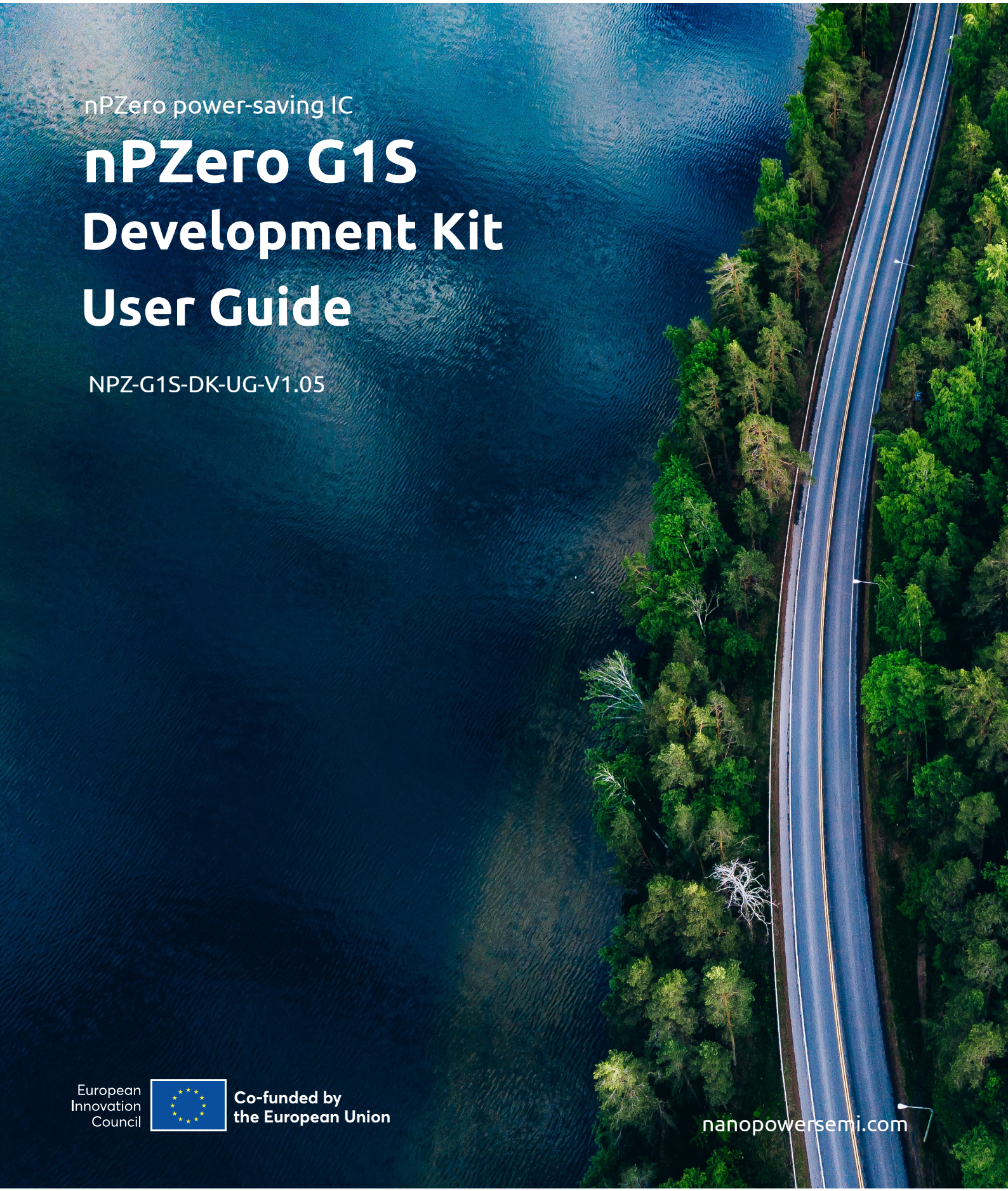
nanopower semiconductor

nPZero power-saving IC

nPZero G1S Development Kit User Guide

NPZ-G1S-DK-UG-V1.05



European Innovation Council

Co-funded by the European Union

nanopowersemi.com

Revision History

Revision	Date	Description
1.05	2026.08.28	• *Hardware description* subsection updated; • *Building and flashing the example code* subsection updated; • *Legal Notices* section updated and renamed to ”Terms of Use”; • Terms referring nPZero G1S device and the DevKit changed to “nPZero G1S” and “nPZero G1S Development Kit” accordingly; • DevPortal links updated; • Document name changed to ”NPZ-G1S-DK-UG-V1.05”
1.04	2025.11.07	• Code testing instructions added; • Code running instructions updated; • *Custom Project* section added; • Sections updated; • Images updated

European Innovation Council
Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or EISMEA. Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction **5**
1.1 Hardware description . . . 6
1.2 Hardware overview . . . 6
1.3 Block Diagram . . . 8
1.3.1 Host I2C . . . 8
1.3.2 Host UART . . . 8
1.3.3 Peripherals . . . 8
1.3.4 Stackable connectors . . . 8
1.3.5 nPZero G1S IC . . . 8
1.3.6 Power sources . . . 9
1.3.7 Power switches . . . 9
1.3.8 Debug interface . . . 9
1.3.9 Reset . . . 9
1.3.10 LEDs . . . 9

2 Development Kit contents **10**
2.1 Hardware content . . . 10
2.2 Software content . . . 13

3 Work requirements **14**
3.1 Hardware prerequisites . . . 14
3.2 Software prerequisites . . . 14

4 Example Code **15**
4.1 Connecting the hardware . . . 15
4.2 Assembly . . . 17
4.3 Displaying logs through a terminal emulator . . . 18
4.4 Testing the pre-programmed example . . . 20
4.5 Building and flashing the example code . . . 20

5 Custom Project **24**
5.1 STM32 System settings . . . 24
5.2 nPZero IC configuration . . . 32

6 Health, Safety and Environmental Notices **35**

7 Terms of Use **36**
7.1 Liability disclaimer . . . 36
7.2 Definitions and Acceptance . . . 36
7.3 Ownership and Intellectual Property . . . 36
7.4 Purpose and Permitted Use . . . 36
7.5 Prohibited Uses . . . 36
7.6 Safety Responsibilities . . . 37
7.7 Software and Firmware . . . 37
7.8 Regulatory Compliance . . . 37
7.9 Warranty Disclaimer . . . 37
7.10 Limitation of Liability . . . 37
7.11 Indemnification . . . 37
7.12 Governing Law and Jurisdiction . . . 37
7.13 Severability . . . 37
7.14 Waiver . . . 37
7.15 Independent Contractors . . . 38
7.16 Survival . . . 38
7.17 Waste and Recycling (WEEE) . . . 38

Acronyms and abbreviations

In this document, several technical terms are abbreviated for ease of reference. Below is a list of key acronyms and abbreviations used throughout the document.

ADC Analog-to-Digital Converter (a device that converts analog signals, such as voltage, into a digital format that can be processed by a microcontroller or other digital systems)

GUI Graphical User Interface (a visual system that allows users to interact with graphical elements like icons, buttons, and menus, making the interaction more intuitive and user-friendly)

I^2C Inter-Integrated Circuit (a communication protocol used for short-distance data exchange between components on a circuit board)

JTAG Joint Test Action Group (a standard for testing and debugging integrated circuits and printed circuit boards using a serial communication interface)

LED Light Emitting Diode (a semiconductor device that emits light when an electric current passes through it)

MCU Microcontroller Unit (a compact integrated circuit designed to govern a specific operation in an embedded system, often containing a processor, memory, and input/output peripherals on a single chip)

PMOD Peripheral Module (a standardized connector used to add functionality to microcontroller development boards, typically for connecting peripherals like sensors, displays, or communication modules)

POR Power-On Reset (a circuit that initializes a system by resetting it when the power is first applied, ensuring that the device starts in a known state)

SPI Serial Peripheral Interface (a communication protocol used to transfer data between microcontrollers and peripheral devices over short distances)

SRAM Static Random-Access Memory (a type of memory that retains data as long as power is supplied, used in computers and other devices for fast access to data)

UART Universal Asynchronous Receiver-Transmitter (a communication protocol used for serial data transmission between a microcontroller and other devices, such as sensors or a terminal application)

1 Introduction

The nPZero G1S Development Kit (DevKit) includes hardware and firmware source code facilitating efficient prototyping and development of applications based on the nPZero power-saving IC. This kit combines ultra-low power consumption with autonomous operation, making it perfect for devices where energy efficiency is critical.

The kit can autonomously power up sensor peripherals, read data and wake up the system as needed. It manages the system without an active MCU host by taking over the controller role, powering down the host and autonomously handling power-up, configuration and reading of peripheral data from up to 4 sensors. When required, it will power up the host and return control based on user-defined rules.

This allows the host to remain off for longer periods while the system continues to detect and respond to events, conserving energy. It significantly reduces power consumption both in active and sleep modes when managing peripherals.

Key features of the nPZero G1S Development Kit:

- Compatible with one I2C host MCU;
- Supports up to four peripheral sensors;
- I2C and SPI supported for peripheral communication;
- Flexible I/O pins can be used as interrupt inputs or trigger outputs;
- Autonomous sensor polling with threshold-based triggers;
- Ultra-low power operation;
- Crystal oscillator with programmable clock output;
- ADC with two channels: one internally connected to VBAT, and one external input on pin ADC_IN;
- 128-byte internal SRAM for initialization commands for the peripherals;
- Operates in a supply voltage range from 1.7 V to 3.6 V;
- Operating temperature range is from -20°C to +70°C;
- Integrated power switches allow power gating of the host and peripherals.

1.1 Hardware description

This section provides a detailed overview of the hardware components essential for the nPZero G1S Development Kit. The following descriptions cover key features, specifications and connections of each component, offering a comprehensive understanding of the role of the various hardware blocks within the system.

The DevKit supports up to four I2C or SPI sensor peripheral boards, external interrupts and internal SRAM for peripheral initialization commands, making it suitable for a wide range of applications. In addition, an internal crystal oscillator with a dedicated clock output is available for precise timekeeping. A two-channel ADC in its turn can be used to monitor battery voltage and external input.

Please note that nPZero G1S reference layout has been updated, please refer to *Typical application subsection* in nPZero G1S Datasheet V0.95 or later.

1.2 Hardware overview

The DevKit images illustrate the nPZero G1S Development Board (DevBoard), the STM32 MCU host and the sensor peripheral boards as shown in Figures 1, 2, 3 and 4. The MCU host is located at the centre of the Development Board. It can also be replaced with other radio and MCU boards that utilize the same Arduino Uno compatible connectors. The MCU host has a 10-pin JTAG connector labelled as J1 which allows programming the MCU directly. Additionally, the kit includes a PMOD interface connector (J17) allowing manual connection of any compatible MCU or radio chip.

The SPI sensor peripheral board shown in Figure 3 and the I2C sensor peripheral board illustrated in Figure 4 can be connected to the corresponding PMOD connectors on the Development Board. The PMOD connectors (J1-J4) allow connections for up to four different sensor boards. Any I2C or SPI compatible sensor boards using PMOD connectors can be connected in addition to the two provided peripheral boards. Furthermore, a connector (J14) is available for connection of an analog sensor.

