Zephyr Setup - GitHub Release

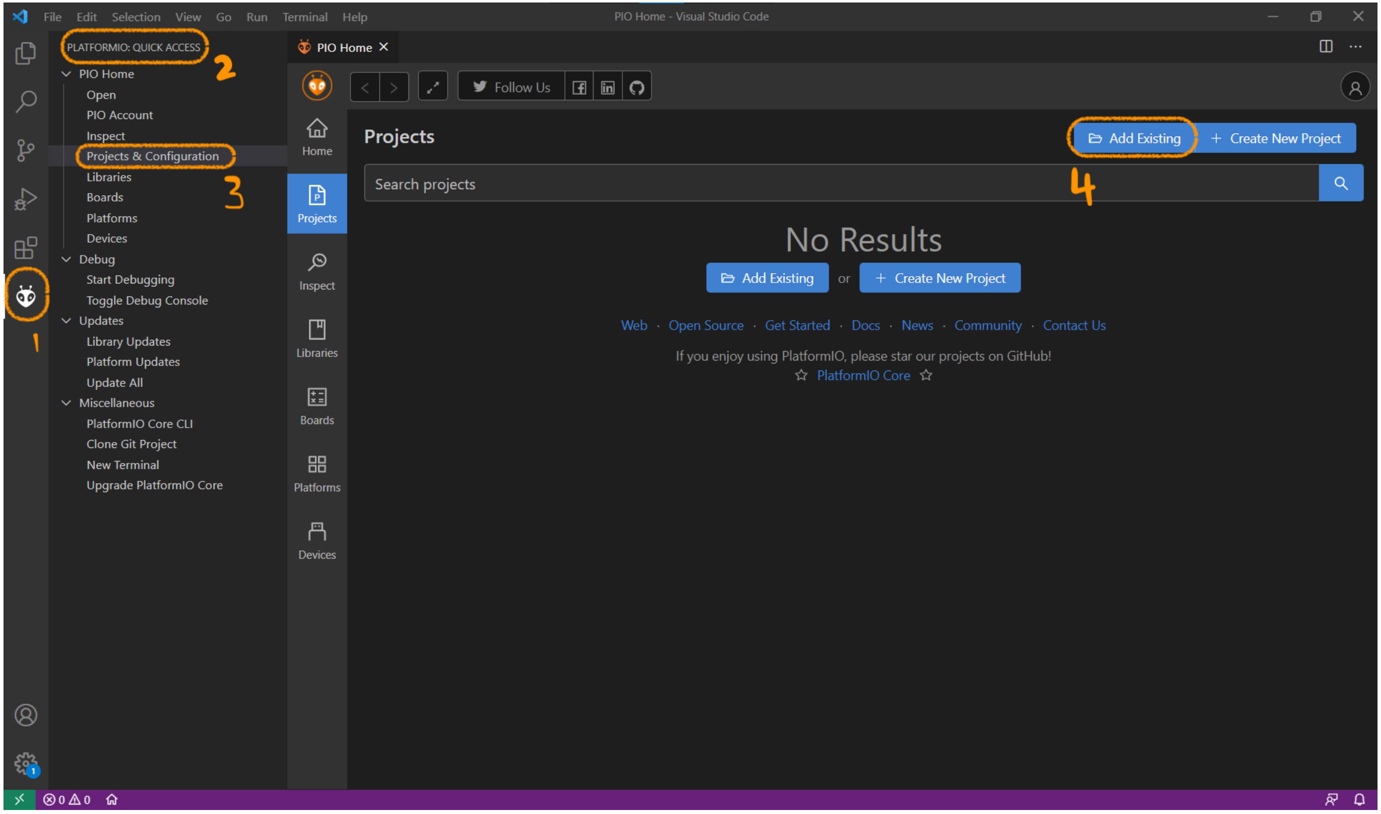
The repository uses PlatformIO to bootstrap Zephyr for ease of development and deployment of software.

1.0 Environment Requirements

- **Development Machine Operating System:** Windows, macOS or Ubuntu
    - **clang-format** *(optional; for 2.1)*: Either through snapshot builds or the downloads page.
        - For MacOS users: `$ brew install clang-format`.
    - For Windows, a few external dependencies have to be installed:
        - **libUSB**: Follow the instructions here; MinGW64 .dll will work fine for x64 systems.
- **IDE:** VSCode
    - Extensions
        - C/C++ (ms-vscode.cpptools)
        - PlatformIO IDE (platformio.platformio-ide).
            - PlatformIO provides an easy way of bootstrapping the Zephyr RTOS and has built-in support for various tools
        - DeviceTree for the Zephyr Project (trond-snekvik.devicetree)
        - Clang-Format (xaver.clang-format)
            - Requires clang-format for enforcing consistent code styling
        - Visual Studio IntelliCode (visualstudioexptteam.vscodeintellicode).
            - *Optional - for code auto-completion.*
        - Git History (donjayamanne.githistory)
            - *Optional - for visualization of git changes*
        - CMake Tools (ms-vscode.cmake-tools)
            - *Optional - manually create/modify cmake files*

2.0 Importing the Repository

On the sidebar, navigate to **PlatformIO > Quick Access > Projects & Configuration > Add Existing**



Navigate your directory and select `decada-embedded-example-zephyr`.

