A121 Cargo Example Application User Guide

User Guide

A121 Cargo Example Application User Guide

User Guide

Author: Acconeer AB

Version:a121-v1.12.0

Acconeer AB October 15, 2025

Contents

1 Acconeer SDK Documentation Overview

To better understand what SDK document to use, a summary of the documents are shown in the table below.

Table 1: SDK document overview.

Name	**Description**	**When to use**
	RSS API documentation (html)	
rss_api	The complete C API documentation.	- RSS application implementation - Understanding RSS API functions
	User guides (PDF)	
A121 Assembly Test	Describes the Acconeer assembly test functionality.	- Bring-up of HW/SW - Production test implementation
A121 Breathing Reference Application	Describes the functionality of the Breathing Reference Application.	- Working with the Breathing Reference Application
A121 Distance Detector	Describes usage and algorithms of the Distance Detector.	- Working with the Distance Detector
A121 SW Integration	Describes how to implement each integration function needed to use the Acconeer sensor.	- SW implementation of custom HW integration
A121 Presence Detector	Describes usage and algorithms of the Presence Detector.	- Working with the Presence Detector
A121 Smart Presence Reference Application	Describes the functionality of the Smart Presence Reference Application.	- Working with the Smart Presence Reference Application
A121 Sparse IQ Service	Describes usage of the Sparse IQ Service.	- Working with the Sparse IQ Service
A121 Tank Level Reference Application	Describes the functionality of the Tank Level Reference Application.	- Working with the Tank Level Reference Application
A121 Touchless Button Reference Application	Describes the functionality of the Touchless Button Reference Application.	- Working with the Touchless Button Reference Application
A121 Parking Reference Application	Describes the functionality of the Parking Reference Application.	- Working with the Parking Reference Application
A121 STM32CubeIDE	Describes the flow of taking an Acconeer SDK and integrate into STM32CubeIDE.	- Using STM32CubeIDE
A121 Raspberry Pi Software	Describes how to develop for Raspberry Pi.	- Working with Raspberry Pi
A121 Ripple	Describes how to develop for Ripple.	- Working with Ripple on Raspberry Pi
A121 ESP32 User Guide	Describes how to develop with A121 and ESP32 targets.	- Working with ESP32 targets
XM125 Software	Describes how to develop for XM125.	- Working with XM125
XM126 Software	Describes how to develop for XM126.	- Working with XM126
I2C Distance Detector	Describes the functionality of the I2C Distance Detector Application.	- Working with the I2C Distance Detector Application
I2C Presence Detector	Describes the functionality of the I2C Presence Detector Application.	- Working with the I2C Presence Detector Application
I2C Breathing Reference Application	Describes the functionality of the I2C Breathing Reference Application.	- Working with the I2C Breathing Reference Application
I2C Cargo Example Application	Describes the functionality of the I2C Cargo Example Application.	- Working with the I2C Cargo Example Application
	A121 Radar Data and Control (PDF)	
A121 Radar Data and Control	Describes different aspects of the Acconeer offer, for example radar principles and how to configure	- To understand the Acconeer sensor - Use case evaluation
	Readme (txt)	
README	Various target specific information and links	- After SDK download

2 Cargo

2.1 Introduction

This *Example Application* demonstrates how to utilize the Distance Detector and the Presence Detector for detecting the utilization level in a container as well as detecting whether a person is present in the container or not. The algorithms are optimized for three different container sizes 10 feet (3 m), 20 feet (6 m), and 40 feet (12 m). The user can decide if only one of the algorithms should be activated or to use both. If both detectors are used, the application alternates between measuring distance and measuring presence. To get an accurate presence estimate, the algorithm needs to run for a few seconds. This burst time is set to five seconds. Hence, each time the Presence Detector is activated, it measures for five seconds. From the utilization level part of the Example Application, the output is the distance to the cargo as well as the utilization level in meters and in percentage. The presence part outputs if presence is detected or not.