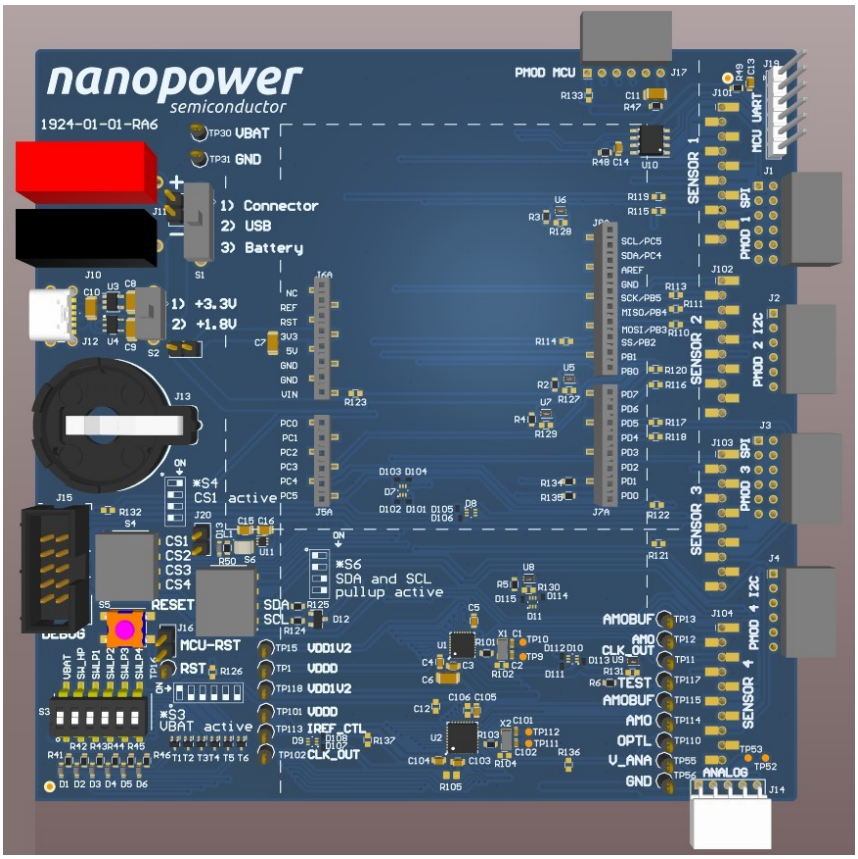


Figure 1: nPZero G1S Development Board

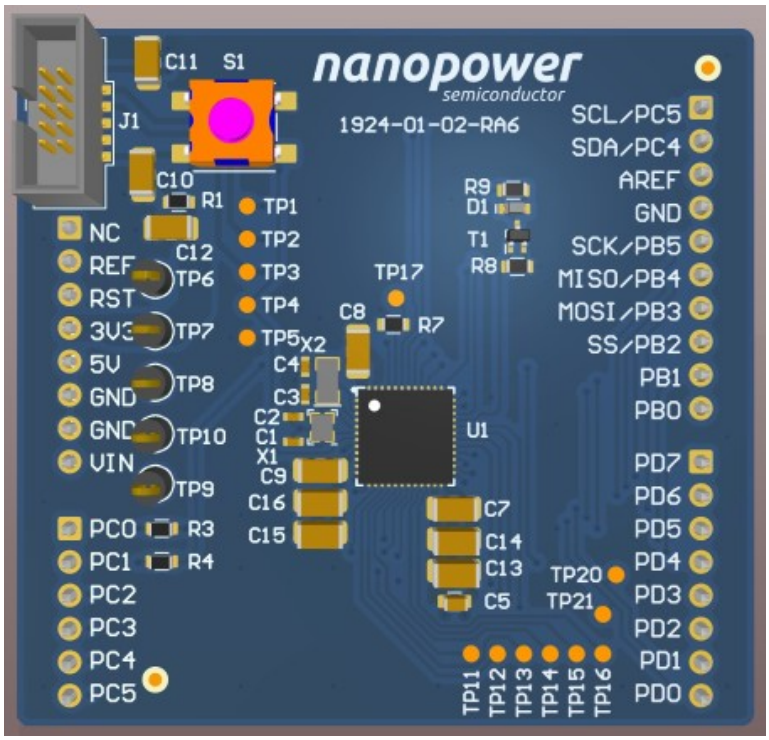


Figure 2: STM32 MCU host

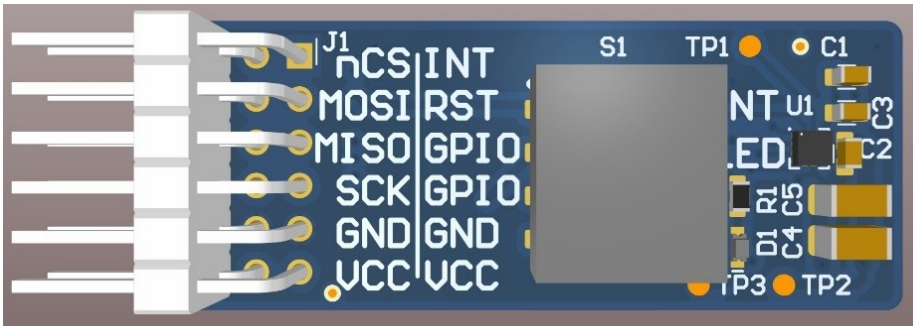


Figure 3: LIS2DW12 SPI accelerometer sensor peripheral board

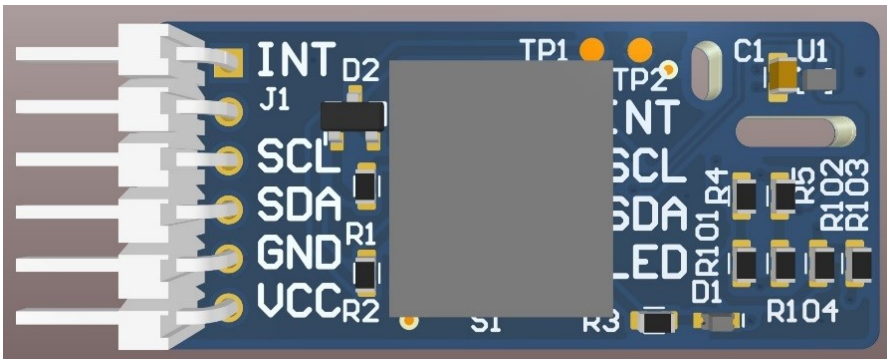


Figure 4: AS6212 I2C temperature sensor peripheral board

1.3 Block Diagram

The following block diagram provides a high-level overview of the DevKit system architecture, illustrating key components and their interconnections. This diagram explains the overall layout and each part of the system.

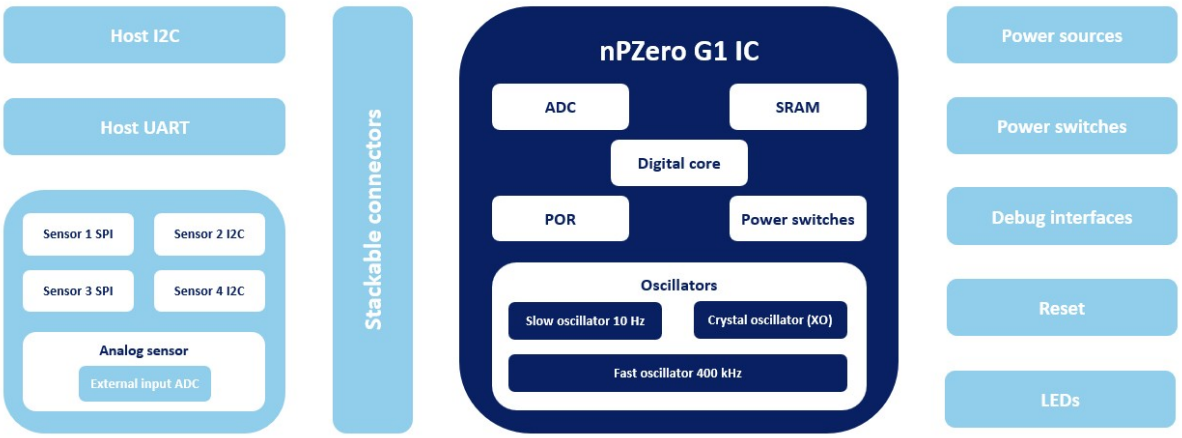


Figure 5: nPZero Development Kit

1.3.1 Host I2C

Communication between the MCU (also referred as “host” in this document) and the nPZero IC is handled over the I2C interface. During normal operation the nPZero IC operates as an I2C slave device supporting the Standard mode with communication speeds of up to 100 kHz. In this mode, the MCU host can read from and write to the available registers as well as the SRAM memory space of the nPZero IC.

1.3.2 Host UART

The MCU’s UART (Universal Asynchronous Receiver-Transmitter) is a serial communication protocol commonly used for sending and receiving log data, making it useful for printing messages to a terminal application.

1.3.3 Peripherals

The PMOD connectors (J1-J4) on the DevBoard allow connection of up to four peripheral sensor boards. Additionally, the connector J14 is available for manually connecting an analog sensor.

1.3.4 Stackable connectors

The connectors J5A, J6A, J7A, and J8A are stackable connectors that follow the Arduino Uno R3 pinout standard. The connectors are located at the center of the DevBoard , through them the MCU host board is connected.

1.3.5 nPZero G1S IC

The nPZero G1S is a power-saving IC that can significantly decrease the power consumption of sensor applications where energy management is critical. Detailed information on technical specifications, structure and operation of the device can be found in the nPZero G1S Datasheet.

1.3.6 Power sources

The DevBoard supports multiple power source options, providing flexibility to choose the most suitable power input.

- **Power Source Selection (Switch S1):** Switch S1 is used to choose between three different power sources:
    - **Top Position:** Selects banana connectors J9 and J10 or pin header J11.
    - **Middle Position:** Selects USB connector J12.
    - **Bottom Position:** Selects coin cell battery J13.
- **Voltage Selection (Switch S2): This switch allows you to select the operating voltage for the system, either 1.8V or 3.3V.**

1.3.7 Power switches

The switch S3 on the DevBoard enables manual selection and control of power distribution, as well as connection and disconnection of LEDs. It is configured with VBAT as the input voltage from the battery, SW_HP as the high-power switch output, and SW_LP [1-4] as the low-power switch outputs.

The behaviour of the power switch outputs (pins SW_LP [1-4]) can be set as a power output that can supply power directly to the peripheral or as a logic output (with configurable polarity) to control an external power switch.

Additionally, each power switch includes output voltage detection, which can be enabled to ensure that peripheral communication only starts once the voltage at the switch output is close to the system supply.

1.3.8 Debug interface

The J15 debug connector on the DevBoard is a 10-pin connector used for debugging I2C and SPI interfaces. It is recommended to connect the Aardvark, Beagle, Cheetah or a similar debugger from Total Phase when using this interface for debugging. Through the switch S4 it is possible to select which SPI interface is currently being debugged.

1.3.9 Reset

The connector J16 is a 2-pin connector on the DevBoard that can be shorted with a jumper. When shorted, the MCU host can be used to reset the board. If there is a risk of current leakage into the MCU host when it is powered off, remove this jumper. Furthermore, switch S5 is a reset button on the DevBoard, used for resetting the entire board. The button S1 on the STM32 MCU host board can be used to reset the MCU exclusively.

1.3.10 LEDs

The LEDs on the DevBoard have a corresponding position on the switch S3 that can be toggled off to disconnect the LED. The LEDs on the sensor boards have a corresponding position on the switch S1 that can be toggled off to disconnect the LED to allow accurate measurements of system current consumption.

2 Development Kit contents

The nPZero G1S Development Kit (DevKit) provides the hardware necessary to evaluate and develop applications for the nPZero G1S chip, as well as access to software examples and documentation for the nPZero Driver API.

2.1 Hardware content

The DevKit contains the following hardware boards.

nPZero G1S Development Board

The DevKit includes the nPZero G1S Development Board (DevBoard) which features the nPZero power-saving IC and can work with other boards in the kit.

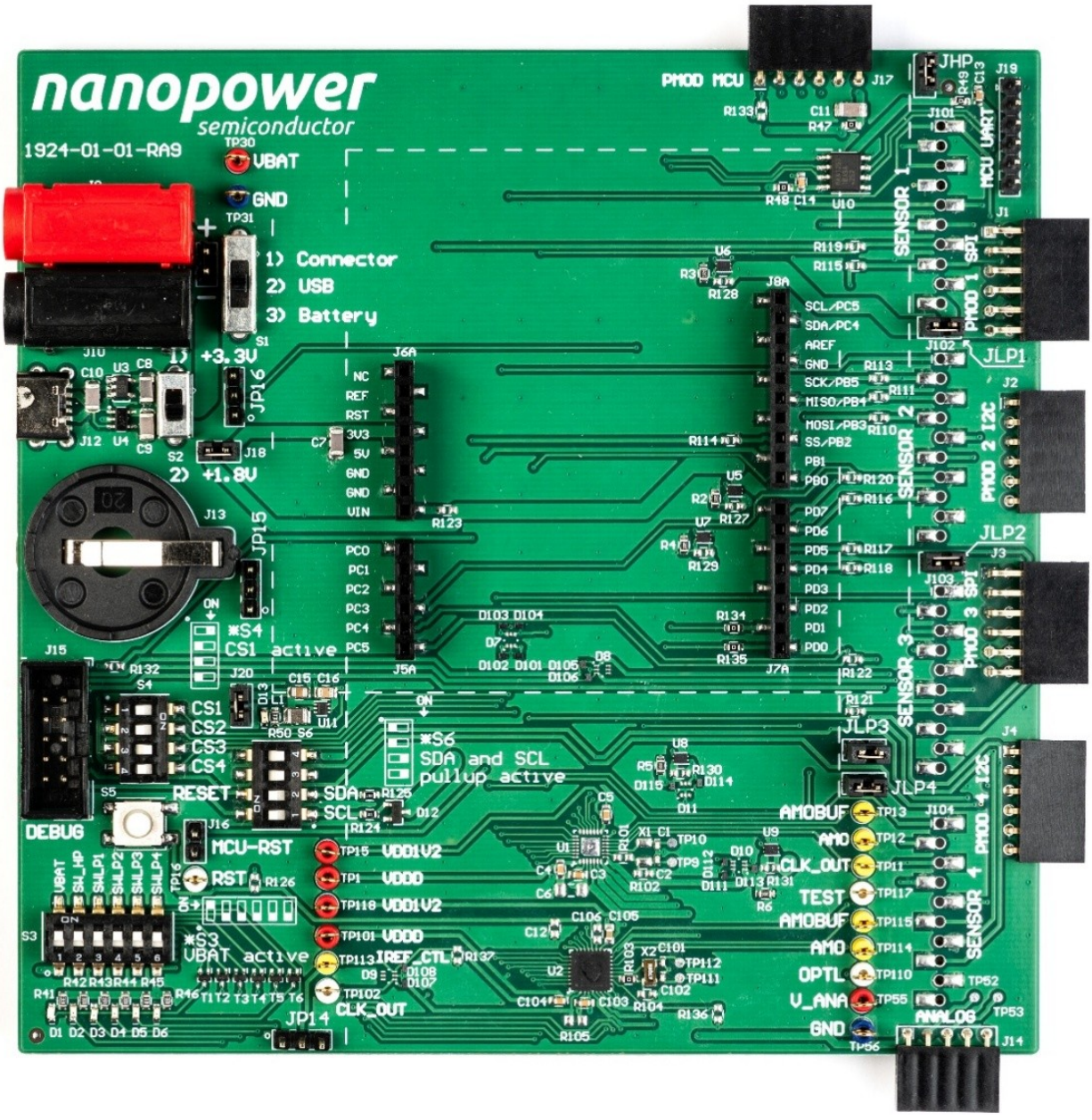


Figure 6: G1S Development Board

MCU Host

The DevKit includes the STM32L053C8U6 MCU which can be easily connected to the carrier board via Arduino Uno compatible stackable connectors. This low-power MCU from STMicroelectronics can be used to test the benefits of the nPZero power-saving IC. An onboard JTAG connector is used for programming the MCU.

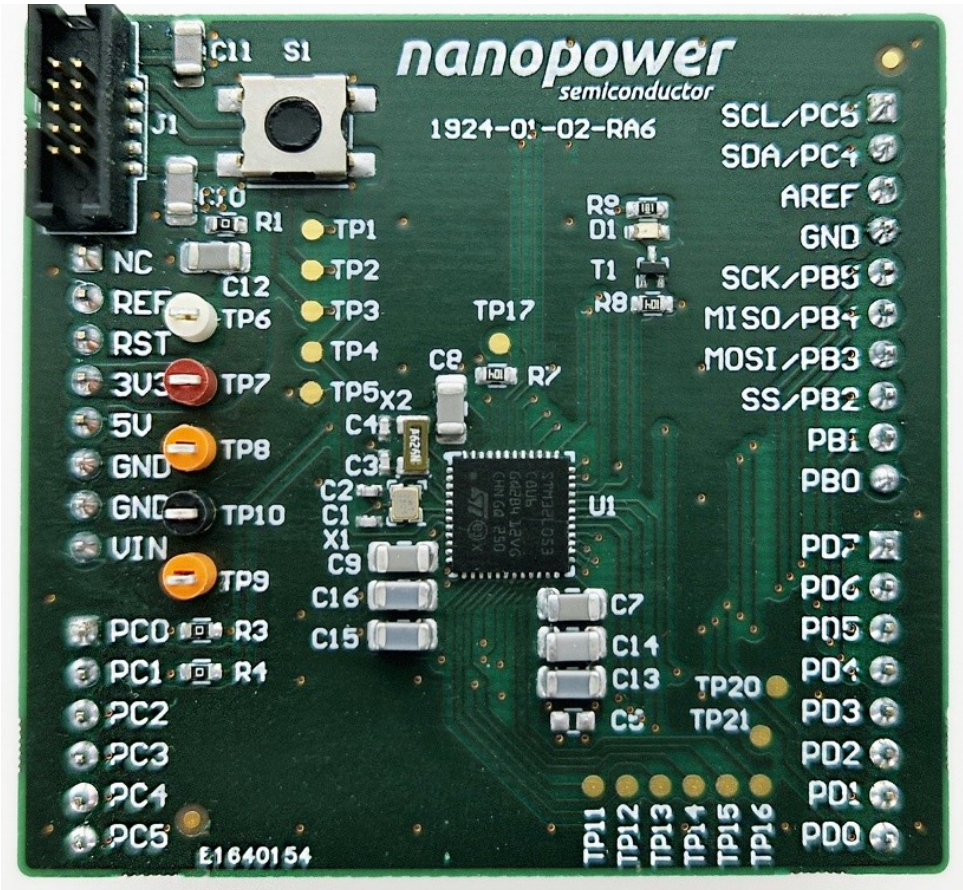


Figure 7: STM32L053C8U6 Host

Acceleration sensor

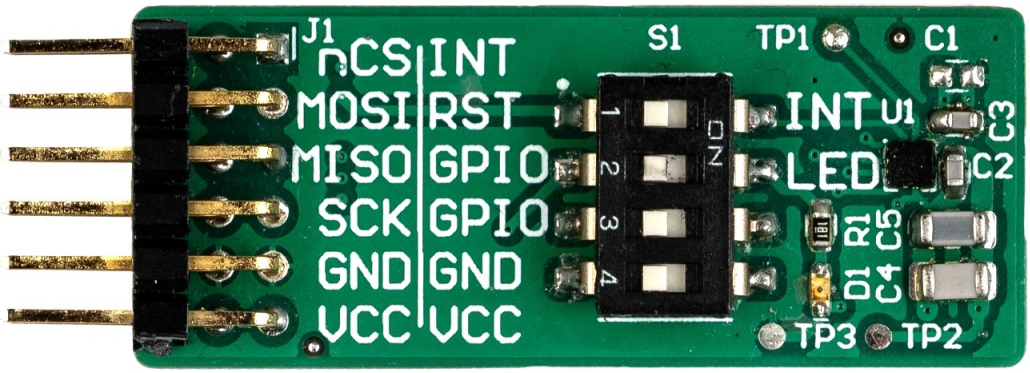


Figure 8: LIS2DW12 accelerometer

The DevKit includes the LIS2DW12 accelerometer from STMicroelectronics. This SPI sensor can be easily connected to the DevBoard through the Peripheral 1 or Peripheral 3 PMOD connectors.

