it into the Magic box using the port that is farthest from the LED's.*

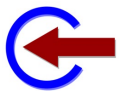


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



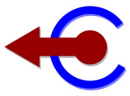
SECTION 2 – DEFINING INPUTS AND OUTPUTS

INPUTS – Defined as any data or signals that a robot receives that provides feedback about its environment. Information that is brought "into" the controller.



Examples: *Sensor Values, Signals from another Device or Robot, User Commands.*

OUTPUTS – Defined as an outgoing device or signal that is controlled by the robot. An output is often something that the robot can turn on or off light a light. An output can also be a value that is produced by the robot's programming.



Examples: *LED, Sound, Motor ON/OFF, Motor Speed and Direction Control, Electromagnet, Vacuum Gripper.*

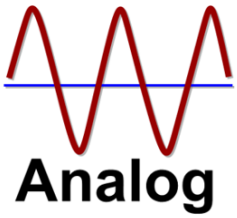
SECTION 3 – DEFINING DIGITAL & ANALOG INPUTS

DIGITAL INPUT/SIGNAL – Defined as a signal that can only be read in one of two states: YES or NO, ON or OFF, TRUE or FALSE, HIGH or LOW.

Digital sensors are often wired as either Normally Open (NO), or Normally Closed (NC). When a digital sensor is wired NO, no signal is received until the sensor is triggered. Once the sensor is active, it reports a high or true signal to the controller. Most often this signal is controlled by electric current or voltage (voltage present or not present). Sensors wired NC work in the opposite state.

Examples: Limit Switch, IR Photoelectric Sensor, Inductive Sensor

ANALOG INPUT/SIGNAL – Defined as a type of signal that has a range of values. This value is often affected by the intensity or magnitude of a measured external quantity (something found in the robot's environment like light or sound).



Examples: Potentiometer, Color Sensor, Humidity Sensor, Light Sensor

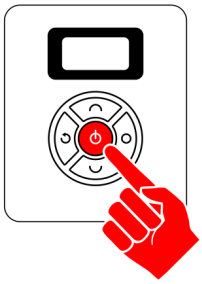
SECTION 4 – CONNECTING the MAGIC BOX to DOBOTLAB

1. Open up DobotBlock Lab in the software.

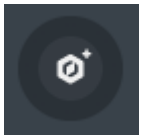
Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.

2. Once all of the wiring is done (sensors connected), power ON the Magic Box

   Dual Button Sensor – ***PORT 2***
   Potentiometer Sensor – ***PORT 4***
   LED Module – ***PORT 6***

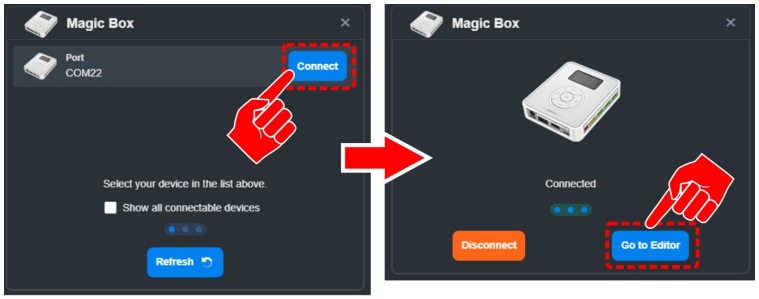


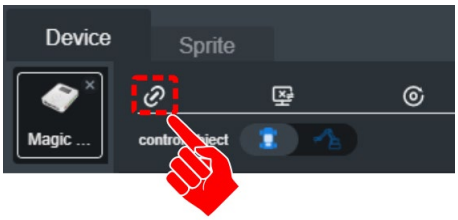
3. **You do not need a robot for his activity.** Delete both the Magician and Magician Lite Devices. Note: This will then default to the Sprite tab once there are no devices loaded.
4. Click back on the devices tab and click on Choose a Device



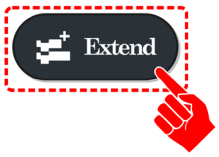
5. Select the Magic Box

6. Connect to the Device

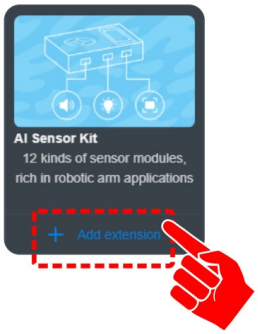




7. For this activity, we will need to add an EXTENSION to use the MAGIC BOX. Click the EXTEND icon in the bottom left corner.



8. Click on “+ Add extension” for the AI SENSOR KIT.



SECTION 5 – EXPLORATION - DIGITAL INPUTS & with OUTPUTS

SKILL BUILDER 1 – LED MODULE – DIGITAL OUTPUT

1. The first device used in this activity is the LED OUTPUT MODULE.

2. From the AI SENSOR KIT toolbox, LIGHT CATEGORY, drag over the LED RGB and LED STATE blocks.

   Settings: PORT 6, LIGHT 1

   Now turn on the FIRST LED as RED for 3 seconds, then turn off the LED STATE and end the program.

3. The values to the right are examples of the values used to establish a specific color for the LED Module.
   The combination of R255 G0 B0 sets the color RED

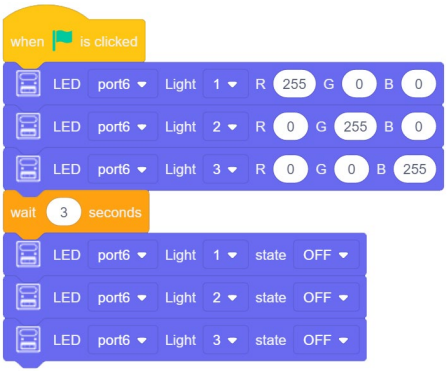
WHITE	rgb (255,255,255)
RED	rgb (255,0,0)
LIME	rgb (0,255,0)
BLUE	rgb (0,0,255)
YELLOW	rgb (255,255,0)

TEST your code, troubleshoot if necessary.
Be sure the LED module is plugged in correctly and the port is called out correctly.

Can you make the LED display a color like Chartruese? Or Cyan? Look up what those colors are on the internet and see if you can make one of these colors with one of the LED’s!

4. Next turn on all three LEDS, 1=RED, 2=GREEN, 3=BLUE for 3 seconds

WHITE		rgb (255,255,255)
RED		rgb (255,0,0)
LIME		rgb (0,255,0)
BLUE		rgb (0,0,255)
YELLOW		rgb (255,255,0)



If the program does not call for the LED Outputs to turn OFF at the end of the program, they will stay on even after the program has ended.

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

Can you make the LED display one after the other for ½ a second each? Can you make the LED's "chase"one another? How?

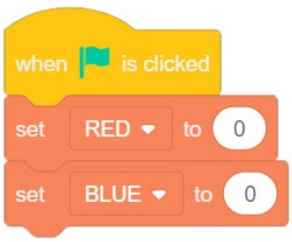
If your set up did not work correctly the first time, what did you have to do to make it work?

SKILL BUILDER 2 – DUAL BUTTON – DIGITAL INPUT

5. The second device in this activity is the DIGITAL DUAL BUTTON INPUT MODULE.

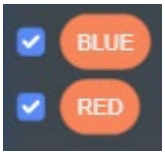
6. Since the DUAL BOTTON MODULE is an **Input** signal, we need to create a way to "SEE" the values that the sensor is reporting to the **Magic Box** (ON/OFF, 1/0)

   Create TWO new variables (RED and BLUE) and set their initial values to ZERO



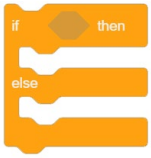
HELPFUL TIPS

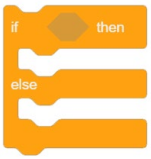
As long as there are check marks next to the variables you created, they should appear in the SPRITE window.



7. Next, setup two separate IF/ELSE Statements.

   CONDITIONS: If the Button is pressed, set the Variable to ONE, else set it to ZERO.





8. From the AI SENSOR KIT toolbox, SENSOR CATEGORY, drag over the DUAL BUTTON MODULE blocks and place it inside one of the IF/ELSE Statements.

   Settings: PORT 2, RED

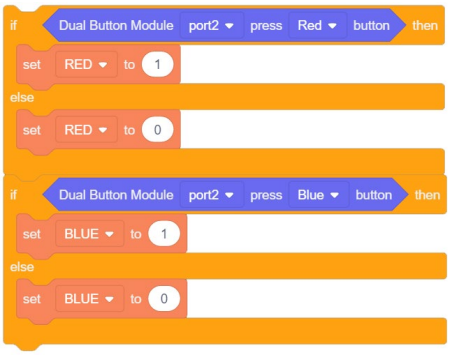
   This will serve as the condition for this statement.
   These sensors are wired NORMALLY OPEN (NO).
   BUTTON PRESSED = TRUE (ON, 1)
   BUTTON NOT PRESSED = FALSE (OFF,))

9. Next, setup what we want the IF STATEMENT to DO

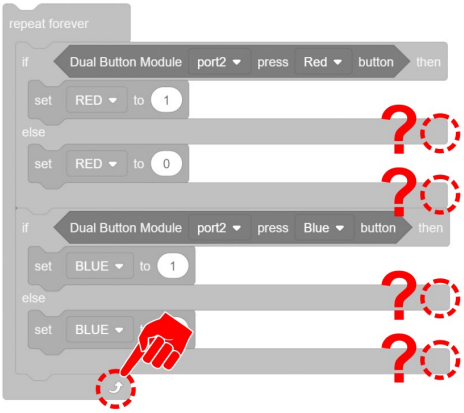
   Drag the SET Variable block into both parts of the IF STATEMENT

   STATEMENT IS TRUE = VARIABLE is 1
   STATEMENT IS FALSE (ELSE) = VARIABLE is 0

10. Repeat this process for the other IF/ELSE Statement (or duplicate the first one and change the values)



11. You need to continuously evaluate these sensors to see when they are true and false. To do this, we will put them inside a forever loop

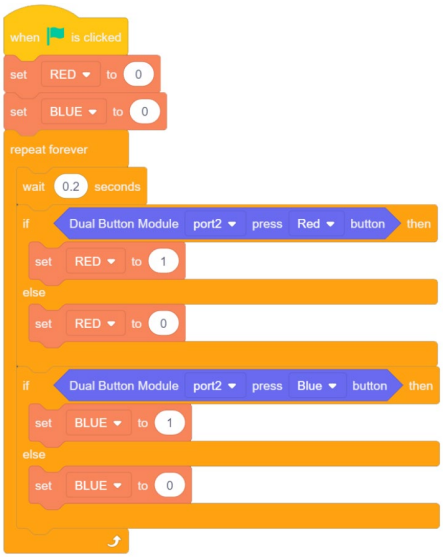


Notice that the IF/ELSE STATEMENTS do not have repeat arrows at the bottom of them. IF STATEMENTS will only be evaluated at the start of each loop. If they are true, they will run the entire process inside the IF portion, FALSE, everything inside the ELSE portion. IF STATEMENTS then move onto the next section of code (NO REPEAT). Both of our Statements do not have a perceivable time needed to run the code and move on, this will allow both buttons to be pressed at the same time and both return true conditions.

12. Add on the header code to finish the program

13. Add a **Micro Wait** inside the Forever Loop

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



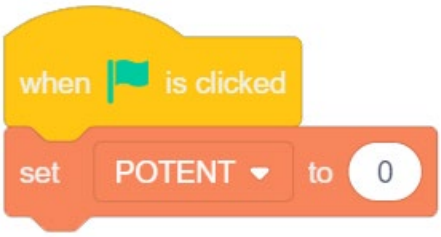
If your set up did not work correctly the first time, what did you have to do to make it work?

SKILL BUILDER 3 – POTENTIOMETER KNOB – ANALOG INPUT

14. The Third device used in this activity is the POTENTIOMETER KNOB INPUT MODULE. This device is an **analog input**.

15. Since the POTENTIOMETER MODULE is an Input signal, create a VARIABLE as a way to "SEE" the values the sensor is reporting to the magic box.

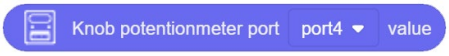
    Create a new variables (POTENT) and set its initial values to ZERO. This will clear out any initial values for this variable.



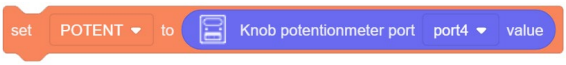
16. Next inside a FOREVER LOOP block, we will assign the variables value to the sensors value.

    From the AI SENSOR KIT toolbox, SENSOR CATEGORY, drag over the KNOB POTENT VALUE block.

Settings: PORT 4

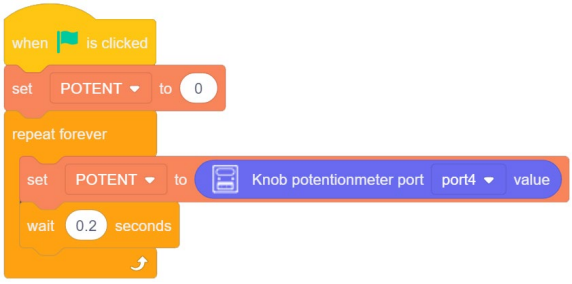


17. Drag the sensors value into the SET VARIABLE block.



    This will allow the variable to update to whatever reading the sensor is currently reading… but only if it is in a FOREVER LOOP block.

    Place a SMALL MICRO WAIT (0.2) after setting the variable to the sensors value before it repeats the loop. This will slow down the cycle and not pull the value from the sensor so quickly.



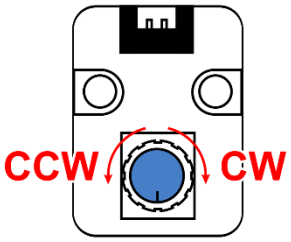
Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

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

18. What was the **Highest** Value the sensor reported?

19. What was the **Lowest** Value the sensor reported?

20. Which direction **increases** the value (CW / CCW)



CHALLENGE

SECTION 6 – CONTROL OUTPUTS WITH INPUTS

Challenge Scenario: ALL of the conditions below are in the same program

- When the BLUE button is pressed, LIGHT 1 will turn BLUE
- When the RED button is pressed, LIGHT 3 will turn RED
- The corresponding lights will be OFF when the corresponding buttons are NOT pressed
- Divide the POTENTIOMETER High value into thirds –

***NOTE – Use GREATER THAN, LESS THAN, and possibly the AND OPERATORS for the next portion to create the needed conditions (Hexagon Shapes) for the analog sensor?*

- When the POTENTIOMETER value is in the bottom third or the top third, LIGHT 2 will be WHITE
- When the POTENTIOMETER value is in the middle third, LIGHT 2 will be YELLOW

Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does. SHOW YOUR INSTRUCTOR WHEN THE CHALLENGE IS COMPLETE

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

CONCLUSION

1. *What type of values are reported from analog sensors?*

2. *What type of sensors would be would you find in an automated vehicle? List as many as you can think and identify them as analog or digital.*

3. *What type of sensors might a robot use/need to allow people to work near the robot or inside its work envelope?*

4. *What are the RGB values for*

   MAGENTA: *ORANGE:* *GREY:*

GOING BEYOND

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

_____ 1. RED BUTTON is pressed, LIGHT TURNS RED, but only for 3 seconds (even if the button is still held)

_____ 2. ALL three lights are off unless, the red button and the blue button are pressed and held as well as the potentiometer is at its lowest setting.

If all three conditions are met, make all three lights turn magenta… until the three conditions are not met again.

_____ 3. When the RED BUTTON IS PRESSED make the 3 LED's "chase one another to the right as RED. When the BLUE BUTTON IS PRESSED, make the LED's "chase" one another to the left 5 times as BLUE.

_____ 4. When the POTENTIOMETER is in the FIRST THIRD OF ITS TRAVEL Turn on the LED TO RED. When the POTENTIOMETER is in the MIDDLE THIRD OF ITS TRAVEL Turn on the LED TO YELLOW. When the POTENTIOMETER is in the LAST THIRD OF ITS TRAVEL Turn on the LED TO GREEN.

3.2 Block - Exploring I/Os with the Magic Box

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

INTRODUCTION

In this activity you will get a chance to use many other types of sensors, inputs, and outputs with the magic box. These include:

- PIR Sensor
- Sound Sensor
- Gesture Sensor
- Humiture sensor
- Light Sensor
- Photoelectric switch
- Joystick
- Microservo

KEY VOCABULARY

- Input / Output (I/O)
- Pulse width modulation (PWM)
- UART
- Analog/Digital
- Analog to digital conversion (ADC)

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Micro Servo
- Photoelectric Sensor
- Joystick
- Light Sensor
- Sound Sensor
- PIR Sensor
- Gesture Sensor
- Humiture Sensor
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable

PROCEDURE

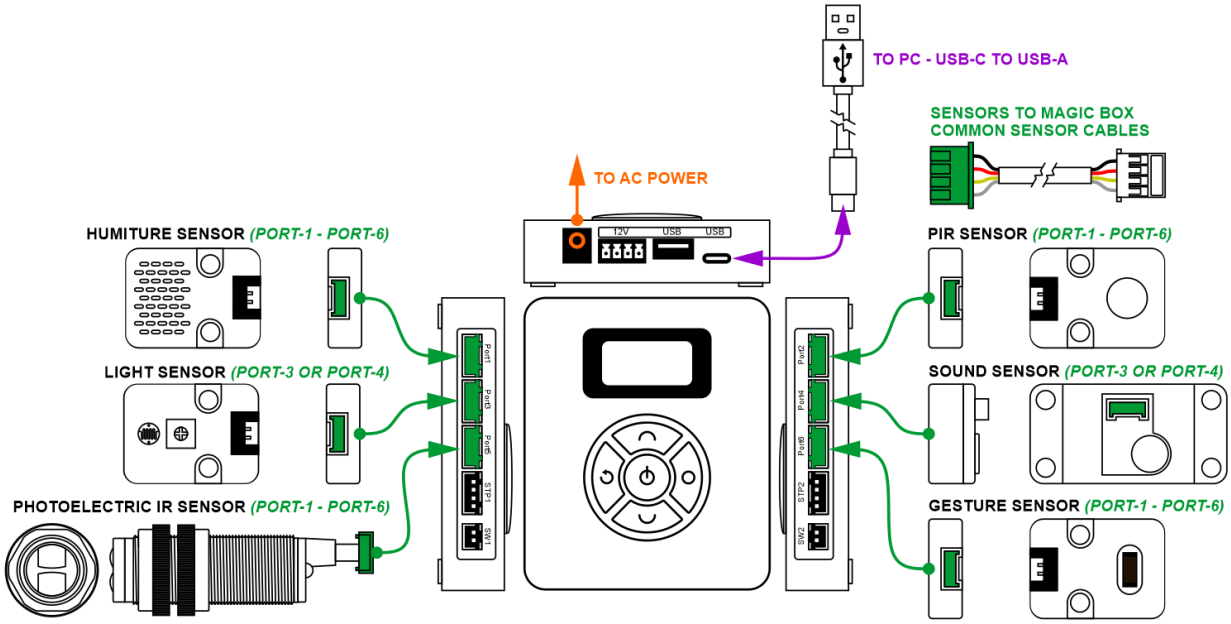
SECTION 1 – WIRING the MAGIC BOX AND SENSORS – 1st SETUP

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

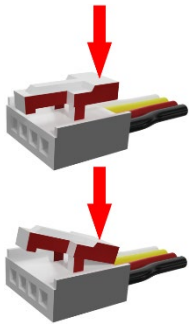
SETUP #1

1. For this activity, you will need:
   - 1x Magic Box
   - 1x USB-A to USB-C Cable
   - 1x AC/DC Power Adapter (12V)
   - 5x Common Sensor Cables (Photoelectric Sensor does not need a common sensor cable)
   - 1x PIR Sensor – ***PORT 2***
   - 1x Sound Sensor – ***PORT 4***
   - 1x Gesture Sensor – ***PORT 6***
   - 1x Humiture Sensor – ***PORT 1***
   - 1x Light Sensor – ***PORT 3***
   - 1x Photoelectric IR Sensor – ***PORT 5***

2. Wire the Magic Box as shown below:

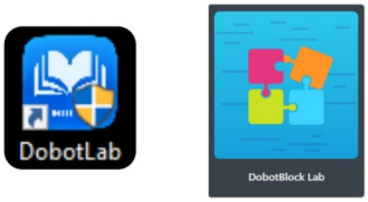


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

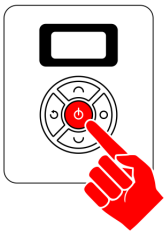


SECTION 2 – CONNECTING the MAGIC BOX to DOBOTLAB

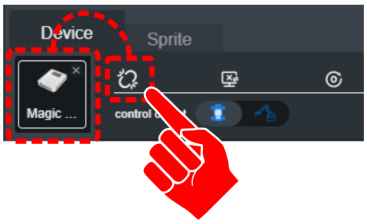
1. Open up DobotBlock Lab in the software.



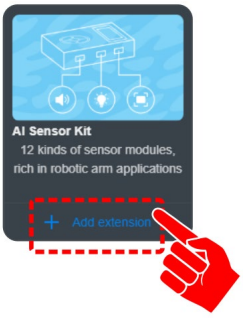
2. Once all of the wiring is done and all of the sensors connected, power ON the Magic Box



3. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).



4. Click on "+ Add extension" for the AI SENSOR KIT.



SECTION 3 – SETUP #1 - EXPLORING ADVANCED SENSORS

- ***PIR, SOUND, GESTURE, HUMITURE, LIGHT, PHOTOELECTRIC***

SKILL BUILDER 1 – PIR SENSOR – DIGITAL INPUT

1. The first sensor explored in this activity is the PIR SENSOR. In the software it is referred to as a **Human Sensor**.

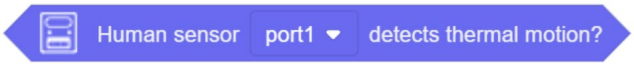
   The PIR sensor is a body infrared sensor. It detects the infrared radiation emitted or reflected by the human body or other objects. When the sensor detects infrared it outputs a **TRUE** signal. It then carries out a time delay (during which the true signal is maintained and repeated triggering is allowed) until the triggering signal disappears (restoring **FALSE** signal). Note: There is a two-second delay after each detection is triggered.

ANY GREEN PORT

PORT 1-6

2. Drag over a Human Sensor Detects block from the AI SENSOR KIT toolbox

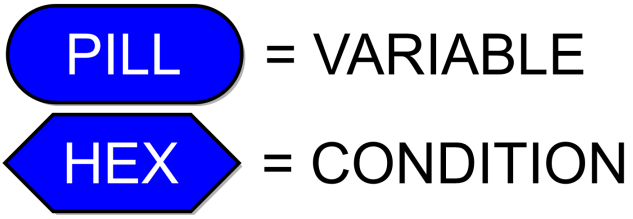
   This block will produce a TRUE result if thermal motion is detected



3. Change the PORT value to the correct port (see page 2).

4. A few simple lines of code are needed to retrieve values from the sensors used throughout this activity.

   Some of the sensors can be tied to a **variable** (PILL shaped) and some will need to be used as a **condition** in a loop (HEX shaped)

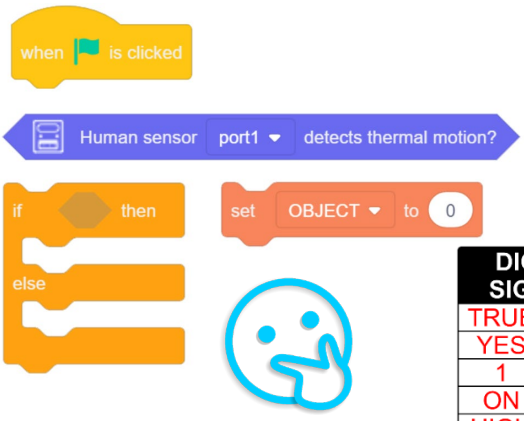


5. This sensor is a digital sensor. It will only report a TRUE/HIGH or FALSE/LOW signal.

   The values for this sensor are reported by HEX shapes. Create a variable (OBJECT) and an IF/ELSE Statement.

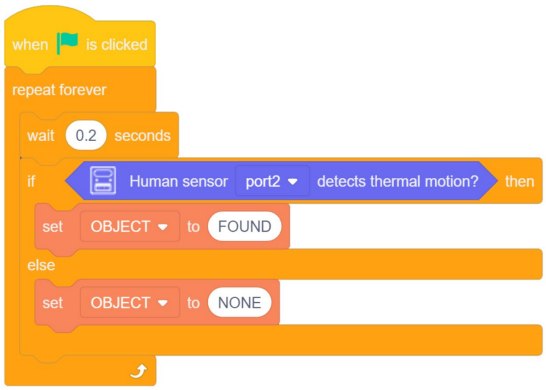
   A TRUE condition will change the variable OBJECT to FOUND

   ELSE (not true) the variable will show as NONE



DIGITAL SIGNALS	
TRUE	FALSE
YES	NO
1	0
ON	OFF
HIGH	LOW

6. Place everything inside a forever loop with a small micro wait.
7. Download the code to the Magic Box. Practice with this group of code and record your results below.



A. What type of range does the sensor have? (How far can the sensor detect motion?)

B. What type of vision does the sensor have? (Different angles, all the way around?)

C. What type of motion does the sensor detect? (side to side, forward backward, rotation?)

D. After the object is still (no motion) for more than two seconds, what happens?

E. Can the sensor detect motion of an object?
-Ball rolling, does it have to be large (marble?)
-Something tossed across its field of view (did it detect the object or you tossing the object)

F. What happens if the motion is smooth and continuous (rotating fan or oscillating fan, back & forth)

G. Can you or an object move slow enough to keep the sensor from reading TRUE?

SKILL BUILDER 2 – SOUND SENSOR – ANALOG INPUT

1. The second sensor this activity will explore is the SOUND SENSOR.

The sound sensor is used to detect the sound intensity of the surroundings. The greater the sound intensity it receives, the stronger the output signal and the greater the return value is.

PORT 3 or 4

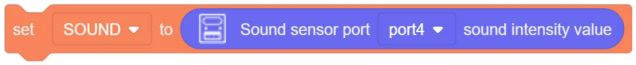
2. Drag over a Sound Intensity Value from the AI SENSOR KIT toolbox

This sensor is an analog sensor. The values produces by this sensor will increase or decrease depending on the level/intensity of the sound it receives.

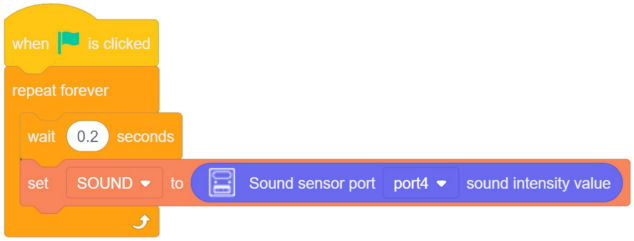
3. Change the PORT value to the correct port (see page 2).

4. The values for this sensor are reported to a PILL shape. Create a variable (SOUND).

1. Add this block to a SET VARIABLE TO "SOUND".



5. Place everything inside a forever loop with a small micro wait.



6. Download the code to the Magic Box. Practice with this group of code and record your results on the next page.

A. Can the sensor detect a whisper?

B. What analog value is reported when no sound is present?

C. Does the length of time the sound is heard by the sensor change the value?

D. What is the highest value you can get the sensor to report?

E. Document additional observations below.

SKILL BUILDER 3 – GESTURE SENSOR – DIGITAL INPUT

1. The Third sensor this activity will explore is the GESTURE SENSOR

   The gesture sensor is a 3D gesture recognition sensor using a I2C communication interface. It supports eight types of gestures by default and it is suitable for a variety of applications, including non-contact remote control, robot interaction, human-machine interaction games, and gesture lighting control.

ANY GREEN PORT

PORT 1-6

2. Drag over the two main blocks related to the gesture sensor from the AI SENSOR KIT toolbox. *NOTE: We will not use the additional block for GESTURES in these activities.*

HELPFUL TIPS

Notice that this sensor has both PILL and HEX Blocks. The GESTURE SENSOR is an intelligent sensor that can report a range of responses.

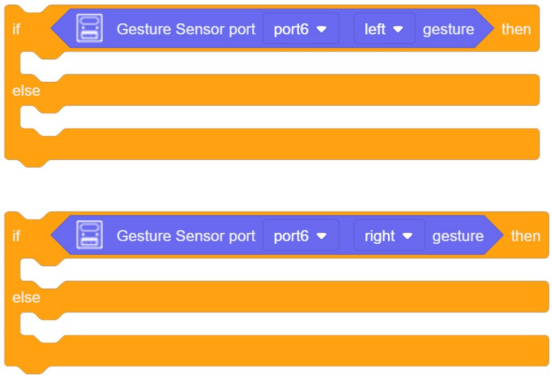
It can be used as a condition for a loop as we did with the PIR Sensor.
Example: Is left motion detected (Set variable to TRUE / FALSE)

It can also be paired with a variable to report which of the 8 motion is being detected in words (string values)

3. Change the PORT value to the correct port (see page 2).

4. Create TWO IF/ELSE Statements

5. Place the a GESTURE TRUE block inside each (one of the set to LEFT is TRUE and one the RIGHT is TRUE)

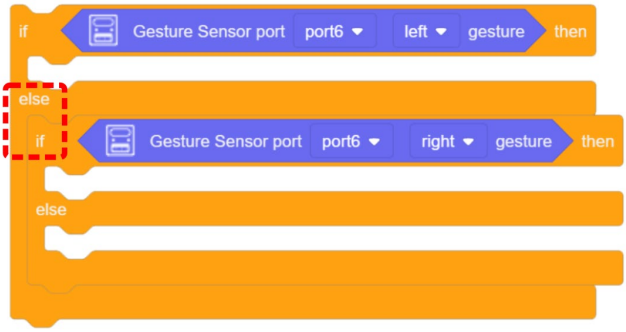


6. The next step will be a new use of the IF/ELSE statement.

   Drag the second IF/ELSE into the ELSE condition of the first IF/ELSE.

   This creates a condition known as an ELSE IF. This type of formatting creates a hierarchy or order of importance to the IF statement.

   If two conditions could be true at the same time, nesting them in this fashion lets the program make a decision on which operation to do (which is more important). It also allows the program to still contain an ELSE (what to do if no conditions are TRUE).



Two IF/ELSE loops will not work for this example. When either condition is not true, the ELSE will always want to set the Variable to NONE.

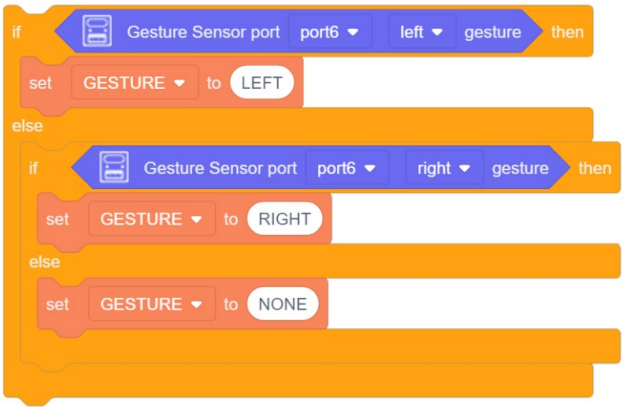
Two single IF statement loops will partially work for this example. Each will clear out the variable when the other becomes true, but nothing will clear out the variable when neither is true. It will start blank (since we did not initialize it to a value), but once either condition is true, it will keep repeating that until told to do something different.

7. Create a new variable (GESTURE) and drag over (or duplicate) three of them (one for each condition).

   GESTURE LEFT (Most important, *do this if left is true… even if right is also true)*

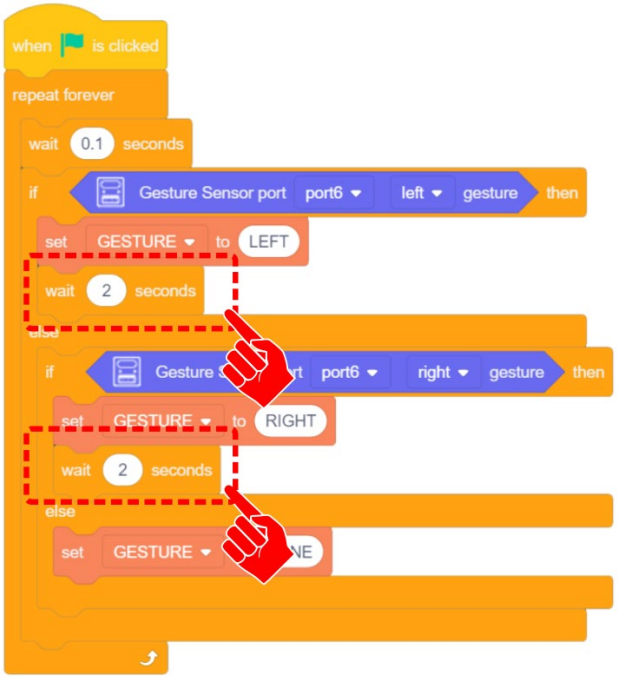
   GESTURE RIGHT (2nd Most important, *do this if right is true… as long as left is not true*)

   GESTURE NONE (What to do if LEFT or RIGHT are not TRUE)



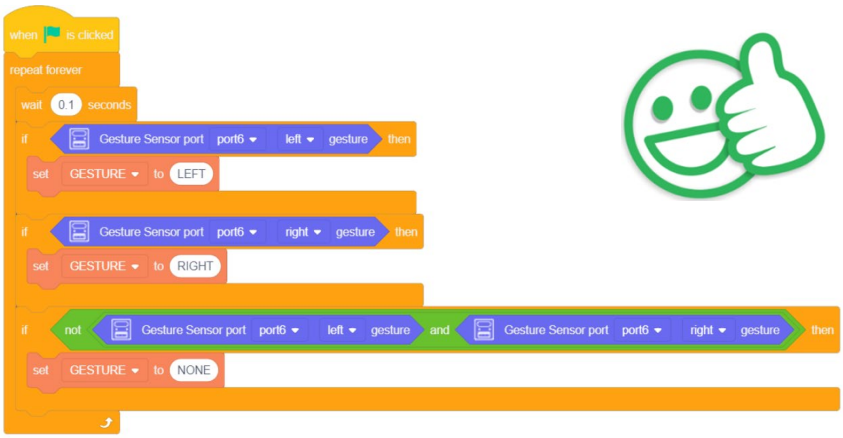
8. Add a WAIT inside the LEFT and RIGHT conditions so the string of text will remain up for a set time before clearing

9. Place everything inside a forever loop with a small micro wait.



What is the micro wait for?

An alternative to the ELSE IF would be to create three IF Statements with a NOT condition (BOTH LEFT and RIGHT are NOT TRUE)

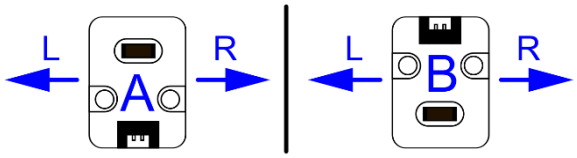


10. Download the code to the Magic Box. Practice with this group of code and record your results below.

A. Does SPEED matter (how fast the movement goes across the sensor)?

B. Does DISTANCE matter (how far the movement is from the sensor)?

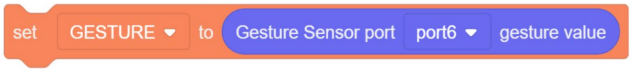
C. Which orientation reads correctly "A" or "B"?



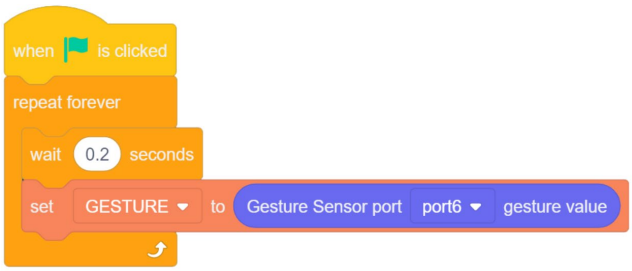
D. Does one direction or the other (left or right) seem to read more often (more consistently)?

E. Document additional observations below

1. This time, add the GESTURE SENSOR VALUE to a SET VARIABLE TO “GESTURE”.



2. Place everything inside a forever loop with a small micro wait.



3. Download the code to the Magic Box. Practice with this group of code and record your results below.

F. List all 8 gestures the sensor is capable of reading and indicate if they can be read consistently (majority of the time), inconsistently (sometimes), or rarely.

1. Consistent / Inconsistent / Rarely	5. Consistent / Inconsistent / Rarely
2. Consistent / Inconsistent / Rarely	6. Consistent / Inconsistent / Rarely
3. Consistent / Inconsistent / Rarely	7. Consistent / Inconsistent / Rarely
4. Consistent / Inconsistent / Rarely	8. Consistent / Inconsistent / Rarely

G. Does the orientation of the sensor matter?

H. Can the sensor see the gesture if it is a moving object? Move the sensor across the object instead of moving an object across the sensor… rotate the sensor… move the sensor toward an object…

*I. Is this block **MORE** consistent, **LESS** consistent, or about the **SAME** as the other block with LEFT and RIGHT gestures?*

J. Document additional observations below.

SKILL BUILDER 4 – HUMITURE SENSOR – ANALOG INPUT

1. The forth sensor this activity will explore is the HUMITURE SENSOR

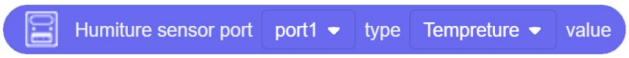
   The humiture sensor is used to detect either temperature or humidity (amount of water vapor in the atmosphere) found in the current environment. It reports analog signals to the controller

ANY GREEN PORT

PORT 1-6

2. Drag over a Humiture Intensity Value from the AI SENSOR KIT toolbox

   This one block is capable of reading either Temperature or Humidity.



