

ROBOTICS

Application manual

PickMaster™ Twin External Sensor UDP



Trace back information:
Workspace Main version a551
Checked in 2023-11-22
Skribenta version 5.5.019

Application manual
PickMaster Twin External Sensor UDP

Document ID: 3HAC089202-001

Revision: A

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damage to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

© Copyright 2023 ABB. All rights reserved.
Specifications subject to change without notice.

Table of contents

Overview of the manual	7
1 Introduction	9
2 Safety	11
3 Configuration	13
3.1 System requirements	13
3.2 Configuring the PickMaster Twin External Sensor UDP	14
4 Communication	19
4.1 Acquisition and communication flowchart	19
4.2 XML protocol	20
5 Calibration	23
6 Developer	25

This page is intentionally left blank

Overview of the manual

About this manual

This manual contains information and instructions for installing and using PickMaster XML External Sensor.

Usage

This manual describes PickMaster XML External Sensor and includes step-by-step instructions on performing the various tasks that the software offers.

Who should read this manual?

This manual is intended for:

- System integrators
- Machine builders
- Installation personnel
- Programmers
- Operators
- ABB sales and product support

Prerequisites

A reader on a beginner level should have:

- Some basic experience with RobotStudio
- Some basic knowledge of robot picking applications and also vision based guidance
- Good skills in PickMaster and the ABB robot controller

Revisions

Revision	Description
A	First edition

References



Tip

All documents can be found via myABB Business Portal, www.abb.com/myABB.

This page is intentionally left blank

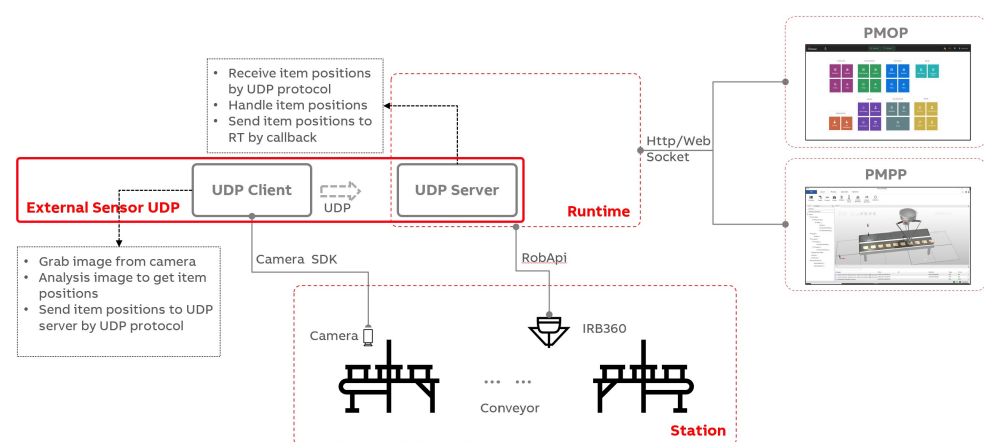
1 Introduction

PickMaster™ Twin External Sensor UDP

The External Sensor UDP is an extension module, which enables object detection by an external sensor to be fed into the PickMaster application.

The External Sensor UDP is an implemented Python script based on the External Sensor (ES) function, which including a ready-to-use UDP server and sensor configuration with GUI, and it is intended to alleviate the developing workload of ES. In addition, External Sensor UDP is compatible with the PickMaster 3 ES.

Topography



xx2300001310

Terms and concepts

Following are some terms with specific meaning when used in this manual.

Terms list

Term	Definition
Work area	A work area is a dedicated area where a robot picks or places objects.
Item source	RAPID data type that represents a work area. Sometimes referred to as a queue.
Conveyor work area	A work area placed on a conveyor. Picking or placing are performed using robot movements coordinated with the conveyor, that is, conveyor tracking.
Indexed work area	A work area placed anywhere near the robot. Picking or placing are performed using uncoordinated robot movements, that is, without conveyor tracking.
External Sensor (ES)	A generic object in PickMaster for any kind of source to generate position data. Each type of external sensor requires custom implementation.
Item	An item is an object that can be picked and placed.
Container	A container is an object that can be filled with items.
Container Pattern	A container pattern is a predefined pattern of items in a container.

