

How to implement in-sensor vibration monitoring with ISM330IS, with embedded ISPU



Denise SANFILIPPO
ST Employee

on 2023-07-26 06:35 AM -
edited on 2023-09-22 06:37 AM
by [ADMIN](#) [Laurids_PETERSEN](#)

Summary

This knowledge article explains how you can easily implement accurate in-sensor vibration-monitoring applications with an IMU featuring an embedded intelligent sensor processing unit (ISPU). This intelligent ISM330IS sensor can implement sliding discrete Fourier transform (SDFT) for continuous and accurate monitoring, on a power budget of a few microwatts.

Prerequisites

Software

In this tutorial the following software components are used:

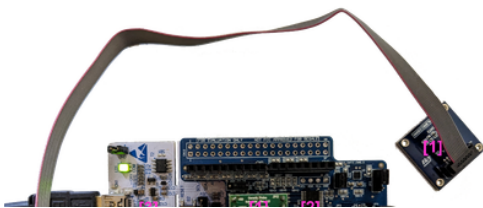
- [X-CUBE-ISPU](#), expansion software package for STM32Cube;
- [Unicleo-GUI](#), graphical user interface for X-CUBE-MEMS1, motion MEMS and environmental sensor software expansion for STM32Cube;
- [STM32CubeIDE](#), integrated development environment for STM32.

Moreover, the following [GitHub repository](#), in which are present ready examples, is considered.

Hardware

The following hardware components are used:

- [1] [STEVAL-MKI230KA](#), which is the DIL24 adapter board for [ISM330IS](#), the industrial IMU MEMS sensor with ISPU.
- [2] The industrial motion MEMS sensor expansion board [X-NUCLEO-IKS02A1](#).
- [3] [NUCLEO-F401RE](#), STM32 Nucleo development board.



Version history

Last update:
2023-09-22 06:37 AM

Updated by:
[ADMIN](#) [Laurids_PETERSEN](#)

Contributors

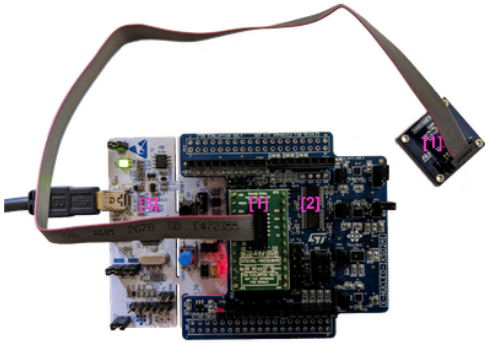
[Denise SANFILIPPO](#)

[Laurids_PETERSEN](#)

Feedback

Related content

[How to implement in-sensor inclination monitoring ...](#)

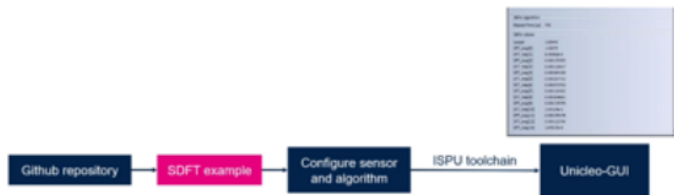


Hands-on

This section shows how to customize the new SDFT example present on the GitHub repository. This example implements a sliding algorithm to compute the discrete Fourier transform (SDFT) that you can use to build vibration monitoring applications. Unicleo-GUI is used to visualize the results.

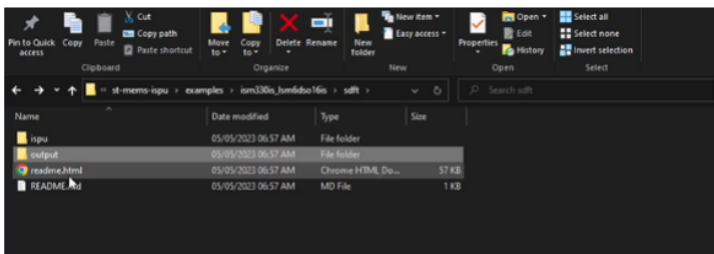
Vibration monitoring

Monitor vibrations in the frequency domain with SDFT



Open the GitHub repository.

