Pick[+] URCap guide

For Pick[+] version 1.3

This document serves as a brief guide on how to use the Pick[+] URCap, how to use it to create a custom Pick & Place application and how to take advantage of all its functionalities. This document explains the different program nodes provided by the Pick[+] URCap and how to use them. Additionally, it explains in detail the information provided by the Pick[+] server to the robot, available from the program in the form of global variables.
Examples on how to use the URCap also can be found on the several video tutorials that cover some of the most common functionalities of *Pick[+]*. More precisely, it is recommended to revise the videos called *Create a U-Pick application* and *K-Pick: application creation*. The videos can be found in the following link: https://www.bitmetrics.es/resources

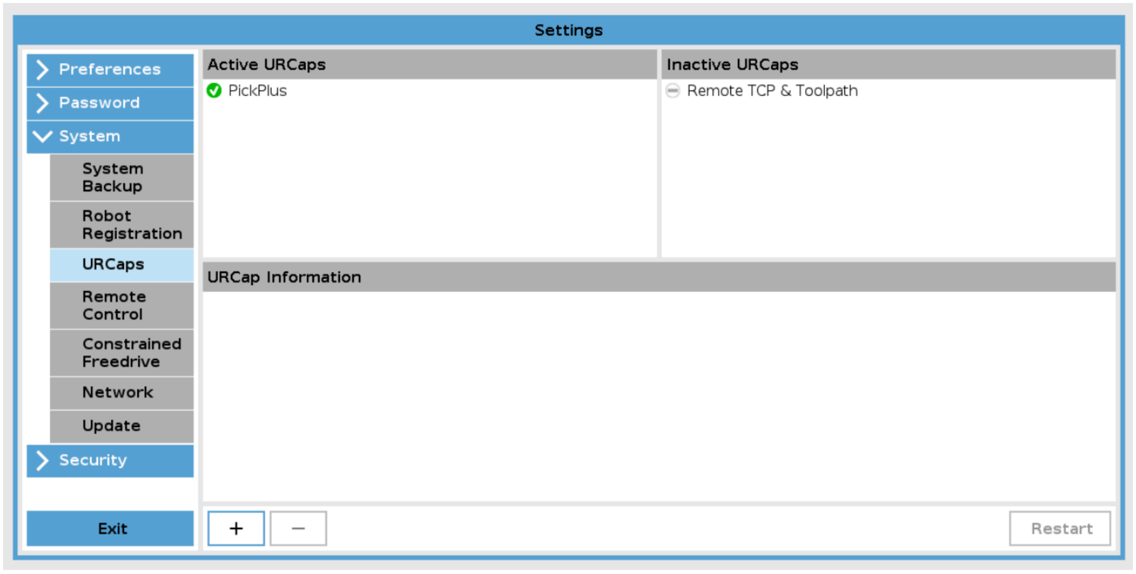
Installation

Prerequisites:

- URCap file (.urcap extension) located on a USB drive or downloaded from the Pick[+] repository (refer to the Installation Guide's "Downloads" section).

Procedure:

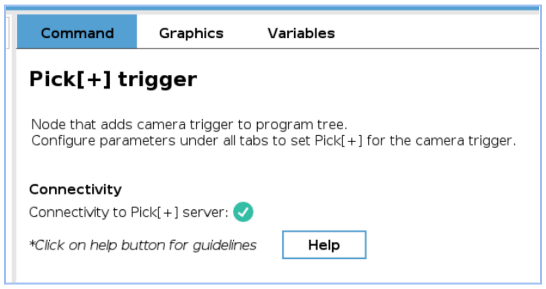
1. **Insert USB Drive:** Insert the USB drive containing the URCap file into the UR PolyScope teach pendant.
2. **Access URCaps Settings:** Navigate to the following path within UR PolyScope:
    - **Settings** > **System** > **URCaps**
3. **Add URCap:** Click the "**+**" button to open the file selection window. Locate and select the desired URCap file from the USB drive.
4. **Restart PolyScope:** Restart PolyScope to finalize the installation process.



Pick[+] URCap Nodes

This section provides an overview of the program nodes integrated within the Pick[+] URCap. These nodes are essential for the execution of Pick[+] applications, offering the necessary functions to interact with the Pick[+] system from your robot program. This section will cover their correct usage, configuration parameters, and specific features.

For all nodes, the user has access to help documentation from the teach pendant itself, accessed clicking on the *Help* button.



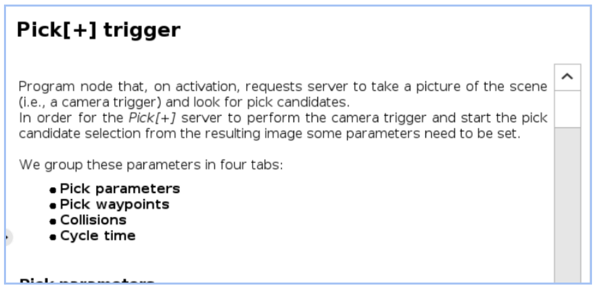
Pick[+] trigger

Node that adds camera trigger to program tree.
Configure parameters under all tabs to set Pick[+] for the camera trigger.

