HOW-TO Guide for Beluga

Installing the Toolchain

This section will detail the steps for installing the toolchain and setting up VS code to build and flash the Beluga firmware to hardware.

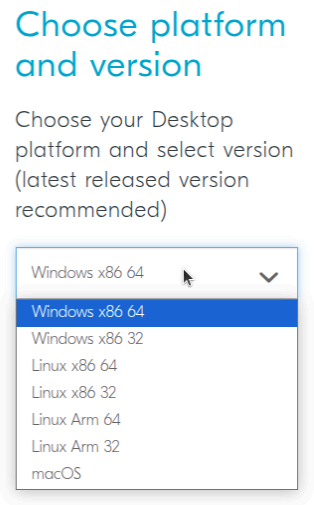
Tools to Install

- nRF Command Line Tools
- nRF Connect
- SEGGER J-Link
- VS code

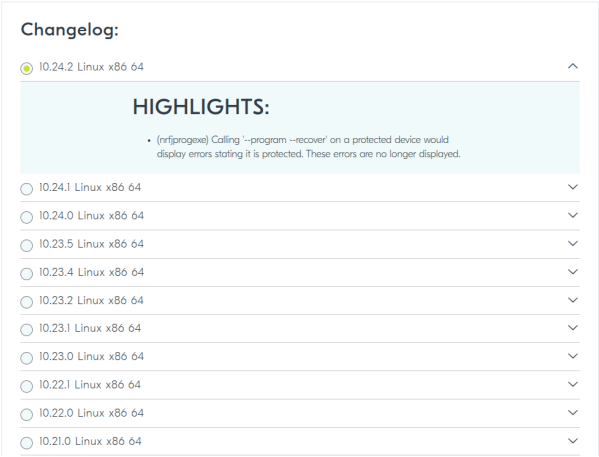
Installing nRF Command Line Tools and nRF Connect

This section details how to install the nRF tools since their website is non-intuitive.

1. Navigate to the link specified in Tools to Install
2. Select the platform you are running on



3. Select the version you want (latest version should be fine)



4. Click on the download link

Additional Steps for nRF Connect (Linux)

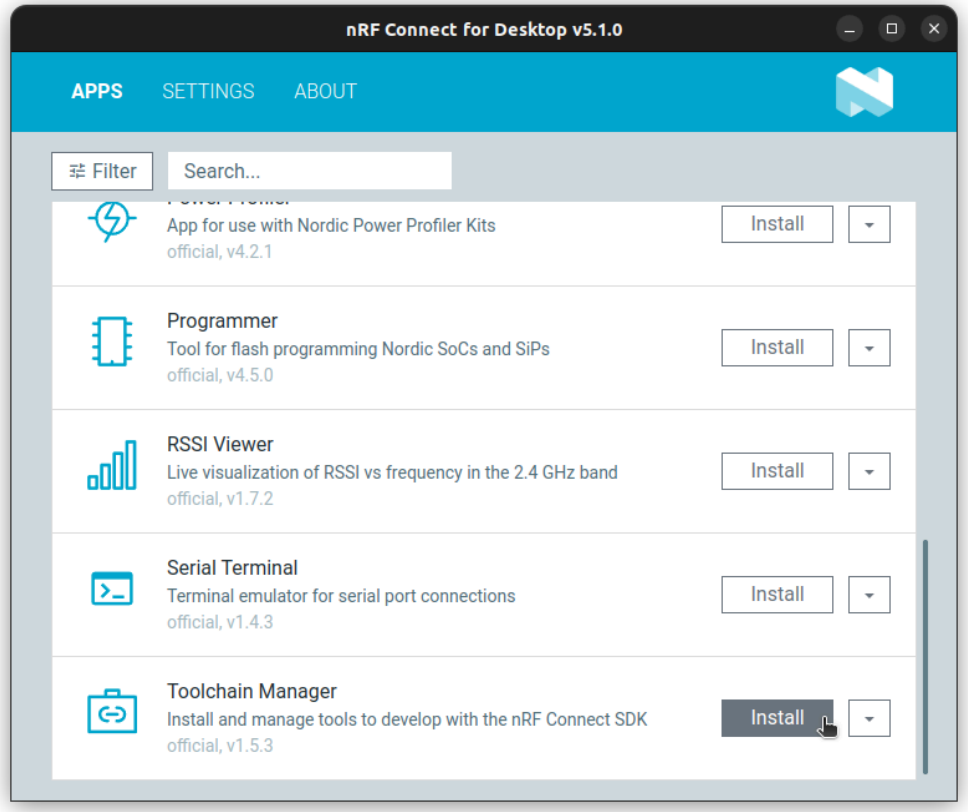
When you download nRF Connect for linux, it downloads as an app image. Before running it, you need to install libfuse2 and mark the file as executable.

```
Unset
sudo apt install libfuse2
sudo chmod +x nrfconnect-5.1.0-x86_64.appimage # Make sure to change image name
```

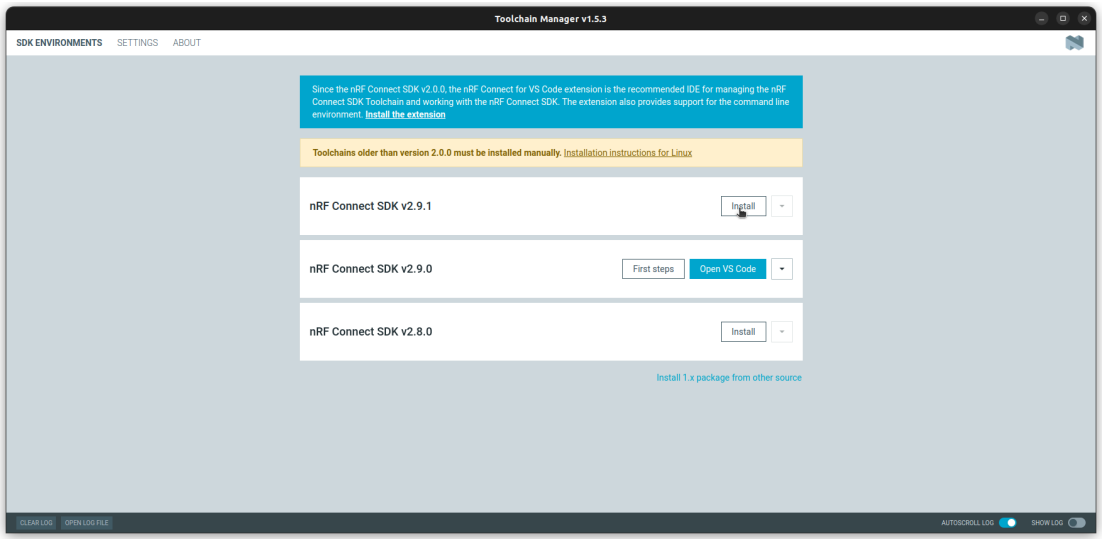
Installing the NCS toolchain

Even though the Nordic extension in VS code has the option to install the toolchain from there, it does not work. This method is proven to work.

1. Open nRF Connect
2. Install the “Toolchain Manager”



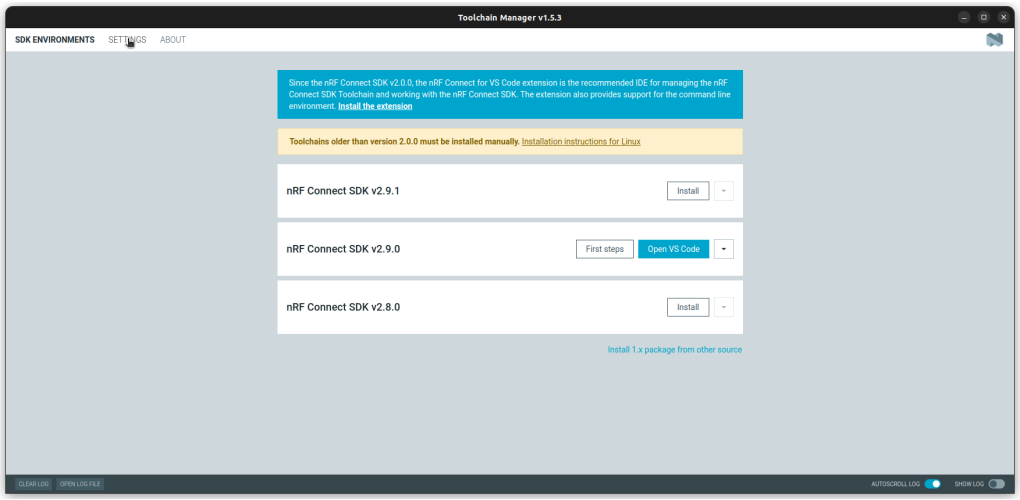
3. Open the Toolchain Manager and install nRF Connect SDK v2.9.0



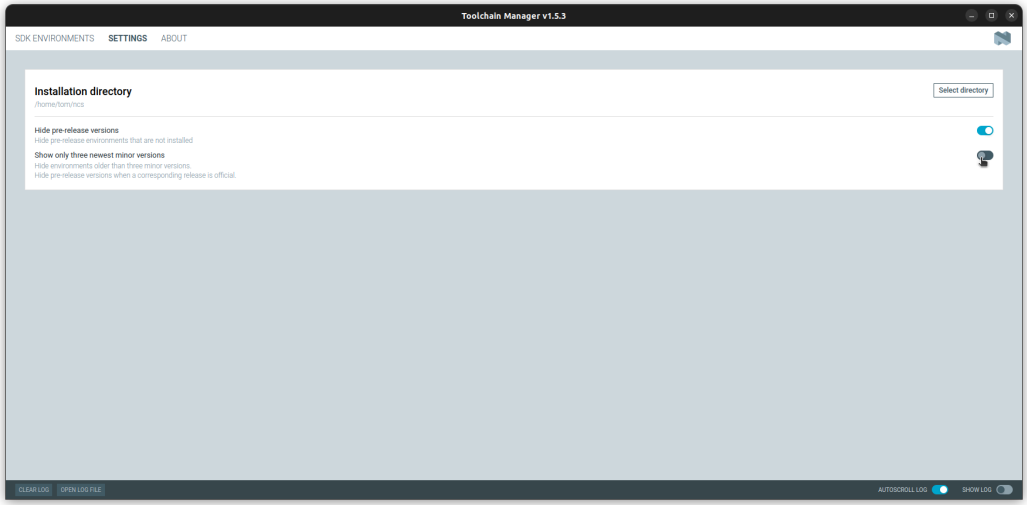
Cannot find v2.9.0? No Problem!

