How to configure, program and simulate with MachineLogic

Contents

Introduction

This guide covers how to configure a design and program an application using MachineLogic.
MachineLogic provides a development environment to create customized applications using either code-free or Python programming. MachineLogic also provides simulation and deployment tools dedicated to the creation of applications that can be easily deployed to the MachineMotion Controller.
To learn more about how to deploy applications on the MachineMotion controller, click here.
Throughout this document, we will be using this design as an example.

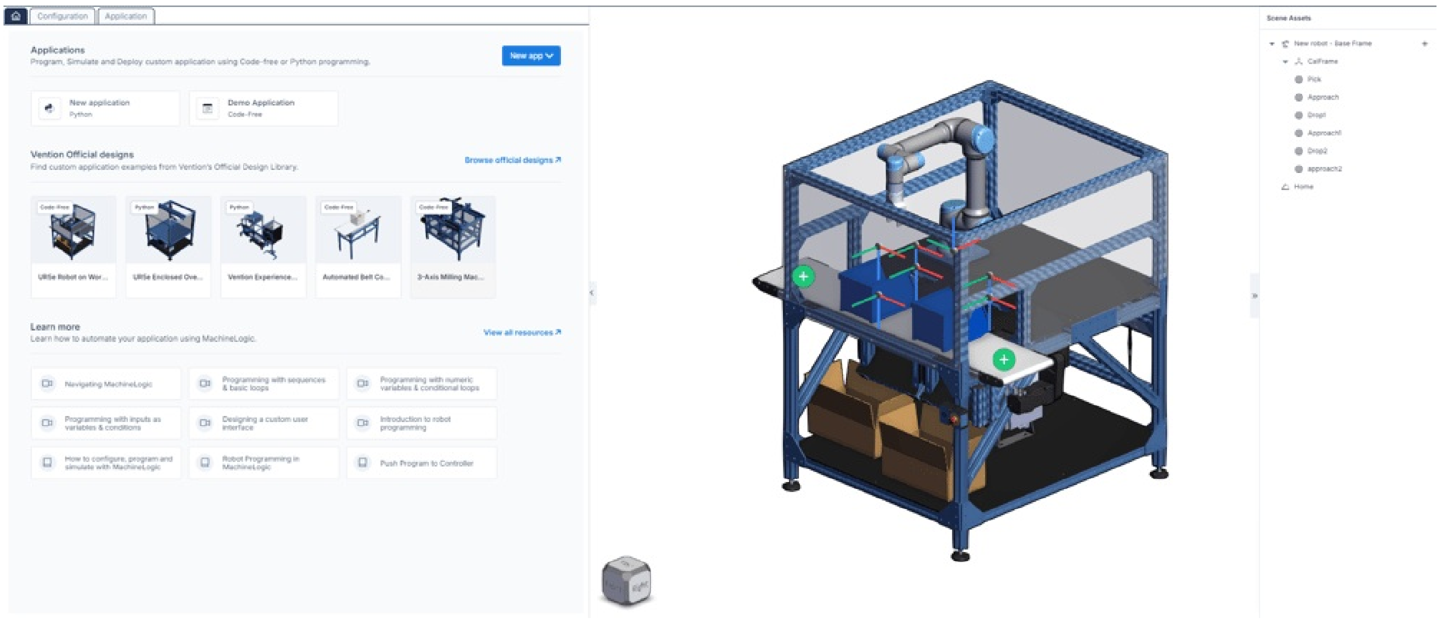
Configuration

Prerequisite: In order to configure and program in MachineLogic, the design must contain at least:

- A MachineMotion Controller
- One of the following component types:
  - Linear Motion
  - Rotary Motion
  - Pneumatics
  - Material Handling
  - Control and Motors
  - Sensors
  - Robots

To learn more about Robot Programming within MachineLogic, click here

To begin, select the configuration tab from MachineLogic’s landing page:



Applications

Program, Simulate and Deploy custom application using Code-free or Python programming.

Vention Official designs

Find custom application examples from Vention's Official Design Library.

Learn more

Learn how to automate your application using MachineLogic.

