

Photoneo Add-In Manual

Photoneo Add-In 1.0.1

Checked in 2025-10-10

What is a Photoneo Add-In?

Photoneo is a leading provider of robotic vision and intelligence. Based on a patented 3D technology, Photoneo developed the world's highest-resolution and highest-accuracy 3D camera, thus unlocking the full potential of powerful, reliable, and fast machine learning and also reducing the training and deployment time.

The Photoneo add-in installs all necessary robotic modules, wizard, webapp and performs the required reconfiguration of the controller to save as much time as possible when starting a new project.

Localization products

Locator Studio is a fast and intuitive tool for 3D object localization in environments with minimal collision risk.

Key Features:

- Intuitive web interface with simple setup
- Compatibility with various robot types (6-axis, SCARA, DELTA, 7-axis)
- Supports extrinsic and hand-eye calibration
- Communication via open TCP/IP protocol
- Rapid deployment without the need for advanced path planning
- Built-in CAD-based object localization engine
- Built-in AI-based object localization engine

Typical Applications:

- Picking oriented or semi-oriented parts from flat surfaces
- Simple tasks like delayering, destacking, deracking
- Accurate placement for assembly or screwing operations

Bin Picking Studio is a comprehensive solution for advanced robotic picking from unstructured environments.

Key Features:

- Advanced path planning with collision avoidance
- Extensive library of supported 6-axis robot models
- Supports extrinsic and hand-eye calibration
- Communication via open TCP/IP protocol
- Smart tools for grasp point selection and interactive tuning
- Built-in CAD-based object localization engine
- Built-in AI-based object localization engine

Typical Applications:

- Picking randomly placed items from deep bins
- Tasks requiring advanced trajectory planning and collision checking
- Applications involving multiple object types and complex setups

How to install the Hardware



When connecting the Vision Controller and the robot controller, it is important to determine whether the devices should communicate via a Public or Private network. A Private network enables redirecting the web application from the Vision Controller directly to the FlexPendant screen.

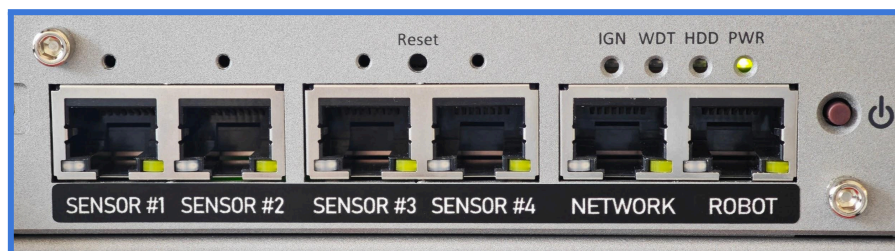
Private Network:



Public Network:



These are the ports located on the Vision Controller:



Product dependencies

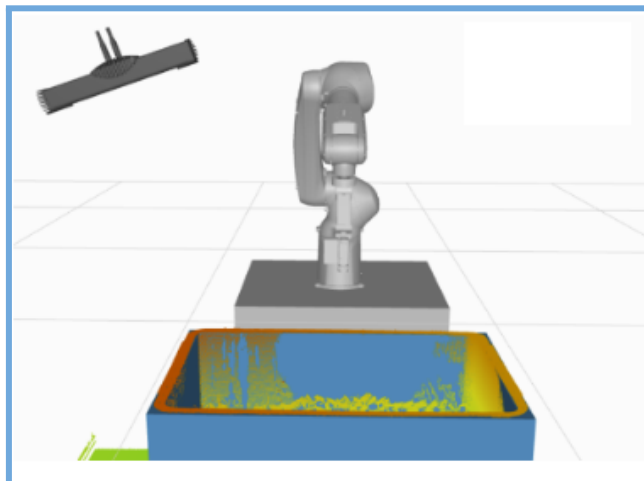
Options Requirements:

- The license for **Omnicores** must include the **Multitasking** options for the **Bin Picking** application.
For the **Locator** application **does not necessary any special license**.
- **RobotWare** must be version **7.17** or higher for Omnicores.

How you can install Photoneo devices

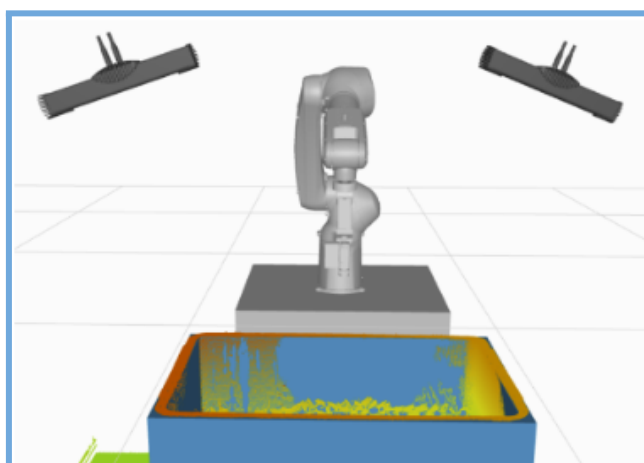
Extrinsic

The sensor is mounted in a fixed position above the bin and calibrated with the robot. The position of the localized object is transformed into the robot base frame using a calibration matrix.



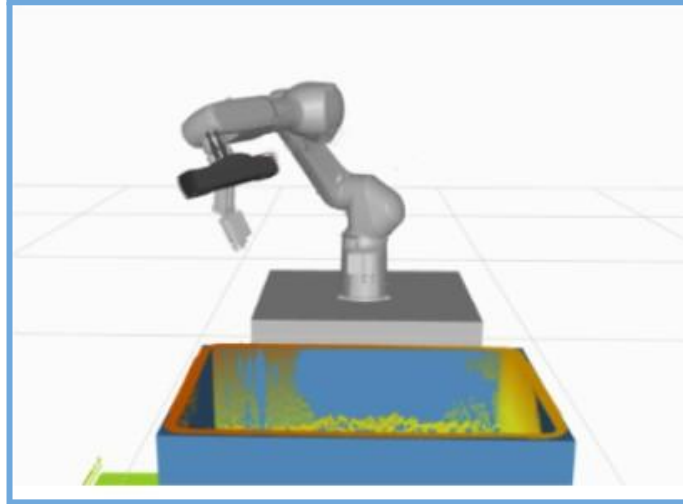
Multiview vision system

One primary sensor is calibrated with respect to the robot, while additional sensors are calibrated relative to the primary sensor. Each sensor performs scanning sequentially.



HandEye

This approach uses a Photoneo 3D sensor mounted on the robotic arm. This allows for virtually unlimited scans from various viewpoints instead of just a single scan.



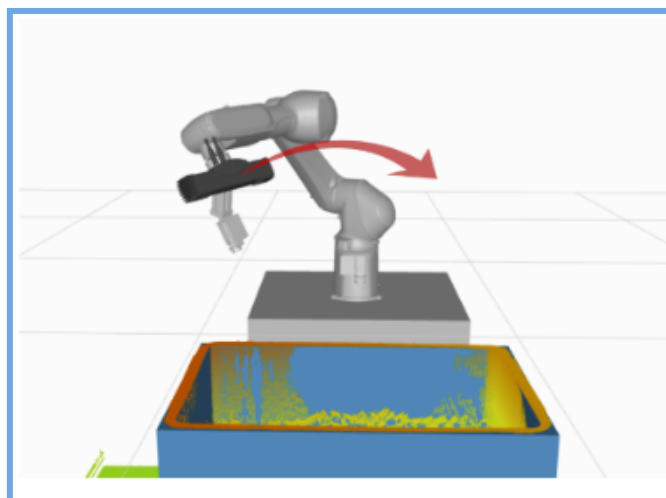
HandEye Multiview Static or Dynamic

This approach uses a Photoneo 3D sensor mounted on the robotic arm. In a static setup, the robot moves to a specific viewpoint and **stops completely** before triggering the scanner to capture a 3D image. This "stop-and-scan" process is repeated from several different positions until the entire object is covered.

This method is essential for scanners that project a light pattern, like the Photoneo PhoXi or MotionCam-3D, because they require a perfectly still scene to ensure a sharp, accurate capture.

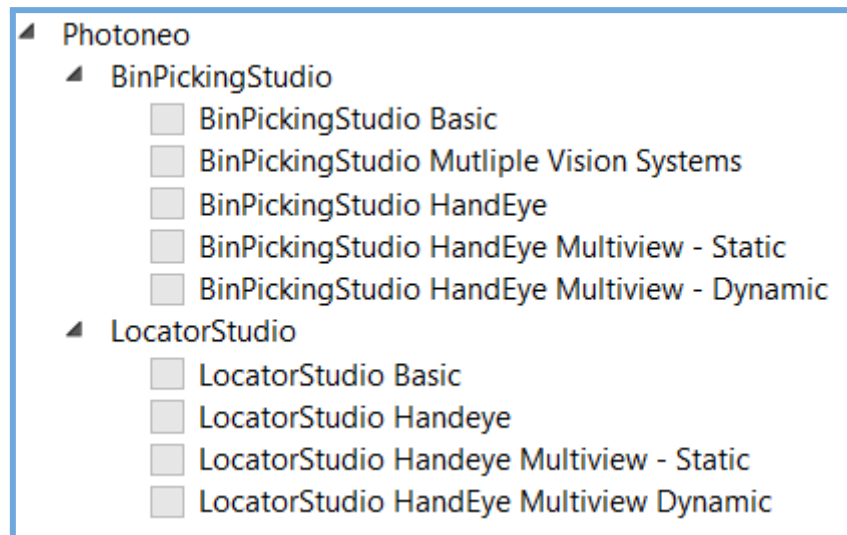
In a dynamic setup, the MotionCam-3D captures data while the robot is in **continuous motion**. The scan is performed "on-the-fly" as the robot arm moves smoothly along a pre-planned path around the object.

Ultimately, the data from both methods is combined to create a single, comprehensive point cloud, which is then used to generate a complete 3D mesh of the Scene.



Different modules according to the application

In the Photoneo Add-in, you'll find a variety of robotic module examples and a comprehensive library of requests for communication between the robot and the vision controller.



The Basic means that the device is mounted extrinsic. The process operates in a continuous loop consisting of scanning, localisation, and pick-and-place actions.

The Multiple Vision System example demonstrates how to use more than one device. For example: an application with more than one bin = more devices, one device and one solution but more vision systems inside.

The HandEye example can be used when the device is mounted on the effector.

The HandEye Multiview Static / Dynamic example is suitable when the device is mounted on the end effector, allowing the robot to scan the scene from multiple positions. The final object localization is performed on a mesh generated from these combined scans.

Dynamic: Time-Based Scanning (Default)

By default, the system is set to scan based on time intervals. This is controlled by the **nFrequency** variable.

Dynamic: Distance-Based Scanning

To switch to distance-based scanning, the **nScanningDistance** variable is used to define the distance the robot must travel between each scan.

To enable this mode, you must modify the trap routines:

1. **Comment out** the trap routine definition for time-based scanning.
2. **Uncomment** the trap routine for distance-based scanning.

Note: When using distance-based scanning, it is recommended to set the **nFrequency** variable to a low value (e.g., 0.1) to ensure proper operation.

The modules ensuring communication between the robot and the visual controller are:

- **Pho_BPS_BASE.sys** (Bin picking studio)
- **Pho_LS_BASE.sys** (Locator studio)

Pho Customer Definition module (Bin picking) contains procedures that handle actions such as opening, closing the gripper and picking of the objects.

Pho State Server module (Bin picking) contains a loop that monitors the permission for socket communication over either the private or public network and sends the robot's current position to the Vision Controller. This module contains the Firewall check in the robot system.

Pho BPS / LS Calibration is a module that contains a procedure for teaching calibration positions, as well as a procedure for starting the automatic calibration process.

Pho BPS / LS Wizard is a module that contains procedures with logic for block-based programming.

What is installed automatically?

- Rapid program and system modules according to the selected product (Bin picking studio / Locator studio)
- Configuration files (Sys.cfg) with automatic load modules or background task
- Copy examples to the Home folder of the robot controller according to the selected product
- Photoneo.coblox file for Wizard and modules with source code according to the product
- WebApp – a redirect from the web server on the Photoneo Visual Controller.