Continues on next page

1 Introduction

Continued

Term	Definition
Load balancing	Load balancing is a distribution type. Load balancing is used to share the work load between the robots. The detected objects are distributed to different work areas.
Scene	A scene represents all objects that are detected at a single recording of a sensor, that is, all the objects visible in one captured image. In production, a sensor generates a sequence of scenes
UDP (User Datagram Protocol)	A long standing protocol used together with IP for sending data when transmission speed and efficiency matter more than security and reliability.
Position generator index	A number which defined by user. This number is the key for different models or products detected by the External Sensor system and is transferred by the <code>PosGen</code> tag in the XML protocol.
GUID (Globally unique identifier)	A serial of unique numbers used to identify resources.
PickMaster PowerPac	The market name of PickMaster PC engineering software that is used for simulating and commissioning picking lines with virtual and real Runtime.
PickMaster Operator	The market name of PickMaster production operator interface software that is used for running PickMaster applications in production. PickMaster Operator can read and write to solutions generated by PickMaster Twin External Sensor UDP. It has access to real Runtime.
PickMaster Real Runtime	The core engine that orchestrates all the calculation of pick and place operation in real product. Runtime communicates with cameras and the robots. It's also called as Runtime.
Solution	Format for storing a PickMaster Twin configuration result.

2 Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement.

Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the *Operating manual - General safety information*.

This page is intentionally left blank

3 Configuration

3.1 System requirements

Hardware requirements

Following are the hardware requirements:

- PC requirement according to PickMaster specification
 - Calibrated external sensor system, including cables for power and communications.
-

Software requirements

Following are the software requirements:

- Operating system according to PickMaster Twin Specification
- PickMaster Twin 2.3 (or newer)
- Required software to configure and/or manage the external sensor equipment.


3 Configuration

3.2 Configuring the PickMaster Twin External Sensor UDP

3.2 Configuring the PickMaster Twin External Sensor UDP

Predefined script file

There are several predefined files should be placed into the destination folder for External Sensor UDP:

- *ExternalSensorIcon.ico*
 - *ExternalSensorInterface.py*
 - *ExternalSensorUDP.py*
 - *SensorFunctions.py*
-  *irtualPosGeneratorPMTW.exe*

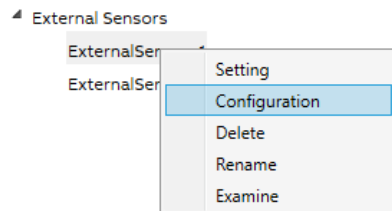
Tip

The predefined script file(s) should be put into *C:\Users\xxxx\Documents\PickMaster\PMScripts* folder before use any script function.

Procedure

Use the following procedure for PickMaster Twin External Sensor UDP configuration:

- 1 Create a new solution including controllers, robots, work areas, external sensors and so on.
- 2 Switch to real Runtime.
- 3 Right-click the external sensor in the tree view **Layout** and select **Configuration**.

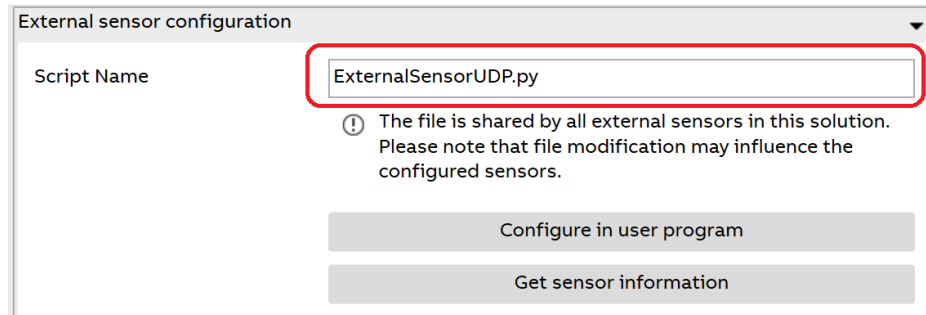


xx2300000899

The **External Sensor Configuration** dialog is opened.

