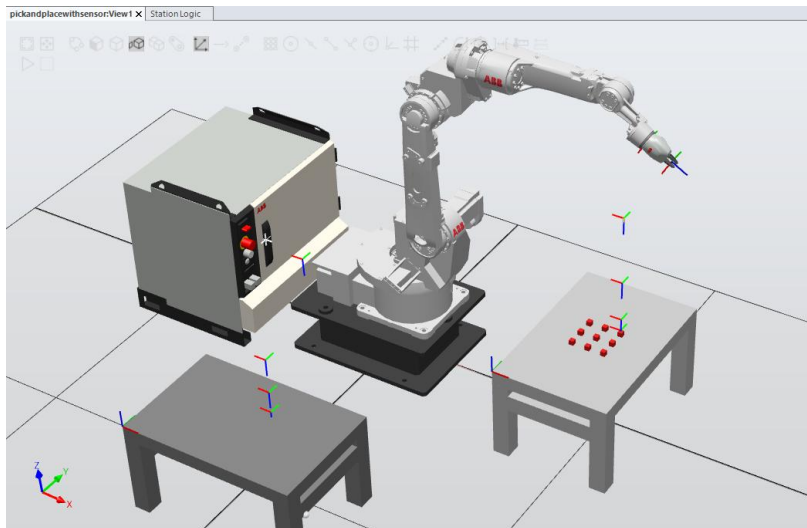


Pick and Place with Gripper

Overview: Utilize the advanced rapid program to efficiently maneuver the gripper and successfully pick up a total of nine square boxes. This will ensure precision and speed in your task, enhancing productivity and accuracy. Make sure to monitor the gripper's alignment and positioning closely as it interacts with each box to avoid any mishaps or misplacements. By following this method, you will achieve optimal results in your box-handling operation.

1. SetUp:



Robot : IRB1520ID_4_150_02

Robotic arm span : 150 cm

Lift: 4kg

Gripper : Smart_Gripper_Servo_Fingers

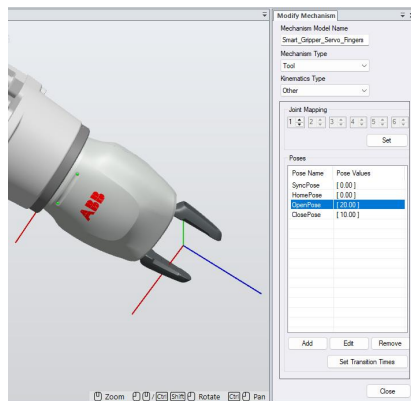
Object to pick up : Box_3D(L=H=W= 20mm),quantity = 9

H = Height,W = Width,L = Length

Object of support : 2 Tables_3D,1 Robotpedestal1400_H240.

IRC5 : ICR5 Control Module

2. Develop Two Distinct Poses for the Tool



1. OpenPose – For Opening or Detaching Objects

Design a pose, called OpenPose, that configures the tool to perform actions such as opening its mechanism or detaching an object. In this configuration, the tool's components should move to positions that allow for:

Releasing Objects: The mechanism is set to a wide or neutral state, enabling the easy release of any held object.

Detachment: The configuration is such that it facilitates the smooth and controlled separation of an object from its attached position without causing any undue stress or damage.

2. ClosePose – For Picking Up or Attaching Objects

Similarly, create another pose, referred to as ClosePose, where the tool is optimized for grasping or securing objects. In this pose, the tool should:

Grasp Objects: Adjust its parts (such as fingers, clamps, or arms) to come together securely, ensuring a firm grip on the object.

Attach Objects: Be configured to hold or attach an object firmly in place, which is especially important during operations where stability and precision are critical

Implementation Considerations:

Smooth Transitioning: Ensure that the movement between OpenPose and ClosePose is smooth.

Define clear angular limits and movement sequences to allow for seamless transitions that do not introduce mechanical strain or instability.

Calibration and Testing: Fine-tuning the poses through iterative testing is vital. Calibration will help in achieving the optimal configuration for each pose, ensuring that: OpenPose reliably facilitates object release or detachment. ClosePose securely handles objects during pickup or attachment operations.

Safety and Efficiency: Properly defining these poses not only improves the overall efficiency of the tool but also enhances safety. Well-calibrated poses minimize the risk of damage to both the tool and the objects being manipulated by preventing misalignment or overexertion of the tool's mechanical components.