2.1 Extension Configuration (Optional)

It is recommended to enable automatic formatting of the source code.
With **Ctrl+Shift+P**, enter **Preferences: Open Workspace Settings**.

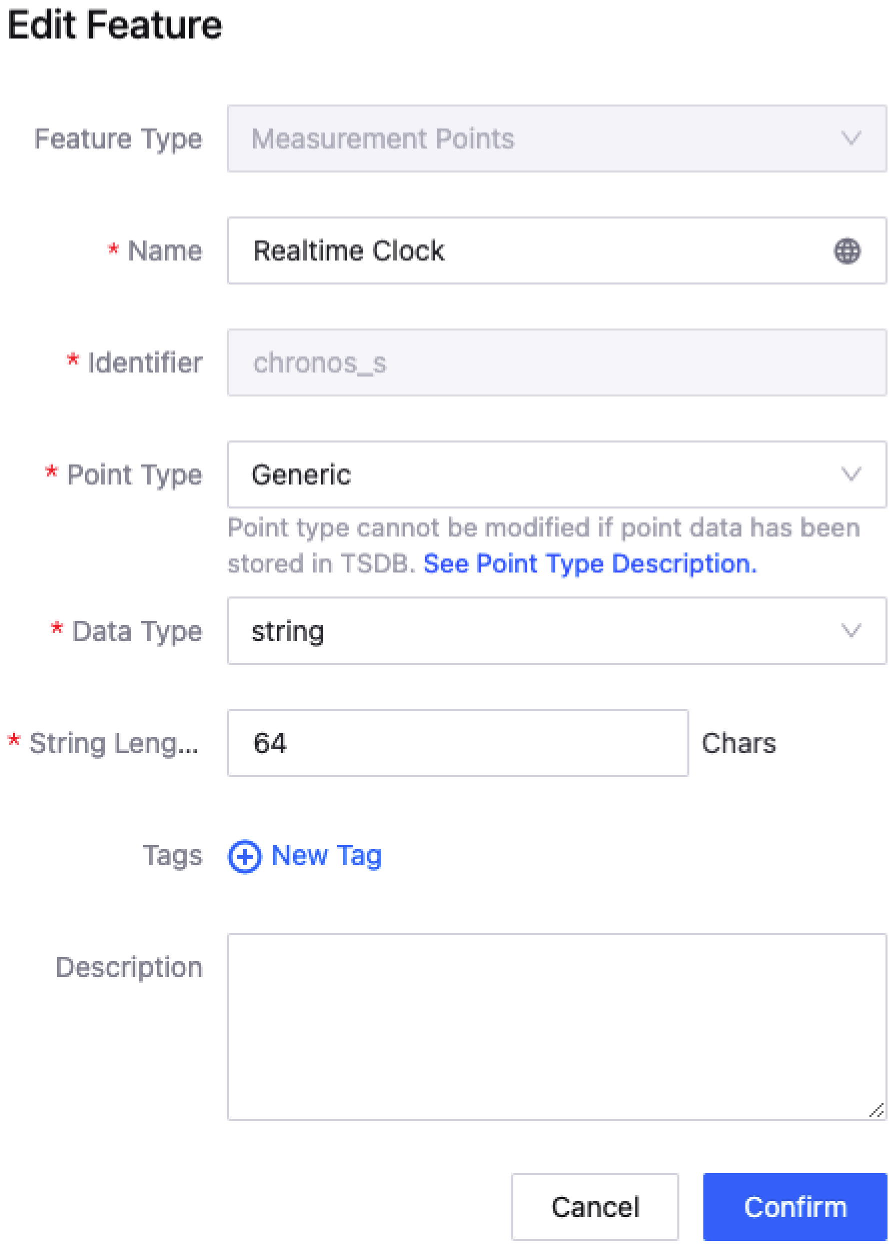
1. Search for **C_Cpp: Formatting** and select **clangFormat**
2. Search for **Editor: Format On Save** and select the checkbox

** Note that `clang-format` must already be installed (see 1.0).

3.0 Setting up your Cloud Service

Refer to https://www.siot.gov.sg/starter-kit/set-up-your-cloud-service/.