The nRF Connect Toolchain Manager shows the latest 3 versions of the SDK by default. To get around this, follow the steps below:

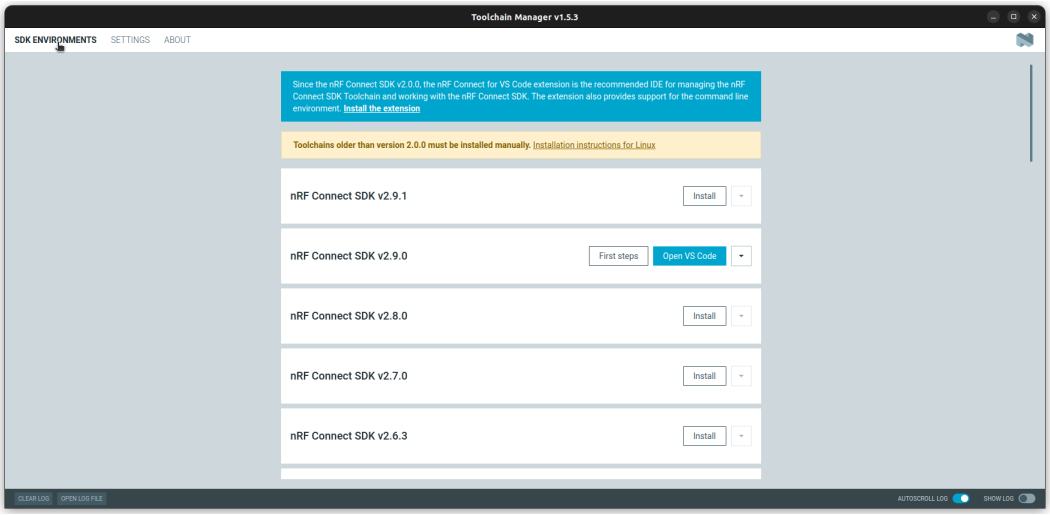
1. Click on the settings tab



2. Toggle the “Show only three newest minor versions” option



3. Navigate back to the “SDK Environments” tab

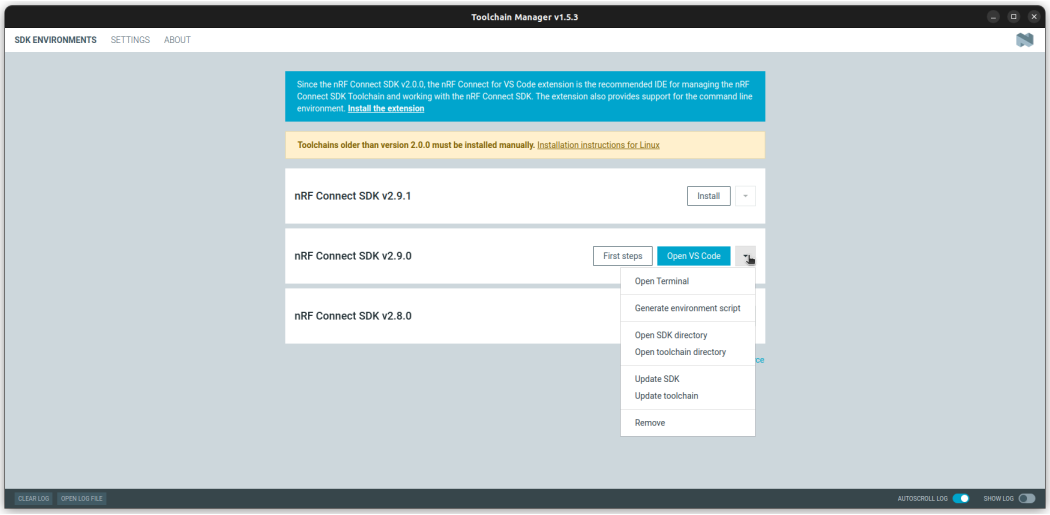


Now you can access versions no older than v2.0.0.

(Optional) Generate Environment Script

Sometimes, it is necessary to generate the environment script. For instance, if you want to use a Jetbrains IDE, this is a necessary step. Another use case for the environment script is using the NCS environment in a terminal.

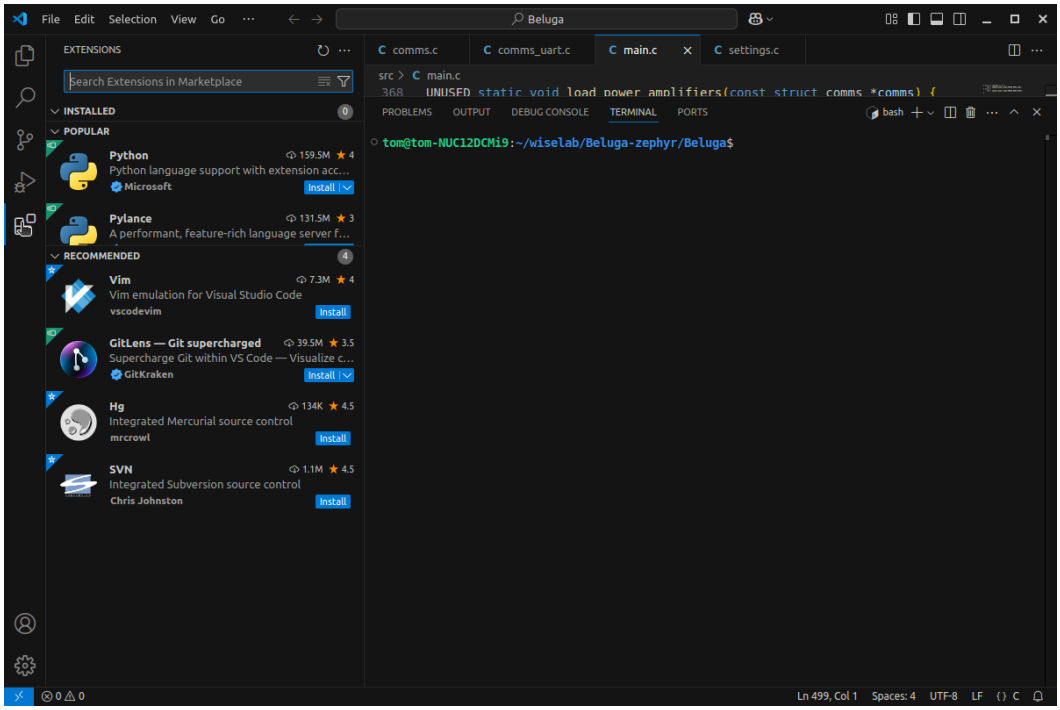
1. Click on the Down Arrow



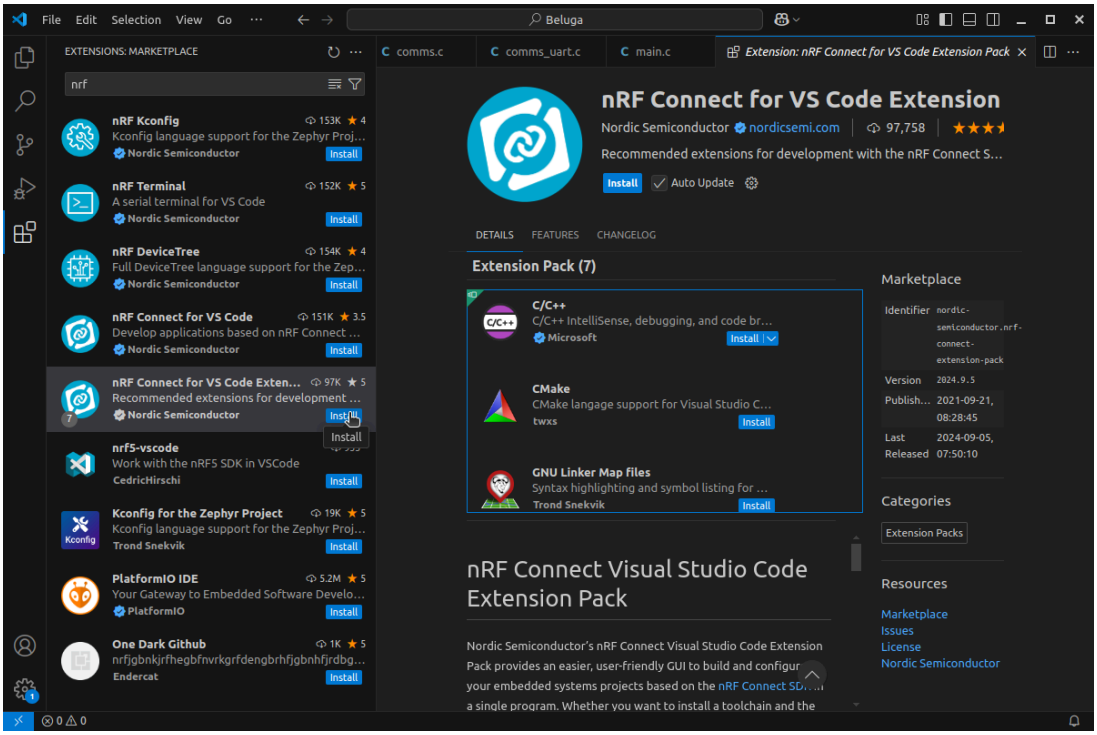
2. Click “Generate environment script”
3. Select the save location for the environment script

Installing VS code extension

1. Open VS code
2. Open “Extensions”



3. Search ‘nRF Connect” in the marketplace
4. Install the “nRF Connect for VS Code Extension”

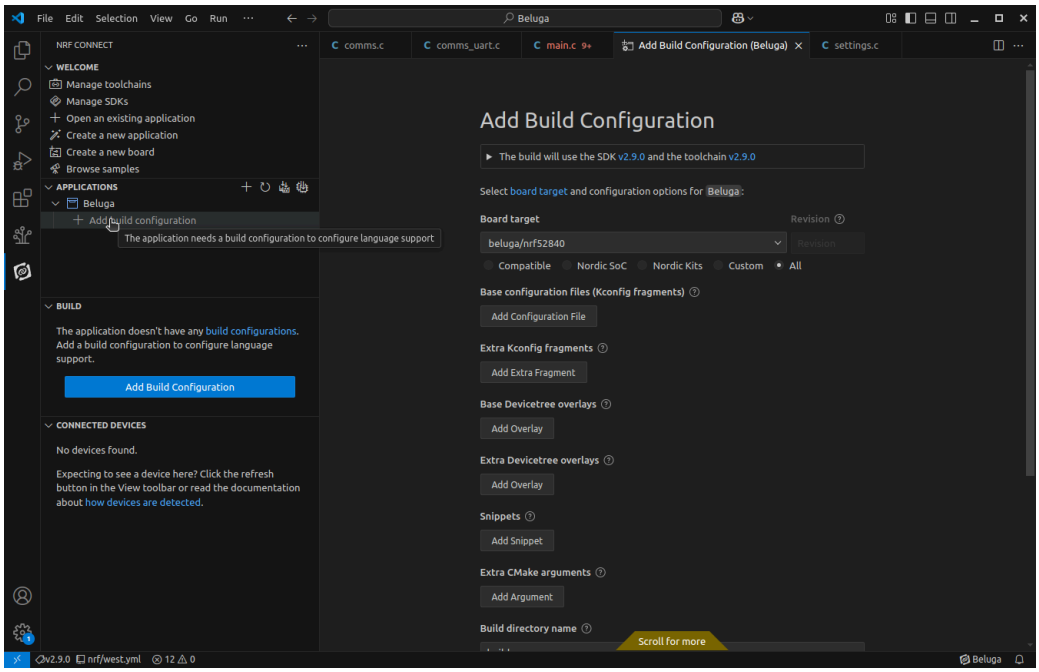


Configuring Builds

This section will detail the steps for creating a board configuration in VS code. It will also detail the requirements for both the decawave dev kit as well as the custom Beluga board.