3. Change the PORT value to the correct port (see page 2).

4. Create a new variable (HUMITURE) and place it inside a Set Variable loop as done in previous activities

5. Start with the Temperature value

6. Download the code to the Magic Box. Practice with this group of code and record your results below.

A. What value do you get a room temperature (a default reading)?

B. Looking at the reading you receive at room temperature, does the value make sense (F vs C… maybe…)?

C. Try to increase and decrease the temperature. Think of creative ways to increase or decrease the temperature reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method do you use?

Lowest Value: Method:	Highest Value: Method:

D. If the readings do not directly relate to something known (F or C). Can the data still be used? If so, how?

YES / NO HOW:

E. Document additional observations below.

7. Change the Sensor Reading to HUMIDITY

8. Download the code to the Magic Box. Practice with this group of code and record your results below.

F. What default value do you get (current amount of water vapor in the air)?

G. Try to increase and decrease the sensor reading. Think of creative ways to increase or decrease the humidity reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method do you use?

Lowest Value:
Method:

Highest Value:
Method:

H. Document additional observations below.

SKILL BUILDER 5 – LIGHT SENSOR – ANALOG INPUT

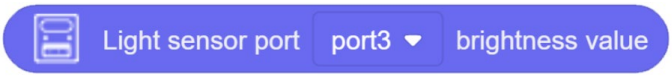
1. The fifth sensor this activity will explore is the LIGHT SENSOR

   PORT 3 or 4

   The light sensor contains a photosensitive resistor. The resistance value decreases with the increase of light intensity. Based on this, the change of its voltage is detected, and the light intensity data is obtained through AD conversion.

2. Drag over a Light Brightness Value from the AI SENSOR KIT toolbox

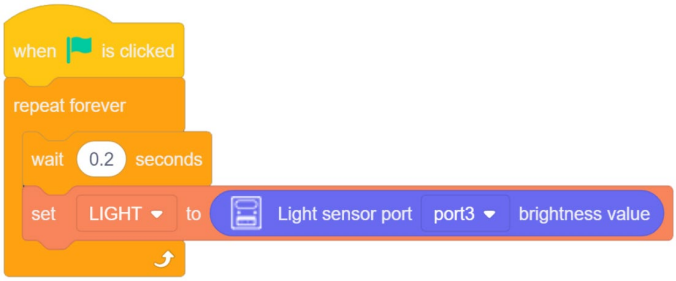
   

   This one block is capable of reading the brightness or intensity of the light seen in an environment.

3. Change the PORT value to the correct port (see page 2).

4. Create a new variable (LIGHT) and place it inside a Set Variable loop as done in previous activities

5. Download the code to the Magic Box. Practice with this group of code and record your results below.



A. What default value do you get (Amount of ambient light, no direct alteration to the amount of light in an environment)?

B. Try to increase and decrease the sensor reading. Think of creative ways to increase or decrease the light reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method did you use?

Lowest Value: Method:	Highest Value: Method:

C. Do shadows have effect on the sensor readings (stand over the sensor to create a shadow)?

D. Document additional observations below.

SKILL BUILDER 6 – PHOTOELECTRIC IR SENSOR –DIGITAL INPUT

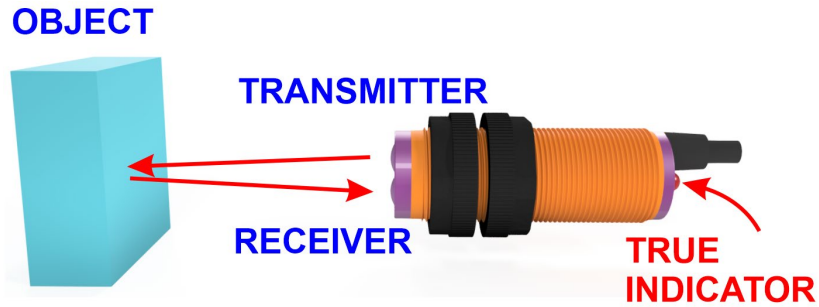
1. The sixth sensor this activity will explore is the PHOTOELECTRIC IR SENSOR.

   The photoelectric sensor, often called a photoelectric proximity switch, or IR sensor is a digital input. It detects the presence of an object through the connected circuit if there is an object shielding or reflecting the IR beam. The photoelectric sensor converts the input current into an optical signal on the transmitter side. The receiver side detects a target object if the transmitter signal is received by the receiver.

   The detection distance can be adjusted by rotating the screw on the backside of the sensor. Rotating the screw clockwise will increase the distance.

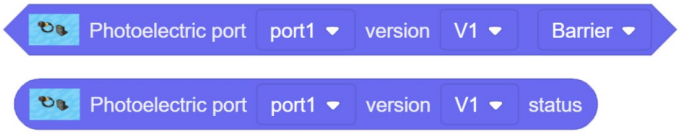
ANY GREEN PORT

PORT 1-6



1. Drag over both Photoelectric tools from the AI SENSOR KIT toolbox

These blocks both report the status of the photoelectric sensor.



2. Change the PORT values to the correct ports (see page 2) and set the VERSION to V2.

3. Create a new variable (PHOTOIR) and place it inside a Set Variable loop as done in previous activities

4. Download the code to the Magic Box. Practice with this group of code and record your results below.

A. What values does the variable report? If the program does not work, try to change the version number from V2 to V1.

What value for ON - Object Found:	What value for OFF – NO Object Found:

B. Document additional observations below.

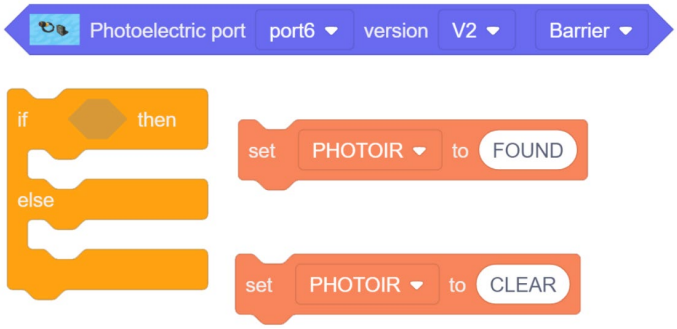
Note that there is a small red LED on the back of the sensor the light when a singal is received.

5. Notice from the previous test that photoelectric sensor is a digital sensor. The last block can only report either a 1 or a 0. This new block will allow us to set the variable to report the same numbers values **OR** a string value of our choosing.

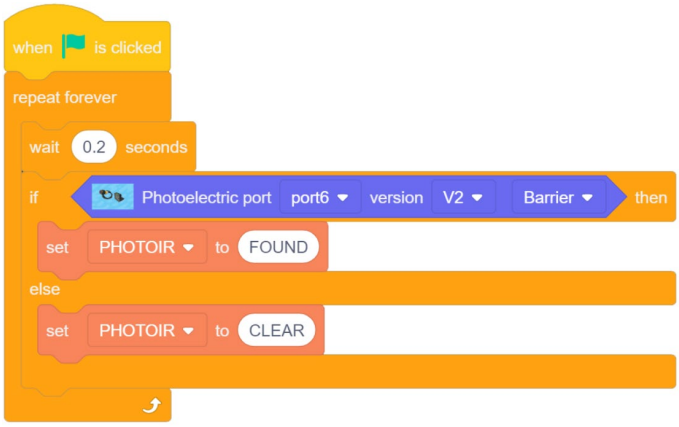
   Create an IF/ELSE Statement.

   A TRUE condition will change the variable PHOTOIR to FOUND

   ELSE (not true) the variable will show as CLEAR



6. Place everything inside a forever loop with a small micro wait.
7. Download the code to the Magic Box. Practice with this group of code and record your results below.



C. *Using the setting BARRIER, what values does the variable report?*

What value for ON:	What value for OFF:

D. *Using the setting NO BARRIER, what values does the variable report?*

What value for ON:	What value for OFF:

E. *What happens when you switch between the two settings?*

F. *What is the terminology used for when a switch reads False by default when no signal is present? (Term was identified in a previous activity when digital inputs were introduced)*

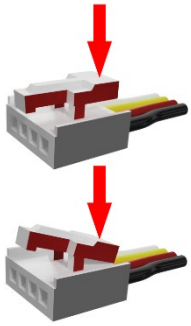
G. *What is the terminology used for when a switch reads True by default when no signal is present? (Term was identified in a previous activity when digital inputs were introduced)*

H. *Document additional observations below.*

SECTION 4 – WIRING the MAGIC BOX AND SENSORS –SETUP #2

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

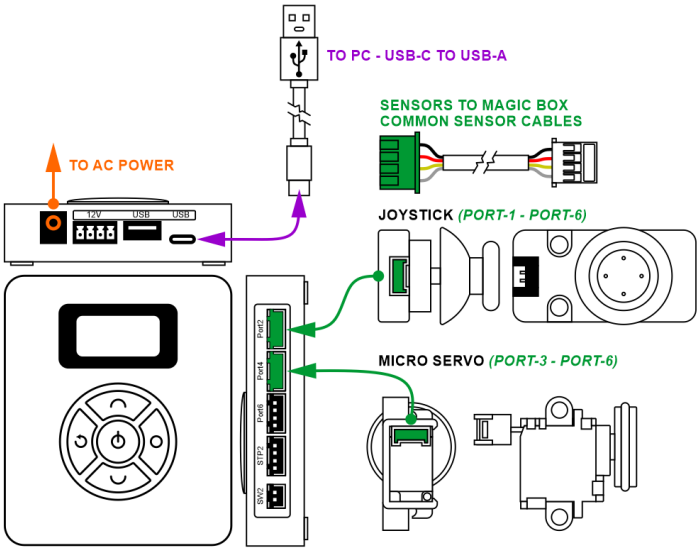
Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



CARFEULLY DISCONNECT ALL OF THE CURRENT INPUTS FROM THE MAGIC BOX

SETUP #2

3. For this part of the activity, you will need:
   1x Magic Box
   1x USB-A to USB-C Cable
   1x AC/DC Power Adapter (12V)
   6x Common Sensor Cables
   1x Joystick – ***PORT 4***
   1x Micro Servo – ***PORT 6***



4. Wire the Magic Box as shown.

SECTION 5 – SETUP #2 - EXPLORING ADVANCED SENSORS & OUTPUTS

- ***JOYSTICK & MICRO SERVO***

SKILL BUILDER 7 – JOYSTICK – ANALOG/DIGITAL INPUT

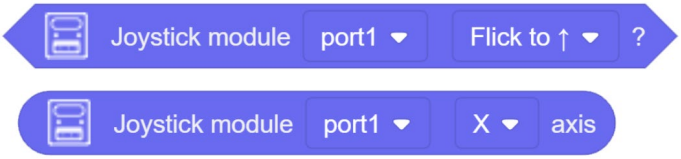
1. The sixth sensor this activity will explore is the Joystick

   The working principle of the joystick is similar to that of a general gamepad joystick. The X and Y axis correspond to two 10K potentiometers respectively. When the joystick moves, it generates a corresponding analog signal and outputs the offset value. The Z-axis acts as a button (momentary digital signal)

ANY GREEN PORT

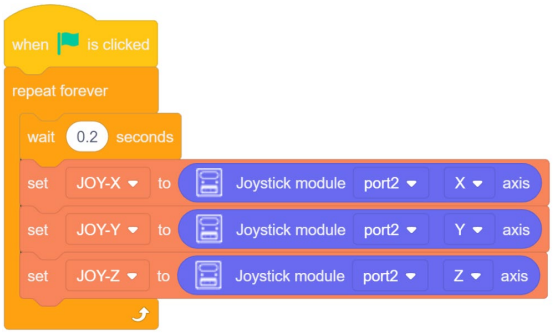
PORT 1-6

2. Drag over the two blocks for the joystick Value from the AI SENSOR KIT toolbox



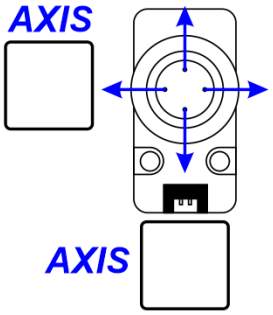
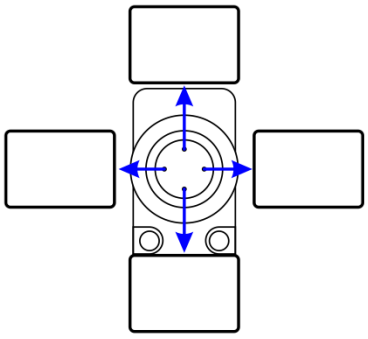
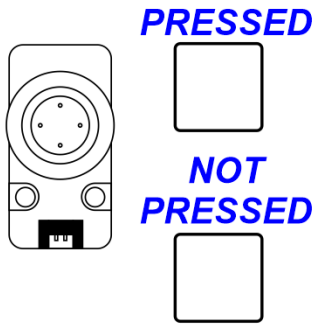
3. Change the PORT values to the correct ports (Green 1-6).
4. Create three new variables (JOY-X), (JOY-Y), (JOY-Z) and place them inside a Set Variable loops as done in previous activities.

5. Download the code to the Magic Box. Practice with this group of code and record your results below.

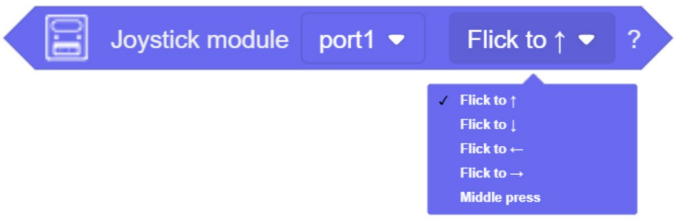


A. Fill in the chart below.

- *Which axis is the Y and which is the Z.*
- *Push the joystick to its four limits & document the extreme values (lowest or highest value reported)*
- *Press down on the joystick, what values are reported when it is pressed and not pressed*

LABEL THE AXIS	LABEL THE EXTREME VALUES	PRESS DOWN Z AXIS
		

8. The conditional block allows a programmer to create statements that can perform various operations when each motion is detected (disregarding the amount of motion).

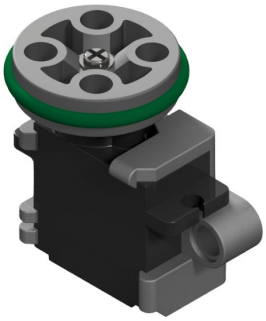


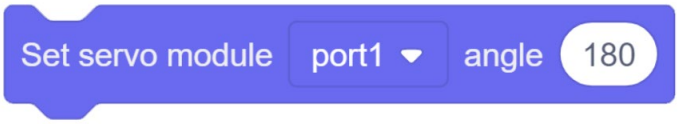
SKILL BUILDER 8 – MICRO SERVO - OUTPUT

1. This device is the only **output** used in these experiments.

   The servo acts as a motor that can be accurately positioned between 0 and 180 degrees. The position can be held indefinitely and is controlled by an internal potentiometer. The servo runs at a constant speed.

PORT 3-6



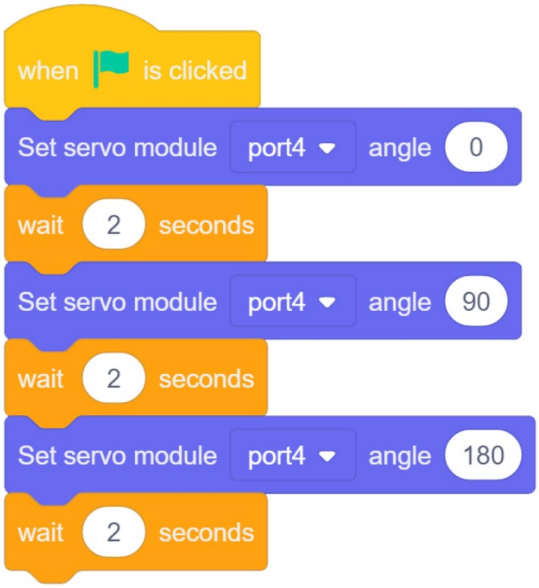


2. Drag over the Set servo angle block from the AI SENSOR KIT toolbox. This block allows the programmer to drive the servo to a specific location and hold that position.

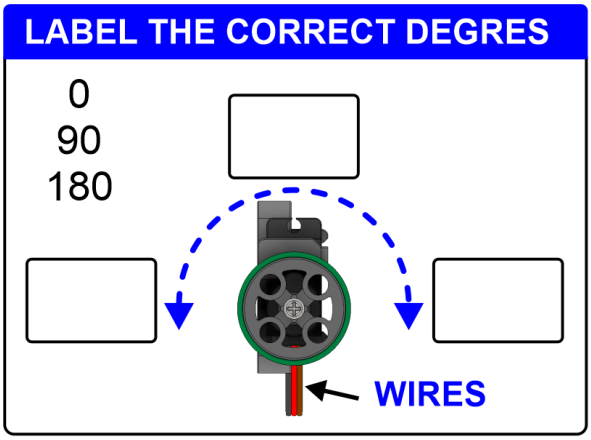
3. Change the PORT values to the correct ports (Green Ports 3-6)

4. Create the simple program to the right

5. Download the code to the Magic Box. Practice with this group of code and record your results below.



A. Hold the servo in the orientation shown to the right. After running the program, fill in the servo positions (0, 90, 180)



CHALLENGE

SECTION 6 – USE the JOYSTICK TO CONTROL THE SERVO

1. Use the joystick to control the position of the servo.

- If the joystick is pushed left, the servo will turn CCW (to the left) to its extreme degree and stay there.
- If the Joystick is pushed right, the servo will turn CW (to the right) to its extreme degree and stay there.
- If the joystick is pressed down, the servo will center itself and stay there.
- This process will repeat forever.

Notes/Observations:

CONCLUSION

Each sensor has its own conclusion questions built in. Answer the questions above for each section and get your instructor's approval.

GOING BEYOND

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

_____ 1. Use the joystick's analog values to control the position of the servo

_____ 2. Scenario: Wave your hand to the right and the robot looks right. Wave your hand to the left and the robot looks left.

Use the gesture sensor to control the position of the servo

_____ 3. Scenario: Nighttime, the windows close. Daytime the windows open.

Use the light sensor to control the servo. If the light is low, the servo will turn to on extreme. If the light is high, the servo will turn to the opposite extreme.

_____ 4. Scenario: Swipe your hand right, and the robot picks up a block from a common position, and places it on a pallet to the right. If you swipe your hand left, it will pick up the block from the common position and place it on the left pallet. The robot will wait for a hand swipe.

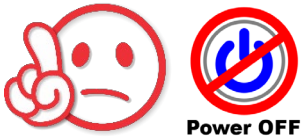
3.3 Block - Exploring the Color Sensor with the Magic Box

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

INTRODUCTION

A **Color Sensor** is an **analog input** device that allows technologists, engineers, and others, to add an additional level of intelligence to their programs written for a robot. These types of sensors can be used for processes that require the need to distinguish between different products, detect data coding, or even determine if a product may have a defect that is distinguished by varying color.

Color sensors can often be found in automation processes that involve textiles, paints, food, printing, and even pharmaceuticals.



Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Color Sensor
- Input
- Magic Box
- Analog

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Color Sensor
- LED Module
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable
- Printed – Color Test Template

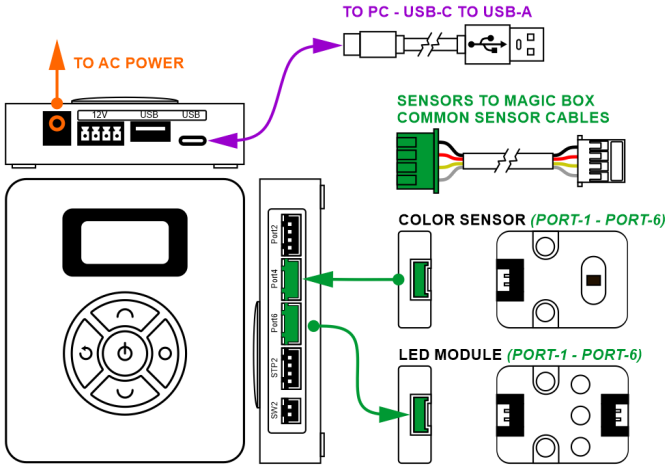
PROCEDURE

SECTION 1 – WIRING the MAGIC BOX AND SENSORS

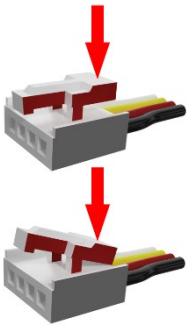
Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
   1x Magic Box
   1x USB-A to USB-C Cable
   1x AC/DC Power Adapter (12V)
   2x Common Sensor Cables
   1x Color Sensor – ***PORT 4***
   1x LED Module – ***PORT 6***

2. Wire the Magic Box as shown to the right

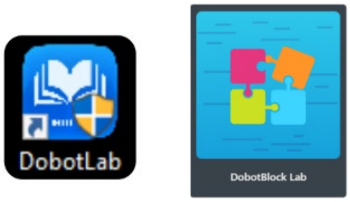


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

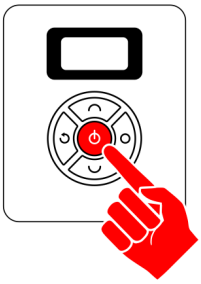


SECTION 2 – CONNECTING the MAGIC BOX to DOBOTLAB

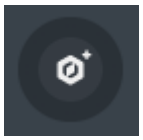
1. Open up DobotBlock Lab in the software.

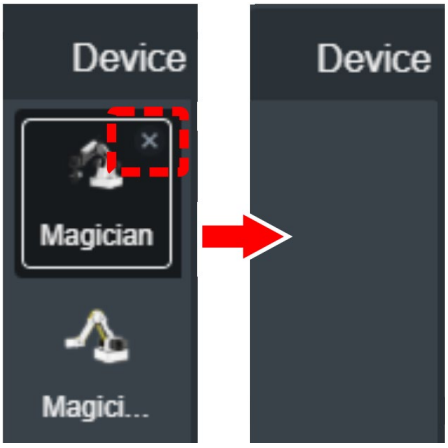


2. Once all of the wiring is done (sensors connected), power ON the Magic Box



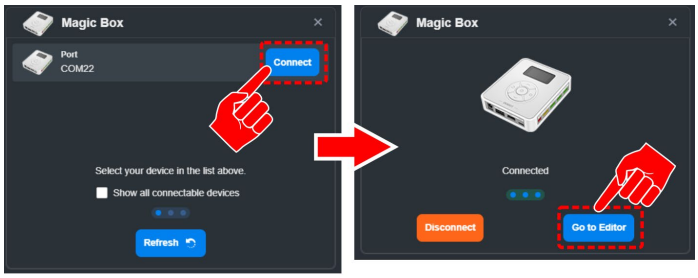
3. **No robot is necessary for this activity.** Delete both the Magician and Magician Lite Devices. Note: This will send you over to the Sprite tab once there are no devices loaded.
4. Click back on the devices tab and click on Choose a Device

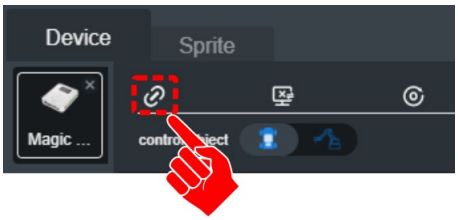




5. Select the Magic Box

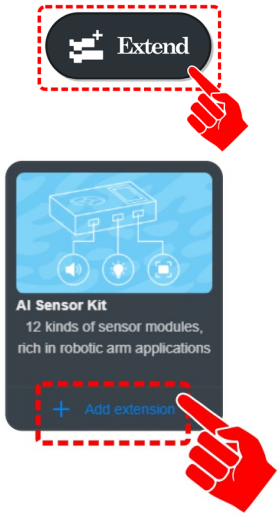
6. Connect to the Device





For this activity, you will need to add an EXTENSION to use the MAGIC BOX. Click the EXTEND icon in the bottom left corner.

Click on "+ Add extension" for the AI SENSOR KIT.



SECTION 3 – EXPLORE COLOR SENSOR

The color sensor can produce two different sets of values.

1. The sensor can identify the color of the object and return a set of RGB values
2. The sensor can identify six preset colors and return a true value if one of these colors are detected.
   BLACK, WHITE, RED, GREEN, BLUE, YELLOW

ANY GREEN PORT

PORT 1-6

SKILL BUILDER 1 – FIND INDIVIDUAL VALUES FOR RED / GREEN / BLUE

1. Drag over a Color Sensor Color Value block from the AI SENSOR KIT toolbox

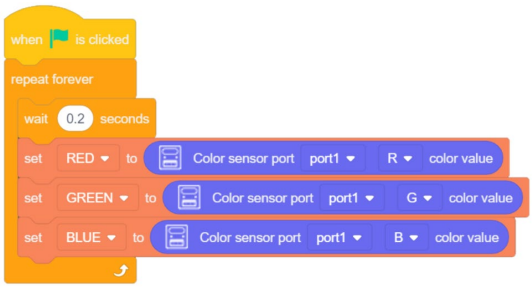
   This block will report a separate numeric value for the amount of R, G, B the sensor sees.

2. Change the PORT value to the port you are using

3. Next, create three separate VARIABLES

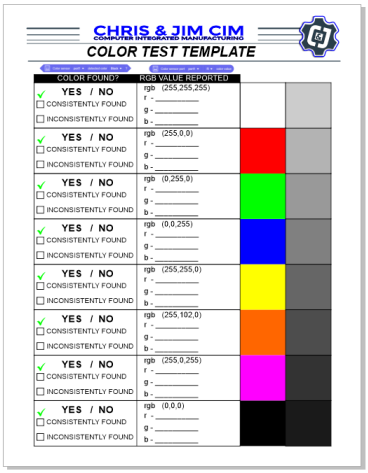
RED / GREEN / BLUE

4. Create the program to the right.



The 0.2 MICROWAIT will slow down the program. It only allows a new sensor values to be pulled (looked at) every 0.2 seconds

5. Print a copy of the Color Test Template



   Run the program and document the values reported to the variables by the sensor.

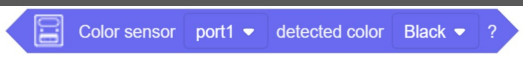
Are the values reported to the variables the same or close to the typical values used in industry for each color?

Is there a pattern to the numbers reported that can be used to alter the variables (a computation / formula) to show a value that better represents the values expected? If so, what formula computation could be used for the variables?

Using the grey scale colors, what type of values are reported when no red, green, or blue values are present?

SKILL BUILDER 2 – DETECT KNOWN COLORS

1. Drag over a Color Sensor Detected Color block from the AI SENSOR KIT toolbox

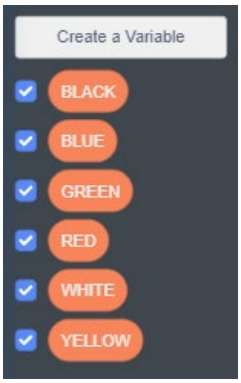


This command will report if a specific color is present.
BLACK, WHITE, RED, GREEN, BLUE, YELLOW

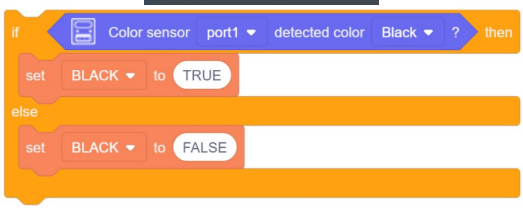
2. Change the PORT value to the port you are using

One way to use this block is to create sets of IF/ELSE statements. If the color is TRUE/PRESENT set the variable to 1, else set it to 0

3. Using the same program from TEST 1, create additional variables for the remaining colors (six total).



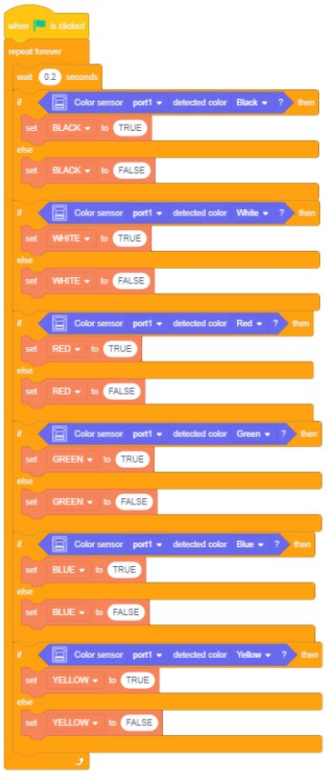
4. Create an IF/ELSE statement for EACH color



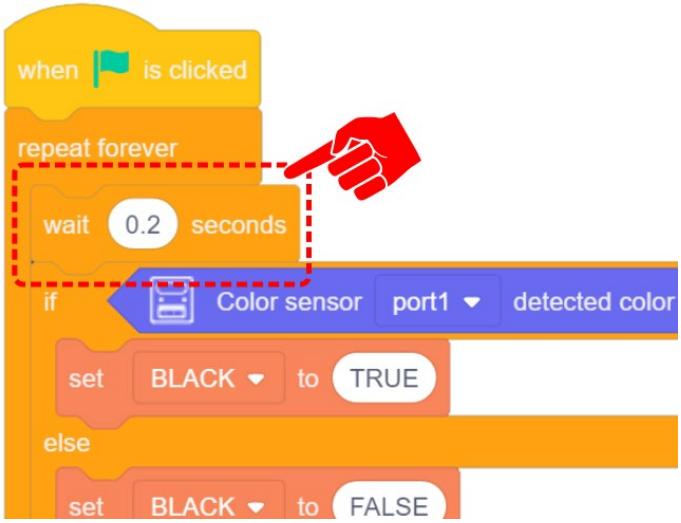
Ensure the PORT number matches the port you are using and that the color is different for each Statement

In previous activities, the SET VARAIBLES have been set to INTERGERS (whole numbers) to represent YES (1) and NO (0). Notice in the example to the right, STRING values (text) can also be used.

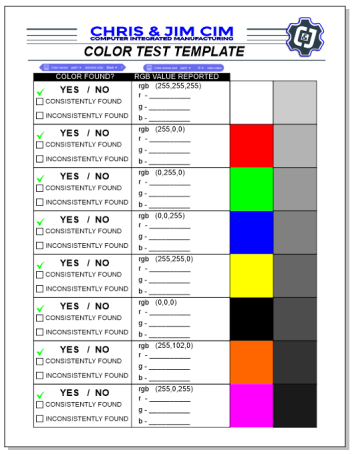
5. Link all of the IF/ELSE statements together inside a FOREVER Block



6. Insert a small 0.2 second MICROWAIT at the begging of the program to slow the cycle down



7. Run the program and document if the colors are seen correctly (circle YES or NO).



Check the "*Consistently Box*" if the correct color is reported by the sensor (all the time or almost all the time)
or
Check the *"Inconsistently Box"* (sometimes) if the value reported by the sensor is often incorrect or sometimes does not read and report the color at all.

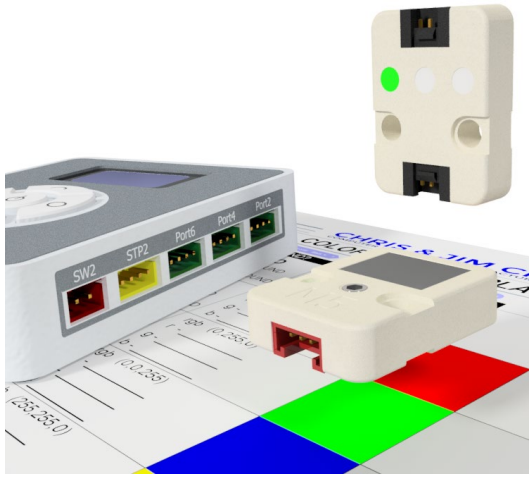
CHALLENGE

SECTION 4 – USE the COLOR SENSOR TO CONTROL THE LED

1. Use the values reported by the color sensor to control the color of LED1

   As the color sensor is held over the Color Test Template, the color of the LED 1 will change color to match the color seen (sensor reports RED, LED becomes RED).

2. As an added level of intelligence to your program, have the variables show the value received as well.



HELPFUL TIPS

- ***The LED will show no color when black is received.***
- ***Do not use the grey scale for this activity.***
- ***Do not use ORANGE and MAGENTA for this activity.***

Continue to test your program until it works as needed.

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

CONCLUSION

1. *How accurate was the color sensor at displaying RGB values?*
2. *How accurate was the color sensor at determining colors from the color test template?*
3. *Would you be able to determine the color of the cubes provided by your instructor using this color sensor? Defend your answer, and then try it.*
4. *Explain how you might be able to use the robot with the color sensor. What tasks will you be able to do with it?*

GOING BEYOND

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

_____ 1. Get some colored cubes from your instructor. Write a program to determine the color of the blocks (hint: this is already done) Test it. How accurate is it at determining block color?

_____ 2. Program the LED to mimic the color of at least 4 different color blocks. Example: (Red Block? Red LED turns on, ect.)

3.4 Block - Using the Color Sensor to Sort

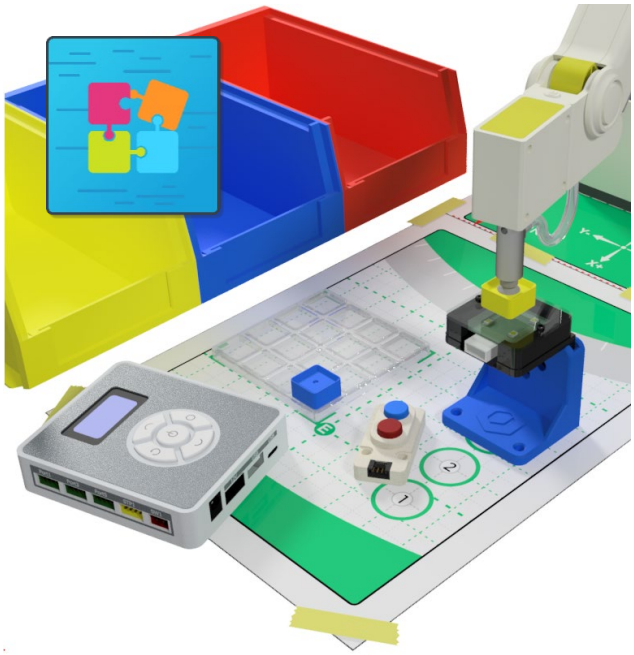
NAME: _____ *Date:* _____ *Section:* _____

INTRODUCTION

Sensors are often added to industrial robots in order for them to perform specific tasks. These sensors can be as simple as a color detecting sensor, or as complex as a full vision system that will allow a robot to be aware of its surroundings or find a part and determine its location and orientation.

In this activity you will learn how to use and program the color sensor in blocky.

The Dobot Lite will pick up a part and move it above the color sensor. The Dobot will then check the part's color and place it in a specific location for that specific color part. The robot will repeat the process each time a limit switch is pressed.



Caution: NEVER wire anything to the Magician Lite or Magic Box while it has power on. ALWAYS shut them down before making connections or damage could occur.