Continues on next page

- 4 Input the name of the predefined main file *ExternalSensorUDP.py* in **Script Name**.



External sensor configuration

Script Name

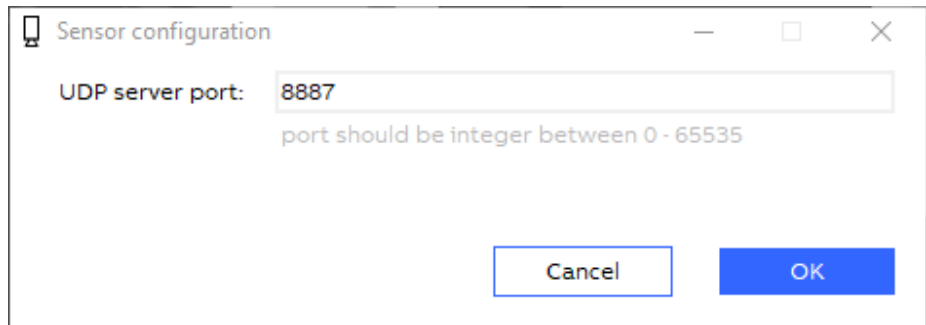
ⓘ The file is shared by all external sensors in this solution. Please note that file modification may influence the configured sensors.

Configure in user program

Get sensor information

xx2300001778

- 5 Click **Configure in user program** to configure the external sensor. The **Sensor Configuration** window is opened.
- 6 Input the correct port in **UDP server port**.



Sensor configuration

UDP server port:

port should be integer between 0 - 65535

Cancel OK

xx2300001332

- 7 Click **OK** to save the update in **Sensor Configuration**.
- 8 Click **OK** to save the update in **External Sensor Configuration**.