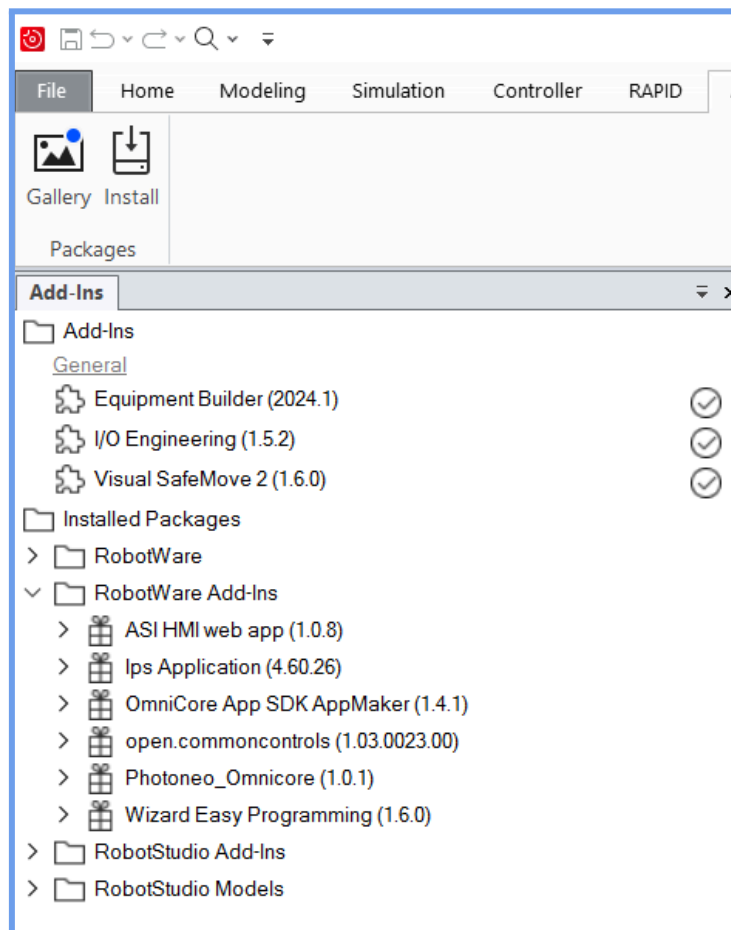
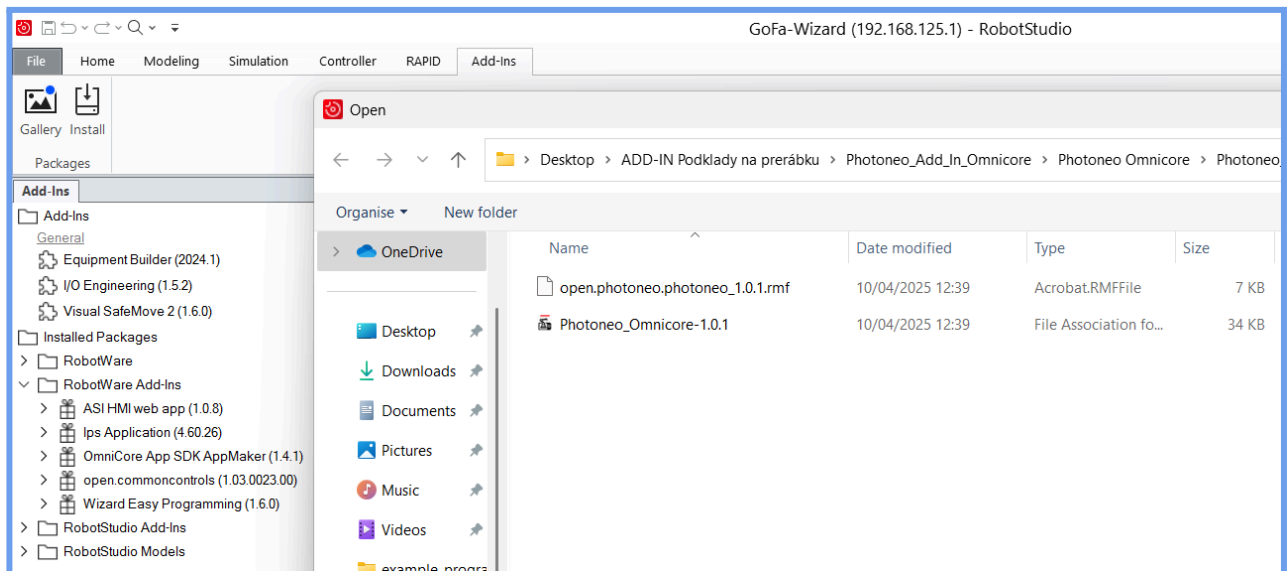
How to install Photoneo Add-In by RobotStudio

1. Download the Required Application

Download the **Photoneo Add-In** from our web - [BinPickingStudio](#) or [LocatorStudio](#).

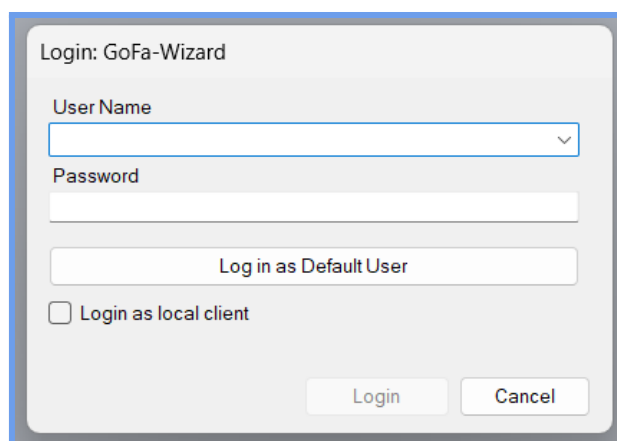
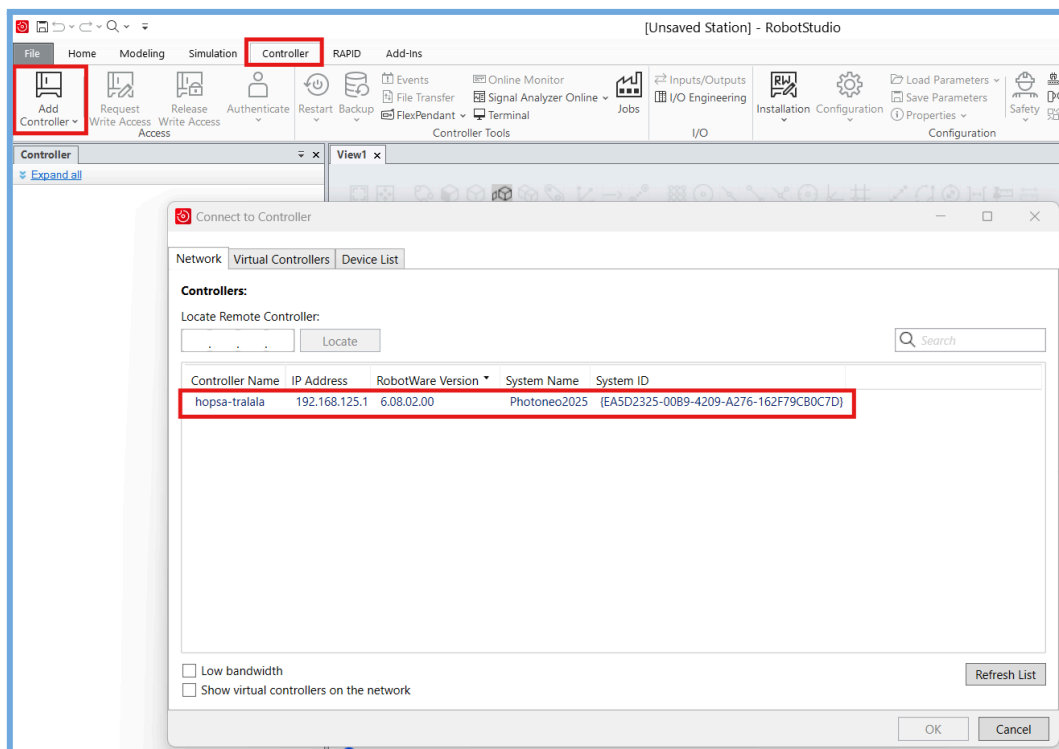
2. Install the Add-in in RobotStudio

1. Open **RobotStudio**.
2. Navigate to the **Add-Ins** window.
3. Click the **Install** button.
4. Select **Photoneo Add-In**.
5. Click **Open** to install the Add-in.
6. The Add-in will be automatically installed in RobotStudio.

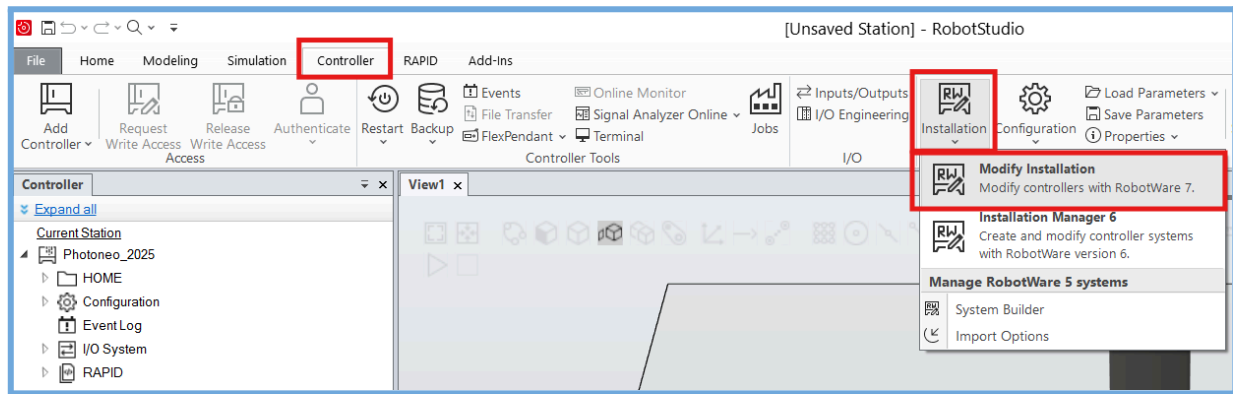


3. Create a New System or Add the Add-in to an Existing System

1. Open the **Controller** window.
2. Click **Add Controller**.
3. Double-click your controller in the list.
4. Sign up as an Admin user or user with a grant to adjust the system and program.