Figure 1: MachineLogic Landing pager

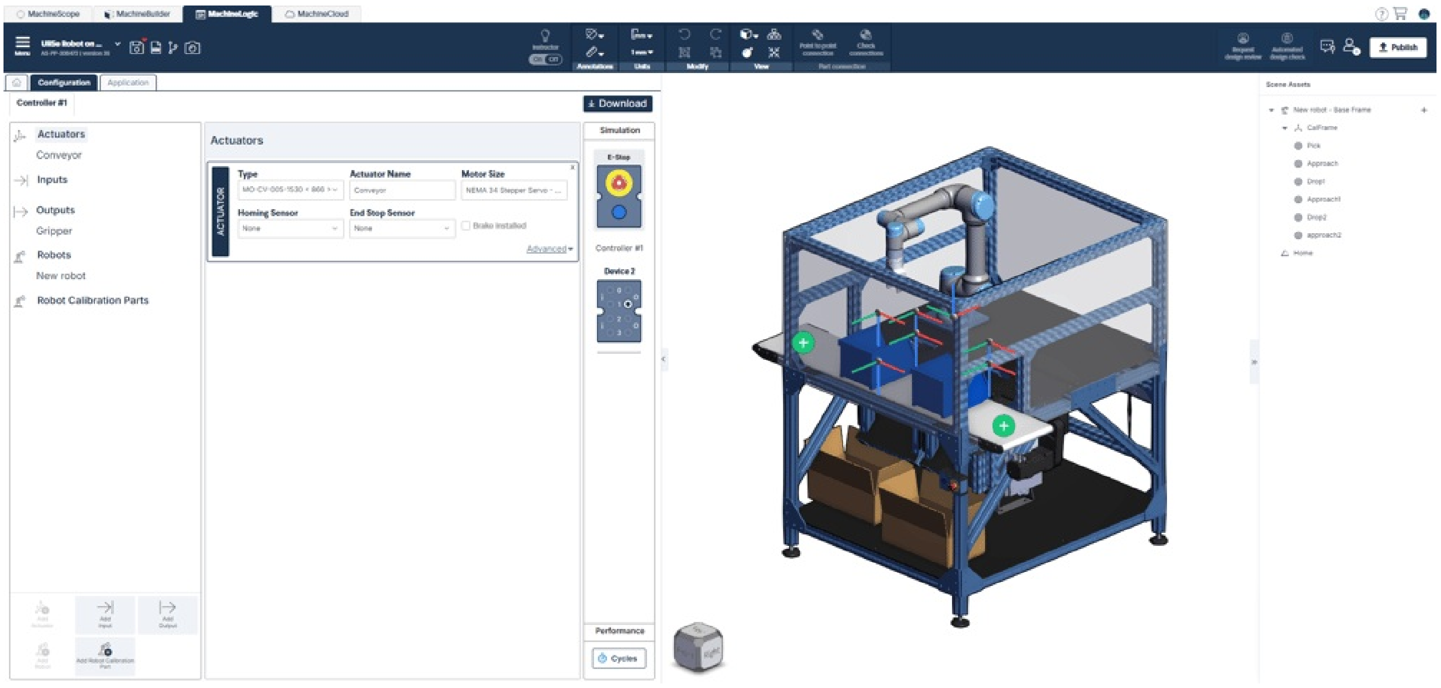
Linear/Rotary motion and Material Handling devices

Click the MachineLogic tab to begin configuring the automated equipment in your design.

1. Click the “Add Actuator” to begin your machine configuration
2. In the “Type” dropdown menu, select the actuator you would like to configure from your design.
3. The following fields will be auto-populated based on the parts connected to the selected actuator from the “Type” drop down menu:

- Actuator name: Friendly name given to the actuator.
- Motor size: Motor size connected to the actuator.
- Homing sensor: We will guess which sensor should be the gantry’s home position. If you would like to reverse the motor direction, switch the homing sensor with the end-stop sensor.
- End-stop sensor: We will guess which sensor should be the gantry’s end-stop sensor. If you would like to reverse the motor direction, switch the homing sensor with the end-stop sensor.
- Brake installed: This checkbox represents the presence of a brake on the associated actuator.
- Gearbox installed: This checkbox represents the presence of a gearbox on the associated actuator.
- Advanced: Allows you to configure the following fields:
  - Custom Current: A different current value for your motor. The default value will be shown.

Tuning profile: This allows you to tune your step-servo motor using various profiles, to achieve the best performance for your application.



Actuators

ACTUATOR

Type	Actuator Name	Motor Size
MO-CV-005-1530 × 866 ›	Conveyor	NEMA 34 Stepper Servo - …

Homing Sensor	End Stop Sensor	
None	None	☐ Brake installed

Advanced

Figure 2: Actuator configuration in MachineLogic

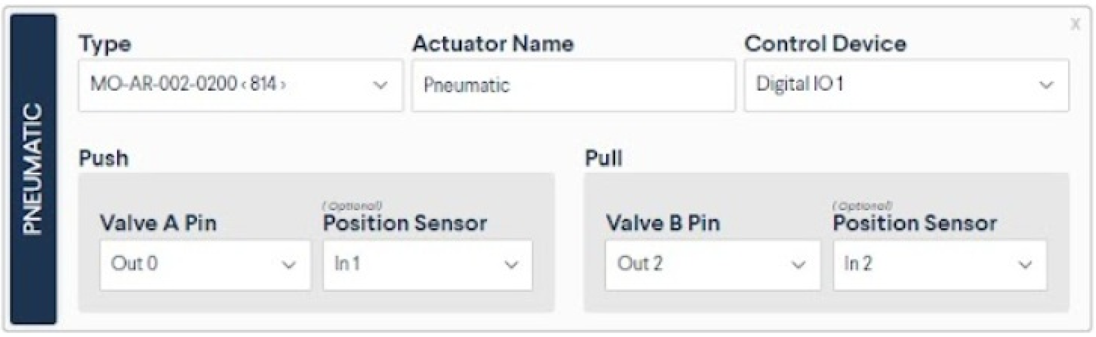
Pneumatic actuators

Click the MachineLogic tab to begin configuring a pneumatic actuator.

1. Click the “Add Actuator” to begin your machine configuration
2. In the “Type” dropdown menu, select the pneumatic actuator you would like to configure from your design.