3. Station Logic:

1.Creat Signal

Attach	Digital Output			N/A		Default	0	N/A	N/A	No	N/A	N/A
AUTO1	Digital Input	PANEL	Automatic Mode(X9.6)	5	safety	ReadOnly	0	0	0	No	N/A	N/A
AUTO2	Digital Input	PANEL	Automatic Mode backup(X9.2)	6	safety	ReadOnly	0	0	0	No	N/A	N/A
CH1	Digital Input	PANEL	Run Chain 1	22	safety	ReadOnly	0	0	0	No	N/A	N/A
CH2	Digital Input	PANEL	Run Chain 2	23	safety	ReadOnly	0	0	0	No	N/A	N/A
Detach	Digital Output			N/A		Default	0	N/A	N/A	No	N/A	N/A

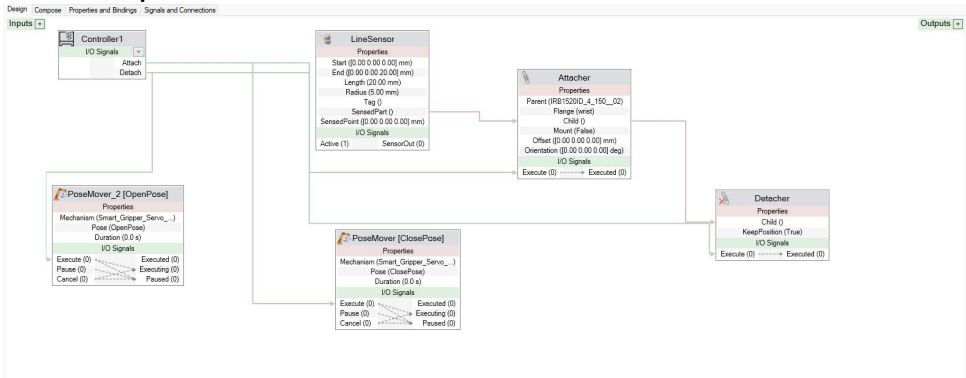
In a robotic system, signals are used to command actions and monitor the state of various components. The two new signals you are introducing—Attach and Detach—are designed to manage object manipulation tasks. Specifically, they allow the robot to:

Attach: Secure an object (e.g., by closing a gripper or engaging a clamp).

Detach: Release an object (e.g., by opening a gripper or disengaging a clamp).

By creating the Attach and Detach signals, you add an essential layer of control and monitoring to the robot’s operations. These signals not only facilitate precise object manipulation but also provide critical feedback that enhances the overall safety and efficiency of the system. Whether in industrial automation, service robotics, or other applications, these signals help ensure that the robot interacts with its environment in a controlled and predictable manner

2.Creat Components



In the simulation environment, a linear sensor plays a critical role by accurately detecting the gripper's contact position with an object. This precise measurement is essential for:

Checking Object Position: The sensor confirms that the object is correctly aligned with the gripper before any manipulation begins.

Correcting Alignment: If discrepancies in position are detected, the system can adjust the gripper’s position to achieve the high precision needed during grasping and handling.

Enhanced Precision: Particularly when the task demands high accuracy, such as handling delicate workpieces or components, the linear sensor ensures that the robot's actions are both reliable and repeatable.

Signal Connections:

1.PoseMover[ClosePose] Connection:

Function: This pose is associated with securing or grasping an object.

Connections: It is connected to the Attach signal, which in turn controls the mechanism of the gripper to move into the ClosePose configuration. This ensures that when an attach command is issued, the gripper moves to a position optimized for grasping.

2.PoseMover[OpenPose] Connection:

Function: This pose is related to releasing or detaching an object.

Connections: It is connected to the Detach signal, commanding the gripper to move into the OpenPose configuration. This prepares the gripper for a safe release of the object.

3. Attacher and Detacher Components

To further streamline the robot's operation, two additional components—Attacher and Detacher—are integrated into the simulation:

Attacher:

Purpose: This component is responsible for attaching objects or devices to the robot.

Usage Example:

When a robot is working with a workpiece, the Attacher ensures that the workpiece is securely attached to the robot. This allows the robot to manipulate or process the workpiece without the risk of it becoming misaligned or dropped.

Operational Role: By linking the Attacher to the Attach signal (via PoseMover[ClosePose]), the robot can confidently initiate a sequence to secure the object using its gripper.

Detacher:

Purpose: The Detacher component facilitates the detachment of objects or devices from the robot.

Usage Example:

Once the robot has completed its task with the workpiece, the Detacher activates to release the object, allowing the robot to proceed to its next task.

Operational Role: Connected to the Detach signal (via PoseMover[OpenPose]), this component ensures that the release process is smooth and that the object is safely disengaged from the robot.

4. Integration for Optimized Robotic Operation

The combined use of the linear sensor, PoseMover, Attacher, and Detacher creates a robust framework for precise object manipulation. Here's how the integration optimizes operations:

Precision Handling:

The linear sensor ensures that any object is in the correct position before the gripper attempts to grasp or release it. This minimizes errors during critical operations that require high precision.