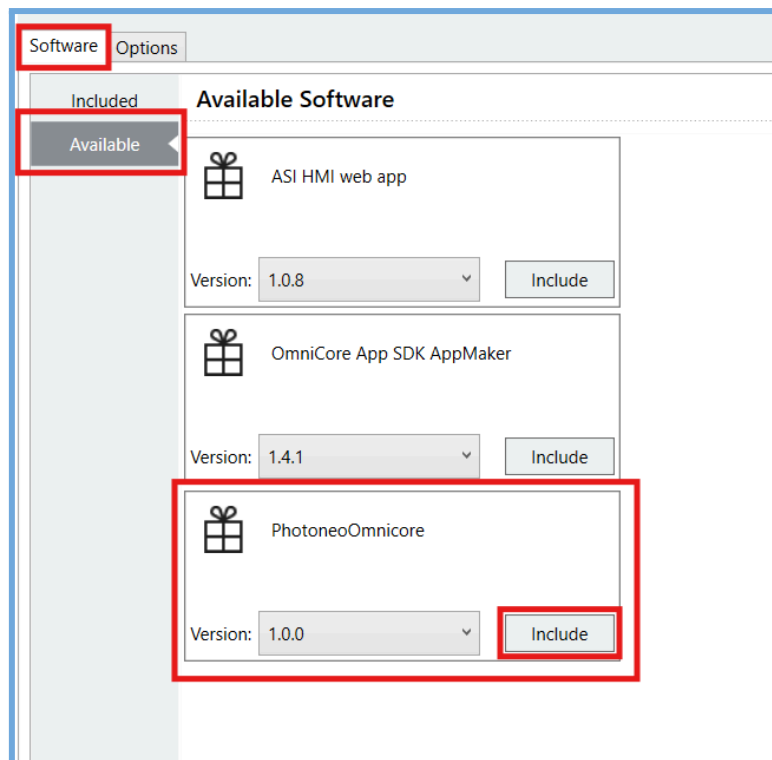


4. In the **Controller** window, click **Installation**, then select **Modify Installation**.



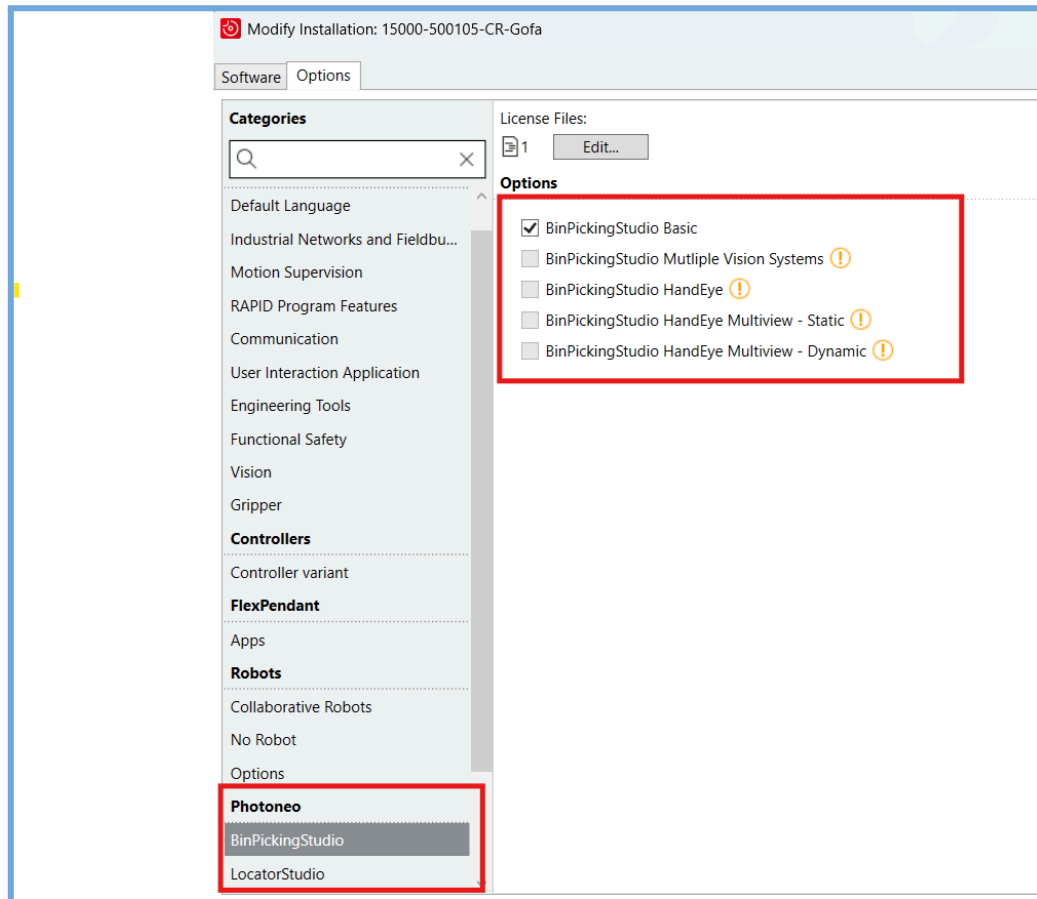
5. In **Modify Installation** you can adjust your Omnicore system

- In windows **Software** click **Available**.
- Select **PhotoneoOmnicore** from the installed add-ins.
- Confirm with **Include**.



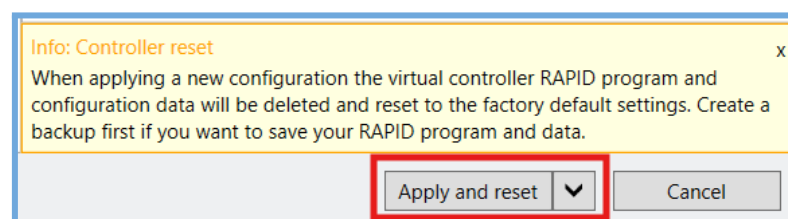
6. In the **Options** window, scroll to the bottom and select a product - **Bin Picking Studio** or **Locator Studio**.

- In the middle section of the window, choose a module example that best fits your application.



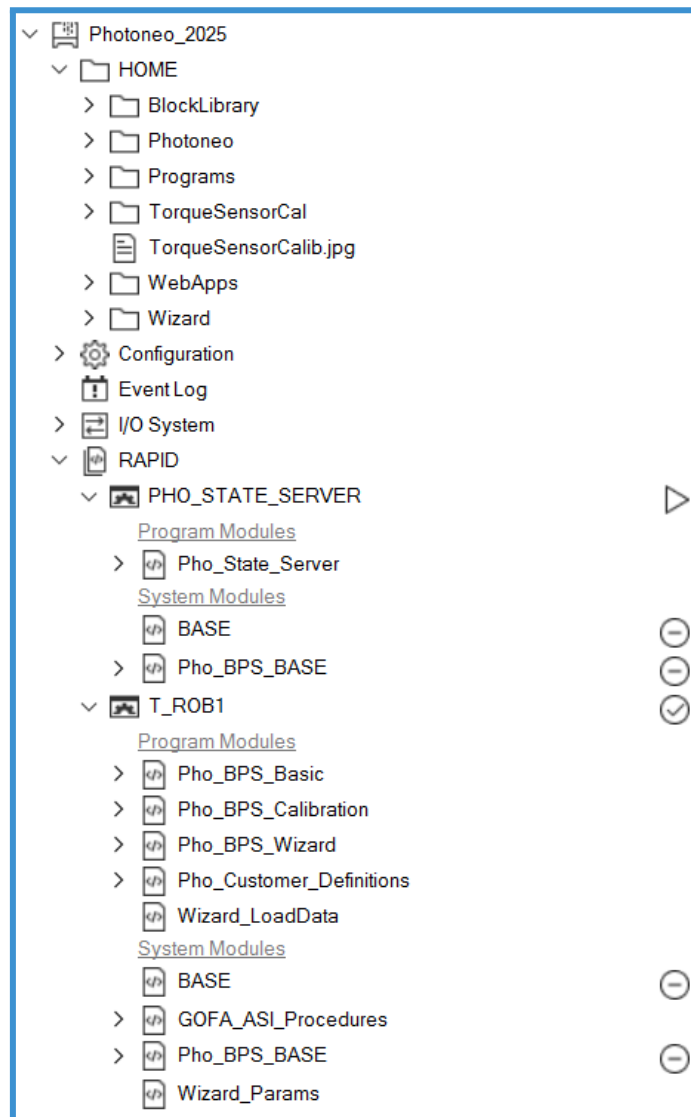
7. In the final step, review your selections and click **Apply and reset**.

- Ensure you have **write access** on the FlexPendant.
- When applying a new configuration the controller RAPID program and configuration data will be deleted and reset to the factory default settings. Create a backup first if you want to save your RAPID program and data.

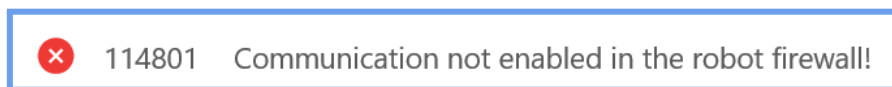


8. Now it's time for a coffee break. Enjoy!

The installed system will depend on your selected product. The final setup might resemble the example shown below (**BPS Basic**).



NOTE:



To enable communication between the Robot Controller and the Vision Controller, socket communication must be allowed:

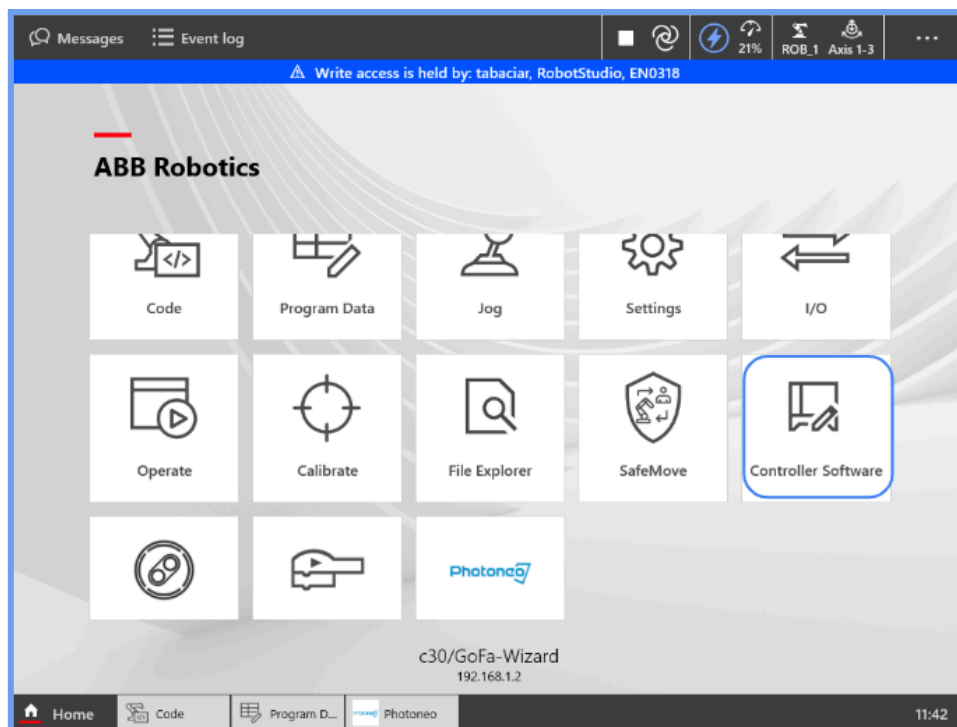
RobotStudio → Configuration → Communication → Firewall Manager → RapidSockets → Public / Private → Yes / Yes → Restart Robot Controller

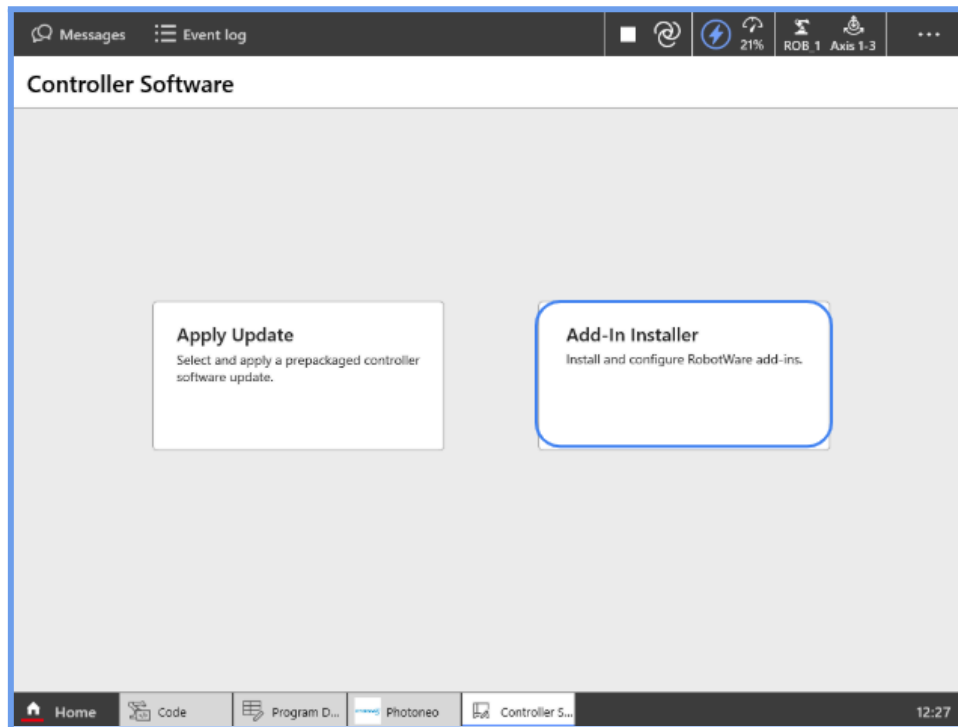
If you're using the private network (typically on a 192.168.125.xxx network), you must enable the firewall rule for the RAPID socket on the private network. If you're using the **WAN port** for your connection, you **must enable** the firewall rule for the RAPID socket on the **public network**.

