# MVPI Zephyr setup README

## Aloxy MVPI Manual Valve Position Indicator using Zephyr

This repository contains the Application for the MVPI Pulse.

The structure of this repository is modeled after the Zephyr example application

### Getting Started

Before getting started, make sure you have a proper Zephyr development environment. Follow the official Zephyr Getting Started Guide.

#### Initialization

The first step is to initialize a workspace folder (`my-workspace`) where the `zephyr-mvpi` and all Zephyr modules will be cloned. For development of a different project, you can also add a new application folder directly into the same workspace folder,

Run the following command:

```
# initialize my-workspace for the example-application (main branch)
west init -m  git@gitlab.com:aloxy/zephyr-mvpi --mr main my-workspace
# update Zephyr modules
cd my-workspace
west update
# ensure west uses make by default
west config build.generator "Unix Makefiles"
```

#### Building and running

To build the application, run the following command:

```
west build -b $BOARD app -G"Unix Makefiles"
```

where `$BOARD` is the target board.

You can use the `mvpi_pulse_STM32L071` board found in this repository. Note that Zephyr sample boards may be used if an appropriate overlay is provided (see `app/boards`).

-G passes which generator to use with CMake and for this project, we use "Unix Makefiles"

A sample debug configuration is also provided. To apply it, run the following command:

```
west build -b $BOARD app -- -DOVERLAY_CONFIG=debug.conf
```

Once you have built the application, run the following command to flash it:

```
west flash
```

### Testing

To execute Twister integration tests, run the following command:

```
west twister -T tests --integration
```

## Repository structure

A Zephyr application consists of hardware descriptions, application firmware and configuration files. Knowing how these files work together will aid development.

### Hardware descriptions

Hardware descriptions in the device tree format are used to pass the right configurations for the Make systems. Descriptions for specific SoCs and development boards are provided by the Zephyr Project. For a

The overlay structure is as follows:

Describes all available hardware.
*Located in existing Zephyr folder in your workspace ( under* `/zephyr/soc` )

- SoC family
  - Specific SoC
    - Pin description ( `*-pinctrl` ) \
      *Located in your project repository (* `/boards` )
      - Custom board ( `mvpi_pulse` )
      - Board overlay

\
***Board overlay*** describes specific implementation of available board hardware. \
*Located in application specific folder (* `app/boards` )

### Application firmware

Application firmware in Zephyr makes use of the configurable `Zephyr kernel` and is built up from the `main.c` code

The firmware structure is as follows:

- main.c *(Located in* `/app/src` )

```
    - include <zephyr/kernel.h>                    *Kernel*
    - include <zephyr/device.h>                    *Specific drivers and subsystems*
    - include <zephyr/drivers/gpio.h>
    - include <zephyr/logging/log.h>
    - include <path-to/lib/customlib/customlib.h>   *Custom libraries*

    //code comes here//
```

### Configuration files

The configuration files in the project are used to help development, debugging and testing.

- Cmake files are used to configure the CMake system
- Kconfig files are used to configure the Zephyr kernel and subsystems under Zephyr
- yaml files to bind required and optional variables to the device tree for drivers and subsystems and for use with Twister for testing