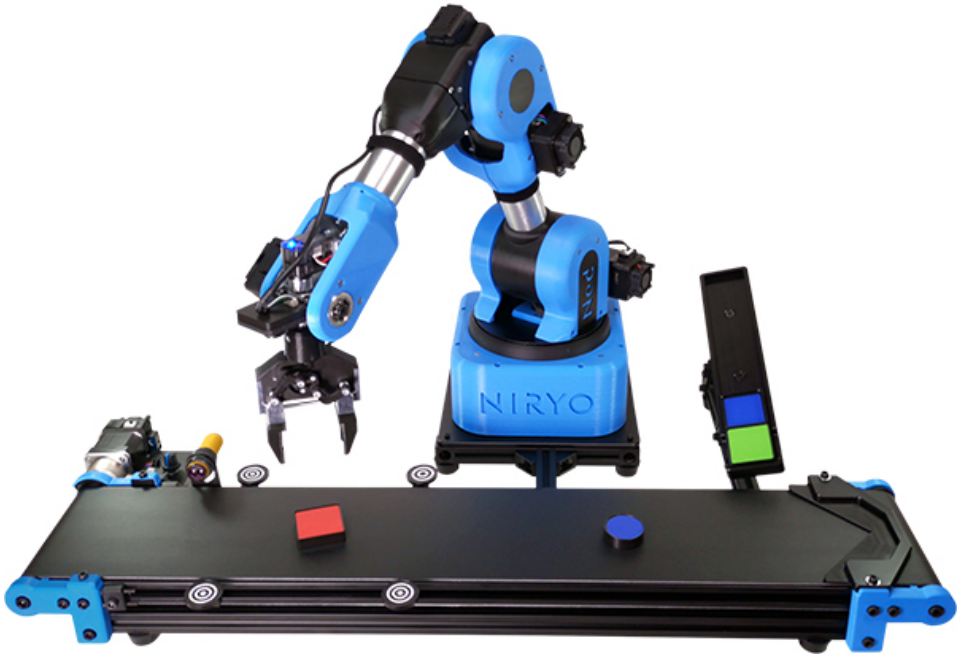
Niryo Conveyor Belt - User Manual



This manual provides a general overview of Niryo's Conveyor Belt and describes in detail how to control and use the product.

The Conveyor Belt can be used to move small objects in two directions with an adjustable speed to simulate a production line automation. It can be controlled using different programming methods and can be autonomous by using the Control Box provided.

Safety Precautions

- Please be sure to respect the electrical connection of the Conveyor Belt, as detailed in this manual,
- Please be sure to do not exceed the maximum payload of the Conveyor Belt as detailed in the product specifications.

Prerequisites

If you are using the Conveyor Belt with Ned, there are three software layers, from higher to lower:

- Software Niryo One Version ≥ 2.3 Or Ned version (this version is installed on every Niryo One bought from the 30/06/2020.)
- Niryo Studio Version ≥ 2.3
- Conveyor Belt firmware version ≥ 1.0

If you do not have the recommended versions, please visit our download page (https://niryo.com/download/) and refer to those tutorials:

- Update Niryo Studio (https://niryo.com/docs/niryo-one/update-yourrobot/update-niryo-one-studio/)
- Update the Raspberry Pi image (https://niryo.com/docs/niryo-one/updateyour-robot/update-raspberry-pi-image/)

Kit Overview

Part	Quantity	Description
Conveyor Belt	x1	Niryo Conveyor Belt
Niryo - Conveyor Belt cable	x1	Allows you to connect the Conveyor Belt to Ned
IR Obstacle sensor	x1	Detects the presence of any obstacle in front of itself
Niryo - Sensor cable	x1	Allows you to connect the IR sensor to Ned
Control Box	x1	Makes the Conveyor Belt autonomous: controls the speed and the direction of the Conveyor Belt. Also has a digital input pin to connect the IR sensor
Control box connector	x1	Allows you to connect the Control Box to the Conveyor Belt.
Power supply adapter	x1	Power supply for the Control Box.
Squared Container *	x3	Handable 3D squares that can be used as containers for the circles.
Circles *	x3	Handable 3D circles that can be placed in the squared containers.
Upgraded jaws for the Standard Gripper *	x2	Upgrades the Standard Gripper to let you grip the objects provided.
Vision Workspace landmarks *	x4	Define a workspace for the Vision Set directly on the Conveyor Belt.
End stop *	x1	Stops objects at the end of the Conveyor Belt.
Slope *	x1	Places a series of objects in a row and let them go to a defined position each time an object is picked at its end.