The **BPS** checks these settings as part of the state server, since without enabling this communication, the state server cannot operate. The **LS** checks these settings during the initialisation of communication with the Vision Controller.

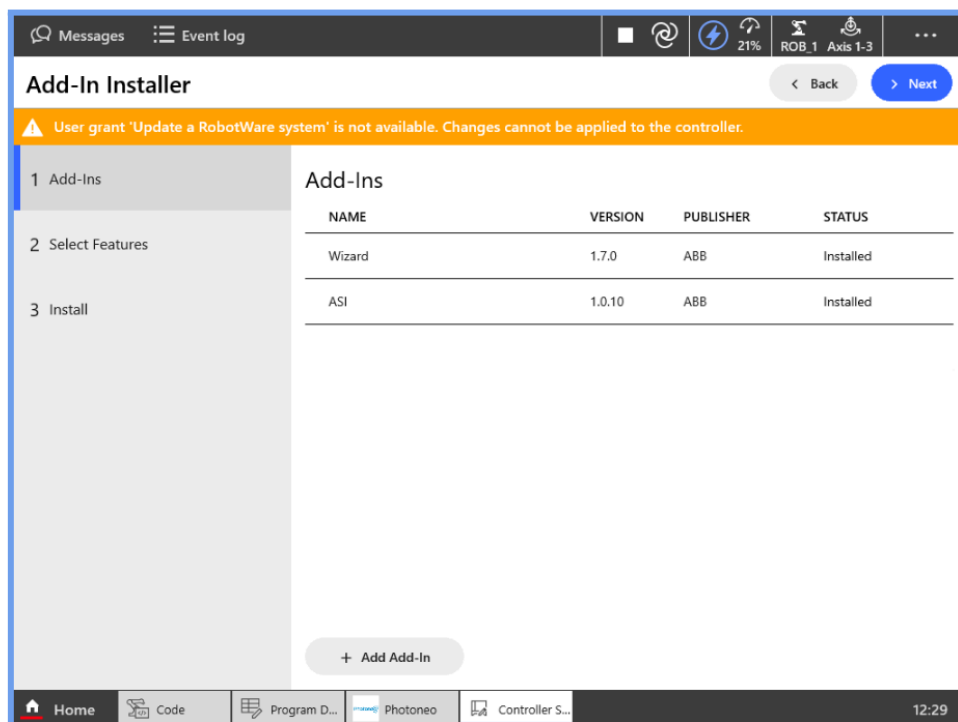
How to install Photoneo Add-In with USB stick

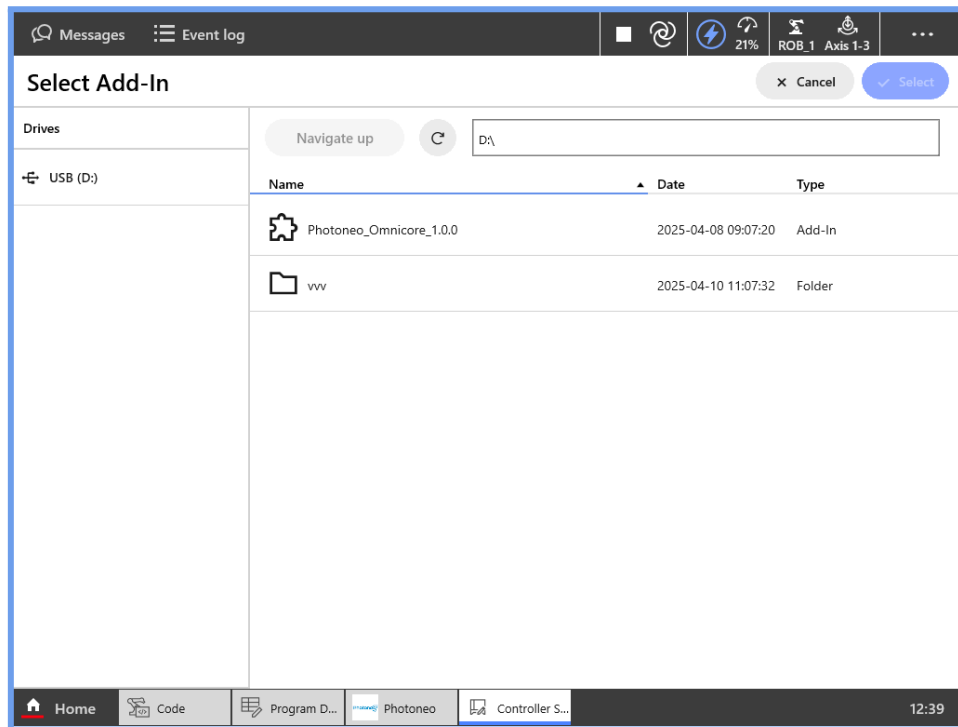
1. Download the **Photoneo Add-In** from our web - [BinPickingStudio](#) or [LocatorStudio](#) and place it on a USB stick.
2. Insert the USB stick into the FlexPendant.
3. Make a backup.
4. Sign up as an Admin user (User: admin | Password: robotics) or a user with a grant to adjust the system and program
5. Open the *Controller software on Flexpendant* → Add-In Installer



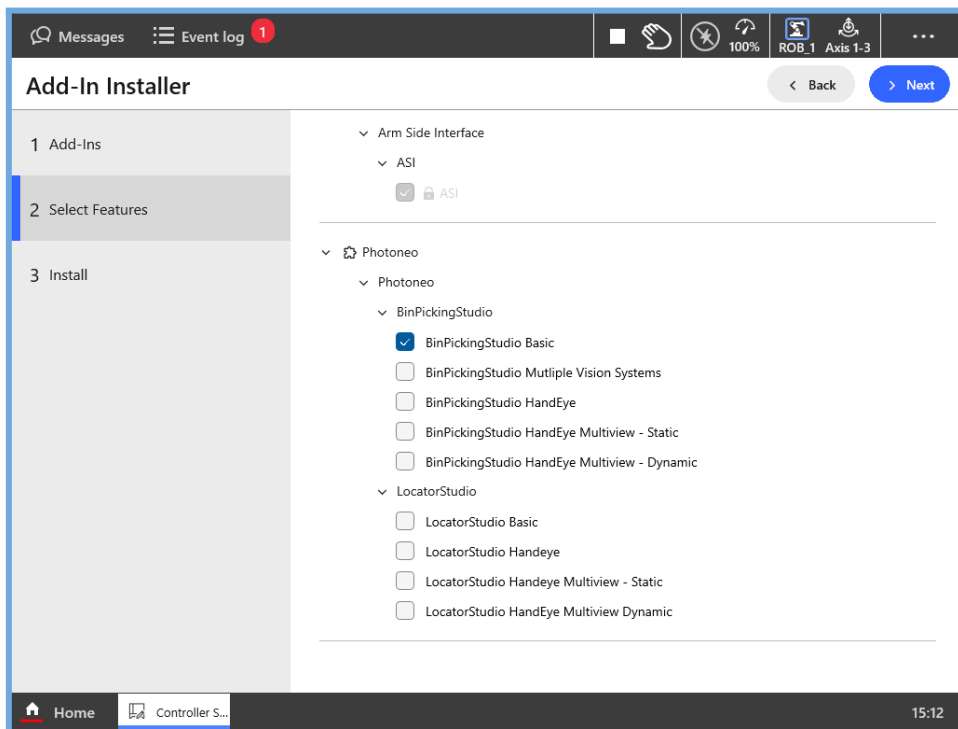


6. Press “+ Add Add-In” button and select the *Photoneo-Omnicores Add-In*.





7. Choose the product that best suits your application type.



8. Click *Next* and then *Apply* to complete the setup.

When applying a new configuration the controller RAPID program and configuration data will be deleted and reset to the factory default settings. Create a backup first if you want to save your RAPID program and data.

9. Now it's time for a coffee break. Enjoy!

NOTE:

 114801 Communication not enabled in the robot firewall!

To enable communication between the Robot Controller and the Vision Controller, socket communication must be allowed:

RobotStudio → **Configuration** → **Communication** → **Firewall Manager** → **RapidSockets** → **Public / Private** → **Yes / Yes** → **Restart Robot Controller**

If you're using the private network (typically on a 192.168.125.xxx network), you must enable the firewall rule for the RAPID socket on the private network. If you're using the **WAN port** for your connection, you **must enable** the firewall rule for the RAPID socket on the **public network**.

The **BPS** checks these settings as part of the state server, since without enabling this communication, the state server cannot operate. The **LS** checks these settings during the initialisation of communication with the Vision Controller.

The Add-In is installed – what's next?

BPS Initial setup before the first test

Before running your first **Bin Picking** application test, you need to configure the following:

- **Tool and Load Data:**
 - Adapt **the mass** and **center of gravity** of the tPho_Tool based on your gripper.
 - Adapt **the mass** and **center of gravity** of the IPho_Load based on your application.
 - The recommended way to do this on the FPU is as follows:
FlexPendant home screen → **Program Data** → **Tooldata / LoadData** → select tPho_Tool / IPho_Load → tap the "..." icon on the right → **View Value**.
 - If you don't have the data, you can use the LoadIdentify service routine:
Flexpendant home screen → Calibrate → hamburger menu → Service routines → LoadIdentify

- **IP Addresses: Network**

Robot IP Address (Server Role) - This IP address is required for the Pho_State_Server, a background task running on the robot controller.

How it works: In this role, the robot acts as a server.

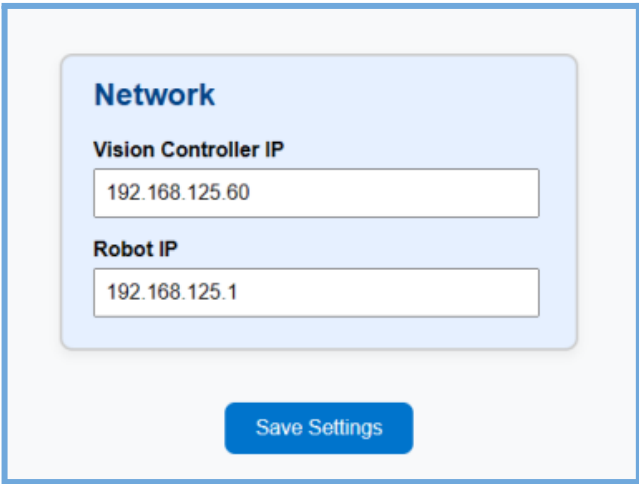
What it does: This server's job is to continuously send the robot's current position data to another application (BPS) for real-time visualisation.

Vision Controller IP Address (Client Role) - This IP address is the target for the robot's primary task, T_ROB1, when it needs to communicate with the vision system.

How it works: In this role, the robot acts as a client.

What it does: The robot initiates a connection to the vision controller (the server) to send a request or receives a result.

- You can use the private network “**192.168.125.XXX**”.
 - You can use the switch, the port on the IO card, or the MGMT port.
 - The IP address of the robot is static “**192.168.125.1**”.
- You can use the public network (**WAN** port)
- The predefined IP address of the Photoneo Visual Computer in the robots module is “**192.168.125.60**”. You can change the IP address if needed using the WebApp or directly via the string variable **sPho_IP_VC**.
- The address of the robot is necessary to adjust in the Rapid program using the WebApp or directly via the string variable **sPho_IP_StateServer**.
- The current IP addresses are displayed on the **Network** page within the Photoneo web application.



- **Robot Positions:**

- In the robot module, modify the **pPho_home_pose**, **pPho_start_bin_picking**, and **pPho_end_bin_picking** as needed.
- Adjust the placement positions in the Rapid to suit your specific application.