Temperature sensor

The DevKit includes the AS6212 temperature sensor from ams. This I2C sensor can be easily connected to the DevBoard through the Peripheral 2 or Peripheral 4 PMOD connectors.

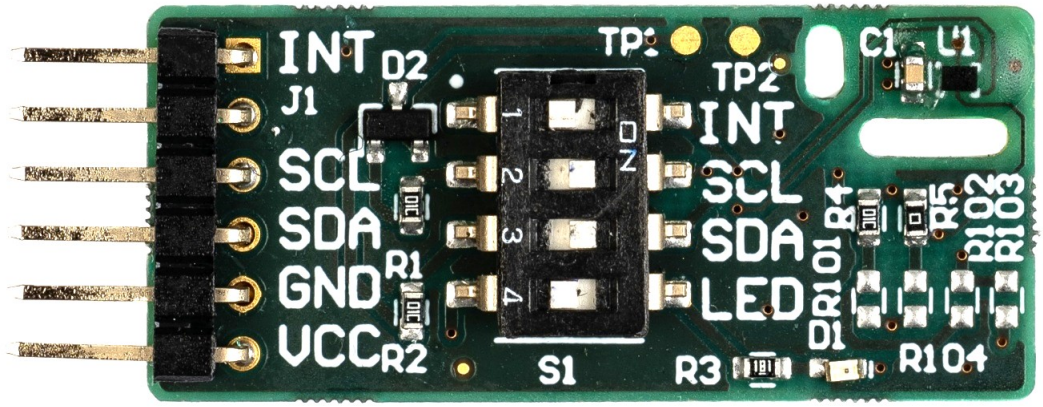


Figure 9: AS6212 temperature sensor

2.2 Software content

The DevKit downloadable content includes access to example code for running the sensors bundled with the DevBoard on the STM32 host MCU. Additionally, it provides documentation for the nPZero Driver API. This driver offers an interface for the STM32 MCU host to control and manage the nPZero IC at the register level, allowing users to efficiently handle power-related functionalities in their applications.
The downloadable content also includes a Hardware Abstraction Layer (HAL) for I2C operations, which can be modified by users to suit their specific hardware configurations. The nPZero Driver API can be downloaded from the Nanopower Semiconductor DevPortal here

Furthermore, a Graphical User Interface (GUI) tool called nPZero Configurator is available to help you easily set up the device and its peripherals. This tool simplifies the configuration process by allowing you to adjust settings through an intuitive interface without manually writing the code. Once your setup is complete, the nPZero Configurator will automatically generate the necessary code for the API ensuring that everything is ready to be used. This makes the entire setup process faster, more efficient and user-friendly, especially for those who want to minimize manual coding efforts. The nPZero Configurator GUI can be accessed from the Devportal here.

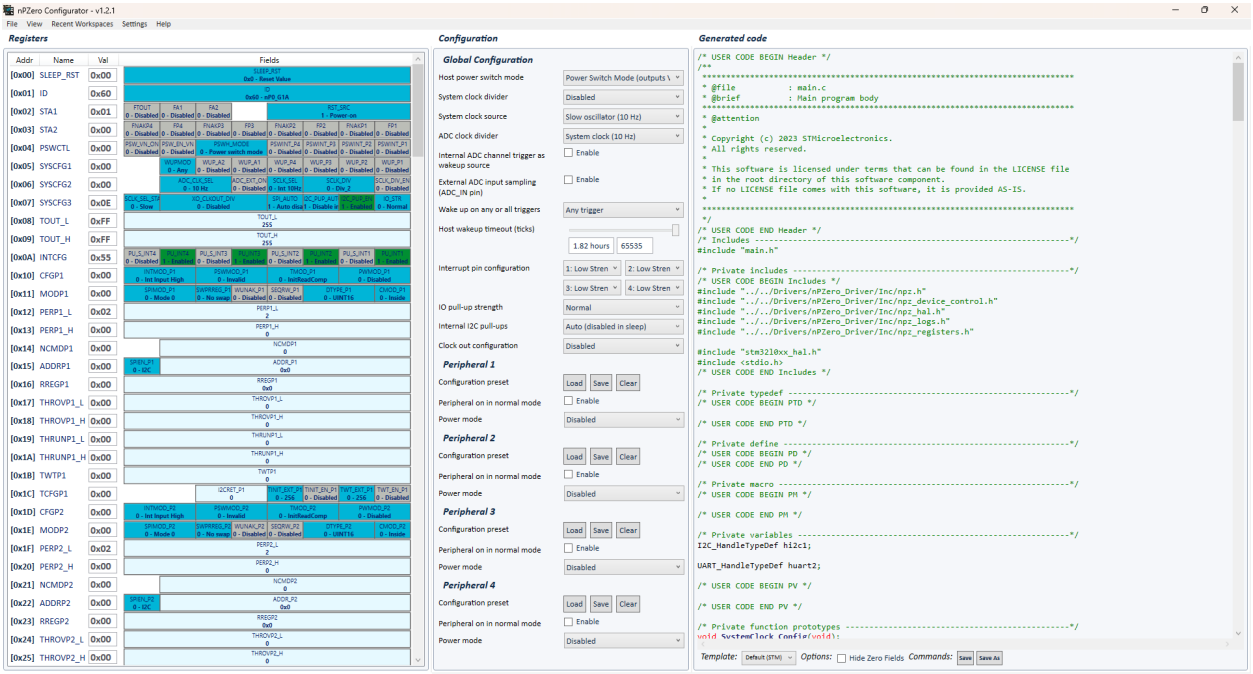


Figure 10: nPZero Configurator GUI

3 Work requirements

Before proceeding to the next chapter, please ensure the required hardware and software prerequisites are available.

3.1 Hardware prerequisites

Assembling and flashing projects on the DevKit requires the following hardware:

- nPZero G1S Development Kit;
- Power source:
  - Coin cell battery, external power through USB-C or external power through banana connectors;
- Segger J-LINK Debugger;
- USB to UART adapter (e.g. FTDI TTL-232R-5V or FTDI TTL-232R-5V-WE).

3.2 Software prerequisites

Required hardware for developing and running projects on the DevKit includes:

- STM32CubeIDE;
- SEGGER J-Link Software;
- nPZero Configurator;
- nPZero Driver API;
- Terminal application (e.g. PuTTY).

4 Example Code

The host MCU comes pre-flashed with example code out of the box, and it is recommended to test the pre-flashed example before building and flashing any other examples in order to ensure everything is set up correctly. To test the pre-flashed example, please go through the steps below:

4.1 Connecting the hardware

1. Unpack the DevKit.
2. Connect the host MCU to the DevBoard via the Arduino Uno compatible stackable connectors illustrated in Figure 12.
3. Connect the J-LINK debugger to the JTAG connector on the host MCU indicated in Figure 13.
4. Plug in a USB cable between the PC and the JTAG probe.
5. Connect the FTDI cable to the MCU UART connector (J19) on the DevBoard , the pinout of the J19 connector is provided in Figure 11.
6. Attach the USB side of the FTDI cable to the PC.
7. Connect the J20 jumper to enable operation of D1-D6 LEDs on the DevBoard. Please note that LEDs D1-D6 indicate the connection of the following:
    - D1 - Power supply;
    - D2 - Host power switch;
    - D3 - Peripheral 1 power switch;
    - D4 - Peripheral 2 power switch;
    - D5 - Peripheral 3 power switch;
    - D6 - Peripheral 4 power switch.
8. Verify that the J18 jumper is connected, providing power to the host MCU.
9. To power up the board, you can choose one of the following options, as shown in Figure 12:

- Connect a USB-C cable through the USB-C connector;
- Connect an external power supply through the banana connectors;
- Insert a coin cell battery.

Depending on which power option you chose, slide the power switch S1 to the corresponding option (1 is for banana connectors, 2 – for USB-C, 3 – for a battery), as presented in Figure 12.

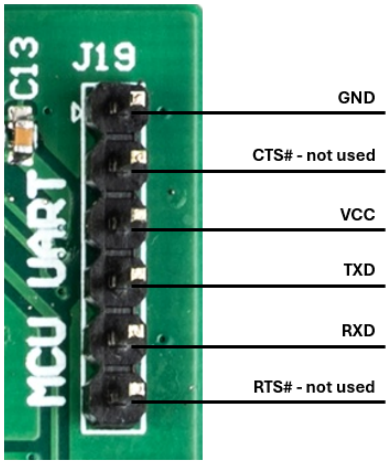


Figure 11: MCU UART connector J19 pinout

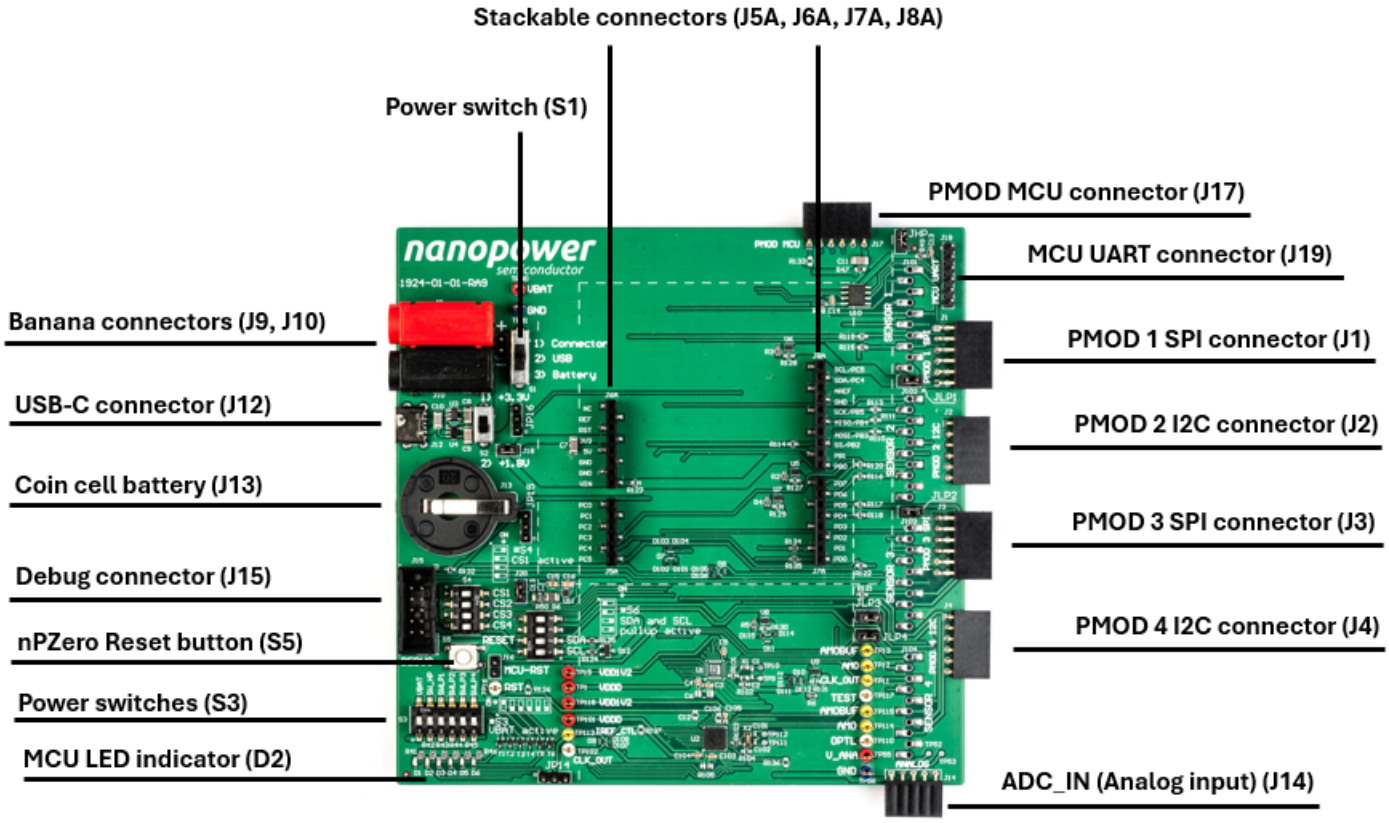


Figure 12: DevBoard connectors

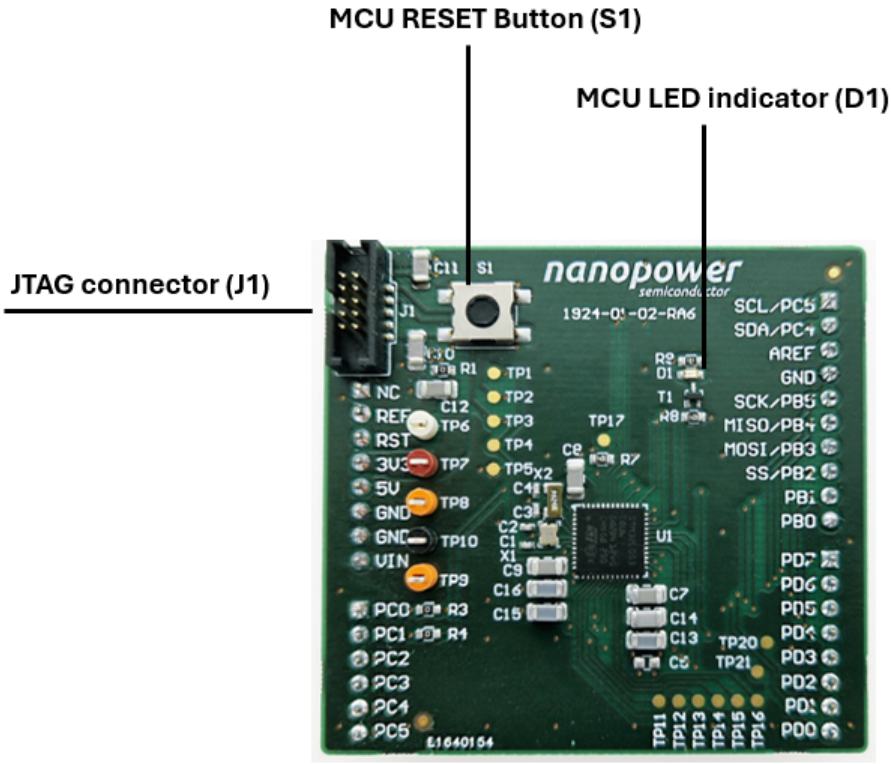


Figure 13: Host MCU JTAG connector

4.2 Assembly

After setting up the Development Kit hardware, please follow the steps below to connect the accelerometer and temperature sensor, which are both required to run the standard example:

1. Connect the accelerometer to the DevBoard using the **PMOD 3 SPI connector** as shown in Figure 14.
2. Connect the temperature sensor to the DevBoard using the **PMOD 4 I2C connector** as shown in Figure 14.