After cloning the repository locally, it can be opened. It contains different folders, among which examples and tutorials. Inside the folder *examples*, consider the SDFT example folder. The readme file describes that this example implements the sliding discrete Fourier transform (SDFT) algorithm at 26 Hz. The output folder contains the prebuilt files.



During this hands-on session, the focus is on the *ISPU* folder to configure the example as needed.

Open the STM32CubeIDE and import the SDFT example, by browsing the SDFT example and selecting the *Eclipse* folder inside the GitHub repository. When the example is imported, the code is open, in particular focusing on the *main.c* file.

At line 24, `#define WIN_LEN 26u` defines the length of the window over which the discrete Fourier transform is computed. In this specific case it is configured at 26 samples, which means 1 s at 26 Hz. The data rate can be changed to 12.5 Hz and the window length is updated accordingly. For example, configure it at 2 s, so this

Feedback



Open the STM32CubeIDE and import the SDFT example, by browsing the SDFT example and selecting the *Eclipse* folder inside the GitHub repository. When the example is imported, the code is open, in particular focusing on the main.c file.

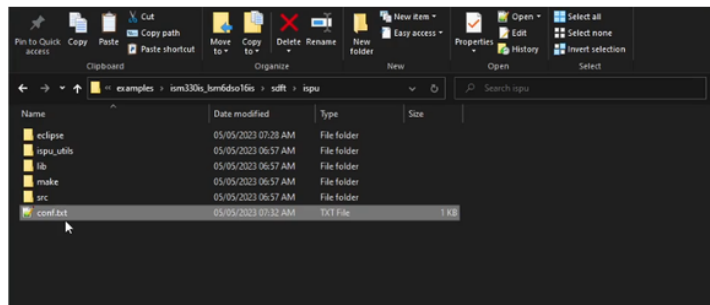
At line 24, #define WIN_LEN 26u defines the length of the window over which the discrete Fourier transform is computed. In this specific case it is configured at 26 samples, which means 1 s at 26 Hz. The data rate can be changed to 12.5 Hz and the window length is updated accordingly. For example, configure it at 2 s, so this corresponds to 25 samples: #define WIN_LEN 25u (line 24 of main.c).

```
1 | #define WIN_LEN 25u
```

This is the only line of the code (line 24 of main.c) which can be modified if desired, because everything is applied automatically in the configuration during the initialization code. In fact, the MotionFT library (line 55 of main.c) is run to compute the magnitudes of the DFT. Then with a for cycle (line 58 of main.c) the values are copied to the output registers of the ISPU embedded inside ISM330IS.

```
1 | void __attribute__((signal)) algo_00(void)
2 | {
3 |     MFT_input_t data_in;
4 |     float dft_mag[DFT_LEN];
5 |     MFT_output_t data_out = { .dft_mag = dft_mag };
6 |
7 |     data_in.sample = (float)cast_sint16_t(ISPU_ARAW_Z) * ACC_SENS;
8 |     MotionFT_update(mft, &data_out, &data_in);
9 |
10 |    cast_float(ISPU_DOUT_00) = data_in.sample;
11 |    for (uint16_t i = 0u; i < DFT_LEN; i++) {
12 |        cast_float(ISPU_DOUT_02 + ((int16_t)i * 4)) = dft_mag[i];
13 |    }
14 |
15 |    int_status = int_status | 0x1u;
16 | }
```

Open the configuration file conf.txt file inside the *ISPU* folder.

