Zephyr OS for Arch-based distributions and MCUboot

- **Required Dependencies**

```
sudo pacman -Syyu git python-pip python-setup-tools python-wheel \
    python-pyserial gperf git wget curl xz ninja file \
    cmake bison make flex gcc drc openocd arm-none-eabi-gcc \
    arm-none-eabi-bitutils arm-none-eabi-gdb \
    patchet dfu-util gcovr dtc\
    python-pytest python-anytree \
    python-breathe python-intelhex \
    python-packagin python-ply python-pyaml \
    python-pyelftools python-pywalify \
    python-tabulate ccache doxygen
```

- **Setting up west**

  West is developed in its own repository.

```
cd /tmp
git clone https://aur.archlinux.org/python-west.git

#Create the Installation package
cd python-west
makepkg -s

#Chrome archive with Update option
sudo pacman -U python-west*.tar.xz

#Cleanup
cd ..

rm-rf python-west
```

- **Creating the directory**

```
mkdir ~/Workspace
```

- **Downloading all Zephyr packages** (Caution -> ~2.6GByte)

```
cd ~/Workspace

west update
```

- **Additional Python dependencies after**

```
cd ~/Workspace

sudo pip instal -r zephyr/scripts/requirements.txt
```

- **Creating the Environment setup file**

```
cd ~/Workspace

#Enable Initialuzation
echo "source \${HOME}/Workspace/zephyr/zephyr-env.sh" > env.sh

#Using te ARM GCC Compiler
echo "export ZEPHYR_TOOLCHAIN_VARIANT=cross-compile" >> env.sh

#Set the path for the ARM GCC Compiler
echo "export CROSS_COMPILE='which arm-none-eabi-gcc | sed 's/gcc/g'`" >> env.sh
```

- **Setting up openocd (Open On-Chip Debugger)**

```
#creating temp location
cd /temp

wget -0 60-openocd.rules https://sf.net/p/penocd/code/ci/master/tree/contrib/ 60-openocd.rules

#Copy to system
sudo cp -60-openocd.rules /etc/udev/rules.d

#Reload the USB sub-system
sudo udevadm control --reload

#Cleanup
sudo rm -60-openocd.rules

cd ~
```

- **Setup of Zephyr is done here**
- **To build use**

```
west build -p auto -b <your-board-name> samples/basic/blinky
```

- **To flash use**

```
west flash
```

taken from https://www.zephyrproject.org/zephyr-os-getting-started-on-manjaro-arch-linux/

MCUboot

MCUboot is a secure bootloader for 32-bits microcontrollers. It defines a common infrastructure for the bootloader and the system flash layout on microcontroller systems, and provides a secure bootloader that enables easy software upgrade.

can be found here -> https://github.com/mcu-tools/mcuboot

- **My Testing**
- **Clone locally into Zephyr directory, which was set on previous state (see above guide). Mine is ~/Workspace**
- **Next, we need to manually set the partitions needed for the bootloader.**

```
- boot_partition    # : for MCUboot itself
- image_0_primary_partition #: the primary slot of Image 0
- image_0_secondary_partition #: the secondary slot of Image 0
- scratch_partition #: the scratch slot
```

- **Then we need to find the device tree of the board we want to load the bootloader into. The device tree can be found on the zephyr directory**

```
 ~/Workspace/zephyr/boards/*/*/*.dts
```

- **Now we need to manually set the following partitions**

```
&flash0 {

    partitions {
        compatible = "fixed-partitions";
        #address-cells = <1>;
        #size-cells = <1>;

        boot_partition: partition@0 {
            label = "mcuboot";
            reg = <0x00000000 0x00010000>;
            read-only;
        };
        /*
        * The flash starting at 0x00010000 and ending at
        * 0x0001ffff (sectors 16-31) is reserved for use
        * by the application.
        */
        storage_partition: partition@1e000 {
            label = "storage";
            reg = <0x0001e000 0x00002000>;
        };
        slot0_partition: partition@20000 {
            label = "image-0";
            reg = <0x00020000 0x00060000>;
        };
        slot1_partition: partition@80000 {
            label = "image-1";
            reg = <0x00080000 0x00060000>;
        };
        scratch_partition: partition@e0000 {
            label = "image-scratch";
            reg = <0x000e0000 0x00020000>;
```

```
        };
    };
};
```

- **As far as I know the addresses stated above are universal and not board-pecific**

- **Then we need to set the partition from which the MCU bootloader will be booted from when flashed**
- **On the file ~/Workspace/mcuboot/boot/zephyr/dts.overlay change to slot0_partition (from oroginal boot_partition I think?)**

```
/ {
chosen {
    zephyr,code-partition = &slot0_partition;
    };
};

^^needs verification
```

- Side-note testing:

  When tested on a board with limited SRAM, an overflow error was occured.