Note

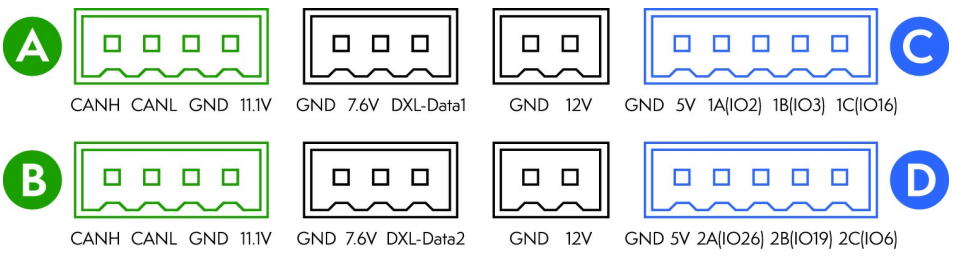
* : in the Education Set only

With Ned

On the GPIO panels, the ports **A** and **B** can be used to connect **the Conveyor Belt**, and the ports **C** and **D** can be used to connect the **IR sensor's adaptator**.

Important

Be careful to connect the IR sensor's adapter in the correct way, respecting the wire's color code



Ned's GPIO panels

Ned - The Conveyor Belt

The Conveyor Belt communicates with Ned via Can Bus. To use the Conveyor Belt with Ned, connect the “Niryo - Conveyor Belt cable” on the side of the Conveyor Belt’s motor and on the back of Ned (port A or B).

Ned - The IR Sensor

This infrared device can detect on object from a given distance. This distance can be adjusted between 6 and 80 cm. To get more information about the sensor and to set up the distance detection, please refer to this document (../Conveyor Belt - Manual_IR-Sensor Switch E18.pdf).

- **When the distance of the sensor < adjusted distance: an obstacle/object is detected.**
    - The LED lights up,
    - The output is set to 0,
- **When the distance of the sensor > adjusted distance: no obstacle/object is detected.**
    - The LED lights off,
    - The output is set to 1.

To mount it with the Conveyor Belt, please follow the steps illustrated below:

Step 1. Screw the first ring

Step 2. Add the sensor support

Step 3. Screw the second ring

Step 4. Put a blocker in the rail where you want the sensor to be

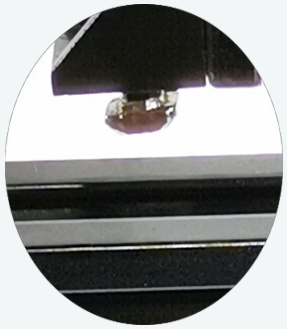
Step 5. Adjust the screw to secure the position

To use the IR Sensor with the Conveyor Belt and Ned, please connect it to the adapter for Ned and connect it on the back of the Ned (port C or D). Be sure to respect the keying of each connector to avoid any short circuit between GND, 5V and other GPIO pins.

The Mechanical Connector

We designed for Ned’s ecosystem a Mechanical Connector that is composed of an aluminum base for Ned and two links to connect a Workspace and a Conveyor Belt. This Mechanical Connector is included in the Vision Set. Assembling this mechanical structure allows the workspace and/or the Conveyor Belt to always

keep the same relative position to Ned. If you own a Mechanical Connector, please follow these steps to mount the Conveyor Belt on it:



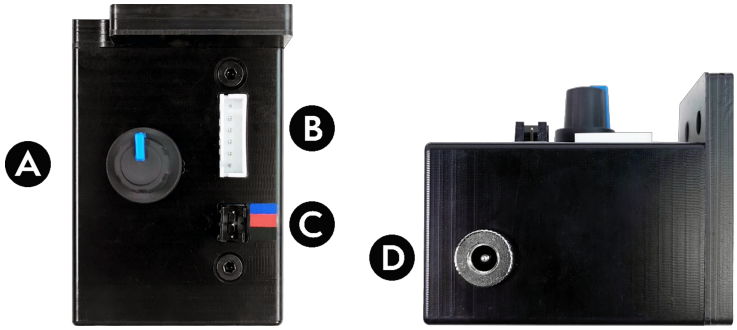
Step 1. Align the aluminum rail and the stopper

Step 2. Push the link into the rail

Step 3. Screw until the stopper turns a quarter turn and blocks into the rail

With the Control Box for an autonomous use

We designed the Conveyor Belt to be used with or without Ned. This section details how to use the Control Box to control the Conveyor Belt. If you use Ned to control it, please refer to the previous section.



A. Rotary Potentiometer which controls the conveyor speed and direction
B. Conveyor connector interface
C. Digital input interface (Blue: GND + Red: 5V + Black: input pin)
D. Power supply

Control Box - The Conveyor Belt

First connect the Conveyor Belt motor to the Conveyor connection interface on the control box**(B)**. Then, plug the power supply adapter to the Control Box **(D)**.

Warning

Please be sure to respect this order. Connecting the Control Box to the power supply before connecting it to the Conveyor Belt could damage the products.