When creating your Measurement Point(s), note that the data identifier used in this example is `chronos_s` of String Data Type as shown in the example below:



Edit Feature

Feature Type: Measurement Points

* Name: Realtime Clock

* Identifier: chronos_s

* Point Type: Generic

Point type cannot be modified if point data has been stored in TSDB. See Point Type Description.

* Data Type: string

* String Leng...: 64 Chars

Tags: New Tag

Description:

4.0 Changing WiFi details and DECADA Credentials

Navigate to `/src/user_config.h`

Populate your WiFi credentials:

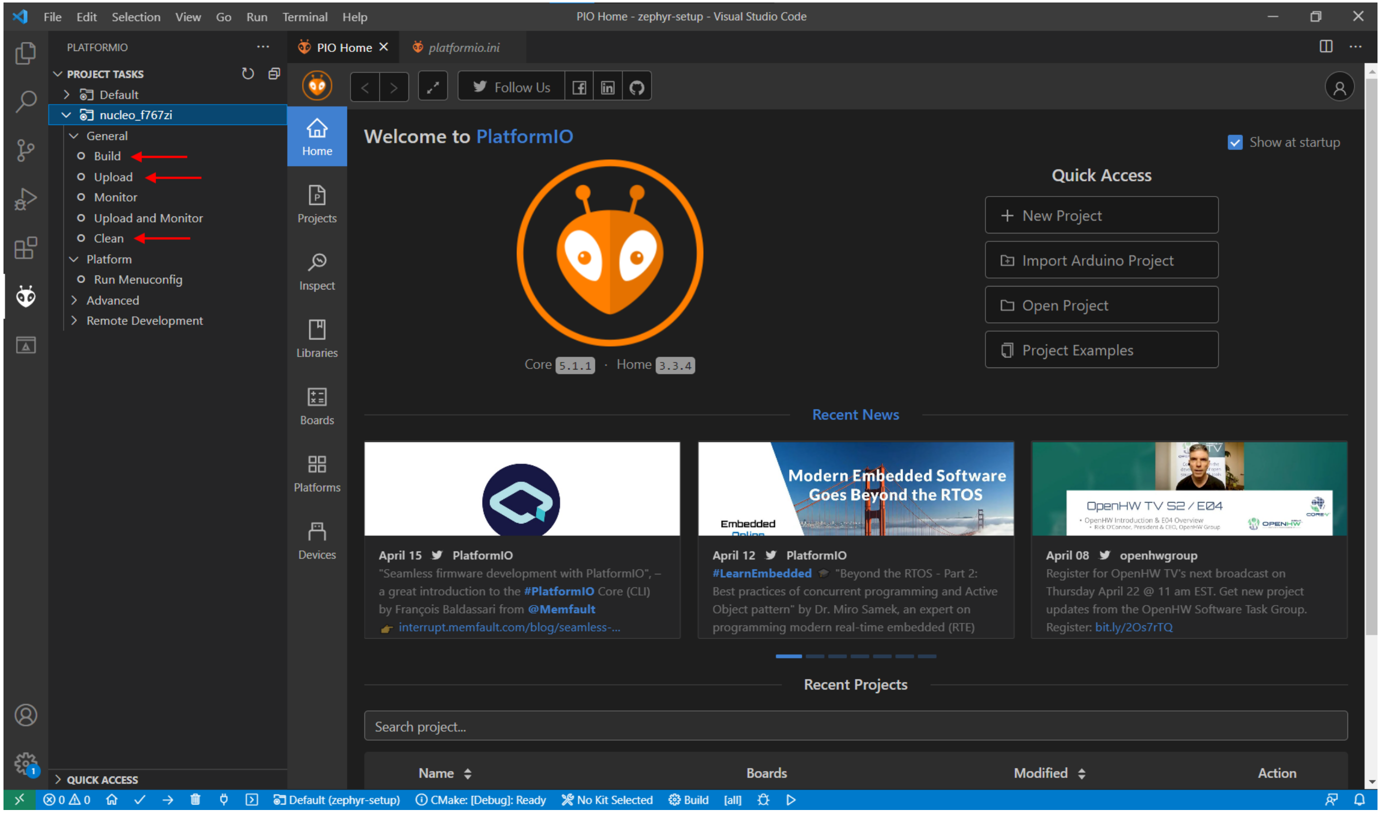
```
/**
 *      WiFi Details
 */

// Maximum of 32 characters
#define USER_CONFIG_WIFI_SSID \
        ("MyHomeWiFiNetwork")

// Between 8 - 64 characters
#define USER_CONFIG_WIFI_PASS \
        ("S3cretP4ssword*")
```