Adding a build configuration

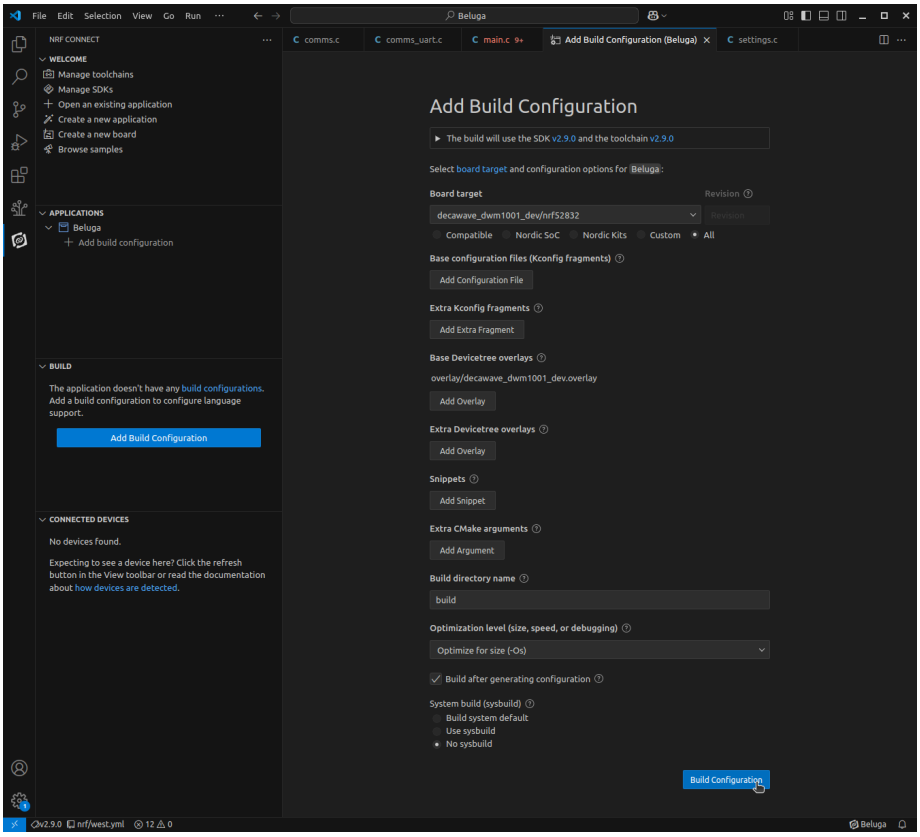
1. Open VS code
2. Switch to the nRF Connect tab (CTRL+ALT+N)



3. Click “Add Build Configuration”
4. Follow steps for the specific board you are configuring for

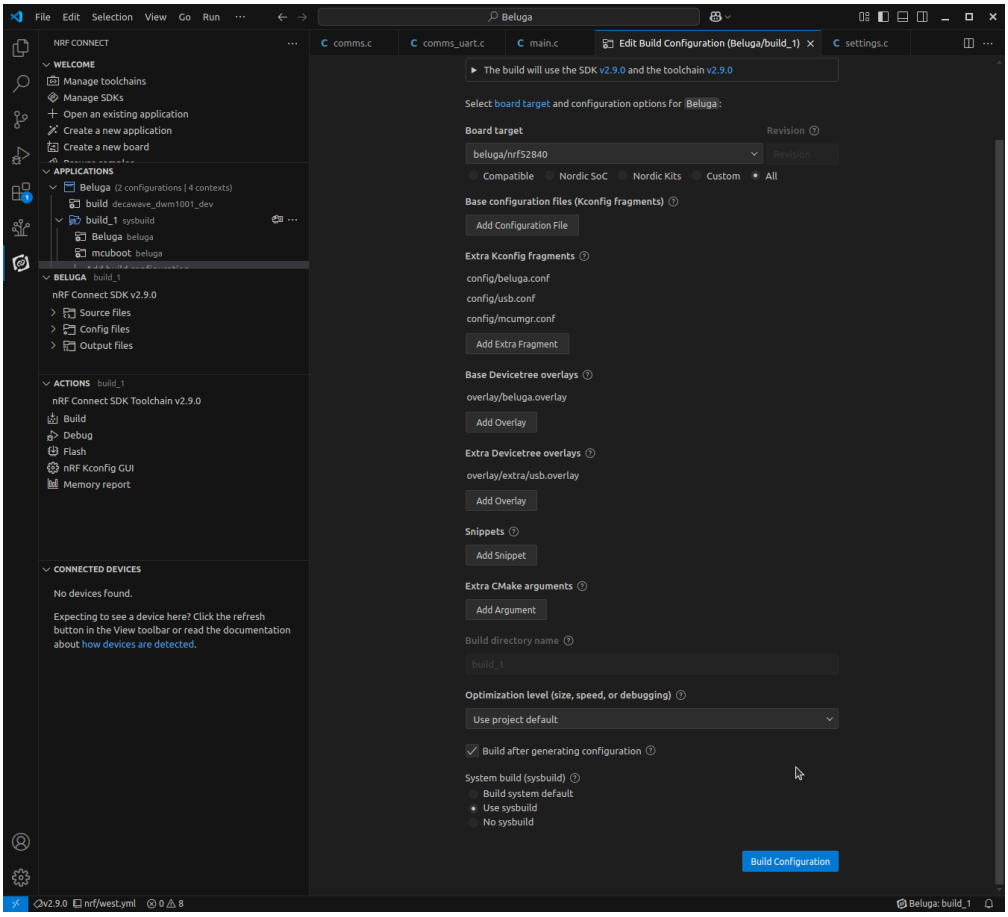
Decawave DWM1001 Dev Kit

1. Select **Board target** to be “decawave_dwm1001_dev/nrf52832” (You may have to select “All” under board target)
2. (Optional) Add “prj.conf” as the **Base configuration files (Kconfig fragments)**
    a. This is not needed, but if there is a weird issue with the build, then it is worth a try
3. Add “overlay/decawave_dwm1001_dev.overlay” to the **Base Devicetree overlays**
4. Select “Optimize for size (-Os) for **Optimization level (size, speed, or debugging)**
5. Select “No sysbuild” for **System build (sysbuild)**
6. Confirm your settings match the ones in the image



Custom Beluga Hardware

1. Select **Board target** to be “beluga/nrf52840” (You may have to select “All” under board target)
2. (Optional) Add “prj.conf” as the **Base configuration files (Kconfig fragments)**
    a. This is not needed, but if there is a weird issue with the build, then it is worth a try
3. Add the following files to **Extra Kconfig fragments**
    a. conf/beluga.conf
    b. conf/mcumgr.conf
    c. conf/usb.conf
4. Add “overlay/beluga.overlay” to the **Base Devicetree overlays**
5. Add “overlay/extra/usb.overlay” to the **Extra Devicetree overlays**
6. Select whatever **Optimization level** you want
7. Select “Use sysbuild” under **System build (sysbuild)**
8. Confirm that you have the correct settings



Building for Custom Beluga Hardware with External Flash

1. Add the following configurations to **sysbuild/mcuboot.conf**

```
CONFIG_NORDIC_QSPI_NOR=y
CONFIG_BOOT_MAX_IMG_SECTORS=256
```

2. Add the following to **sysbuild/mcuboot.overlay**

```
&mx25r64 {
        status = "okay";
};
```

```
/ {
        chosen {
                nordic,pm-ext-flash = &mx25r64;
        };
    };
```

3. Add the following to **sysbuild.conf**

```
Unset
SB_CONFIG_PM_EXTERNAL_FLASH_MCUBOOT_SECONDARY=y
```

4. Follow the steps in Custom Beluga Hardware, but also add the following files to the specified sections below.
    a. Add “config/flash.conf” to **Extra Kconfig fragments**
    b. Add “overlay/extra/flash.overlay” to **Extra Devicetree overlays**

Signing the image with a custom key

When deploying in the world, it is highly recommended to sign the image with a custom key. This key can easily be generated and prevents anyone from uploading their own firmware.

Setting up generation environment

Before generating a key, a python environment needs to be set up. Run the following commands to get a Python environment set up with *imgtool* installed.

```
Unset
mkdir -p keys && cd keys
python3 -m venv .venv
source .venv/bin/activate
pip install imgtool
```

Note that the commands may be a little different for Windows (Why would you even run Windows?).

Before proceeding, ensure that the tool got installed correctly by running `imgtool --help`. If it shows usage information, then it installed correctly. If it shows a `Module not found error` then you need to run the following command:

```
Unset
pip install cryptography intelhex click cbor2 pyyaml
```

Generating a key

Once the environment is set up, a new key can be generated by running one of the commands below:

```
Unset
imgtool keygen -t ecdsa-p256 -k private_key.pem
imgtool keygen -t rsa-2048 -k private_key.pem
imgtool keygen -t rsa-3072 -k private_key.pem
imgtool keygen -t ed25519 -k private_key.pem
```

Remember which algorithm you used to generate the key as it will be important for the firmware. Additionally, backup the kay somewhere safe. It is not uncommon to lose the key and thus be unable to ever do DFU on the device again (until the device is flashed again over JTAG).

Incorporating the key into firmware

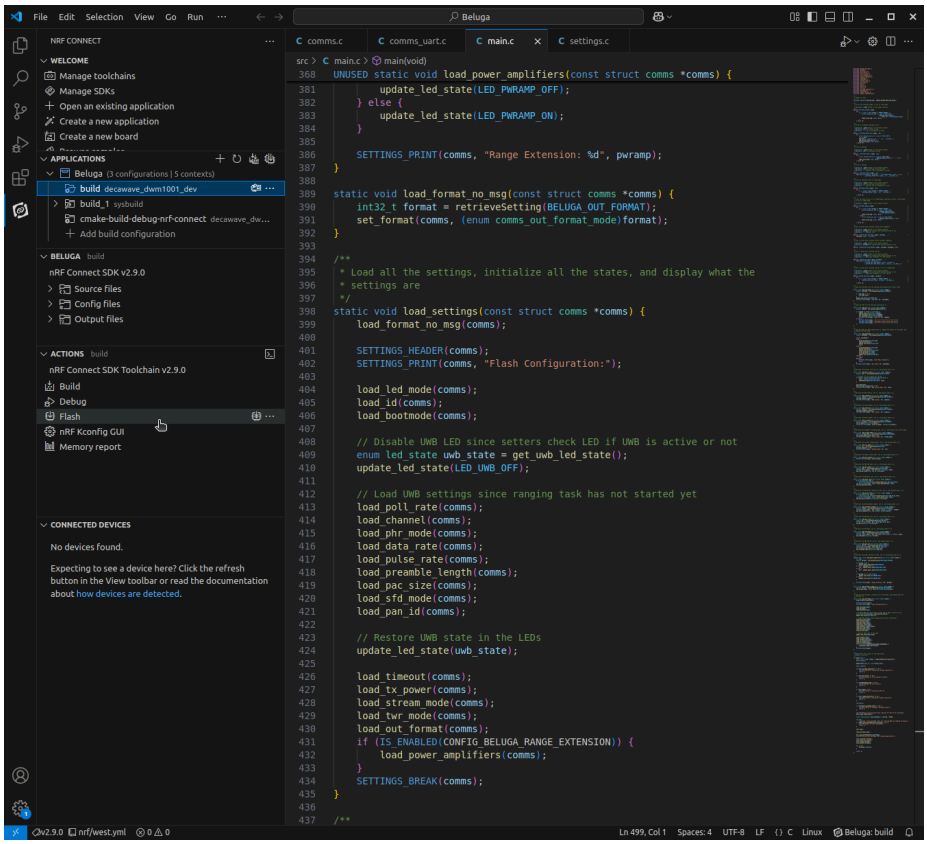
Once the key is generated, it needs to be incorporated into firmware. This is relatively easy as it requires you to update sysbuild.conf. For example, if an ecdsa-p256 key was generated in Beluga/keys, the the following lines would have to be added to **sysbuild.conf**:

```
Unset
SB_CONFIG_BOOT_SIGNATURE_KEY_FILE="\${APP_DIR}/keys/private_key.pem"
SB_CONFIG_BOOT_SIGNATURE_TYPE_ECDSA_P256=y
```