Connectivity
Connectivity to Pick[+] server: ✓

Click on help button for guidelines Help

→



Pick[+] trigger

Program node that, on activation, requests server to take a picture of the scene (i.e., a camera trigger) and look for pick candidates.
In order for the *Pick[+]* server to perform the camera trigger and start the pick candidate selection from the resulting image some parameters need to be set.

We group these parameters in four tabs:

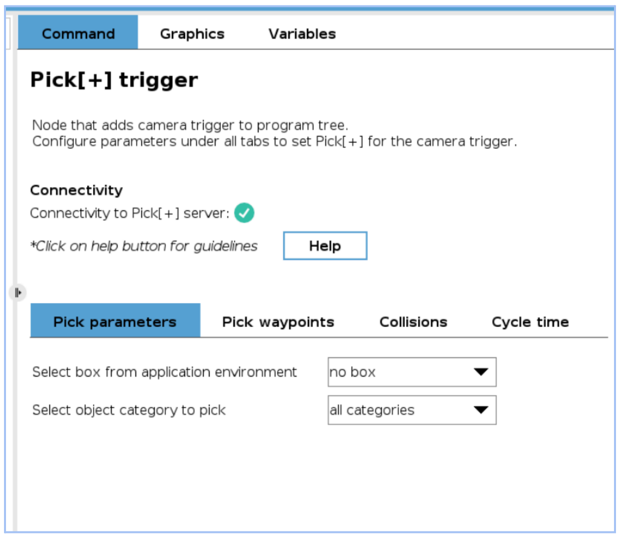
- **Pick parameters**
- **Pick waypoints**
- **Collisions**
- **Cycle time**

Pick[+] trigger

Functionality: Upon activation, the *Pick[+] trigger* node instructs the Pick[+] server to capture a scene image (camera trigger) and initiate the pick candidate detection process.

Configuration Parameters: To ensure effective image capture and pick candidate selection, the following parameter sets must be configured:

- **Pick Parameters:** To configure the detection (category of the object to pick and environment box)
- **Pick Waypoints:** Specify robot waypoints for the picking trajectory.
- **Collisions:** Enable/disable collision checking.
- **Cycle Time:** Configure cycle time measuring.



Command | Graphics | Variables

Pick[+] trigger

Node that adds camera trigger to program tree.
Configure parameters under all tabs to set Pick[+] for the camera trigger.

Connectivity

Connectivity to Pick[+] server: ✓

Click on help button for guidelines Help

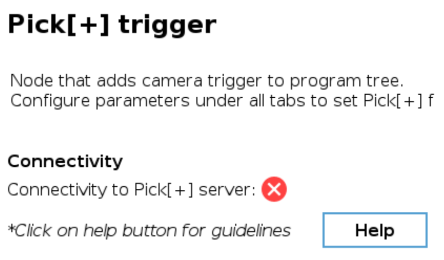
Pick parameters | Pick waypoints | Collisions | Cycle time

Select box from application environment	no box
Select object category to pick	all categories

Server connection required

To configure certain parameters of the *Pick[+] trigger* node (e.g., pick category, environment box), an active connection to the Pick[+] server in execution mode is required. This means the robot must be connected to the Pick[+] server, and a Pick[+] application must be running. If the connection is not established, the node will display an error message indicating the configuration failure.

Node without connection



Pick[+] trigger

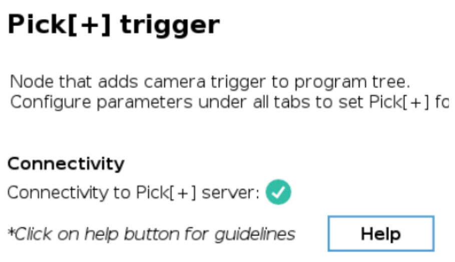
Node that adds camera trigger to program tree.
Configure parameters under all tabs to set Pick[+] f

Connectivity

Connectivity to Pick[+] server: ✗

Click on help button for guidelines Help

Node with connection



Pick[+] trigger

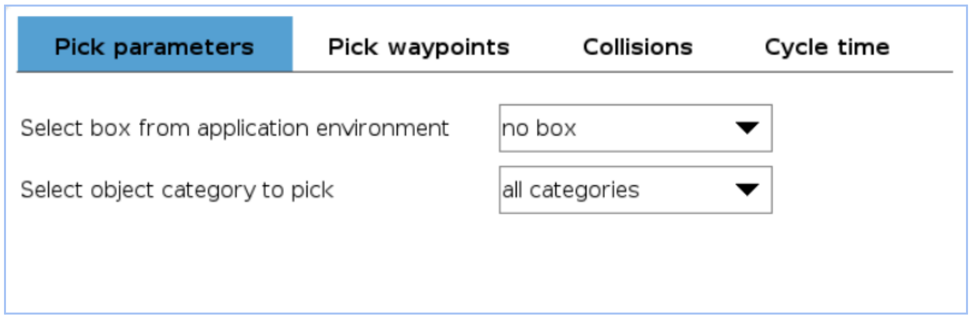
Node that adds camera trigger to program tree.
Configure parameters under all tabs to set Pick[+] fc