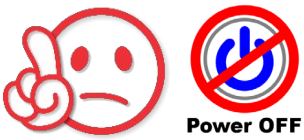
KEY VOCABULARY

- Color Sensor
- If / Else If / Else Statement
- List
- Return True
- MY BLOCKS (Function / Voids)
- Identify Color
- Sum of List

EQUIPMENT & SUPPLIES

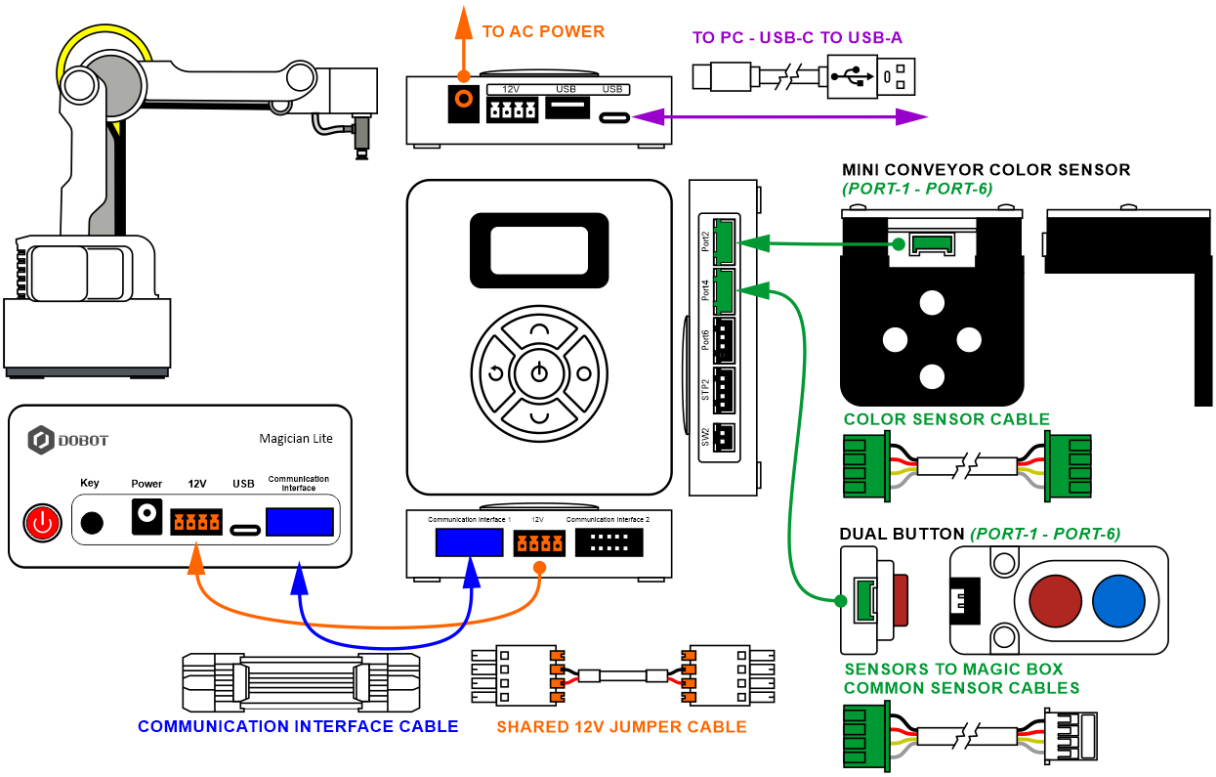
- Dobot Magician Lite
- Magician Lite Mat
- 1" cylinders or 1" cubes (RED, BLUE, GREEN)
- Dobot Mini Conveyor Color Sensor
- Dobot Lab software
- Suction Cup Gripper
- *Dobot Lite Input/Output Guide*
- 3 Small Containers

PROCEDURE

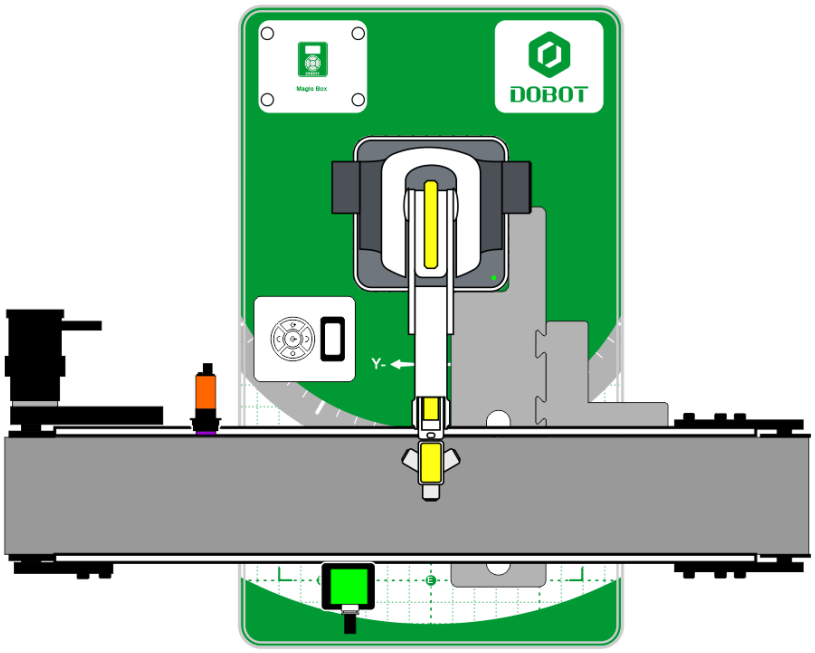


Caution: NEVER wire anything to the Dobot Magician Lite or the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up the robot with a suction cup
2. Wire the MAGIC BOX with the Color Sensor plugged into **PORT 2**
3. Wire the MAGIC BOX with the Dual Button plugged into **PORT 4**



Set up the robot, conveyor, and color sensor as shown in the diagram below:



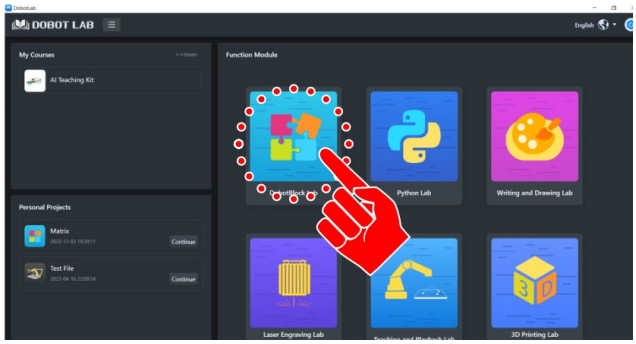
HELPFUL TIPS

The only part of the conveyor that you will be using for this activity is the color sensor. You do not have to connect anything else to the robot at this time. Just the color sensor.

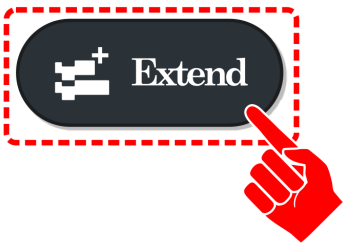
Order of operations

- Start with the robot in a safe/home position.
- Place a single cube, one at a time for the robot to pick up from a common location.
- A limit switch will be used to call for the robot to come and get the block for evaluation.
- After determining its color, drop the cube off at a specific location for that color.
- Manually remove the cube once it has been placed and send the robot to its home position.

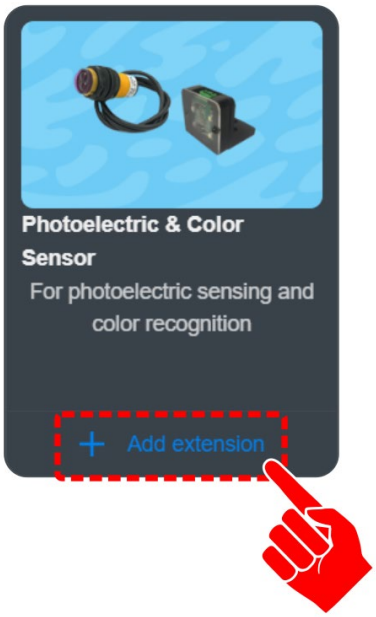
Open a new file for DobotBlock Lab.



For this activity, you need to add an EXTENSION to use the COLOR SENSOR. Click the EXTEND icon in the bottom left corner.



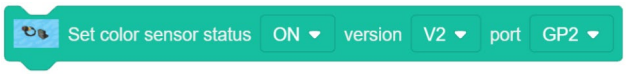
Click on “+ Add extension” for the Photoelectric and Color Sensor



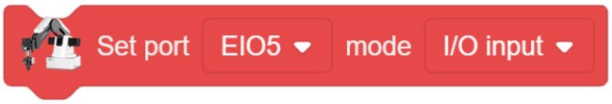
The first step in programming this activity is to set up the inputs. You need to teach the program which sensors we are adding and which port they are plugged into.

Color Sensor Extended Toolbox:

- SET COLOR SENSOR STATUS
  Settings: ON, V2, GP2

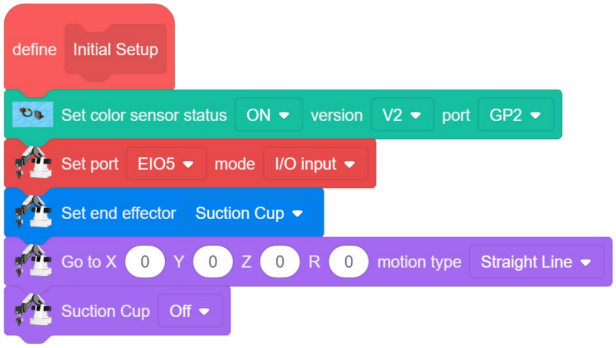


Next, set up the limit switch as we have done in past activities.

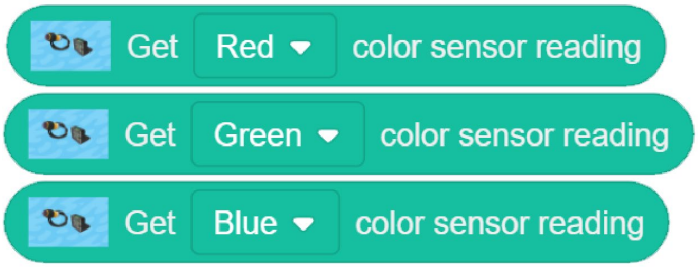


SET PORT GP5 EI05 I/O INPUT

Add these setup blocks to our typical *Initial Setup* MY BLOCK



Now that the INPUTS are setup, the COLOR SENSOR needs to be tested to make sure that three colors are read correctly. The color sensor should report what color it is currently reading.



This sensor is only setup to read RED, GREEN, and BLUE.

(The values for the different colors are preset in the blockly codes so they cannot be seen or changed).

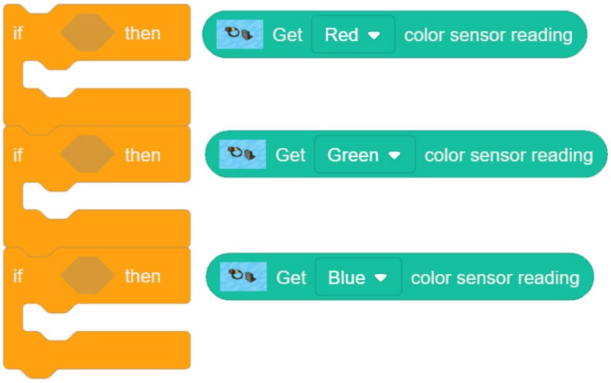
Create a loop that will show the value currently being reported

IF the color is RED

Or IF the color is GREEN

Or IF the color is BLUE

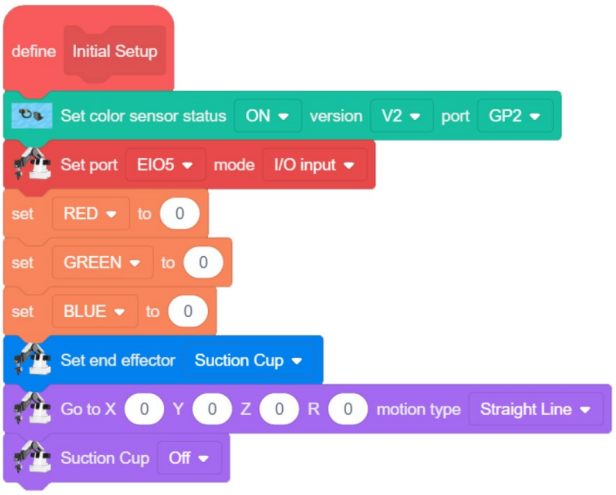
This can be done by creating an ***IF/ IF/ IF Loop***



In order to allow us to see which color is "TRUE", we will create three different variables; one for each color. We will be able to watch the values change in the SPRITE window.

Can you change the sprite to a Magician as shown in the diagram?

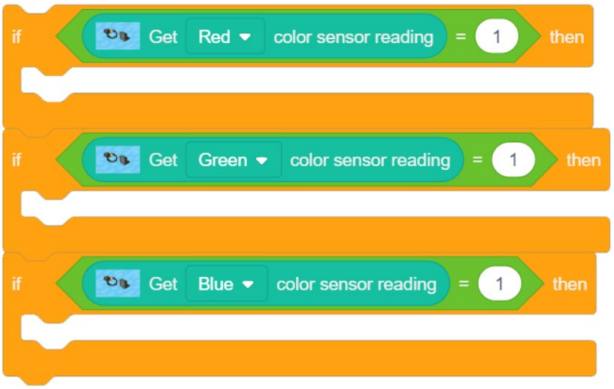
Add these variables to the INITIAL SETUP and set their initial values to zero.



Next create three separate true statements that will go into each IF STATEMENT.

Use the EQUALS condition as has been done in the past and set the SENSOR READING equal to one (1).

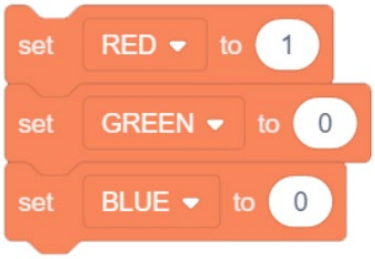
Place the conditions into the IF STATEMENTS.



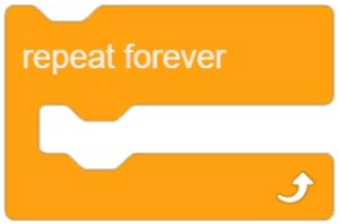
Place code inside each IF STATEMENT that will show us which color the sensor sees. Use the variables we just created to do this.

In order for this to work correctly, turn on one variable and turn off the other two.

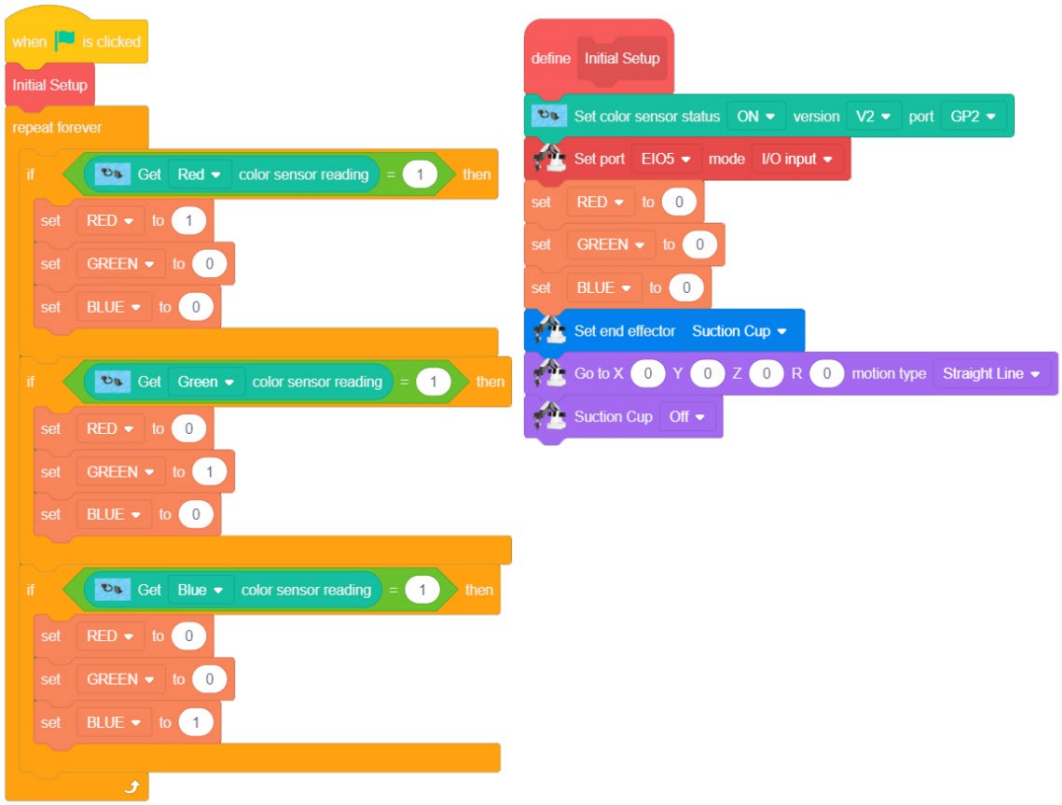
Remember: in the digital world, 1 = ON, and 0 = OFF



Put the IF STATEMENTS in a FOREVER LOOP, otherwise the program will only be checking for color once.



Put together everything and this is is what it looks like



Once the program is completed, run it and see if it works correctly. Every time you put a colored block in front of the sensor, you should see the correct color reported in the running log. If it does not work, troubleshoot it until it does.

Points for discussion:

What does the sensor reads when there is no cube placed on the sensor?

What happens if you put the yellow cube on the sensor?

What type of reading do you get as you raise the cube up and away from the sensor?

Sometimes you will get a random incorrect reading as it changes between colors. Why?

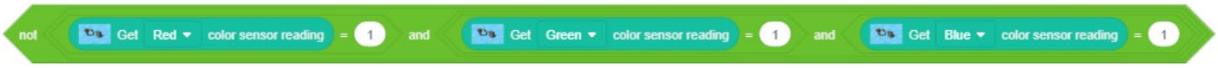
You may notice that the sensor seems to report a certain color, even when there is no cube present.

Is this because of the order of the If statements, and which color comes first? Try changing the order of the IF STATEMENTS

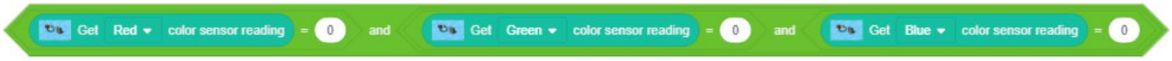
Do you get the same results? Did changing the order effect the readings?

If you noticed there was never any option in the program telling it what to do if none of the values were seen. Try adding one additional IF STATEMENT that turns off the VARAIBLES if none of the colors are reported from the sensors. Something like the codes below:

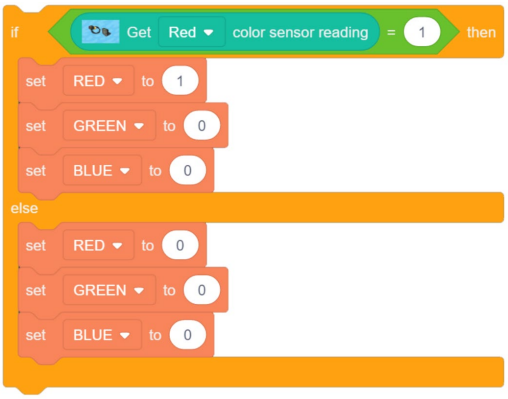
Sensor Readings are NOT TRUE (RED and GREEN and BLUE)



Sensor Readings are FALSE (ALL ZERO) (RED and GREEN and BLUE)



But there is an alternate solution.
If you use three separate IF/ELSE STATEMENTS (Variable on if TRUE, all off if NOT TRUE)… this may not make the program any shorter… if anything, it may even make it longer.



Instead, try nesting a few IF/ELSE statements. The final ELSE will be ALL OFF.

If your setup is like ours… it did not help. It seems to fluctuate a little less often. You could even try to put some micro waits in the program to slow down each cycle and look for the sensor value at a slower pace. The IF/ELSE Statements are better code theory, but since you cannot see the raw data from the sensor you cannot change how it works. As long as you know how the sensor is going to act, this will work as long as the robot does not drop the cube (no cube present when testing the color).

The color sensor works best if the object is held at a consistent distance from the sensor Dobot suggested distance = (0mm to 10mm) or (0 to ⅜ inch)

The different versions of the conveyor kits can come with different shades of colored cubes. The V2 sensor (GREEN COLORED PORT) tends to be more forgiving when it comes to shades of colors.

Now that we have the color sensor reading correctly, you can develop the rest of the program.

In order to keep the program short and organized it will be easiest to use several *MY BLOCKS* (*functions/voids)*.

Use a manual input to start each loop

Process Flow

- Start at HOME
- Get a block from a common location
- Take the block to the color sensor station
- Evaluate the color
- Drop the block off in one of three bins (one for each color)
- Return HOME and wait for a new signal (a new block is in place a ready for evaluation)

MY BLOCK – INITIAL SETUP

Use what we have already developed. Add any additional Variables as needed.

MY BLOCK – PNP

- Create a Pick and Place routine.
  - Wait for INPUT SIGNAL
  - Start with the robot at a **Home** position
  - **Pick and Place** - Get the block and put it on (near) the color sensor.
  - ADD a 1 second wait at the end of the PnP routine before looking for the color (allow the reading to become a solid reading)

MY BLOCK – EVALUATE

- Evaluate the COLOR

Use the IF / IF / IF Statements we used earlier

There are so many different creative ways you can proceed with dropping the cube off in their correct colored bins once the color is known. Using what you have already learned, implement one that works for you.

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

Once completed, run the program and see if it works correctly.

If it does not work, troubleshoot it until it does. If there are issues with the colors, try using different blocks of the same color. Does this make a difference?

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

CONCLUSION

1. *What happens if a block isn't there when the color sensor is told to get a color with the current program? Give a reason why.*

2. *How might you keep track of how many blocks there are of each color?*

GOING BEYOND

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

_____ 1. Randomly arrange cubes in a 3x3 palletized matrix for the robot to pick from. After determining its color, drop the cube off at a specific color location. Manually remove the cube once it has been placed.

_____ 2. Have the robot stack the cubes in columns by color.

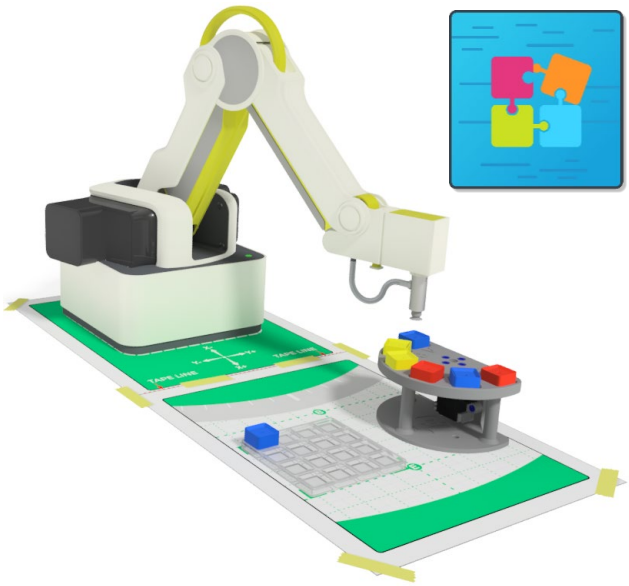
_____ 3. Color the 9 squares on the Dobot field diagram either red, blue, or green. You will then write a program that takes cubes from a cube matrix and puts the respective color cubes on the colored squares.

3.5 Block - Using a Servo Driven Part Feeder

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

INTRODUCTION

In industry efficiency rules all and is one of the main reasons to use robots. Robots and the automation that comes with them saves time, energy, and money, which are all resources important to manufacturing. In industry parts feeders are run mainly by electricity or pneumatics or a combination of both.



Imagine having a single point from which a robot can pick up multiple items time and time again. You only have to teach it one "pick" position, and because of this, it will be much more accurate, as well as easier to program.

With a **Part Feeder**, you can accomplish this. Even more important, is the fact that you can make the robot communicate with the Feeder, so it only puts a part in position when you are ready for it. In this activity you have the opportunity to build a 3D printed parts feeder to use with the **servo** in the Magician Lite Sensor Kit, and then program it to work with the robot.

Caution: NEVER wire anything to the Magician Lite or Magic Box while it has power on. ALWAYS shut them down before making connections or damage could occur.

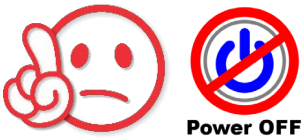
KEY VOCABULARY

- Automation
- Servo
- 3D Print
- Efficiency
- Accuracy
- Part Feeder

EQUIPMENT & SUPPLIES

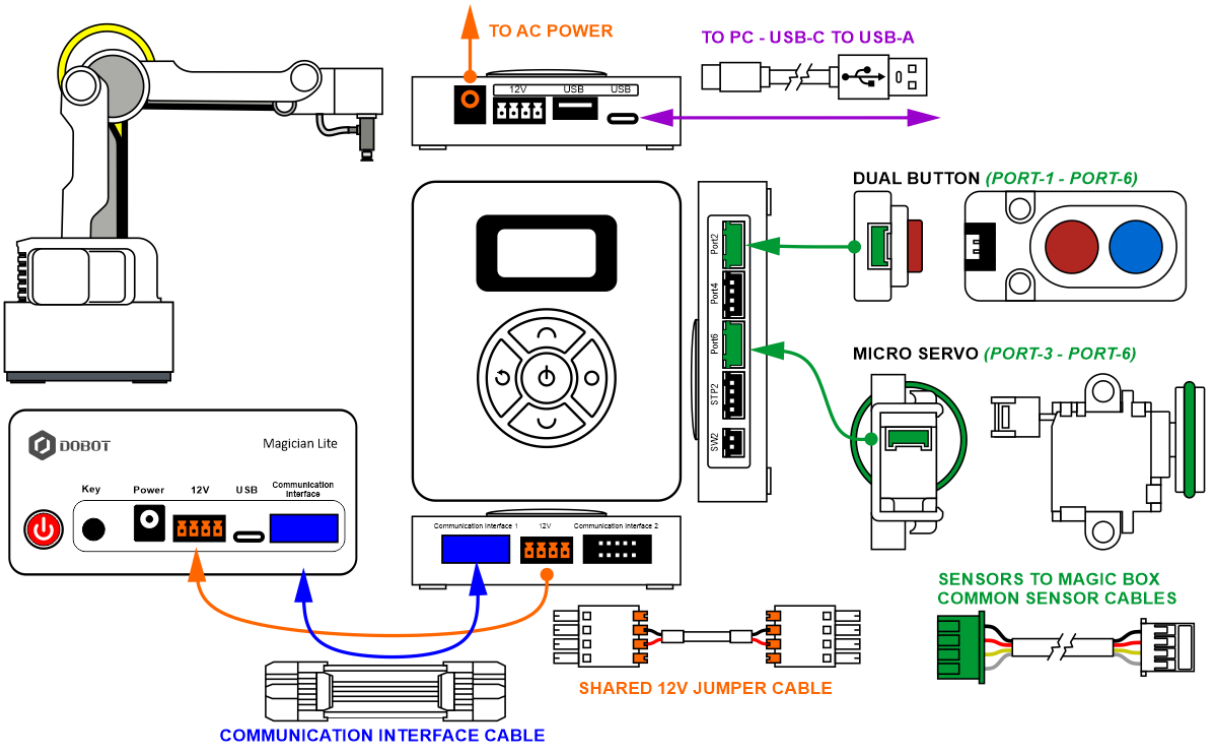
- Dobot Magician Lite
- Magician Lite Printed Template
- 1” cylinders or 1” cubes and Template
- Magic Box
- Magic Lite Sensor Kit - Servo
- Dual Button Module
- Suction Cup Gripper
- 3D Printed Servo Base
- 3D Printed Servo Disc Top
- Lego Technic Pins (3D printed)
- Double sided tape

PROCEDURE

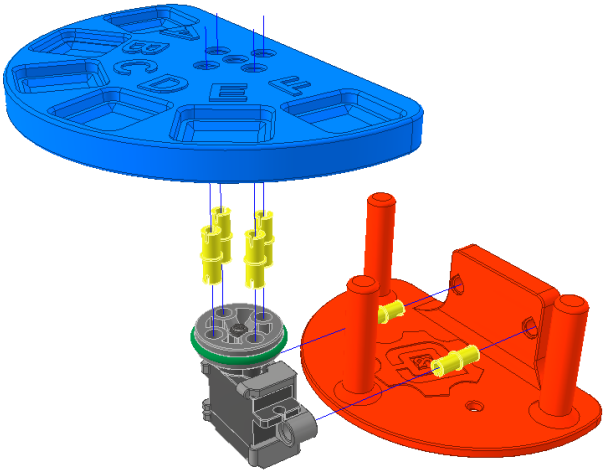


Caution: NEVER wire anything to the Dobot Magician Lite while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up the robot with a suction cup
2. Wire the MAGIC BOX with the Dual Button plugged into **PORT 2**
3. Wire the MAGIC BOX with the Micro Servo plugged into **PORT 6**

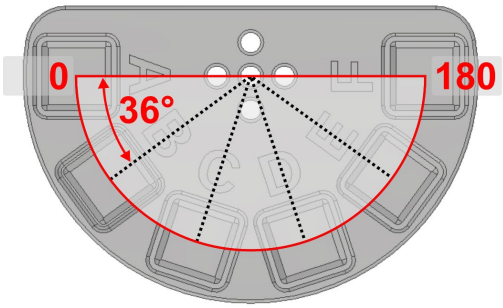


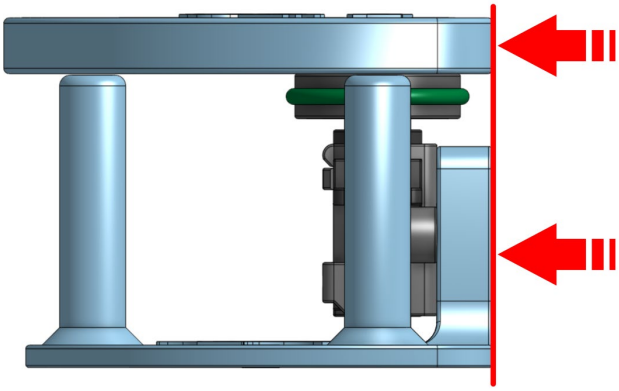
4. *Download the STL files for the Magician Lite Part Feeder here: [Click here for link to the folder and print them with your 3D printer.]*



Alternatively, design one on your own, or make this one better, reprint, test it and share your ideas with the class.

5. This feeder was designed to have six parts placed into it, and when you program the servo to move to 0 the top and bottom plates should be aligned.

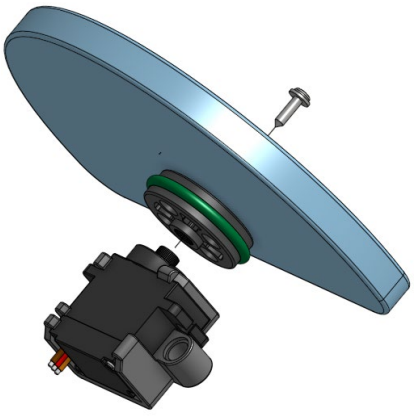


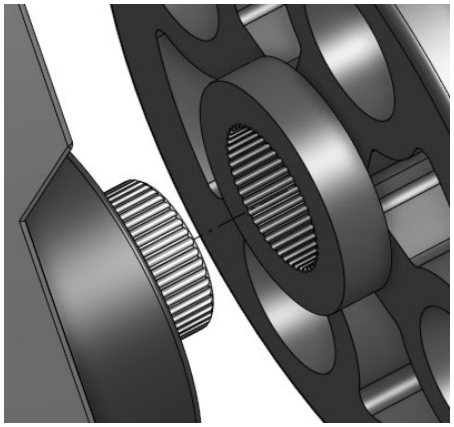


If you write a short program to make the servo move to angle 0, you can then complete the following steps to be sure the feeder is aligned.

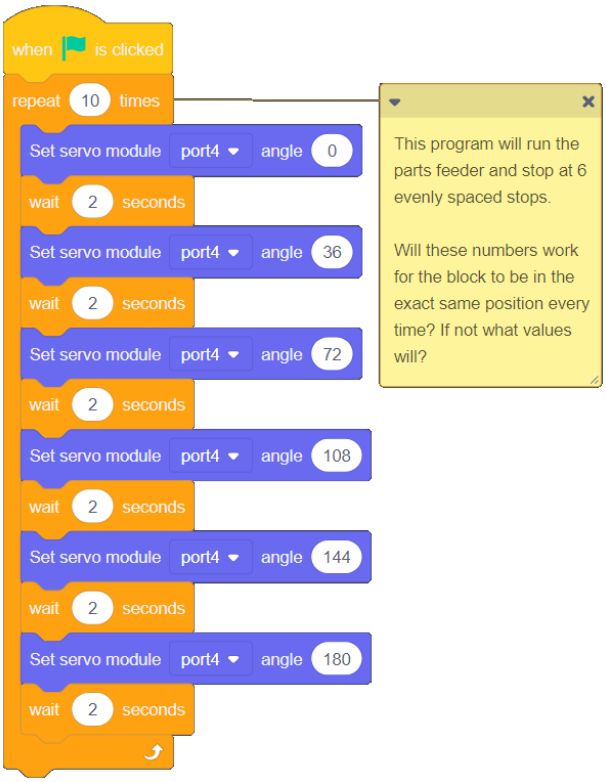
HELPFUL TIPS

1. Write the program to make the servo move to angle 0.
2. Run the program.
3. Carefully remove the top of the feeder from the servo.
4. Align the two parts as shown above.
5. Push the top of the feeder back on.





6. This simple program will make the feeder move through 180 degrees with six stops. Notice there are six blocks on the feeder.



If the math is right, the servo should put the block in the same "pick" position every time.

Try this program and see how it works.

Do these values work? If not, what do they have to be changed to so that the block is in the exact same position each time?

Once you have determined the stops, write them below:

Feeder Position	Angle
A	_____
B	_____
C	_____
D	_____
E	_____
F	_____

Use the information above as a starting point to learn how the feeder works. Once these positions are known, you should be able to write the rest of the program with what you have learned from other activities.

7. Set up the robot, feeder, and pallet similar to what is shown in the diagram. Be sure that the feeder and the pallet are within the work envelope of the robot.

Use double sided tape to hold the feeder and pallet in place so that they are less likely to move throughout the activity.

8. Using what you have learned in previous activities, develop a program that will make the following happen:

- Load the feeder with cubes.
- Press the button to make the operation start.
- The robot should pick a part off the feeder and place it on the pallet, filling two rows or two columns.
- The feeder should move the next part into position.
- Repeat until the feeder is empty and there are six parts on the pallet in two columns or rows of three.
- Robot goes to a home position, and the feeder resets to the first position.

Is there a better way to add the parts to the pallet, using math, instead of teaching it so many points? Look at activity 1.7 Developing a cube matrix in the Magician curriculum found here: https://chrisandjimcim.com/v3-dobot-magician-curriculum-done/

DEBRIEF

In the space below outline what worked well for you, and what you would change if you didi it again. Be sure to explain why!

NOTES:

CONCLUSION

1. *What happens if a block isn't there on the feeder when there is supposed to be?*

2. *How does a part feeder provide an advantage over picking from a pallet?*

3. *Did you have any problems printing the feeder? What settings did you use to make it on your printer?*

4. *Can you think of any way to make this 3D printed feeder better? How? Do it!*

5. *Place a screen capture of your program below or show it to your instructor to get full credit for the project.*

GOING BEYOND

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

_____ 1. Use an LED to let an operator know that the feeder is empty.

_____ 2. Look at activity 1.7 Developing a cube matrix in the Magician curriculum found here: https://chrisandjimcim.com/v3-dobot-magician-curriculum-done/ and make the robot use this method to add the blocks to the pallet

_____ 3. Use the color sensor to sort the blocks by color on the pallet when done.

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING
MAGICIAN LITE

- CHAPTER 4 -
ROBOTICS SCENARIOS & PERIPHERALS

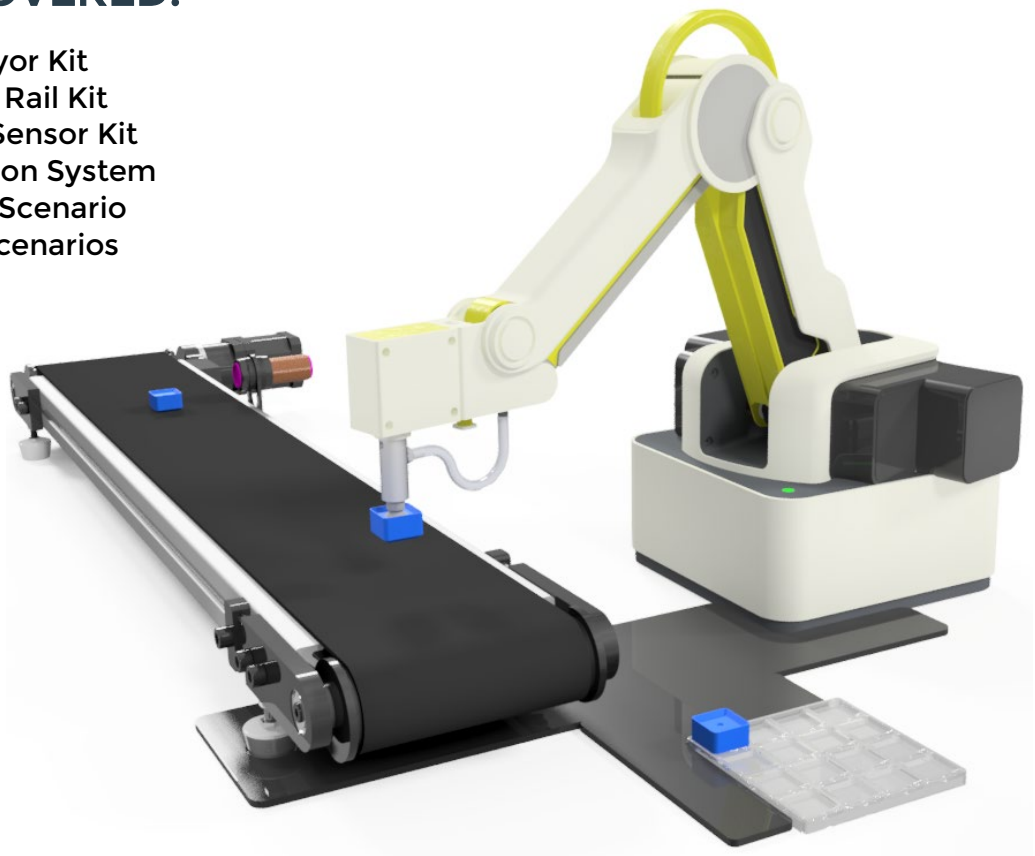
CHAPTER INTRODUCTION:

In this chapter, you will explore how to extend the usefulness of a robotic arm in an industrial setting by adding different peripherals. Imagine if a robot could move further, its parts could be automatically brought to it, or it had a camera to be able to identify different parts. This can be easy to do, and you will get the opportunity to these capabilities in these activities.

This chapter will introduce you to the use, wiring, and programming of these peripherals and provide you the chance to model a wider variety of industrial operations and scenarios.

TOPICS COVERED:

- Mini Conveyor Kit
- Linear Slide Rail Kit
- Magic Box Sensor Kit
- Camera Vision System
- Stack Light Scenario
- Work Cell Scenarios



4.1 Block - Starting & Stopping a Conveyor

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

INTRODUCTION

Robotic arms need to communicate with each other as well as other **peripherals** such as **conveyors** or **linear rails** in order to move materials or products through stages of a work cell. The robot is used to control all of these peripherals, making it the "Boss" of the work cell.