- Actuator name: Friendly name given to the pneumatic actuator.
- Valve A Push: Represents the output pin from the digital IO module (CE-MD-001-0001) to activate the extended piston position of the pneumatic actuator.
- Position Sensor (push): This field is optional. Represents the pneumatic actuator position sensor (CE-SN-008-0001) to give the feedback if the pneumatic actuator is in the “push” position. Select the input pin that will be connected to this sensor.
- Valve B Pull: Represents the output pin from the digital IO module (CE-MD-001-0001) to activate the retracted piston position of the pneumatic actuator.

Position Sensor (push): This field is optional. Represents the pneumatic actuator position sensor (CE-SN-008-0001) to give the feedback if the pneumatic actuator is in the “pull” position. Select the input pin that will be connected to this sensor.



PNEUMATIC

Type	Actuator Name	Control Device
MO-AR-002-0200 ‹ 814 ›	Pneumatic	Digital IO 1

Push: Valve A Pin	Push: (Optional) Position Sensor	Pull: Valve B Pin	Pull: (Optional) Position Sensor
Out 0	In 1	Out 2	In 2

Figure 3: Configuring a pneumatic actuator in MachineLogic

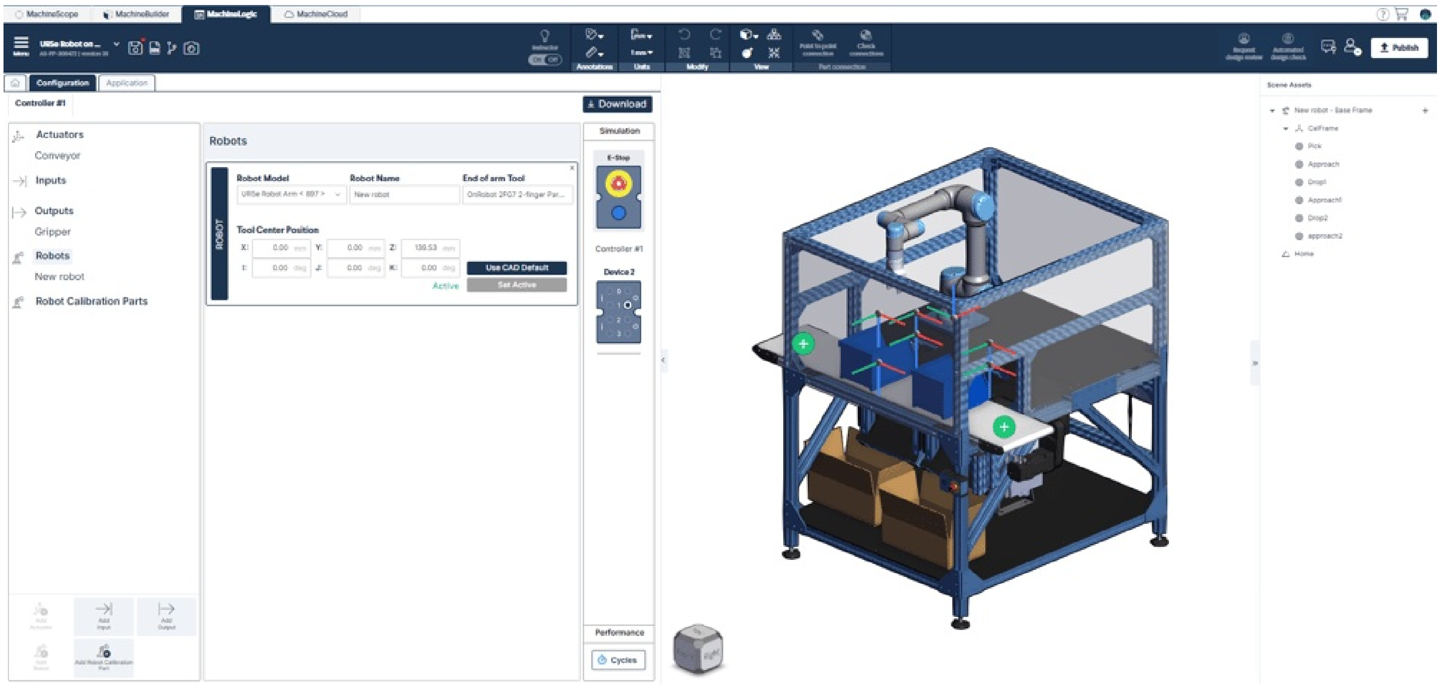
Inputs and outputs

1. Click the “Add Input” or “Add Output” button
2. Provide the following information:

- Input/Output: Specfy IO type

- Name: Friendly name of the associated IO signal
- Device: Device number of the associated module as shown in the simulation pane.
- Pin: Associated Pin used on the corresponding device

Position Sensor (push): This field is optional. Represents the pneumatic actuator position sensor (CE-SN-008-0001) to give the feedback if the pneumatic actuator is in the "pull" position. Select the input pin that will be connected to this sensor.



Robots

Robot Model: UR5e Robot Arm < 897 >
Robot Name: New robot
End of arm Tool: OnRobot 2FG7 2-finger Par...

