A121 Vibration Example Application User Guide

User Guide

A121 Vibration Example Application User Guide

User Guide

Author: Acconeer AB

Version:a121-v1.13.0

Acconeer AB March 26, 2026

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 Vibration Example Application	Describes the functionality of the Vibration Example Application.	- Working with the Vibration Example 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 Vibration Measurement

This example application illustrates how the A121 sensor can be used to estimate the frequency content of a vibrating object, measured at a distance.

The Sparse IQ service produces complex data samples where the amplitude reflects the amount of returning energy and the phase the relative timing between the transmitted and received pulse. The phase can be used to detect small displacements. The vibration measurement application takes advantage of this, by forming a time series of the measured phase at a single distance point, which is then analyzed using an FFT. See interpreting_radar_data for more information about the produced data.

2.1 Usage

This section describes how to get started with the Vibration Example Application, how to configure it and important concepts.

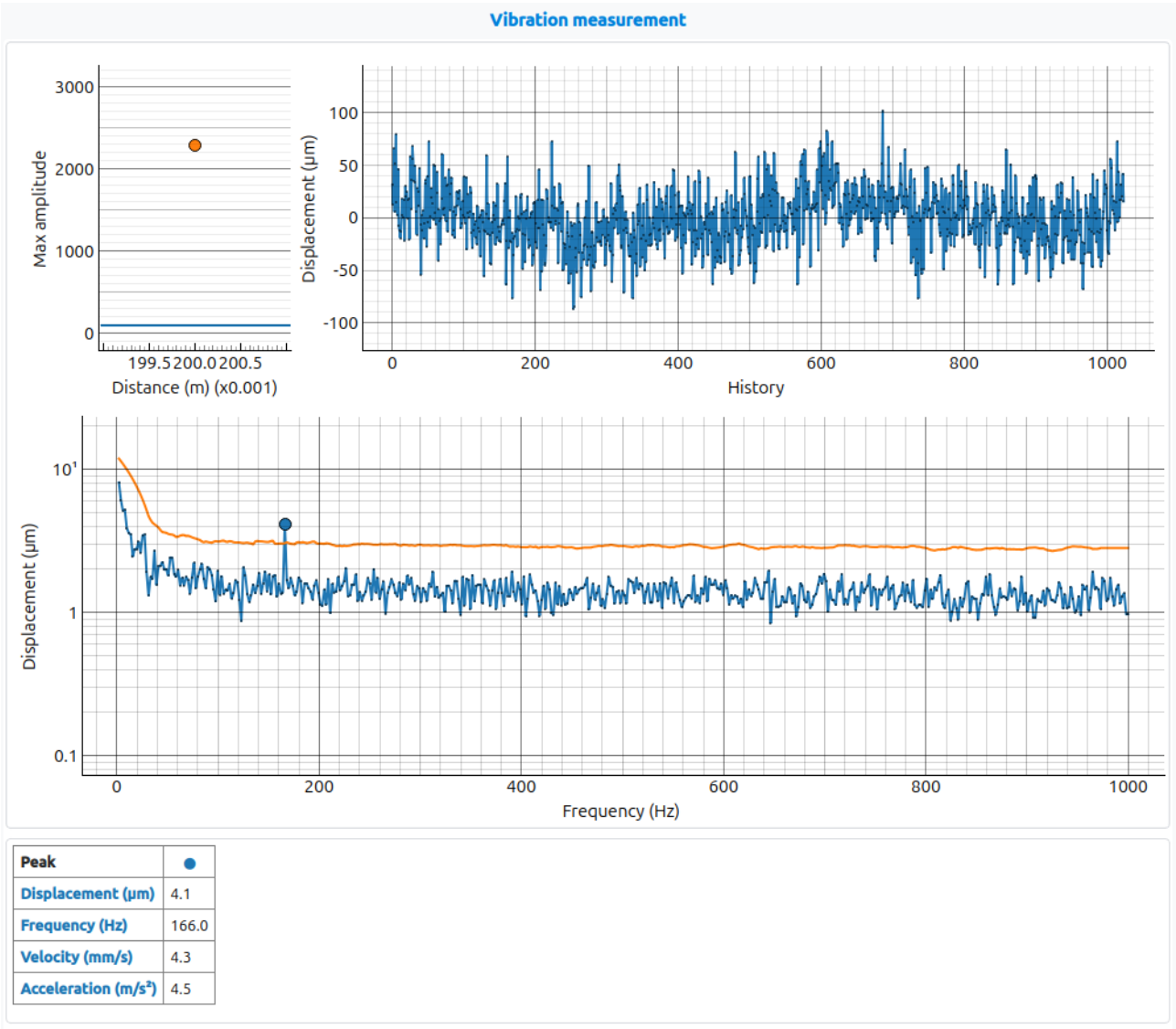
Your first measurement

To make a first measurement, it is recommended to connect one of Acconeer's evaluation kits to the Exploration Tool. Choose Vibration Measurement in the GUI. Place the sensor facing the object in the vibrating direction. Set the distance from the sensor to the object by adjusting the parameter *measured_point* and then press start measurement.