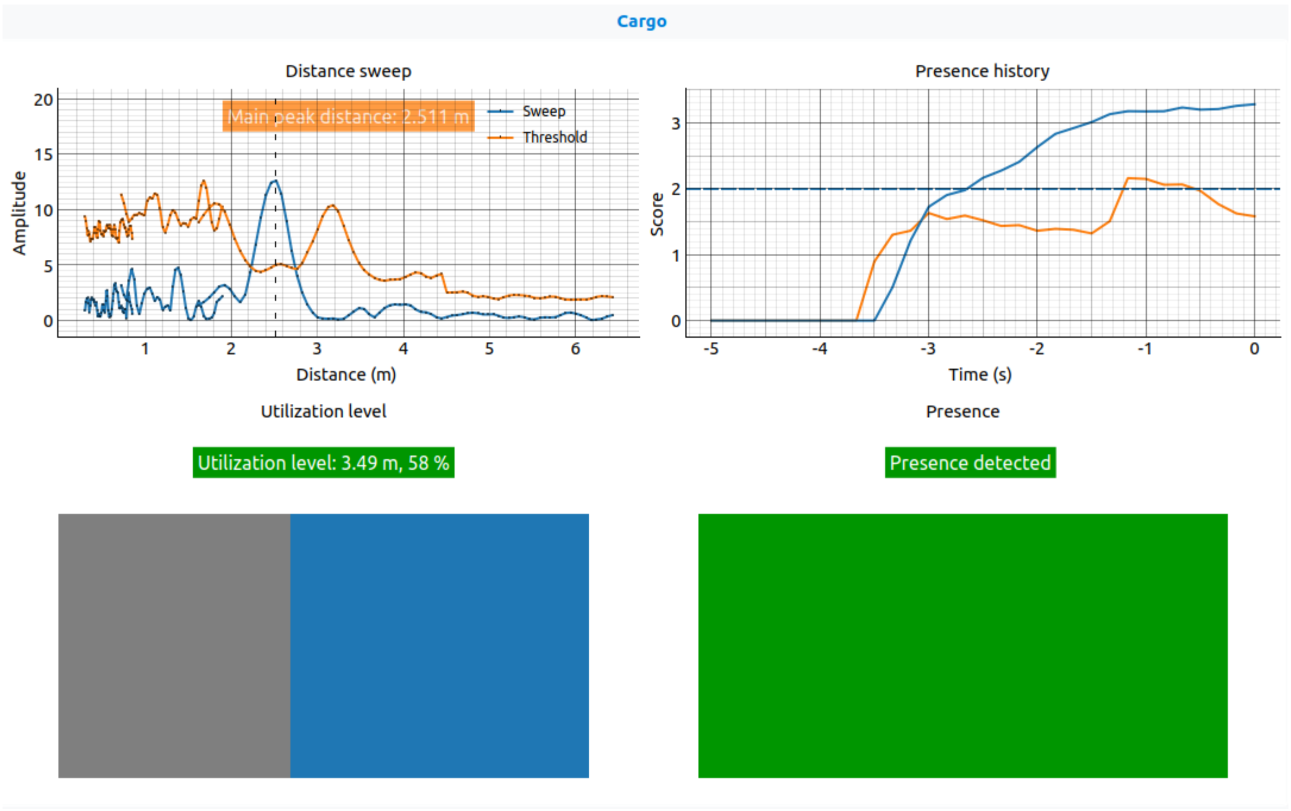
2.2 Your First Measurement

In this section, you will find the information needed to make your first measurement with the Cargo Example Application.

Exploration Tool

The Cargo Example Application is best evaluated with one of our *Evaluation Kits* together with the *Exploration Tool* application. Follow the steps in Getting started to get it up and running.