Tool Center Position
X: 0.00 mm Y: 0.00 mm Z: 139.53 mm
I: 0.00 deg J: 0.00 deg K: 0.00 deg
Active
Use CAD Default
Set Active

Figure 4: Configuring an I/O in MachineLogic

Code-free Programming

With the configuration now defined, create a new code-free application from the MachineLogic Landing page. The following section will focus on explaining the different functionalities of Code-Free Programming Interface:

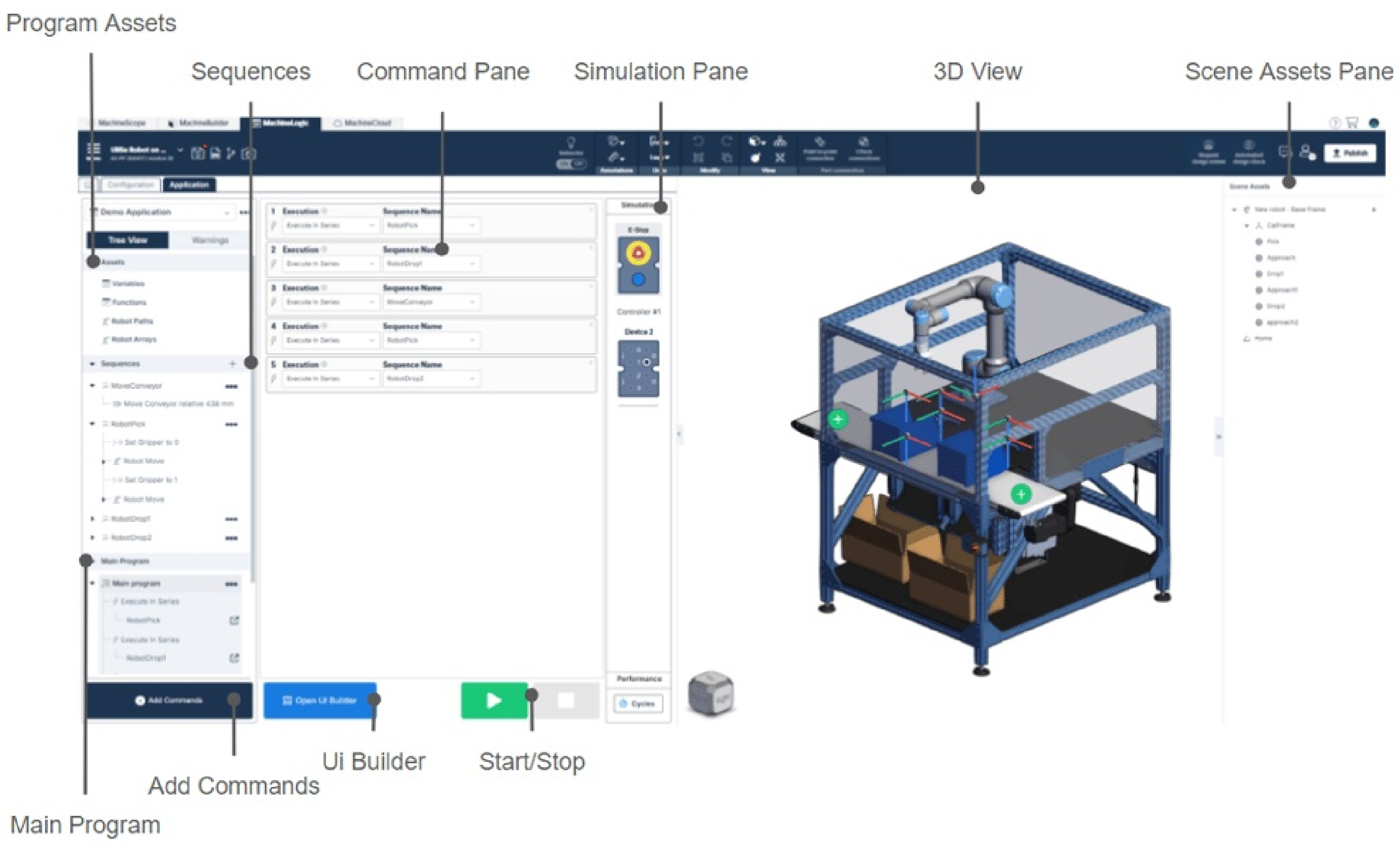


Figure 5: MachineLogic Code-Free graphical user Interface

Program Assets

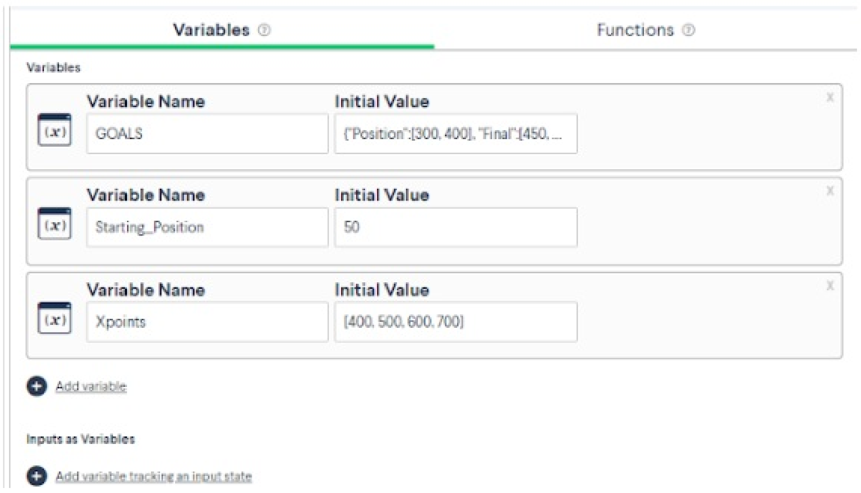
Variables and Functions can defined from the Program Assets section:

Variables

Variables give the ability to change a parameter easily throughout the entirety of the MachineLogic program. This can be used to expose important application parameters such as speeds and positions, and makes it generally easier to collaborate with colleagues on the same application.
Variables support the following data type:

- Integers and floats
- Arrays
- Strings
- Objects
- Json files

Variable values can be inputted directly in MachineLogic commands and changed using the Set Variable command.
It is also possible to define variables tracking input states, this can be useful when setting flags in the application that reflect the state of a configured input.