Control Box - The IR Sensor

This infrared device can detect a distance with an obstacle. This distance can be adjusted between 6 and 80 cm. To get more information about the sensor and to set up the distance detection, please refer to this document (../Conveyor Belt - Manual_IR-Sensor Switch E18.pdf).

- **When the distance of the sensor < adjusted distance: an obstacle/object is detected.**
  - The LED lights up,
  - The output is set to 0,
- **When the distance of the sensor > adjusted distance: no obstacle/object is detected.**
  - The LED lights off,
  - The output is set to 1.

Simply connect the IR sensor to the digital input interface of the Control Box **(C)** by following the colors shown aside.

Note

Before following the following instructions, make sure **the Conveyor Belt and the IR sensor are connected as detailed** in the corresponding section (see With Ned (index.html#document-source/setup)).

With Niryo Studio

You can have all the details about the application in Niryo Studio's documentation. (https://docs.niryo.com/product/ned/source/software/niryo_studio.html)

Niryo Studio - The Conveyor Belt

In Niryo Studio, open the Conveyor Belt tab in the left menu. The interface will let you control up to two Conveyors directly. You can enable them and control their speed and direction.

Hint

Click on **Scan** to detect and add the connected Conveyor Belt to Niryo Studio.

To use a single Conveyor Belt, connect the product to Ned and toggle the button of the "Conveyor Belt 1" in Niryo Studio. You can now adjust the speed by using the slider or entering the percentage value and choosing if the Conveyor Belt has to move forward or backward.

To use two Conveyor Belts, follow the steps detailed in the previous paragraph, then, on the "Set Conveyor Belt ID" section of Niryo Studio, select "Conveyor Belt 2" and click "Update". Now, connect your second Conveyor Belt to Ned, which will be considered as "Conveyor Belt 1". Please note that the ID are saved until you disconnect the Conveyors. These steps are required each time you reconnect two Conveyors.

Niryo Studio - IR sensor

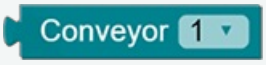
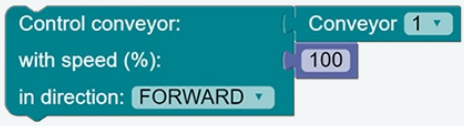
In Niryo Studio, in the "Digital I/O panel", make the following change:

- **If you connected the Sensor on the A** connection in the previous section, change the mode of "1A" to "INPUT".
- **If you connected the Sensor on the B** connection in the previous section, change the mode of "2A" to "INPUT".

When the distance measured is **smaller** than the adjusted distance, the "INPUT" is "LOW", it means an obstacle is detected. When the distance measured is **higher** than the adjusted distance, the "INPUT" is "HIGH", it means that there is no obstacle.

Blockly

Special blocks have been designed to be used with the Conveyor Belt. These blocks keep Ned's Ecosystem easy to use with Niryo Studio.

Images	Description
	Allows the user to choose the Conveyor ID
	Allows the Conveyor Belt to be controlled via Ned
	This block will control the Conveyor Belt: • With the ID (1 or 2), • With the speed (from 0 to 100%), • With the direction (FORWARD and BACKWARD)
	This block stops the Conveyor Belt

With the Control Box

The rotary potentiometer allows you to control the speed and the direction of the Conveyor Belt.
When the potentiometer is at its origin, centered, the Conveyor Belt is stopped.
The more you turn the potentiometer clockwise, the fastest the Conveyor Belt will move forward.
Conversely, the more you turn the potentiometer counterclockwise, the fastest the Conveyor Belt will move backward.

With PyNiryo

Here are provided two scripts for the Conveyor Belt and the IR sensor. You can find more details in PyNiryo's Documentation (https://docs.niryo.com/dev/pyniryo/source/api_doc/api.html#conveyor).

PyNiryo - The Conveyor Belt

Here is a code example showing how to scan the connected Conveyor Belt and launch its motor:

```
#!/usr/bin/env python

from pyniryo import *

# Connecting to robot
robot = NiryoRobot(<robot_ip_address>)

# Activating connexion with Conveyor Belt
conveyor_id = robot.set_conveyor()

# Running the Conveyor Belt at 50% of its maximum speed, in forward direction
robot.run_conveyor(conveyor_id, speed=50, direction=ConveyorDirection.FORWARD)

# Waiting 3 seconds
robot.wait(3)

# Stopping conveyor's motor
robot.stop_conveyor(conveyor_id)

# Deactivating connexion with the Conveyor Belt
robot.unset_conveyor(conveyor_id)
```