Connectivity

Connectivity to Pick[+] server: ✓

Click on help button for guidelines Help

Pick parameters



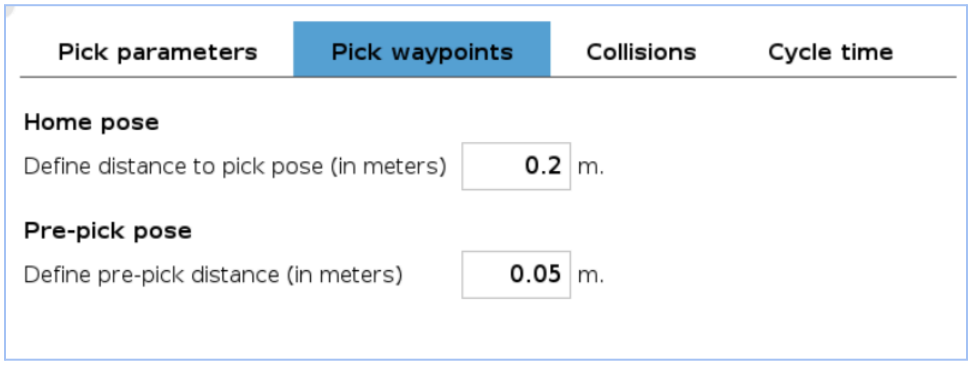
Select box from application environment: no box
Select object category to pick: all categories

The configuration of pick parameters involves two elements:

- **Environment Box**
  - This parameter references the environment box from the currently active pick environment for the Pick[+] application. By default, the field is set to ‘no box’.
  - The selected box will be used to filter out any detections outside its region, which can be useful to filter out undesired detections in cluttered environments. If “no box” is selected, this filtering step is not executed.
  - Additionally, the selected box will be used to compute the pick angle. The pick angle is the angle that the z-axis of the TCP will form with respect to the normal of the base of the box. Any candidate with an angle greater than the angle threshold (set in the *Pick[+] template* node) will be disregarded. If “no box” is selected, angle will be computed with respect to the default case:
    - For *eye-in-hand* configuration, the robot XY plane will be used as the reference working surface.
    - For *hand-eye* configuration, the pick angle will be computed with respect to the z-axis of the camera.
  - ⚠**WARNING**: While users retain the autonomy to use this feature, it is **strongly recommended to define and use a box.**
- **Pick Category**
  - This parameter specifies the category of interest from the set defined within the Pick[+] application. The field defaults to ‘all categories’, meaning all defined positive categories are considered eligible candidates for the trigger action.

⚠**WARNING**: The boxes and categories available in the current environment and application are updated once the Trigger node is opened. Therefore, if a different environment or application is set and the same robot program is used, it is necessary to open the Trigger node to ensure that the specified box and category exist in the current environment and application.

Pick waypoints



Home pose

Define distance to pick pose (in meters) 0.2 m.

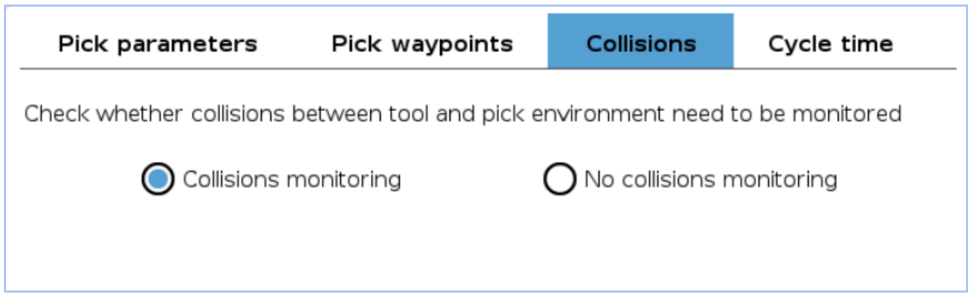
Pre-pick pose

Define pre-pick distance (in meters) 0.05 m.