Select *Cargo* in the left hand side menu. The default configuration is the *No lens* preset, which has a higher signal quality and lower thresholds compared to the other presets. This is to get a good user experience when first testing the application in an open space with no lens. For testing in a container, we recommend using a lens (please contact Acconeer for more information) and one of the presets for 10 feet, 20 feet or 40 feet, see *Physical Integration* and *Presets*. Per default, only the utilization level detection is activated. However, it is only to tick the *activate presence* check box to get presence running as well. If both detectors are activated, there will be four plots in the GUI. The upper left plot shows the distance sweep measured for utilization level detection. The lower left plot shows an illustration of the container and visualizes the utilization level in the container. Presence is shown in the right plots. The upper plot shows the presence scores for inter frame presence and intra frame presence together with the respective thresholds. The lower right plot visualizes if presence is detected or not in the container.



2.3 Configuration

This section outlines how to configure the Cargo Example Application in common scenarios.

Presets

The Cargo Example Application has four predefined presets available:

10 feet (3 m)
This preset is optimized for a container with 10 feet (3 m) length.

20 feet (6 m)
This preset is optimized for a container with 20 feet (6 m) length.

40 feet (12 m)
This preset is optimized for a container with 40 feet (12 m) length.

No lens
This preset is for demo purposes in an open space without a lens.

Only utilization level is activated in the presets. However, the presence detector is also configured for each preset and can be activated by the user. All settings in the presets are set with an aim of low power consumption together with a good user experience during initial testing. This means that the burst and update rates are set higher than typical end-product levels and can be reduced to save power.

Further Configuration

Activation of Detectors

Utilization Level - The utilization level detection is activated with the *activate_utilization_level* parameter.

Presence Detection - Presence detection is activated with the *activate_presence* parameter. The presence detector measures for five seconds in each burst, which means that the maximum burst rate is 0.2 Hz. If the maximum burst rate is used and utilization level detection is activated, there will be maximum one frame measured for utilization level between the bursts even if the update rate is set higher. After each burst, the presence detector restarts, and the presence scores start from zero.

Adjusting for Decreased Power Consumption

Utilization Level - In the presets, the *update_rate* parameter is set quite high to get good response for initial testing. To lower power consumption, a much lower update rate could be used. It is possible to lower the *signal_quality* parameter to decrease power consumption as well. Due to limited cargo variation in the testing of this application, the default is set to 10 to have margin for less reflective materials. In our tests we used a flat plywood board facing the radar, see *Test Results*.

Presence Detection - Decreasing the *burst_rate* is the best way to lower power consumption for presence detection. This only affects the interval of the results and not the performance for each presence burst. The *update_rate* and *sweeps_per_frame* could also be decreased, but this will affect performance. The same thing applies for *signal_quality*, the trade off is performance versus power consumption.

Adjusting Thresholds

Since a container is made out of metal, which is highly reflective, the presence score in an empty container is higher compared to in an open space. Due to this, the thresholds for both utilization level and presence detection are set very high. With good physical integration, see *Physical Integration*, this might be improved, and the thresholds could be decreased to improve performance. The thresholds are adjusted with the *threshold_sensitivity* parameter for utilization level and *inter_detection_threshold* together with *intra_detection_threshold* for the presence detection.

Setting the Measurement Range

The measurement range is set through the *container_size* parameter. There are three sizes to choose from, 10 feet (3 m), 20 feet (6 m) and 40 feet (12 m).

Predefined Configurations

The Cargo Example Application utilizes both the Distance Detector and the Presence Detector. Some settings are predefined for both detectors to optimize performance for detection in a container. These cannot be changed. For the Distance Detector, these includes:

- `peaksorting_method = CLOSEST`
- `reflector_shape = PLANAR`

For the Presence Detector, these includes:

- `automatic_subsweeps` = True
- `inter_frame_fast_cutoff` = *update_rate* (no filtering)
- `inter_frame_presence_timeout` = None

For the settings that are not mentioned, the default values are used.

2.4 Physical Integration

A container is made of metal, which is highly reflective for radar signals. The signals bounce around and create a higher noise floor compared to open space. Due to this, all unwanted side reflections should be minimized with respect to integration. A lens is preferred to narrow down the beam and increase measurement range. Please contact Acconeer for more information about the lens suitable for this application. The position of the radar is preferably as centered as possible, at around 70 cm height in one of the container door's indentations.