  The error was resolved by reducing the parameter shown bellow by 50% (need to be examined)

  This was on file mcuboot/boot/zephyr/prj.conf

```
#changed from

CONFIG_USB_CDC_ACM_RINGBUF_SIZE=10240

#to

CONFIG_USB_CDC_ACM_RINGBUF_SIZE=5120
```

  Testing was terminated though, due to errors upon flashing the bootloader. The output of serial was:

```
*** Booting Zephyr OS build zephyr-v3.0.0-2633-ga6f71586170f  ***
I: Starting bootloader
W: Failed reading sectors; BOOT_MAX_IMG_SECTORS=128 - too small?
E: Unable to find bootable image
```

- **Next, we need to enable the bootloader on the project config file.**
- **The configurations of a project can be found on the project/prj.conf file (example: zephyr/samples/basic/blinky/prj.conf)**

```
#added parameter
CONFIG_BOOTLOADER_MCUBOOT=y
```

- **Now we are ready to build the bootloader on out board**

```
cd ~/Workspace/mcuboot

west build -s boot/zephyr -b stm32f469i_disco

west flash
```

- **The output was:**

```
*** Booting Zephyr OS build zephyr-v3.0.0-2633-ga6f71586170f  ***
I: Starting bootloader
I: Primary image: magic=bad, swap_type=0x1, copy_done=0x2, image_ok=0x2
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Boot source: none
I: Swap type: none
E: Unable to find bootable image
```

- **The bootloader was flashed succesfully, but no signed image was flashed, thus the error**
- **Now to build the project from zephyr**

```
cd ~/Workspace/zephyr

#or cd $ZEPHYR_BASE

#build

west build -b stm32f469i_disco  samples/basic/blinky -p

#--pristine command is used when board is already flashed or build with another project
```

- **Now we need to generate the signed images from the bootloader**

```
cd ~/Workspace/mcuboot/boot/zephyr
mkdir build && cd build
cmake -GNinja -DBOARD=stm32f469i_disco ..
ninja

make BOARD=stm32f469i_disco all

#the last command generated the following files
    #mcuboot.bin
    #signed-hello1.bin
    #signed-hello2.bin

cd ../mcuboot/scripts

./imgtool.py sign --key ../root-rsa-2048.pem --header-size 0x200 -S 0x6000 --align 8 --version 1.2
../../zephyr/build/zephyr/zephyr.hex signed-zephyr.hex

cd ../

west flash --hex-file scripts/signed-zephyr.hex
```

- **Flash the generated bin files**

```
make flash_full
pyocd flash -e chip -a 0 full.bin
```

- **Now, the following error occured:**

```
0000520 C Target type stm32f469nihx not recognized. Use 'pyocd list --targets' to see currently ava
ilable target types. See <https://github.com/pyocd/pyOCD/blob/master/docs/target_support.md> for ho
w to install additional target support. [__main__]
Traceback (most recent call last):
    File "/usr/lib/python3.10/site-packages/pyocd/board/board.py", line 100, in __init__
    self.target = TARGET[self._target_type](session)
KeyError: 'stm32f469nihx'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
    File "/usr/lib/python3.10/site-packages/pyocd/__main__.py", line 161, in run
    status = cmd.invoke()
    File "/usr/lib/python3.10/site-packages/pyocd/subcommands/load_cmd.py", line 81, in invoke
    session = ConnectHelper.session_with_chosen_probe(
    File "/usr/lib/python3.10/site-packages/pyocd/core/helpers.py", line 263, in session_with_chose
n_probe
    return Session(probe, auto_open=auto_open, options=options, **kwargs)
    File "/usr/lib/python3.10/site-packages/pyocd/core/session.py", line 213, in __init__
    self._board = probe.create_associated_board() or Board(self)
    File "/usr/lib/python3.10/site-packages/pyocd/probe/stlink_probe.py", line 123, in create_assoc
iated_board
    return MbedBoard(self.session, board_id=self._board_id)
    File "/usr/lib/python3.10/site-packages/pyocd/board/mbed_board.py", line 67, in __init__
    super(MbedBoard, self).__init__(session, target)
    File "/usr/lib/python3.10/site-packages/pyocd/board/board.py", line 102, in __init__
    raise exceptions.TargetSupportError(
pyocd.core.exceptions.TargetSupportError: Target type stm32f469nihx not recognized. Use 'pyocd list
 --targets' to see currently available target types. See <https://github.com/pyocd/pyOCD/blob/maste
r/docs/target_support.md> for how to install additional target support.
make: *** [Makefile:179: flash_full] Error 1
```

- **The problem was that my board was missing from the package support offered out-of-the-box for pyocd**
- **Solution was to manually detect it and download the missing packages**

```
#detects device
pyocd list

#to download specific device's support package for pyOCD

pyocd pack install stm32f469nihx
```