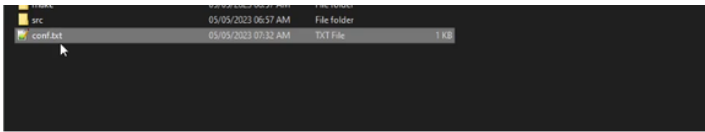


By default, both the accelerometer and the ISPU are configured at 26 Hz. Line 1 is modified setting:

```
acc odr 12.5
```

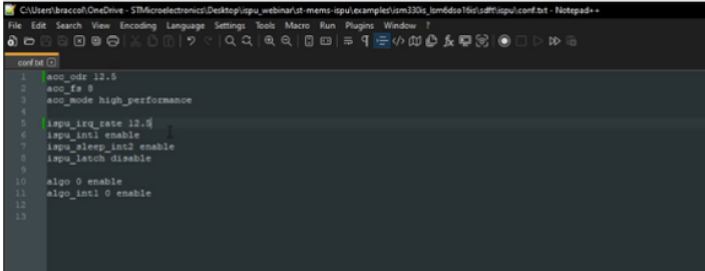
Feedback





By default, both the accelerometer and the ISPU are configured at 26 Hz. Line 1 is modified setting:

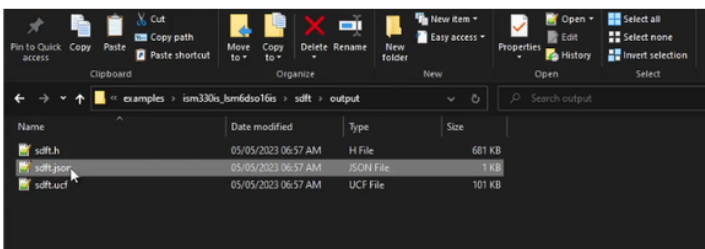
```
acc_odr 12.5
ispu_irq_rate at 12.5 Hz
```



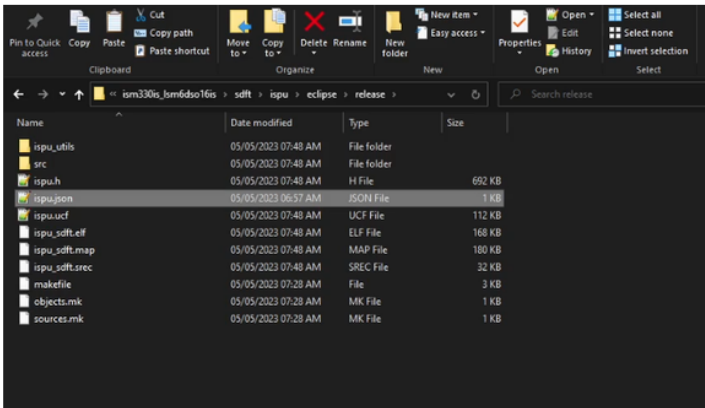
After saving and closing the conf.txt file, go back to STM32CubeIDE and build the example ispu_sdft.

To visualize the results, you can use UnicLeo-GUI. For this purpose, you need the JSON file describing the outputs.

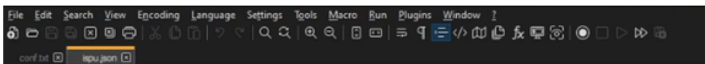
Copy the sdft.json from the example sdft inside the output folder.



Paste the file into the build folder, where the copied sdft.json is renamed as ispu.json



Open the ispu.json file, the size of the array of the DFT magnitude (line 11 of the ispu.json file) is modified by knowing that the length DFT_LEN is calculated by dividing the window length by 2, rounding down and adding 1: $((WIN_LEN / 2u) + 1u)$ as described at line 25 of file main.c in STM32CubeIDE. In this case, WIN_LEN is 25u, so the "size" (line 11 of the ispu.json file) is set to 13. Save and close the file.