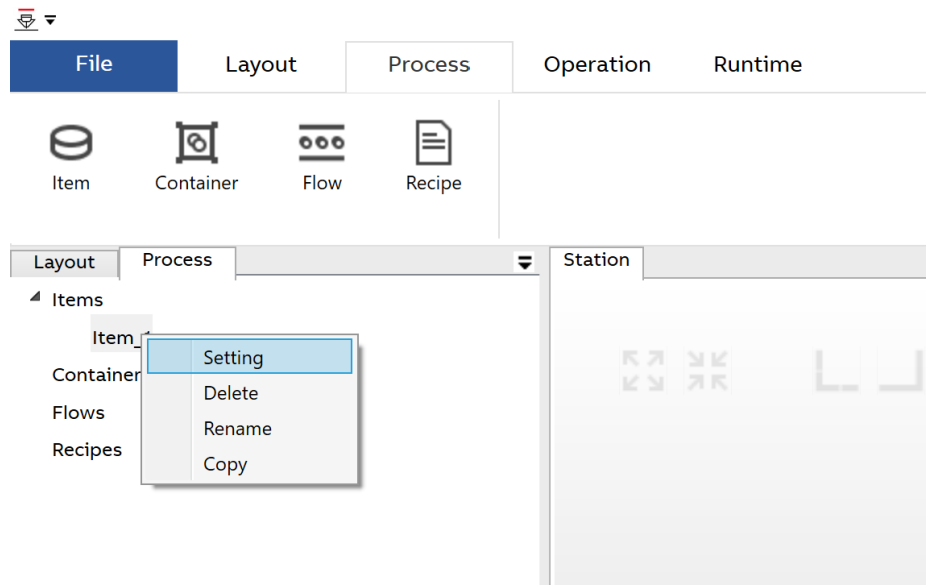
Continues on next page

3 Configuration

3.2 Configuring the PickMaster Twin External Sensor UDP

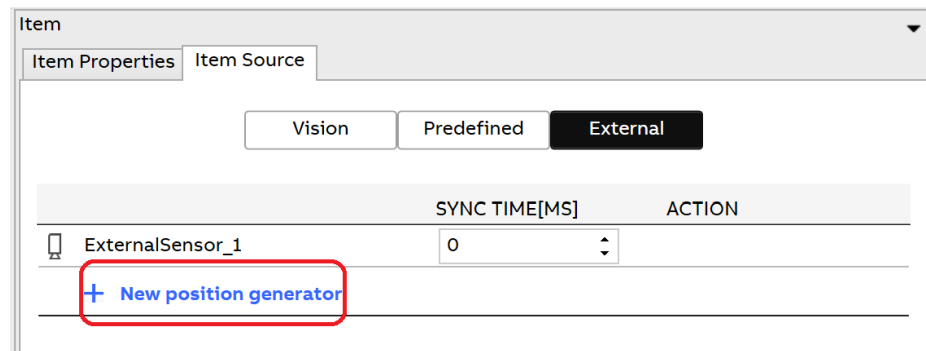
Continued

- 9 Right-click the desired item/container in the tree view **Process** and select **Setting**.



xx2300001779

- 10 Switch to **Item Source/Container Source** page.
- 11 Click **New Position Generator** to create the position generator for desired external sensor under the **External** tab.

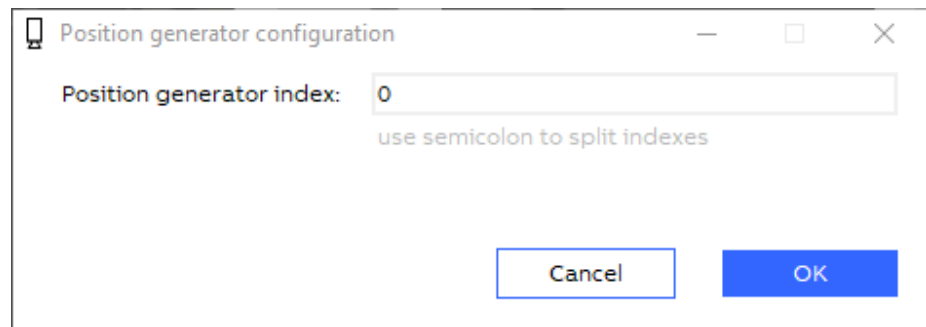


Tip

All created external sensors in this solution will be listed in **External** tab.

Continues on next page

The **Position Generator Configuration** window is opened.



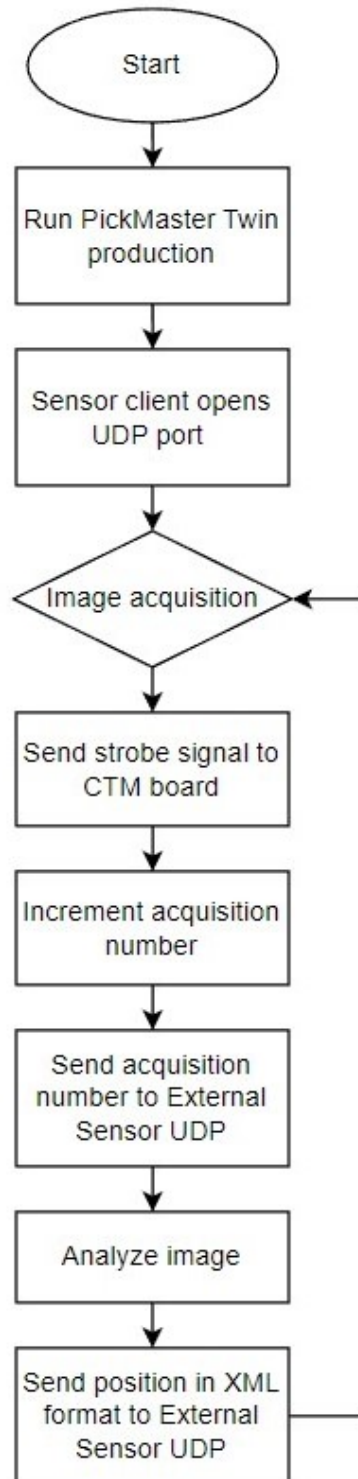
xx2300001333

- 12 Configure the position generator according to the user defined in external sensor script file.
- 13 Click **Ok** to save the position generator index information.
- 14 Click **Save** to save the configuration data to PMPP for the position generator.
- 15 Click **OK** to finish the configuration.