Seamless Transitions:

By mapping specific poses (ClosePose and OpenPose) to the corresponding attach and detach signals, the system guarantees that the robot's gripper transitions smoothly between different operational states.

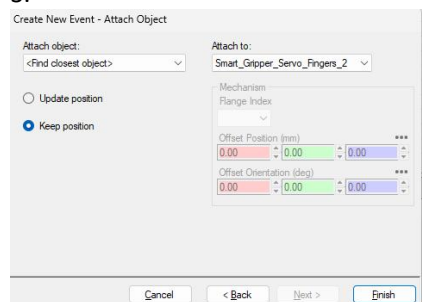
Operational Efficiency:

The smart component architecture of PoseMover simplifies the logical connection between digital signals and mechanical actions. This modular design supports rapid adjustments and calibration, which is essential in a dynamic production environment.

Enhanced Safety:

The precise control afforded by the combination of sensors and smart signal mapping reduces the risk of mishandling objects. It ensures that the robot performs each task accurately and safely, minimizing the chance of damage to both the robot and the workpiece.

3.



The screenshot shows a 'Create New Event - Attach Object' dialog box. It has two main sections: 'Attach object:' with a dropdown menu set to '<Find closest object>', and 'Attach to:' with a dropdown menu set to 'Smart_Gripper_Servo_Fingers_2'. Below these are two radio buttons: 'Update position' (unselected) and 'Keep position' (selected). To the right of the radio buttons is a 'Mechanism Range Index' section with two rows of input fields. The first row is 'Offset Position (mm)' with three fields set to '0.00'. The second row is 'Offset Orientation (deg)' with three fields set to '0.00'. At the bottom of the dialog are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

Events	Activation	Trigger Type	Trigger System	Trigger Name	Trigger Parameter	Action Type	Action System	Action Name	Action Parameter	Time (s)
Add...	On	I/O	Controller1	Attach	1	Attach Object		Attach Object	Smart_Gripper_Servo_Fingers...	
Delete	On	I/O	Controller1	Detach	0	Detach Obj...		Detach Object	Smart_Gripper_Servo_Fingers...	9:56:31 AM
Copy										

In summary, the simulation environment leverages a combination of advanced sensing and smart control components to manage the robot's interactions with its environment effectively. The linear sensor provides the necessary precision for verifying object alignment, while PoseMover facilitates a

direct and logical connection between digital commands and physical actions. With the addition of the Attacher and Detacher components, the system ensures that objects are both securely grasped and reliably released, thereby optimizing the overall production process and enhancing the robot's operational efficiency.

4. Rapid Programming:

```
CONST num spacing := 70;
CONST num Safety_zone := 120;
VAR num Xpos := 0;
VAR num Ypos := 0;
VAR num Zpos := 0;
```

Variable declaration:

- **spacing** : Sets the distance between boxes.
- **Safety_zone** : Provides a buffer for safe picking and dropping operations.
- **Xpos** : Contains the X-axis positions for each of the N boxes.
- **Ypos** : Contains the Y-axis positions for each of the N boxes.

These variables work together to ensure that the robot can accurately and safely navigate the workspace, particularly when precise handling is required. Adjust the values as needed to suit the specific configuration and safety requirements of your application

```
61 PROC PickNPlace_1(bool change_W)
62   MoveJ Home_pick,v100,z10,Servo\WObj:=wobj0;
63   MoveJ Home_center,v100,z10,Servo\WObj:=wobj0;
64   MoveJ Offs(Target_10,-Xpos,-Ypos,Safety_zone),v100,z10,Servo\WObj:=Wobj_pick; !Prepair to pick
65   WaitTime 1;
66   PulseDO Detach;
67   MoveJ Offs(Target_10,-Xpos,-Ypos,0),v100,z10,Servo\WObj:=Wobj_pick; !Pick
68   WaitTime 1;
69   PulseDO Attach;
70   WaitTime 1;
71   MoveJ Offs(Target_10,-Xpos,-Ypos,Safety_zone),v100,z10,Servo\WObj:=Wobj_pick;
72   MoveJ Home_center,v100,z10,Servo\WObj:=wobj0;
73   MoveJ Home_pick,v100,z10,Servo\WObj:=wobj0;
74   MoveJ Home_place,v100,z10,Servo\WObj:=wobj0;
75   MoveJ Center_Place_1,v100,z10,Servo\WObj:=Wobj_place;
76   MoveJ Offs(Target_place_10,-Xpos,-Ypos,Safety_zone),v100,z10,Servo\WObj:=Wobj_place;
77   MoveJ Offs(Target_place_10,-Xpos,-Ypos,0),v100,z10,Servo\WObj:=Wobj_place;
78   WaitTime 1;
79   PulseDO Detach;
80   WaitTime 1;
81 ENDPROC
```

1. Calculating Pick and Drop Coordinates with MOVEJ OFFS

• Target_10:

This is the base coordinate from which the offsets are applied. For example, if Target_10 is defined as [10, 10, 10], it serves as the starting position.

