Date 3 June, 2021 | Version 0.1

**Getting Started with Zephyr**



Contents

[1 Introduction 3](#__RefHeading___Toc8359_834234553)

[1.1 Zephyr Project Getting Started Guide 3](#__RefHeading___Toc8361_834234553)

[Update OS 3](#__RefHeading___Toc8593_834234553)

[Install dependencies 3](#__RefHeading___Toc8595_834234553)

[Get Zephyr and install Python dependencies 4](#__RefHeading___Toc8597_834234553)

[Install a Toolchain 5](#__RefHeading___Toc8599_834234553)

[Build the Blinky Sample 6](#__RefHeading___Toc8601_834234553)

[Flash the Sample 6](#__RefHeading___Toc9007_834234553)

[1.2 Merlot MCU Source Code 7](#__RefHeading___Toc8365_834234553)

[2 ubluebox 9](#__RefHeading___Toc8367_834234553)

[2.1 Installation 9](#__RefHeading___Toc8369_834234553)

[2.2 ………………………………. 9](#__RefHeading___Toc8371_834234553)

# Introduction

## Zephyr Project Getting Started Guide

I have done steps from link below

https://docs.zephyrproject.org/latest/getting\_started/index.html

## Update OS

This guide covers Ubuntu version 18.04 LTS and later

|  |
| --- |
| sudo apt update  sudo apt upgrade |

## **Install dependencies**

Next, you’ll install some host dependencies using your package manager

1.Use apt to install the required dependencies:

|  |
| --- |
| sudo apt install --no-install-recommends git cmake ninja-build gperf \  ccache dfu-util device-tree-compiler wget \  python3-dev python3-pip python3-setuptools python3-tk python3-wheel xz- utils file \  make gcc gcc-multilib g++-multilib libsdl2-dev |

2.Verify the version of cmake that is installed on your system by entering:

|  |
| --- |
| cmake --version |

The version must be 3.13.1 or later,If your version is older, complete the following steps to add the [Kitware third-party apt repository](https://apt.kitware.com/), which contains an updated version of cmake.

a.Add the Kitware signing key:

|  |
| --- |
| wget -O - https://apt.kitware.com/keys/kitware-archive-latest.asc 2>/dev/null | sudo apt-key add - |

b.Add the Kitware apt repository for your OS release. For Ubuntu 18.04 LTS:

|  |
| --- |
| sudo apt-add-repository 'deb https://apt.kitware.com/ubuntu/ bionic main' |

c.Then install the updated version of cmake with apt:

|  |
| --- |
| sudo apt update  sudo apt install cmake |

## **Get Zephyr and install Python dependencies**

Next, clone Zephyr and its [modules](https://docs.zephyrproject.org/latest/guides/modules.html" \l "modules) into a new [west](https://docs.zephyrproject.org/latest/guides/west/index.html" \l "west) workspace named zephyrproject. You’ll also install Zephyr’s additional Python dependencies.

1.Install west, and make sure ~/.local/bin is on your PATH [environment variable](https://docs.zephyrproject.org/latest/application/index.html" \l "env-vars):

|  |
| --- |
| pip3 install --user -U west  echo 'export PATH=~/.local/bin:"$PATH"' >> ~/.bashrc  source ~/.bashrc |

2.Get the Zephyr source code:

|  |
| --- |
| west init ~/zephyrproject  cd ~/zephyrproject  west update |

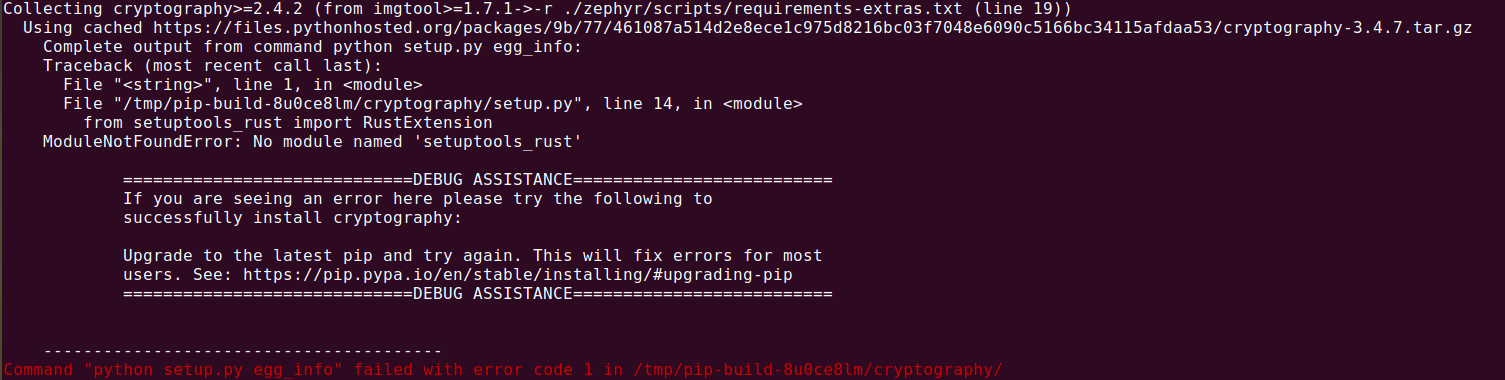
3.Export a [Zephyr CMake package](https://docs.zephyrproject.org/latest/guides/zephyr_cmake_package.html" \l "cmake-pkg). This allows CMake to automatically load boilerplate code required for building Zephyr applications.

|  |
| --- |
| west zephyr-export |

4.Zephyr’s scripts/requirements.txt file declares additional Python dependencies. Install them with pip3

|  |
| --- |
| pip3 install --user -r ~/zephyrproject/zephyr/scripts/requirements.txt |

Note: If you get any error like below



Then need to install updated pip version using below command.

**sudo -H pip3 install --upgrade pip**

## **Install a Toolchain**

A toolchain provides a compiler, assembler, linker, and other programs required to build Zephyr applications.

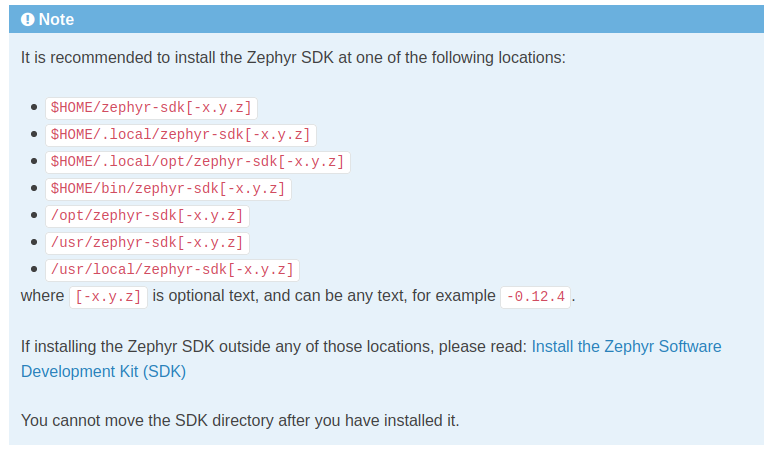
The Zephyr Software Development Kit (SDK) contains toolchains for each of Zephyr’s supported architectures.

1.Download the [latest SDK installer](https://github.com/zephyrproject-rtos/sdk-ng/releases):

|  |
| --- |
| cd ~  wget https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.12.4/zephyr-sdk-0.12.4-x86\_64-linux-setup.run |

2.Run the installer, installing the SDK in ~/zephyr-sdk-0.12.4:

|  |
| --- |
| chmod +x zephyr-sdk-0.12.4-x86\_64-linux-setup.run  ./zephyr-sdk-0.12.4-x86\_64-linux-setup.run -- -d ~/zephyr-sdk-0.12.4 |



3.Install [udev](https://en.wikipedia.org/wiki/Udev) rules, which allow you to flash most Zephyr boards as a regular user:

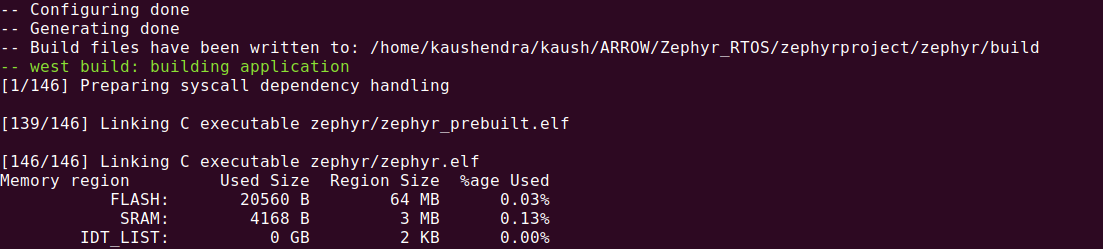
|  |
| --- |
| sudo cp ~/zephyr-sdk-0.12.4/sysroots/x86\_64-pokysdk-linux/usr/share/openocd/contrib/60-openocd.rules /etc/udev/rules.d  sudo udevadm control --reload |

## **Build the Blinky Sample**

Note:Blinky is compatible with most, but not all, [Supported Boards](https://docs.zephyrproject.org/latest/boards/index.html" \l "boards). If your board does not meet Blinky’s [Requirements](https://docs.zephyrproject.org/latest/samples/basic/blinky/README.html" \l "blinky-sample-requirements), then [Hello World](https://docs.zephyrproject.org/latest/samples/hello_world/README.html" \l "hello-world) is a good alternative.

1.Build the [Blinky](https://docs.zephyrproject.org/latest/samples/basic/blinky/README.html" \l "blinky-sample) with [west build](https://docs.zephyrproject.org/latest/guides/west/build-flash-debug.html" \l "west-building), changing <your-board-name> appropriately for your board:

|  |
| --- |
| cd ~/zephyrproject/zephyr  west build -p auto -b <your-board-name> samples/basic/blinky  eg:west build -p auto -b mimxrt685\_evk\_cm33 samples/basic/blinky  eg:west build -p auto -b qemu\_x86 samples/hello\_world  run:west build -t run |



The -p auto option automatically cleans byproducts from a previous build if necessary, which is useful if you try building another sample.

Note: Facebook didn’t provide any guide or command to build their source code,we are debugging how to build.

## **Flash the Sample**

Connect your board, usually via USB, and turn it on if there’s a power switch. If in doubt about what to do, check your board’s page in [Supported Boards](https://docs.zephyrproject.org/latest/boards/index.html" \l "boards).

Then flash the sample using [west flash](https://docs.zephyrproject.org/latest/guides/west/build-flash-debug.html" \l "west-flashing):

|  |
| --- |
| west flash |

You may need to install additional [host tools](https://docs.zephyrproject.org/latest/guides/flash_debug/host-tools.html" \l "flash-debug-host-tools) required by your board. The west flash command will print an error if any required dependencies are missing.

If you’re using blinky, the LED will start to blink as shown in this figure:

**Running Emulator**

**link:-https://docs.zephyrproject.org/latest/guides/emulator/index.html**

## Merlot MCU Source Code

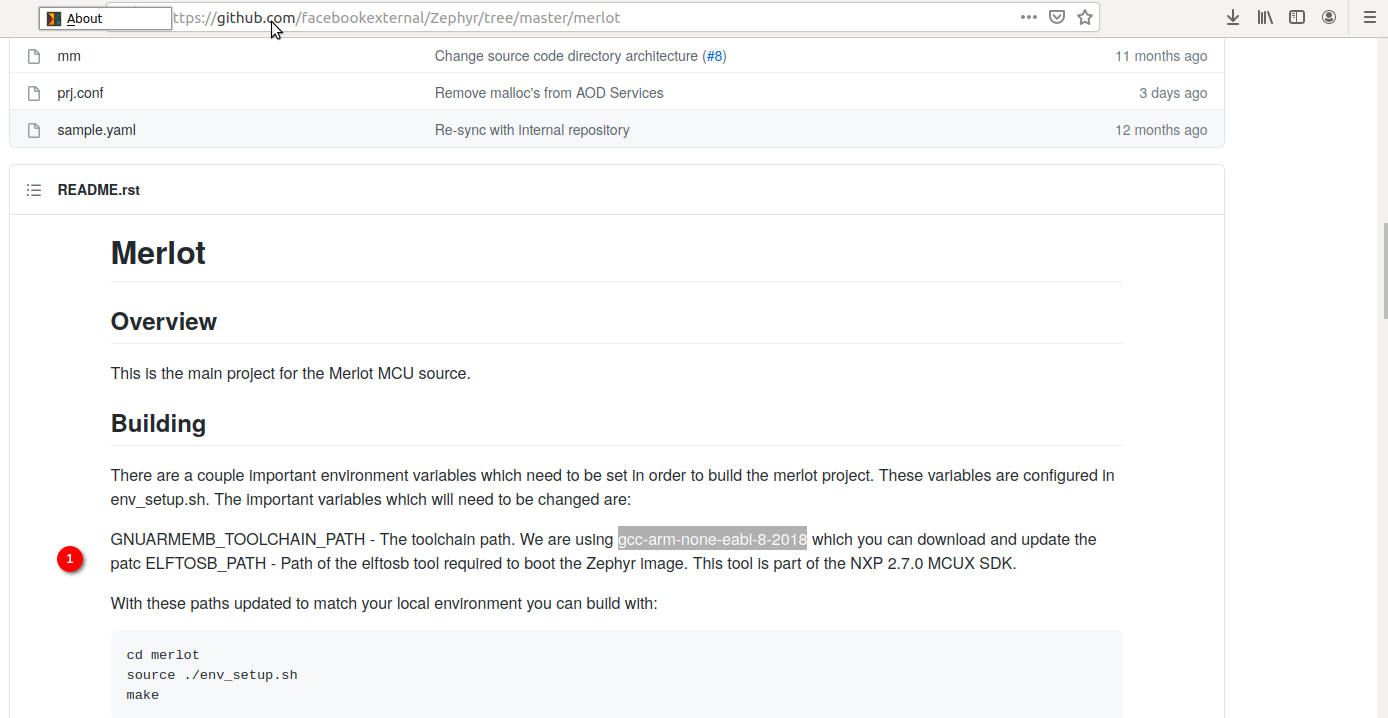
**read from https://github.com/facebookexternal/Zephyr/tree/master/merlot**

Toolchain required: **gcc-arm-none-eabi-8-2018**

export PATH=/home/kaushendra/kaush/ARROW/FRL/gcc-arm-none-eabi-8-2019-q3-update/bin/:$PATH

export BUILD\_TOOLS\_PATH=/home/kaushendra/kaush/ARROW/FRL/gcc-arm-none-eabi-8-2019-q3-update/bin/

didn’t understand this below



I opened env\_setup.sh in merlot folder, I want to know how OVRSOURCE\_DIR is defined and where is third-party??

#!/bin/bash

ABSOLUTE\_PATH="$(cd "$(dirname "${BASH\_SOURCE[0]}")" && pwd)/"

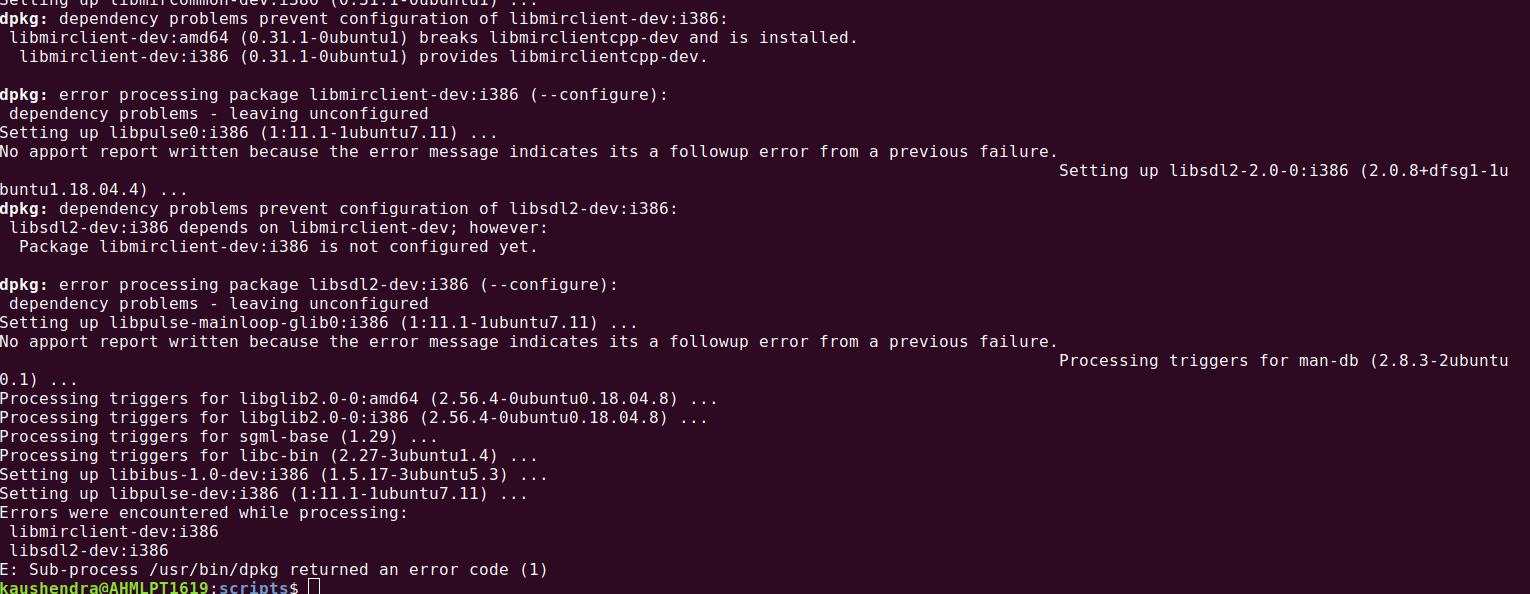
export ZEPHYR\_TOOLCHAIN\_VARIANT=gnuarmemb