This page is intentionally left blank

4 Communication

4.1 Acquisition and communication flowchart



xx2300001311

4 Communication

4.2 XML protocol

4.2 XML protocol

The PickMaster does not send any request message for new data to the external sensor system. The sensor system sends the data as soon as it is available.

All data transfer is made over socket communication and the messages are sent in XML (text) format. The external sensor system sends all message data to the External Sensor UDP.

The data sent for each image must be sent in two messages.

The acquisition number has to be sent as soon as the image has been acquired, Contacting:

```
<Image Acquisition Number>
```

When the image has been analyzed, one or more position data messages will be sent, each containing:

```
<Image Acquisition Number>  
<Data for item #1>  
<Data for item #2>  
<Data for item #3>
```

and so on.

Refer next section for the specific data formats.

Acquisition number message

Immediately after a new image acquisition is made, the acquisition number is incremented and the number is sent to the PickMaster as soon as possible. This number is a unique identifier and can be thought of as a 'serial number' for the image data. This is made with a XML formatted message, like:

```
<?xml version="1.0" encoding="utf-8"?>  
<NewAcq AcqNo="237" />
```

Variable attribute is:

- **AcqNo:** The acquisition number that is incremental for every trigger.

Position Message

The position information message is sent to PickMaster after the image is analyzed. This message consists of the acquisition number and a number of position elements, each one consisting of one position like:

```
<?xml version="1.0" encoding="utf-8"?>  
<Positions AcqNo="237" PackNo="0" FinalPackNo="0">  
<Position Valid="1" PosGen="0" Accept="2" Score="0.0" Tag="0"  
  X="0.0" Y="0.0" Z="0.0" RX="0.0" RY="0.0" RZ="0.0" Vall="0.0"  
  Val2="0.0" Val3="0.0" Val4="0.0" Val5="0.0" />  
<Position Valid="1" PosGen="0" Accept="2" Score="0.0" Tag="0"  
  X="0.0" Y="0.0" Z="0.0" RX="0.0" RY="0.0" RZ="0.0" Vall="0.0"  
  Val2="0.0" Val3="0.0" Val4="0.0" Val5="0.0" />  
</Positions>
```

Variable attributes are:

- **AcqNo:** The acquisition number that corresponds to every trig, generally the last sent acquisition number sent in the acquisition message. Required.

Continues on next page

- **PackNo:** Sequence number of the current packet of all the Positions packets related to this image (should start at 0).
- **FinalPackNo:** Sequence number of the last packet of all the Positions packets related to this image (when this is equal to PackNo you know that this is the final packet).
- **Valid:** Flag to indicate if this position is a valid one to take into consideration. 1=TRUE, 0=FALSE. Required.
- **PosGen:** The position generator number. This number corresponds to the position generator index seen in the position source dialog for each external position generator. Required.
- **Accept:** Flags the level of inspection to the PickMaster, 2=Accepted, 1=Rejected, 0=Discarded. Required.
- **Score:** Transfer the score of the position to PickMaster. 100=Best score, 0=Worst score. Required.
- **Tag:** A value that can be accessed in the ltmTgt variable in rapid for each item position. The GetltmTgt instruction can be used to request a new item with a specific tag value. Value is expressed in mm and can be decimal between -8388607 and +8388608 optional.
- **X:** The X-values of the position relative to the calibrated origin. Value is expressed in mm and can be between -8388647 and +8388608, optional.
- **Y:** The Y-value of the position relative to the calibrated origin. Value is expressed in mm and can be between -8388647 and +8388608, optional.
- **Z:** The Z-value of the position relative to the calibrated origin. Value is expressed in mm and can be between -8388647 and +8388608, optional.
- **'RX':** 0.0, # RX refers to the rotation angle value of the item in X direction, unit is degree
- **'RY':** 0.0, # RY refers to the rotation angle value of the item in Y direction, unit is degree
- **'RZ':** 0.0, # RZ refers to the rotation angle value of the item in Z direction, unit is degree
- **'Val1':** 0.0, # optional value, used in rapid
- **'Val2':** 0.0, # optional value, used in rapid
- **'Val3':** 0.0, # optional value, used in rapid
- **'Val4':** 0.0, # optional value, used in rapid
- **'Val5':** 0.0, # optional value, used in rapid

This page is intentionally left blank

5 Calibration

Overview

When using some External Sensor mean with a PickMaster system, the sensor needs to be jointly calibrated with the robots and workareas.