The configuration of pick waypoints involves two elements:

- **Home pose:**
  - Serves as the starting point for the robot's pick action or trajectory. The home waypoint is automatically set and it's computed from the received pick point and the user-specified offset.
  - The home pose is obtained by translating the user-specified distance (in meters) away from the picking pose, in the negative direction of the pick point z-axis.
- **Pre-pick pose:**
  - Represents a pose slightly offset from the pick point, determined by the user-provided offset, in a similar manner as the home pose.
  - This pose is recommended for actuators/grippers with a variable Tool Center Point (TCP), such as certain finger grippers. In those cases, the pre-pick pose should be configured with the same distance that the TCP moves when actuating.
  - In scenarios involving a variable TCP actuator, the grip action should be initiated from the pre-pick pose.

Collisions



Check whether collisions between tool and pick environment need to be monitored

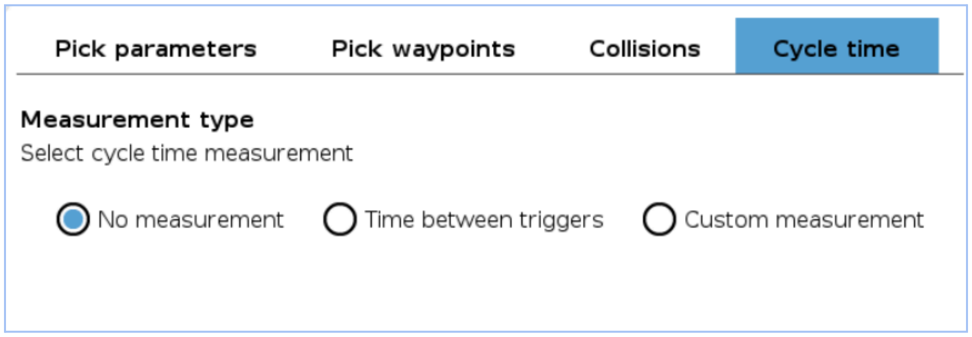
[✓ Collisions monitoring] No collisions monitoring

This tab allows to enable/disable the collision analysis of the pick points. The decision to perform the analysis is contingent upon the specific requirements of the application:

- For applications operating in a 2D environment, collision analysis may be deemed unnecessary. Omitting this analysis can result in a slight reduction of the cycle time.
- Conversely, in a 3D environment, conducting collision analysis is crucial to ensure the safety and efficacy of the robot's trajectory and actions.

⚠**WARNING**: While users retain the autonomy to activate or deactivate collision analysis, it is **strongly recommended to enable this feature**.

Cycle time



Measurement type
Select cycle time measurement

[✓ No measurement] Time between triggers Custom measurement

The cycle time measurement can be approached in three distinct ways, depending on the requirements of the application:

- **No measurement:**
  - In this mode, no cycle time measurement is conducted.
- **Time between triggers:**

    - Cycle time is automatically calculated based on the duration between two successive calls of a Pick[+] trigger node.
- **Custom measurement:**
    - Cycle time is measured using the *Pick[+] cycle timer* node, allowing for a custom measurement of cycle time. Refer to the *Pick[+] cycle timer* node section for more information.

If cycle time measurement is selected, the user will be able to set a threshold value. Surpassing this threshold will not cause any exception, but will be reflected on the value of the global variable *PP_error* (refer to the Global variables section). The current and average cycle time will be displayed on the Pick[+] client.

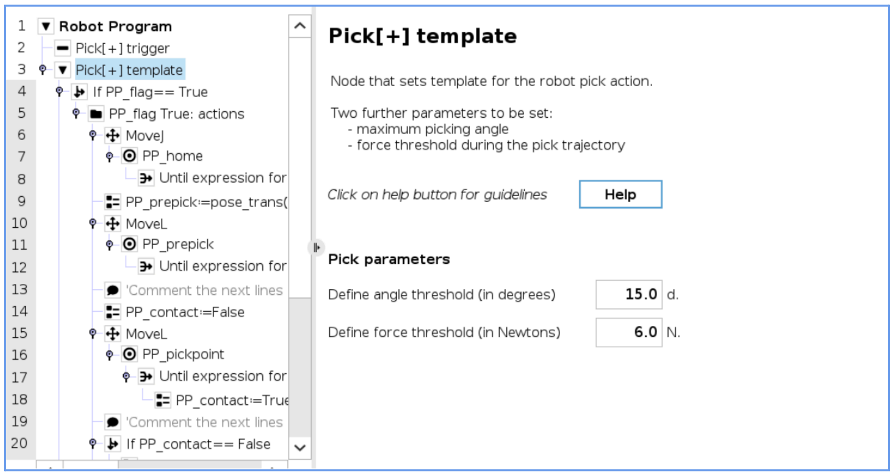
Pick[+] template

Functionality: This node has two main functionalities. First, it requests to the Pick[+] server the next pick candidate. Second, it generates a template for the robot pick program, including:

- Movement to the home pose (approximation pose).
- Movement to the pre-pick pose
- Movement to the pick pose
- Movement to the pick deeper pose

Configuration Parameters: The following parameter sets must be configured:

- **Angle threshold:** As commented in the *Pick[+] trigger* section, for every pick candidate received by the robot, its angle with respect to the working surface is measured (depending on the environment box selected in the *Pick[+] trigger* node. **If the measured angle is greater than the angle threshold, the candidate will be disregarded.** The measured angle can be monitored using the global variable *PP_angle*.
    - ⚠**WARNING**: The choice of camera is critical in determining the expected measured angle. While the expected angle of the pick pose might be 0 degrees, minor discrepancies in camera readings can result in a measured angle that exceeds 0 degrees.
    - ⚠**WARNING**: It is crucial to understand that the angle value set in this node is a threshold for candidate selection rather than a configuration of the pick pose itself. This value is utilized to eliminate unsuitable candidates and does not enforce the pick pose to conform to this angle.
- **Force threshold:** When moving to the different poses in the template, the force at the tool flange will be monitored in the form of an *Until condition*. This means that the robot will move towards the indicated pose until it reaches it or until the force at the tool flange is greater than the indicated threshold.



```
▼ Robot Program
    Pick[+] trigger
  ▼ Pick[+] template
      If PP_flag== True
        PP_flag True: actions
          MoveJ
            PP_home
              Until expression for
          PP_prepick:=pose_trans(
          MoveL
            PP_prepick
              Until expression for
          'Comment the next lines
          PP_contact:=False
          MoveL
            PP_pickpoint
              Until expression for
                PP_contact:=True
          'Comment the next lines
          If PP_contact== False
```

Pick[+] template

Node that sets template for the robot pick action.

Two further parameters to be set:
- maximum picking angle
- force threshold during the pick trajectory

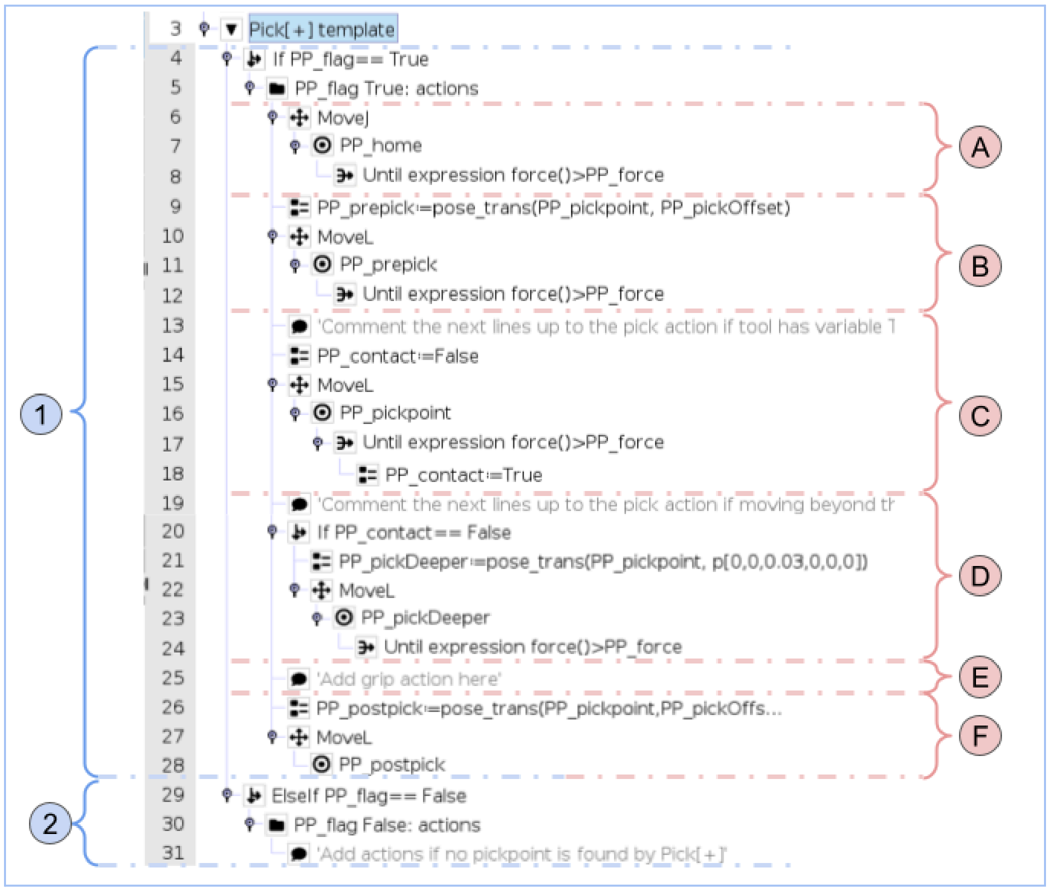
Click on help button for guidelines Help

Pick parameters

Define angle threshold (in degrees)	15.0	d.
Define force threshold (in Newtons)	6.0	N.