• Offsets (x, y, z):

The values x, y, and z are incremental offsets added to the base coordinates.

Example: **MOVEJ OFFS(Target_10, 20, 30, 40)**

The robot calculates the new destination as:

X coordinate: $10 + 20 = 30$

Y coordinate: $10 + 30 = 40$

Z coordinate: $10 + 40 = 50$

Therefore, the new position becomes [30, 40, 50].

This mechanism is used both for picking up objects (approaching the object with an offset) and for dropping objects (moving to a safe drop-off position).

2. Gripper Control with I/O Signals

After moving to the desired position, the robot uses specific I/O signals to control the gripper. Two main commands are involved:

PulseDO Attach (ClosePose: 10):

Purpose: This signal is used to pick up an object.

How it Works: When the robot reaches the pick-up position, the PulseDO Attach signal is sent. It activates the gripper in the ClosePose mode, which is configured with the value 10, thereby closing the gripper to securely hold the object.

PulseDO Detach (OpenPose: 15):

Purpose: This signal is used to release or drop an object.

How it Works: When the robot reaches the drop-off position, the PulseDO Detach signal is sent. It activates the gripper in the OpenPose mode, which is set with the value 15, allowing the gripper to open and safely release the object.

Conclusion

The handling function for pick and drop operations integrates precise coordinate calculation with robust gripper control:

Coordinate Calculation:

The MOVEJ OFFS(Target_10, x, y, z) command computes new positions by adding incremental offsets to a base target.

Gripper Commands:

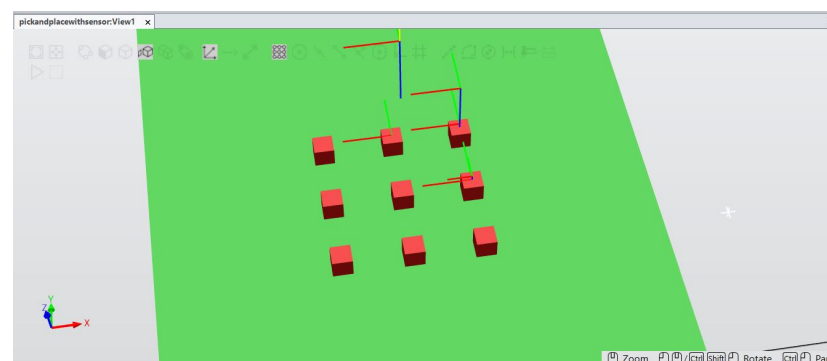
PulseDO Attach (ClosePose: 10) is used to close the gripper and pick up objects.

PulseDO Detach (OpenPose: 15) is used to open the gripper and release objects.

Together, these elements ensure that the robot accurately moves to the correct positions and safely handles objects during both the pick-up and drop-off phases.

```
PROC main()
    WaitTime 1;
    PulseDO Detach;
    FOR i FROM 1 TO 3 DO
        FOR j FROM 1 TO 3 DO
            Xpos := (i-1) * spacing;
            Ypos := (j-1) * spacing;

            PickNPlace_1(Change_W);
        ENDFOR
    ENDFOR
ENDPROC
```



In our workspace, we have 9 objects neatly arranged in a 3x3 box formation. This layout is not only visually organized but also highly predictable, which is ideal for systematic processing by the robot. To handle each object efficiently, we implement a dual-loop structure. One loop iterates through the rows, while a nested loop iterates through the columns. This ensures that every object in the grid is addressed one after the other.

The key to this operation lies in two arrays: Xpos and Ypos. These arrays store the positional coordinates of the objects along the X and Y axes, respectively. For instance, if the objects are equally spaced in the grid, the Xpos array might represent the horizontal positions of the columns, and the Ypos array might represent the vertical positions of the rows. As we traverse through the grid, the current values from Xpos and Ypos are used to determine the exact coordinates for each object.

The handling function `PicknPlace_1(Change_W)` plays a crucial role in this process. Its purpose is to update the current target position of the robot based on the new coordinates derived from the Xpos and Ypos arrays. The term "Change_W" refers to the change in the working position of the robot's end-effector, allowing it to move from one object to the next or from a pick position to a drop position. In essence, every time we update the coordinates for an object, we are telling the robot, "This is your new target location."

By leveraging this structured method, the entire process—from calculating the target positions to executing the pick and drop operations—is both efficient and scalable. Whether the objects are arranged in a simple grid or a more complex pattern, the fundamental approach remains the same, ensuring that every object is correctly encountered and handled according to the precise coordinates provided by the Xpos and Ypos arrays.