Flashing the firmware

Flashing onto a Decawave DWM1001 Dev

1. Plug the decawave DWM1001 Dev into your computer
2. Select the DWM1001 build
3. Press flash



Flashing onto the Custom Beluga Hardware

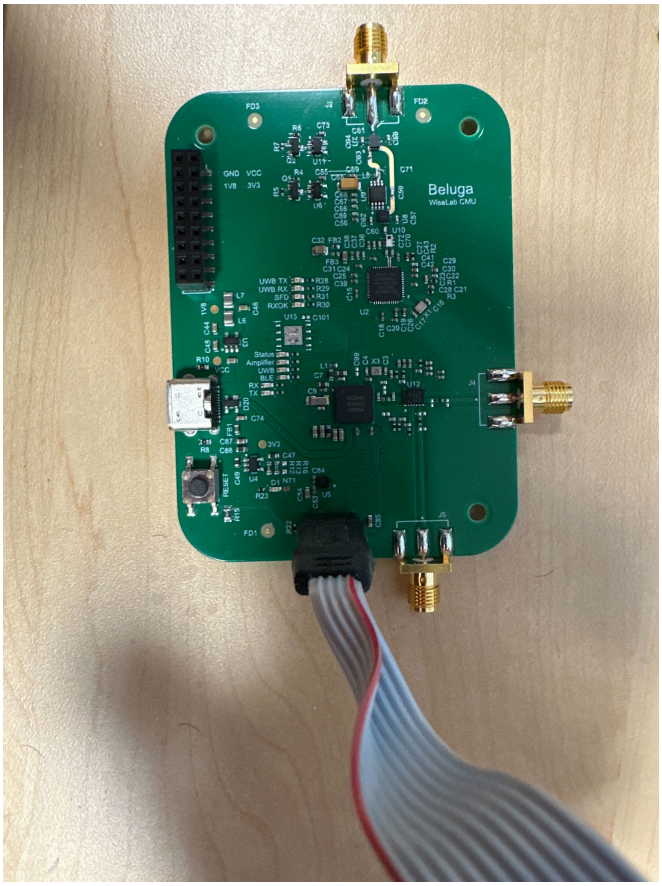
Flashing using JTAG is required for the first time. Any time after that can be done over DFU.

Using JTAG

Before flashing, make sure to select the Beluga image.

Using SEGGER

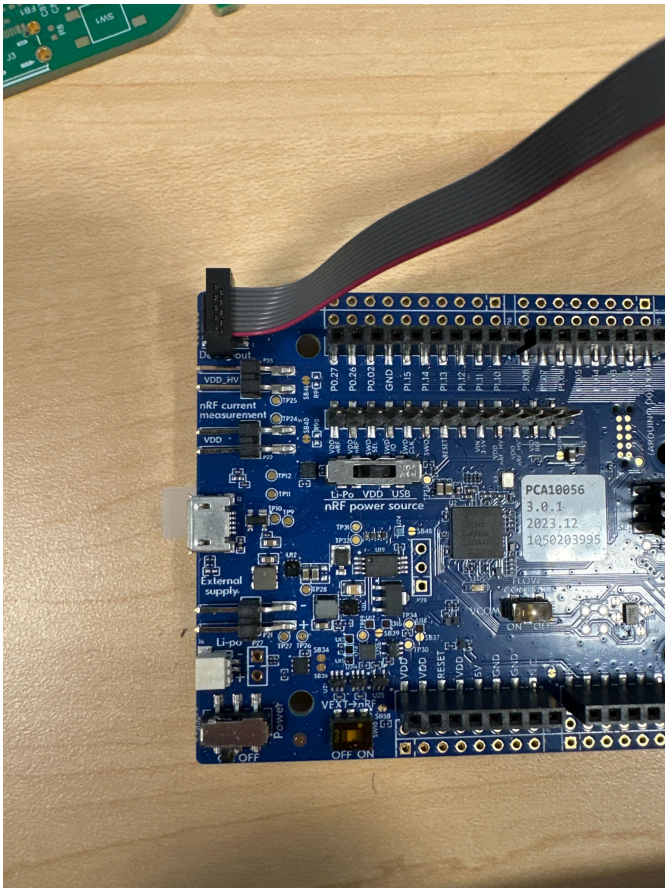
1. Plug the SEGGER J-Link into your computer
2. Connect the J-Link to the board using the Tag-Connect TC2050-IDC



3. Select Beluga build
4. Press flash

Using nRF52 devkit

1. Connect a ribbon cable to the Debug Out connector on an [nRF52 dev kit]



2. Connect the ribbon cable to an adaptor
3. Connect adaptor to the board using the [Tag-Connect TC2050-IDC]
4. Connect dev kit to computer
5. Select beluga build
6. Press flash

Using DFU

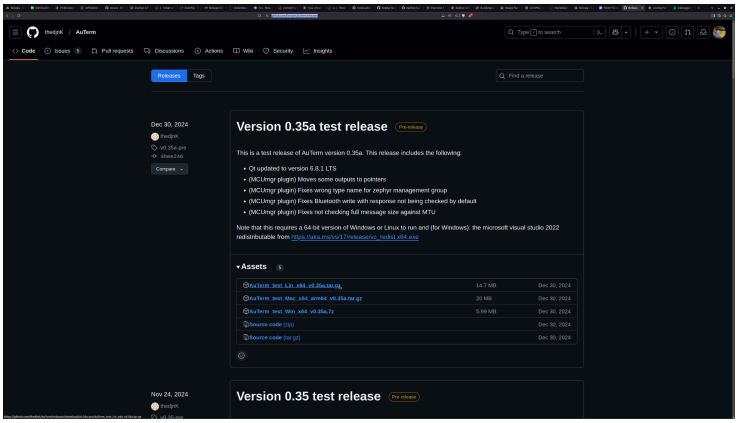
After flashing using JTAG, the custom Beluga hardware will be able to be flashed over USB. This will require an SMP client to do. There are a few tools and libraries that Zephyr recommends, all of which can be found here. If the web page is no longer working, reference the table below:

Name	OS support			Type	Language	License
	Windows	Linux	Mac			
AuTerm	Yes	Yes	Yes	Application	C++	GPL-3.0
mcumgr-client	Yes	Yes	Yes	Application	Rust	Apache-2.0

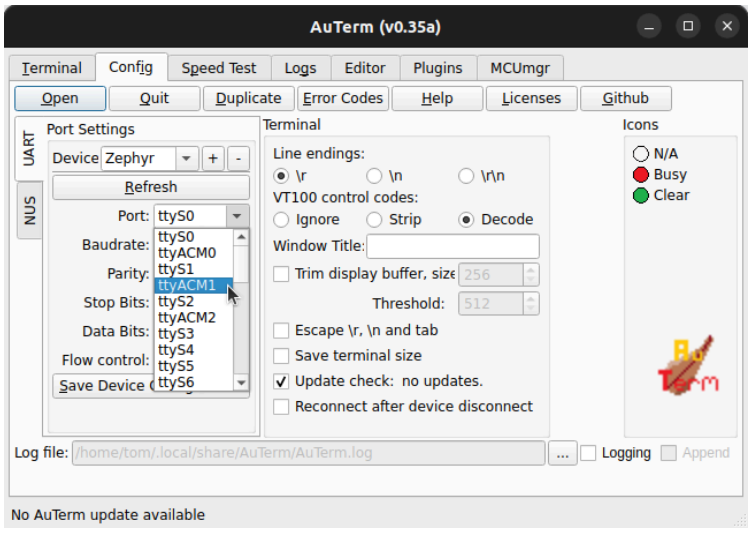
smpclient	Yes	Yes	Yes	Library	Python	Apache-2.0
Zephyr MCUmgr client (in-tree)	No	Yes	No	Library	C	Apache-2.0

For this guide, AuTerm and imgflash will be used (because I find it incredibly entertaining to annoy the rust community). The usage for imgflash can be found later in this guide.

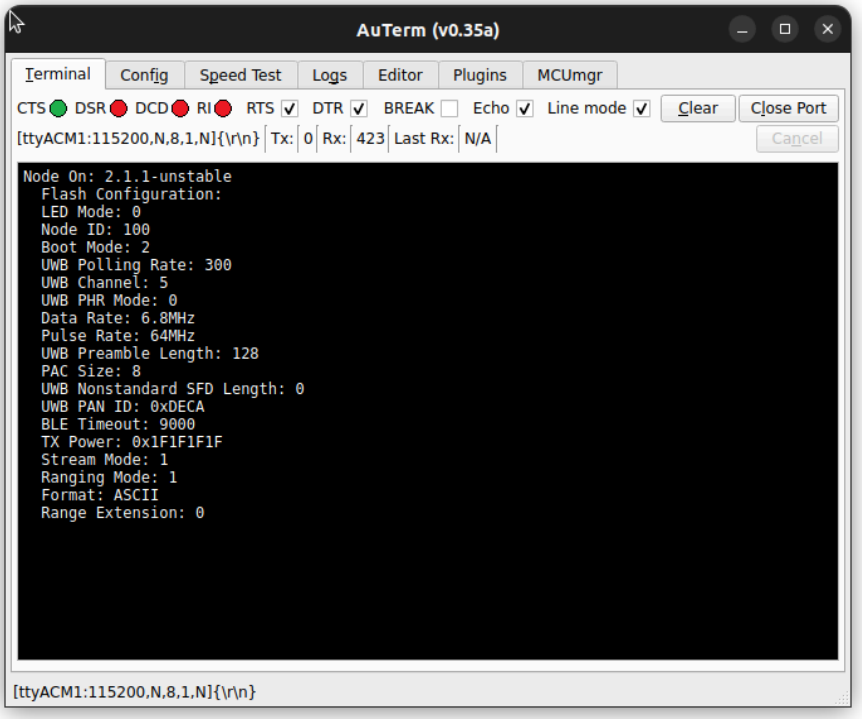
1. Download the latest release of AuTerm from the releases page (At the time of writing, there were no releases, only pre-releases).
    a. Download the build for your operating system