Template structure

When the *Pick[+] template* node is added to the program, a template of the robot program to perform the pick action is generated. The following figure shows the generated template.



```
▼ Pick[+] template
    If PP_flag== True
      PP_flag True: actions
        MoveJ
          PP_home
            Until expression force()>PP_force
        PP_prepick:=pose_trans(PP_pickpoint, PP_pickOffset)
        MoveL
          PP_prepick
            Until expression force()>PP_force
        'Comment the next lines up to the pick action if tool has variable T
        PP_contact:=False
        MoveL
          PP_pickpoint
            Until expression force()>PP_force
              PP_contact:=True
        'Comment the next lines up to the pick action if moving beyond th
        If PP_contact== False
          PP_pickDeeper:=pose_trans(PP_pickpoint, p[0,0,0.03,0,0,0])
          MoveL
            PP_pickDeeper
              Until expression force()>PP_force
        'Add grip action here'
        PP_postpick:=pose_trans(PP_pickpoint,PP_pickOffs...
        MoveL
          PP_postpick
    ElseIf PP_flag== False
      PP_flag False: actions
        'Add actions if no pickpoint is found by Pick[+]'
```

- ① **Program executed if a candidate has been found**
    - Ⓐ Movement to the home pose

      This part of the program moves the robot to the home pose (or approximation pose). This pose is computed based on the configuration set in the *Pick[+] trigger* node. By default, the movement is done in the joint space (MoveJ) and with the force *Until condition*. However, for this movement in particular, this *Until condition* can be disregarded if the user sees it fit.

    - Ⓑ Movement to the pre-pick pose

      This part of the program moves the robot to the pre-pick pose, computed based on the configuration set in the *Pick[+] trigger* node. **This pose is only necessary for grippers with variable TCP without self compensation.** As explained in the *Pick[+] trigger* section, these kinds of grippers may require that the tool is activated at a certain distance from the piece. This pose can be used for that purpose.

      ⚠Comment out or delete this part of the program if your gripper doesn't match the previous description. Otherwise, this pose will only add cycle time.

      ⚠If used, it's recommended to change the *Until condition* from *Expression* to *Until contact* (if possible).

    - Ⓒ Movement to the pick pose

      This part of the program moves the robot to the pick pose. This pose is the actual pose estimated by the Pick[+] server. In addition to the *Until condition* used to monitor the force on the tool flange, **it is recommended to use any extra sensors that are available**. For example, if a vacuum gripper is used and it has a signal (analogue or binary) that indicates whether the vacuum is successful or not, an *Until condition* can be added with that signal. In this way, the robot will move towards the pick pose until it arrives, until a contact force is detected or until the vacuum has been performed. This ensures a more accurate and safer pick for the robot and the rest of the elements.

⚠It's recommended to change the *Until condition* from *Expression* to *Until contact* (if possible).

- Ⓓ Movement to the pick deeper pose

  If no contact has been detected when moving to the pick pose, this part of the program will move the TCP of the robot a small distance towards the object. This additional movement enables the correction of pick point estimation errors that occur when the pick point is located above the object, either due to a poor camera reading or other reasons.

  ⚠It's recommended to change the *Until condition* from *Expression* to *Until contact* (if possible).

- Ⓔ Grip action

  If the tool requires to be at the pick point to be activated, the user should actuate the gripper here. However, depending on the tool and application, it may be more effective to activate the tool at a different point in the program. For instance, a vacuum gripper may work better if the vacuum is activated at the home pose rather than upon contact with the part. This also allows for vacuum sensor monitoring, as mentioned in a previous example.

- Ⓕ Movement to the post pick pose

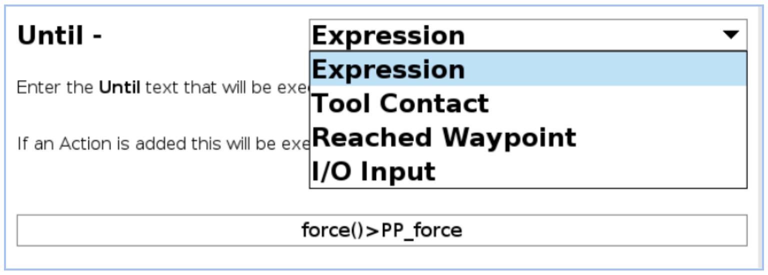
  Once the robot has grasped the object, this section of the program moves the robot to a pose away from the object. By default, this position is the same as the pre-pick pose. The user can choose to use the home pose instead (by changing the waypoint from *PP_postpick* to *PP_home* in the program) or to use another position of their choice.

- ② **Program executed if no candidate has been found**

If no suitable candidates have been detected or all have been discarded for other reasons, such as being out of the box or incorrect pick angle, this part of the program will be executed. The user is free to add any additional actions here.