PyNiryo - IR sensor

You can get the IR sensor state and control the Conveyor Belt accordingly as follows. Please note that, in this example, the sensor is connected to the GPIO_1A.

```python
#!/usr/bin/env python

from pyniryo import *

# Connecting to the robot
robot = NiryoRobot(<robot_ip_address>)

# Activating connexion with the Conveyor Belt
conveyor_id = robot.set_conveyor()

# -- Setting variables
sensor_pin_id = PinID.GPIO_1A

# Run conveyor and wait until the IR sensor detects an object
robot.run_conveyor(conveyor_id)
while robot.digital_read(sensor_pin_id) == PinState.LOW:
    robot.wait(0.1)

# Stopping conveyor's motor
robot.stop_conveyor(conveyor_id)

# Deactivating connexion with the Conveyor Belt
robot.unset_conveyor(conveyor_id)
```

With Modbus

More details about how to use Modbus with Ned and the Conveyor Belt can be found in our Modbus documentation (https://docs.niryo.com/dev/modbus/index.html).

Modbus - The Conveyor Belt

Here is a python code example that shows how to control the Conveyor Belt through a Modbus TCP Client:

```python
#!/usr/bin/env python

from pymodbus.client.sync import ModbusTcpClient
import time

# Connect to the robot
client = ModbusTcpClient('<robot_ip_address>', port=5020)
client.connect()

# Enable Conveyor 1
client.write_register(520, 1)
time.sleep(1)

# Set direction to forward
client.write_register(523, 1)
time.sleep(1)

# Set speed to 50%
client.write_register(524, 50)
time.sleep(1)

# Start conveyor 1
client.write_register(522, 1)

time.sleep(10)

# Stop conveyor 1
client.write_register(526, 1)

# Close connection to modbus server
client.close()
```

Modbus - IR sensor

Here is how you can set the digital IO connected to the IR sensor to Input mode and get its state:

```
#!/usr/bin/env python

from pymodbus.client.sync import ModbusTcpClient

# Connect to the robot
client = ModbusTcpClient('<robot_ip_address>', port=5020)
client.connect()

# Set digital IO mode - input on GPIO_1A
client.write_coil(0, True)

# Read digital IO state. 1 = High, 0 = Low
ir_state = client.read_discrete_inputs(100, count = 1)
print 'IR sensor state is :', ir_state.bits

# Close connection to modbus server
client.close()
```

With Python ROS Wrapper

More details about how to use the Python ROS Wrapper with Ned and the Convevor Belt can be found our ROS wrapper documentation. (https://www.docs.niryo.com/dev/ros/source/ros_wrapper.html)

Important

If you are **using Ned in real** and you don't want to write the following code directly on the robot, you will need to setup a **multi-machines** environment, following this tutorial. (https://docs.niryo.com/applications/ned/source/tutorials/control_ned_ros_multi_machines.html) On the other hand, you can directly use PyNiryo following the With PyNiryo section.

Python ROS Wrapper - The Conveyor Belt

Here is a code example on how to control the Conveyor through a the Python ROS Wrapper:

```
#!/usr/bin/env python

# Imports
from niryo_robot_python_ros_wrapper import *
import rospy

niryo_robot = NiryoRosWrapper()

# Activating connexion with conveyor and storing ID
conveyor_id = niryo_robot.set_conveyor()

# Running conveyor at 50% of its maximum speed, in Forward direction
niryo_robot.control_conveyor(conveyor_id, True, 100, ConveyorDirection.FORWARD)

# Stopping conveyor's motor
niryo_robot.control_conveyor(conveyor_id, True, 0, ConveyorDirection.FORWARD)

# Deactivating connexion with conveyor
niryo_robot.unset_conveyor(conveyor_id)
```

Python ROS Wrapper - IR sensor

You can get the IR sensor state and control the Conveyor Belt accordingly as follows. Here, the Conveyor Belt is connected to the GPIO_1A.

```python
#!/usr/bin/env python

# Imports
from niryo_robot_python_ros_wrapper import *
import rospy

def run_conveyor(robot, conveyor):
robot.control_conveyor(conveyor, bool_control_on=True,
                       speed=50, direction=ConveyorDirection.FORWARD)

# -- Setting variables
sensor_pin_id = PinID.GPIO_1A

niryo_robot = NiryoRosWrapper()

# Activating connexion with conveyor
conveyor_id = niryo_robot.set_conveyor()

# Run conveyor and wait until the IR sensor detects an object
run_conveyor(niryo_robot, conveyor_id)
while niryo_robot.digital_read(sensor_pin_id) == PinState.LOW:
    niryo_robot.wait(0.1)

# Stopping conveyor's motor
niryo_robot.control_conveyor(conveyor_id, True, 0, ConveyorDirection.FORWARD)

# Deactivating connexion with conveyor
niryo_robot.unset_conveyor(conveyor_id)
```

Suggest a modification | Download as PDF