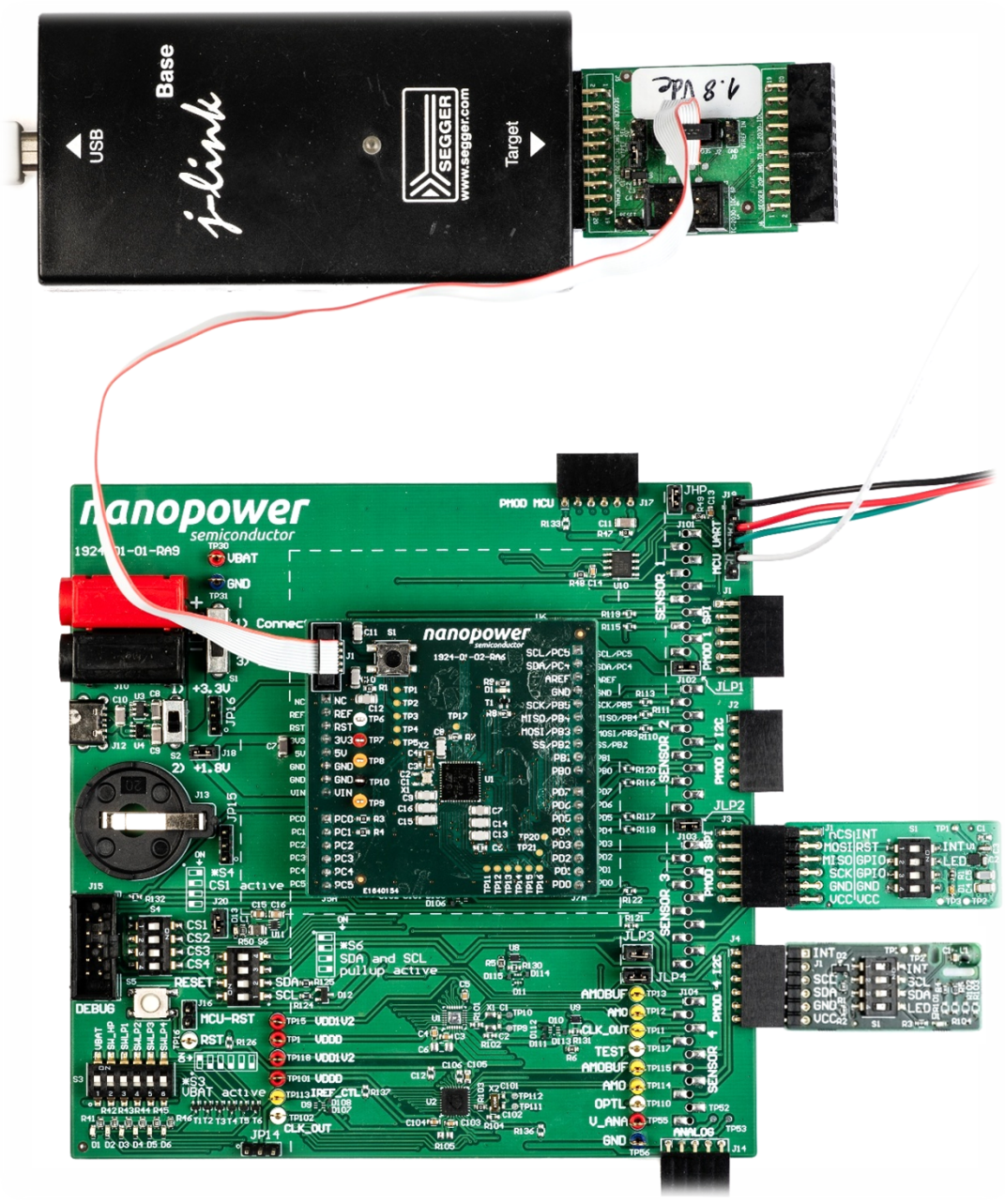


Figure 14: DevBoard with the accelerometer and temperature sensor connected

4.3 Displaying logs through a terminal emulator

The log output from the host MCU can be displayed in a terminal emulator such as PuTTY. Start the terminal emulator software and connect to the COM port being used with the following UART settings:

- Baud rate: 115200;
- 8 data bits;
- 1 stop bit;
- No parity;
- No flow control.

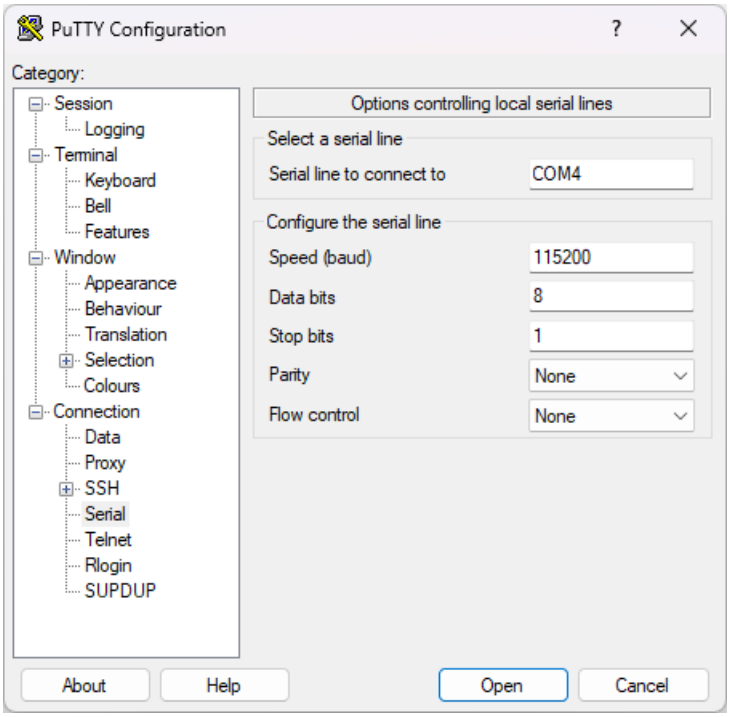


Figure 15: Connecting the COM port in the PuTTY terminal emulator

- Note that it might be necessary to reset the DevBoard by pressing the **RESET (S5)** button to restart the nPZero and host MCU to get the COM device to be displayed in the terminal window correctly.

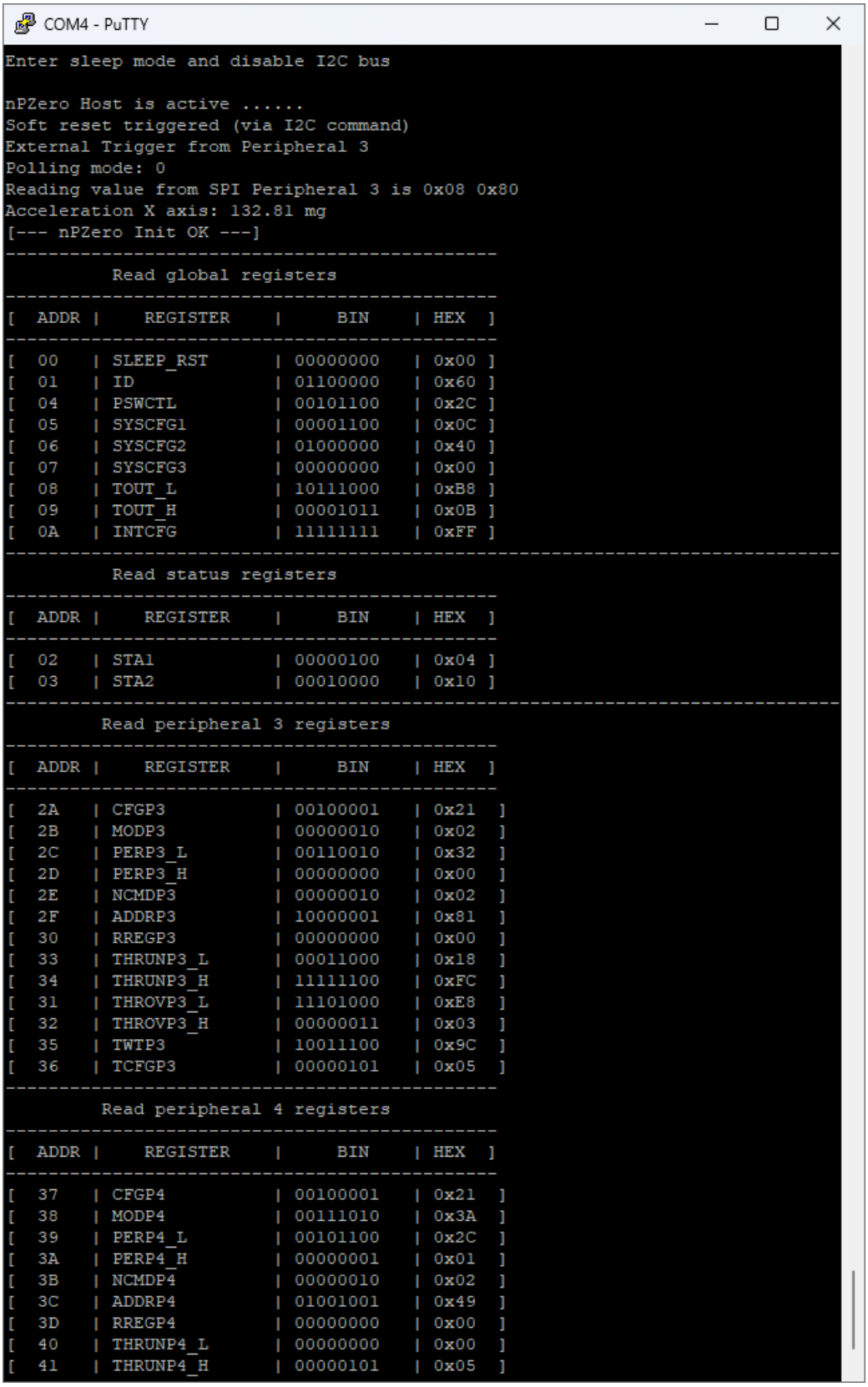


Figure 16: DevBoard recognized as COM4 displayed in PuTTY

4.4 Testing the pre-programmed example

Once the DevKit is assembled and the UART terminal is configured as described in the previous sections, reset the DevBoard and verify that the example is running as expected:

- After reset D2 should be lit, while the log output from the host MCU should be printed to the terminal as shown in Figure 15.
- After a couple of seconds the host MCU should be disconnected, and D2 turned off.
- The two sensors should be polled repeatedly, indicated by D5 and D6 blinking.
- After seconds to minutes the host MCU should be woken up again, repeating the process from the top.
- If one of the sensors were triggered, the respective temperature or accelerometer readings should be printed to the log.

4.5 Building and flashing the example code

This section details how to download, compile and flash the example from scratch, allowing you to change the code if needed.

This includes essential steps for developing software for the STM32 host MCU using the STM32CubeIDE, which is the Integrated Development Environment for the STM32 MCU. It is a comprehensive development platform that simplifies the MCU configuration and programming processes. Follow these steps to import the source code in the STM32CubeIDE:

1. Download STM32CubeIDE from https://www.st.com/en/development-tools/stm32cubeide.html
2. Open STM32CubeIDE.
3. Log in to your myST account by going to **Help > STM32Cube updates > Connection to myST**

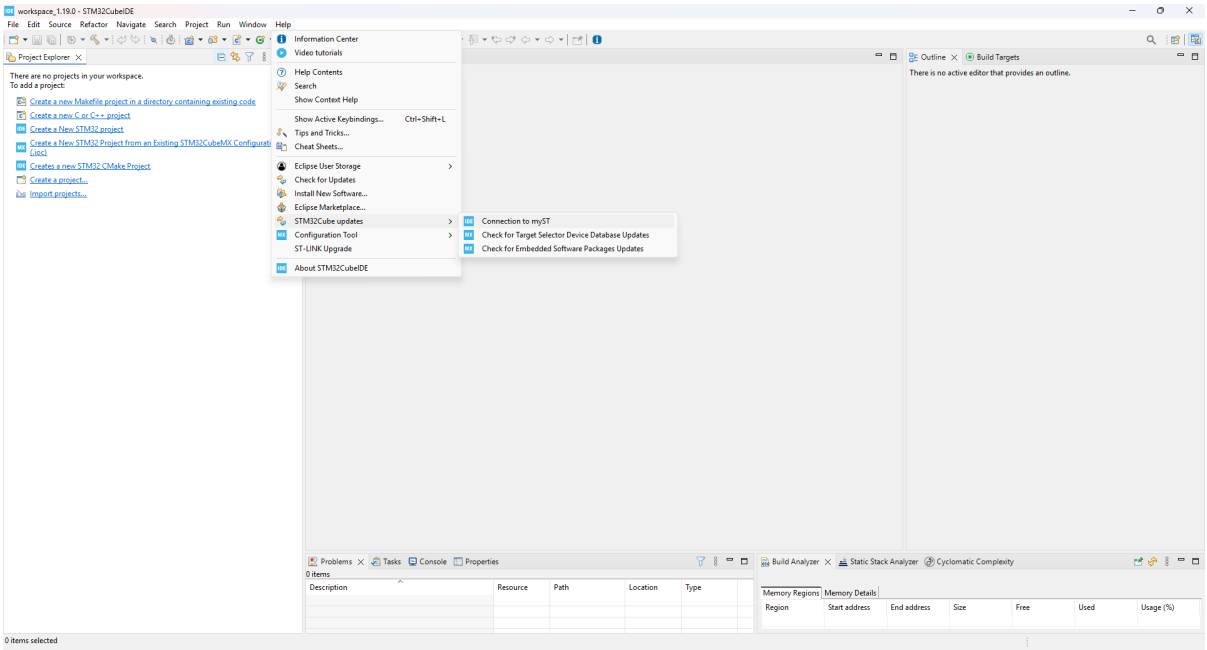


Figure 17: Entering ST account

4. Download the example code and nPZero Driver API from https://dev.nanopowersemi.com/nPZero_Driver_API.html

5. Open the project file shown in Figure 18 by double clicking it.

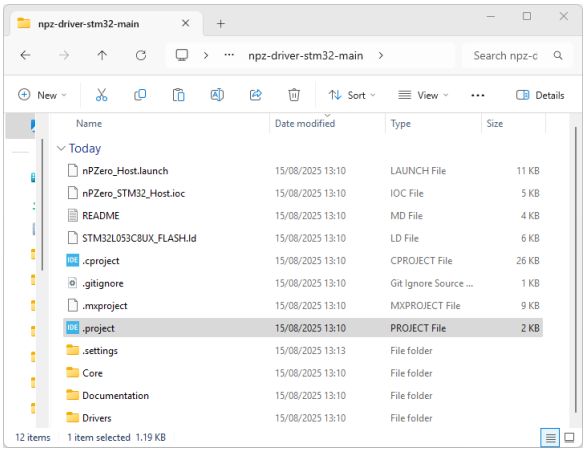


Figure 18: Opening the project file

6. To create a run configuration for an nPZero project, right-click on **nPZero_STM32** project name in Project Explorer view and select **Run As > STM32 C/C++ Application** as indicated in Figure 19.