Key considerations

- To be able to add a *Pick[+] template* node to the program, at least one *Pick[+] trigger* node must be present as well.
- In all movements involving the *Until condition* with the force threshold (Ⓐ-Ⓓ), it's recommended to use (if possible), the *Until contact* condition instead.

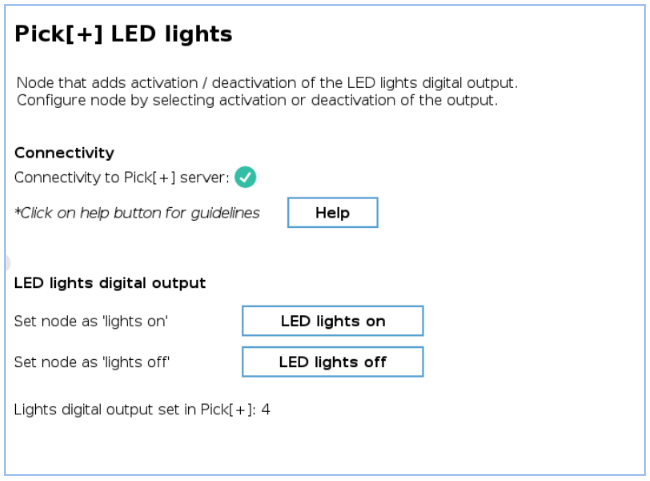


- If no pick candidates are found after an iteration, the variable *PP_triggers* will be incremented by 1. When this variable reaches 3, a pop-up will be launched, stopping the execution of the program and informing the user that no picks have been found. Depending on its value, this variable can be used to change the photo pose, allowing for up to 3 different photo poses. This can be particularly useful for bin-picking, as changing the camera angle could detect objects that were previously unidentifiable from the previous pose. **If a candidate is detected, *PP_triggers* will be reset to 0.**

Pick[+] LED lights

Functionality: The *Pick[+] LED lights* node is designed to control the camera illumination by activating or deactivating the corresponding digital output. To ensure proper operation, the corresponding digital port must be configured within the Pick[+] main interface. For detailed setup instructions, refer to the *Application Settings* section in the Pick[+] guide.

Configuration: When adding this node to the program, the user is required to specify the desired state of the lights—either to turn them ON or OFF.


Pick[+] LED lights

Node that adds activation / deactivation of the LED lights digital output.
Configure node by selecting activation or deactivation of the output.

Connectivity
Connectivity to Pick[+] server: ✓
*Click on help button for guidelines

LED lights digital output
Set node as 'lights on'
Set node as 'lights off'

Lights digital output set in Pick[+]: 4

Server connection required

To obtain from the Pick[+] server to which digital port the camera lights are connected, an active connection to the Pick[+] server in execution mode is required. This means the robot must be connected to the Pick[+] server, and a Pick[+] application must be running. If the connection is not established, the node will display an error message indicating the configuration failure.

Key considerations

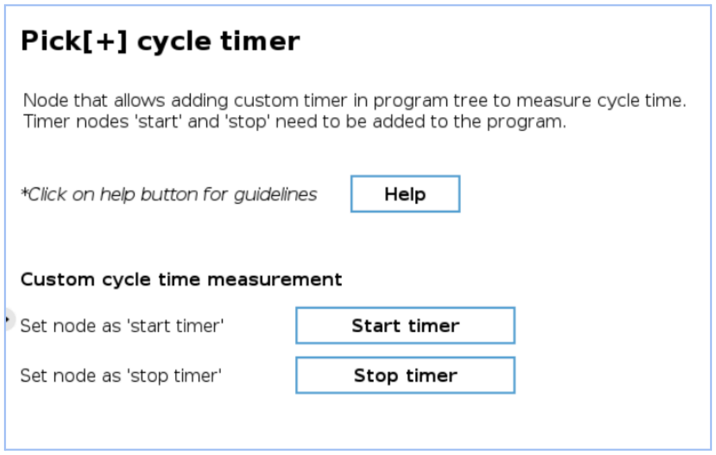
- Continuous operation of lights is not recommended. It is preferable to engage the lights only as needed rather than maintaining them in an always-on state. This intermittent approach promotes more effective heat dissipation, which is particularly beneficial in environments with elevated temperatures.
- To make sure that the Pick[+] server captures the image with the lights ON, they should be switched OFF at least 0.3 seconds after the camera trigger is

performed. The value of 0.3 seconds is a guideline, the actual limit value may vary depending on the equipment and camera used. The user is recommended to test with a higher value (0.5 seconds) and go as low as possible.

Pick[+] cycle timer

Functionality: When the *Custom Measurement* option is selected during the setup of the Pick[+] trigger node, this node enables the cycle time measurement of the process, measuring the time elapsed between two specific points in the robot program.

Configuration: When adding this node to the program, the user is required to specify if the node should start or stop the timer.



Pick[+] cycle timer

Node that allows adding custom timer in program tree to measure cycle time.
Timer nodes 'start' and 'stop' need to be added to the program.