The work area calibration is a base frame calibration for conveyor work areas and a work object definition for indexed work areas. The key concept is to define a coordinate system origin that is the same for the external sensor system and a robot base frame or work object.

The calibration method is depending on which type of External Sensor system, but every type of system needs to be calibrated with an origin to which every reported position is related to.

The conveyor baseframe calibration and indexed workarea calibration if performed , should be done according to regular PickMaster method.

Calibration procedure

Use the following procedure to calibrate the External sensor system in cooperation with PickMaster.

- 1 **Counts Per Meter Calibration:** Perform the counts per meter calibration of the encoder described in the PickMaster PowerPac Application Manual.
- 2 **Initiation of Camera to Robot Calibration:** Initiate the conveyor baseframe calibration by executing the PrepareCalibration program for all the robots being supplied with positions by the External sensor system, according to the description in the PickMaster PowerPac Application Manual.
- 3 **External sensor system Calibration:** Follow the calibration procedure for the external sensor system in use.
- 4 **Conveyor BaseFrame Calibration:** Finalize the calibration by performing the BaseFrame calibration for all consecutive robots by pointing on the center reference point of the calibration plate by all the robots according to the PickMaster PowerPac Application Manual.

This page is intentionally left blank

6 Developer

Overview

The External Sensor UDP is an implemented Python script based on the External Sensor (ES) function. To implement the function of the sensors, `SensorFunction.py` is added to the External Sensor UDP script template, which uses `SensorFunction` class. This class controls the critical functions of a sensor, e.g., the implementation of UDP server port, the implementation of position generator index, and the sensor start mechanism etc.

For more detailed information of the external sensor framework, please see *PickMaster PowerPac Application manual - Configuring external sensor*.

An UDP server is integrated in method `startSensor` in `SensorFunction.py`. If the user wants to use other data transmission protocols (such as TCP), overwrite the corresponding part in method `startSensor`.

Implementing UDP server port

The implementing procedure is performed with user interface by calling the method `showSensorPortConfigDialog`. The entered UDP server port is serialized into a string and is checked if the port number is valid when the OK button is clicked. The valid string will be saved in the Python dictionary `sensorConfigurationDict[sensorId]`.

```
def showSensorPortConfigDialog(self, inputTitle, configInfo,
    callBackFunc)
```

Argument	Description	Note
<code>self</code>	Python syntax Refer to the class	
<code>inputTitle</code>	Window name. It is Sensor configuration in the predefined design.	
<code>configInfo</code>	Last saved UDP server port.	
<code>callBack-Func</code>	To generate system log.	Example: <code>logCallback.ShowPythonLog("xxxx")</code>

Code:

```
def showSensorPortConfigDialog(self, inputTitle: str, configInfo:
    str, callBackFunc):
    ...
def inputChecker(input: str):
    pattern='^\d+$'
    return re.match(pattern, input) and int(input) >= 0 and int(input)
        <= 65535
def closeWindow():
    nonlocal serverPort, isPortValid, callBackFunc
    if(inputChecker(txtPort.get())):
        serverPort = txtPort.get()
        isPortValid = True
        sensorConfigWindow.destroy()
    else:
```

Continues on next page

```
lbWrongInput.config(fg='red')
...
return (isPortValid, serverPort)
```

Implementing position generator index

The implementing process is performed with user interface by calling the method `showPositionGeneratorConfigDialog`. The entered position generator index is serialized into a string and is checked if the index is valid. The valid string will be saved in the Python dictionary `posGenConfigurationDict[posGenId]`.

```
def showPositionGeneratorConfigDialog(self, inputTitle,
configInfo, callBackFunc)
```