Variables

Variable Name	Initial Value
GOALS	{"Position":[300, 400], "Final":[450, ...
Starting_Position	50
Xpoints	[400, 500, 600, 700]

Inputs as Variables

Figure 6: Variables in MachineLogic

Functions

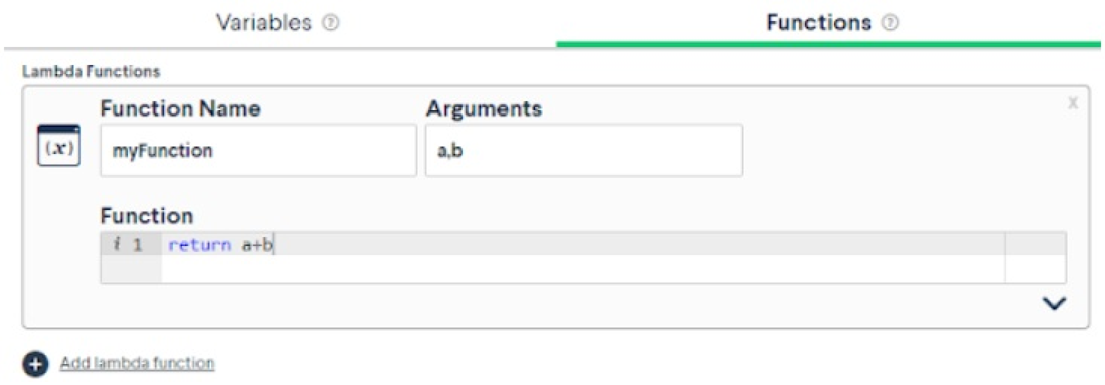
functions give the ability to create complex expressions to return a string, a number, an array or an object, to be inputted in other MachineLogic commands. To use functions, enter a function name, arguments to allow names to be passed in the body and the actual Function body.
Functions accept regular JavaScript syntax, meaning for or while loops, if-else conditionals, operators, and much more can be written in the function's body. Additionally, JavaScript objects such as Math, Array, or Map, and their respective functions can also be called directly in-line. Learn more about what can be done by visiting the official JavaScript Documentation.
functions can be called in any of the MachineLogic sequence commands found below, in the following format: FunctionName(Arguments).

- Motion commands
- Wait commands
- Output commands
- Set variable commands
- Condition commands
- Loop commands
- Message commands

Functions should be used when similar calculations are needed to be used in the commands mentioned above. For example, if a position could be computed by adding two numbers together, it is possible to create a function that takes in two arguments and returns their sum.

Example:
First, create a function in the functions tab:

myFunction(a,b) {return a + b}



Lambda Functions

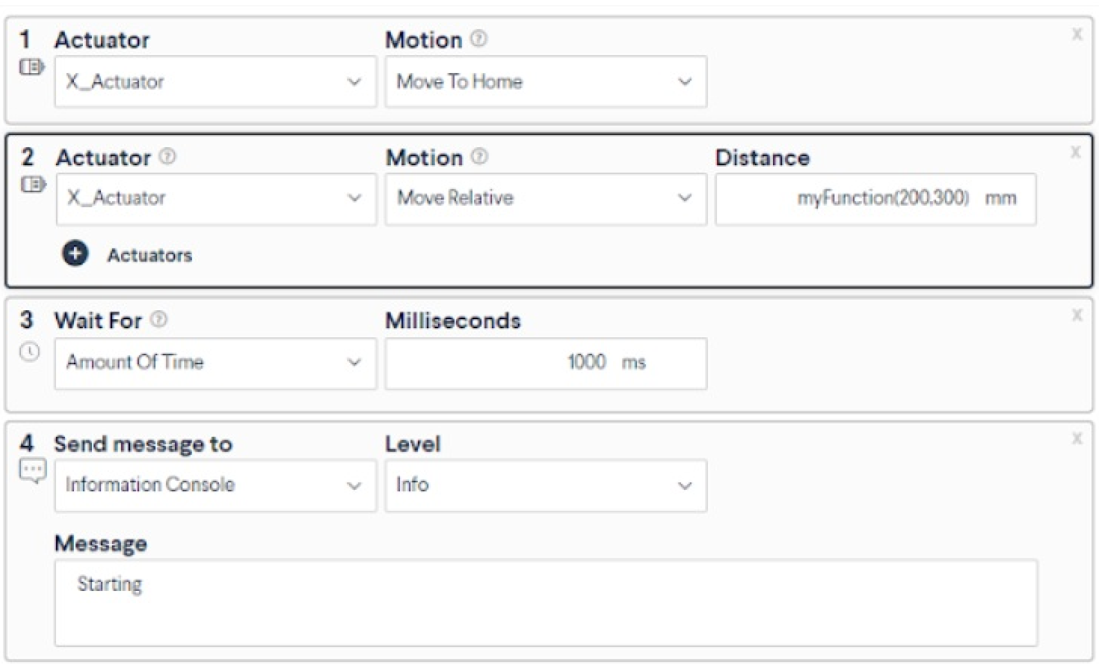
Function Name: myFunction

Arguments: a,b

Function

```
return a+b
```

The example below shows how functions can be inputted in a "add motion" command, in this example, myFunction(200,300) will return a value of 500.