Populate your DECADA credentials:

```
/**
 *      DECADA Credentials
 */

// Organization Unit ID for DECADA cloud
#define USER_CONFIG_DECADA_OU_ID \
        ("enter_organization_unit_id_here")

// Access key for DECADA cloud application
#define USER_CONFIG_DECADA_ACCESS_KEY \
        ("enter_access_key_here")

// Access secret for DECADA cloud application
#define USER_CONFIG_DECADA_ACCESS_SECRET \
        ("enter_access_secret_here")

// Product key for DECADA cloud product
#define USER_CONFIG_DECADA_PRODUCT_KEY \
        ("enter_product_key_here")

// Product secret for DECADA cloud product
#define USER_CONFIG_DECADA_PRODUCT_SECRET \
        ("enter_product_secret_here")
```

5.0 Building and Flashing

On the sidebar, navigate to **PIO > Project Tasks > General** to build and flash (Upload) the binary. When a build problem arises due to configuration errors, it is advisable to clean the project and rebuild.



April 15 PlatformIO
"Seamless firmware development with PlatformIO", – a great introduction to the #PlatformIO Core (CLI) by François Baldassari from @Memfault interrupt.memfault.com/blog/seamless-...

April 12 PlatformIO
#LearnEmbedded "Beyond the RTOS - Part 2: Best practices of concurrent programming and Active Object pattern" by Dr. Miro Samek, an expert on programming modern real-time embedded (RTE)

April 08 openhwgroup
Register for OpenHW TV's next broadcast on Thursday April 22 @ 11 am EST. Get new project updates from the OpenHW Software Task Group. Register: bit.ly/2Os7rTQ

6.0 Custom Boards

To include your custom boards, additional board definitions have to be added.

6.1 Linking the Custom Board from Zephyr to PlatformIO

Under the project root, create a new `<BOARD_NAME>.json` file in `/boards`. This allows PlatformIO to recognize the target in order to build and flash properly.

- Example: `./boards/manuca_dk_revc.json`
- See Custom Embedded Boards for more details

6.2 Setting Up The Zephyr Custom Board Configs

Under `/zephyr`, create the folders `boards/<BOARD_ARCH>/<BOARD_NAME>`.
For reference, refer to the Zephyr repository. The files can be copied from a board that uses the same SoC and renamed (e.g., the MANUCA DK uses the STM32F767ZI, so custom files can be referenced from `boards/arm/nucleo_f767zi`).

- Example:
```
./zephyr/boards/arm/manuca_dk_revc/board.cmake
./zephyr/boards/arm/manuca_dk_revc/Kconfig.board
./zephyr/boards/arm/manuca_dk_revc/Kconfig.defconfig
./zephyr/boards/arm/manuca_dk_revc/manuca_dk_revc_defconfig
./zephyr/boards/arm/manuca_dk_revc/manuca_dk_revc.dts
./zephyr/boards/arm/manuca_dk_revc/manuca_dk_revc.yaml
./zephyr/boards/arm/manuca_dk_revc/support/openocd.cfg
```
- The `[BOARD_NAME].dts` file will require changes to reflect pin mapping on the custom board.

6.3 Setting PlatformIO Build Environment

Add the following lines the `platformio.ini` file to reflect a new environment for the target:

```
[env:<BOARD NAME>]
platform = <DEV_PLATFORM>
board = <BOARD_NAME>
framework = zephyr
build_type = debug
upload_protocol = stlink
debug_tool = stlink
platform_packages =
    toolchain-gccarmnoneeabi@<MAJOR.MINOR.BUGFIX>
```

- The list of development platforms and their configurations can be found here: <DEV_PLATFORM>
- Developers can refer to this page to see what toolchains are supported by PlatformIO

A complete example looks as such:

```
[env:manuca_dk_revc]
platform = ststm32
board = manuca_dk_revc
framework = zephyr
build_type = debug
upload_protocol = stlink
debug_tool = stlink
platform_packages =
    toolchain-gccarmnoneeabi@1.90301.200702
```

- **gnu-arm-none-eabi Release 9-2020-q2-update** is used as the toolchain for the GitHub example code.
- The GCC-ARM version is freeze in the example code to provide uniform development experience.

7.0 License

This document is prepared by the Sensors & IoT Capability Centre at GovTech Singapore, and follows the Apache-2.0 License that is attached to the repository (GovTechSIOT/decada-embedded-example-zephyr).