Click on help button for guidelines

Custom cycle time measurement

Set node as 'start timer'

Set node as 'stop timer'

Global variables

This section outlines the global variables generated by the URCap program nodes. These variables can be helpful to the user when programming the rest of their program.

PP_flag

Boolean variable that indicates whether a candidate has been selected for pick or not.

PP_pickpoint

This global variable stores the pick pose estimated by Pick[+]. The pose is defined with respect to the robot base.

- **Variable type**: Pose variable.
- **Example**: PP_pickpoint = p[0.2, 0.2, 0.0, 2.15, 2.15, 0]

PP_home

This global variable stores the home pose. The home pose is the robot arm's safe starting spot, used to approach the picking pose. Essentially, the home pose is a translation of the pick pose in the negative direction of the picking pose's z-axis, moving away from the object.

- **Variable type:** Pose variable.
- **Example:** PP_home = p[0.2, 0.2, 0.2, 2.15, 2.15, 0]

PP_prepick

This global variable stores the pre-pick pose.

- **Variable type:** Pose variable.
- **Example:** PP_home = p[0.2, 0.2, 0.2, 2.15, 2.15, 0]

PP_pickDeeper

This global variable stores the *pick deeper* pose.

- **Variable type:** Pose variable.
- **Example:** PP_home = p[0.2, 0.2, 0.2, 2.15, 2.15, 0]

PP_postpick

This global variable stores the post-pick pose.

- **Variable type:** Pose variable.
- **Example:** PP_home = p[0.2, 0.2, 0.2, 2.15, 2.15, 0]

PP_category

This global variable stores the name of the category of the object to be picked. If no suitable candidate was detected, its value is "-1 (None)".

- **Variable type:** String variable.
- **Example:** PP_category = "object"

PP_pickID

This global variable stores the pick point ID for the selected pick. If alignment was selected for the detected category (by texture or geometry), this variable identifies the picking point in the currently active picking point set. If a suitable candidate is detected its value is "0", "1", etc. If not, its value is "-1 (None)".

- **Variable type:** String variable.
- **Example:** PP_pickID = "0"

PP_error

This global variable stores an error code that can be used to monitor and debug the program.

- **Variable type:** Integer list variable. Format: [C1, C2, C3]
- **Notes:** The values of the elements on the list represent the error type (C1), error code (C2) and error sub-code (C3). The possible values of this variable are explained in the following table.

C1	C2	C3	Description
0	0	0	No error or warning to report.
0	2	x	⚠ **Warning**: current pick pose is not reachable. Based on x value: - 1: pick pose not reachable - 2: pre-pick pose not reachable - 3: home pose not reachable

			- 4: pick pose angle too high
1	0	0	⚠️**Warning**: Cycle time is greater than the specified threshold.
2	0	0	⚠️**Disconnection error**: Could not communicate with Pick[+].
3	x	x	⚠️**Communication error**: Could communicate with Pick[+] but there was a problem sending/receiving data.

PP_angle

Variable that takes as value the angle of the current pick candidate. The first element of the array contains the angle (in degrees) of the current candidate. A value of -1 indicates no candidate is currently being analyzed. The second element is for internal use only.

- **Variable type**: Float array.
- **Example**: PP_angle = [15, 1]

PP_boxOrigin

This global variable stores the origin point of the selected environment box. If a box or working surface was selected in the *Pick[+] trigger* node, the variable contains the origin of the picking box or working surface coordinate system. This pose lets the system filter objects by angle for better picking. If none is selected ("no_box"), the angle is still calculated, but differently based on the camera setup (*eye-in-hand* or *hand-eye*). For the safest and most reliable results, always define a box or working surface.

- **Variable type:** Pose variable.
- **Example:** PP_boxOrigin = p[0.2, 0.2, 0.2, 0, 0, 0]

PP_camSetting

Variable that indicates the camera setting we are working with: *hand-eye* or *eye-in-hand*.

- **Variable type:** String variable.
- **Example:** PP_camSetting = "eye-in-hand"

PP_pickOffset

Variable that takes as value the offset from the pick point.

PP_homeOffset

Variable that contains the offset with respect to the home pose selected by the user.

PP_calibration

Variable that stores the camera calibration. For *eye-in-hand,* it's the pose of the camera with respect to the tool flange. For *hand-eye,* it's the pose of the camera with respect to the base of the robot.

PP_connectivity

Boolean variable that monitors connectivity with the Pick[+] server.

PP_triggers

Variable that registers the number of consecutive triggers without any successful pick (up to 3 at maximum). This can be used to change the photo pose when no candidate is detected. If a candidate is found, the value will be reset back to 0.

PP_unreachable

Counter variable for the unreachable pick points among all candidates found.

PP_force

This global variable takes as value the force threshold set by the user.

PP_contact

Flag that indicates whether the TCP has made contact with an object.