Argument	Description	Note
self	Python syntax Refer to the class	
inputTitle	Window name. It is Sensor configuration in the predefined design.	
configInfo	Last saved position generator index.	
callBack-Func	To generate system log.	Example: <code>logCallback.ShowPythonLog("xxxx")</code>

Code:

```
def showPositionGeneratorConfigDialog(self, inputTitle: str,
configInfo: str, callBackFunc):
...
def inputChecker(input:str):
nonlocal positionGeneratorIndexNum, lbWrongInput
pattern='^(\d+;)+(?=\d+$)|^\d+$'
if(re.match(pattern, input)):
positionGeneratorIndexList = re.findall('\d+', input)
positionGeneratorIndexSet = set(positionGeneratorIndexList)
for index in positionGeneratorIndexSet:
if positionGeneratorIndexList.count(index) > 1:
lbWrongInput.config(text="Duplicated index detected, please check
your input.")
return False
for index in positionGeneratorIndexList:
if(int(index) < 0 or int(index) > 1000):
lbWrongInput.config(text="The entered index is invalid, please
check your input.")
return False
else:
lbWrongInput.config(text="The entered index is invalid, please
check your input.")
return False
return True
def closeWindow():
nonlocal positionGeneratorIndex, isIndexValid
if(inputChecker(positionGeneratorIndexNum.get())):
positionGeneratorIndex = positionGeneratorIndexNum.get()
```

Continues on next page

```

isIndexValid = True
positionGeneratorConfigWindow.destroy()
else:
lbWrongInput.config(fg='red')
...
return (isIndexValid, positionGeneratorIndex)

```

Implementing start function: Sensor operating mechanism

The implementing process is performed by calling the method `startSensor`. When the production is starting, the `startSensor` method is called, and a UDP server is initialized with the user configuration. The UDP server will trigger an endless loop to listen message over specific port. When UDP server received any message, the message will be deserialized and stored in a dictionary. The valid data will finally be sent to the Runtime.

```

def startSensor(self, callback, sensorId, positionGeneratorId,
sensorConfigInfo, positionGeneratorCon-figInfo, logCallback)

```

Argument	Description	Note
<code>self</code>	Python syntax Refer to the class	
<code>callBack-Func</code>	Which contains <code>GetStrobeTime()</code> and <code>NewPosition(pos)</code>	
<code>sensorId</code>	GUID of the sensor to be configured	
<code>positionGeneratorId</code>	GUID of the position generator to be configured	
<code>sensorConfigInfo</code>	UDP server port configured by user	
<code>positionGeneratorConfigInfo</code>	Position generator index configured by user	
<code>logCallback</code>	To generate system log.	Example: <code>logCallback.ShowPythonLog("xxxx")</code>

Code:

```

self.server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
self.server.bind(('', int(sensorConfigInfo)))
while self.isRunning:
try:
data, addr = self.server.recvfrom(4096)
recv_str = data.decode("ascii")
# check the received data format
# log = {'LogLevel': 0, 'Log': recv_str}
# logCallback.ShowPythonLog(log)
UDPDataExtractor(recv_str, logCallback)
except OSError as ex:
if ex.errno == 10040:
log = {'LogLevel': 2, 'Log': "Received data is too large. Please
send less positions at the same time."}
logCallback.ShowPythonLog(log)
except ET.ParseError:
log = {'LogLevel': 2, 'Log': "Invalid XML format. Please check the
data."}

```

Continues on next page

```
logCallback.ShowPythonLog(log)
else:
if hasNewPos:
sendNewPosition(callback, sensorId, positionGeneratorId,
logCallback)
```




ABB AB

Robotics & Discrete Automation

S-721 68 VÄSTERÅS, Sweden

Telephone +46 10-732 50 00

ABB AS

Robotics & Discrete Automation

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong New District

SHANGHAI 201319, China

Telephone: +86 21 6105 6666

ABB Inc.

Robotics & Discrete Automation

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

abb.com/robotics