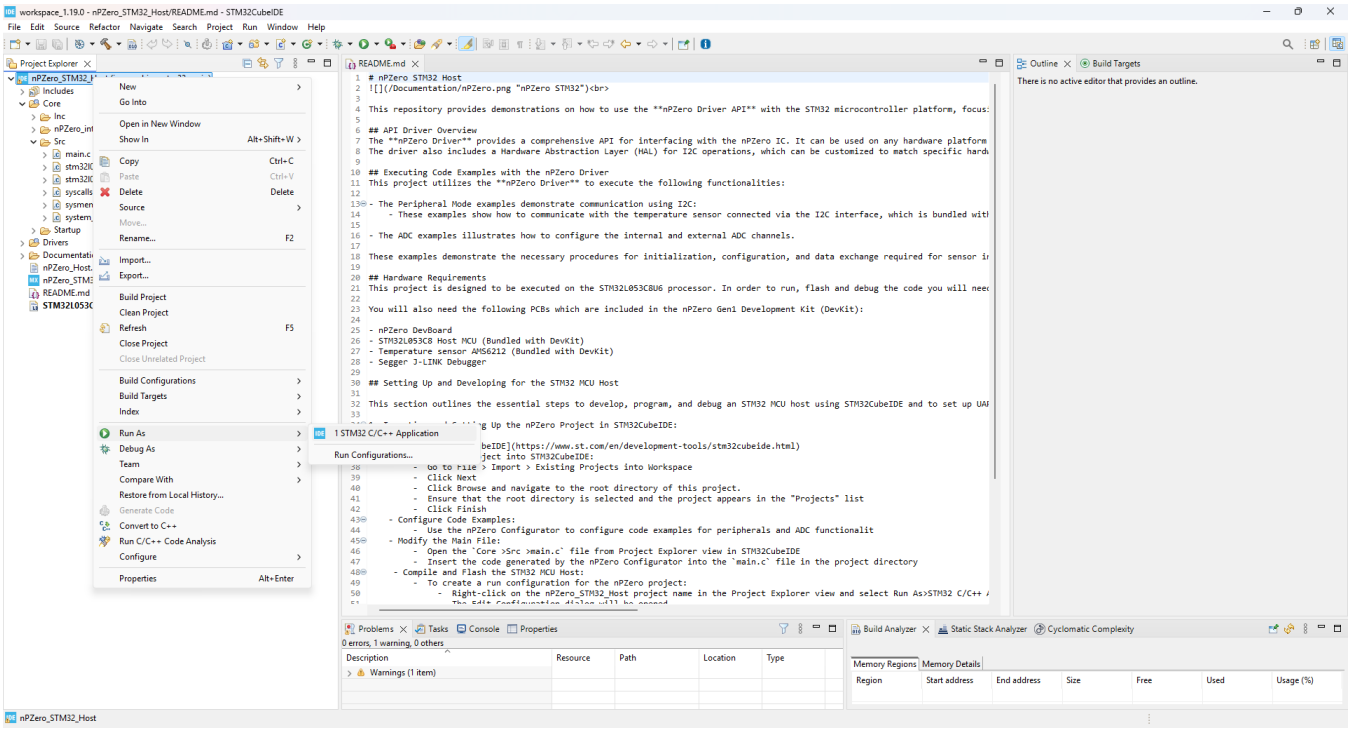


Figure 19: Run the code example in STM32

- **Edit Configuration** dialog will be opened.

7. Go to the **Debugger** tab and select **SEGGER J-LINK** as a debug probe as presented in Figure 20.

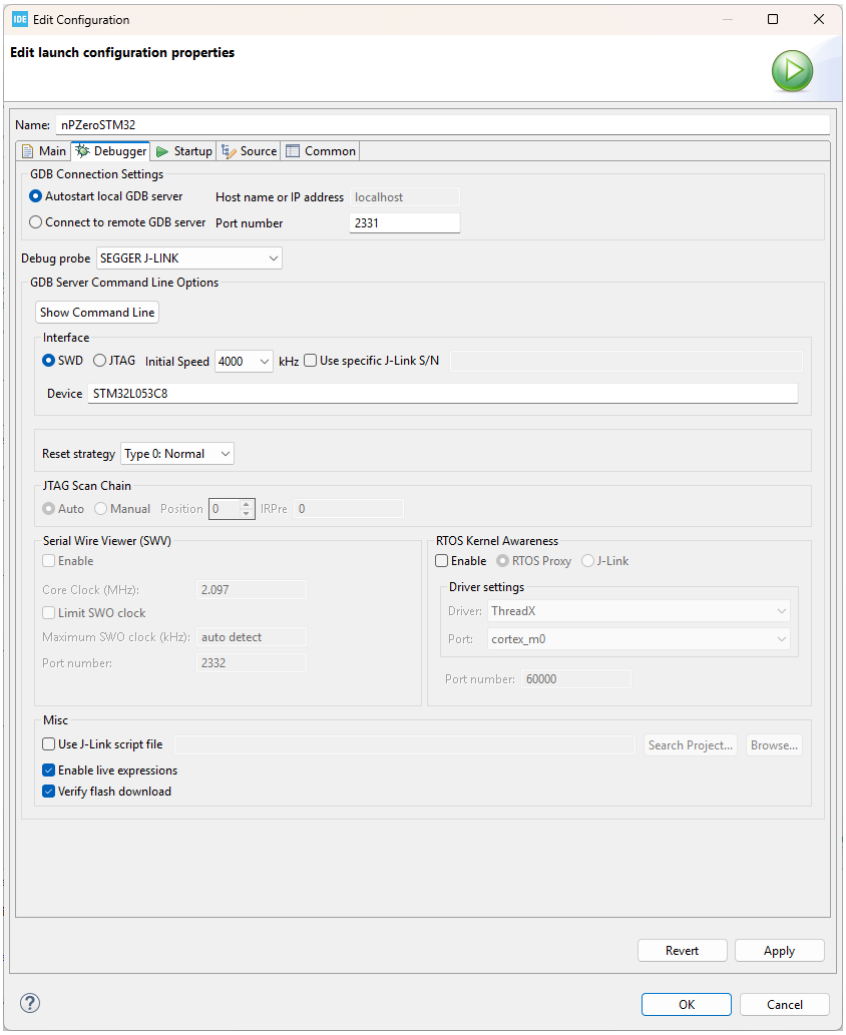


Figure 20: Setting up J-Link as debug probe

- Note that the host MCU needs to be powered when the STM32CUBE application is being flashed into the host, as indicated by the MCU **LED indicator D2** on the DevBoard. MCU can be powered off, if it has been disabled by the nPZero previously.

8. To ensure the host MCU is enabled, connect the testpoint **VBAT (TP30)** and **VCC (pin 6)** of the **PMOD MCU connector (J17)** to enable the host, as shown in Figure 21.
9. Just before flashing press the **RESET (S5)** button on the DevBoard to restart the nPZero and the host.
10. When the flash operation is completed, remove the connection between **VBAT (TP30)** and **VCC (pin 6)** of the **PMOD MCU connector (J17)**.

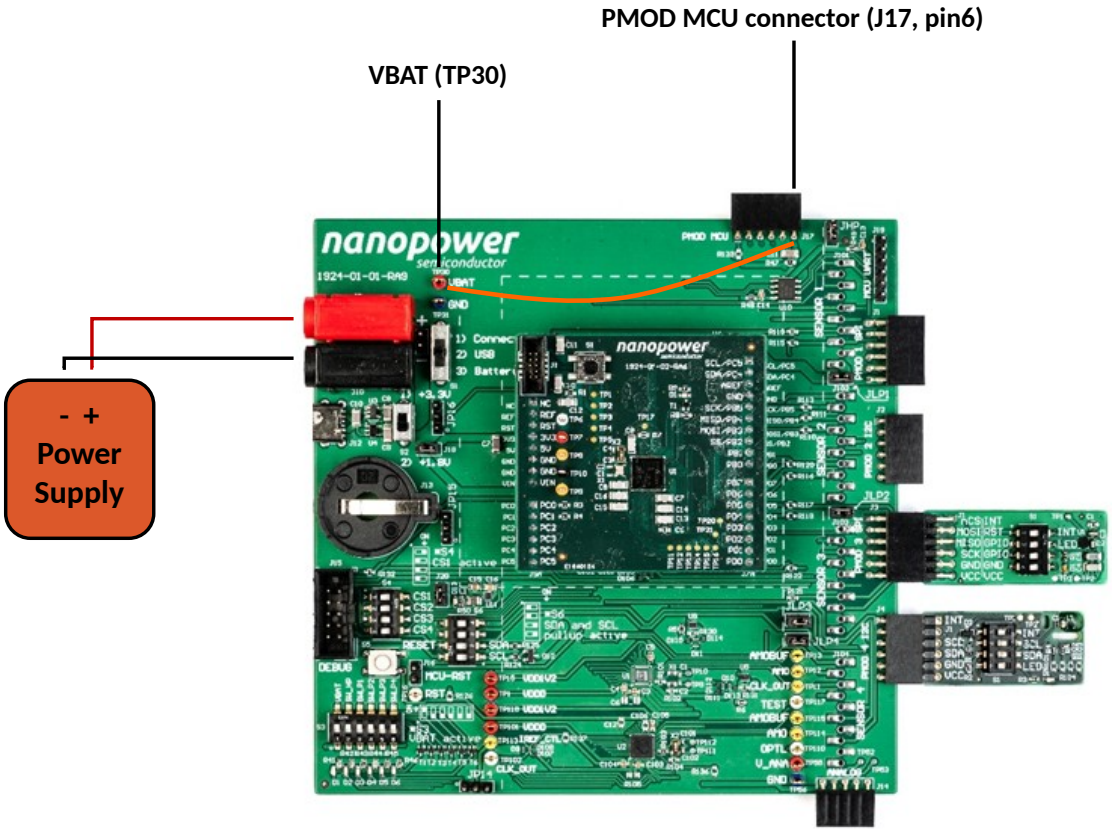


Figure 21: Enabling host MCU during flashing

5 Custom Project

To create a new custom project from scratch, please go through the steps described below. This chapter can also be used as a guide for adapting an existing project for use with the nPZero. In this case, simply skip the steps in *part 1. Project Setup* of the *'STM32 System settings' subsection* and start instead at *part 2. Configure I2C* of the same subsection.

5.1 STM32 System settings

1. Project Setup

    - To start a new project, go to **File > New > STM32 project**.

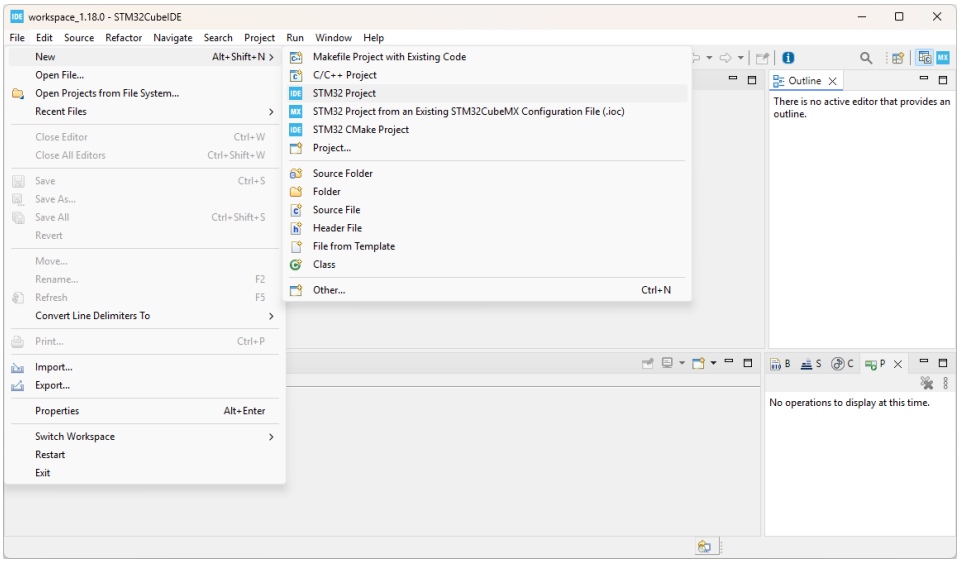


Figure 22: Creating a new STM32 project

    - Select the desired STM32 device or board. As demonstrated below, the selected board is **STM32L053C8U6**.

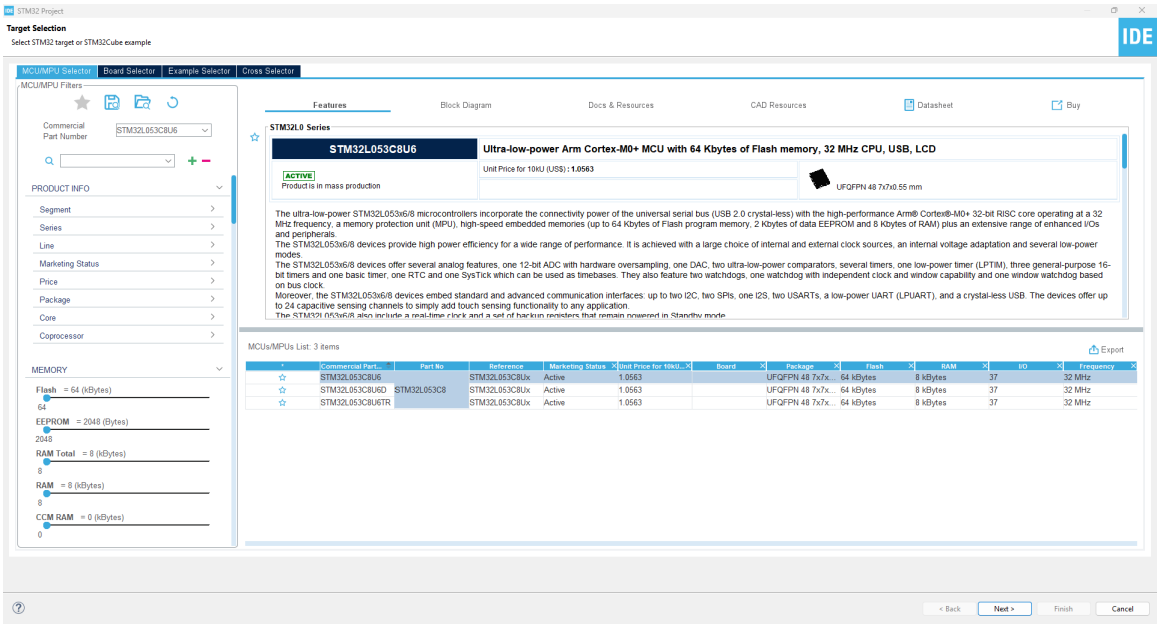


Figure 23: Target selector in STM32CubeIDE

- Click **Next**.
- Enter the project name to set up the project.