2.5 Calibration

This Example Application does not have any calibration of its own. However, since it uses the Distance Detector in the background, there will be some calibrations when starting the application. Due to the predefined configuration of the Distance Detector, it is the *Noise Level Estimation* and the *Offset Compensation* that is performed. See sec_detector_calibration for more information.

2.6 Burst Rate and Timing

To get a good estimate of the presence detection, the Presence Detector measures for five seconds in a row before possibly switching to the Distance Detector. This means that the *update_rate* set for the presence detector controls the update rate within the burst. The measuring interval between the presence bursts is adjusted with the *burst_rate* parameter. Since the burst is five seconds long, the maximum burst rate is 0.2 Hz.

When the presence burst is not running, the time can be used to measure with the Distance Detector. In Exploration Tool, the update rate of the Distance Detector can be set independently of the presence burst rate. This means that the Distance Detector will measure as many frames as it can before it is time for a new presence burst, i.e., there could be a small drift in the presence burst rate if the measurement time does not add up perfectly with the presence detection gap. This does not affect the presence performance, since there is no filtering between presence bursts.

When only presence detection is active or when the burst rate of the presence detection is higher compared to the update rate of the utilization level measurement, the sensor will be measuring in the *update_rate* frequency of the presence detector even if it is between bursts. This is to keep pace when switching detectors in the Python Exploration Tool implementation.

2.7 Example App Output

The *mode* in the result indicates if the result belongs to a presence measurement or a distance measurement. If the result is from a presence measurement, the mode will be set to *presence* and all results relevant to distance will be set to None. The *presence_detected* will be true if presence is detected and false otherwise. If the result is from a distance measurement, the mode will be set to *distance* and all results based on the presence detector will be set to None. The *distance* is the actual distance measured, while *level_m* is the utilization level in meters from the back of the container. The *level_percent* is the utilization level converted to percentage.

2.8 Algorithm Signal Processing

The algorithm uses the exploration_tool-a121-distance_detector and the Presence Detector. Check out their respective documentation pages for more information.

2.9 Test Results

The tests were performed in a 20-foot container. Two different setups were tested, both using lens:

- Sensor placed flat on about 70 cm height on the door.
- Sensor placed at an angle at the top of the door.

Utilization Level

Utilization level was tested using a plywood board as cargo at different distances. For each distance, 30 frames were measured. The detection score was calculated as the percentage of the frames that correctly detected the distance. It is concluded that the flat sensor outperforms the angled one. The angled sensor does not detect the plywood board when the distance becomes closer than 4 m to the sensor, as seen in the table below.

Sensor position	Detection score 2 m	Detection score 3 m	Detection score 4 m
Flat	100%	100%	100%
Angled	0%	0%	100%

Presence Detection

Presence detection was tested with a person standing in different positions inside the container, creating a detection grid. For each position, the detection score was calculated as the percentage of correctly detected frames. To avoid false detections due to the high noise floor, the thresholds were set to 4 for inter presence and 2.75 for intra presence. A two second long transient of no detection in the beginning was allowed per position. With these conditions, a detection rate of roughly 100% was almost established. The performance can be seen in the figure below, where the sensor placement is illustrated with a red *"v"*.

3 Memory

3.1 Flash

The reference application compiled from example_cargo_main.c on the XM125 module requires around 104 kB.

3.2 RAM

The RAM can be divided into three categories, static RAM, heap, and stack. Below is a table for approximate RAM for an application compiled from example_cargo_main.c.

RAM	**Size (kB)**
Preset	*No Lens (20 ft)*
Static	1.3
Heap	12.3
Stack	3.4
Total	17.0

4 Power Consumption

The Cargo Example App utilizes the Distance- & Presence Detectors under the hood. Refer to the respective User Guides for power consumption of the detectors.

5 Disclaimer