export OVRSOURCE\_DIR=${ABSOLUTE\_PATH}../../../../

export ZEPHYR\_BASE=${OVRSOURCE\_DIR}/third-party/zephyr/zephyr\_rtos/v2.3.0/zephyr

step1 :cd merlot/scripts

./setup\_linux\_emulator.sh



# ubluebox

Path:

Zephyr/merlot/lib/include/arvr/firmware/lib/ubluebox

## Installation

VS Code:<https://code.visualstudio.com/>

sudo dpkg -i ./code\_1.56.2-1620838498\_amd64.deb && sudo apt-get install -f

## we have refered the documents

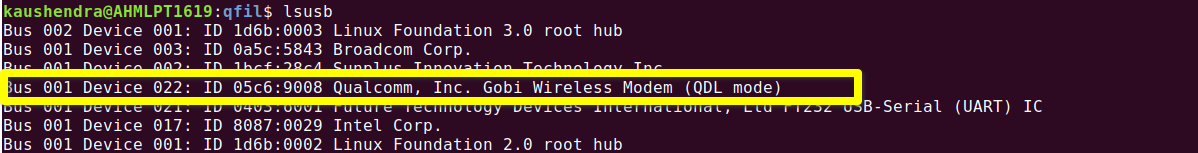
**3 QFIL Recovery Steps**

**how to identify device is in QDL mode:**

If there is no sign of life on display and the device is completely bricked (it might enumerate in QDL mode) you need to use this recovery method

Note: Connect USB-B and USB-C cable as well

**$**lsusb



dependencies:

sudo apt-get install libxml2-dev libudev-dev

Enter qfil mode via either:

$adb reboot edl

or force the device to boot into QDL (Qualcomm Download mode a.k.a Emergency

Download mode), in case the device doesn't enumerate in adb mode:

1. Unplug power and usb cables to the EDB.

2. Press and hold SW4(FORCE USB BOOT) button - Third last one towards the

bottom of EDB.

3. While still holding SW4, connect the power cable followed by usb cable.

4. Wait for ~5 seconds and then release SW4.

5. The device will boot in QDL mode. The screen will be blank. You can run lsusb

or ioreg -p IOUSB or host equivalent will enumerate the device in this mode

Note: Connect USB-B and USB-C cable as well

6. Proceed to recover the device using latest qfil pakcage from our build

dashboard.

7. From the above package execute:

$python flash\_qfil\_package.py

Note 1: You may get warning saying that an application called Santa is not yet

enabled on your computer for Security. This is FB's security to guard against

unauthorized apps. It should open a window to allow this application to run. Do

this and try again.

Note 2: If you get an error in kickstart\_darwin when programming

prog\_emmc\_firehose\_8937\_ddr.min, try restarting from step 1 (unplug power

and usb)... and executing python script without explicitly calling python so just

./flash\_qfil\_package.py

8.The device will reboot into the new image after a successful recovery.