- The recommended way to do this on the FPU is as follows:
Jog the robot to the desired position, then navigate to the FlexPendant home screen → **Program Data** → **Robtargets** → select your robtarget → tap the "..." icon on the right → **Update Position**.
- **Gripper Control:**
 - Add gripper control commands (set/reset IO signals) to the routines **Gripper_attach** and **Gripper_detach** in **Pho_CustomerDefinition.mod**.
 - These routines call the **GripLoad** function.
- **Calibration:**
 - Adjust the 9 calibration positions using the **Pho_CalibPositions** routine, then execute the calibration process with the **Pho_Calib** routine - run before the calibration process in BinPicking Studio. Further details can be found in the *Calibration* section.

You can find more information about calibration in the Photoneo documentation (WebApp or Photoneo Vision controller).

LS Initial setup before the first test

Before running your first **Locator** application test, you need to configure the following:

- **Tool and Load Data:**
 - Adapt **the mass** and **center of gravity** of the tPho_Tool based on your gripper.
 - Adapt **the mass** and **center of gravity** of the IPho_Load based on your application.
 - The recommended way to do this on the FPU is as follows:
FlexPendant home screen → **Program Data** → **Tooldata / LoadData** → select tPho_Tool / IPho_Load → tap the "..." icon on the right → **View Value**.
 - If you don't have the data, you can use the LoadIdentify service routine:
Flexpendant home screen → Calibrate → hamburger menu → Service routines → LoadIdentify

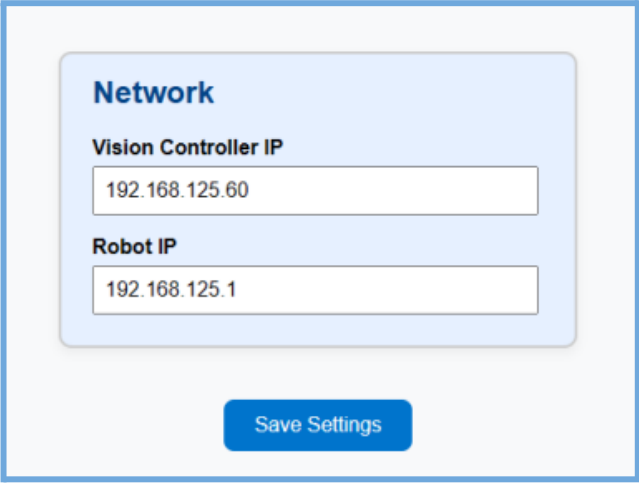
- **IP Addresses: Network**

Vision Controller IP Address (Client Role) - This IP address is the target for the robot's primary task, T_ROB1, when it needs to communicate with the vision system.

How it works: In this role, the robot acts as a client.

What it does: The robot initiates a connection to the vision controller (the server) to send a request or receives a result.

- You can use the private network "[192.168.125.XXX](#)".
 - You can use the switch, the port on the IO card, or the MGMT port.
 - The IP address of the robot is unused ("**192.168.125.1**")
- You can use the public network (**WAN** port)
- The predefined IP address of the Photoneo Visual Computer in the robots module is "**192.168.125.60**". You can change the IP address if needed using the WebApp or directly via the string variable **sPho_IP_VC**.
- The address of the robot is unnecessary to adjust - for LS Studio, communication with the state server is not required, so its IP address can be ignored. Although a default address of "**192 . 168 . 125 . 1**" may be present, this setting is not used by the program.
- The **only IP address critical for operation** is the **Vision Controller IP**.
- The current IP addresses are displayed on the **Network** page within the Photoneo web application.



- **Robot Positions:**

- Adjust the **pPho_home_pose**, **pPho_start_pose**, and **Place positions** to fit your application and add the commands to gripper control.
- The recommended way to do this on the FPU is as follows:
Jog the robot to the desired position, then navigate to the FlexPendant home screen → **Program Data** → **Robtargets** → select your robtarget → tap the "..." icon on the right → **Update Position**.

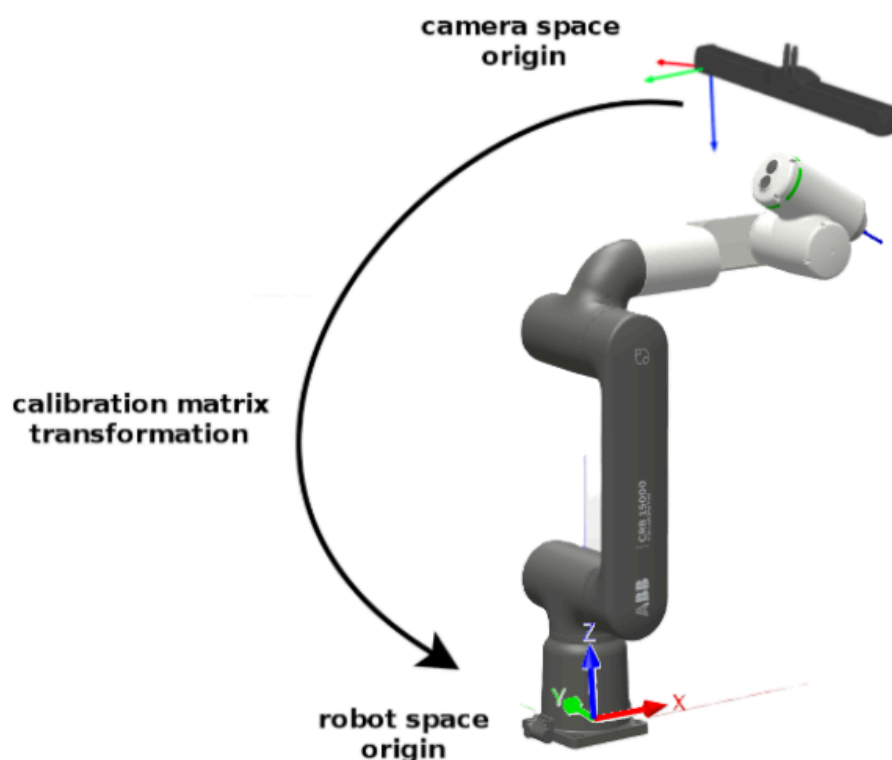
- **Calibration:**
 - Adjust the 9 calibration positions using the **Pho_CalibPositions** routine, then execute the calibration process with the **Pho_Calib** routine - run before the calibration process in BinPicking Studio. Further details can be found in the *Calibration* section.

You can find more information about calibration in the Photoneo documentation (WebApp or Photoneo Vision controller).

Calibration

The point cloud from a 3D sensor has its origin in the camera unit of the device. This means that when objects in the scene are localized, localization returns their position and rotation coordinates in camera space. On the other hand, the robot (usually) has its origin in its own base - robot space - regardless of the position of the camera.

In order for the robot to be able to navigate to a localized object, we need to transform the object's coordinates to robot space first (or other common coordinate systems). This transformation is done by applying a calibration matrix to the coordinates of the origin of the localized object in camera space. The calibration matrix is the result of a successful calibration procedure and it contains the exact rotation and translation between camera space and the target coordinate space.



Types of calibration:

As already mentioned, the target space does not have to be the robot's base frame. The robot can work in a custom user frame. In both of these cases, we perform the robot-camera calibration. In the Studio, we recognize two of its basic types:

- Robot space, external fixed sensor position (Extrinsic calibration)
- Robot space, sensor mounted on the robotic flange (Hand-eye calibration)

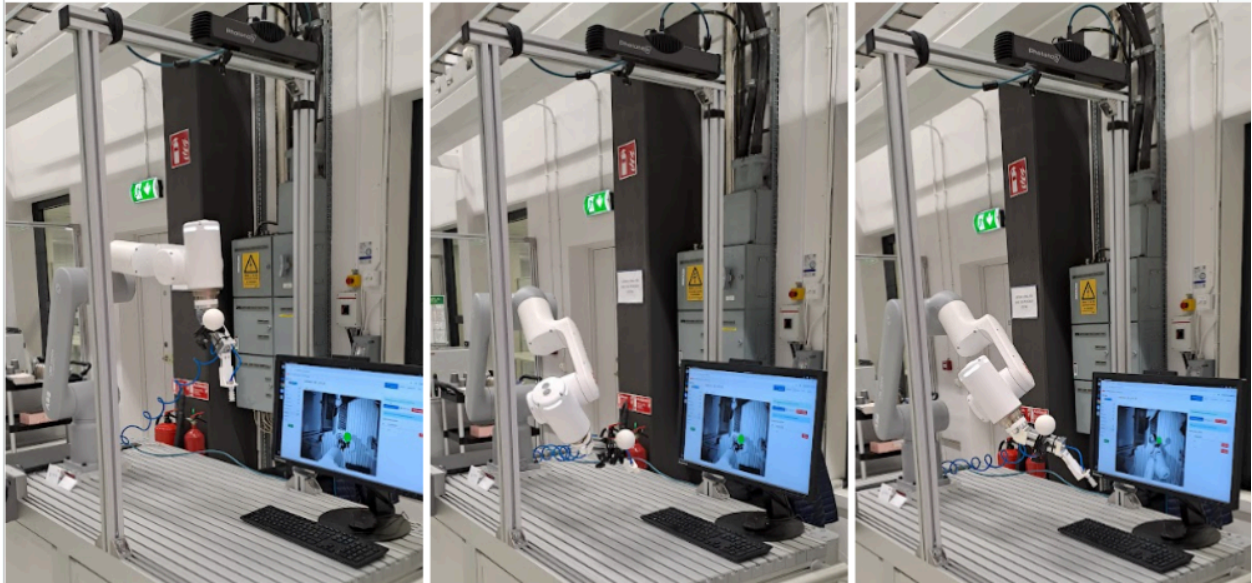
Two other calibration types are not connected to the robot directly.

- Marker space, single sensor position
 - Only available in the locator solution types
→ marker-camera calibration
- Multi-sensor calibration
 - for Static MultiView approach available for the CAD localization engine
→ camera-camera calibration

Robot space, external fixed sensor position (Extrinsic calibration)

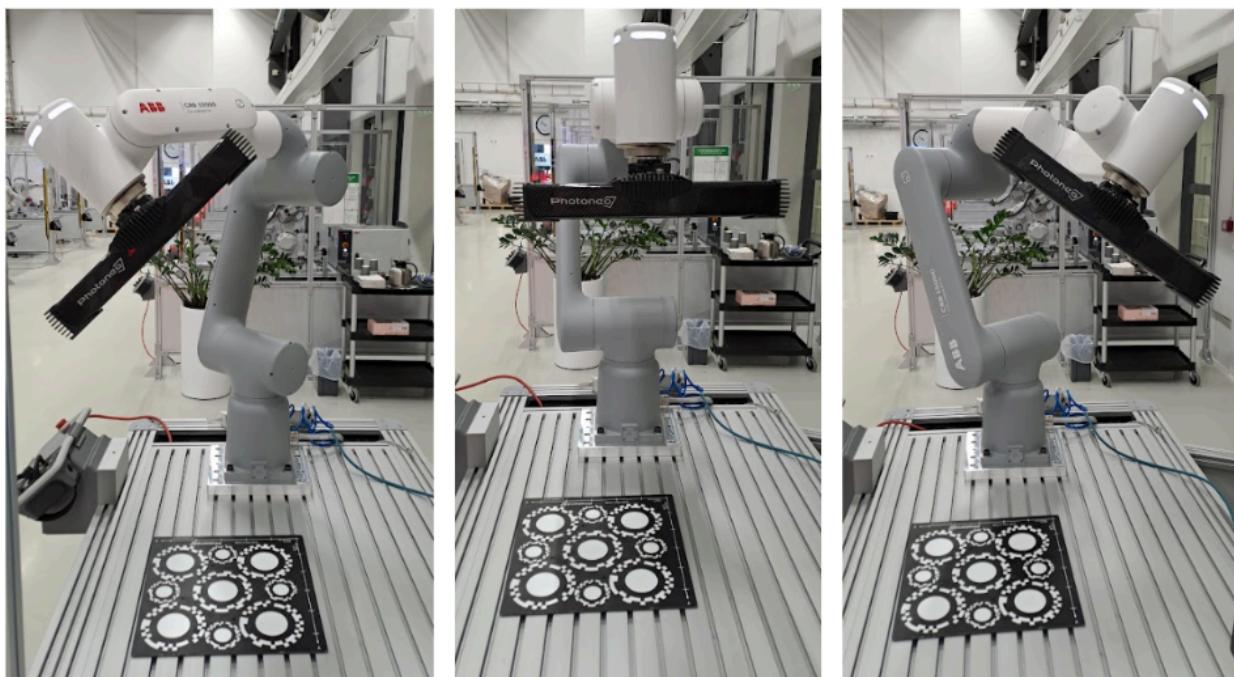
This calibration type is used in setups where the 3D sensor is mounted in a stationary position in the robotic cell, usually above the bin. It is only required that the 3D sensor be stationary relative to the robot's base; the 3D sensor does not need to be stationary relative to the robotic cell itself. Any change in the relative position between the 3D sensor and the robot after the calibration will render the calibration matrix invalid and the whole calibration procedure must be repeated. In other words, the 3D sensor must not be moved in relation to the robot after the system has been calibrated. Since the 3D sensor is fixed, the calibration matrix defines the transformation directly to the robot base coordinate system.