In this activity you will learn how to program a robotic arm to control a conveyor belt and use an Infrared Sensor to create a **closed loop system**. The sensor can be used to start or stop the conveyor in the same spot every time. This increases the accuracy of where the part is placed.

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

KEY VOCABULARY

- Stepper Motor
- Conveyor
- Linear rail
- Output
- IR- Infrared Sensor
- Closed loop system

EQUIPMENT & SUPPLIES

- Dobot Magician Lite
- Small cubes & pallets
- Dobot Mini Conveyor
- Magic Box
- IR Sensor
- Suction Cup Gripper
- DobotLab software

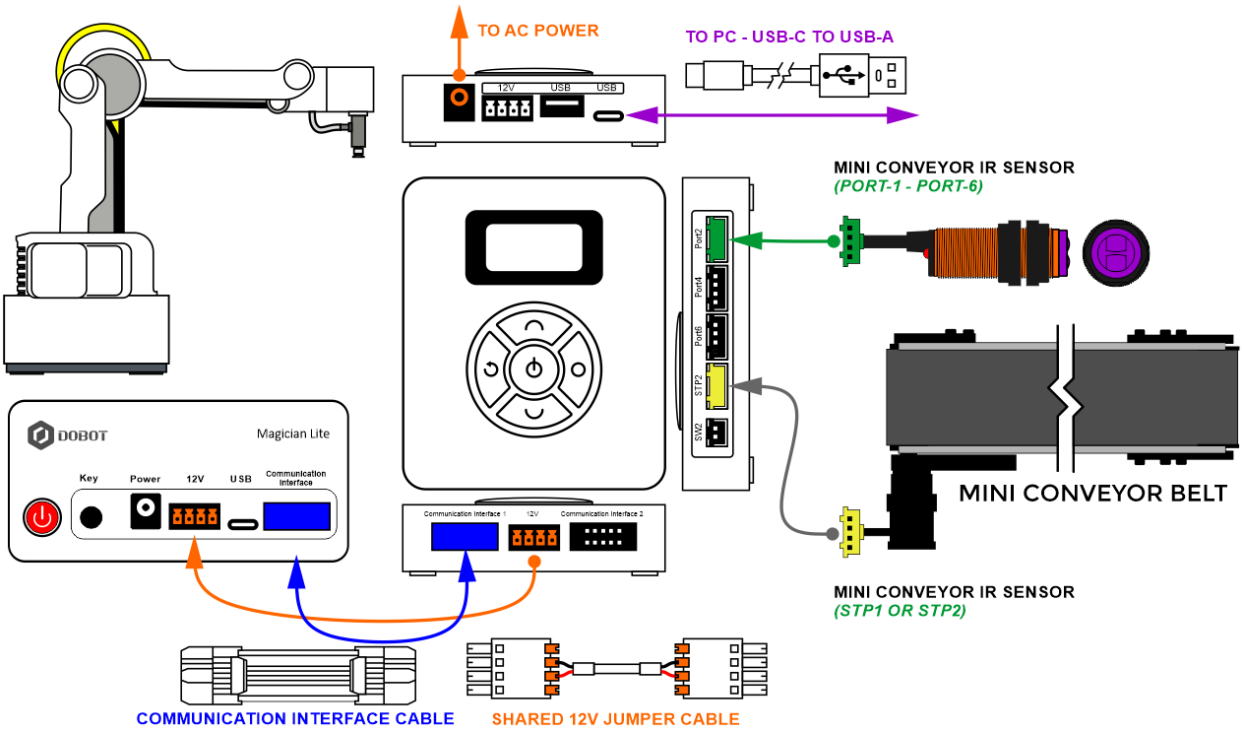
PROCEDURE

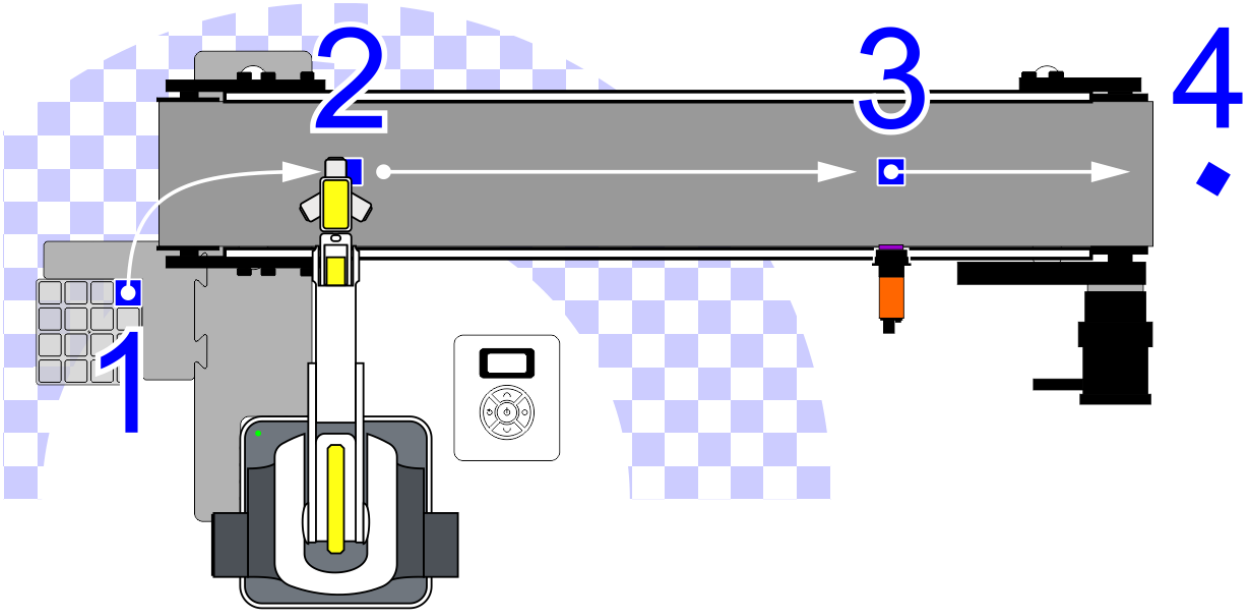
Order of operations

- The Robot will pick up a cube from a known location and place it on the conveyor belt.
- The Robot will return home and then start the conveyor belt.
- The conveyor belt will run until the block arrives at the IR Sensor for inspection.
- The block will be manually removed from the belt, inspected and then returned to the belt.
- Once the block is returned to the belt, the belt will run again until the parts runs off the belt and into storage.
- The process will loop forever.

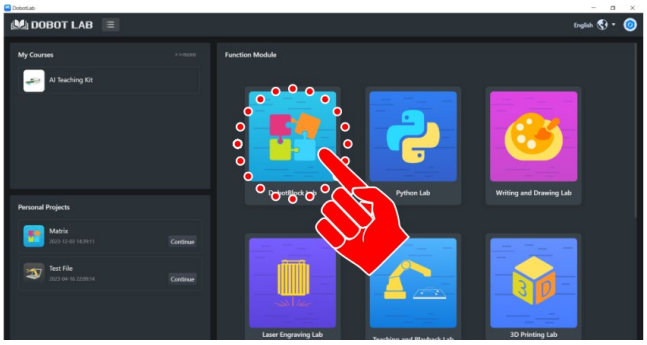
Caution: NEVER wire anything to the Dobot Magician or the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up the robot with the suction cup gripper.
2. Plug the Conveyor Belt into **STEPPER2** of the MAGIC BOX and plug the IR sensor into **PORT2** of the MAGIC BOX

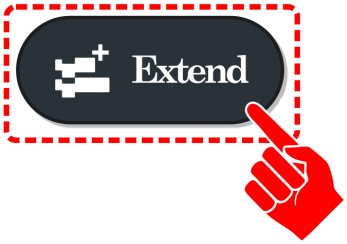




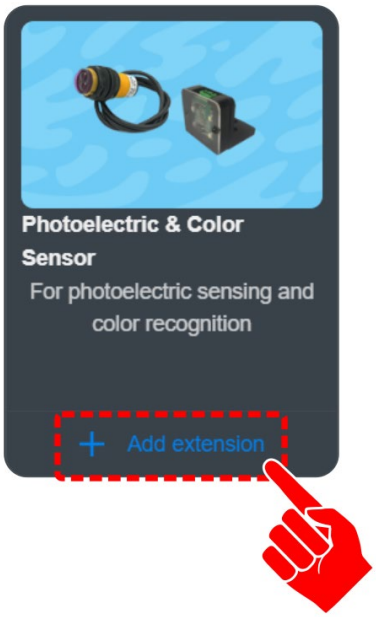
Open a new file for DobotBlock Lab



For this activity, we will need to add an EXTENSION to use the PHOTOELECTRIC SENSOR (IR Sensor). Click the EXTEND icon in the bottom left corner.



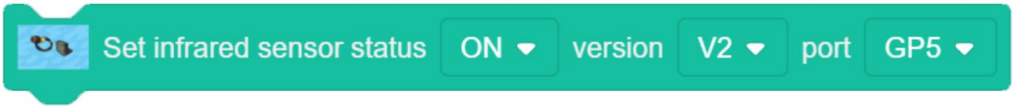
Click on “+ Add extension” for the Photoelectric and Color Sensor



We will also need additional new blocks for this activity:

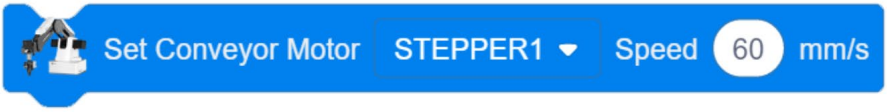
Photoelectric Sensor Extended Toolbox:

- **SET INFRARED SENSOR STATUS**
  Settings: ON, V2, GP5



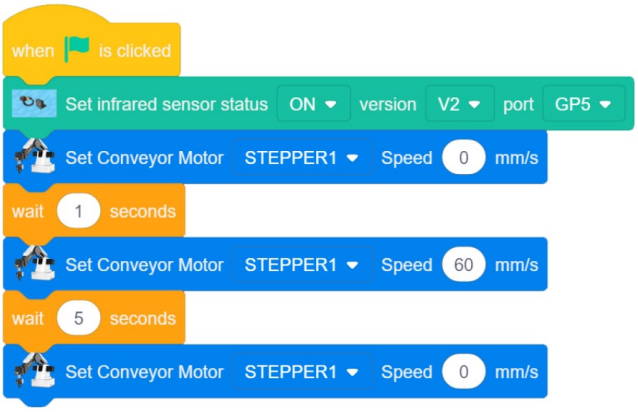
Settings Toolbox:

- **SET CONVEYOR MOTOR**
  Settings: STEPPER1, 60 mm/s



Write a quick code that will turn on the conveyor for 5 seconds

In this example, 60 mm/s is used for the speed of the conveyor. Play with this value until you have a controllable speed for this activity.

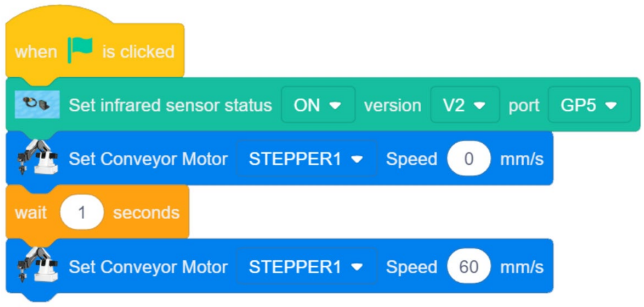


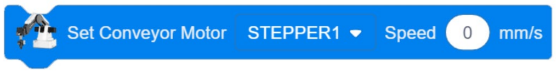
Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

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

Now edit the current code to get the conveyor belt to stop when an object is detected by the IR Sensor.

Remove the 5 second ***DelayTime*** from the program.

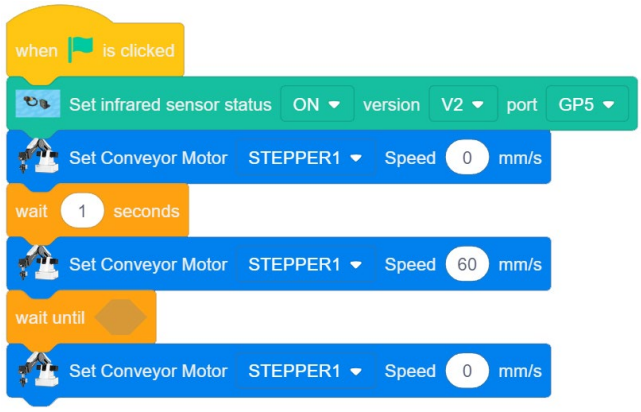




Change this OPEN loop system to a CLOSED loop system by using the sensor to add feedback.

To create a loop that will keep checking the IR Sensor value until an object is detected, we are going to replace the *wait for time* with a *WAIT UNTIL*.

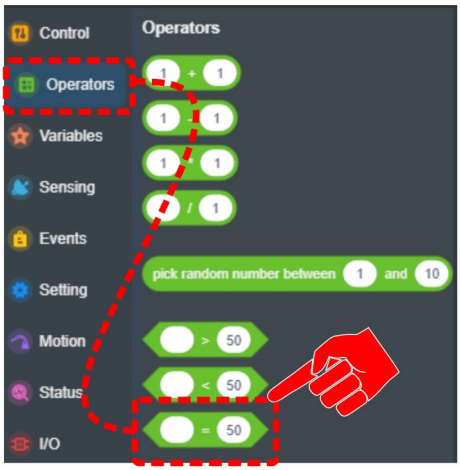
Insert a ***WAIT UNTIL*** loop and insert it where the wait for time block was.



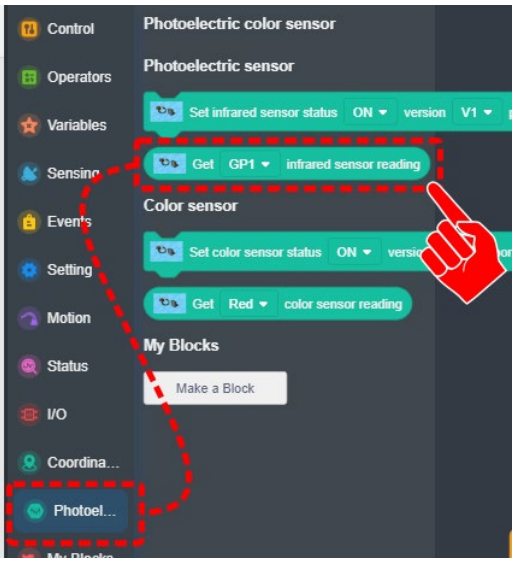
Create a ***condition*** that evaluates the IR sensors value. We are looking for the sensors value to change from 0 to 1.

The IR Sensor reads true/high/1 when an object is present. The small LED on the back of the IR Sensor will also light up.

Insert the ***EQUALS*** condition from the Operators Toolbox.

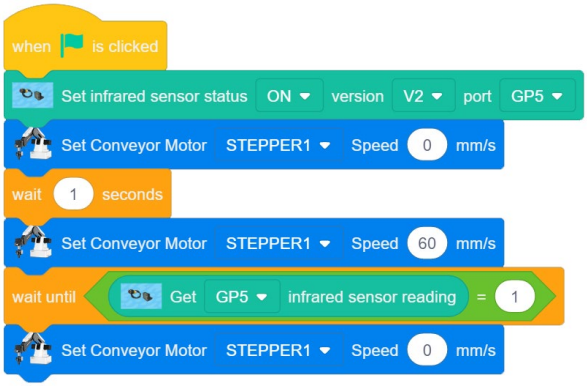


Insert the ***GET INFRARED SENSOR READING*** from the Photoelectric Color Sensor Extended Toolbox.



Add both blocks together to complete the statement as shown.

Make sure to change the ***GET INFRARED SENSOR READING*** value from GP1 to GP5 and the number value to 1



Run the program. Let the conveyor run for a few seconds, then manual place a cube in front of the sensor. The conveyor should stop.

If it does not work, troubleshoot it until it does.

What are all of the things that could possibly be wrong if it doesn't work? Look at all of the variables in the blue and the green blocks in the program above.

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

Now that the conveyor starts and stops using the IR Sensor, you can find positions, set any needed VARIABLES, and create MY BLOCKS and set variables for this activity.

Create a Pick and Place routine.

- Start with the **robot at a Home** position
- **Pick and Place** - Get the block and put it on the conveyor and return to home position.

Next, start the conveyor until the block reaches the other end.

- **Start the belt** and run it **until the cube reaches the sensor**
- **Stop** the belt

Create a variable named “waiting”. Set this variable to one (1) when the robot is waiting and zero(0) when it is not.

Once the program is completed, 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?

Reminder:

This activity requires the following operations:

- The Robot will pick up a cube from a known location and place it on the conveyor belt.
- The Robot will return home and then start the conveyor belt.
- The conveyor belt will run until the block arrives at the IR Sensor for inspection.
- The block will be manually removed from the belt, inspected, and then returned to the belt.
- Once the block is returned to the belt, the belt will run again until the part runs off the belt and into storage.
- The process will loop forever.

Now that the conveyor starts and stops, the next step is to create a loop that will allow the block to be removed and then wait for it to be returned. The issue is that we have no idea when the block has been removed, how long it will take for the inspection, or when it will be returned.

We need to create a closed loop system that will look for the following conditions without respect to time.

The Block has been REMOVED (Sensor Feedback)
The Block has been RETURNED (Sensor Feedback)

*Some of this program can be condensed into separate **MY BLOCKS** (functions) in order to simplify the main program. This makes the program much more efficient and easier to troubleshoot.*

The final task is to start the conveyor again and run the part off the end and into a container. Use the same code you started with to accomplish this task. The time needed to run the part off the conveyor will be different for each project, depending on where the sensor is mounted.

SAFETY: Place a small ***WAIT*** in front of this section of code to allow time for the user to get their hand away from the belt before it starts running.

Once the program is written, try it and make sure that it works as expected. If it does not work correctly, troubleshoot until it does.

Remember to check all the variables in the blocks first like:

- Have you used the right ports?
- Are the speeds correct?
- Are 1's and 0's used correctly?
- Are the MyBlocks called out correctly?

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

CONCLUSION

1. *Why is it better to have the infrared sensor stopping the conveyor belt rather than just running it for time?*

2. *How would the program be different if the conveyor belt could not be run as a linear rail?*

3. *What's one way to determine where the robot is at any given time in the program?*

GOING BEYOND

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

_____ 1. Use functions to make your program as short as possible.

_____ 2. Add the AI extension as done in previous activities and use the color sensor and make the robot report what color block is being sent down the conveyor using voice.

4.2 Block - PnP with Linear Slide Rail

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

INTRODUCTION

When the workspace of Dobot Magician is not enough, you can extend it with a **sliding rail**. This peripheral allows you to mount the robot on a rail, and precisely control its position along the rail. The robot controls its position on the rail by use of a **stepper motor**.

This device can be useful, especially in an industrial setting. It allows you to do long distance pick and place and palletization routines for one, sometimes making it possible for one robot to do the work of two or more, saving money and time.

It is also very useful when **Machine tending**. A robot can put raw materials into a machine and remove finished product allowing for **Lights Out Manufacturing**.

Caution: NEVER wire anything to the Magician or the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Sliding Rail
- Stepper Motor
- Machine Tending
- Lights out manufacturing

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Analog – Potentiometer Module
- 12V Power Cable and Supply
- Dobot Magician Lite
- Suction Cup Gripper
- Module Connecting Wires
- USB C to A Cable

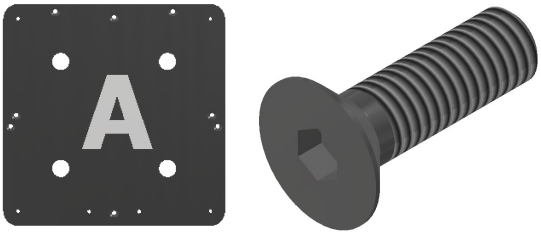
PROCEDURE

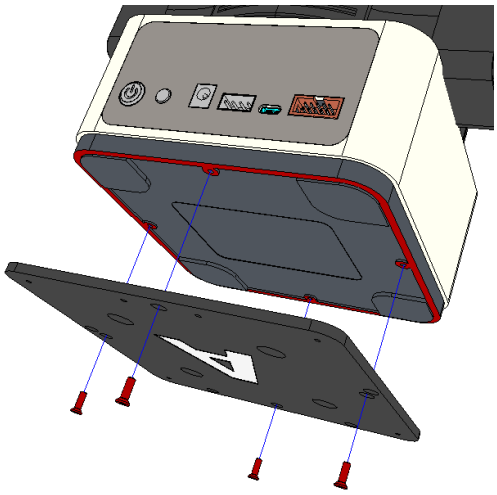
SECTION 1 – INSTALLING THE ROBOT ON THE LINEAR SLIDE RAIL

If this section is already complete, skip to SECTION 2 – WIRING.

HELPFUL TIPS

Step 1 – Mount the Dobot Magician on **PLATE A** with **FOUR** M3x10 countersink cap screws. Ensure the countersinks of **PLATE A** point outward. Orient the plate as shown in the image to the right.

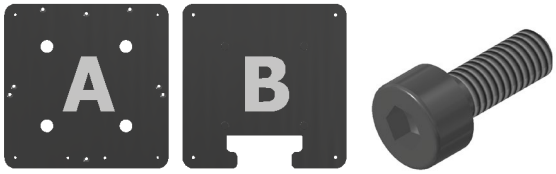


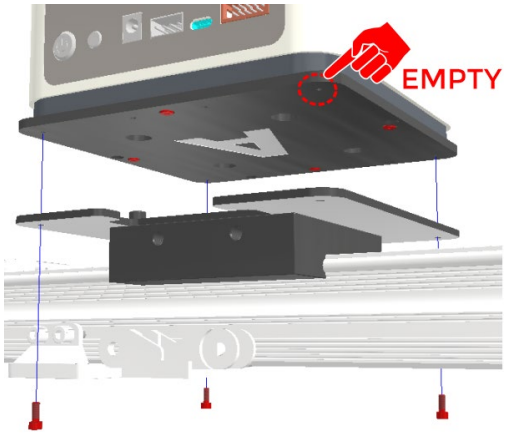


STEP 2 – Mount **PLATE A** (with the Dobot Magician attached) onto **PLATE B** with **THREE M3x8** Hex-Socket Head Cap Screws. Make sure the back of the robot faces the cut out on the back of **PLATE B.**

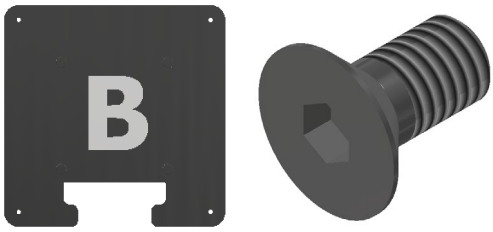
HELPFUL TIPS

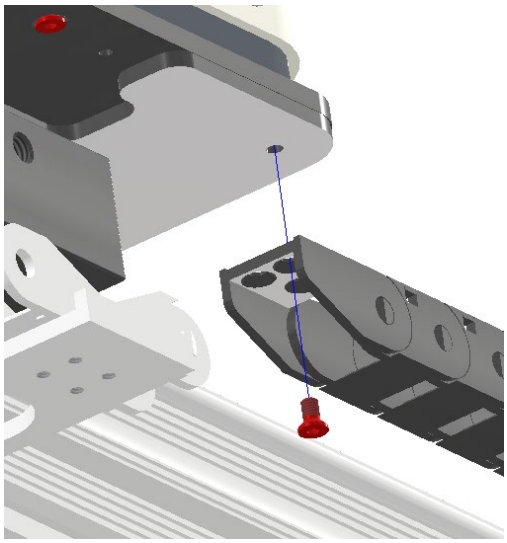
*Do not put a screw in the back right corner (**EMPTY**). We need this hole for the next step*



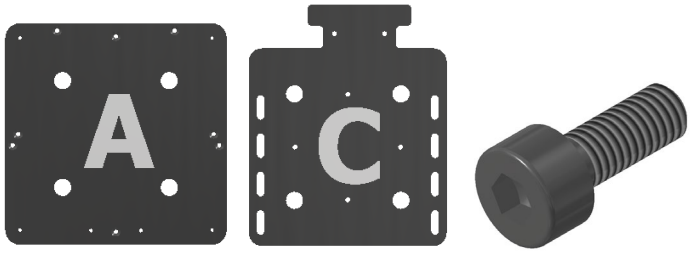


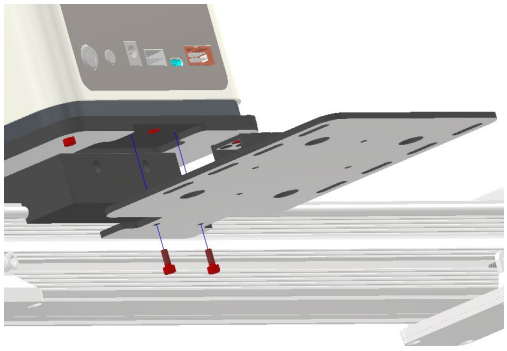
STEP 3 – Connect the free end of the **WIRE DUCT** to **PLATE B** with **ONE M3x6** Countersink screw.



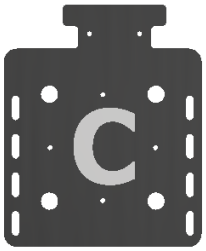


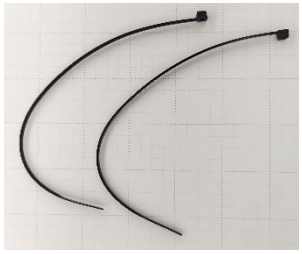
STEP 4 – For this activity, in order for use with the Magic Box we will need **PLATE C** to carry it. Attach **PLATE C** onto **PLATE A** with **TWO M3x8** Hex-Socket Head Cap Screws**.**

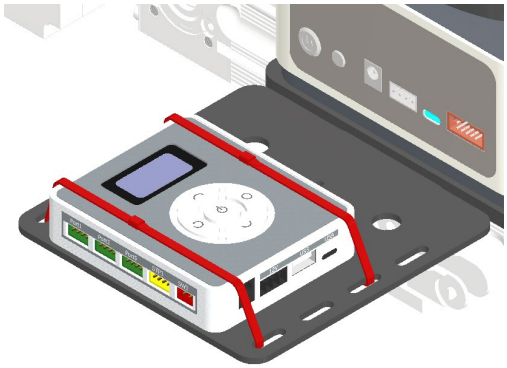


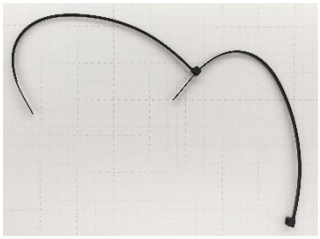


STEP 5 – Use the included **ZIP-TIES** to attach the **VACUUM PUMP** to **PLATE C**. It may be necessary to link/chain more than one zip tie in order to make a complete loop around the Magic Box and bracket.









SECTION 2 – WIRING THE ROBOT & LINEAR SLIDE RAIL

If this section is already complete, skip to SECTION 3 – SETUP & PROGRAMMING.

HELPFUL TIPS

STEP 1 – Connect the four wires from the linear slide rail to the back of the MAGIC BOX. The wires should reach from the wire duct to the MAGIC BOX.

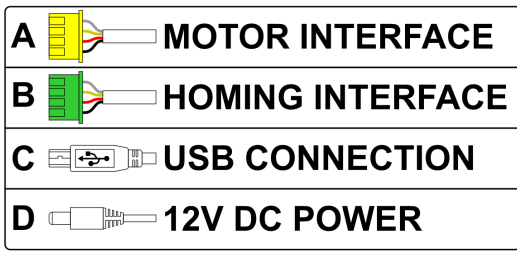
GREEN – PORT 4	USB-B to **CONVERTER** (Convert to USB-C) to Magic Box
YELLOW – STP 2	12V SHARED POWER to Magic Box

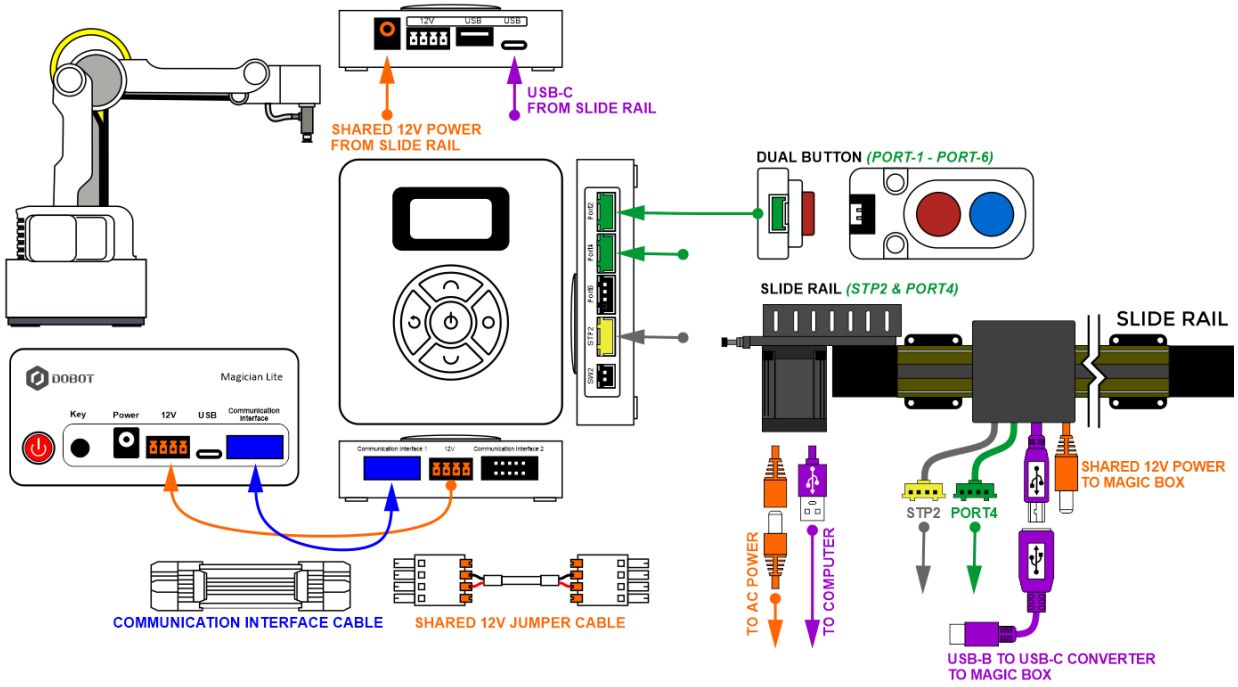
(A) – The motor interface controls the direction and location of the stepper motor

(B) The homing cable is attached to the limit switch at the end of the conveyor and reports when the slide base has been homed to a zero location.

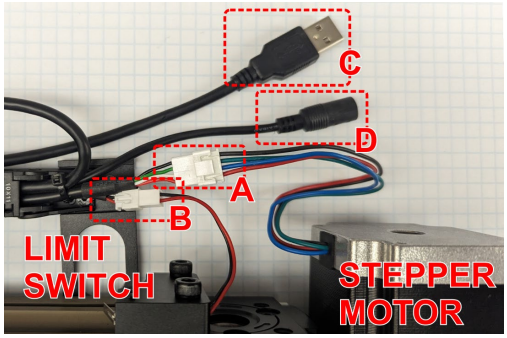
(C) - The USB connection is an extension cable routed through the wire duct and allows the robot to communicate directly with the computer.

(D) - The 12V DC cable powers both the robot and the linear slide base .





STEP 3 – Ensure the stepper motor is connected to the Stepper Motor Extension cable (Four-Wire Connector - Opposite end of cable "A") and the Limit Switch is plugged into the Homing Extension cable (Two-Wire Connector - Opposite end of cable "B")



Cable C will go directly to the computer

Cable D will get its power from the robot's original DC power connector.

LINEAR SLIDING RAIL SPECIFICATIONS & PAREMETERS

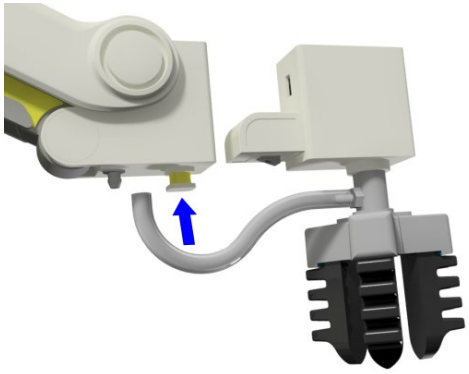
Parameters	Limitations
Maximum Payload	5kg or 11 lbs
Maximum Distance	1000 mm or 39.7 in
Maximum Speed	150 mm/s or 5.9 in/s
Maximum Acceleration	150 mm/s or 5.9 in/s
Repeated Positioning Accuracy	0,01 mm or 0.0004 in
Absolute Positioning Accuracy	0,25 mm or 0.01

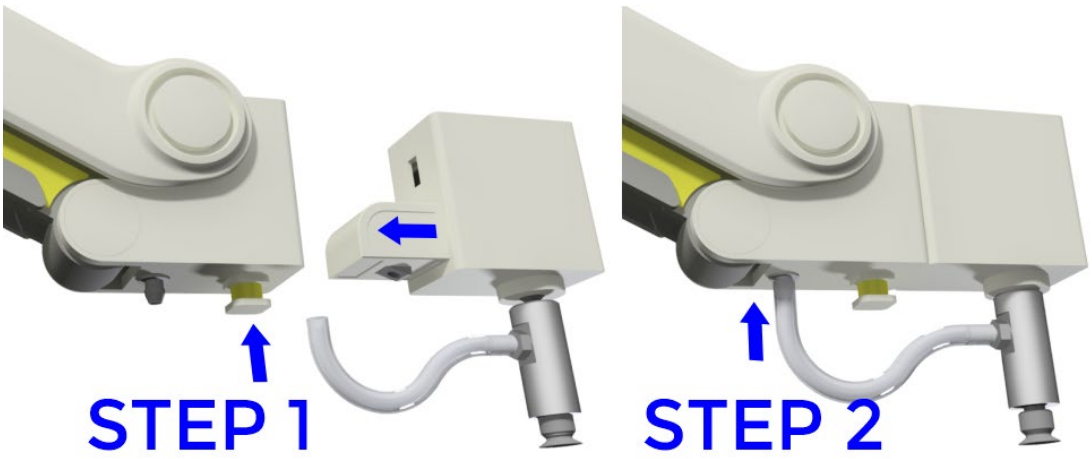
SECTION 3 – SETTING UP & PROGRAMMING THE ROBOT & LINEAR SLIDE RAIL

Caution: NEVER wire anything to the Dobot Magician or Magic Box 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. **Typical Start Up Procedure**

- Disconnect any existing **END of ARM TOOLING (EoAT).**
- Carefully disconnect any existing vacuum tubes.
- Press and hold the release button on the bottom of the arm and pull off the EoAT.

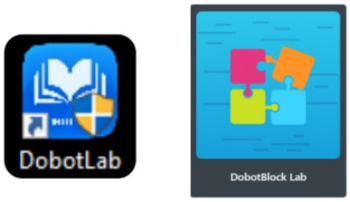




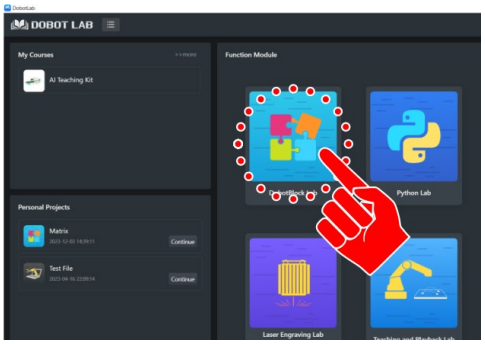
2. Attach the *Suction Cup* as the **END EFFECTOR** or **END of ARM TOOLING (EoAT)** on both robots.

- STEP 1 - Push the *Suction Cup* in until it snaps in place.
- STEP 2 - Attach the hose to the air nozzle.

3. Open up DobotBlock Lab in the software and connect the robot.

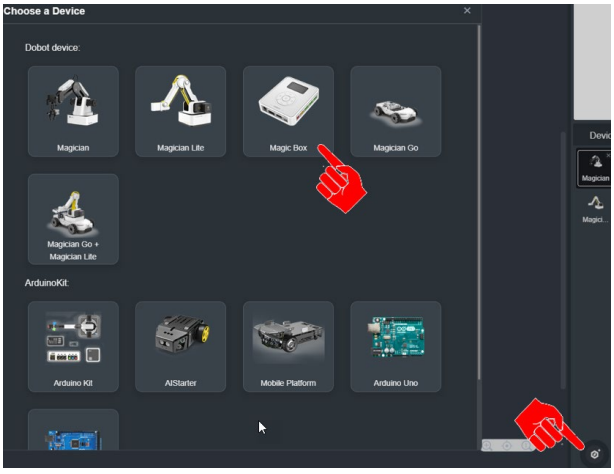


4. Open a new file for DobotBlock Lab

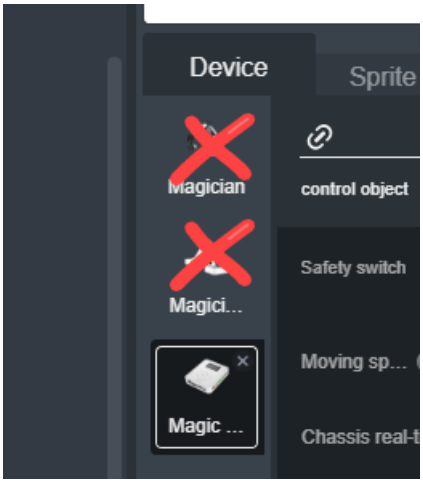


Turn on the power to the Magic Box. The Magician Lite will automatically power up.