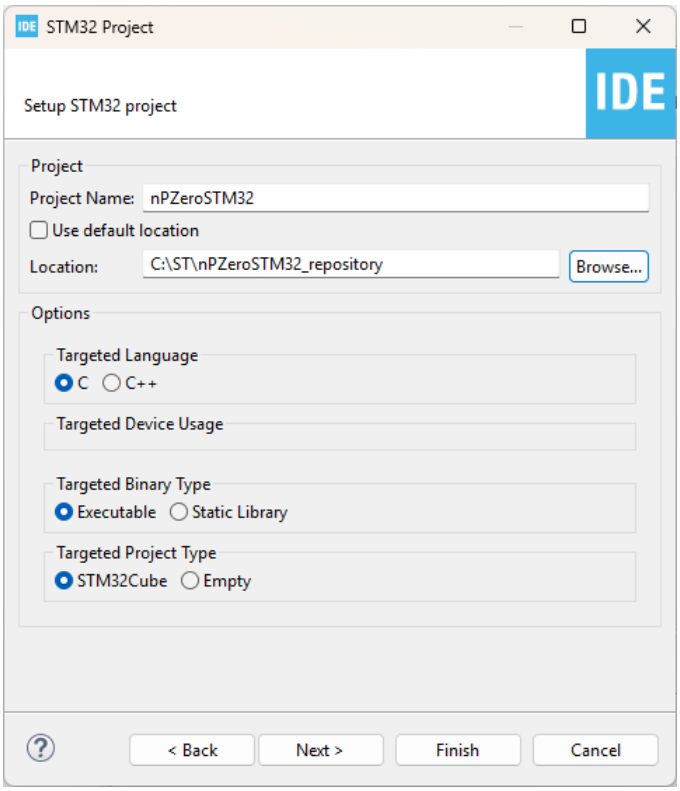


Figure 24: Project setup

- Click **Next** to open Firmware Library Package Setup page.

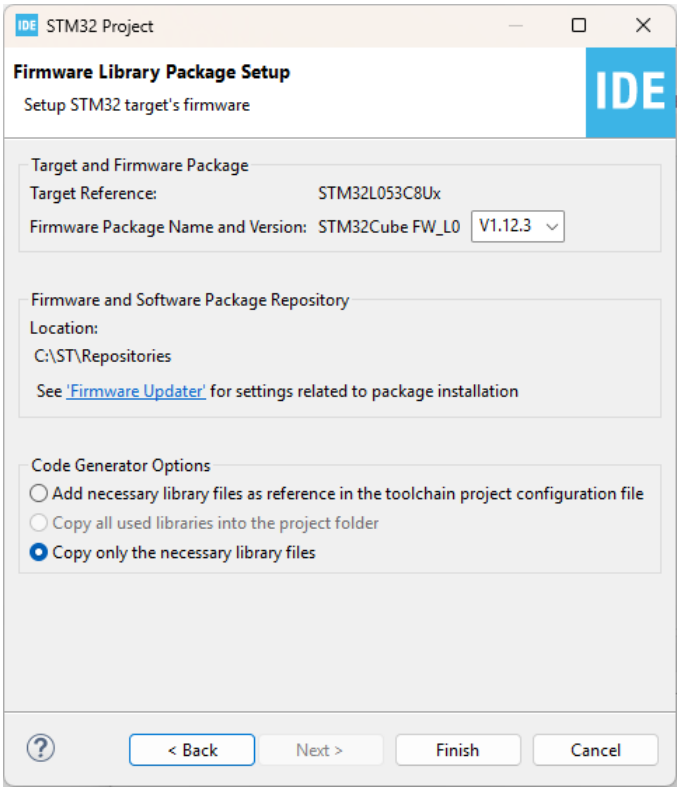


Figure 25: Firmware library package setup

- Click **Finish** to create the project.
- As a result, the following dialog is displayed. Click **Yes**.

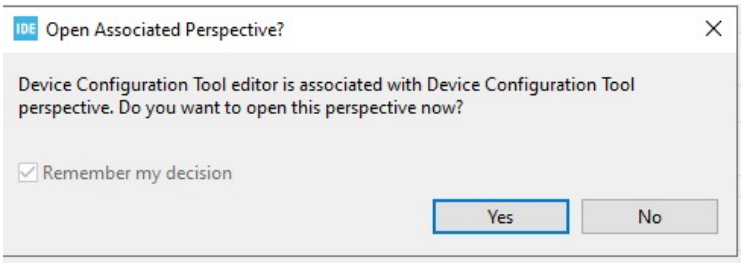


Figure 26: STM32CubeMX perspective opening

- When the project is created, the following STM32CubeMX perspective is opened with a window for configuring the peripherals.

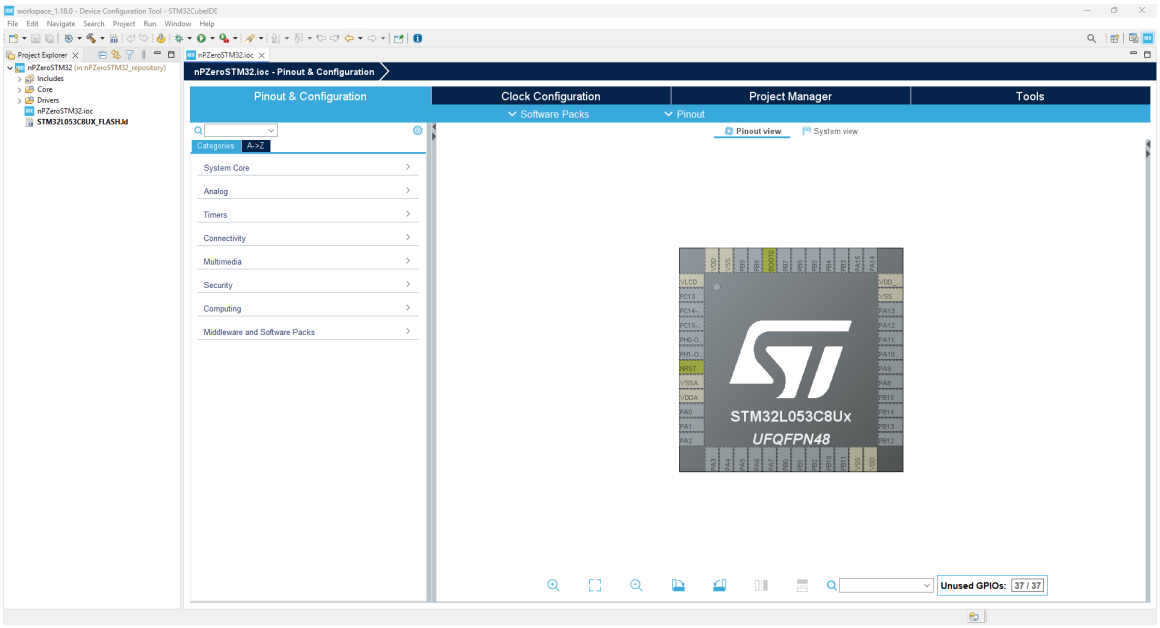


Figure 27: STM32CubeMX

2. Configure I2C

- Under the **Connectivity**, select **I2C1**.
- Choose **I2C** in **Mode** drop-down list.
- Set the I2C mode to I2C (**Standard Mode**) and configure the I2C frequency to **100 kHz** in **Parameter Settings** as depicted in Figure 28
- Choose **PB6** for SCL and **PB7** for SDA as the pins for **I2C1** in **GPIO Settings**.

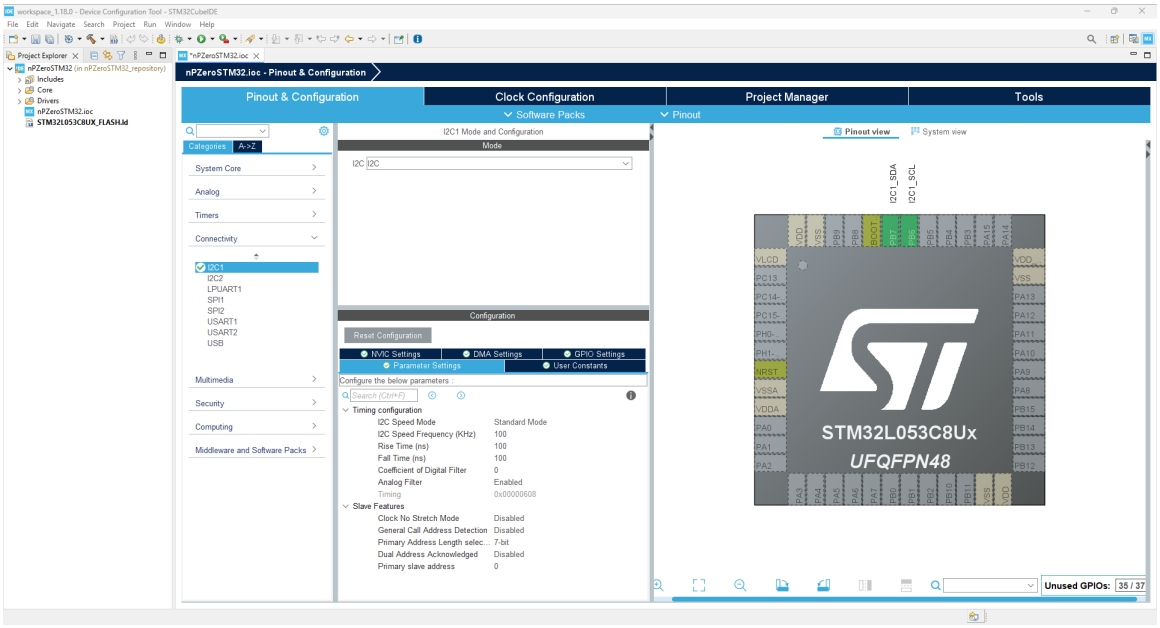


Figure 28: Configure I2C

3. Configure UART

   - Under **Connectivity**, select **USART2**.
   - Set **USART2** mode to **Asynchronous mode**.
   - Adjust the parameter settings as shown in Figure 29.

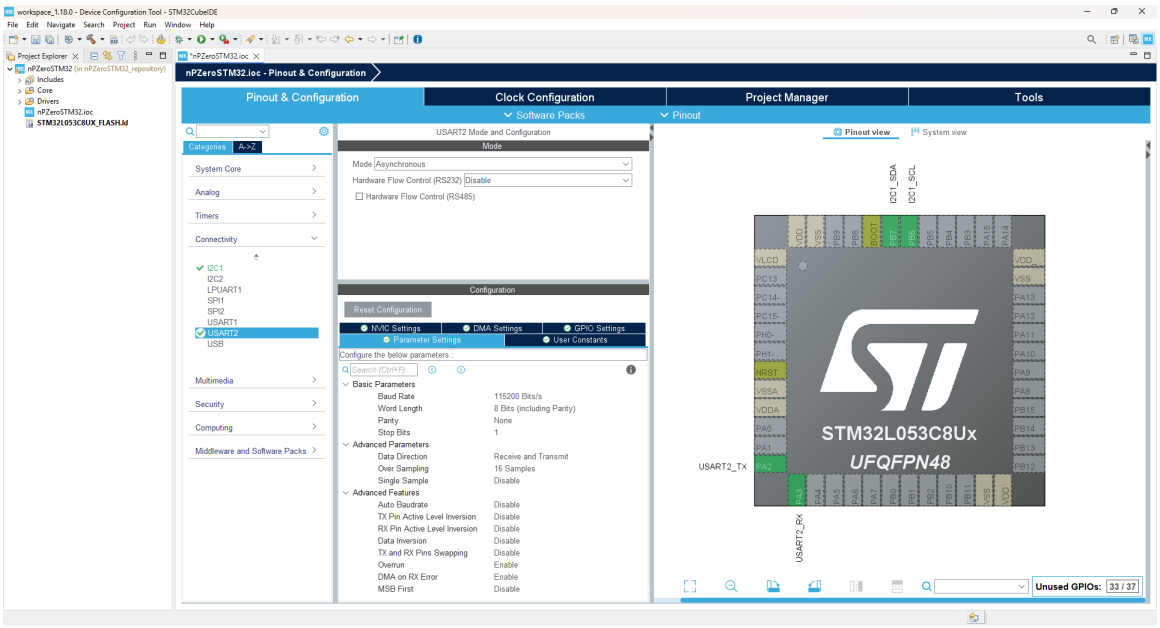


Figure 29: Configure UART

4. Configure GPIO LED toggling

    - Select pin PA12 and configure it as GPIO_Output to control the LED.

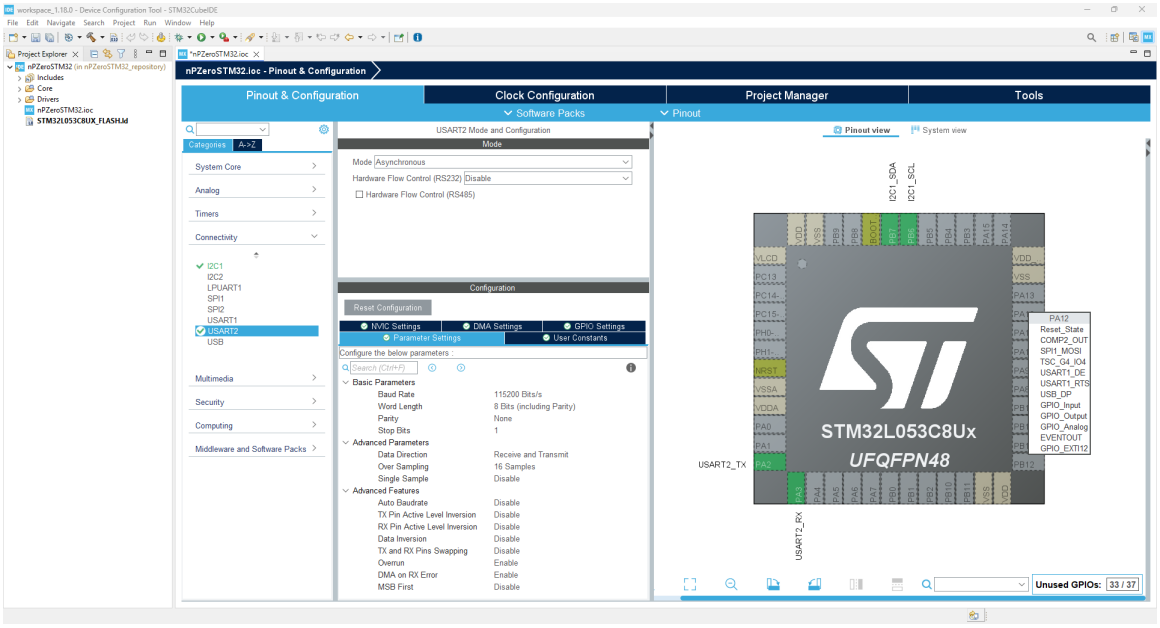


Figure 30: Selecting PA12 pin

- Under **System Core**, select **GPIO**.
- Adjust the GPIO configuration for **PA12** as shown in Figure 31.