This setup does not require the robot to stop its movement during the scan acquisition (as is the case with the hand-eye approach) and the robot's movement is not limited in any way.



Robot space, sensor mounted on the robotic flange (Hand-eye calibration)

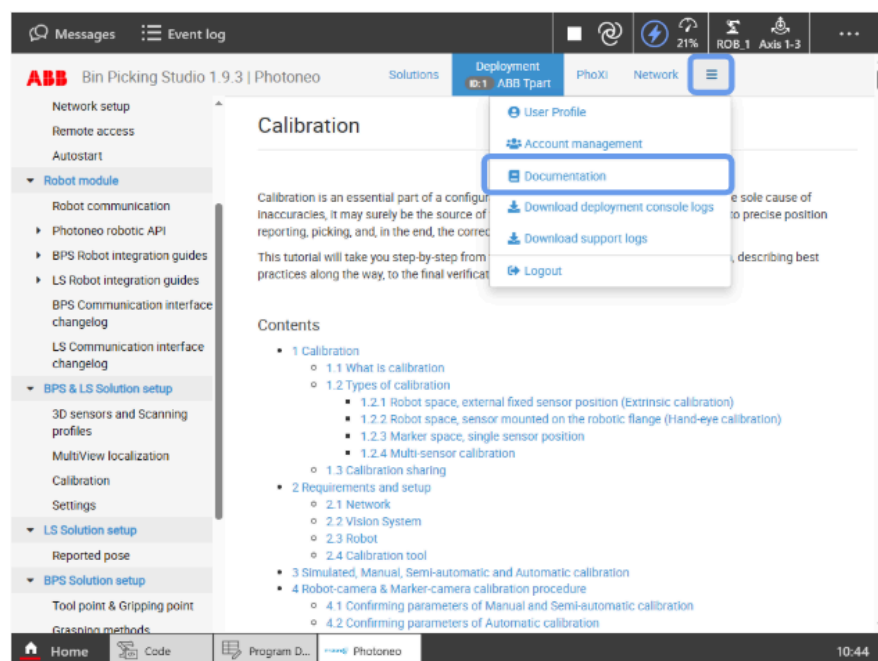
This type of calibration is used in setups where the 3D sensor is mounted directly on the robotic arm. For hand-eye calibration to function correctly, the sensor must be positioned behind the final joint—typically on the robotic flange or gripper. Any modification to the sensor's mounting position after calibration will invalidate the calibration matrix, requiring the entire calibration process to be repeated.



Since the 3D sensor is not fixed, the calibration matrix defines the transformation to the robot's TCP which needs to be known in the moment of object localization. Using the TCP data the pose of the object can be transformed to the robot base coordinate system.

This setup allows for dynamic change of the scanning distance from the bin just by moving the robot. Similarly, the scene can be scanned from various angles. On the other hand, the robot must not move during the scan acquisition, which may affect the cycle time - PhoXi Scanner. If you want to scan while the robot is in motion, you must use the MotionCam.

You can find more information about calibration in the Photoneo documentation (WebApp or Photoneo Vision controller).

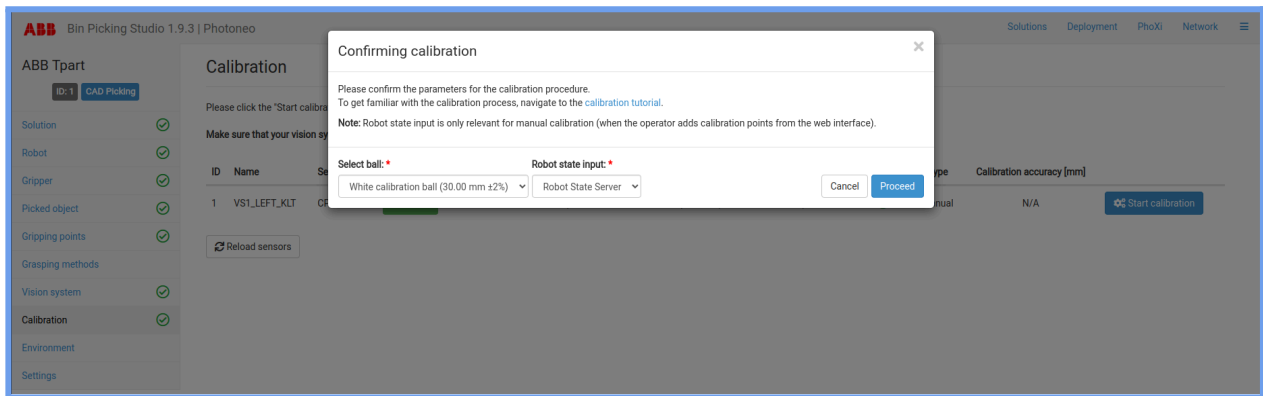


How to Calibrate the robot and Photoneo device

Open the **Pho_CalibPosition** procedure, which contains 9 robtarget positions. Attach the calibration sphere to the robot tool and ensure that the sphere remains fixed and does not move during the calibration process.

For future calibrations, it is sufficient to achieve an approximate alignment of the calibration sphere relative to the tool.

To achieve the highest possible calibration accuracy, you need to teach all 9 positions. Each position should be located in a different part of the scanner's scanning range. Additionally, the joint configuration should vary significantly between each position, and the calibration sphere must always remain visible to the camera. Please ensure that all transitions between the taught positions are collision-free. Next, start the calibration process on the Photoneo Controller, as shown in the image below.



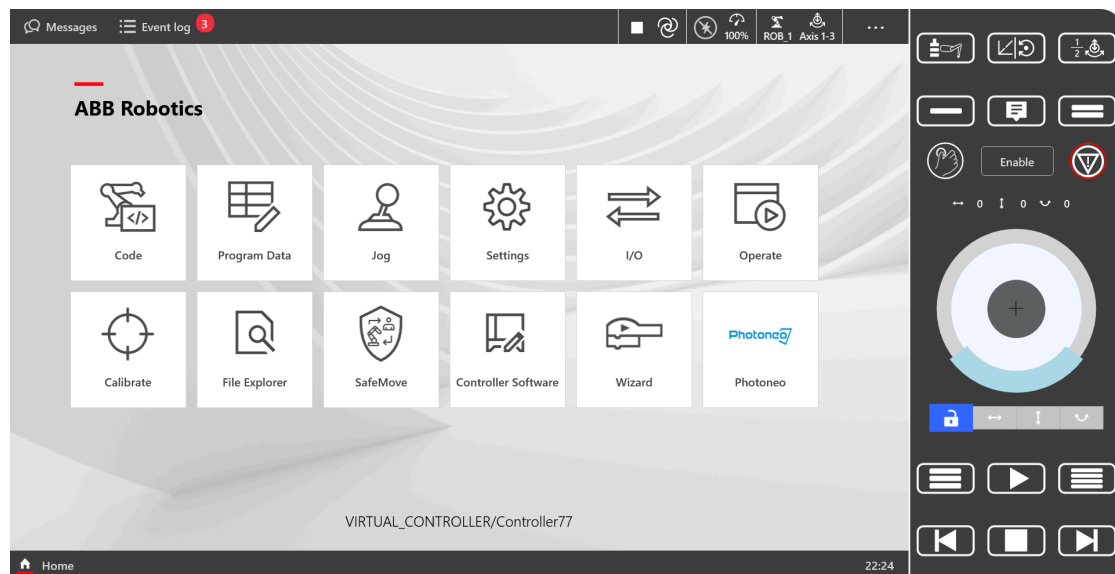
Press the Proceed button, and after that, run the **Pho_Calib** routine on the robot. If the calibration sphere is successfully detected in all 9 positions without any errors, the calibration result will be displayed. You can then save the result, and the calibration process is complete.

WebApp

Using the Photoneo WebApp, you can control the Binpicking Studio / Locator Studio interface directly from the FlexPendant screen. At the same time, the full documentation is readily available at your fingertips.

The WebApp consists of three sections:

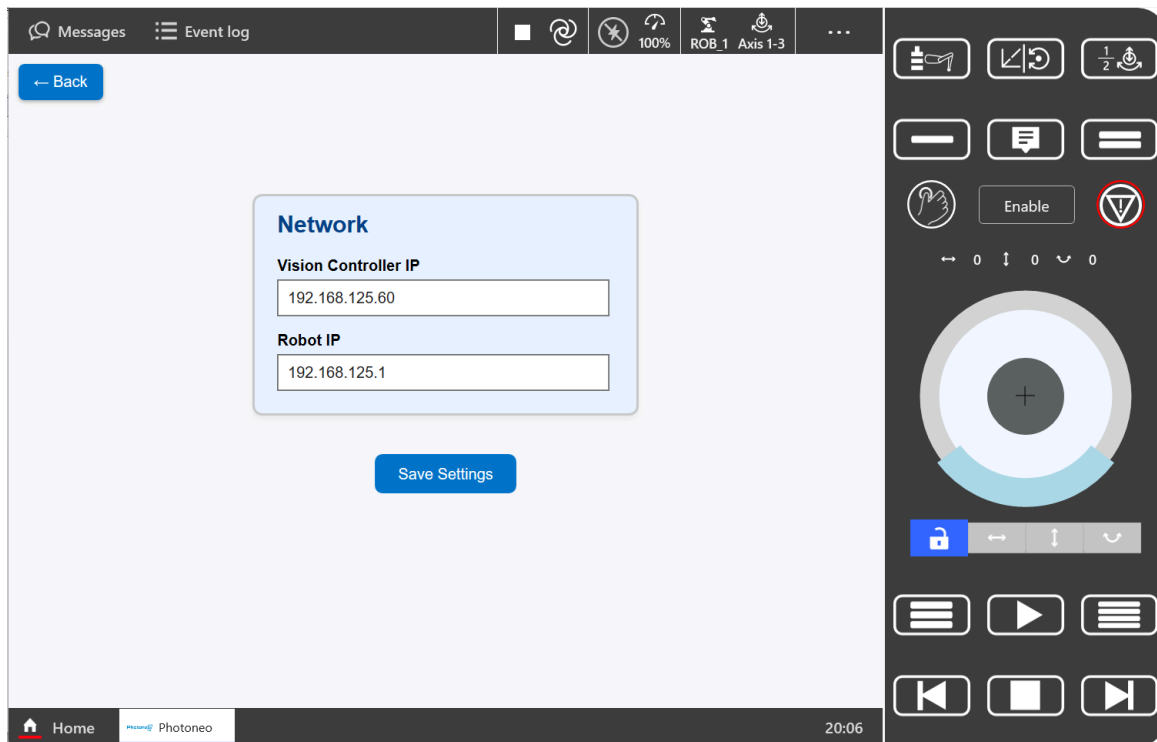
- **Manual** → Access to documentation and guides
- **Network** → Configuration of IP addresses for the Photoneo Vision Controller and the Robot Controller
- **Redirect** → Displays the web application running on the Vision Controller (can only be used if the device is connected to the robot's internal network)