- **Repeat**

```
make flash_boot

pyocd flash -e chip -a 0 mcuboot.bin
```

- **Upon following error I found that I needed to flash the full.bin binary**

```
0001028 I Loading /home/harrkout/Workspace/mcuboot/samples/zephyr/mcuboot.bin at 0x00000000 [load_c
md]
0006036 C no memory region defined for address 0x00000000 [__main__]
Traceback (most recent call last):
    File "/usr/lib/python3.10/site-packages/pyocd/__main__.py", line 161, in run
    status = cmd.invoke()
    File "/usr/lib/python3.10/site-packages/pyocd/subcommands/load_cmd.py", line 117, in invoke
    programmer.program(filename,
    File "/usr/lib/python3.10/site-packages/pyocd/flash/file_programmer.py", line 169, in program
    self._format_handlers[file_format](file_obj, **kwargs)
    File "/usr/lib/python3.10/site-packages/pyocd/flash/file_programmer.py", line 194, in _program_
bin
    self._loader.add_data(address, data)
    File "/usr/lib/python3.10/site-packages/pyocd/flash/loader.py", line 231, in add_data
    raise ValueError("no memory region defined for address 0x%08x" % address)
ValueError: no memory region defined for address 0x00000000
make: *** [Makefile:170: flash_boot] Error 1
```

- **Solution**

```
pyocd flash full.bin
```

- **Serial Output**

```
*** Booting Zephyr OS build zephyr-v3.0.0-2633-ga6f71586170f  ***
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x4, copy_done=0x1, image_ok=0x1
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Boot source: none
I: Swap type: none
I: Bootloader chainload address offset: 0x20000
I: Jumping to the first image slot
```

Image Signing with custom key

- **Generate a key using openssl command : (on dir /mcuboot)**

  openssl genrsa -out my_key.pem 2048

This command will create a file named my_key.pem which contains both private and public key.

- **Modify the prj.conf file to include the generated key**

```
#boot/zephyr/prj.conf
Modify the prj.conf file to include the generated key
```

- **Rebuild and flash the MCUboot bootloader**

```
west build -s boot/zephyr -b stm32f469i_disco

west flash
```

- **Serial output**

```
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x2, copy_done=0x1, im3
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_o3
I: Boot source: none
I: Swap type: revert
E: Image in the secondary slot is not valid!
E: Image in the primary slot is not valid!
E: Unable to find bootable image
```

- **To sign via west instead of imgtool.py**

```
pip3 install --user imgtool

#sign the image

west sign -t imgtool -- --key ../mcuboot/my_key.pem
<br/>
```

Errors occured

- **Changed made according to errors on ROM address**

```
#on /Workspace/mcuboot/build/zephyr/.config
#change line
CONFIG_ROM_START_OFFSET=0
#to
CONFIG_ROM_START_OFFSET=0x200
```

- **Next error:**

WARNING: CONFIG_BOOTLOADER_MCUBOOT is not set to y in
/home/harrkout/Workspace/mcuboot/build/zephyr/.config; this probably won't

every time the bootloader reflashes it gets deleted???

- **Output after sign**

```
=== image configuration:
partition offset: 131072 (0x20000)
partition size: 393216 (0x60000)
rom start offset: 512 (0x200)
=== signing binaries
unsigned bin: /home/harrkout/Workspace/mcuboot/build/zephyr/zephyr.bin
signed bin:   /home/harrkout/Workspace/mcuboot/build/zephyr/zephyr.signed.bin
unsigned hex: /home/harrkout/Workspace/mcuboot/build/zephyr/zephyr.hex
signed hex:   /home/harrkout/Workspace/mcuboot/build/zephyr/zephyr.signed.hex
```

- **Rebuild mcu and reflash**

This will generate the signed bin and hex in the build directory.

```
  build/zephyr/zephyr.signed.bin
  build/zephyr/zephyr.signed.hex
```

- **after a while though this happens**

```
*** Booting Zephyr OS build zephyr-v3.0.0-2633-ga6f71586170f  *
    I: Starting bootloader
    I: Primary image: magic=unset, swap_type=0x1, copy_done=0x3, i3
    I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_o3
    I: Boot source: primary slot
    I: Swap type: test
    I: Starting swap using scratch algorithm.
    I: Bootloader chainload address offset: 0x20000
    I: Jumping to the first image slot
    E: ***** BUS FAULT *****
    E:   Precise data bus error
    E:   BFAR Address: 0xfffffffc
    E: r0/a1:  0x00000030  r1/a2:  0x00000000  r2/a3:  0x00000000
    E: r3/a4:  0xfffffff8 r12/ip:  0xffffffff r14/lr:  0x08004d2f
    E:  xpsr:  0x01000000
    E: Faulting instruction address (r15/pc): 0x08000daa
    E: >>> ZEPHYR FATAL ERROR 0: CPU exception on CPU 0
    E: Current thread: 0x20000128 (unknown)
    E: Halting system
```

- **When signed image is invalid**

```
*** Booting Zephyr OS build zephyr-v3.0.0-2633-ga6f71586170f  *
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x2, copy_done=0x1, im3
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_o3
I: Boot source: none
I: Swap type: revert
E: Image in the primary slot is not valid!
E: Unable to find bootable image
```

How image signing works

This signs the image by computing hash over the image, and then signing that hash. Signature is computed by newt tool when it's creating the image. This signature is placed in the image trailer.

The public key of this keypair must be included in the bootloader, as it verifies it before allowing the image to run.

Conclusion

MCUboot uses Asymmetric encryption for secure booting. Here, Bootloader uses public key and application image uses private key for signing the image.

---

todo: openhub vs golioth??