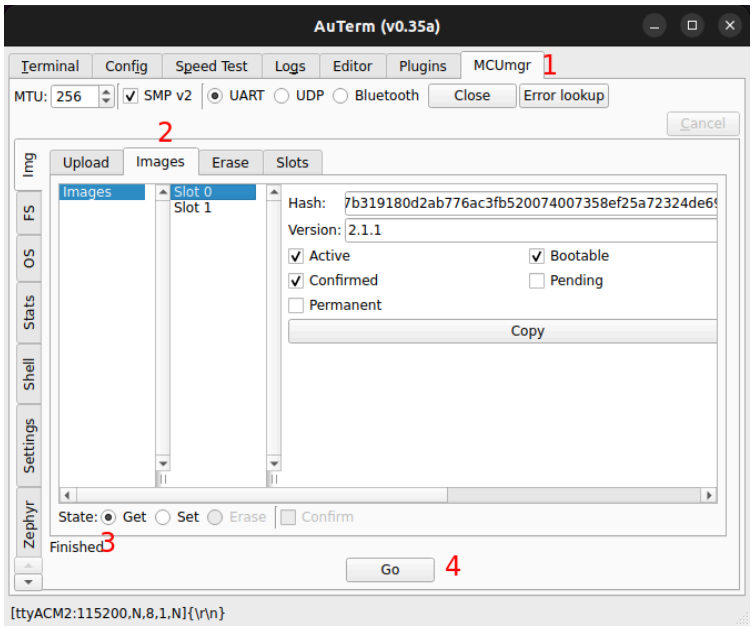
2. Extract the binary
3. Launch AuTerm
    a. For Linux and maybe macOS, you need to mark the binary as executable
4. Press refresh and click on the dropdown list to find the serial port to connect to
    a. On Linux, it should be one of the ttyACM ports
    b. On MacOS, it should be one of the <insert port here> ports



5. Before opening, make sure you have no parity, 115200 baud, 1 stop bit, 8 data bits, and no flow control. Press open once you have those settings.
    a. If you see something like this in the terminal, it means you have the wrong port

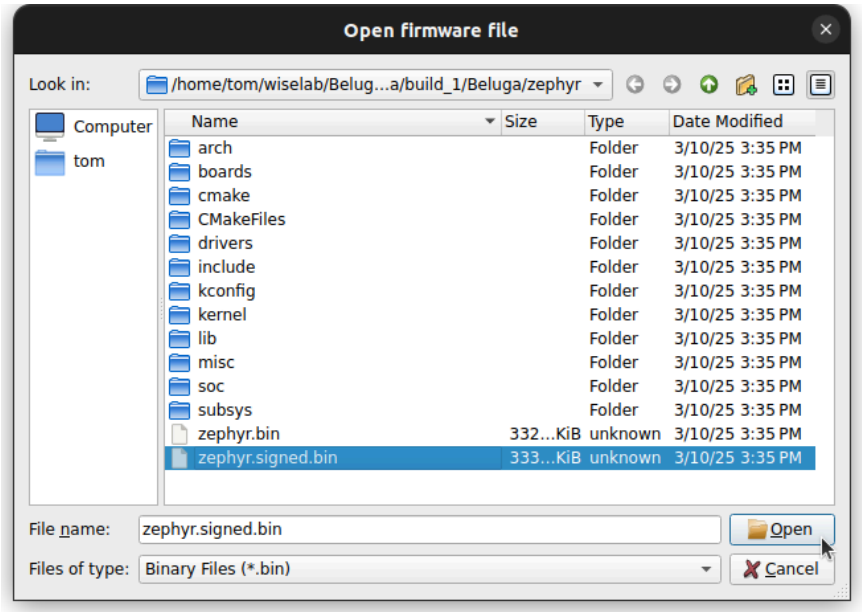


    b. If you see nothing in the terminal, then you can try running “AT+ID” to see if the settings have already been printed somewhere else. If no response is seen, then you may have the correct port.
6. Once you have a potentially correct port follow the instructions below:
    a. Open the MCUmgr tab
    b. Open the images tab
    c. Select state “Get”
    d. Press “Go”

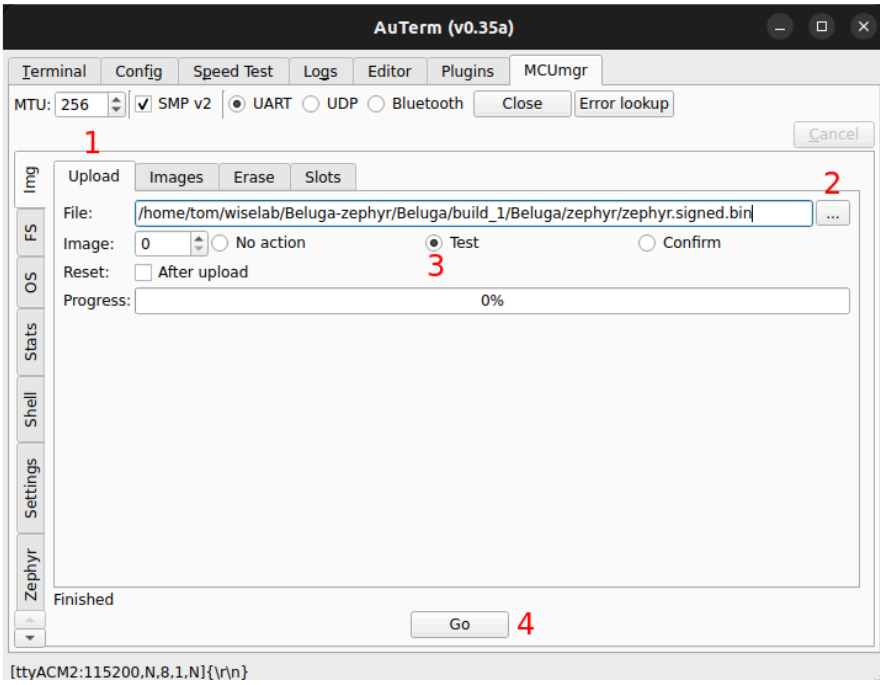


e. If you see images (like the figure above), then you have the correct port

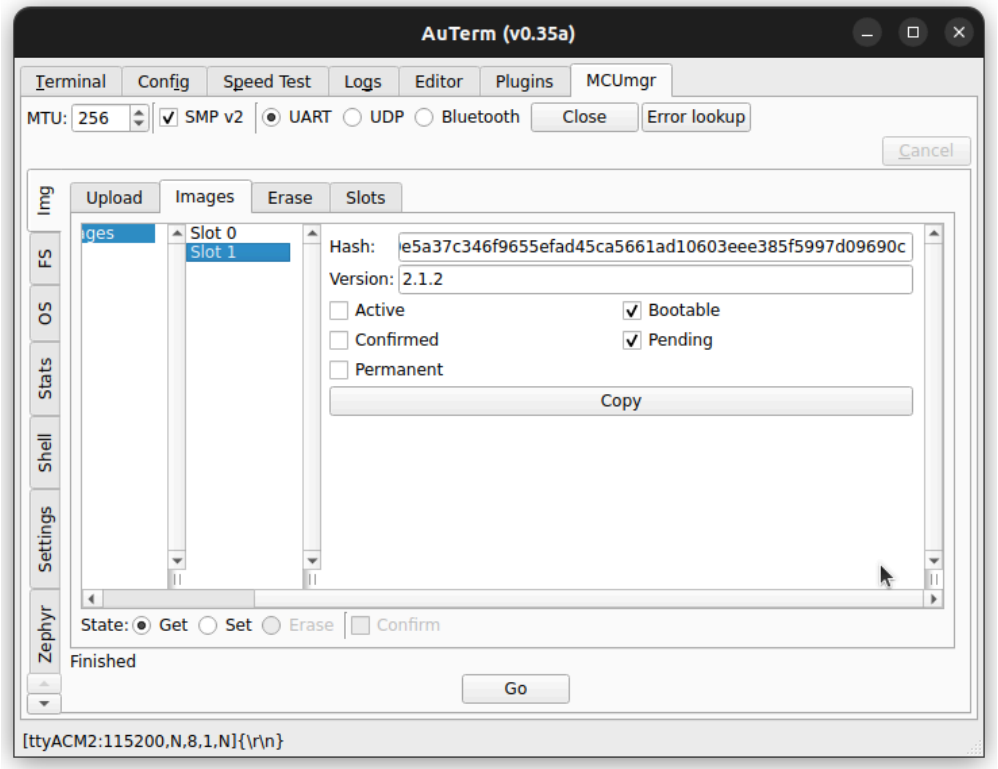
7. Switch to the upload tab and do the following:
    a. Select a binary file to upload
        i. Make sure it is the signed file
        ii. This should be in beluga-2.0/Beluga/build*/Beluga/zephyr



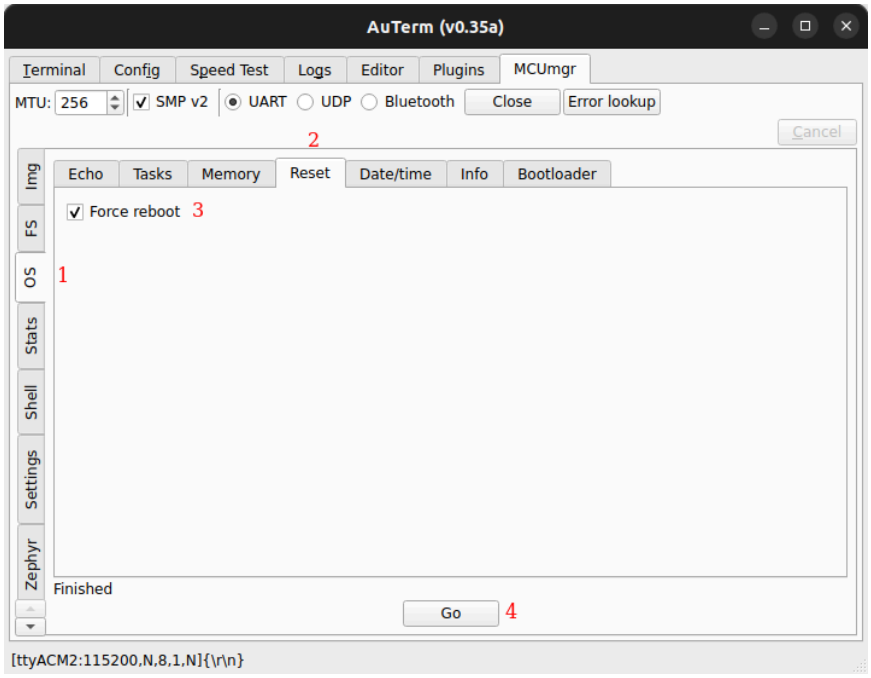
    b. Select “Test”
    c. Press “Go”
        i. Sometimes it will time out. If it does, just retry.