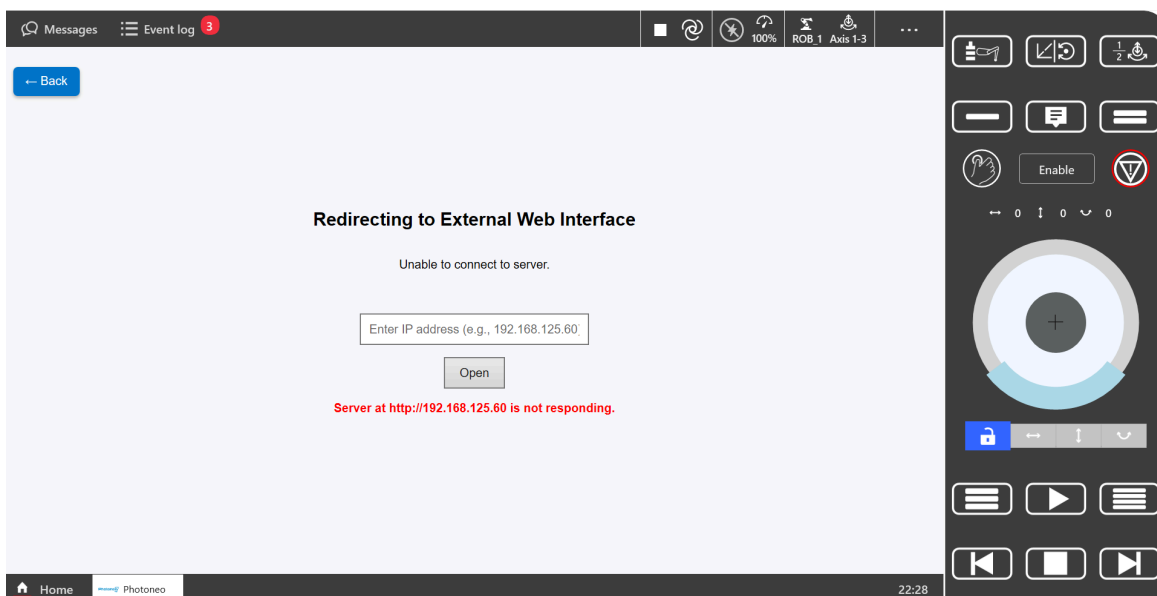
In the NETWORK section, you can enter the IP addresses of the Vision Controller and the Robot Controller. These addresses will be saved into the variables **sPho_IP_VC** and **sPho_IP_StateServer**, and will subsequently be used in the robot program.

If the IP addresses do not match the required format or if the data cannot be saved to the variables, the SAVE button will turn red for 2 seconds. Otherwise, it will turn green.

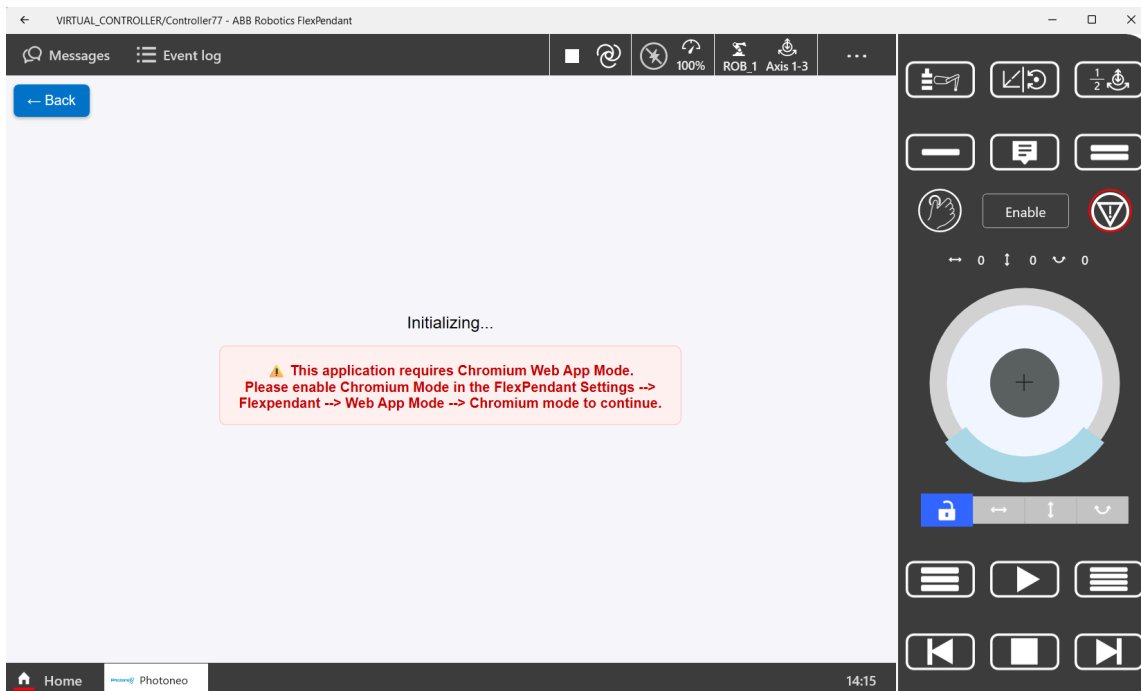
If the user uses the Private Network, they will gain the ability to use a redirect to access the web application running on the Vision Controller.



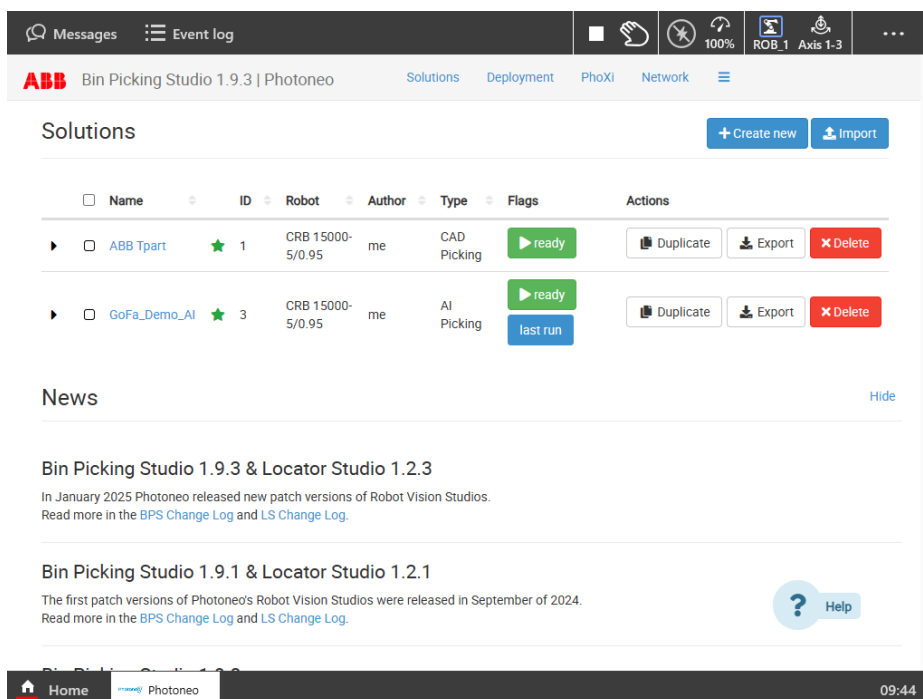
If the automatic redirect does not complete successfully, it is likely that the IP address entered in the NETWORK section was incorrect. A manual input field will then appear, allowing you to enter the address manually. This address will subsequently be saved to the variable sPho_IP_VC.



If this message appears on the FlexPendant, you need to enable Chromium Mode according to the displayed instructions.



If the hardware is properly connected and the IP addresses are correctly configured, the web application will be displayed on the FlexPendant, allowing you to create or edit Solutions.



Wizard

Wizard Easy Programming is an intuitive graphical tool designed to simplify robot programming for both beginners and experienced users. It enables effortless, fast, and efficient programming of collaborative and industrial robots across various applications. Users can easily create programs by dragging and dropping predefined blocks or selecting them with a button press. Once the setup is complete, simply press play to execute the program—making automation more accessible than ever.

Now you can quickly create applications for bin picking or part localization with ease. For both products, we have pre-designed blocks that simply need to be connected in the correct sequence, parameters filled in, and the application is ready to run.

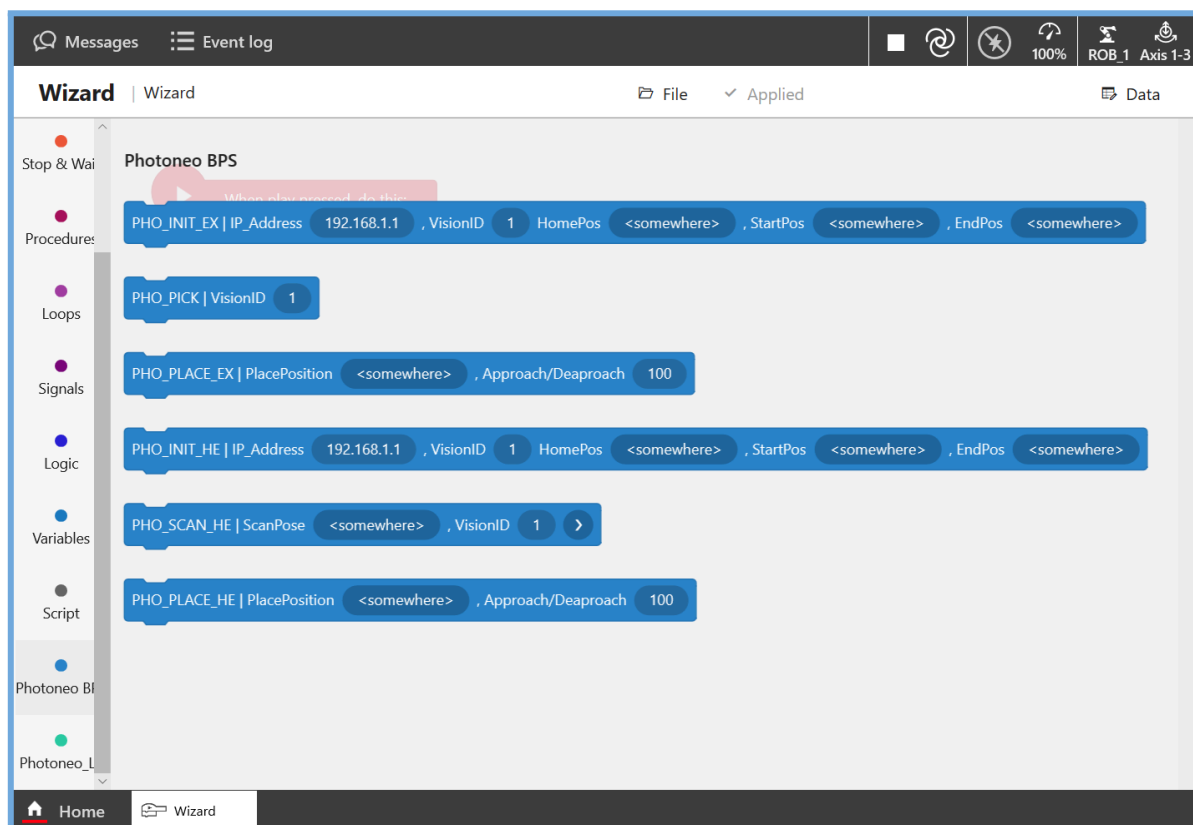
The source code is located in the **PhotoneoWizard_BPS.mod** or **PhotoneoWizard_LS.mod** module. It is recommended to review the code and modify it to suit your needs.

Bin Picking blocs

EX means that these blocs are pre-designed for Extrinsic applications.

HE means that these blocs are pre-designed for HandEye application.

The block without a suffix is universal and can be used in both cases.



PHO_INIT_EX

- **IP_Address** → IP_Address of the Visual controller, same as the settings in the vision controller
- **VisionID** → The ID of the vision system in the currently deployed solution in Studio.
- **HomePos** → The robot's home position, located away from the scanning area. The robot moves to this position during initialization, if the system does not detect any objects, or in case of an error.
- **StartPos** → The starting position from which the system begins executing the planned path.
- **EndPos** → The final position where the robot completes path planning after picking up the part.



PHO_PICK

- **VisionID** → The ID of the vision system in the currently deployed solution in Studio. This block performs scanning and localization based on the VisionID in the active solution within Studio.