5. Add a Magic Box to Devices.

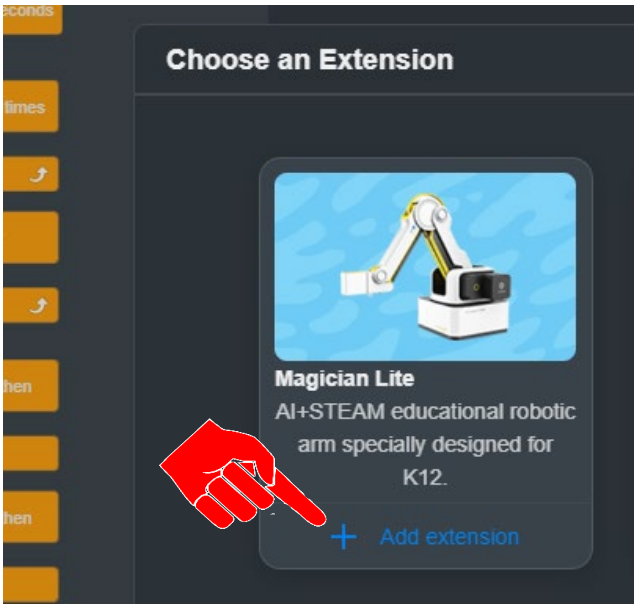


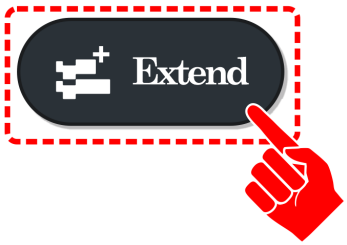
6. Under Devices, delete the Magician and Magician Lite.



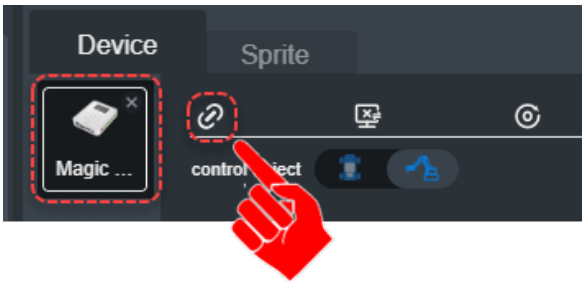
7. For this activity, we will need to add the EXTENSION to use the Magician Lite. Click the EXTEND icon in the bottom left corner.

8. Select Extend, then Select the Magician Lite Extension

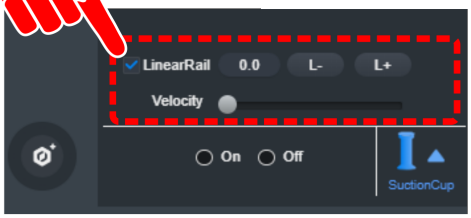




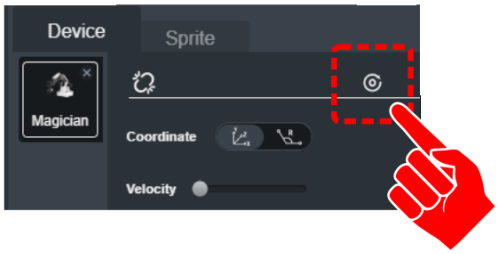
9. Select the connect icon to establish control from the software to the Magic Box. A connection window should pop up.



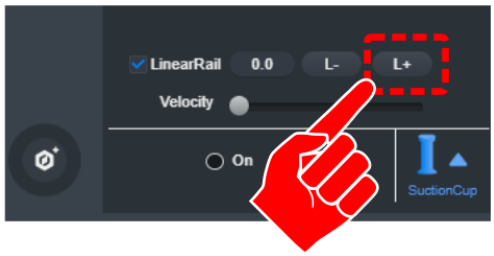
10. Select the CHECK box for the LinearRail BEFORE homing the robot.



11. ENSURE THE AREA IS CLEAR OF OBSTRUCTIONS & LIMIT SWITCH CABLE IS CONNECTED ON BOTH ENDS. When the home icon is selected, the Linear Slide Base will slide all the way to the limit switch (End with the stepper motor). The robot will home after the slide base has homed.

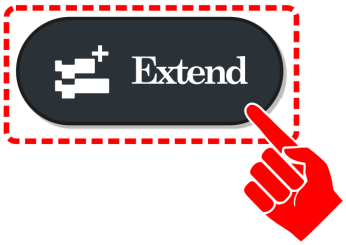


12. Once the Linear Slide Base and Robot are homed, select the L+ icon. The robot should slide away from the stepper motor. The value shown is not actually the distance traveled. Since the motor is a stepper motor, the number shown is a position that the slide base has traveled too.

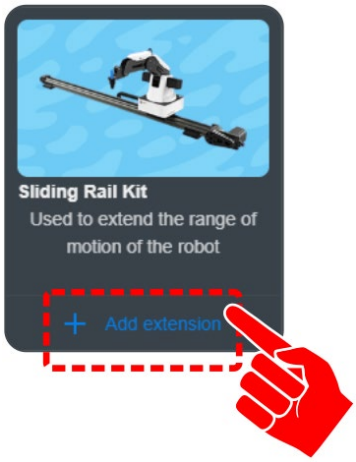


DO NOT ADJUST THE VELOCITY SLIDER!

13. For this activity, we will need to add an EXTENSION to use the COLOR SENSOR. Click the EXTEND icon in the bottom left corner.



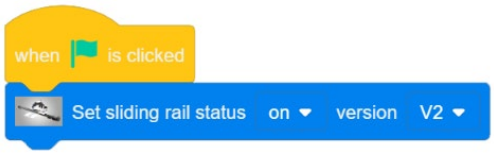
14. Click on "+ Add extension" for the Sliding Rail Kit.



15. The first step in programming this activity is to set the status of the rail.

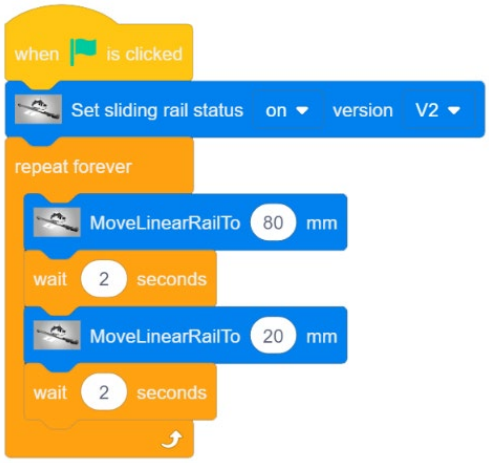
V1 – RED Bumpers at the end of each rail
V2 – BLUE Bumpers at the end of each rail

Sliding Rail Extended Toolbox:
- Set Sliding Rail Status "ON" & "V2"

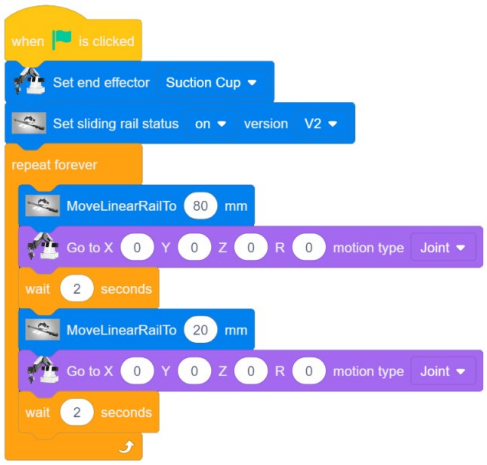


At this time do not use the Set Velocity Block

16. Next, create a simple test program that will send the robot out to location 80mm, wait for 2 seconds, travel back to location 20mm, wait for 2 seconds, and then repeat the loop.

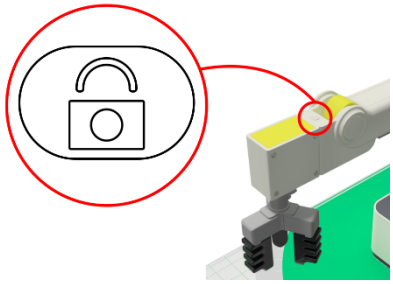


17. Next, add a movement after the robot has reached the 80mm and a different movement at 20mm.



HELPFUL TIPS

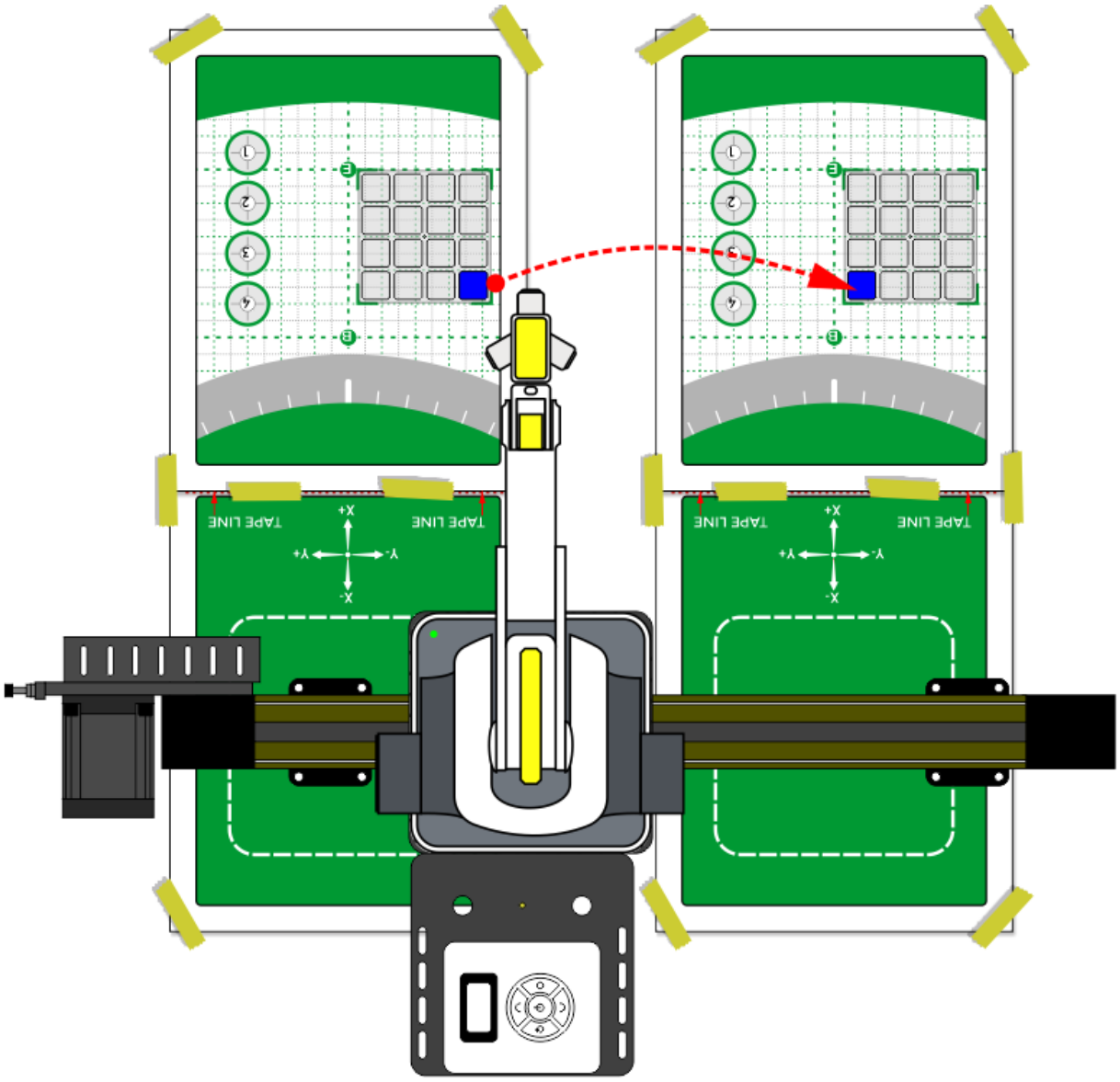
Use the Unlock Arm Control to Grab new locations (any new positions will do for this test) and Fill those Coordinates into the Go To Commands



CHALLENGE

SECTION 4 – PICK AND PLACE USING THE LINEAR SLIDE BASE

1. Tape both field diagrams to the table as shown below (one towards each end of the slide base. See your instructor for specific locations) **NOTE: the slide base shown has been reduced in length*



2. Tape both field diagrams to the table as shown.

3. Create a pick and place activity that will move a block from one field diagram to the other. Get the starting and ending locations from your instructor.

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

CONCLUSION

1. *Find a video example of a working sliding base being used in industry. Explain what operation it is doing.*
2. *How is the slidebase in question #1 similar to the one you are using?*
3. *How is the slidebase in question #1 different from the one you are using?*
4. *What was the most difficult part of this activity for you? Why? Explain fully.*

GOING BEYOND

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

_____ 1. Move a pallet of 4 blocks from one pallet to the other.

_____ 2. Set up the robot to "tend a machine". Use a cardboard box for the machine at one end of the slide, a switch for the machine on/off, a field diagram for the pallet at the other end of the slide, and a cube or cylinder for the part.

   a. Pick a part off the Pallet
   b. Place a part in the machine (Box)
   c. Move the robot out of the machine
   d. When the machine is done (press the switch)
   e. Have the robot get the part out of the machine and return it to the pallet at the other end of the slide

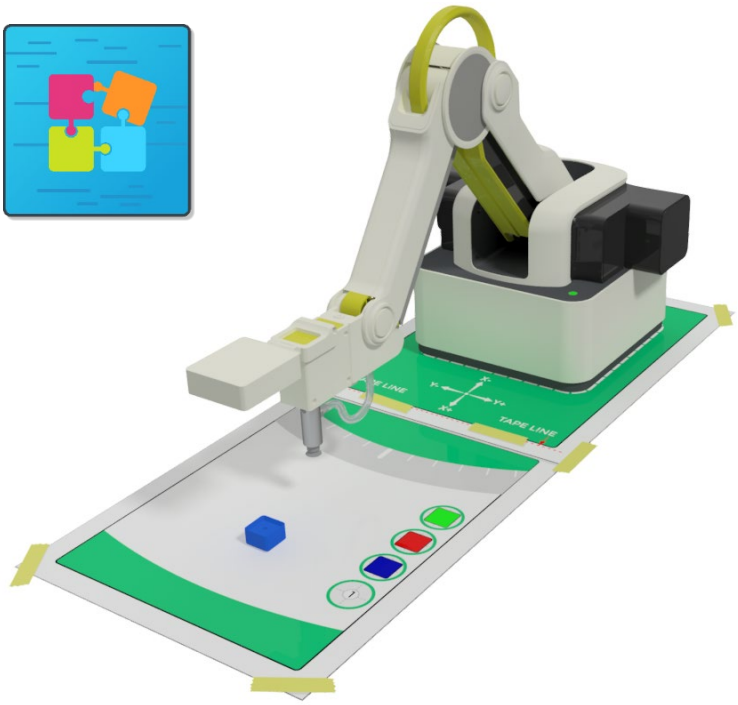
4.3 Block - Camera & Objection Identification

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

INTRODUCTION

Adding a camera to the Magician robot adds vision to the robot. **Vision** systems allow robots to "see" and can be used to help an industrial robot identify, navigate, inspect, handle parts, or complete tasks. This adds a new level of intelligence to the robot allowing it to make more complex choices.

In this activity, the robot will be taught to identify some objects after **calibration**, then see how accurate it is at finding them. It will also use voice to tell us what it has found.



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

KEY VOCABULARY

- Calibration
- Picture/Image Recognition
- Training Model
- Audio Feedback

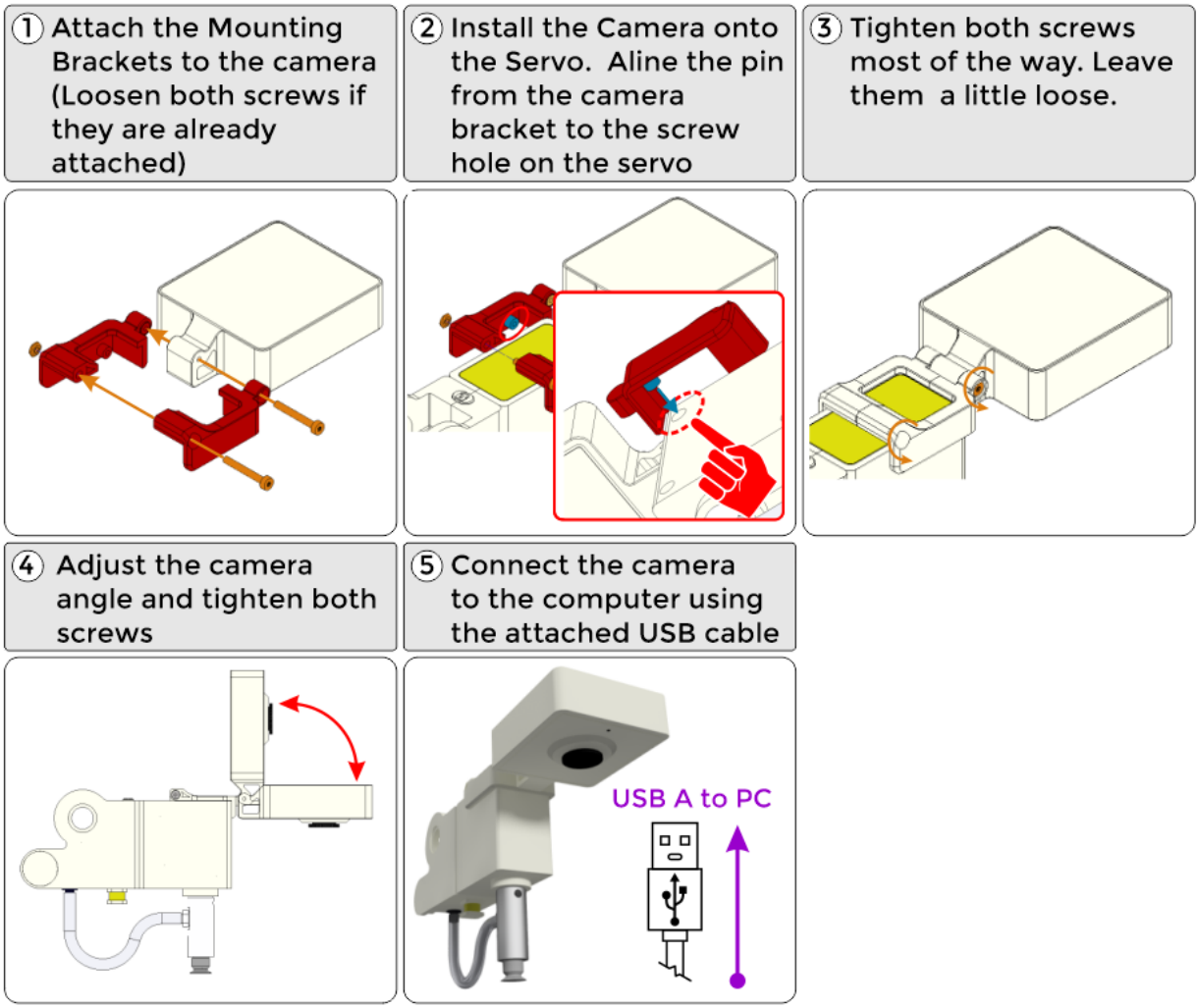
EQUIPMENT & SUPPLIES

- Dobot AI USB Camera
  (Any USB color camera)
- Analog – Potentiometer Module
- 12V Power Cable and Supply
- Magician Robot Arm
- Suction Cup Gripper
- 4 Colored Cubes: Red, Green, Blue, Yellow

PROCEDURE

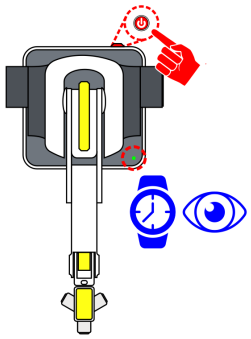
SECTION 1 – CONNECTING the CAMERA to the MAGICIAN

1. Install the camera on the Magician Lite *(Any USB camera can be used, be creative in mounting it.)*

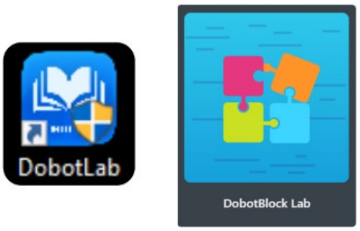


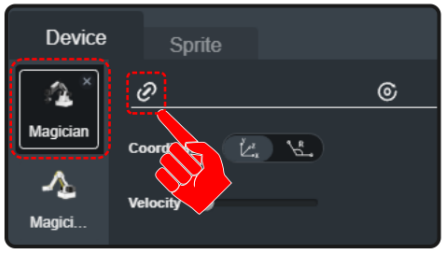
SECTION 2 – CREATING THE CLASSIFICATIONS and TRAINING the MODELS

Once the USB camera is mounted to the robot and the USB cable is plugged into the computer, power ON Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



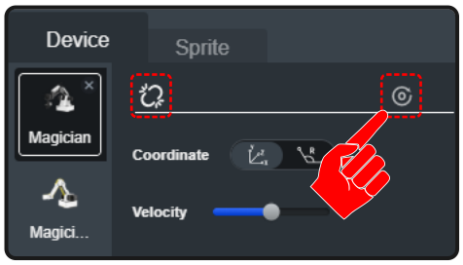
1. Open up DobotBlock Lab in the software and connect to the robot.





2. Follow the same process from previous activities to add the Magician as a device and connect the software to it (establish communication).

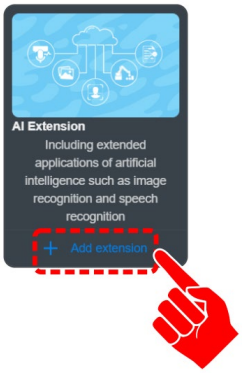
3. Once the robot is connected, we need to home the robot to set its home position.



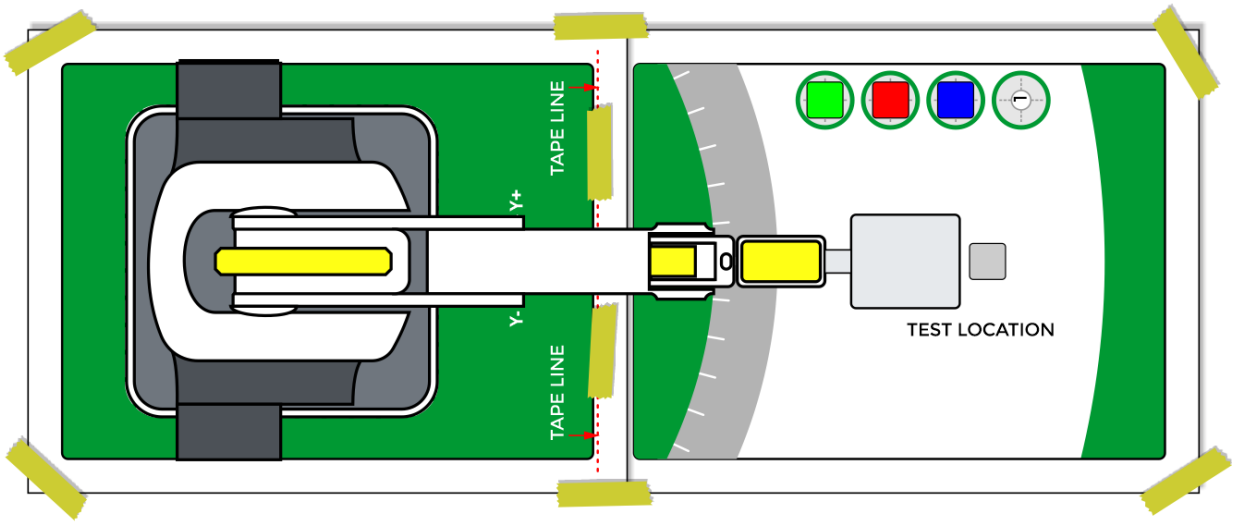
Before **HOMING** the robot, make sure the robot's **WORK ENVELOPE**, the area in which the robot can reach, is clear.

Select the **HOME** icon to start the robots homing process. **HOMING** the robot will return and set the robot to its initial **HOME** position.

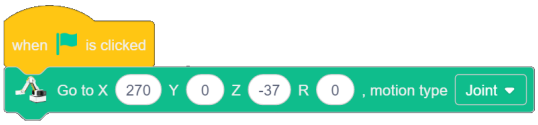
4. Click on "+ Add extension" for the AI EXTENSION.



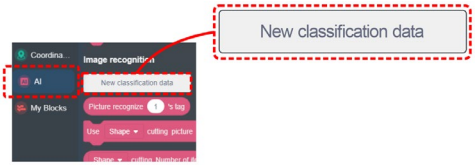
5. Tape a Field Diagram down to the table.



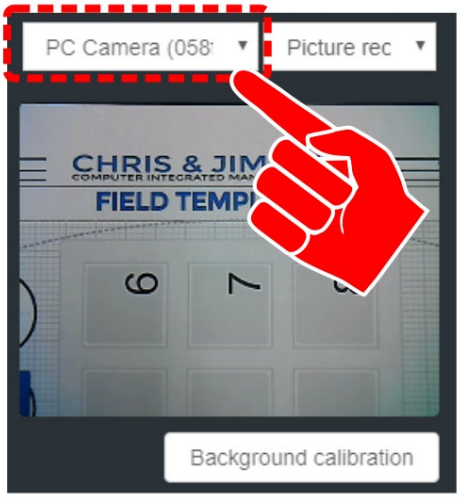
6. Send the Robot to a known position to use the camera.
   For this activity use coordinates similar to:
   X270 Y0 Z-37 R0.



7. Click on NEW CLASSIFICATION DATA from the AI category in the block toolbox.



8. Click the drop down for the camera device selection. Choose the Camera attached to the arm.

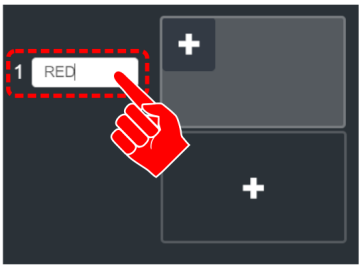


9. The background used in this activity is not a plain white field. Choose Background calibration. This will allow the software to ignore the background as it learns the images during training.

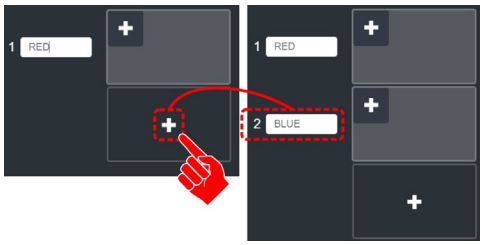
HELPFUL TIPS

When the background of the camera is not pure white, click this button to calibrate the background to improve the automatic recognition rate. Note: pure white background is always the best choice. Before calibration, please make sure that there is no debris in the camera's field of view. Do not place the objects to be trained in the field until after calibration.

10. Label the first classification as RED.

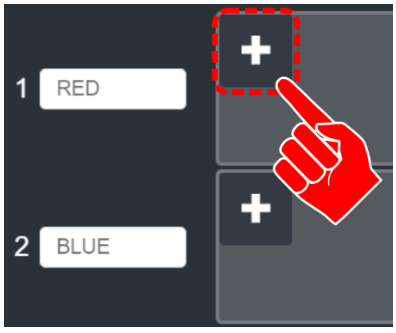


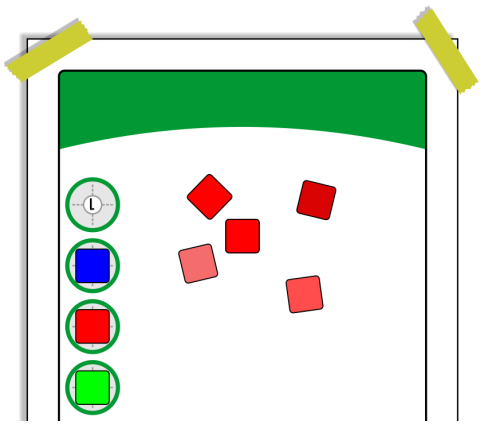
11. Click the Plus in the CENTER of the next window to create a second classification. Label the second class as BLUE.



The software now needs to be taught how to distinguish the difference between the two classifications.

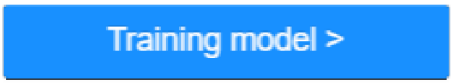
12. Place the red cube within the camera's field of view. Click the PLUS in the top left of the RED class window. A picture will be taken and stored. Move the block to a new location in the field and click the plus again. This will create a second image for the software to use for comparison. Take at least three different images. Try to vary angle, location, different faces, different lighting, and different shadows (any condition that could be present during evaluation).



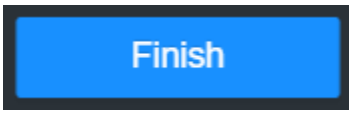


13. Repeat the process for the BLUE class.

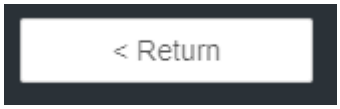
14. Now that the robot has resources to compare against, click the TRAINING MODEL to move to the next phase (testing).



15. Place RED or BLUE cube in the camera's vision. The goal is to have model matching that is close to 100%. Click FINISH to complete the model training.



There can be a delay in switching between classes. If the model matching fails or is not near 100% for each class, click <Return, recollect images, change the lighting, or add additional pictures to re-train the classification.



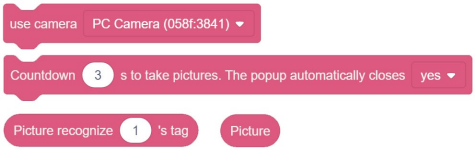
SECTION 2 – TESTING the MODELS with BLOCK CODE

SKILL BUILDER 1 – COMPARING NEW IMAGE TO STORED DATA (MODELS)

Now that the classes have been created and the models have been trained, create a block program to test the accuracy of the setup.

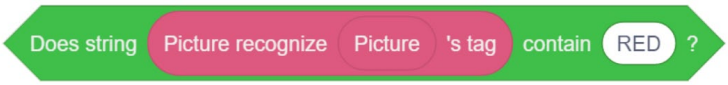
1. Create two variables (RED & BLUE)

2. This program will use a few new blocks from the AI Extension category.
   A. Camera Category – USE CAMERA
   B. Image Acquisition – COUNTDOWN to TAKE PICTURE
   C. Image Acquisition – PICTURE
   D. Image Recognition – PICTURE RECOGNIZE



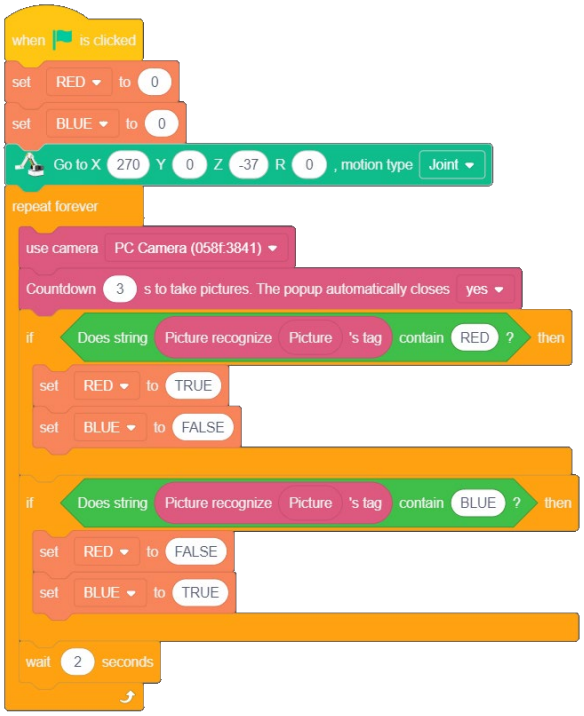
3. The program will also require one new block from the OPERATORS category

4. Link the blocks together as seen to the right.
   This series of blocks will create a condition that decides if the picture it takes fits into the stored model data for the RED classification.

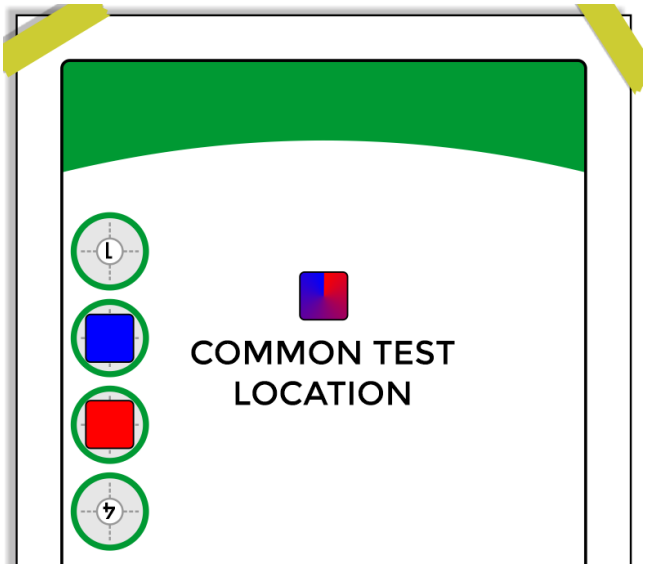


Do not use a variable, be sure to just type "RED" or "BLUE".

5. Using the conditions created, the variables, and a few if statements, create the program to the right that will test the models saved.
   A. Initialize the Variables as False.
   B. Send the robot to or near its HOME location.
   C. Choose which camera to use.
   D. Pause for three seconds and take a picture (the camera popup window will automatically pop up and take the picture. The new picture is stored as "PICTURE".
   E. IF Statement – Ask the question if the new "PICTURE" recognizes/matches the ones stored in the RED class or BLUE class. When they match, set the matching variable to TRUE and display that value for 2 seconds before looping and retesting the model.



Run the program and test the stored models for the RED and BLUE cube.



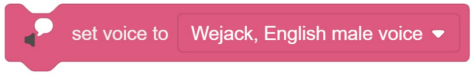
What if there were no block present? Could you make a new classification called "none" and teach it what no block looks like? Try it!

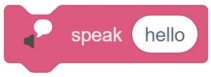
If your set up did not work correctly the first time, what did you have to do to make it work?

SKILL BUILDER 2 – ADDING AUDIO FEEDBACK

Providing user feedback can be a valuable part of programming. Up to this point, the programs have used variables and or LEDs to display visual feedback. This skill builder will show how to add AUDIO FEEDBACK to the program. To do this, the program will new a few blocks from the AI Extension category.

1. Drag over the SET VOICE TO and the SPEAK blocks from the Text tot Speech section of the AI Extension category.

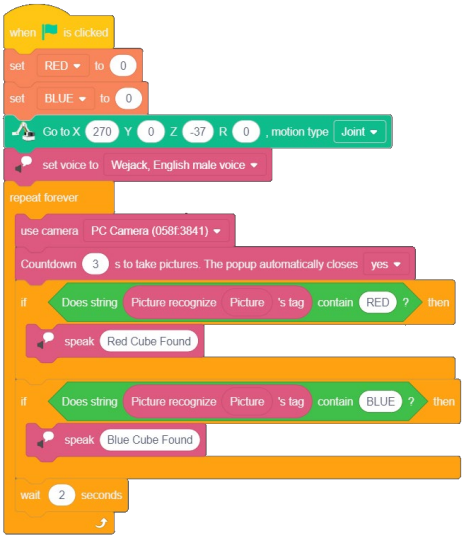




2. In order to keep the program small, remove the variables.

3. Add the Set voice to the header of the program.

4. Replace the variables with the SPEAK Block.



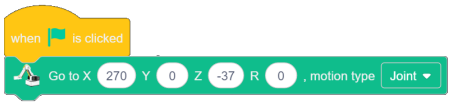
Run the program and test the stored models for the RED and BLUE cube with audio feedback.

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

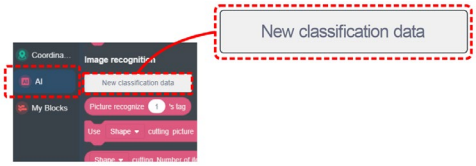
SKILL BUILDER 3 – IMAGE RECOGNITION with NON-CUBE OBJECTS

It may appear that this program is only comparing the color of the cubes. In reality, the images are also comparing things such as size, shape, form, objects with multiple colors and details. They are full color photos that the program is using to build a 3D picture of what it is looking for. This skill builder will test this capability.

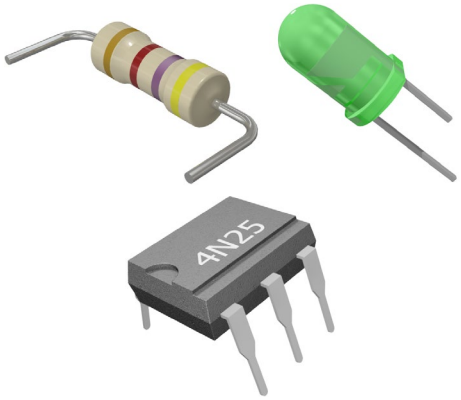
1. For this portion of the activity, lower the Camera's Z to take larger sampling image of the objects & replace the template with a piece of white paper.
   Use coordinates similar to:
   X270 Y0 Z-65 R0.