Exploration Tool

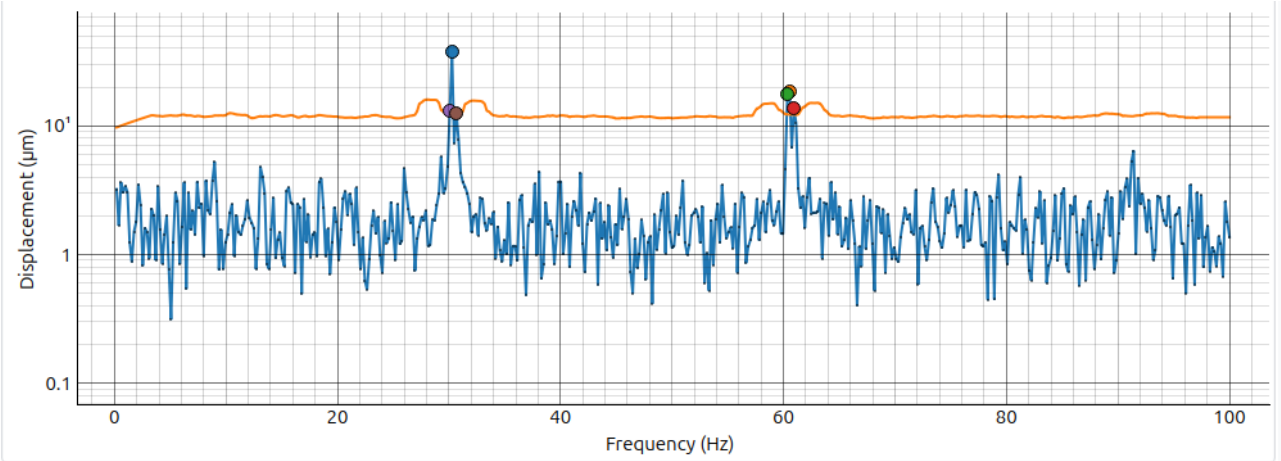
The GUI, see figure below, has three plots. The upper left plot indicates the amplitude at the measured point. The purpose of this is to provide visual feedback on whether or not the object is located at the right distance from the sensor. The blue horizontal line shows the chosen amplitude threshold. With no object in front of the sensor, the value should be close to zero. When moving an object into the correct location, the value increases. Next, the upper right plot shows the displacement of the measured object. Here, the phase has been converted to μm, indicating the physical displacement of the object. Lastly, the lower plot shows the calculated spectrum. In this case, the vibrating object has a dominant frequency at 166 Hz. The orange line shows the CFAR threshold, and the blue dot shows the detected peak. Note that the upper right plot and bottom plot only are updated when the object's amplitude is above the amplitude threshold.

Information about the detected peak is presented in the table below the spectrum plot.



Peak	●
Displacement (µm)	4.1
Frequency (Hz)	166.0
Velocity (mm/s)	4.3
Acceleration (m/s²)	4.5

Up to 30 peaks can be reported in the spectrum plot and the table below it. When multiple peaks are detected, the table might add horizontally scroll bars to accommodate all peaks. The controls below the table allow selecting between peak and RMS values, choosing the sorting order (by displacement, frequency, velocity, or acceleration), and limiting the number of displayed peaks.



Peak	●	●	●	●	●	●
Displacement (µm)	37.9	18.6	17.5	13.6	13.1	12.6
Frequency (Hz)	30.3	60.5	60.4	60.9	30.1	30.7
Velocity (mm/s)	7.2	7.1	6.6	5.2	2.5	2.4
Acceleration (m/s²)	1.4	2.7	2.5	2.0	0.5	0.5

Show values as: peak Sort by: displacement Show: all peaks

Embedded C

An embedded C application is provided in the Acconeer SDK, available at the Acconeer Developer Site.

The embedded application has the same presets as Exploration Tool and has most of the output Exploration Tool has. Max amplitude (upper left plot), peak displacements and frequencies of the spectrum (peak(s) in lower plot) are available in the result struct.

By default, the application prints the result using `printf` which usually is connected to stdout or a debug UART, depending on environment. The application is provided in source code.

Presets

In the Vibration Example App there are two presets that can help as a starting point depending on the desired frequency span that is of interest:

High frequency
: This preset will detect vibrations with frequencies up to 5000 Hz. The measurement distance is 0.2 m.

Low frequency
: For this preset, low frequency enhancement is enabled, see *Low Frequency Enhancement*. Vibrations with frequencies up to 100 Hz will be detected at a distance of 0.2 m.

Time Series

If continuous sweep mode is enabled, set by *continuous_sweep_mode*, the length of the time series is determined by the parameter *time_series_length*. A longer time series will give more points in the resulting spectrum. The frequency resolution is given by the ratio of the sweep rate and the number of points in the time series. If continuous sweep mode is not enabled, the time between two consecutive sweeps and two consecutive frames will not be the same. Due to this, the FFT will be calculated on a per frame basis, meaning that the time series length will be equal to the number of sweeps per frame, *sweeps_per_frame*, and *time_series_length* parameter will be redundant. The sweep rate, *sweep_rate*, should be set high enough to be able to capture the highest frequency of interest. The Nyquist theorem states that the sampling rate must be at least twice the highest frequency that is of interest.