1 Actuator: X_Actuator — Motion: Move To Home

2 Actuator: X_Actuator — Motion: Move Relative — Distance: myFunction(200,300) mm

Actuators

3 Wait For: Amount Of Time — Milliseconds: 1000 ms

4 Send message to: Information Console — Level: Info

Message: Starting

Figure 8: Calling a lambda function in a command

Sequences

Sequences can be thought of as functions that are executed by the Main Program when using either an Execute, Condition or Loop command . Sequences contain a set of commands that will dictate a task accomplished by the machine.

New sequences can be inserted using the “Plus” icon in the header.

Sequences are displayed in a tree-view structure displaying the commands they contain. From the tree-view it is possible to duplicate sequences using the context menu and reorder them using drag and drop.

Main Program

The Main Program is where sequences and commands are executed and simulated in the 3D View. Sequences can be executed in the main program using either Execute, Condition or Loop commands. It is also possible to directly insert commands in the Main Program.

Only one main Program may be created inside an application.

Main Program can also support State Machines. State machines can conveniently associate Events and Actions and can be useful when programming complex machines. Any number of state machines can be added to the Main Program.

For further information on State Machines, click here.

Add Commands

The Add command button is used to access and insert commands in Sequences and Main Programs.
To see the list of available commands, consult the glossary at section X.

Commands are grouped by categories:

- Add Motion: Contains commands such as Move To Position and Move Relative
- Add Robot Move: Contains the Robot move Command, for more information,click here.
- Add Wait: Contains Wait for event and Wait for Amount of time.
  For more information about communication protocols in MachineLogic, click here.
- Add Output: Contains Send Event and Set Digital Output command.
  For more information about communication protocols in MachineLogic, click here.
- Add State Machine: Inserts a state machine into the Main Program.
  For more information aboutState Machines, click here.
- Add Execution: Contains Execute in series and Execute in Parallel commands
- Set Variable: Used to set the value of a variable.
- Add Condition: executes a Sequence only if a specific condition is True.
- Add Loop: Allows to loop a Sequence either Forever, for a finite number of times, or while an expression is True
- Add Message: Allows to send a message either to a Console in the Operator UI or send HTTP requests.
  For more information about communication protocols in MachineLogic, click here.

Command Pane

The Command Pane is where each of the inserted command's properties are displayed. Commands are displayed and executed from top to bottom. It is possible to reorder commands in this view by drag and drop.

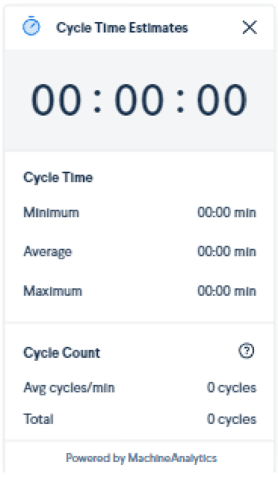
Simulation pane

Device Emulation:

The simulation pane allows for the emulation of IO devices and push buttons added to the design.
When these devices are configured, they will show in the simulation pane and allow interactions with the program and application when the simulation is running.
Emulation is supported for the following devices:
PushButton module
Digital I/O module
Estop Module
Estop Module W/ reset

Cycle Time

The Cycle time functionality displays both cycle time and cycle count when playing a program in simulation.
To calculate cycle time, the application must contain both a MachineAnalytics Cycle Start and Cycle End events. Both commands are found in the Set Output Category.


Cycle Time Estimates

00 : 00 : 00

Cycle Time

Minimum	00:00 min
Average	00:00 min
Maximum	00:00 min

Cycle Count

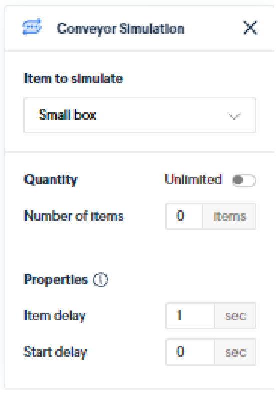
Avg cycles/min	0 cycles
Total	0 cycles

Powered by MachineAnalytics

Figure 9: Cycle Time Display

3D View

This view displays the current loaded design and enables the MachineBuilder context menu.
Conveyor box Mechanics can also be enabled when Conveyors are present in the design.
To enable Conveyor Simulation, select the plus icon on the associated conveyor and specify the desired simulated behavior.


Conveyor Simulation

Item to simulate

Small box

Quantity — Unlimited

Number of items: 0 items

Properties

Item delay: 1 sec

Start delay: 0 sec

Figure 10: Conveyor Simulation

Scene Assets Pane

The scene Assets pane is where robot programming Assets are displayed. To learn more about Scene Assets,click here

UI Builder

Ui Builder is a feature that can be used to create Operator Interfaces that allow interaction with the application.