2. Save the current file and start a new one. Create a NEW CLASSIFICATION DATA model.



3. Create THREE classifications
   A. Resistor B. LED C. Isolator



4. Repeat the process used for the cubes with the electrical components from the handshaking activities. Again, use different angles, orientations, and locations (the more photos taken, the more data the software has to use for comparison).

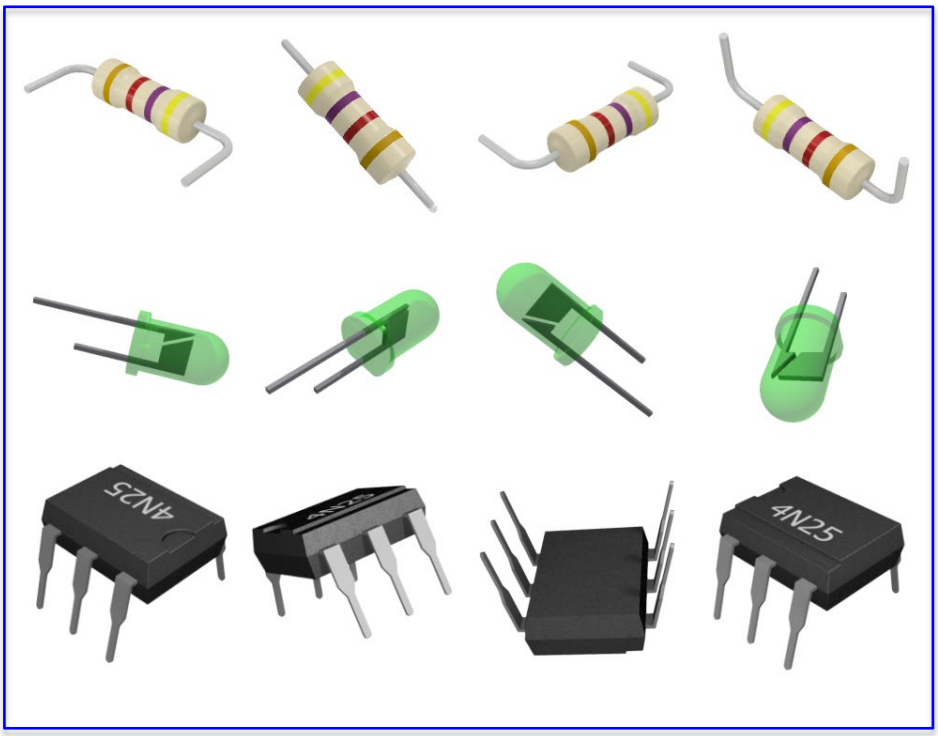
HELPFUL TIPS

If you did not do any of the handshake activities, see if your instructor can give you a few electronics components to test.

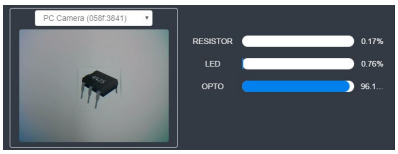
5. Use either Variables (visual feedback) or Speech (audio feedback) to test the program's image recognition capabilities.

HELPFUL TIPS

Print and cut out the images to the right and use them as the reference images (remove the blue border). The images can be used more than once if additional data is needed, just rotate or move the images.



Continue reteaching the models as needed to get consistenet results in the 90% or better range.



Run the program and test the stored models for the electrical components.

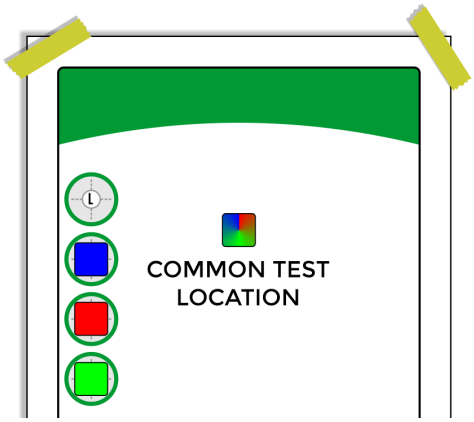
If your set up did not work correctly the first time, what did you have to do to make it work?

CHALLENGE

SECTION 3 – USE the CAMERA to SORT 3 DIFFERENT COLOR CUBES

Develop a program that will test four different cube colors using the camera.

A. Start at a HOME location.
B. Send the robot to a common test location.
C. Test the color of the cube with the camera.
D. Move to a pick-up location.
E. Pick up the cube and place it in a designated drop off location.
F. Loop the program for another cube.



CONCLUSION

1. *Is the camera sensing color or shape only? One or the other? How do you know?*
2. *If you printed the electrical components in black and white, would it still work?*
3. *What percentage was the accuracy for your robot in identifying objects? If it was lower than 100%, what could you do to increase its accuracy?*

GOING BEYOND

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

_____ 1. Complete the skill builder above and have your instructor check your work.

_____ 2. Complete the Challenge above and have your instructor check your work.

_____ 3. What happens when there is nothing in the camera's field of view? Can you make the robot tell you that there is nothing there? How?

_____ 4. Get three or four different electronic components from your instructor. The actual objects, not photographs. Teach the classifications and see if the robot can identify the actual objects.

_____ 5. If you have the sensor kit, teach it three different sensors. *(like the Potentiometer, the light sensor, and the PIR sensor)* Can it accurately see the difference?

4.4 Block - Pick and Place with Stack Light

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

INTRODUCTION

What is a stack light? **Stack lights** are a crucial part of a manufacturing cell. These devices provide visual feedback to the status of a work cell. It allows operators and personnel to know exactly what is happening at any given time. The number of lights and colors vary depending on the application.

Common stack lights consist of three colors:

- **GREEN** – Normal Working conditions
- **YELLOW** – Attention is needed, request for help, or a warning that material may be low.
- **RED** – Indicates that a condition has been met that requires immediate attention. Often comes with an alarm.

Additional Colors

BLUE, **CLEAR/WHITE** – These colors may indicate various conditions such as the machine is running, machine service required, or a user defined signal.

*Humans can trigger **inputs** in a manufacturing cell to start or stop different actions. An output like a **stack light** can provide valuable feedback.*

Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

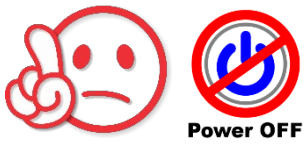
- Stack Light
- 2-Button Input Module
- Magic Box
- Output
- PIR sensor
- Human Sensor
- Input

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Digital – PIR Sensor
- LED Module
- Magician Robot Arm w/ Suction Cup
- 12V Power Cable and Supply
- Common Sensor Cables
- USB C to A Cable

PROCEDURE

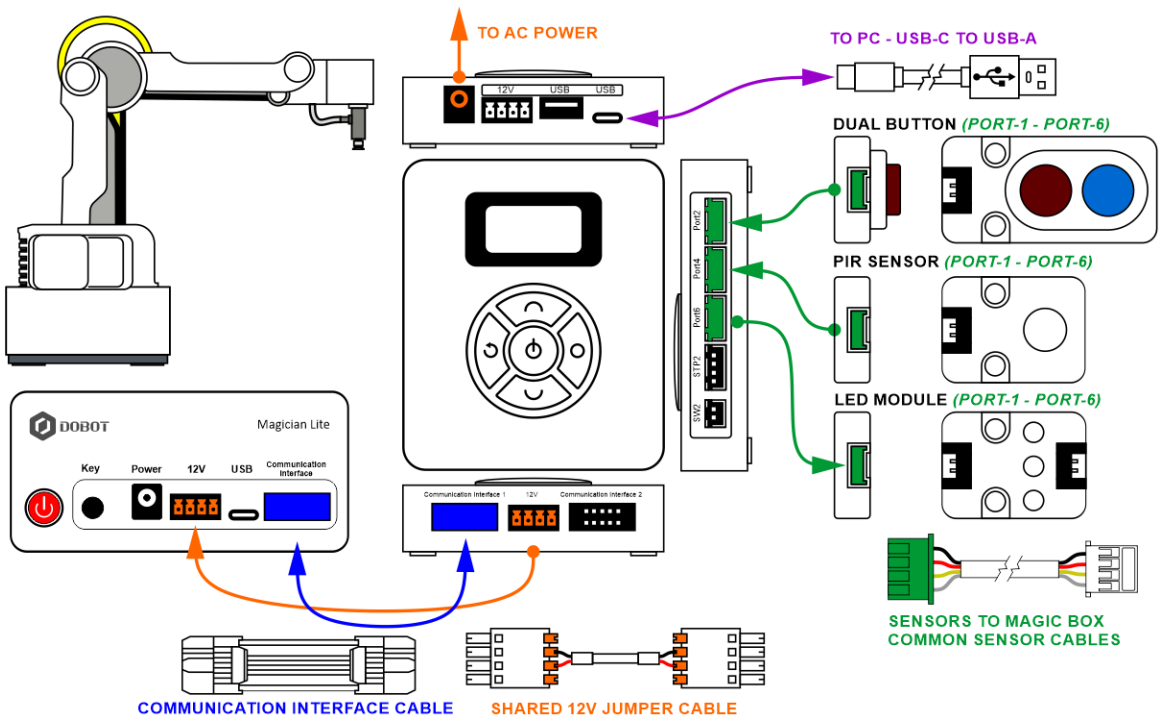
SECTION 1 – WIRING the MAGIC BOX, SENSORS, & MAGICIAN ROBOT



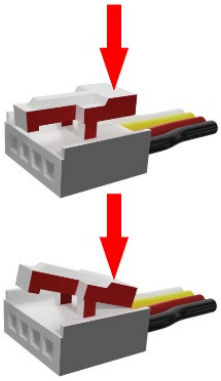
Caution: NEVER wire anything to the Magic Box or Magician while it has power on. ALWAYS turn them off before making connections or damage to the Controllers could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
   - 1x Magic Box
   - 1x USB-A to USB-C Cable
   - 1x AC/DC Power Adapter (12V)
   - 3x Common Sensor Cables
   - 1x DUAL BUTTON – ***PORT 2***
   - 1x PIR SENSOR – ***PORT 4***
   - 1x LED MODULE – ***PORT 6***

2. Wire the Magic Box as shown below



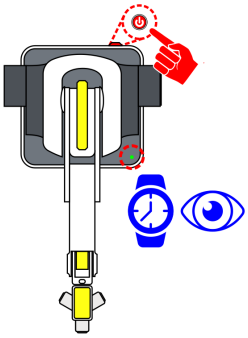
Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! *It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.*

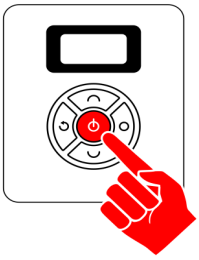


SECTION 2 – CONNECTING the MAGIC BOX & ROBOT to DOBOTLAB

1. Open up DobotBlock Lab in the software.

2. Once **ALL** the wiring is done (sensors connected and robot arm connected), power ON the Magic Box and Robot. **WATCH** and **WAIT** for the robot to completely power on. (GREEN INDICATOR)





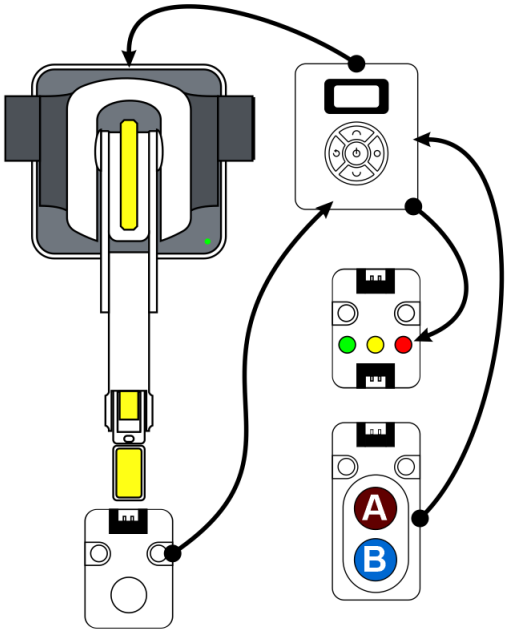
3. Repeat the connection process completed in previous activities.
    a) Connect to the Magic Box in DobotLab.
    b) Add the AI Sensor Kit and Magician Extensions.
    c) Ensure the robot's envelope is clear and home the robot.

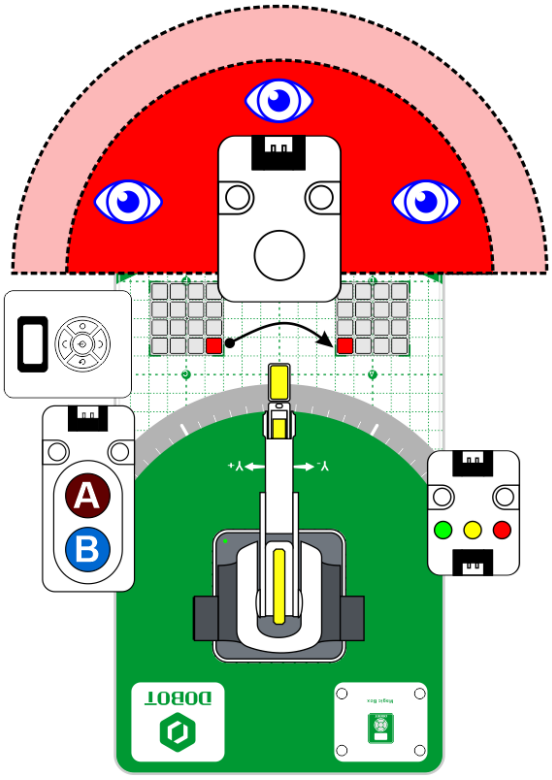
CHALLENGE

SECTION 3 – CREATE A STACK LIGHT PROGRAM

For this challenge, develop a pick and place routine that adds:

a) A push button (B-BLUE) to start each complete loop (block in place)
b) A stack light for visual feedback
    - GREEN ONLY - Normal Operation - Moving
    - GREEN & YELLOW Waiting for Block (B-BLUE Push Button)
    - RED ONLY – Motion detected while in operation
c) A PIR sensor that detects movement around the work cell environment that stops the primary operation (only while in operation)
d) A push button (A-RED) to restart the process from warning

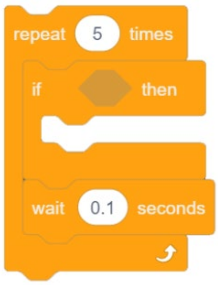




At this time, using Blockly, a code does not exist that will stop motion once it has been started (GO TO or JUMP). For this simulation, the program will only look for motion (from the PIR) inside the work envelope in-between each motion. This is an excellent opportunity to develop a MY BLOCK that can be placed throughout the program.

THINGS TO THINK ABOUT:

- What variables would be helpful? (both to simplify the program but also to debug it if it is not working as planned)
- What MY BLOCKS would be helpful? (break up the program into sections… waiting?)
- Use of JUMPS to cut down on motion blocks… but… does that take away the places to insert "look for motion"?
- Possibly try something like the code below to try to catch motion? 5x 0.1 = 0.5 … half a second?



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

CONCLUSION

1. *Why was the program unable to stop the robot while it was in motion?*

2. *What other sensors/inputs from the kit could be used to add safety in a manufacturing cell, and how would they be used?*

GOING BEYOND

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

_____ 1. Add the servo as an output in the work envelope to simulate dropping a safety shield while the robot is in motion.

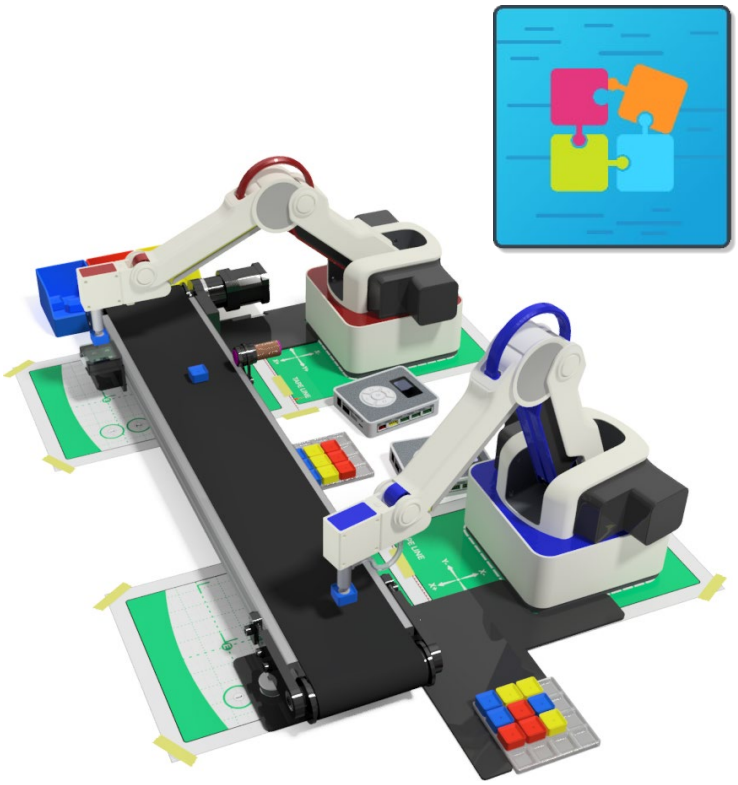
_____ 2. Replace the 2-Button Input module with the gesture sensor. Make “swipe left” do blue button, and “swipe right” do red button.

4.5 Block - Work Cell Scenario: Robots, Sensors & Conveyor - Guided

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

INTRODUCTION

A robotic workcell is defined as the complete environment around a robot. This environment may include tools, machines and/or other robots and perform a variety of different manufacturing processes as well as moving parts around.



In this activity you will use a robot and a microcontroller system to recreate a workcell. Your workcell will incorporate all of the devices and skills that you have learned about in previous activities including:

- Inputs & outputs
- Sensors
- Machines
- Robots

Due Date: _____

KEY VOCABULARY

- Handshaking
- Workcell
- Output
- Nesting
- Optical Isolator
- Sensor
- Input
- Palletize

EQUIPMENT & SUPPLIES

- x2 Magician Lite Robots
- X2 Pneumatic Suction Cups
- X1 Conveyor System
- Dobot Lab Software
- X1 Mini Conveyor Color Sensor
- X1 Mini Conveyor IR Sensor
- X2 Magic Boxes
- X1 Dual Button Module
- X3 Small Sorting Bins
- Assorted Cables

PROCEDURE

Caution: NEVER wire anything to the Dobot Magician or the Magic Box 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.

Create a fully integrated **WORKCELL** using the following scenario.

MAGICIAN LITE - 1

- Primary sequence is initiated by a **DUAL BUTTON (RED BUTTON – START)** that sends a signal to **ROBOT 1**
- Once the sequence is started, **ROBOT 1** will look to see if the **UNSORTED MATRIX** of cubes is empty.

  -If **UNSORTED STORAGE MATRIX** is **NOT** empty, **ROBOT 1** will take one cube at a time and place it onto the conveyor. This process will repeat until the matrix is empty.

  -If **UNSORTED STORAGE MATRIX** is empty (all nine cubes have been sent to sorting), **ROBOT 1** will take parts from the matrix of cube parts in the **BUFFER STORAGE. ROBOT 1** will check to see if the **UNSORTED STORAGE MATRIX** has been refilled (**DUAL BUTTON / RED BUTTON - RESET** switch has been pressed again) after each part is taken from the **BUFFER STORAGE**.
- After a part is placed on the **CONVEYOR** system, **ROBOT 1** will return home
- **ROBOT 1** checks if **ROBOT 2** is sending a busy signal
  -If **ROBOT 2** is sending a **BUSY** signal, **ROBOT 1** will wait
  -If **ROBOT 2** is **NOT** sending a **BUSY** signal, **ROBOT 1** will start the **CONVEYOR** system.
- **ROBOT 1** will stop the **CONVEYOR** system when an object is detected by the **IR SENSOR**
- **ROBOT 1** will send a **PICK UP** signal to **ROBOT 2.** Continue sending the signal until **ROBOT 2** sends back a **BUSY** signal
- Once the busy signal is received, **ROBOT 1** will loop back to pick another cube
- **ROBOT 1** will take a cube from the **UNSORTED STORAGE MATRIX** and place it in the **CONVEYOR** system.

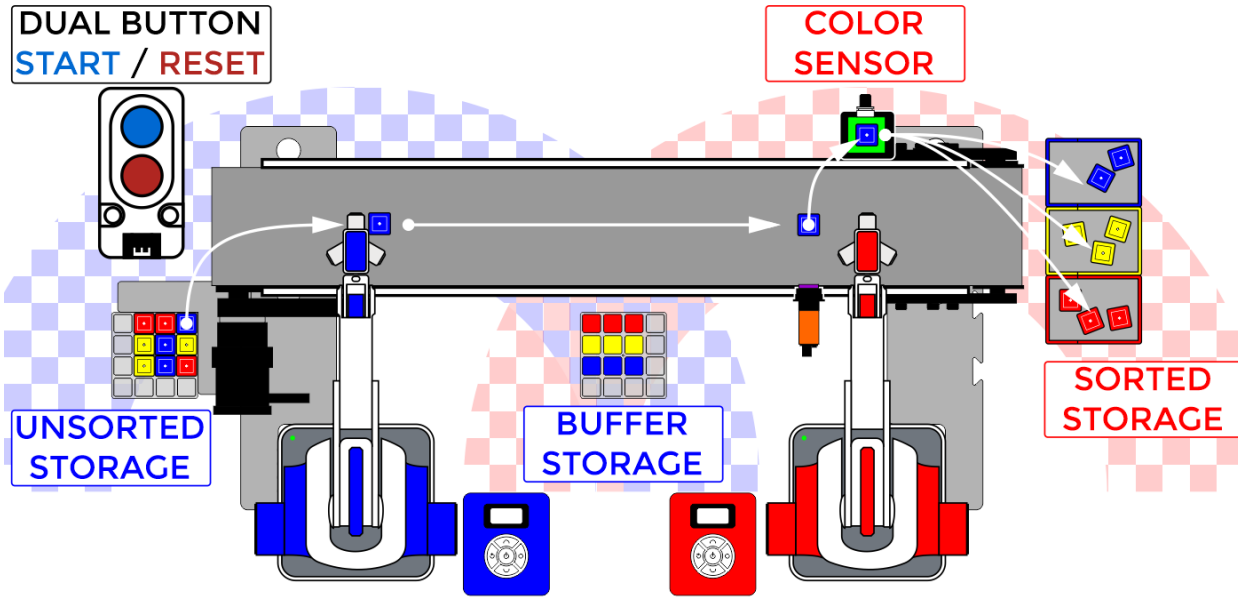
MAGICIAN LITE – 2

- **ROBOT 2** waits for signal

  -If **NO PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** waits

  -If a **PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** will start sending a **BUSY** to **ROBOT 1**. **ROBOT 2** will then pick up the cube sitting on the **CONVEYOR** system at the **IR SENSOR**.
- **ROBOT 2** will evaluate the cubes color by hovering it about 3mm above the **COLOR SENSOR.**
  -If the color is **BLUE**, the robot will place the cube in the **BLUE STORAGE**
  -If the color is **YELLOW**, the robot will place the cube in the **YELLOW STORAGE**

  -If the color is **RED**, the robot will place the cube in the **RED STORAGE**
- **ROBOT 2** will return home and stop sending the **BUSY** signal to **ROBOT 1**.
- **ROBOT 2** will then loop back to waiting for the **PICK UP** signal from **ROBOT 1**.



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

Note any additional parameters that are provided by your instructor. Take notes in the space below.

CONCLUSION

Answer the questions below and hand-in electronically to your instructor.

1. *What are the inputs you used in your workcell?*

2. *What are the outputs you used in your workcell?*

3. *What was the hardest part of this activity for you? Explain how you made it work.*

4. *Describe the process that you and your team went through to complete this project by inserting at least 5 pictures, with captions, that you took throughout the project. Be sure to include programming, designing, wiring, building, and testing.*

GOING BEYOND

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

Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs and consult with your instructor before turning the robot on, as damage to your robot or your other equipment may result.

_____ 1. *Replace the human component of starting the system with a sensor. What sensor did you use and why?*

_____ 2. *Could you use other sensors in this project? Which ones, and how would you use them? Check with your instructor once you come up with an idea, and be sure you get approval before you try it.*

_____ 3. *Integrate a stack light with your robot so that you know when the signals are being sent.*

_____ 4. *Incorporate the Camera from the Sensor Kit and have robot two announce what is going into the bins… (example: red block = case of wrenches, yellow block = case of screwdrivers etc.)*

4.6 Block - Work Cell Scenario: Robots, Sensors & Conveyor - Open Ended

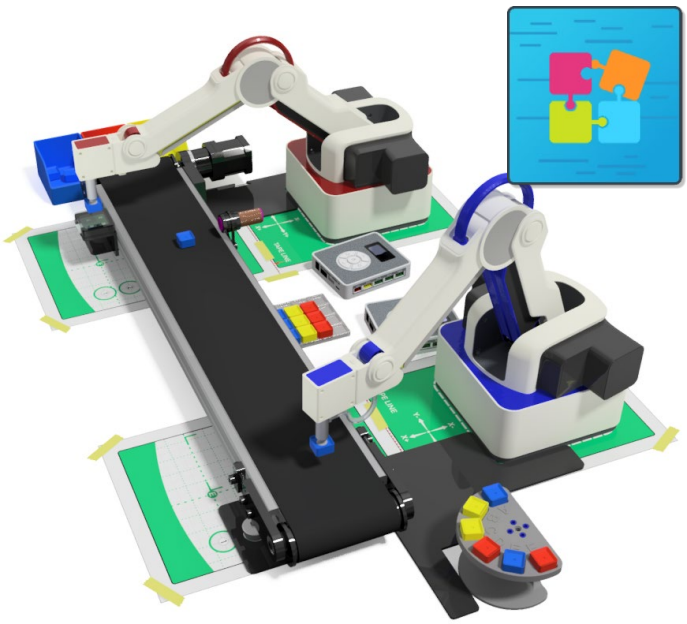
NAME(s): _____ Date: _____ Section: _____

INTRODUCTION

A robotic **workcell** is defined as the complete environment around a robot. This environment may include tools, machines and/or other robots. This environment may include tools, machines and/or other robots and perform a variety of different manufacturing processes as well as moving parts around.

In this activity you will use a robot and a **microcontroller** system to recreate a workcell. Your **workcell** will incorporate all the devices that you have learned about in previous activities including:

- Inputs & outputs
- Sensors
- Conveyor belt
- Machines
- Robots



An example of a workcell using robots, prototyped parts and other components.

KEY VOCABULARY

- Handshaking
- Workcell
- Output
- Nesting
- Conveyor
- Optical Isolator
- Sensor
- Input
- Palletize
- Microcontroller

EQUIPMENT & SUPPLIES

- Magician Lite(s)
- 1" cylinders or cubes
- Conveyor System
- EoAT for each robot
- Magic Box(es)
- DobotLab software
- Sensors
- Block Feeder System and Servo

PROCEDURE

Caution: NEVER wire anything to the Dobot Magician or the Magic Box 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. You and your team will design, organize, create, program, and test a full work cell. Your instructor will have you pick from the list below the items that must be included in your work cell.

☐ **Dobot Magician - # of _____**

☐ **Dobot Conveyor System**

☐ **Dobot IR Sensor**

☐ **Dobot Color Sensor**

☐ **Handshakes - # of _____**

☐ _____

☐ **Magic Boxes - # of _____**

☐ **Student Designed Feeder System**

☐ **Student Designed Machine - # of _____**

☐ **Student Designed Storage System**

☐ _____

☐ _____

2. Be sure to note any additional parameters that are given to you by your instructor (due date, size or storage requirements, and additional items from home. Take notes in the space below.

3. In the time allotted for this project design a workcell that includes the following:
   a. Accurate Robot **PICK AND PLACE** routines.
   b. **JUMPS** where appropriate.
   c. Proper usage of **FUNCTIONS** and **VARIABLES**.
   d. A **PALLETIZE** or **STACKING** routine.

4. Create a video of your workcell.

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.

Notes:

CONCLUSION

1. *Make a flowchart/Process flowchart of your workcell as indicated by your instructor in the space below.*

2. *What's the pseudocode that you used for your microcontroller program? Copy and paste it here.*

3. *What are the inputs you used in your work cell?*

4. *What are the outputs you used in your work cell?*

GOING BEYOND

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

_____ 1. Make your work cell communicate with someone else's work cell in your class. When your process ends, theirs begins.

_____ 2. Continue number one above with as many cells in the class as you can. The same part moves through all of the cells, one by one, until a finished product is produced. See how many work cells you can integrate in the time allotted by your instructor.

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING
MAGICIAN LITE

- CHAPTER 5 -
ROBOTICS RESOURCES

CHAPTER INTRODUCTION:

This chapter contains additional items that may need printed such as field templates and wiring diagrams. It also contains a glossary for industrial robotics and one for blockly programming. It also contains a list of academic standards that can be found throughout the curriculum.

CHAPTER RESOURCES:

- Key Concepts
- Peripheral Wiring Guide
- Magic Box IO Guide
- Work Cell Wiring Guide
- Color Test Template
- Field Diagrams
- 3D Printed Parts Folder Link
- Robotics Glossary
- Blockly Glossary
- Academic Standards
  - Standards for Technology and Literacy
  - Next Generation Science Standards
  - ISTE Standards

Key Concepts

INTRODUCTION

Throughout the chapters in this curriculum, there are many key concepts covered that are of interest to the teachers and administrators that use these activities. All of those are listed here for your use.

KEY CONCEPTS

1.1 Robot Axis & Movement

- Differentiate ways that robot arms can move: Move Linear and Move Joint.
- Differentiate between absolute and relative coordinates.
- Differentiate between teaching and recording points.
- Starting up and connecting the Dobot Magician.
- How to utilize a robot arm to move through a group of points by using the pen end effector.
- How to use DobotLab Teach and Playback Module.

1.2 Pick & Place Routines – Teach & Playback

- How does a robot perform a pick and place operation?
- What end effector or end of arm tooling (EoAT) works best?
- What are Pick and Place conventions in industry?
- How do I attach the Mechanical gripper to the Dobot?
- How do I record positions with the Dobot?
- How do I easily edit a program in DobotLab and Playback Module?

1.3 Pick & Place Routines - Blockly

- Differentiate ways that robot arms can move using blockly: Straight Line and Joint.
- Differentiate between absolute and relative coordinates.
- What Delays are used for in robotics and how to create them.
- What are some of the basic Dobot configurations?
- How suction is used to move objects and how to turn it on and off using blockly.
- Starting up and connecting the Dobot Magician.
- How to utilize a robot arm to move through a group of points by using an end effector.

1.4 Using Jumps & Loops – Teach & Playback

- What are the advantages to using a Jump in a program?
- What are the advantages to using a Loop in a program?
- How do I program Jumps and Loops in DobotBlock?

1.5 Using Jumps & Loops - Blockly

- What's a Jump, and why is it important?
- How to make the robot repeat a motion or a task?
- How do I use a Delay statement? What are they used for?
- How do I use a Forever statement and what are they used for?
- How do I set Jump Height?

1.6 Using Pallets

- Why are variables used in programming?
- How do I use variables in blockly to stack and unstack allets?
- How can I use math to stack or unstack pallets?
- What are MyBlocks and how do they simplify programming?
- What are some of the blockly commands needed to do this?
- Why is palletization and de-palletization important in industry?

2.1 Robot to Robot Communication: Handshaking

- How do I make a robot send an internal signal?
- How do I get a robot to receive an internal signal?
- Why is communication between robots important?
- What ways are there to make this happen through software?

2.1 Robot to Robot Handshake: Magic Box to Magic Box

- How do I get a Magic Box to send a Bluetooth signal?
- How do I get a Magic Box to receive a Bluetooth signal?
- What hardware is required to send and receive signals?
- How are signals programmed in Blockly?

What are the advantages and disadvantages for sending signals wirelessly?

2.2 Magic Box to Magic Box Handshake: Bluetooth

- How do I get a Magic Box to send a Bluetooth signal?
- How do I get a Magic Box to receive a Bluetooth signal?
- What hardware is required to send and receive signals?
- How are signals programmed in Blockly?
- What are the advantages and disadvantages for sending signals wirelessly?

2.3 Pick & Place with the Magic Box

- How do I safely connect the Magic box to the Robot?
- Where do I connect the 2-Button input module on the magic box?
- How do I program in blockly to give the robot some sensor-based feedback?
- Which sensors would make a good "Start" in a blockly program? What makes them good for this?

3.1 Intro to IO with the Magic Box

- What is the difference between an input and an output?
- What is the difference between a digital signal and an analog signal?
- How can a robot use input signals to provide feedback?
- What can robots use outputs for?
- Can a robot use multiple inputs and outputs at the same time? How?

3.2 Sensor Exploration with the Magic Box

- What does a PIR sensor do, and how is it used?
- What does a Sound sensor do, and how is it used?
- What does a Gesture sensor do, and how is it used?
- What does a Humiture sensor do, and how is it used?
- What does a Light sensor do, and how is it used?
- What does a PIR sensor do, and how is it used?
- What does a Photoelectric switch do, and how is it used?
- What can a joystick be used for in robotics?
- What is a micro servo and how is it used in robotics?

3.3 Color Sensor with the Magic Box

- What can a color sensor be used for in industry? How does it work?
- How do you code a color sensor in Blockly?

3.4 Using the Color Sensor

- What's the difference between digital and analog?
- What colors can I sense with the color sensor?
- How does the color sensor work?
- How do I wire the components together?
- How do I get a value from the color sensor?
- How do I print to a log?
- How do I sort items by color with a robotic arm?

3.5 Using a Servo Driven Part Feeder

- How does a servo work?
- Why is a servo motor a good choice for a part feeder?
- What can a Part Feeder be used for in industry? How does it work?
- How do you code a part feeder in Blockly?

4.1 Starting & Stopping A Conveyor

- How does the conveyor and IR sensor get wired to the Magic Box and the Magician?
- How can I use a robot to move the conveyor?
- How can I use an IR Sensor to stop the conveyor?
- What are the practical uses for conveyors in industry?
- How do you code a conveyor in blockly?

4.2 Pick & Place with the Linear Slide Rail

- How does the slide base get wired to the Magic Box and the Magician?
- How can I use a robot to move on a sliding base?
- What can a sliding base be used for?

4.3 Camera & Object Identification

- How does the camera attach to the Dobot Lite?
- What can the camera be used for in industry?
- Does it work with color? Shape? Something else?
- What are classifications, and how do you set them?
- What is the method of testing the camera to see if it works?
- Can you use audio feedback to make the robot tell you what it sees? How?
- How do you program the camera to work with Dobot Block?

4.4 Pick & Place with a Stack Light

- How does a stack light get wired to the Magic Box?
- What is a stack light, and what is it used for in industry?
- How do you program a stack light to work in Dobot Block?

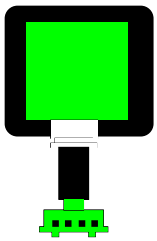
4.5 Work Cell Scenario: Robots, Sensors & Conveyor

- How do you integrate robots and other parts of a workcell to complete a given task?
- How do you safely communicate between a microcontroller and a robot?
- What are the different types and styles of inputs and outputs needed to complete your given tasks?
- Which end of arm tooling is most appropriate for your workcell?
- Where is it appropriate to use the jump command within the workcell?
- Where would it be appropriate in your programming to use either variables or functions while programming?
- What components of blockly did you need to complete this task?
- What other software & hardware will I need to complete this task?

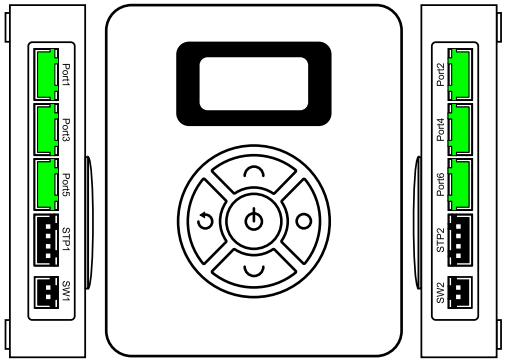
4.6 Work Cell Scenario: Robots, Sensors, & Conveyors: Open Ended

- How do you integrate robots and other parts of a workcell to complete a given task?
- How do you safely communicate between a microcontroller and a robot?
- What are the different types and styles of inputs and outputs needed to complete your given tasks?
- Which end of arm tooling is most appropriate for your workcell?
- Where is it appropriate to use the jump command within the workcell?
- Where would it be appropriate in your programming to use either variables or functions while programming?
- What components of blockly did you need to complete this task?
- What other software & hardware will I need to complete this task?