The example app configuration also has a time constant, *lp_coeff*, applied together with an exponential filter, providing the possibility to time filter the calculated spectrum. A higher time constant gives a more stable spectrum. However, the spectrum will also adapt slower to changes in the frequency content.

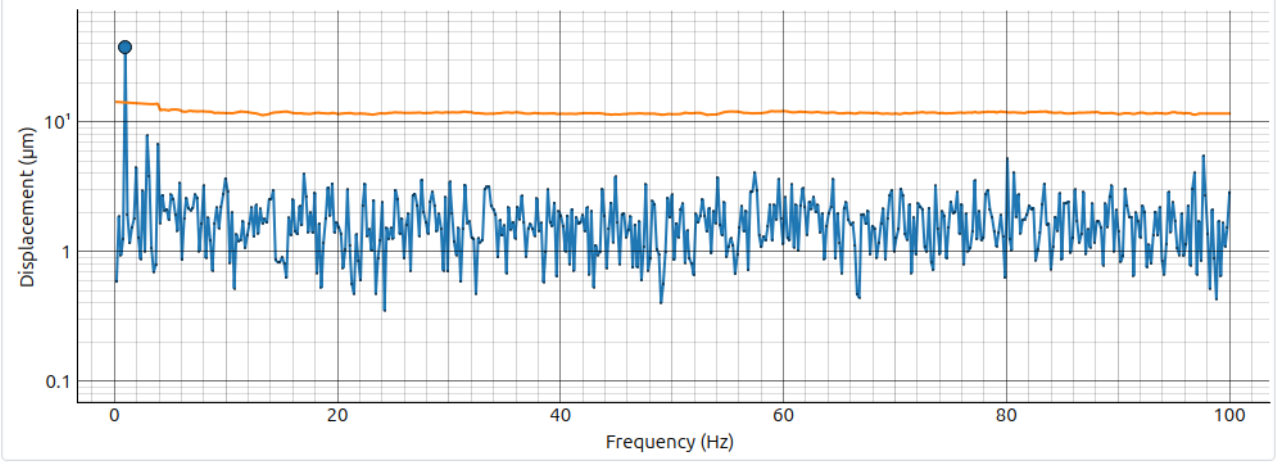
For every frequency in the spectrum, the power is converted to a displacement. The displacements can be computed in two different ways, either as amplitude or peak to peak value. This is set by the *reported_displacement_mode* parameter.

Thresholds

There are two threshold parameters, *amplitude_threshold*, and *threshold_margin*, which are used together. The first one determines if an object is in front of the sensor and if the algorithm should run, while the second one is used to find peaks in the frequency spectrum. The algorithm only calculates the FFT if there is an object in front of the sensor and the amplitude threshold, *amplitude_threshold*, sets the needed amplitude for an object to be considered detected. When an object is detected, the FFT is calculated and a CFAR threshold is computed to determine peaks in the spectrum, and the overall sensitivity of the CFAR threshold is adjusted by the threshold margin, *threshold_margin*.

Low Frequency Enhancement

Low frequencies, down to about 1 Hz, can sometimes be difficult to detect in the spectrum due to an increased noise floor. To correct the noise floor, a subsweep with a loopback measurement (see api_a121_configs for more information) can be added. The loopback measurement is used to adjust the phase for each sweep to compensate for jitter in the measurement. This will create a flat noise floor in the FFT, see figure below. This feature can be enabled with the parameter *low_frequency_enhancement*. Since a subsweep is added, the number of samples is increased and thus, the sweep rate cannot be as high as without this feature. Due to this, it is only recommended to enable this feature if there is an interest in detecting very low frequencies.



Peak	●
Displacement (µm)	37.5
Frequency (Hz)	1.0
Velocity (mm/s)	0.2
Acceleration (m/s²)	0.0

Example App Output

The *ExampleAppResult* class contains all output result from this example application. For the decision if an object is present in front of the sensor or not, the maximal amplitude is used, and this is reported in the `max_sweep_amplitude`. As well as reporting the largest detected displacements, `peak_displacements`, and for which frequencies these are found, `peak_frequencies`, all displacement together with all the frequencies they are calculated for is reported in the `lp_displacements` and the `lp_displacements_freqs`. However, usually it is the maximal displacement and the corresponding frequency that is of highest interest. The last reported result is the `time_series_std`, which is the standard deviation of the last processed time series.

3 Memory

3.1 Flash

The reference application compiled from example_vibration_main.c on the XM125 module requires around 72 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_vibration_main.c.

RAM	Size (kB)
Static	6.8
Heap	4.2
Stack	5.0
Total	16.0

4 Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB (“Acconeer”) will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user’s responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders.

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user’s responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user’s product or application using Acconeer’s product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.