Feedback



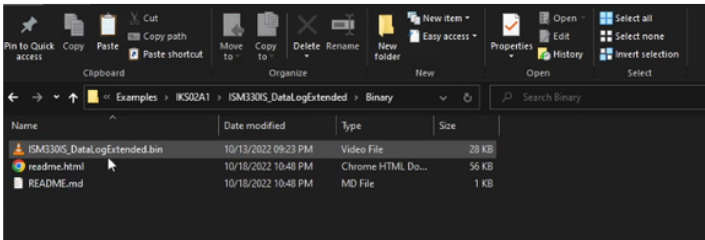
knowing that the length DFT_LEN is calculated by dividing the window length by 2, rounding down and adding 1: $((WIN_LEN / 2u) + 1u)$ as described at line 25 of file main.c in STM32CubeIDE. In this case, WIN_LEN is 25u, so the "size" (line 11 of the ispu.json file) is set to 13. Save and close the file.

```

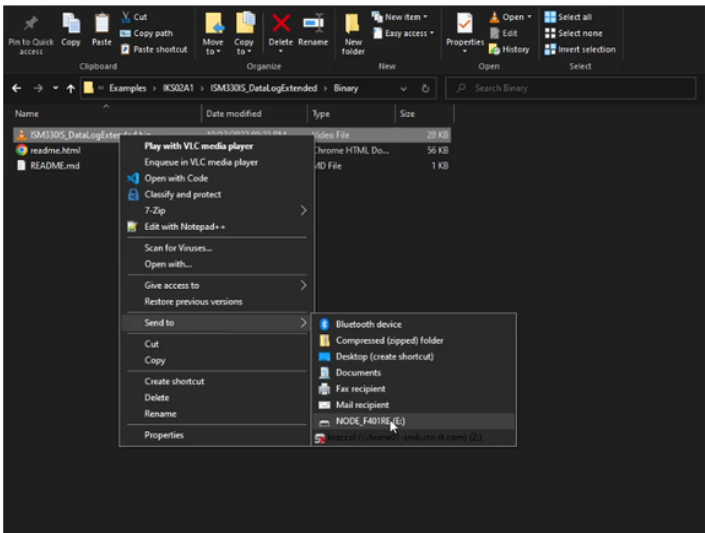
1
2
3
4 "output":
5 {
6   {
7     "name": "sample",
8     "type": "float"
9   },
10  {
11    "name": "DFT_mag",
12    "type": "float",
13    "size": 13
14  }
15 }
16

```

Flash the STM32 Nucleo board with the appropriate firmware. To do so, go back to the x-cube-ispu package into *projects* folder: select NUCLEO-F401RE, then the expansion board that we are using, X-NUCLEO-IK02A1, and our sensor, ISM330IS. In the binary folder is a prebuilt binary file of the firmware.



You can flash this binary file just by copying it to the STM32 Nucleo board.



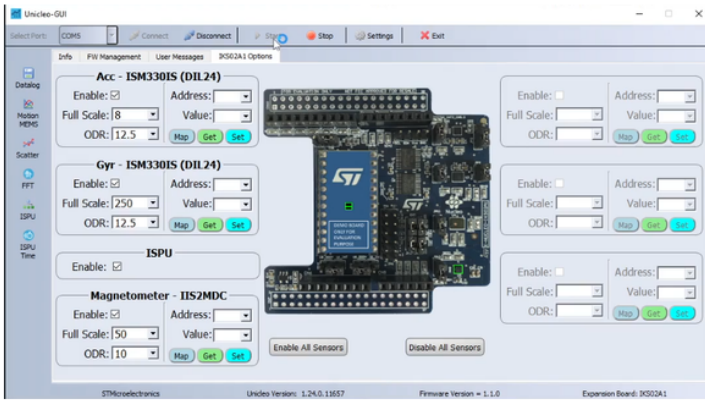
Now that the STM32 Nucleo board is flashed, open the Unicleo-GUI, which automatically connects to the board. Press the start button:



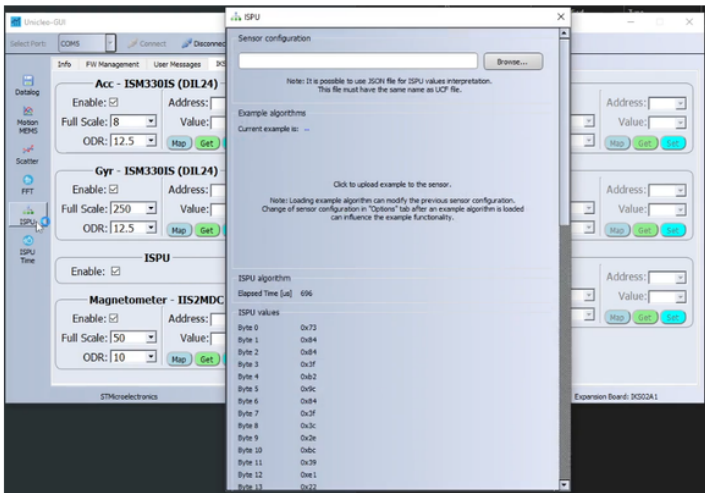
Feedback



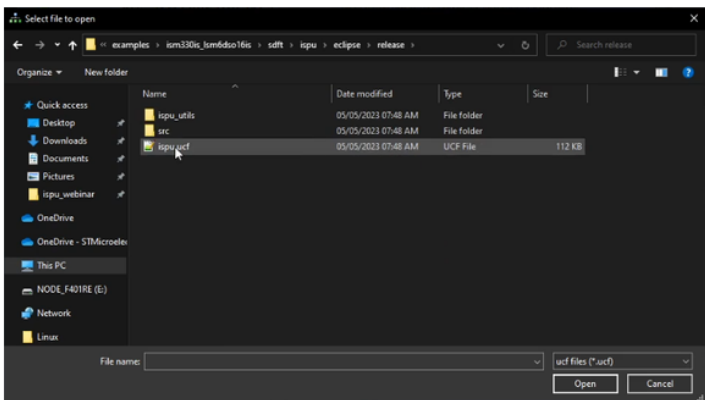
Press the start button:



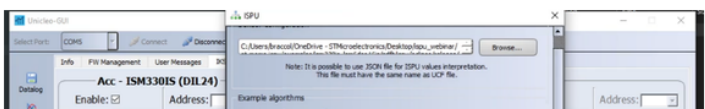
Then navigate to the ISPU tab:



Afterwards, browse for the .ucf file by clicking the browse button, and locate it inside the release folder of the sdtf example and load the file.

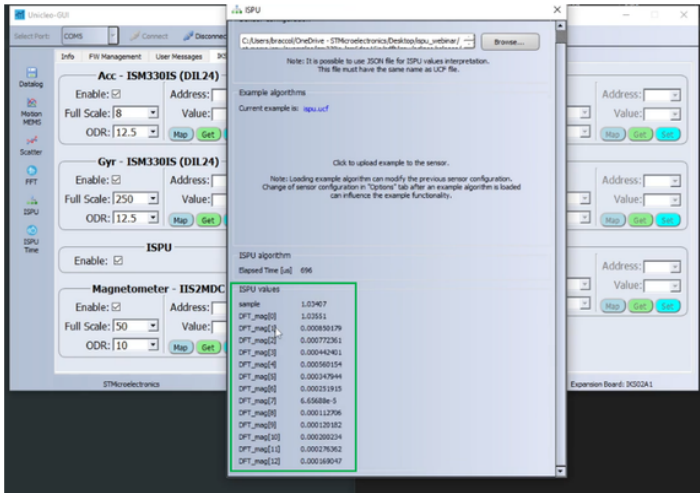


Now, it is computing the DFT magnitudes at various frequencies; in this particular case the values are close to zero because the device is stationary:

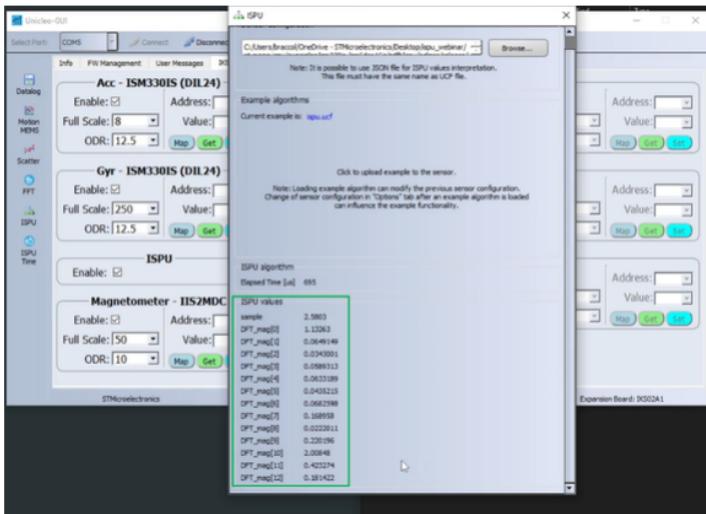


Feedback

Now, it is computing the DFT magnitudes at various frequencies; in this particular case the values are close to zero because the device is stationary:

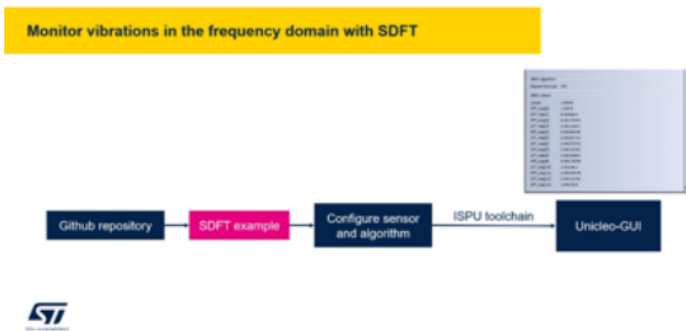


If the board is shaken, some frequency components are appearing:



Unicleo-GUI is used to visualize the results for the SDFT algorithm, but the SDFT algorithm is only the starting point.

Vibration monitoring



Feedback



In a real application, SDFT is run according to our needs. Instead of copying the results into the output registers as described at lines 57-60 of the main.c file, the output of the MotionFT library can be used as an input for further processing and for our vibration monitoring algorithms.

```

1 | cast_float(ISPU_DOUT_00) = data_in.sample;
2 | for (uint16_t i = 0u; i < DFT_LEN; i++) {
3 |     cast_float(ISPU_DOUT_02 + ((int16_t)i * 4)) = dft_mag[i];
4 | }

```

Conclusion

This article shows how it is possible to easily implement accurate in-sensor vibration-monitoring applications with an IMU featuring an embedded intelligent sensor processing unit (ISPU). This intelligent sensor, ISM330IS, can implement sliding discrete Fourier transform (SDFT) for continuous and accurate monitoring on a power budget of a few microwatts.

You can find a dedicated webinar on the topic at the following link, which also includes this demo: [Webinar: In-sensor monitoring with intelligent MEMS sensors](#)

You may also be interested in reading the following knowledge article: [How to implement in-sensor inclination monitoring with ISM330IS, with embedded ISPU](#)

MEMS sensors

0 Kudos

Feedback

Top ↑



About STMicroelectronics

Connect with us

Browse

Compliance, ethics & privacy

Who we are

Contact ST offices

Shortcuts

Ethics and compliance

Investor relations

Find sales offices & distributors

Sitemap

ST ethics hotline

Sustainability

Community

Privacy portal

Innovation & technology

Newsroom

implement sliding discrete Fourier transform (SDFT) for continuous and accurate monitoring on a power budget of a few microwatts.

You can find a dedicated webinar on the topic at the following link, which also includes this demo: [Webinar: In-sensor monitoring with intelligent MEMS sensors](#)

You may also be interested in reading the following knowledge article: [How to implement in-sensor inclination monitoring with ISM330IS, with embedded ISPU](#)

MEMS sensors



0 Kudos

Top



About STMicroelectronics

Who we are

Investor relations

Sustainability

Innovation & technology

Careers

Blog

General terms and conditions

Connect with us

Contact ST offices

Find sales offices & distributors

Community

Newsroom

Events & trainings

Browse

Shortcuts

Sitemap

Compliance, ethics & privacy

Ethics and compliance

ST ethics hotline

Privacy portal

Feedback

Follow us



All rights reserved © 2025 STMicroelectronics | [Terms of use](#) | [Sales terms & conditions](#) | [Trademarks](#) | [Privacy portal](#) | [Manage cookies](#)