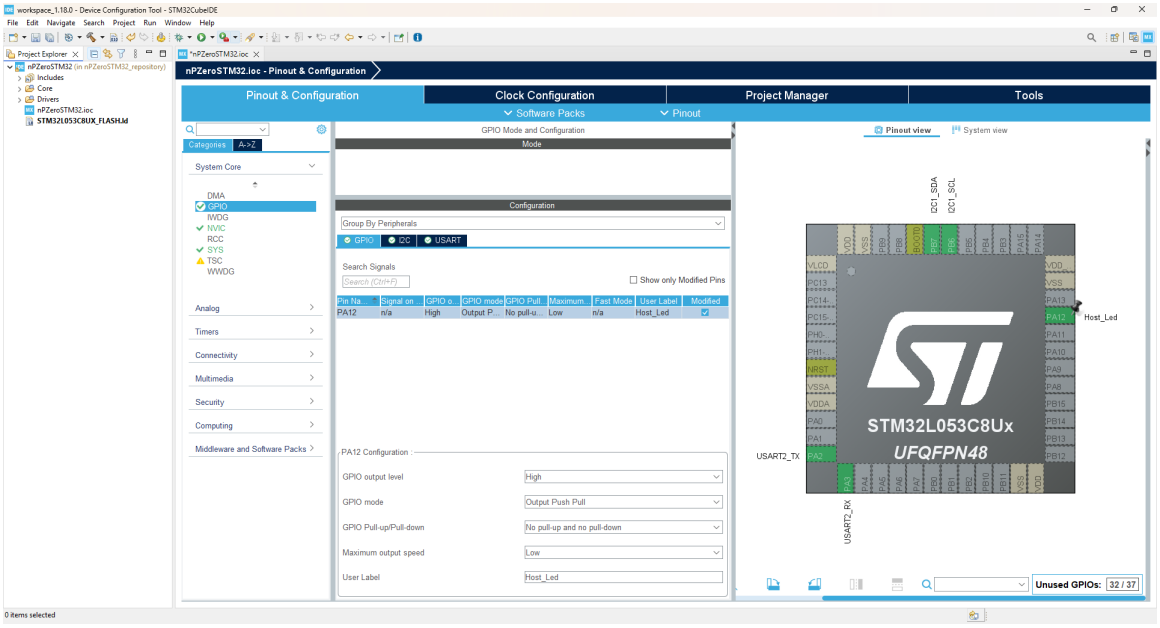


Figure 31: Configure GPIO for LED toggling

5. Press **Ctrl+S** to generate the project, and then click **Yes**.

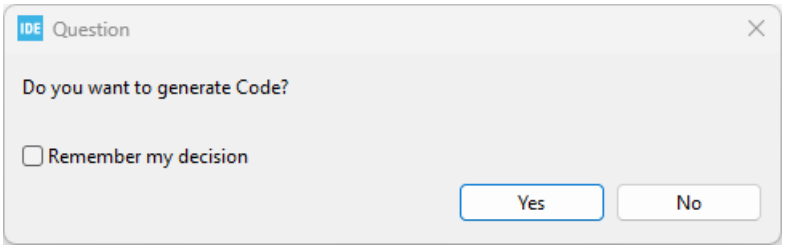


Figure 32: Generate source code

6. As a result, the following dialog is displayed. Click **Yes**.

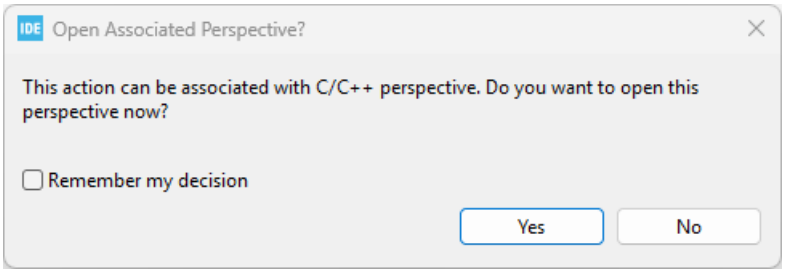


Figure 33: Open C/C++ perspective

7. Include the nPZero Driver API in the project. If the Driver is already downloaded, follow these steps:
   - Go to **File > Import > General > File System**, as illustrated in Figure 34.

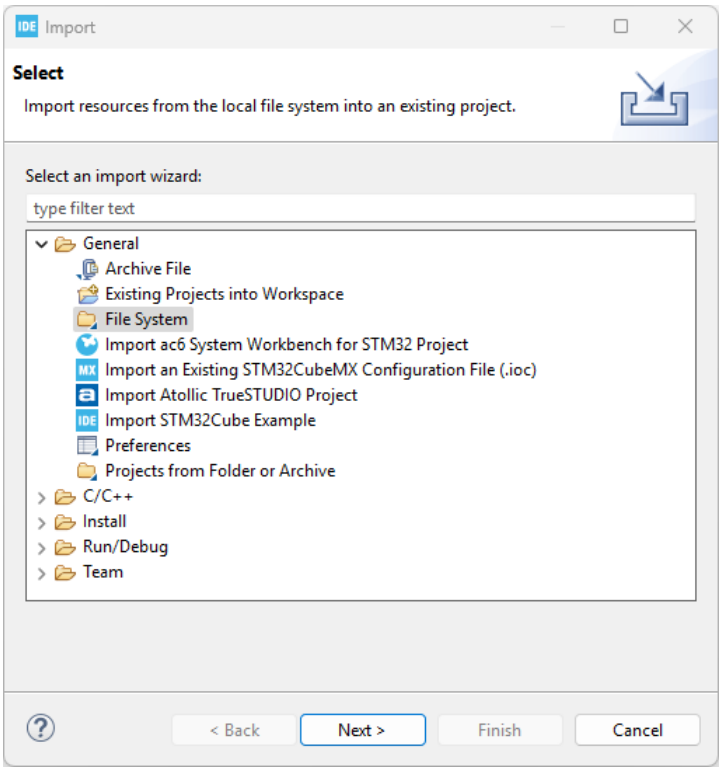


Figure 34: Import nPZero Driver API

   - Click **Next**.

- Import the nPZero Driver API resources from the local file system into the created project choosing **Drivers** folder in the project, as shown in Figure 35 and then click **Finish**.

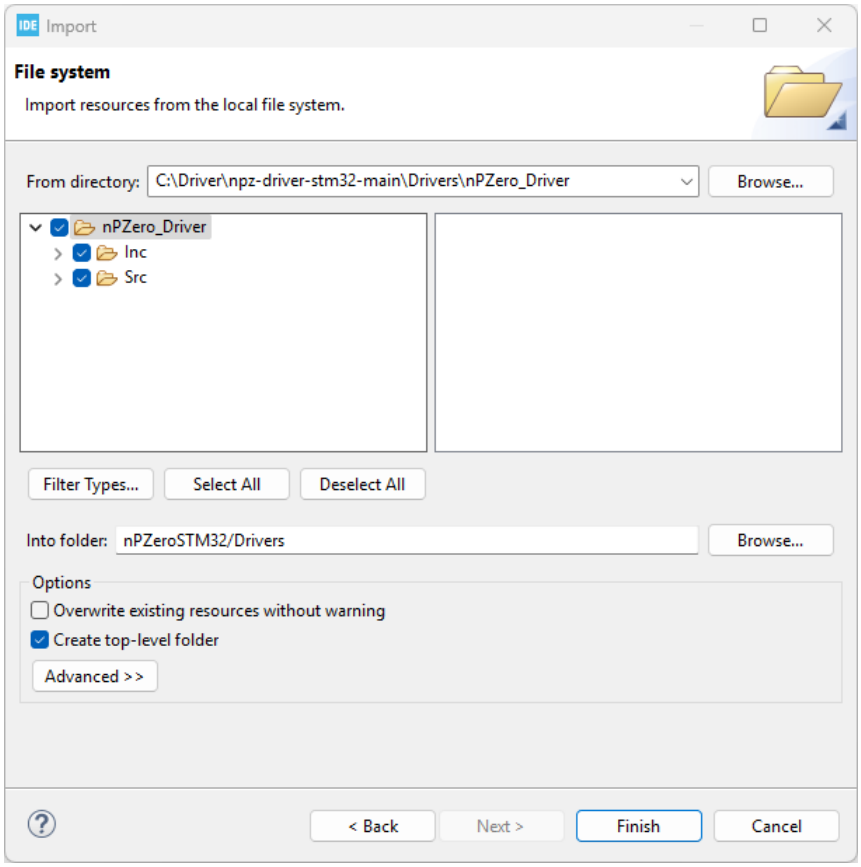


Figure 35: Import file system

8. Go to **Window > Show View > Other > Project Explorer** as shown in Figure 36 to see some of the project files and folders.

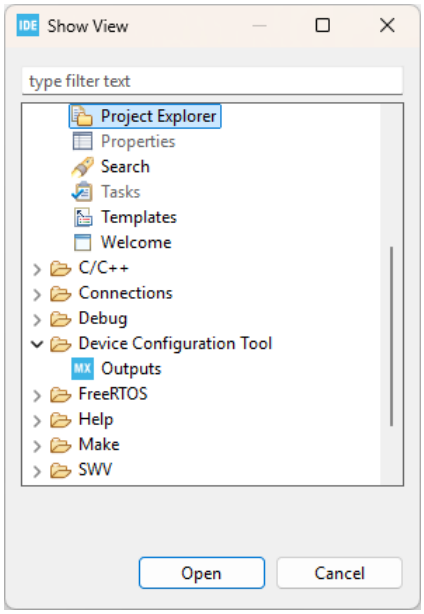


Figure 36: Project Explorer view

9. Now, the nPZero Driver API is present in **Project Explorer**, as shown in Figure 37.

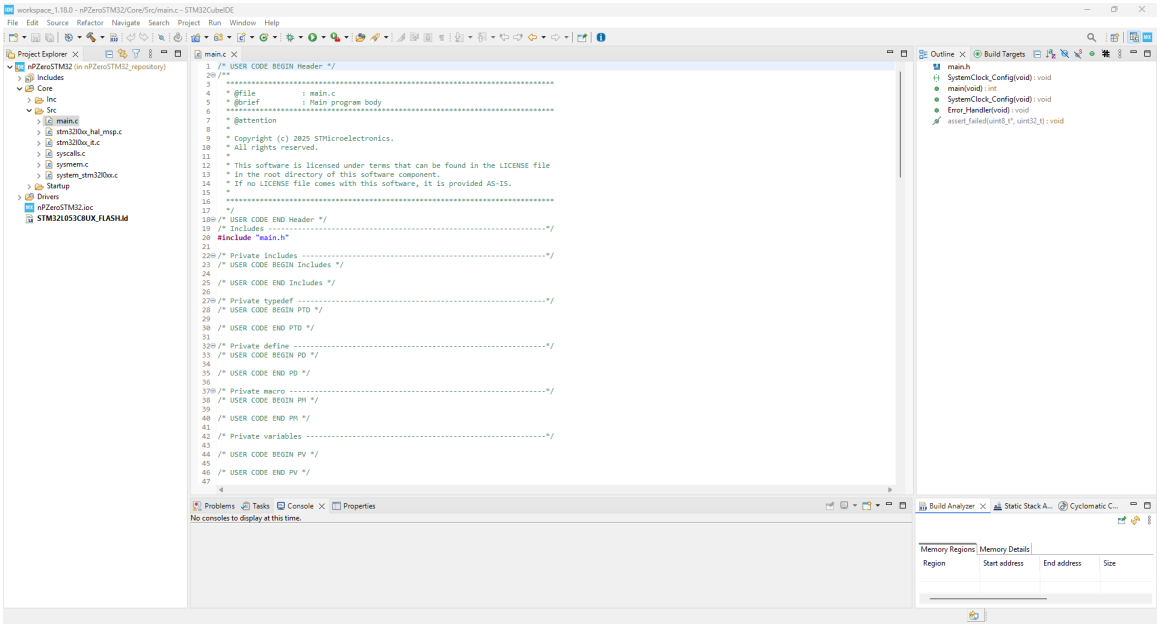


Figure 37: Project Explorer

5.2 nPZero IC configuration

1. Open the nPZero Configurator, in **Global Configuration** window set **Host power switch mode** to **Logic output (High when host enabled)** as shown in Figure 38.

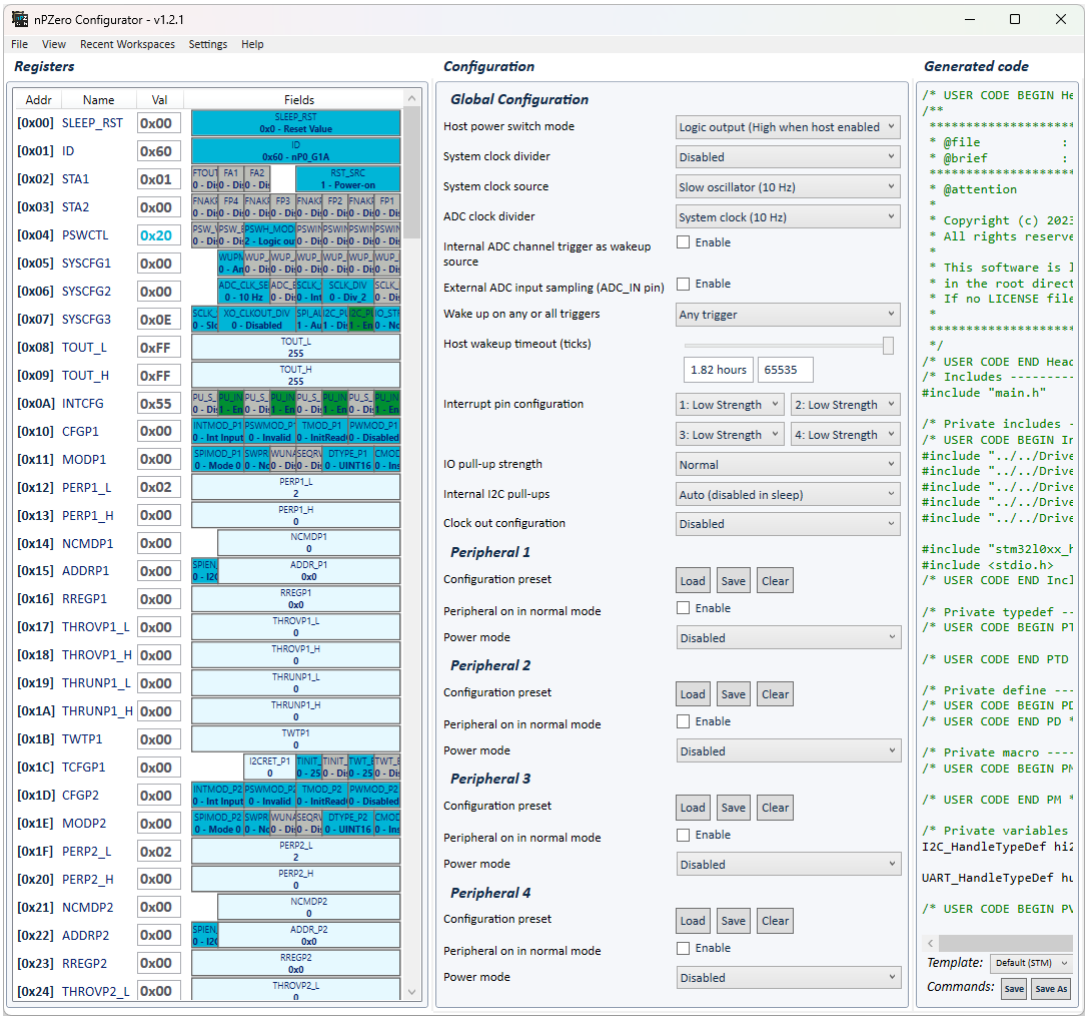


Figure 38: General configurations

2. Set configurations for the connected sensor in the appropriate peripheral section depending on the type of sensor.

   The accelerometer is configured as follows:

   - The accelerometer can be connected to the PMOD 1 or PMOD 3 SPI connector. So, for example, for Peripheral 3 configuration, go to **Load > LIS2DW > (SPI) Sample every second, wake on X axis > OK** as depicted in Figure 39.
   - Set **Polling period** for a desired value, e.g. 1 second.