PHO_PLACE_EX

- **PlacePosition** → The position where the robot places the picked-up part.
- **Approach/Deapproach** → The distance along the Z-axis, in millimeters, within the robot's base coordinate system that the robot reaches before or after the **Place Position**.

PHO_INIT_HE

- **IP_Address** → IP_Address of the Visual controller, same as the settings in the vision controller
- **VisionID** → The ID of the vision system in the currently deployed solution in Studio.
- **HomePos** → The robot's home position, located away from the scanning area. The robot moves to this position during initialization, if the system does not detect any objects, or in case of an error.
- **StartPos** → The starting position from which the system begins executing the planned path.
- **EndPos** → The final position where the robot completes path planning after picking up the part.

PHO_SCAN_HE

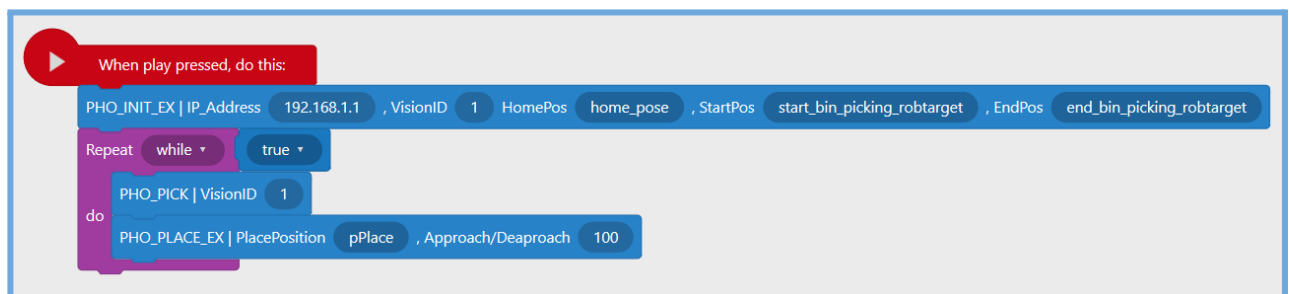
- **ScanPose** → The robot position where the user wants to perform scanning.
- **VisionID** → The ID of the vision system in the currently deployed solution in Studio.
- **WaitTime** → An optional argument that introduces a delay between the robot stopping at the target and scanning. This helps filter out shaking that may occur if the robot's end effector is not rigid enough.

PHO_PLACE_HE

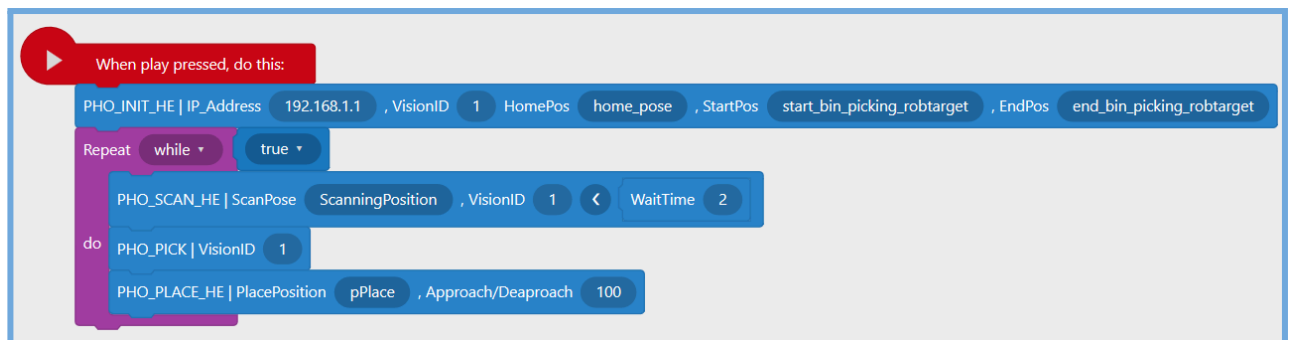
- **PlacePosition** → The position where the robot places the picked-up part.
- **Approach/Deapproach** → The distance along the Z-axis, in millimeters, within the robot's base coordinate system that the robot reaches before or after the **Place Position**.

The difference between **PHO_PLACE_HE** / **PHO_PLACE_EX** and **PHO_INIT_EX** / **PHO_INIT_HE** is that in an **Extrinsic** application, scanning can be performed because the robot is likely outside the scanning volume, reducing cycle time. In a **Hand-Eye** application, however, the robot must first reach the scanning position before scanning.)

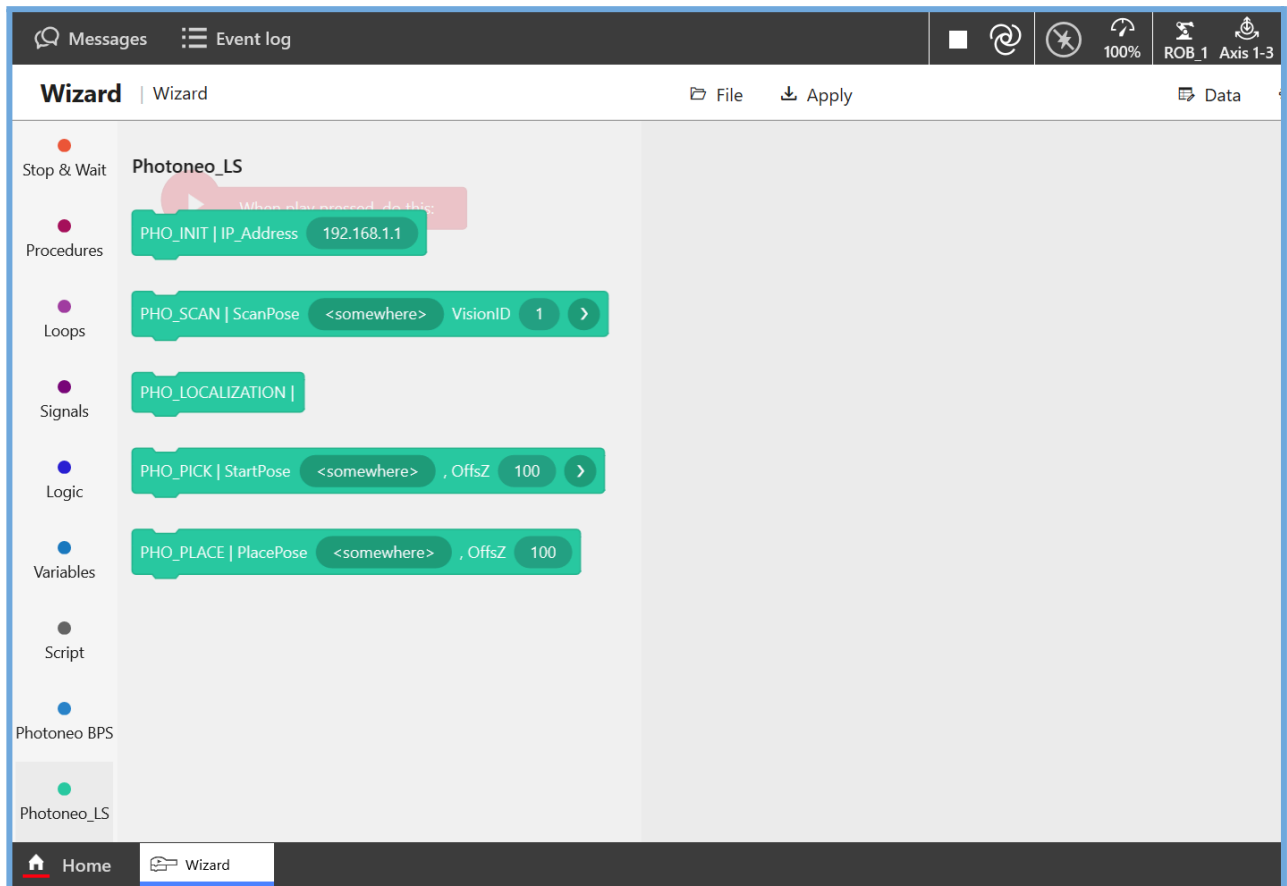
Wizard Bin picking Extrinsic example:



Wizard Bin picking Hand Eye example:



Locator blocs



PHO_INIT

- **IP_Address** → IP_Address of the Visual controller, same as the settings in the vision controller

PHO_SCAN

- **ScanPose** → The robot position where the user wants to perform scanning.
- **VisionID** → The ID of the vision system in the currently deployed solution in Studio.
- **WaitTime** → An optional argument that introduces a delay between the robot stopping at the target and scanning. This helps filter out shaking that may occur if the robot's end effector is not rigid enough.

PHO_LOCALIZATION

- It contains a sequence of requests for starting localization in **Locator Studio**.

PHO_PICK

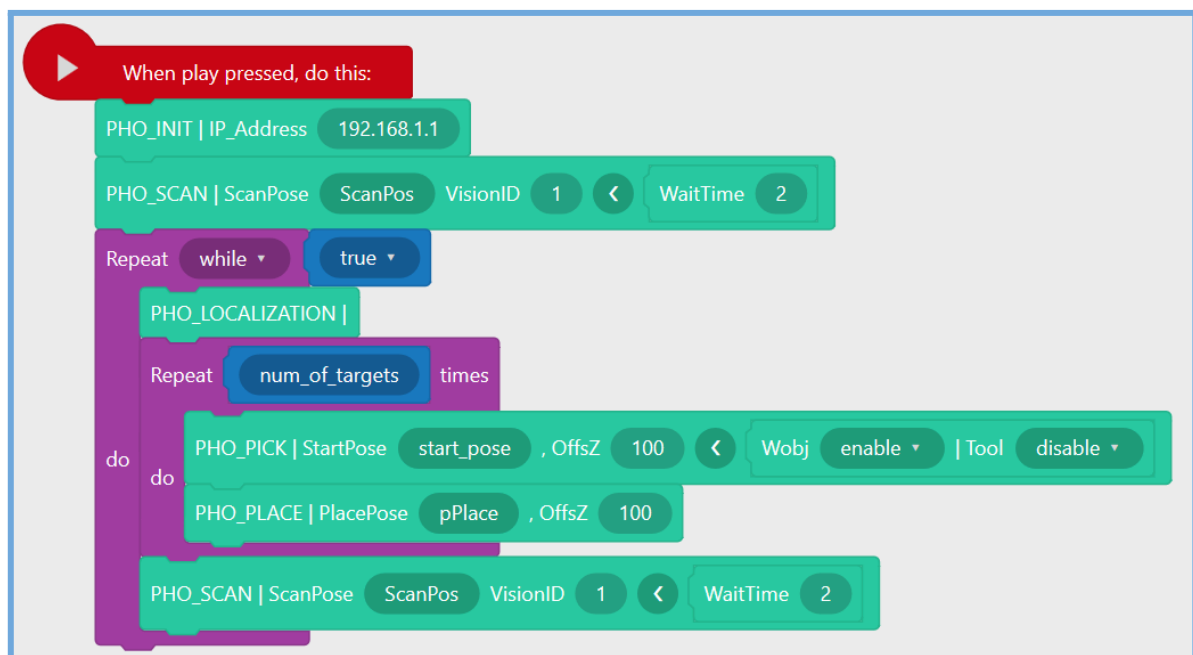
- **StartPose** → The position from which the robot moves to the position reported by Locator Studio. It is typically located above the area where parts are being localized.
- **OffsZ** → The distance along the Z-axis, in millimeters, that the robot reaches before and after the Pick Position (Approach/Deapproach)
- **Wobj | Tool** → An optional argument that defines whether the offset in the Z-axis will be applied in the robot's base coordinate system or the tool coordinate system. For the tool, the value may be positive or negative depending on the orientation of the TCP. If the user does not enable either option, the robot will perform the movement in the base coordinate system by default.

PHO_PLACE

- **PlacePosition** → The position where the robot places the picked-up part.
- **OffsZ** → The distance along the Z-axis, in millimeters, within the robot's base coordinate system that the robot reaches before or after the **Place Position**.

The **PhotoneoWizard_LS** module contains the routines **GRIPPER_ATTACH** and **GRIPPER_DETACH**, where commands for controlling the end effector need to be added.

Wizard Locator example:





Focused on 3D

NOW PART OF ZEBRA TECHNOLOGIES