8. Go back to the images tab and rerun the steps in step 6. This time, you should see a pending image.

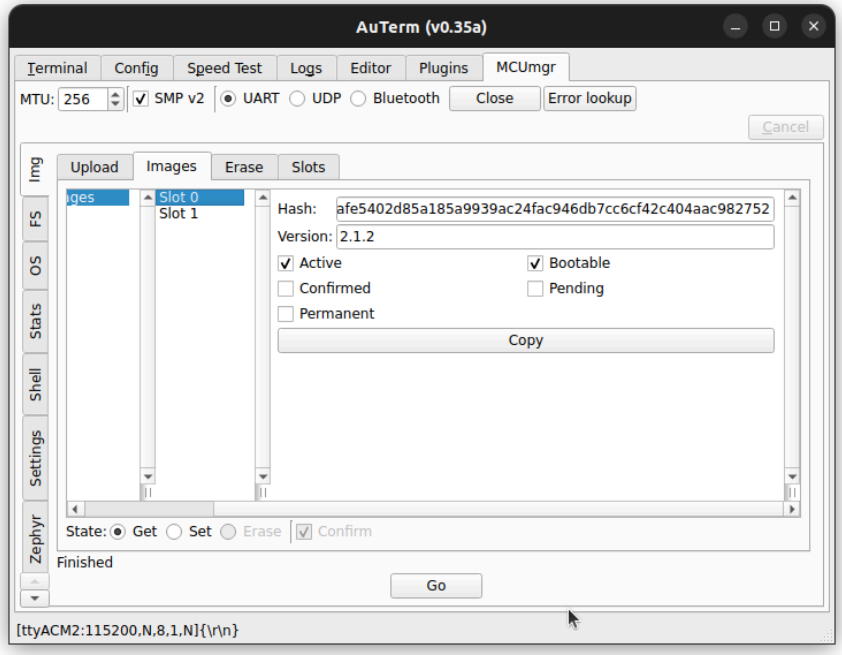


9. Switch to the OS tab
    a. Select the Reset tab
    b. Tick the “Force Reboot” box
    c. Press “Go”

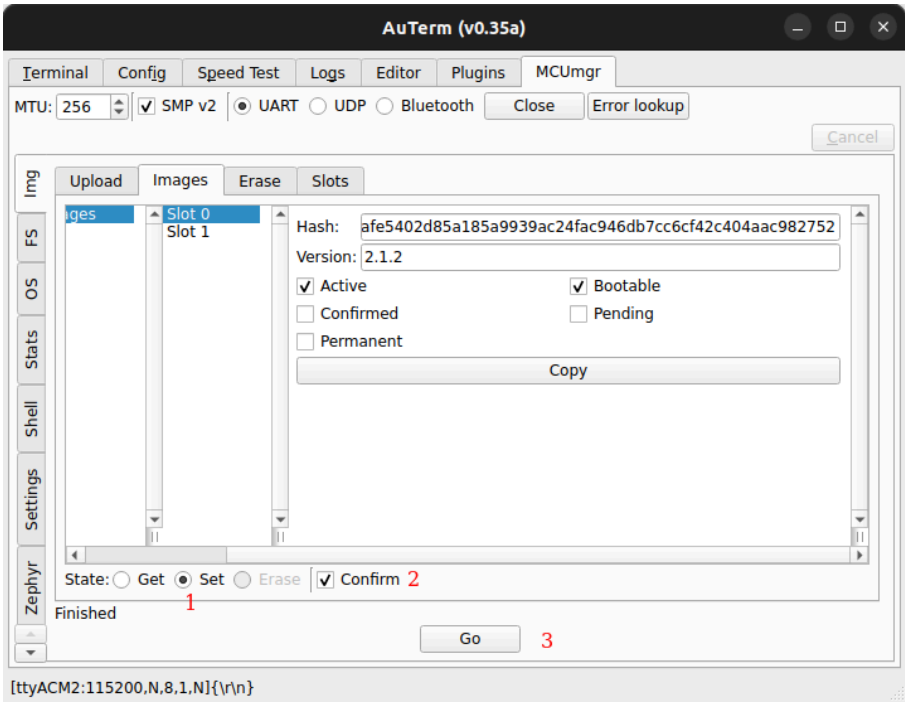


d. The other option is to press the “RESET” button on the hardware

10. Once the device reboots, repeat the steps in step 6 and ensure the new version of the image is running.



11. Once you confirm the image is working correctly, you can confirm it by:
    a. Selecting the “Set” state
    b. Tick the “Confirm” box
    c. Clicking “Go”



d. If the image is faulty, then repeat step 9 to revert back to the original image
    i. If it continues to fail, try using the reset button

DFU Gotcha

The above method was tested and confirmed to work with the custom Beluga hardware. If you do not force a reboot in step 9, then it will ignore the test image. Also, if you use “AT+REBOOT”, it will also ignore the test image.

imgflash

Imgflash is a CLI application developed for flashing devices over USB. It is located within the Beluga tree and can be installed with pip.

Installing

Before installing imgflash, ensure that you meet the following requirements:

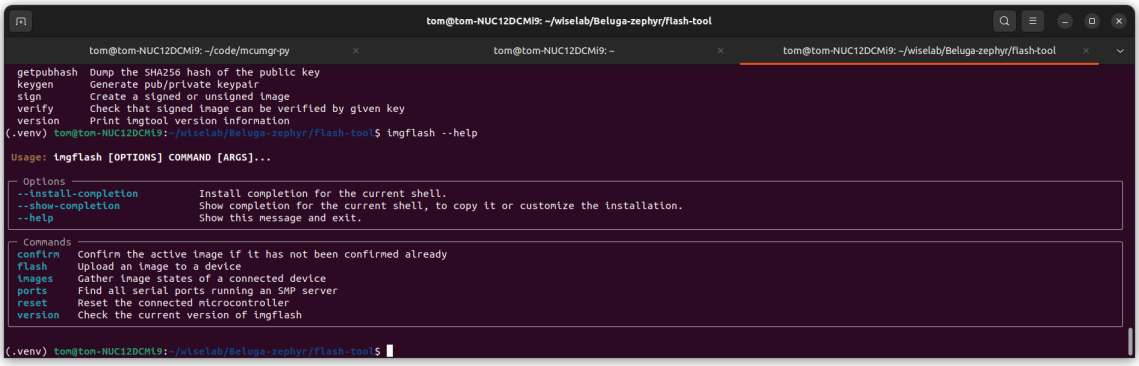
1. Python version >=3.9
2. Beluga repository is cloned

Once you have confirmed the requirements, follow the steps below

1. Open a terminal to the Beluga repository
2. Run the following commands
    a. If on Windows, just google the Windows equivalents to these commands

```
Unset
cd flash-tool
python3 -m venv .venv
source .venv/bin/activate
pip install .
```

3. Confirm that imgflash successfully installed by running “imgflash --help”



Commands

Imgflash has a few commands built already. The simplest command is “imgflash version” which just shows the version of the Python package. The other commands are a little more complex and are described below.

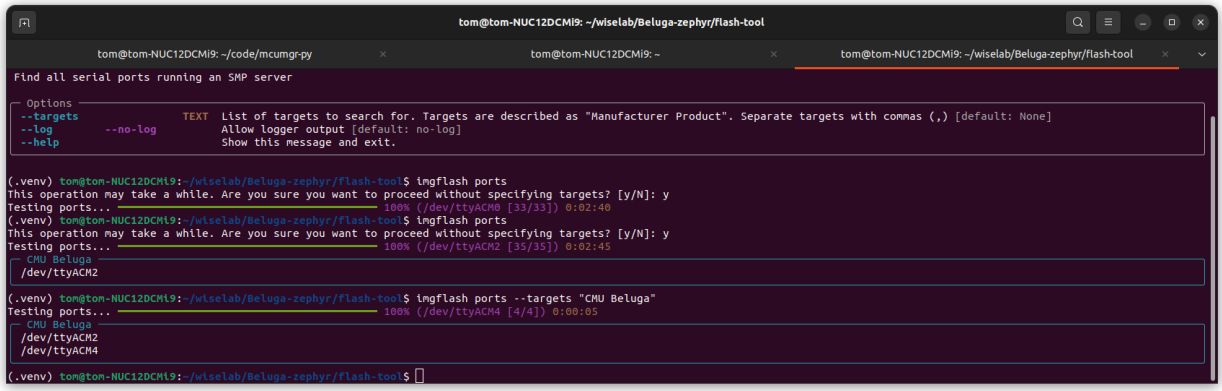
Ports

The ports command scans all of the serial ports on your computer and indicates which ports are running SMP servers. This is important because this application can only communicate with microcontrollers running SMP servers and it makes it easier to identify which ports are running SMP servers.

Options

Option	Type	Description	Default
--targets	TEXT	List of targets to search for. Targets are described as “Manufacturer Product”. The list must be wrapped with quotes and each entry is separated by a comma. If no targets are specified, the application will test each port.	None
--log/--no-log		Turns on/off the logger output	--no-log
--help		Shows the help menu	