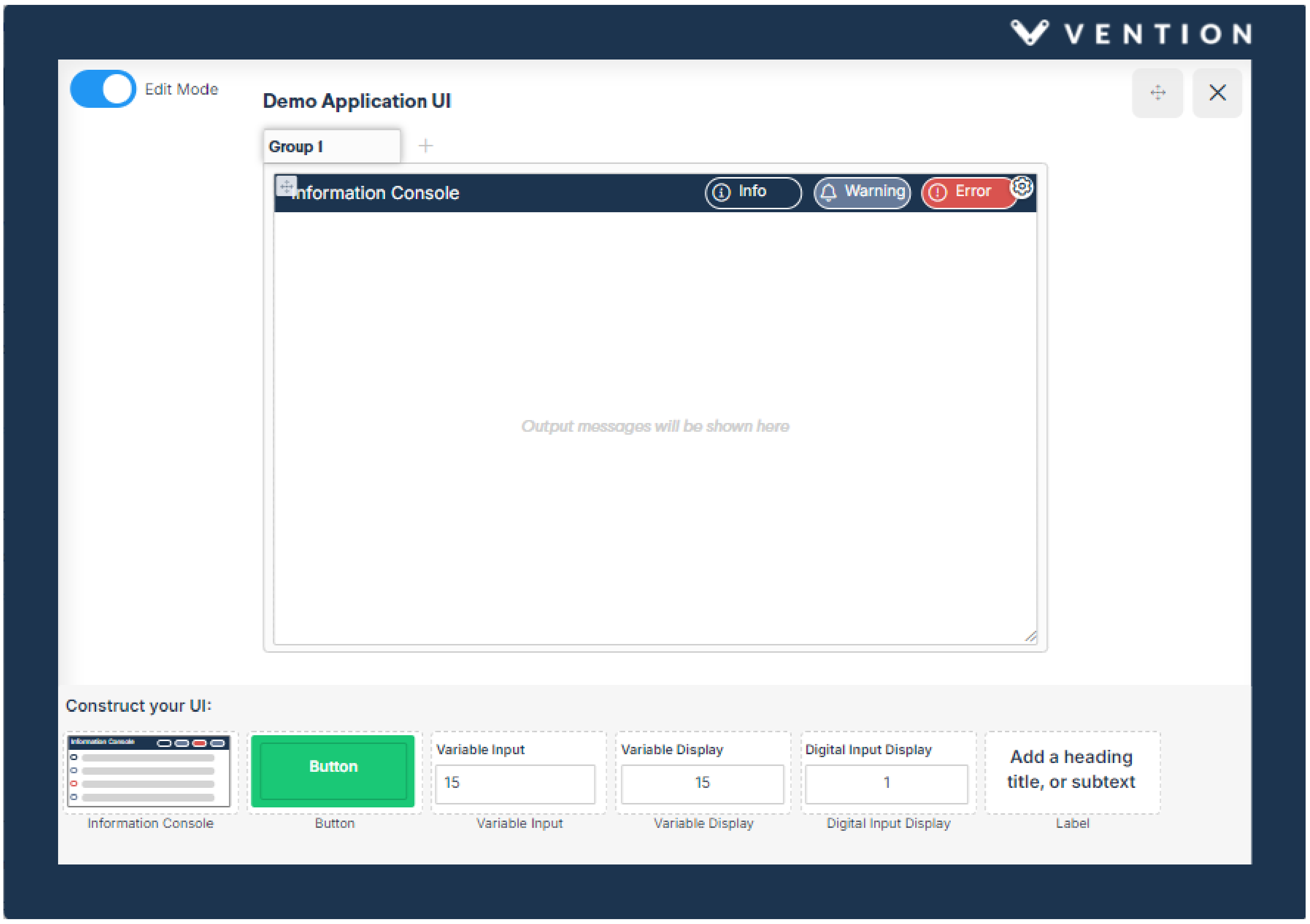


Figure 11: Conveyor Simulation

- Edit Mode: toggles between edit and play mode. The application can be launched in Play mode.
- Group: Allows the creation of additional pages when the plus icon is selected.
- Information Console: Shows message in the console when an Add Message command is read,
- Button: Sends an MQTT topic that can be used within the application to trigger events.
  For more information about Events and MQTT , click here.
- Variable Input: Sets the value of a variable from the Operator Interface
- Variable Display: Displays the current value of a variable.
- Digital Input Display: Displays the current state of a digital Input.
- Label: Add text to the operator interface

Start / Stop

These are the controls for launching and interrupting the simulation of the current application

Motion command glossary

Add Motion	
	Command Icon
Move to home	The "Move to Home" command allows you to position your actuator at its Home position, where the home sensor is located. You need a homing sensor to use this command. This command is recommended at the beginning of each application for each actuator, because its set the "zero" of your position.