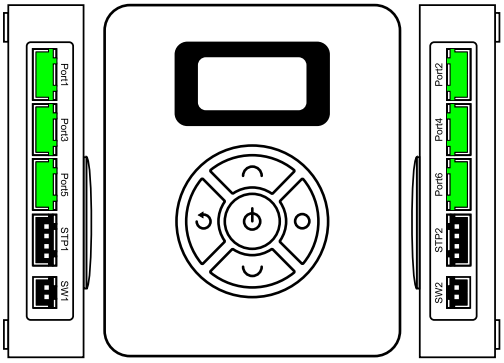
PERIPHERAL GUIDE

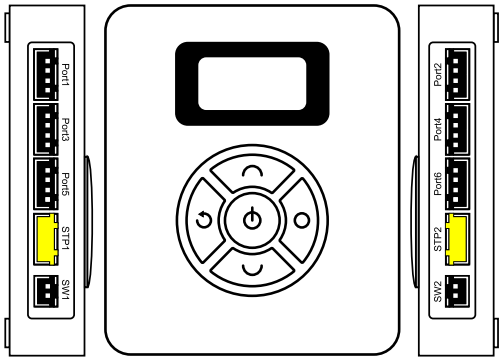


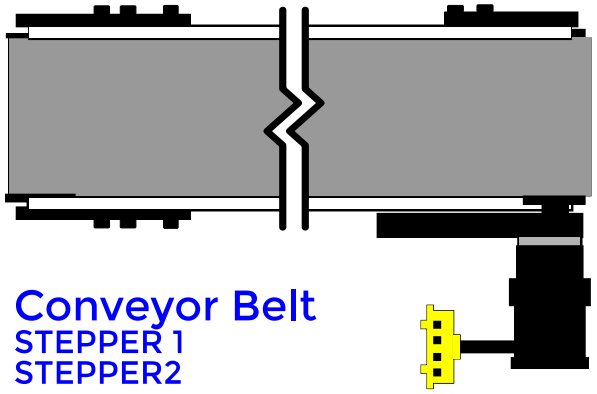
Color Sensor
PORT1 - PORT6



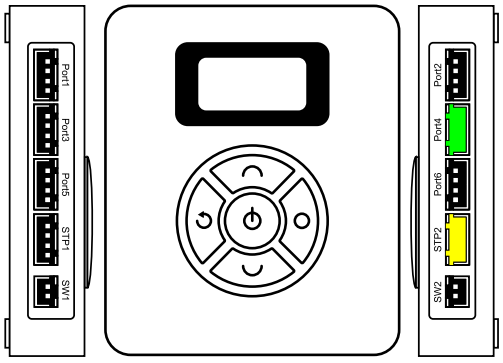
Photoelectric IR Sensor
PORT1 - PORT6



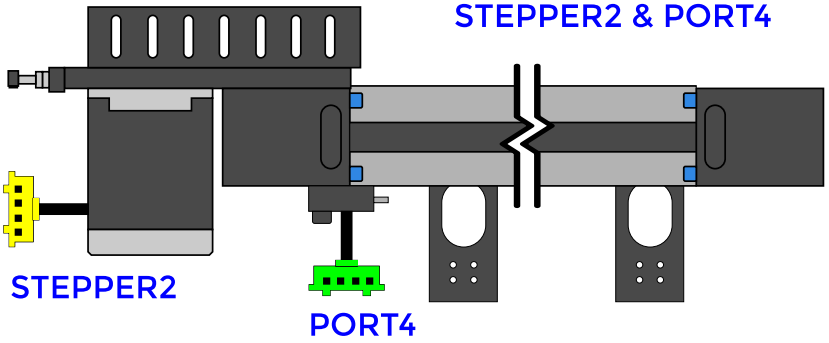




Conveyor Belt
STEPPER 1
STEPPER2



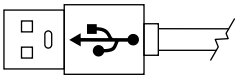
Linear Slidebase
STEPPER2 & PORT4



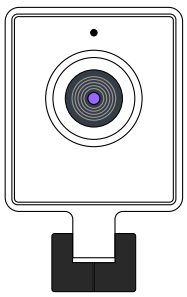
STEPPER2

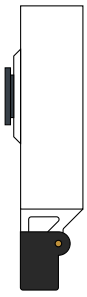
PORT4

Vision / Camera
PC USB A Port



*Use a spare USB port on your computer, not the Magic box

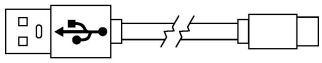




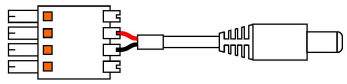
CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING

AI SENSOR GUIDE

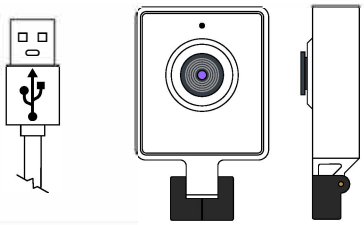
USB-A TO USB-C CABLE



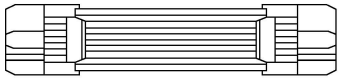
4 PIN POWER CABLE - 12V



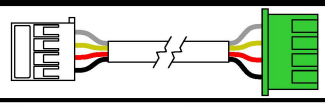
CAMERA

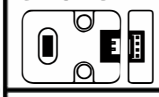
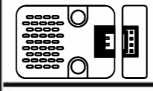
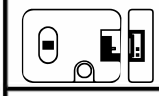
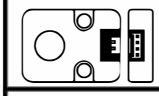
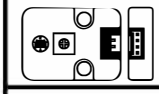
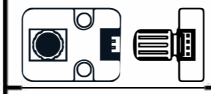
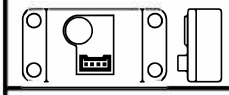
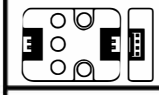
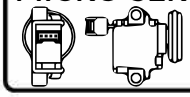


10 PIN COMMUNICATIO CABLE



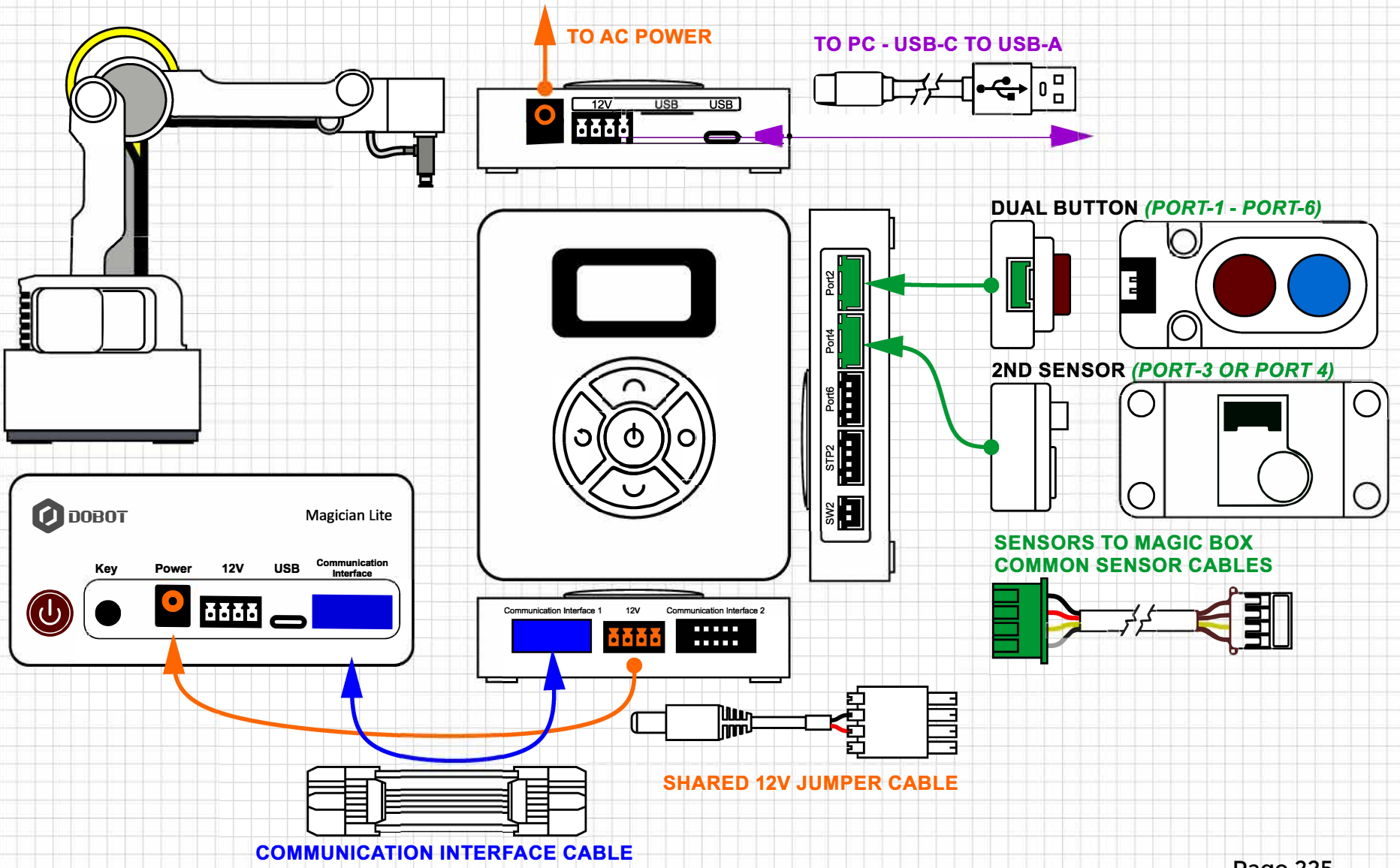
4 PIN COMMON SENSOR WIRE



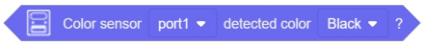
MODULE	COMM MODE	PORTS	SIGNAL	NOTES
GESTURE SENSOR 	I2C	1 - 6	DIGITAL	EFFECTIVE DISTANCE 2in - 6in, UP DOWN RIGHT LEFT FORWARD, BACKWARD, CW, CCW
HUMITURE SENSOR 	I2C	1 - 6	ANALOG	TEMP RANGE 0 - 140 F HUMIDITY RANGE, 10-90 RH
JOYSTICK 	I2C	1 - 6	ANALOG DIGITAL	X/Y VALUE 0-250 Z 0:RELEASED 1:PRESSED
COLOR SENSOR 	I2C	1 - 6	ANALOG	LARGER VALUE = DARKER COLOR, 0:NO COLOR, 1:RED, 2:GREEN, 3:BLUE, 4:YELLOW, 5:BLACK, 6:WHITE
DUAL BUTTON 	IO	1 - 6	DIGITAL	0:RELEASED 1:PRESSED
PIR SENSOR 	IO	1 - 6	DIGITAL	EFFECTIVE DISTANCE 59in,
PHOTOELECTRIC SENSOR 	IO	1 - 6	DIGITAL	VALUE RANGE 0:NONE 1:DETECTED RESPONSE TIME <2ms
LIGHT SENSOR 	ADC	3 & 4	ANALOG	BRIGHTNESS RETURN VALUE 140-600 LARGER VALUE = MORE LIGHT
POTENTIOMETER KNOB 	ADC	3 & 4	ANALOG	RETURN VALUE 0-407
SOUND SENSOR 	ADC	3 & 4	ANALOG	RANGE 0-1023 SENSITIVITY 48db-52db
LED MODULE 	SINGLE BUS	1 - 6	DIGITAL OUT	RGB RANGE 0-255, BRIGHTNESS RANGE 0-100%
MICRO SERVO 	PWM	3 - 6	PWM	DEGREE RANGE 0-180

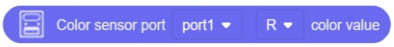
WORK CELL EXAMPLE GUIDE

MAGIC BOX TO MAGICIAN LITE



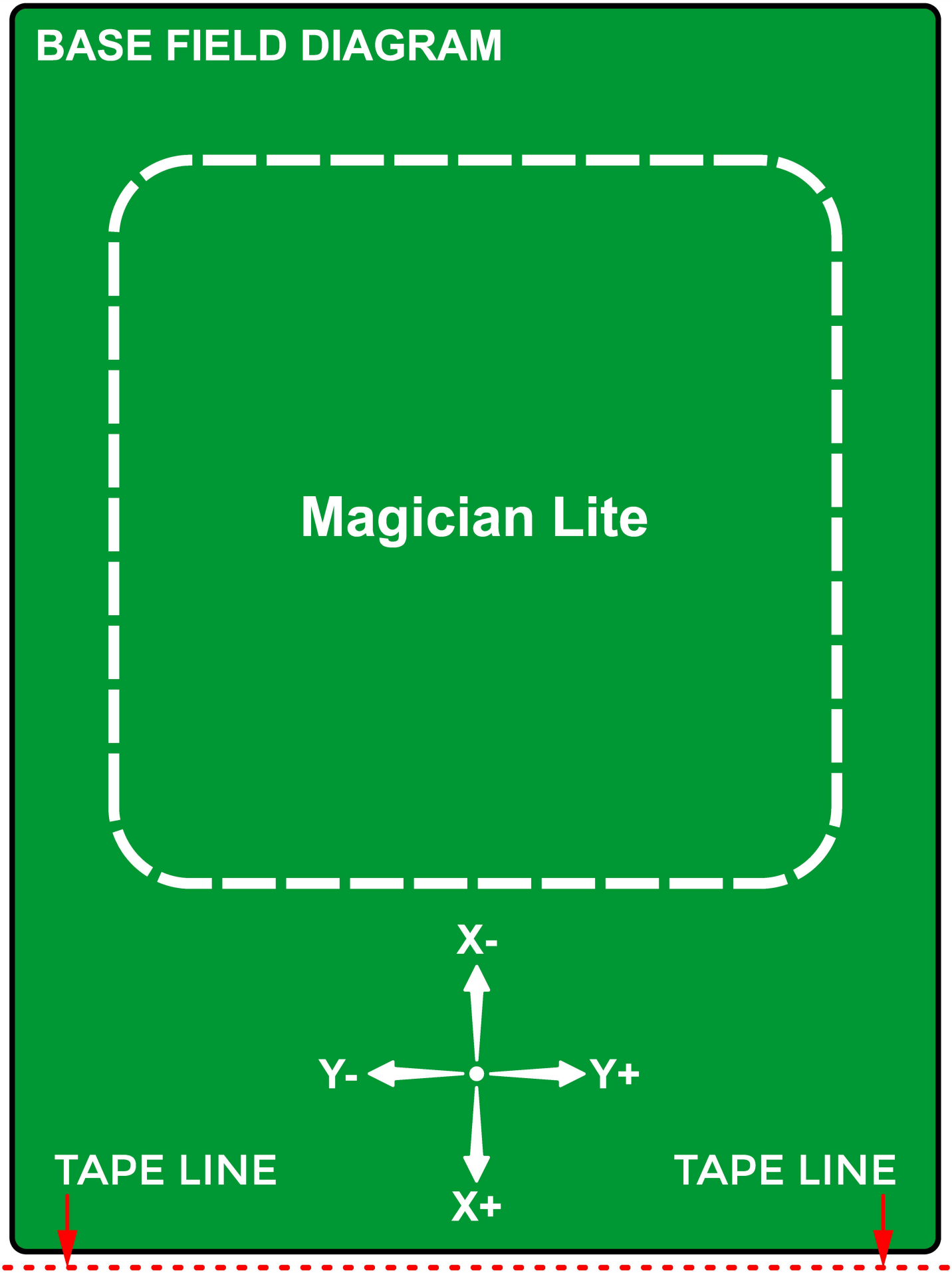
COLOR TEST TEMPLATE



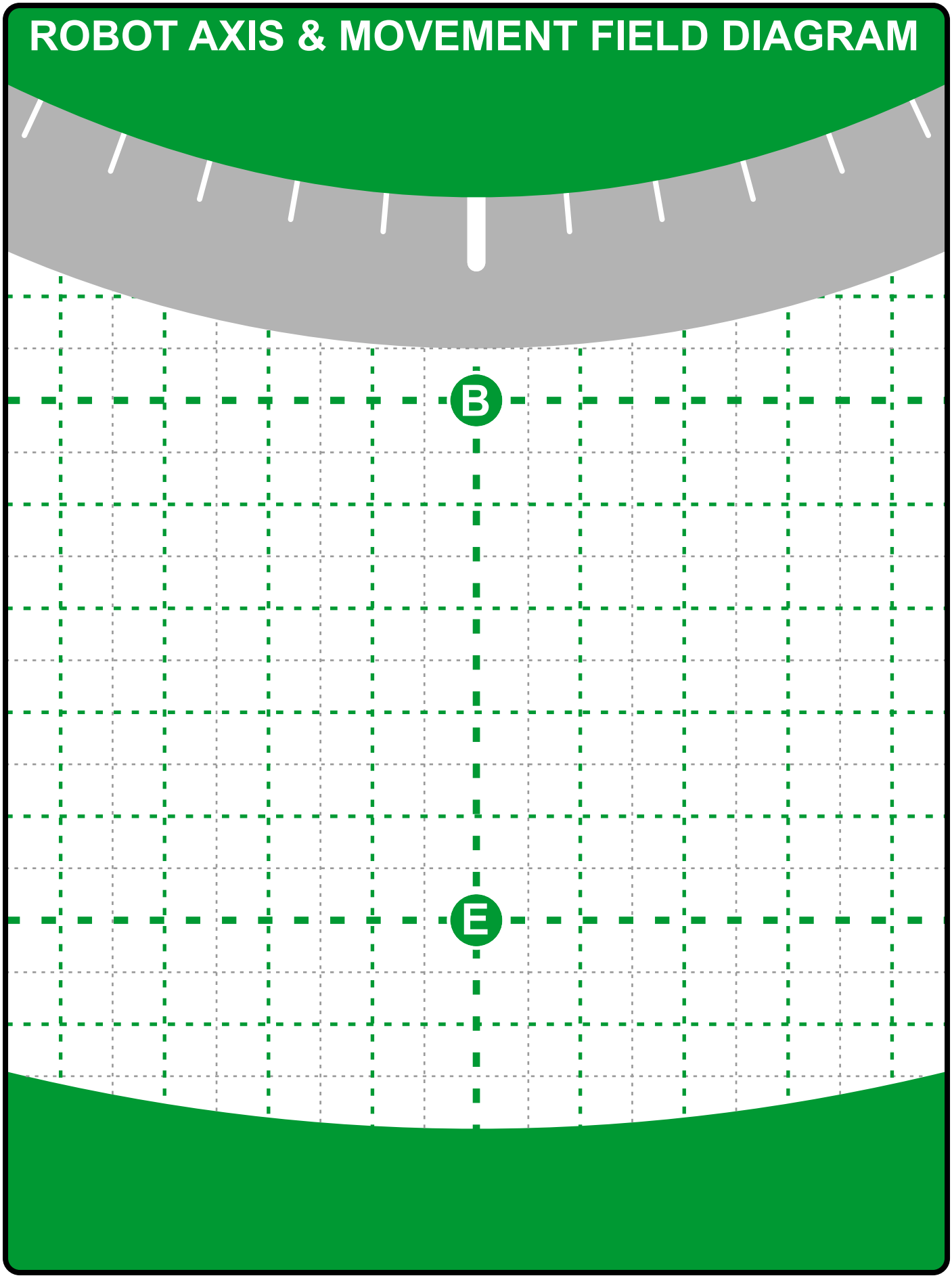


COLOR FOUND?	RGB VALUE REPORTED		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,255) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,0) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,255,0) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,255) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,0) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,102,0) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,255) r - _____ g - _____ b - _____		
✓ **YES / NO** ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,0) r - _____ g - _____ b - _____		

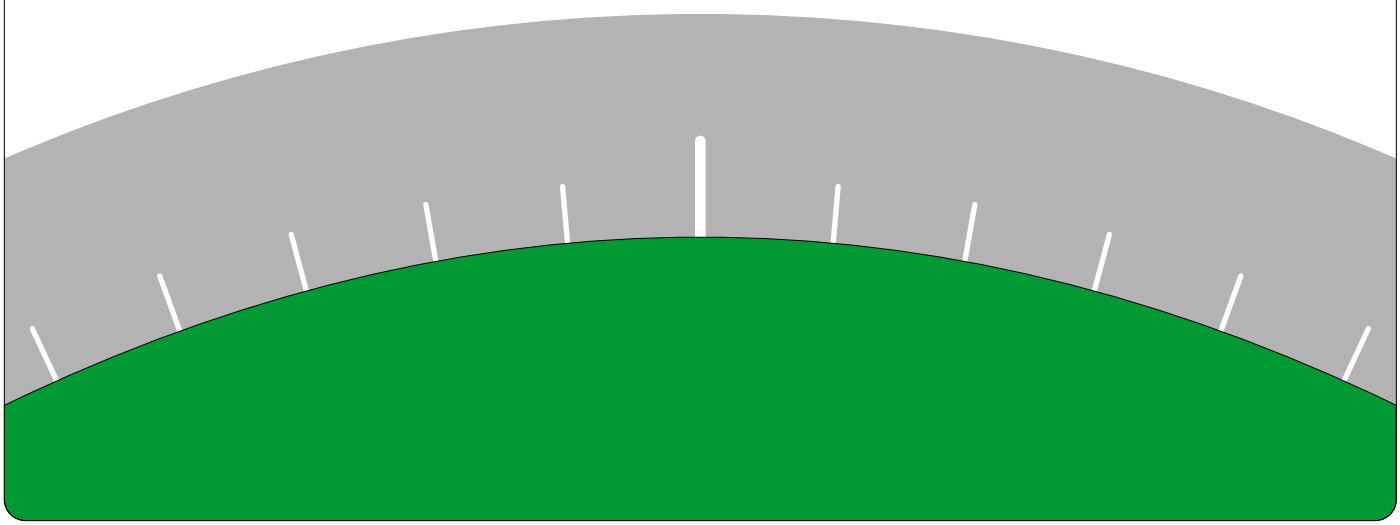
BASE FIELD DIAGRAM



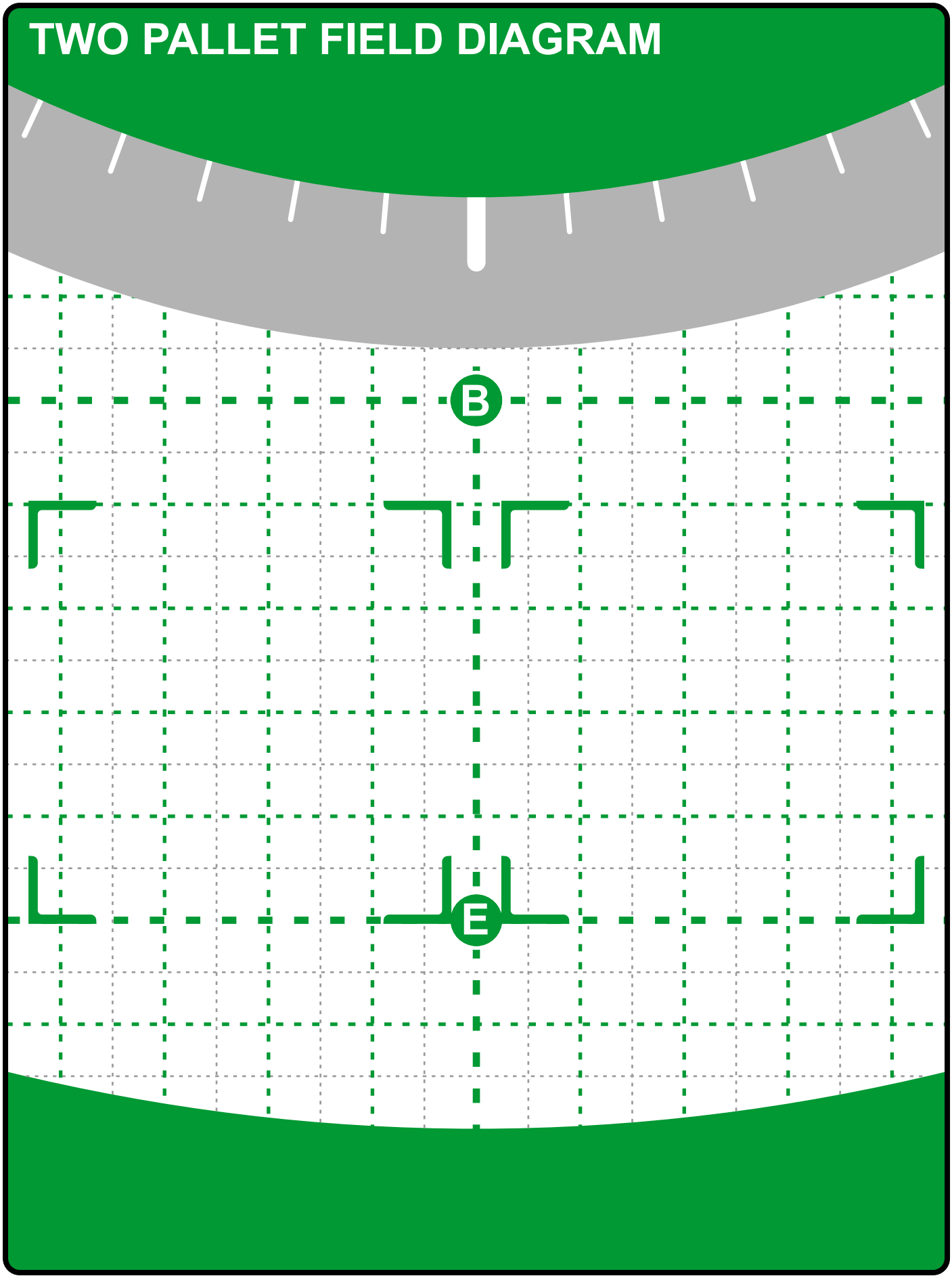
ROBOT AXIS & MOVEMENT FIELD DIAGRAM



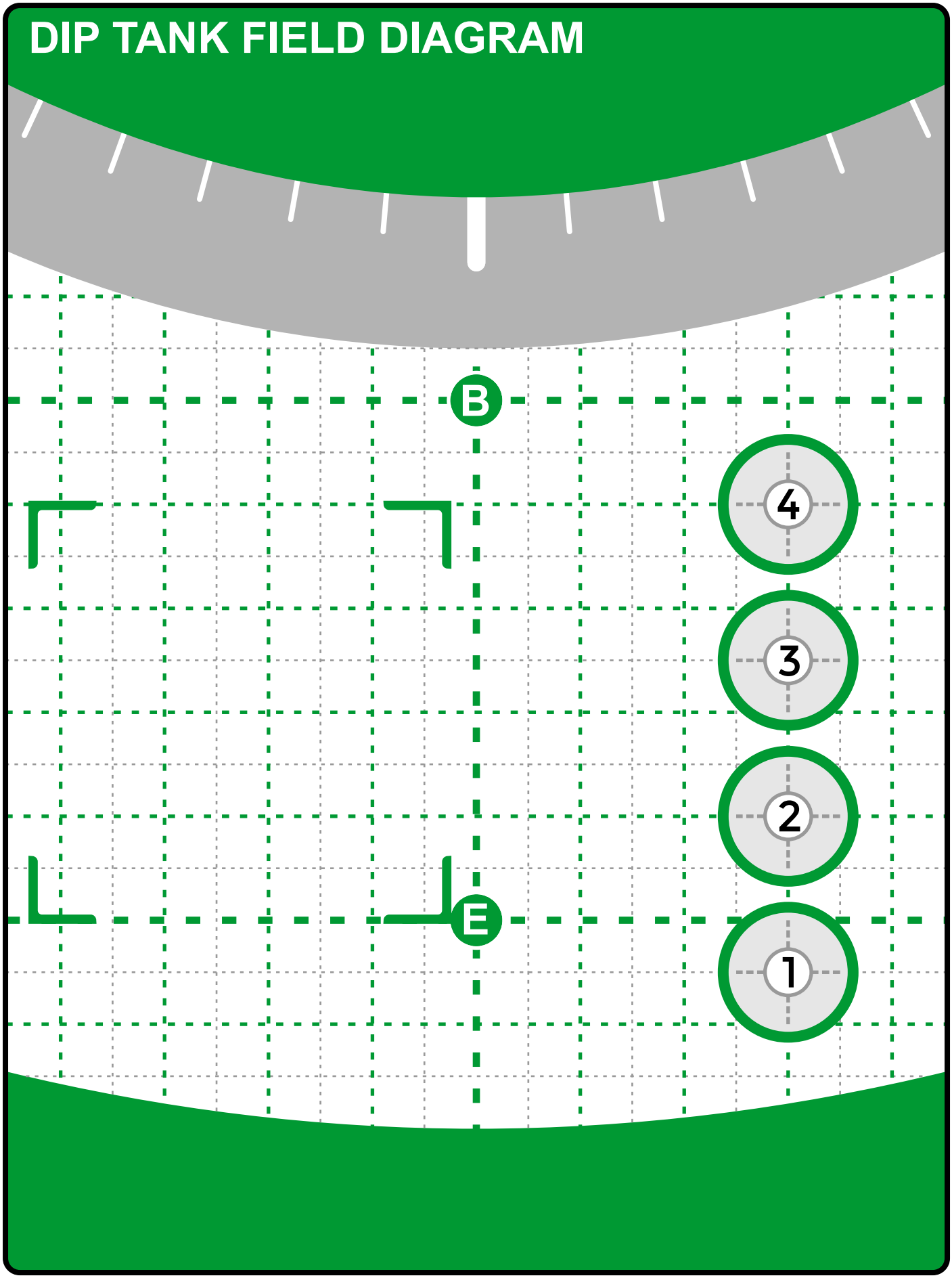
BLANK FIELD DIAGRAM



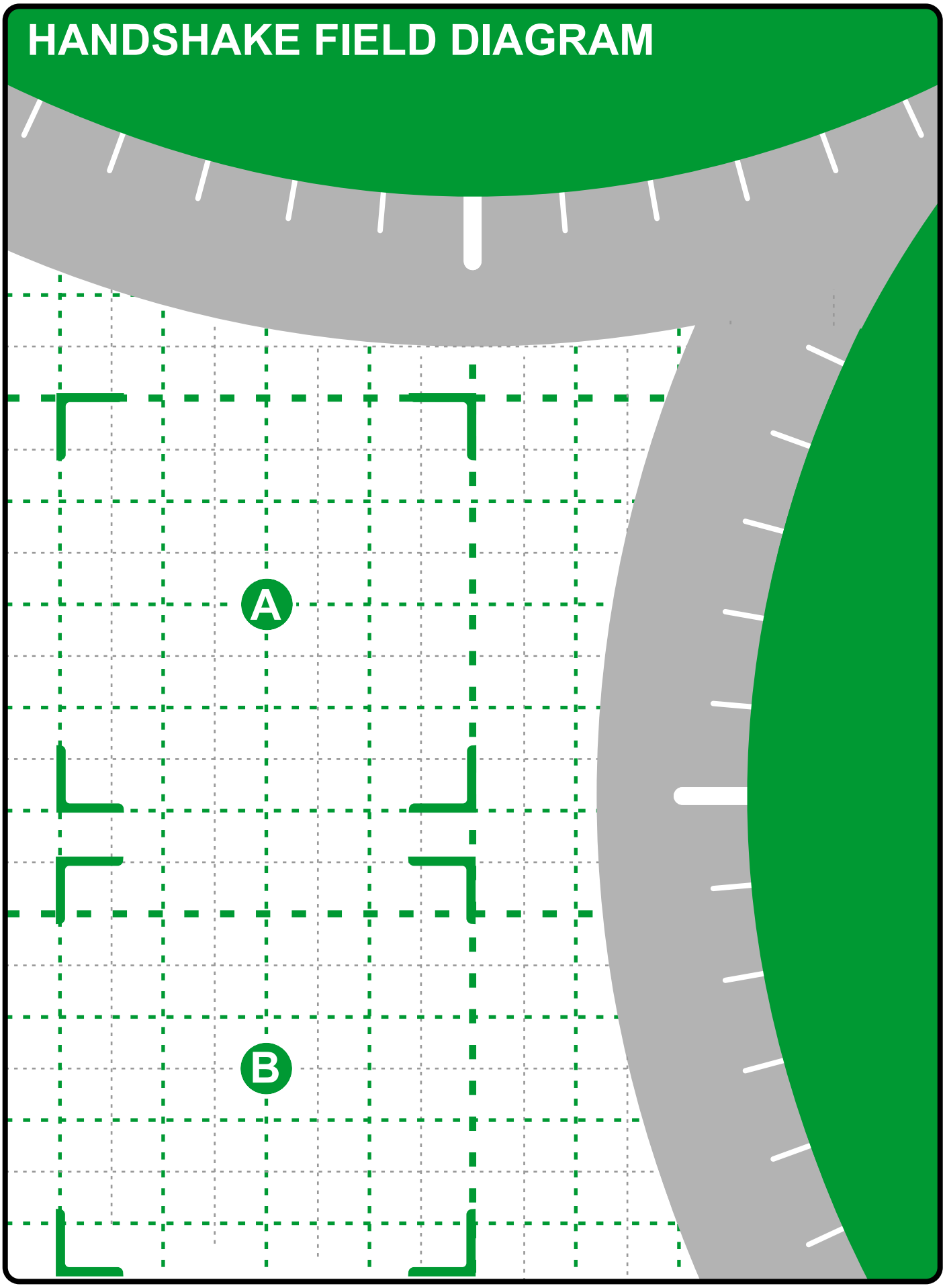
TWO PALLET FIELD DIAGRAM



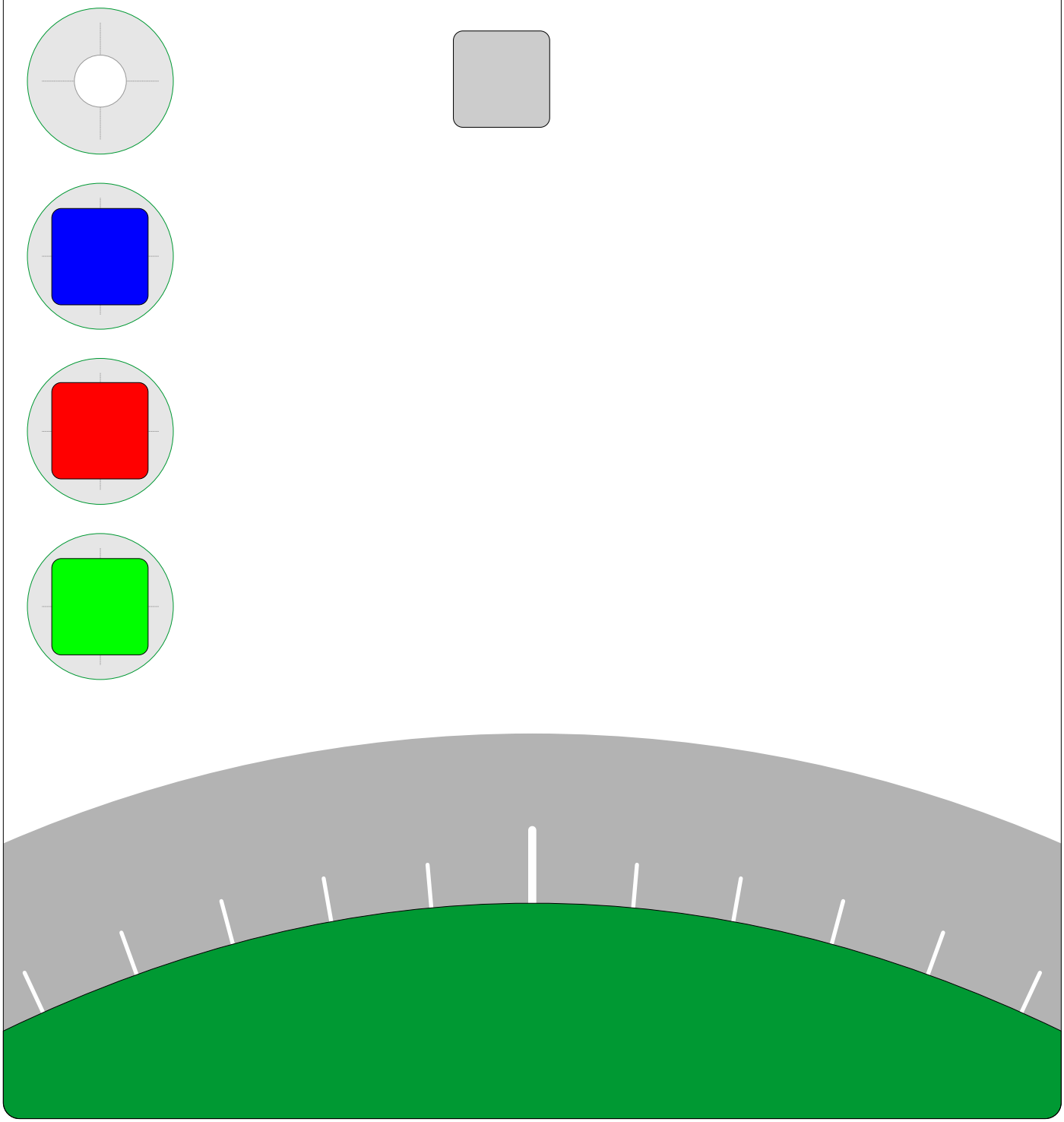
DIP TANK FIELD DIAGRAM



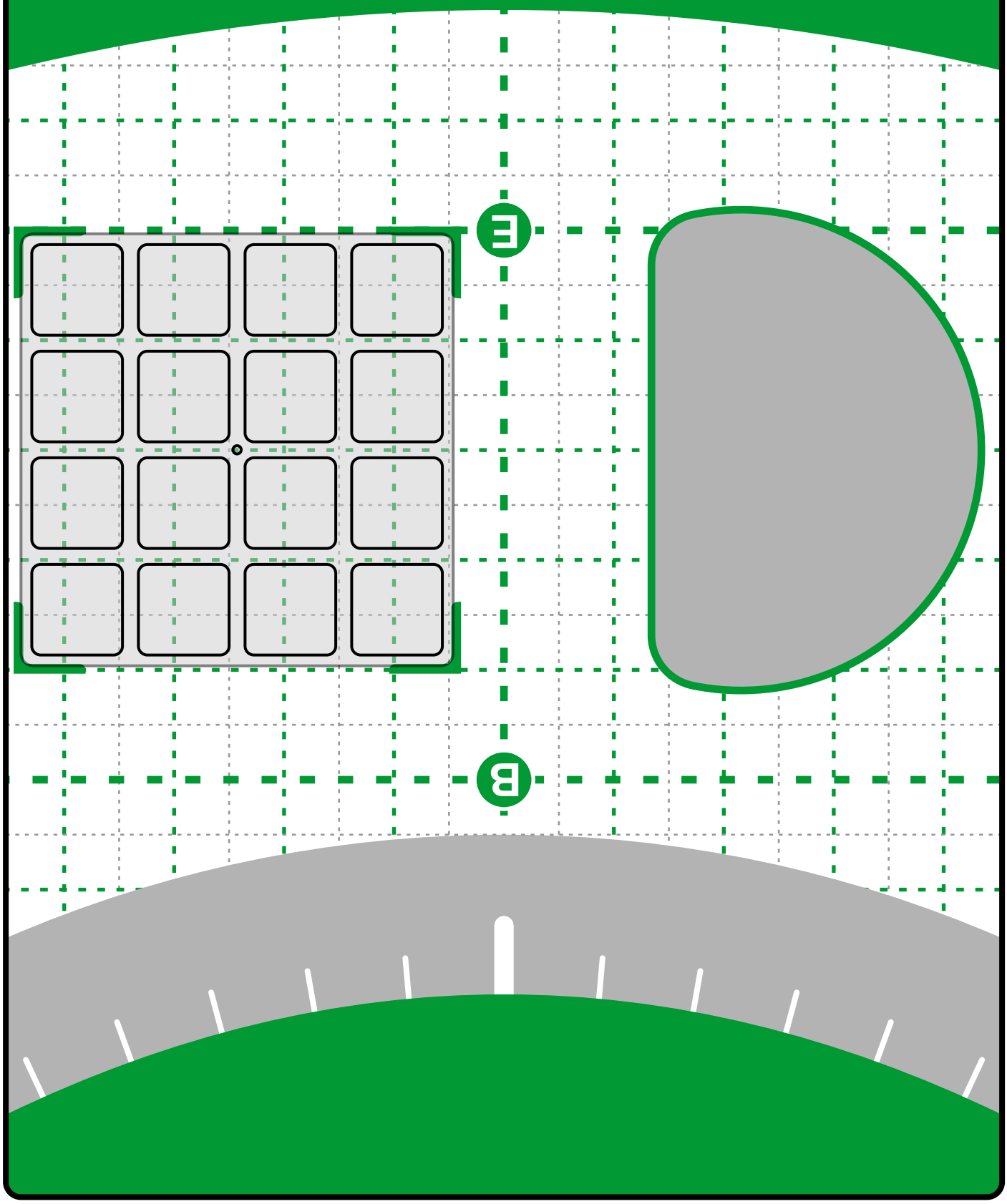
HANDSHAKE FIELD DIAGRAM



CAMERA VISION FIELD DIAGRAM

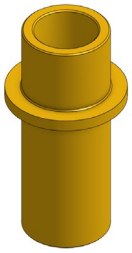
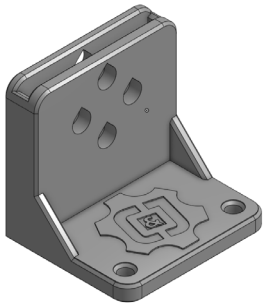
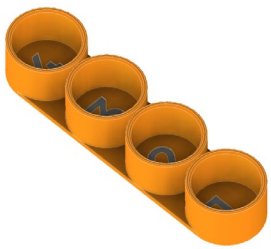


PART FEEDER / PALLET FIELD DIAGRAM

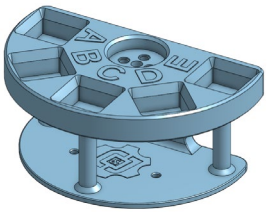


3D Resources for Activities

Below are additional files that can be 3D printed, or laser cut.
Link to CnJ Resource folder: [Click for link to folder]

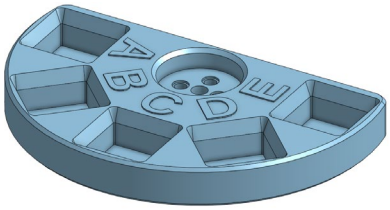
Part	*Definition*
Magician Pen Holder Replacement 	This file is used to 3D print a replacement the internal portion of the existing penholder for the Magician. It has a tapered fit that allows the use of fine-point Sharpies, fine-tip Expo markers, or similar writing tools.
Conveyor Color Sensor Bracket 	This file is used to 3D print a base bracket for the color sensor from the conveyor system.
Magician Dip Tanks – For 1 inch cubes 	This file is used to create three attached dip tanks for 1 in cubes included with the conveyor system.
Magician Lite Dip Tanks – For Small cubes 	This file is used to 3D Print four attached dip tanks for the small cubes included with the Dobot Magician Lite.