Example output



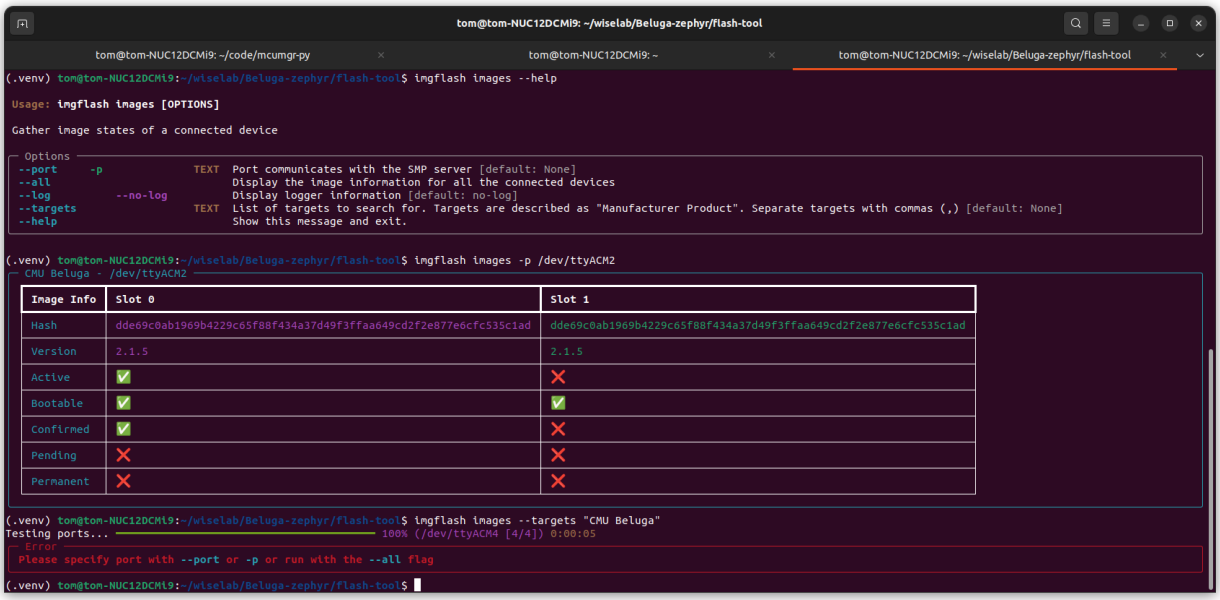
Images

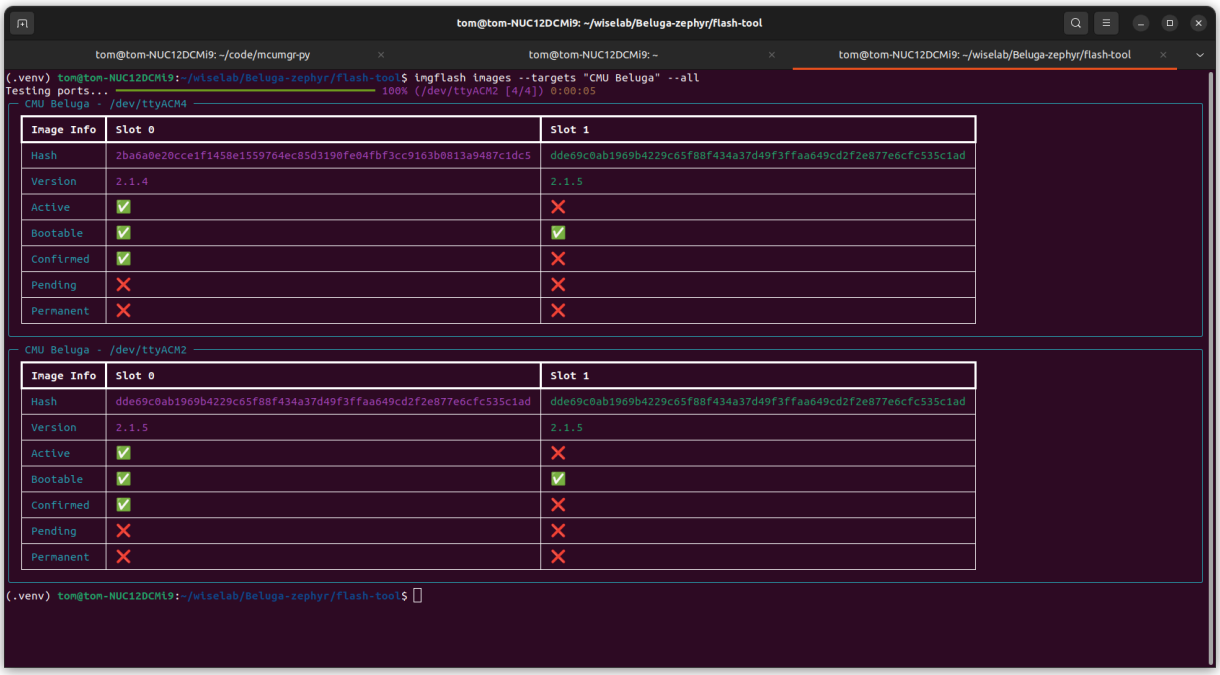
The images command displays the image information on the connected devices. This can be for a specific device, a specific device type, or for all connected devices.

Options

Option	Type	Description	Default
--port/-p	TEXT	Serial port that communicates with the SMP server	None
--all		Display image information for all connected devices if specified	
--log/--no-log		Display logging	--no-log
--targets	TEXT	List of targets to search for. Targets are described as “Manufacturer Product”. The list must be wrapped with quotes and each entry is separated by a comma. If no targets are specified, the application will test each port.	None
--help		Shows help menu	

Example output





Flash

The flash command will upload an image onto the hardware running the SMP server. Unlike the previous commands, this command **requires** a serial port to be specified. Additionally, it requires a build directory and an application name to be specified (Application name is needed because it assumes that sysbuild was used to build multiple applications). If this command fails, make sure that the confirmed image is also the active image. The SMP server will not overwrite the confirmed image.

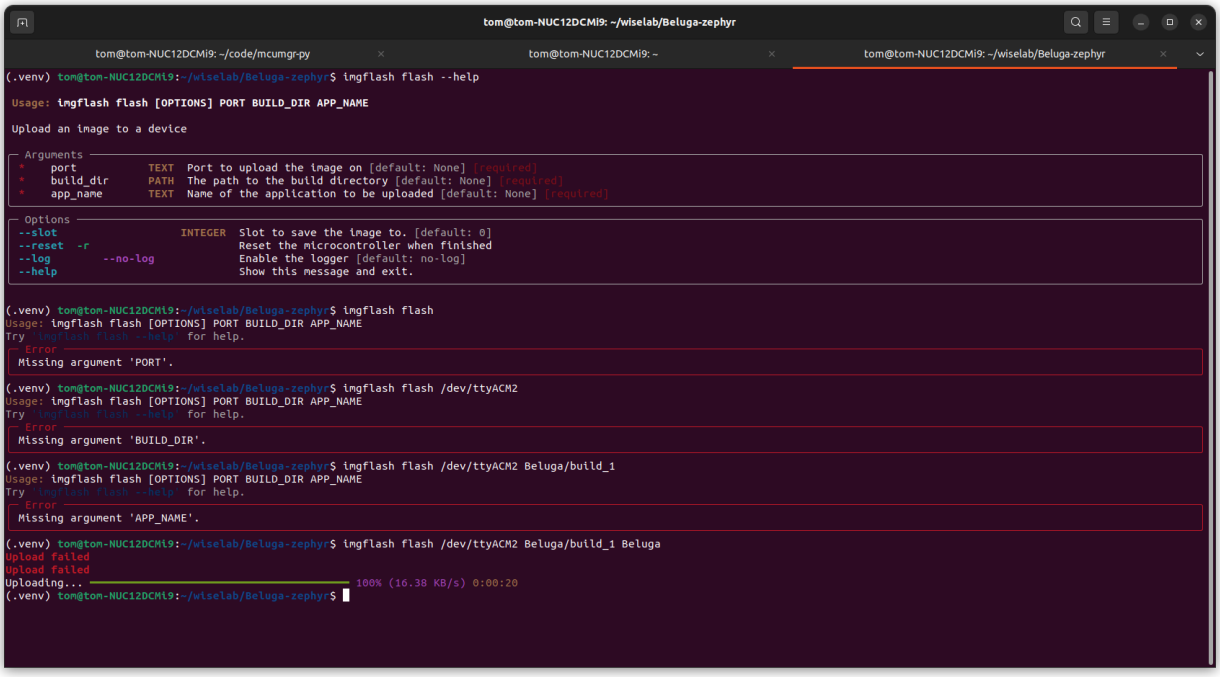
Arguments

Argument	Type	Description	Position
port	TEXT	The port to upload the image on.	1
Build directory	PATH	The file path to the build directory. This can be relative or absolute.	2
Application name	TEXT	The name of the application being uploaded.	3

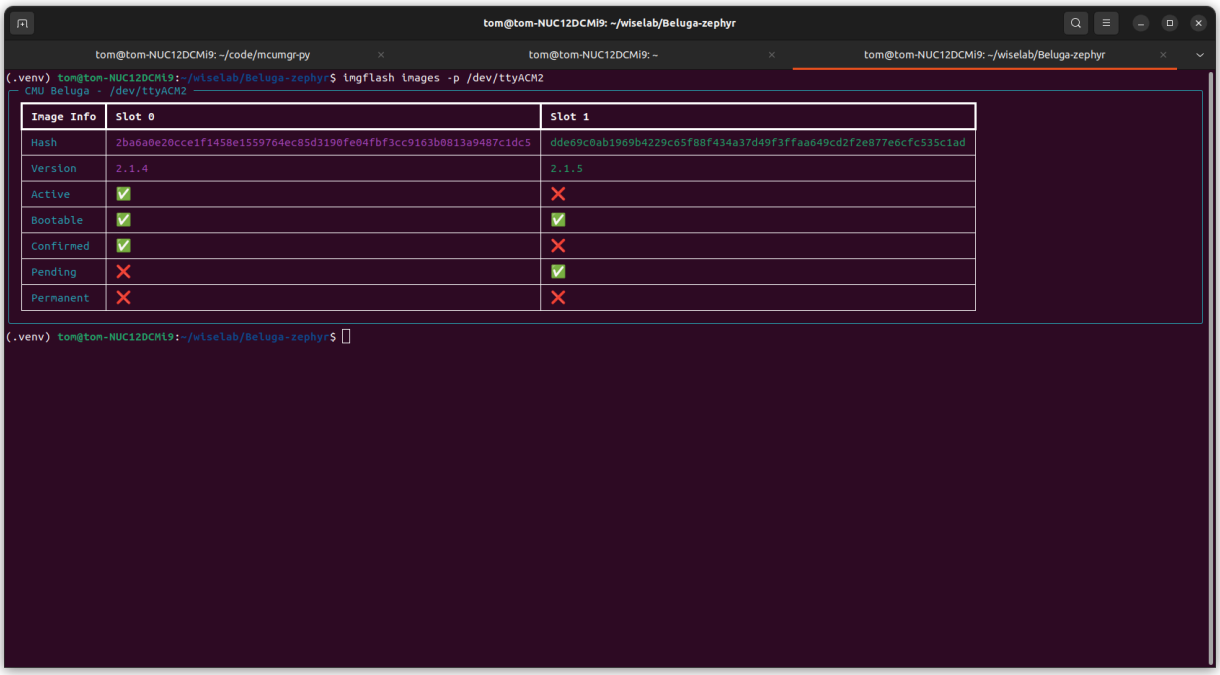
Options

Option	Type	Description	Default
--slot	INTEGER	The slot to save the image to. This is only needed if the hardware needs to run more than 1 image.	0
--reset/-r		Reset the microcontroller after uploading the image.	
--log/--no-log		Enable or disable logging messages	--no-log
--help		Show help message	

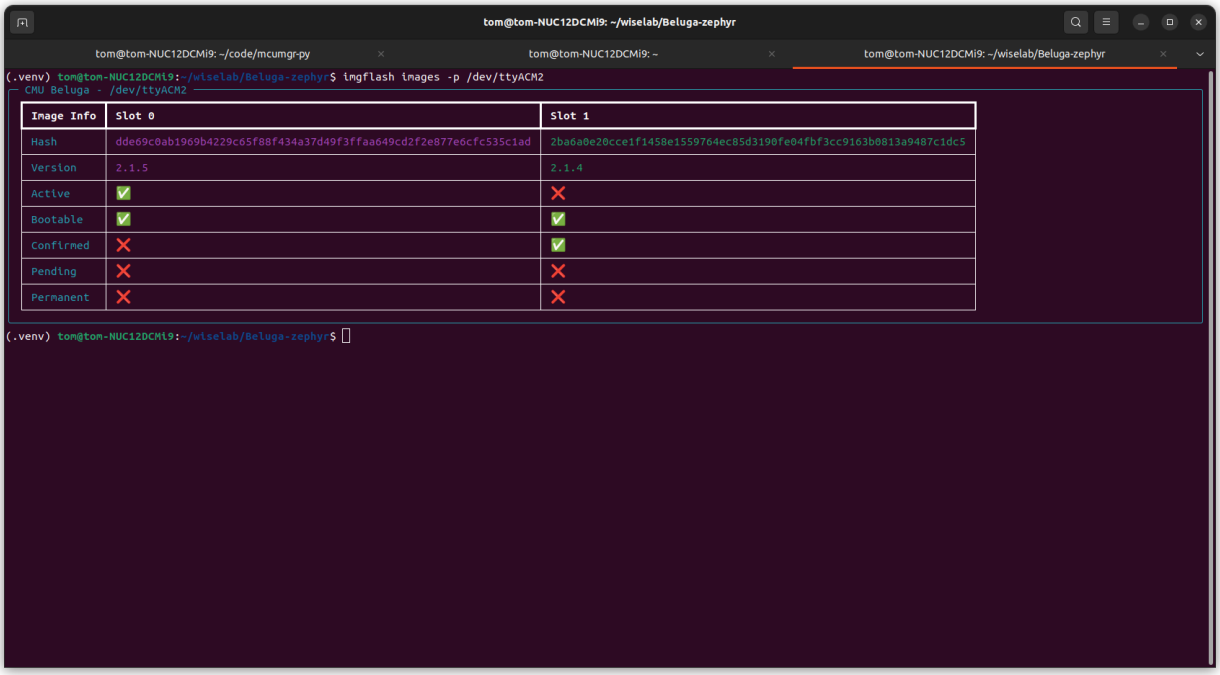
Example output



Images after uploading:



Images after uploading if reset option is specified



Reset

The reset command will reset the microcontroller through the SMP server. This is useful for cases where the reset flag was not specified/forgotten in the flash command or if the test image failed and the old image needs to be restored.