Add Motion	
Move Relative	This command allows you to move relative to the current position of the gantry. This will accept both negative or positive distance (mm) values. If the "Move Relative" command exceeds the travel length of the actuator, it will stop at the closest home or end sensor. Example 1. For a timing belt actuator (travel length is 855 mm) at its home position, and you execute a move relative command +1000 mm, the gantry will stop at the end sensor. Example 2. For a rotary actuator if the current position is 90 deg, a move relative command -20 deg would bring the gantry to 70 deg.
Move to position	Move to Position command allows the actuator to move in an absolute position from the home position. Note that a Move to Home command is required before adding the Move to Position command in the application. Move to Position command only allows for a positive value input.
Set System Speed	Configures coasting speed of all actuators in the system. All movement/commands entered after this "Set Speed" command will follow this speed until a new "Set Speed" command is entered.
Set System Acceleration	Configures acceleration of all actuator in the system. All movement/commands entered after this "Set Acceleration" command will follow this acceleration until a new "Set Acceleration" command is entered.
Set angle	Use this function when you would like to redefine an angle on your current rotary actuator position. This command does not move the actuator, but will override its position value.
Move to closest angle	This command allows you to rotate your rotary actuator to the specified position using the desired direction: • **Positive**: Rotates in the positive direction of your motor. Default: Counterclockwise • **Negative**: Rotates in the positive direction of your motor. Default: Clockwise • **Shortest path**: Rotates either clockwise or counterclockwise using the less rotational movement possible Your rotary actuator will always move less than a full turn. The specified angle needs to be between -360 and 360 deg.
Start continuous move	For conveyors and rotary actuators, you have the ability to actuate them continuously. It may be considered a start command to actuate the conveyor/rotary actuator continuously. Typically, until an event happens or under a timeframe.
Stop continuous move	To stop a continuous motion, you should use the "Stop Continuous Move" command. Please note that the "Stop All Motion" command would also stop any continuous moves.
Stop all motion	To stop all motion from the previous commands. This command is useful when you have various "Set continuous move" commands and you would like all movement to stop using one command.
Push	Extends the piston of the selected pneumatic actuator.
Pull	Retracts the piston of the selected pneumatic actuator.
Idle	Disables the output of the pneumatic actuator, allowing the movement of the pneumatic actuator to be "free" (could easily by extended or retracted with any external force).

Add Wait	
🕒	Command Icon
Amount of time	Provides the ability to wait for a certain time delay before performing the next command in a sequence.

Add Wait	
Event	Wait for event command creates the ability to wait for an event topic and/or message to be generated before executing subsequent commands in the sequence. For an example, this gives the ability for an operator to generate an event by clicking a button in the UI builder to execute a sequence that is waiting on an event topic and/or message. Additionally, it provides the ability to wait for a given message on a given MQTT topic before performing the next command in a sequence.
Motion completion	Provides the ability to wait until all moves complete before performing the next command in a sequence. This command will be useful for conditional statements, where you would only execute another command if a certain motion has completed.
Digital input	Provides the ability to wait for a given digital input of an digital IO module on specified state (0 or 1) before performing the next command in a sequence. For an example, depending on the state of a sensor (0: no object sensed, 1: object sensed), you could add a command that will only move an actuator of the state of the sensor is "1".
Digital input edge	Provides the ability to wait for a given digital input of an digital IO module to transition : from 0 to 1 for a rising edge, from 1 to 0 for a falling edge before performing the next command in a sequence. For an example, if you would like to command a conveyor to move only if a sensor detected a box moving past a sensor, triggering the state to go from 0 to 1 and 1 to 0.

Add output	
↦	Command Icon
Digital output	Add a command to activate your digital I/O compatible components using pins (0, 1, 2, 3). Each pin will trigger an action from that connected component (i.e pneumatic actuator, status light, etc.). Example. If you have set up your configuration with a pneumatic actuator, you could activate your actuator by inputing the output as "Pneumatic actuator" from the drop-down menu. Afterwards, enter your pin associated with that action (0 or 1).
Generate event	Generate Event command gives the ability to generate an event, with a given topic and an optional message. For example, it allows to resolve any active "WaitForEvent" commands that would be waiting on the same topics and messages

Add Execution	
⚡	Command Icon
Execute in parallel	This allows you to play/run a command in parallel with another command. The command you input with the "Execute in parallel" will run in parallel with the next command. Tip: Under your main sequence, you should add most of your execution commands there.
Execute in series	This allows you to play/run a sequence within another sequence, and wait for its completion before moving on to the next instruction. This is useful for repetitive sequential movements. Tip: Under your main sequence, you should add most of your execution commands there.
Terminate	Terminates the execution of the program. If you know your machine is experiencing an error, use this command. This will allow your machine to be in the state of a "software stop".

Set Variable	
(x)	Command Icon

Set Variable	
From device	Set variable command category allows the “Initial Value” of a variable (set from the “Variables tab”) to change. All the commands executed after this command will use that new value. “Set Variable from Device” allows a variable to take the value of one input pin of the digital IO module.
From expression	Set Variable command category allows the “Initial Value” of a variable (set from the “Variables” tab) to change. All the commands executed after this command will use that new value. “Set Variable from Expression” allows a variable (set from the “Variables” tab) to take the value specified in the expression field. The expression field supports numbers, mathematical operators (*, /, +, -) and variable names.
From expression array	**Set Variables from Expression** allows variables (set from the **Variables** tab) to take an array of values specified in the **Expression** field. The **Expression** field supports numbers, mathematical operators (*, /, +, -) and **Variable Names**.

Add Message	
💬	Command Icon
Information Console	Adding a message to the information console allows you to output a message to the UI builder for the operator to view. Variables, functions and text could be entered as a message. To display the message to the operator, ensure you drag and drop the information console widget in the UI Builder.
URL	Adding a message to a URL allows you to send a message (“Pack message”) to an external server and store the resulting output of the server in the “Unpack Variable Name”.