Magician Box Servo Motor Feeder

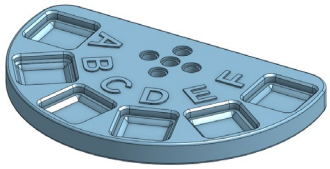


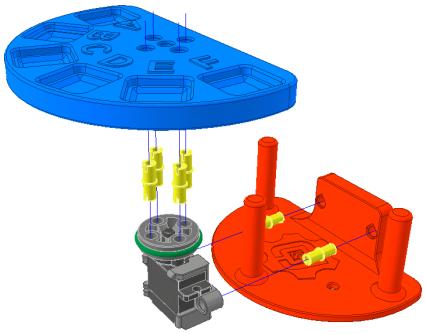
This folder contains all of the files need to build a feeder system using the mini servo from the sensor kits. Lego technic pins are used to mount the top plate to the servo bracket. The pins can also be 3D printed. This folder contains two separate top plates and two separate bases, one for 1 in cubes and one for the cubes included with the Magician Lite.

Large Disc – 1 inch Cubes

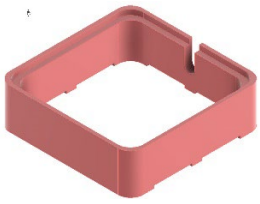


Small Disc – Small Magician Lite Cubes





Magician Base Extender

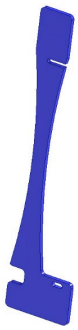


This file is used to 3D Print a riser for the magician base. This base is helpful when using the conveyor system, allows the robot to be homed without running into the conveyor, and allows the vacuum pump to be placed under the magician conserving tabletop space.

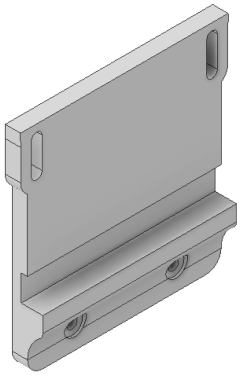
Note: *Also available in two separate parts (Front and Back) if it will not fit on your existing 3D printer bed.*

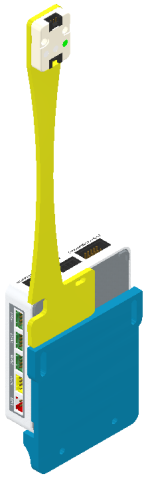
Magic Box LED Module Extension Bracket

This file is used to either 3D print or laser cut a bracket to hold the LED Module at a visible height.



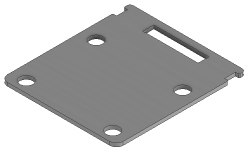
Magic Box Vertical Mounting Plate

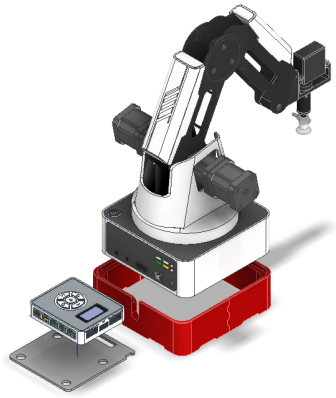


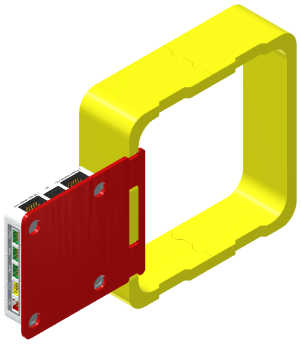


This file is used to 3D print a mounting bracket to vertically mount the magic box. It can be used with the Magic Box LED Module Extension Bracket to create a stack light.

Magic Box Horizontal Mounting Plate





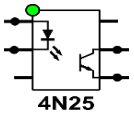


Robotics Glossary

The vocabulary below is used throughout the curriculum and contains general definitions for robotics in industry.

Key Term	*Definition*
Active sensors	Sensors that require an external power supply to operate.
Analog	A type of signal that has a range of values, not just 0 and 1. A good example is an 8-bit light sensor with at least 256 different values.
Audio Feedback	Provides the robot with the ability to use sounds or speech as an output.
Axis Movement	Movement of a robot's wrist in a straight line on the X, Y or Z axis.
Calibration	Calibration is a tool used when creating visual models that allows the user to define a background other than solid white when using a camera with the robot (attempts to ignore background/constant visual information and focus on new elements).
Current limiting resistor	A resistor used in a circuit to limit the current going to a component such as a light emitting diode. (LED)
Conveyor System	A system used in industrial settings to move items around efficiently. Commonly seen in the form of a belt but may also be sets of motorized rubber wheels that can change direction. Can be controlled by stepper motors for positional accuracy.
Degrees of Freedom	The range and flexibility of motion. Each added degree of freedom adds greater range of motion to the robot's tooling within its work area. *Also see Roll, Pitch and Yaw.*
Digital	A signal that can only be read in one of two states: YES or NO, ON or OFF, TRUE or FALSE, HIGH or LOW. There are no values in-between.
Drive system	The power or motors that are used to control the positioning of the robotic arm. Usually electric but may be hydraulic or pneumatic as well.
Electric drive system	A drive system used for high accuracy, repeatability, and speed. Used to control motors, servos, or stepper motors.
End of arm tooling (EoAT)	The tooling added to the end of a robot's arm that allows it to perform specific operations. Also known as an *end effector*.
End Effector	The tooling added to the end of a robot's arm that allows it to perform specific operations. Also known as an *End of Arm Tooling*.
Function	A named section of a program that performs a task. It can also be considered a procedure or a routine and greatly simplifies otherwise complicated programs.

Vocabulary	*Definition*
Gesture Sensor	A 3D sensor that recognizes hand gestures that can be used for human-machine interaction.
Handshaking	Handshaking is defined as safe communication between two or more, like or unlike, pieces of technology.
Home	A position within the robot's work envelope designated as a starting point for the operation.
Humiture Sensor	A sensor used to detect relative humidity and temperature of the environment in which it is placed.
Hydraulic drive systems	A drive system used for larger robots that require large amounts of power. Linear movements are produced by hydraulic pistons while rotary is produced by rotary vanes.
Input	Any data or signals that a robot receives that provides feedback about its environment. Information that is brought "into" the controller. Examples: sensor values, signals from another device or robot, or user commands.
Joint movement	The movement of a robotic arms end effector defined by the movement of its individual joints.
Light Sensor	An analog sensor that detects light using resistance. As light intensity increases, resistance decreases. This results in a change in voltage that can be used to determine light intensity in the environment.
Lights Out Manufacturing	The practice of fully automating a production system in industry that requires no human intervention. Refers to the ability for the system to run overnight, without humans, with the "lights out".
Linear movement	The same as an axis movement; Movement of a robot's end effector in a straight line on the X, Y or Z axis.
Loop	The ability to repeat a given set of instructions or code a specific number of times
Main controller	The brain of the robotic system. Used to control the motion of and programming of the robot as well as control various inputs and outputs and communicate with other elements in a work cell.
Machine Tending	Using a robot to load and/or unload a machine with parts or raw material. An example would be using a robot to put raw materials into a machine, turning it on, and removing finished product for palletization. This is done using a closed loop system.
Magic box	This is Dobot's robotic input/output interface. It allows you to plug in sensors, and outputs that can be used with the Dobot Magician and Macian Lite.
Matrix	A rectangular array of parts, made of a finite number of rows and columns. a good example is an array of boxes on a pallet.
Microservo	A small electric actuator with accurate positioning through at least 180 degrees.
Microwait	A very short delay written into a program so that some event isn't missed. For example: when a switch moves from high to low quickly.
Microcontroller	A microcontroller is a small computer, usually on a single chip. They are used to control many things in industry. Examples of microcontrollers are Arduino, VEX, and PLC's

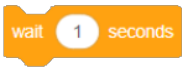
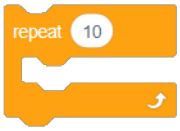
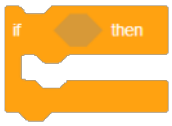
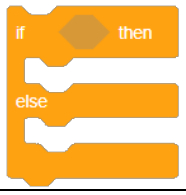
Vocabulary	*Definition*
Nesting	Making parts fit most efficiently on a pallet or in a given space.
Normally closed (NC)	Digital sensor, like a switch where a signal is received until the sensor is triggered. Switch not pressed = ON (1), Pressed = OFF (0). Useful in handshaking when you want something to happen when an input signal turns off.
Normally open (NO)	Digital sensor, like a switch where a signal is received until the sensor is triggered. Switch not pressed = OFF (0), Pressed = ON (1). Useful in handshaking when you want something to happen when an input signal turns on.
Optical isolator	An electronic device that separates two unlike voltages or currents in a robotic communication system. It accomplishes this using an infrared LED and a light sensor embedded in a device. 
Output	An outgoing device or signal that is controlled by the robot. An output is often something that the robot can turn on or off. An output can also be a value that is produced by the robot's programming. Some examples are motors, lights, and displays.
Palletizing	An automated process that uses a robot to load and unload parts, containers, or boxes onto pallets for storage and shipping. Removing boxes or parts from a pallet is known as de-palletization.
Passive sensor	Sensors that detect inputs from the physical environment without external power.
Payload	The size and weight of the material that a robotic arm can safely lift.
Photoelectric Sensor	A digital sensor that can detect the presence or absence of an object. Also known as a *proximity switch*.
Pick and place	When a robot retrieves parts from one location and consistently places them in a new location.
Picture Recognition	The process of identifying a feature or an object in a picture/image. Also known as image recognition.
PIR Sensor	The PIR sensor is a body infrared sensor. When the sensor detects infrared, it outputs a 1 (ON) signal until the triggering signal disappears and outputs a 0 (OFF).
Pitch	The degree of freedom of a robotic arms end effector around its Y axis.
Pneumatic drive system	A drive system used in smaller robots. Used to control rotary actuators or sliding joints. Typically used for high-speed operations and limited movements.
Programable Logic Controller	It is a simple electronic device that allows very precise control of machinery and allows multiple devices to communicate. Also known as a *PLC*.
Pull-down resistor	A resistor used in an electrical circuit to force a signal to be low when not activated (0v). This prevents a signal from floating between a 1(5v) and 0(0v) in a digital circuit.
Pull-up resistor	A resistor used in an electrical circuit to force a signal to be High (5v) when not activated. This prevents a signal from floating in between a 1(5v) and 0(0v) in a digital circuit.
Ramping	To accelerate or decelerate the speed with which a robot moves over a short period of time.

Vocabulary	*Definition*
Record	To save a position with a robot arm by moving the robot to that position and then storing the values of all the robot's joints.
Relative coordinates	A cartesian coordinate system where each successive point is relative to the point before it. Also known as *incremental*.
Relative positions	A system of recording positions where each point is relative in X, Y, and Z coordinates to all the other points in a program.
Robot accuracy	How close a robot's actual movement is to a given position.
Robot repeatability	A robot's ability to return to a given position multiple times.
Robotic arm	The parts of a robot that are used to locate and position the end of arm tooling. Usually includes multiple joints, and axis of motion.
Roll	The degree of freedom of a robotic arms end effector around its Z axis
Roll angle	Adjusting this angle allows a robot's end of arm tooling to line up and nest parts on a pallet.
Sensor	Inputs that provide feedback and bring in a variety of data into the robot. Commonly used for automation of tasks.
Sliding Rail	The slide rail is a peripheral that allows the robot to slide from side to side. It is controlled by the robot through a stepper motor and is very accurate, fast, and powerful.
Stack Light	A set of lights used in industry to indicate the status of a machine or the operation it is performing. Provides visual, and sometimes audible indicators for operators and workers.
Stepper Motor	An electric motor that moves in small steps instead of continuously. These are used in instances where you need precise or powerful movements. A specific position can be held as well. In the activities within this curriculum, the conveyor belt and the linear slide rails are both controlled through the use of stepper motors.
Suction	The gripper or end of arm tooling that uses vacuum and a flexible suction cup to pick up parts easily.
Teach	To save a position with a robot arm by typing in an X, Y and Z value and storing the position.
Teach pendant	A handheld device used to manually control, program, and troubleshoot a robotic arm without the need for a full control terminal.
Tool center point	The X, Y and Z value recorded or taught in robotics that represents where the work is being done in space by the robot's end of arm tooling. (TCP)
Touch up	To easily correct or round saved points in a robotics program by manually adjusting X, Y, and Z values in the control software.
Training Model	Training model is a process of testing camera/photo data that has been stored in user-defined classifications.
Vacuum gripper	An end effector that creates negative pressure to allow a robot to pick up an object easily with a suction cup type device.

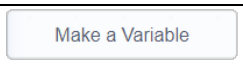
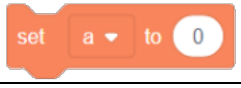
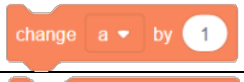
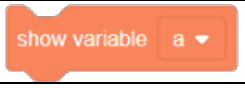
Vocabulary	*Definition*
Variable	A changeable quantity in a program that can be represented by a word or a letter. Variables can be assigned, changed, or referenced throughout a program and makes programming more efficient.
Workcell	The complete environment around a robot. May include tools, machines and/or other robots. Also known as a *manufacturing cell*.
Work envelope	Defined as the range and area each robot may work within.
Yaw	The degree of freedom of a robotic arms end effector around its Y axis

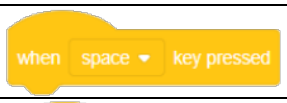
Dobot Block Glossary

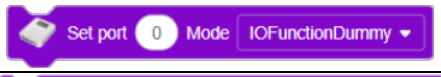
The vocabulary below is used throughout the blockly activities, lessons, and resources for DobotLab software. Please note that not all commands are defined here, just the ones most important to completing the activities. See the Help Section in DobotLab for other definitions.

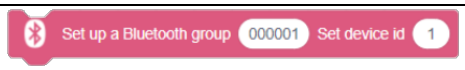
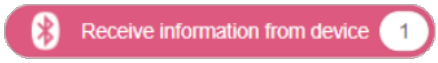
Control Menu		
Function	**Block**	**Description**
Wait		Wait for a specific amount of time. If it is less than one second, it can be considered a micro-wait.
Repeat # of times		Repeat a block of code a specific number of times.
Repeat Forever		Repeat a block of code forever. Also known as a forever loop.
If-Then		Execute a block of code only if the specified condition is true.
If-Then-Else		Executes the first block of code as long as the condition is true, otherwise execute the second block of code.
Wait Until		Execute a block of code if the specified condition is true.
Repeat Until		Execute s a block of code repeatedly until the specified condition is true.
Stop		Stops execution of a block of code.

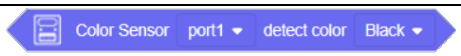
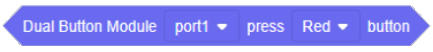
Operators Menu		
Function	**Block**	**Description**
Addition		Execute an addition operation.
Subtraction		Execute a subtraction operation.
Multiplication		Execute a multiplication operation.
Division		Execute a division operation.
Get Random Number		Generate a random number within a specified range.
A Greater Than B		If the value of the first parameter A exceeds the specified value B then the result is True.
A Less Than B		If the value of the first parameter A is less than the specified value B then the result is True.
A Equals B		If the value of the first parameter A is equal to the specified value B then the result is True.
A And B		If A and B are both True at the same time, then the result is True.
A Or B		If either A or B are True then the result is True.
Not A		If the condition is False then return a True.
String Contains		If the condition of a string value (input) contains any of the words or characters return a True value
Camera Classification Development		Creates and saves visual data (models) in defined classifications.

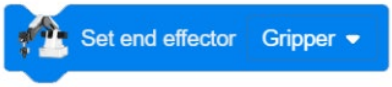
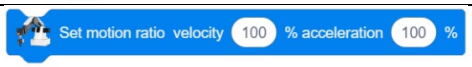
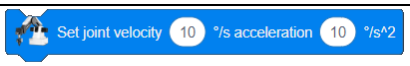
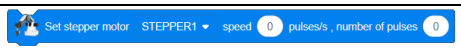
Variables Menu		
Function	**Block**	**Description**
Make a Variable		Make a variable to get, store, or display data.
Set Variable		Set a variable to a specified number.
Increase Variable		Increase the value of a variable.
Show Variable		Display the value of a variable.

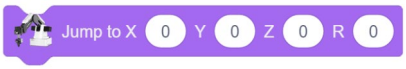
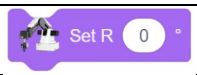
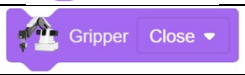
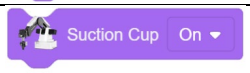
Events Menu		
Function	**Block**	**Description**
When Green Flag Clicked		When the green flag is clicked, operate the blocks below. This is used to start a program.
When Key is Pressed		When the specified key is clicked, operate the blocks below.
When Message Received		When a specified message is received, operate the blocks below. Used for communications between different parts of a program or even different devices.
Broadcast Message		Send a message to another part of the program, or another attached device. Used for communications between different parts of a program or even different devices.
Broadcast Message & Wait		Send a message to another part of the program, or another attached device and wait until the program that it executes is finished.

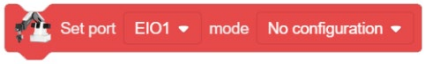
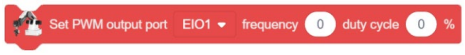
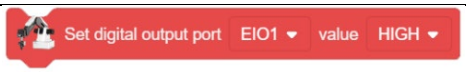
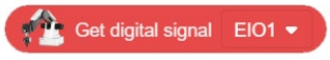
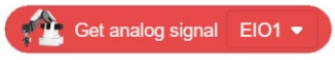
Magic Box Menu		
Function	**Block**	**Description**
Output Digital Signal		Send a digital signal, a 1 or a 0, on the specified port.
7Set Port Status		Allows you to set the port type. Examples are: PWM, ADC, and others.
Set PWM Output		Allows you to change the output of the PWM port by changing the frequency and the duty cycle.
Get Digital Signal		Read the digital signal on the given port.
Get Analog Signal		Read the analog signal on the given port.
Set Stepper Speed		Set the speed, in pulses per second, of the given stepper motor.
Set Conveyor Speed		Set Speed of the conveyor system in millimeters per second.

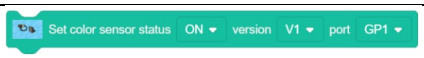
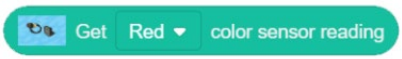
Wireless Menu		
Function	**Block**	**Description**
Set Up Bluetooth		Allows you to set up a six-digit Bluetooth group.
Receive Message		Receive information from a device in a Bluetooth group.
Send Message		Send information to a device in a Bluetooth group.

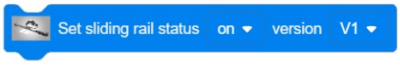
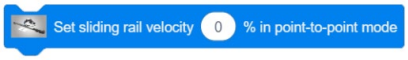
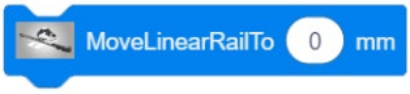
AI Sensor Menu – Added Extension		
Function	**Block**	**Description**
Set LED		Sets the RGB Value of the LED.
Set LED Brightness		Sets the RGB Value of the LED and controls the brightness.
Set LED Status		Sets the LED to an ON or OFF state. Allows for choice of port, and LED number 1, 2, or 3.
Get Potentiometer Value		Returns the analog value of the knob potentiometer.
Get Light Value		Gets the analog value of the light sensor.
Get Sound Value		Returns the analog value of the sound sensor.
Get Temp & Humidity		Returns the analog value of the temperature (C) or the relative humidity. (%)
Get Color Value		Returns the color value of the color sensor.
Detect What Color		Determines what color the color sensor has detected.
Detect Specific Color		Look for a specified color with the color sensor.
Photoelectric Sensor Status		Determines if there is an object (1) or not (0) within its range.
Gesture Sensor Status		Looks for a specific Gesture: left, right, forward, backward, clockwise, counterclockwise, up, or down.
Dual Button Status		Look for a specific button to be pressed, either red or blue. Digital sensor.
Set Servo Angle		Sets the servo at a specific angle between 0 and 180. Analog Sensor.

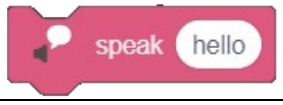
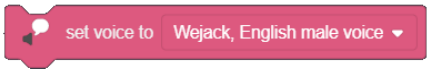
Settings Menu		
Function	**Block**	**Description**
Set End Effector		Tells the program which end effector is being used: Gripper, Suction cup or Pen.
Set Motion Ratio		Sets both acceleration and velocity of the robot's moves. Uses %
Set Joint Speed		Sets the velocity of the robot's joint motion in degrees per second^2.
Set Jump Height		Sets the height of all jump moves in millimeters.
Set Conveyor		Set's the speed of the conveyor's stepper motor in pulses per second.
Set Stepper Speed		Sets the stepper motor's speed in pulses per second.
Set Stepper Speed & Pulse		Sets the stepper motor's speed in pulses per second, and number of pulses.

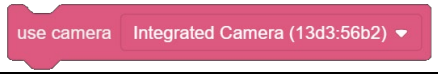
Motion Menu		
Function	**Block**	**Description**
Jump To Position		Jump to a given position. XYZ coordinate with rotation in degrees.
Go to Position		Go to an XYZ position, and move either in a straight line, or a joint move.
Move Relative		Makes the robot arm move relative to the position of where it is on the XYZ axis.
Move Joints		Makes the robot arm move each joint a certain number of degrees.
Set Rotate		Set wrist rotation angle in degrees.
Set Gripper		Turn the gripper on or off.
Set Suction Cup		Turn the suction cup on or off.

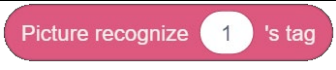
I/O Menu		
Function	**Block**	**Description**
Set Port		Set Digital I/O ports 1-20 to High (1) or Low (0).
Set PWM (Servo)		Sets the frequency and the duty cycle for a pulse width (PWM) modulation port. Used to set an output for a servo.
Set Output		Sets an I/O port's output to high (1) or Low (0)
Get Digital Signal		Get an incoming digital signal on a given I/O port 1-20.
Get Analog Signal		Get an incoming analog signal on a given I/O port 1-20.

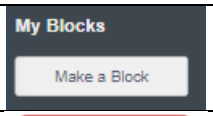
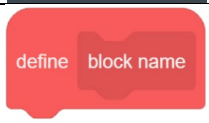
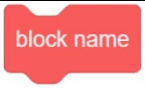
Photoelectric/Color Sensor Menu – Added Extension		
Function	**Block**	**Description**
Set IR Sensor		Sets the state of the infrared sensor to ON (1) or Off (0).
Get IR Reading		Get the digital signal from the infrared sensor. ON (1) or Off (0).
Set Color Sensor		Sets the status of the color sensor to On or Off.
Get Color Sensor Reading		Get a reading from the color sensor when it is Red, Blue, or Green.

Sliding Rail Menu – Added Extension		
Function	**Block**	**Description**
Sliding Rail Status		Sets the status of the sliding rail to off or on.
Set Sliding Rail Speed		Allows sliding rail speed to be set using a percentage of full speed.
Move Linear Rail		Allows linear rail to move to a given position. Measured in millimeters from home.

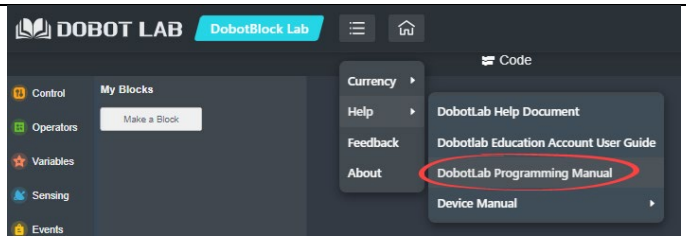
AI Menu: Text to Speech – Added Extension		
Function	**Block**	**Description**
Speak		Allows user input of text the robot will convert to speech.
Set Voice		Allows user input of language and voice characteristics in strings that the robot speaks.

AI Menu: Camera – Added Extension		
Function	**Block**	**Description**
Use Camera		Allows user input as to which camera the robot uses.

AI Menu: Image Recognition – Added Extension		
Function	**Block**	**Description**
Picture Input		Allows the camera to take and store a temporary photo that can be compared to stored models.
Picture Recognition		Allows the user to compare data to stored models.

My Blocks Menu		
Function	**Block**	**Description**
Make a Block		Allows you to make your own block of code by making one block out of many.
Define block		The code attached to this determines what the block does. It is user defined.
Block Name		Gets the My Block and runs its code within the main program. Can be used multiple times.

*Please remember that all programs are dynamic and change continuously. Therefore, some blocks may go away, and more may be added. To get the latest block definitions go to the **Help Menu** in **DobotLab** and look for the **DobotLab Programming Manual** and check out the **DobotBlock** section.*



Curriculum Standards

The standards defined below are far reaching and very large in scope. They include the following standards:

- Standards for Technological Literacy
  https://www.iteea.org/File.aspx?id=67767
- Next Generation Science Standards
  http://www.nextgenscience.org/ pg 102
- ISTE (International Society for Technology in Education
  https://cdn.iste.org/www-root/PDF/ISTE%20Standards-One-Sheet_Combined_09-2021_vF3.pdf

All the activities in this curriculum, once completed, can cover all the standards outlined below, and most of them multiple times. Having students working with Google docs and sharing, or another local high school, will cover even more and give students a more global approach to learning.

Standards for Technological Literacy	
The Standards for Technological Literacy (STL) were developed by the International Technology and Engineering Educators Association (ITEEA) and are available as a complete download for free here: https://www.iteea.org/File.aspx?id=67767	
2-W	*Standard:* Students will develop an understanding of the core concepts of technology. *Benchmark:* Systems thinking applies logic and creativity with appropriate compromises in complex real-life problems.
2-Z	*Standard:* Students will develop an understanding of the core concepts of technology. *Benchmark:* Selecting resources involves trade-offs between competing values, such as availability, cost, desirability, and waste.
2-AA	*Standard:* Students will develop an understanding of the core concepts of technology. *Benchmark:* Requirements involve the identification of the criteria and constraints of a product or system and the determination of how they affect the final design and development.
2-BB	*Standard:* Students will develop an understanding of the core concepts of technology. *Benchmark:* Optimization is an ongoing process or methodology of designing or making a product and is dependent on criteria and constraints.

8-H	*Standard:* Students will develop an understanding of the attributes of design. *Benchmark:* The design process includes defining a problem, brainstorming, researching and generating ideas, identifying criteria and specifying constraints, exploring possibilities, selecting an approach, developing a design proposal, making a model or prototype.
8-I	*Standard:* Students will develop an understanding of the attributes of design. *Benchmark:* Design problems are seldom presented in a clearly defined form.
8-J	*Standard:* Students will develop an understanding of the attributes of design. *Benchmark:* The design needs to be continually checked and critiqued, and the ideas of the design must be redefined and improved.
8-K	*Standard:* Students will develop an understanding of the attributes of design. *Benchmark:* Requirements of a design, such as criteria, constraints, and efficiency, sometimes compete with each other.
9-I	*Standard:* Students will develop an understanding of engineering design. *Benchmark:* Established design principles are used to evaluate existing designs, to collect data, and to guide the design process.
9-J	*Standard:* Students will develop an understanding of engineering design. *Benchmark:* Engineering design is influenced by personal characteristics, such as creativity, resourcefulness, and the ability to visualize and think abstractly.
9-K	*Standard:* Students will develop an understanding of engineering design. *Benchmark:* A prototype is a working model used to test a design concept by making actual observations and necessary adjustments.
9-L	*Standard:* Students will develop an understanding of engineering design. *Benchmark:* The process of engineering design takes into account a number of factors.
11-N	*Standard:* Students will develop the abilities to apply the design process. *Benchmark:* Identify criteria and constraints and determine how these will affect the design process.
11-O	*Standard:* Students will develop the abilities to apply the design process. *Benchmark:* Refine a design by using prototypes and modeling to ensure quality, efficiency, and productivity of the final product.
11-P	*Standard:* Students will develop the abilities to apply the design process. *Benchmark:* Evaluate the design solution using conceptual, physical, and mathematical models at various intervals of the design process in order to check for proper design and to note areas where improvements are needed.
11-Q	*Standard:* Students will develop the abilities to apply the design process. *Benchmark:* Develop and produce a product or system using a design process.

11-R	*Standard:* Students will develop the abilities to apply the design process. *Benchmark:* R. Evaluate final solutions and communicate observation, processes, and results of the entire design process, using verbal, graphic, quantitative, virtual, and written means, in addition to three-dimensional models.
12-P	*Standard:* Students will develop the abilities to use and maintain technological products and systems. *Benchmark:* Use computers and calculators to access, retrieve, organize, process, maintain, interpret, and evaluate data and information in order to communicate.
Next Generation Science Standards	
The **Next Generation Science Standards** is a multi-state effort to create new education **standards** that are "rich in content and practice, arranged in a coherent manner across disciplines and grades to provide all students an internationally benchmarked **science** education." More information can be found here: http://www.nextgenscience.org/ pg 102	
HS.ETS1.2	Engineering Design
	Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
HS.ETS1.3	Engineering Design
	Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.
DCI - ETS1.B	Engineering Design - Developing Possible Solutions
	When evaluating solutions, it is important to take into account a range of constraints, including cost, safety , reliability , and aesthetics, and to consider social, cultural, and environmental impacts. (HS-ETS1-3)
DCI - ETS1.C	Engineering Design - Optimizing the Design Solution
	Criteria may need to be broken down into simpler ones that can be approached systematically, and decisions about the priority of certain criteria over others (tradeoffs) may be needed. (secondary to HS-PS1-6)
	Science and Engineering Practice - Planning and Carrying Out Investigations
	Plan and conduct an investigation or test a design solution in a safe and ethical manner including considerations of environmental, social, and personal impacts.
	Science and Engineering Practice - Using Mathematics and Computational Thinking.
	Create and/or revise a computational model or simulation of a phenomenon, designed device, process, or system.

	Crosscutting Concepts - Systems and System Models
	• A system is an organized group of related objects or components; models can be used for understanding and predicting the behavior of systems. • Systems can be designed to do specific tasks. • When investigating or describing a system, the boundaries and initial conditions of the system need to be defined and their inputs and outputs analyzed and described using models. • Models (e.g., physical, mathematical, computer models) can be used to simulate systems and interactions—including energy, matter, and information flows—within and between systems at different scales. • Models can be used to predict the behavior of a system, but these predictions have limited precision and reliability due to the assumptions and approximations inherent in models.
	Connections to Engineering, Technology, and Applications of Science
	Influence of Science, Engineering, and Technology on Society and the Natural World New technologies can have deep impacts on society and the environment, including some that were not anticipated. Analysis of costs and benefits is a critical aspect of decisions about technology. (HS-ETS1-1) (HSETS1-3)
ISTE (The International Society for Technology in Education) Standards	
The ISTE Standards serve as a framework for innovation and excellence in learning, teaching and leading. As a body of work, the suite of standards has guided educator practice, school improvement planning, professional growth and advances in curriculum. More information can be found here: https://cdn.iste.org/www-root/PDF/ISTE%20Standards-One-Sheet_Combined_09-2021_vF3.pdf	
1.1	**Empowered Learner** Students leverage technology to take an active role in choosing, achieving, and demonstrating competency in their learning goals, informed by the learning sciences.
	1.1a Build networks and customize their learning environments in ways that support the learning process.
	1.1b Use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
	1.1.d. Understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.3	**Knowledge Constructor** Students critically curate a variety of resources using digital tools to construct knowledge, produce creative artifacts and make meaningful learning experiences for themselves and others. Students:
	1.3.c. curate information from digital resources using a variety of tools and methods to create collections of artifacts that demonstrate meaningful connections or conclusions
	1.3.d. build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions
1.4	**Innovative Designer** Students use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions. Students:
	1.4.a. know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems
	1.4.b. select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
	1.4.c. develop, test and refine prototypes as part of a cyclical design process.
	1.4.d. exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems
1.5	**Computational Thinker** Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions. Students:
	1.5.a. formulate problem definitions suited for technologyassisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
	1.5.b. collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
	1.5.c. break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving
	1.5.d. understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6	**Creative Communicator** Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats and digital media appropriate to their goals. Students:
	1.6.a. choose the appropriate platforms and tools for meeting the desired objectives of their creation or communication
	1.6.c. communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations
1.7	**Global Collaborator** Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally. Students:
	1.7.b. use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.
	1.7.c. contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal
	1.7.d. explore local and global issues and use collaborative technologies to work with others to investigate solutions.

For more information or questions, (or compliments!) please contact us. +1 (415) 702-3033 | support@RobotLAB.com
RobotLAB Inc.