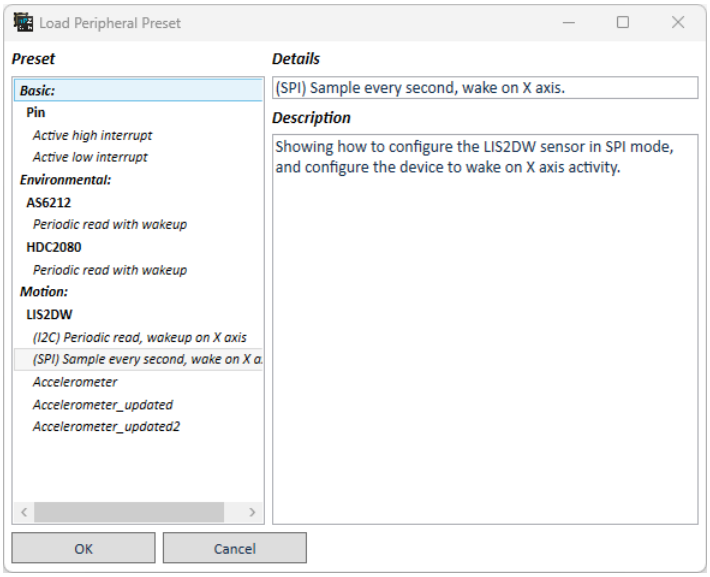


Figure 39: Loading Peripheral Preset for the accelerometer

The temperature sensor is configured as follows:

- In Global Configuration section set **Internal I2C pull-ups** as **Auto (disabled in sleep)**.
- The temperature sensor can be connected to the PMOD 2 or PMOD 4 I2C connector. So, for example, to configure Peripheral 4, go to **Load > AS6212 > Periodic read with wakeup > OK** as demonstrated in Figure 40.
- Set **Polling period** for a desired value, e.g. 2 seconds.
- Turn off switches 2 and 3 on the S1 DIP switch on the temperature sensor board.

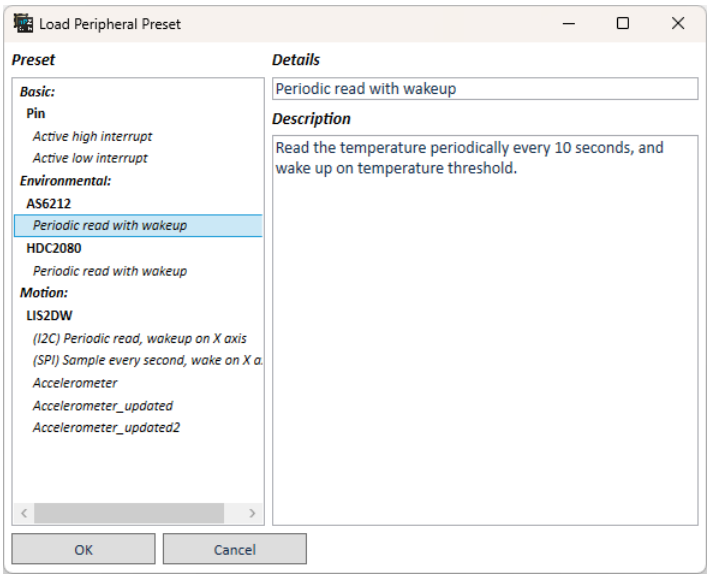


Figure 40: Loading Peripheral Preset for the temperature sensor

Applied configuration changes will be represented accordingly in **nPZero Registers** and **Generated code** windows as demonstrated in Figure 41 as an example.

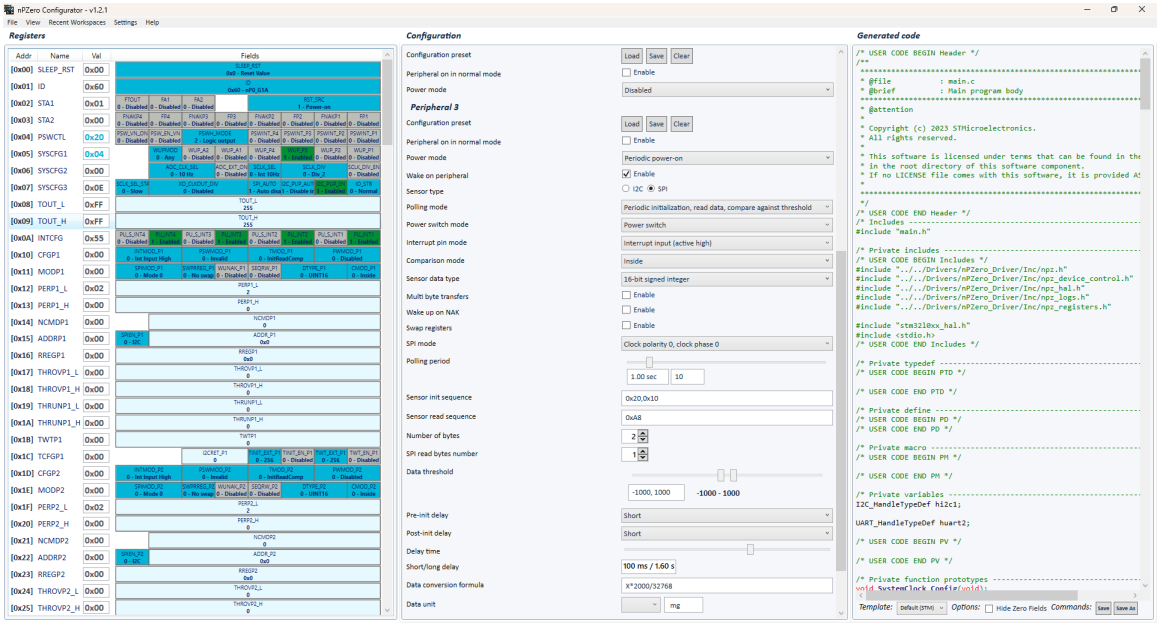


Figure 41: Configuring Peripheral connector for the accelerometer

- For the API implementation, make sure that **Template** for the generated code is set as **Default (STM)**.
- Press **Save as** and choose the same folder where STM32IDE project is located (**Project name > Core > Src**), replace the existing **main.c** file as shown in Figure 42.

3. After the saving **main.c** file, the custom project can be flashed in the same way as instructed in *subsection 4.5 'Building and flashing the example code'*.

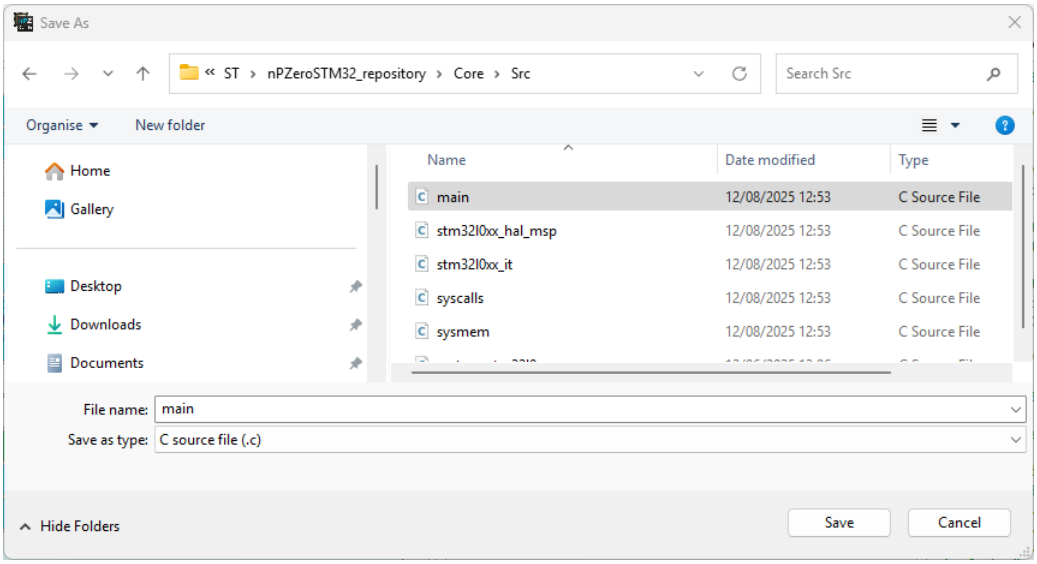


Figure 42: Saving main.c file

6 Health, Safety and Environmental Notices



Qualified personnel

The nPZero G1S DevKit is designed for use by qualified personnel only. A qualified person is defined as someone with the necessary education or experience to recognize potential hazards and take appropriate measures to minimize the risk of injury to themselves and others.



Power specifications

A PS1-class (IEC 62368-1) power supply delivering less 15 W is required to power the nPZero G1S DevKit.



Hot surface warning

During a fault condition, accessible surfaces may reach high temperatures.



Electrostatic discharge (ESD)

The nPZero G1S DevKit is sensitive to electrostatic discharge. To prevent damage, operate the device in an ESD-safe environment, such as a properly equipped laboratory.



Environmental care

Do not dispose of electronic waste in regular household bins. Please recycle at designated centers where available. For more information on recycling, contact your local authority or retailer.

7 Terms of Use

By using this documentation, you agree to comply with Nanopower Semiconductor's terms and conditions of use. Please note that these terms may be updated or modified at any time without prior notice.

7.1 Liability disclaimer

Nanopower Semiconductor AS reserves the right to make changes to its products at any time and without notice, to enhance reliability, functionality or design. The company assumes no liability for any issues arising from the application or use of any products or circuits described in this documentation.

Nanopower Semiconductor makes no representations or warranties, either express or implied, regarding the accuracy or completeness of the information contained herein. The company shall not be held liable for any consequences resulting from the use of such information. In cases of inconsistency, ambiguity, or conflict within Nanopower Semiconductor's documentation, the Datasheet shall take precedence. Additionally, Nanopower Semiconductor AS retains the right to make corrections, improvements, and other modifications to this document at any time and without prior notice.

7.2 Definitions and Acceptance

By using nPZero G1S Development Kit or related materials provided by Nanopower Semiconductor AS (collectively, the "Evaluation Board"), you ("You") agree to be bound by these Evaluation Board Terms of Use (the "Agreement"). If You do not agree, You must not use the Evaluation Board. Your use constitutes acceptance of this Agreement. If You are using the Evaluation Board on behalf of a company or other legal entity, You represent and warrant that You have the authority to bind such entity to this Agreement, and "You" shall refer to such entity.

7.3 Ownership and Intellectual Property

All right, title, and interest in and to the Evaluation Board, including all associated intellectual property rights, remain with Nanopower Semiconductor AS and/or its licensors. You may not remove any copyright or other proprietary rights notices on the Evaluation Board without prior written authorization from Nanopower. Except as expressly set forth in this Agreement, no license, express or implied, is granted under any patents, copyrights, trade secrets, or other intellectual property rights of Nanopower. Subject to the terms of this Agreement, Nanopower grants You a limited, non-exclusive, non-transferable, non-sublicensable, revocable license to use the Evaluation Board solely for evaluation, testing, and development purposes in accordance with subsection 7.4.

7.4 Purpose and Permitted Use

The Evaluation Board is provided solely for evaluation, testing, and development of Nanopower products in a laboratory or research and development environment. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation and testing purposes and has no direct function and is not a finished product. The Evaluation Board is not a finished product and is not intended for consumer, household, medical, safety-critical, military, aerospace, or automotive use unless expressly authorized in writing by Nanopower. The Evaluation Board is not qualified for reliability, safety, or regulatory compliance and has not undergone the same testing as production products.

7.5 Prohibited Uses

You shall not use the Evaluation Board in production systems, commercial products, or for resale. You shall not sell, lease, sublicense, distribute, modify, reverse engineer, decompile, or disassemble Nanopower Semiconductor AS Evaluation Board Terms of Use (April 2026) Page 2 of 3 the Evaluation Board or any related software or firmware, except to the extent such restriction is prohibited by applicable law.

7.6 Safety Responsibilities

You acknowledge that the Evaluation Board may present electrical, mechanical, or other hazards. You are solely responsible for safe handling, use, and compliance with applicable safety standards. Nanopower shall have no responsibility for damages resulting from improper or unsafe use.

7.7 Software and Firmware

Any software or firmware provided with the Evaluation Board is licensed, not sold, on a non-exclusive, non-transferable, and non-sublicensable basis solely for use with the Evaluation Board. All software and firmware are provided ”AS IS” without warranties of any kind.

7.8 Regulatory Compliance

You are responsible for ensuring compliance with all applicable laws, regulations, and directives, including but not limited to EMC, radio, and environmental regulations. The Evaluation Board may not be certified for regulatory compliance. You shall not use, export, re-export, or transfer the Evaluation Board, directly or indirectly, in violation of any applicable export control or sanctions laws or regulations, including those of the European Union, Norway, and the United States. Without limitation, You shall not use the Evaluation Board in connection with the development, production, or use of nuclear, chemical, or biological weapons, missile systems, or in any other prohibited end-use, nor provide access to any restricted or sanctioned persons or entities.

7.9 Warranty Disclaimer

Nanopower warrants only that it has the right to make the Evaluation Board available. TO THE MAXIMUM EXTENT PERMITTED BY LAW, THE EVALUATION BOARD IS PROVIDED ”AS IS” AND ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, ARE DISCLAIMED. YOU WARRANT THAT YOU UNDERSTAND THE DANGERS OF HANDLING AND USING ELECTRICAL MECHANICAL COMPONENTS, SYSTEMS, AND SUBSYSTEMS. YOU ASSUME ALL RESPONSIBILITY AND LIABILITY FOR ANY IMPROPER OR UNSAFE HANDLING OR USE OF THE EVALUATION BOARD BY YOU, YOUR EMPLOYEES, AFFILIATES, CONTRACTORS, AND DESIGNEES.

7.10 Limitation of Liability

To the maximum extent permitted by law, Nanopower shall not be liable for any indirect, incidental, special, or consequential damages, including loss of profits, revenue, or data. Nanopower’s total aggregate liability arising out of this Agreement shall not exceed the amount paid for the Evaluation Board, or EUR 10, whichever is higher.

7.11 Indemnification

Nanopower Semiconductor AS Evaluation Board Terms of Use (April 2026) Page 3 of 3 You agree to defend, indemnify, and hold harmless Nanopower Semiconductor AS and its affiliates from any claims, damages, losses, or expenses arising from Your use of the Evaluation Board in violation of this Agreement.

7.12 Governing Law and Jurisdiction

This Agreement shall be governed by and construed in accordance with the laws of Norway, without regard to its conflict of law principles. The courts of Norway shall have exclusive jurisdiction.

7.13 Severability

If any provision of this Agreement is held unenforceable, the remaining provisions shall remain in full force and effect.

7.14 Waiver

Any failure to enforce a provision of this Agreement shall not constitute a waiver of future enforcement.

7.15 Independent Contractors

Nothing in this Agreement creates any partnership, joint venture, agency, or employment relationship between the parties.

7.16 Survival

Provisions which by their nature should survive termination shall survive.

7.17 Waste and Recycling (WEEE)

Nanopower Semiconductor fulfills its obligations as a producer under applicable EU WEEE legislation. End users are responsible for disposing of the Evaluation Board through appropriate collection and recycling channels and shall not dispose of it as unsorted municipal waste.

Tollbodgata 2 | 4611 Kristiansand | Norway
info@nanopowersemi.com | Telephone +47 989 02 050

European Innovation Council

Co-funded by the European Union

© Nanopower Semiconductor AS – all rights reserved. For more information, visit **www.nanopowersemi.com**
This document is subject to change without further notification.