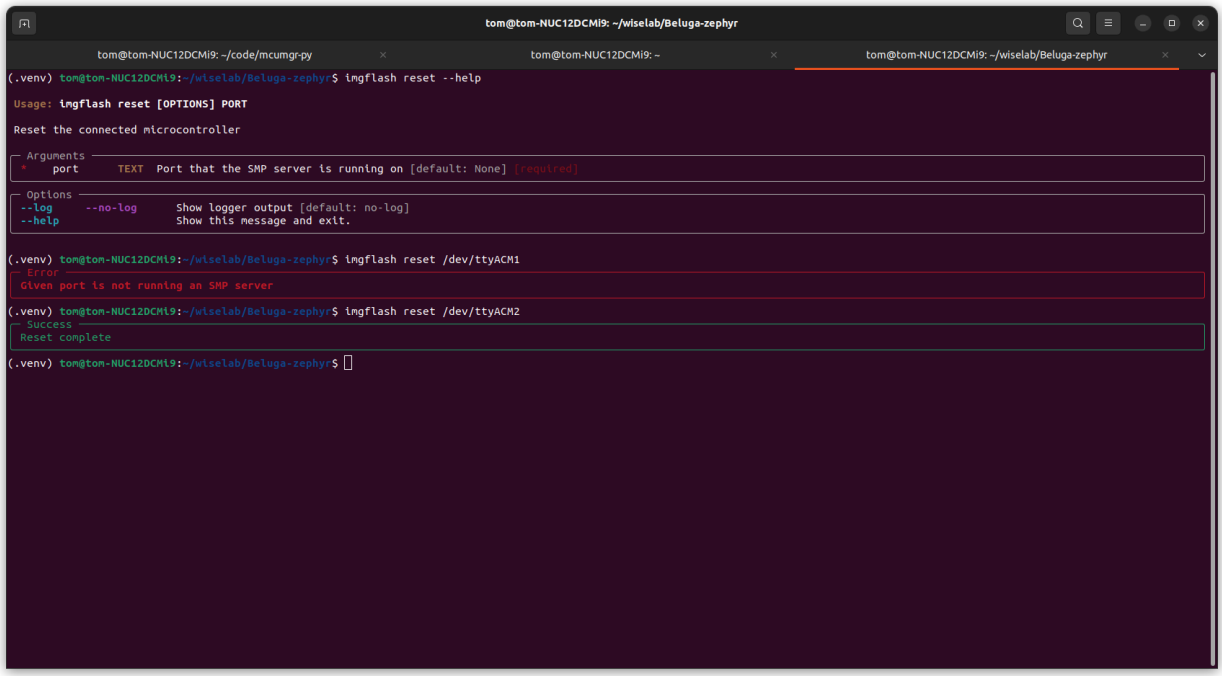
Arguments

Argument	Type	Description	Position
port	TEXT	Port that the SMP server is running on	1

Options

Options	Type	Description	Default
--force/-f		Force reboot	
--log/--no-log		Show logger output	--no-log
--help		Show help menu	

Example output



Notes

Sometimes, reset will not work. I this is the case, then 2 options are available:

1. Press the reset button instead
    a. This is less ideal for situations where there is no human present
2. Confirm the image in slot 1 and rerun the reset command
    a. This will guarantee that the test image will run on the next reboot, however, it risks bricking the microcontroller if the image is faulty

Confirm

The confirm command will confirm the test image that is currently running. If an image has already been confirmed, then it will have no effect.

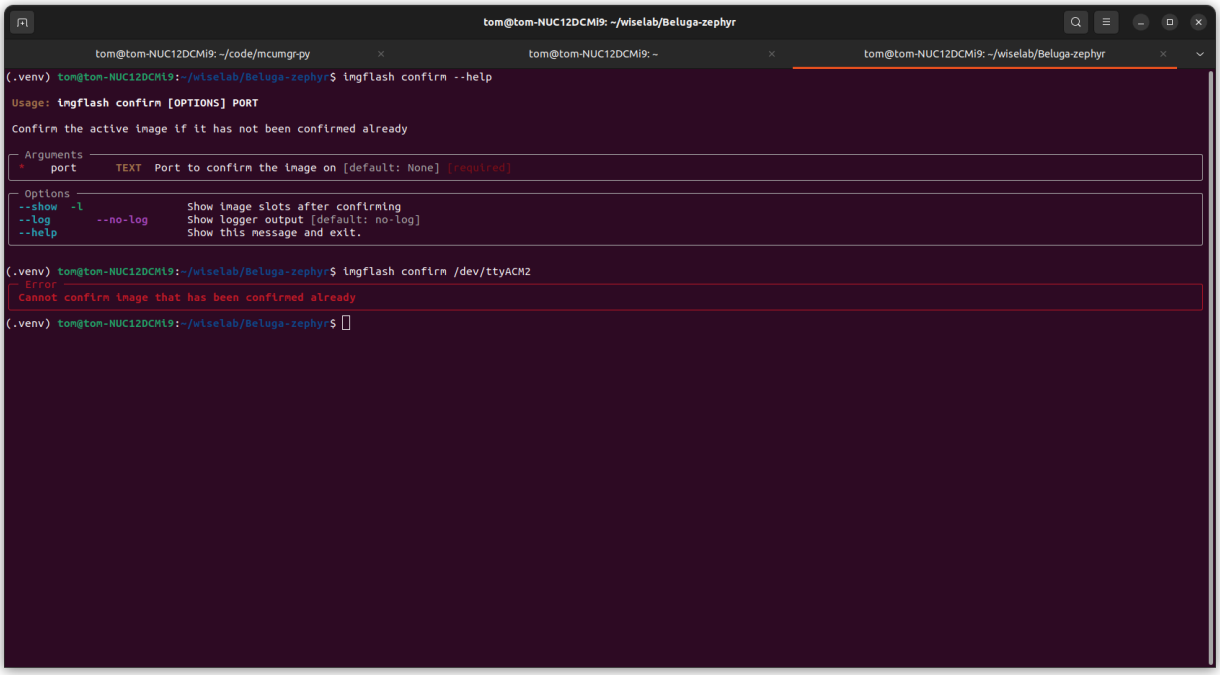
Arguments

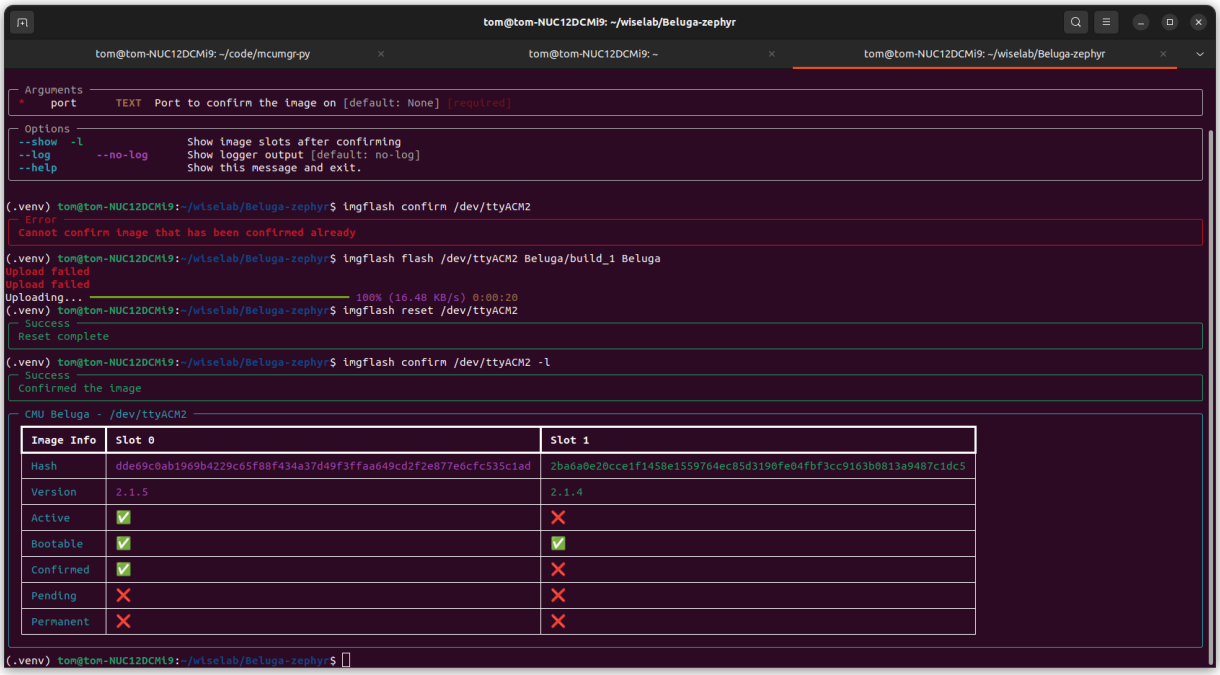
Argument	Type	Description	Position
port	TEXT	Port to confirm the image on	1

Options

Options	Type	Description	Default
--show/-l		Shows the image states after confirming	
--slot	INTEGER	Specifies which image to confirm via the image slot	0
--force/-f		Ignore the running and confirmed checks when marking an image for confirmation. This will not work for non-bootable images	
--log/--no-log		Show logger output	--no-log
--help		Show help menu	

Example output





Swap

The swap command will mark the image in slot 1 as pending. This is useful for instances where the test image fails to boot (this usually happens when the test image is larger than the active image). Instead of reuploading the test image, one can simply mark it as pending again.

Arguments

Argument	Type	Description	Position
port	TEXT	Port the SMP server is running on	1

Options

Options	Type	Description	Default
--show/-l		Shows the image states after confirming	
--log/--no-log		Show logger output	--no-log
--help		Show help menu	

Example Output