Prusa Firmware MK3

This repository contains the source code and the development versions of the firmware running on the Original Prusa i3 MK3S/MK3/MK2.5S/MK2.5 line of printers.

The latest official builds can be downloaded from Prusa Drivers. Pre-built development releases are also available here.

The firmware for the Original Prusa i3 printers is proudly based on Marlin 1.0.x by Scott Lahteine (@thinkyhead) et al. and is distributed under the terms of the GNU GPL 3 license.

This repository contains *development material only!*

Build

Linux

There are two ways to build Prusa-Firmware on Linux: using CMake (recommended for developers) or with PF-build which is more user-friendly for casual users.

CMake

Quick-start

The workflow should be pretty straightforward for anyone with development experience. After installing git and a recent version of python 3 all you have to do is:

```
# clone the repository
git clone https://github.com/prusa3d/Prusa-Firmware
cd Prusa-Firmware

# automatically setup dependencies
./utils/bootstrap.py

# configure and build
mkdir build
cd build
cmake .. -G Ninja -DCMAKE_BUILD_TYPE=Release -DCMAKE_TOOLCHAIN_FILE=../cmake/AvrGcc.cmake
ninja
```

Detailed CMake guide

Building with cmake requires:

- cmake >= 3.22.5
- ninja >= 1.12.1 (optional, but recommended)

Python >= 3.8 is also required with the following modules:

- pyelftools (package `python3-pyelftools`)
- polib (package `python3-polib`)
- regex (package `python3-regex`)

Additionally `gettext` is required for translators.

Assuming a recent Debian/Ubuntu distribution, install the dependencies globally with:

```
sudo apt-get install cmake ninja python3-pyelftools python3-polib python3-regex gettext
```

When using a recent Fedora(non-atomic)/RHEL distribution, install the dependencies globally with:

```
sudo dnf install cmake ninja-build python3-pyelftools python3-polib python3-regex gettext
```

When using a Fedora Atomic/UBlue distribution use `rpm-ostree install --allow-inactive` instead of `sudo dnf install`

Prusa-Firmware depends on a pinned version of `avr-gcc` and the external `prusa3dboards` package. These can be setup using `./utils/bootstrap.py`:

```
# automatically setup dependencies
./utils/bootstrap.py
```

which will download and unpack them inside the `.dependencies` directory. `./utils/bootstrap.py` will also install `cmake`, `ninja` and the required python packages if missing, although installing those through the system's package manager is usually preferred.

You can then proceed by creating a build directory, configure for AVR and build:

```
# configure
mkdir build
cd build
cmake .. -G Ninja -DCMAKE_BUILD_TYPE=Release -DCMAKE_TOOLCHAIN_FILE=../cmake/AvrGcc.cmake

# build
ninja
```

By default all variants are built. There are several ways to restrict the build for development. During configuration you can set:

- `cmake -DFW_VARIANTS=variant` : comma-separated list of variants to build. This is the file name as present in `Firmware/variants` without the final `.h`.
- `cmake -DMAIN_LANGUAGES=languages` : comma-separated list of ISO language codes to include as main translations.
- `cmake -DCOMMUNITY_LANGUAGES=languages` : comma-separated list of ISO language codes to include as community translations.

When building the following targets are available:

- `ninja ALL_MULTILANG` : build all multi-language targets (default)
- `ninja ALL_ENGLISH` : build all single-language targets
- `ninja ALL_FIRMWARE` : build all single and multi-language targets
- `ninja VARIANT_ENGLISH` : build the single-language version of `VARIANT`
- `ninja VARIANT_MULTILANG` : build the multi-language version of `VARIANT`
- `ninja check_lang` : build and check all language translations
- `ninja check_lang_ISO` : build and check all variants with language `ISO`
- `ninja check_lang_VARIANT` : build and check all languages for `VARIANT`
- `ninja check_lang_VARIANT_ISO` : build and check language `ISO` for `VARIANT`

Automated tests

Automated tests are built with cmake by configuring for the current host:

```
# clone the repository
git clone https://github.com/prusa3d/Prusa-Firmware
cd Prusa-Firmware

# automatically setup dependencies
./utils/bootstrap.py

# configure and build
mkdir build
cd build
cmake .. -G Ninja
ninja

# run the tests
ctest
```

PF-build

PF-build is recommended for users without development experience. Download or clone the repository, then run PF-build and simply follow the instructions:

```
cd Prusa-Firmware
./PF-build.sh
```

PF-build currently assumes a Debian/Ubuntu (or derivative) distribution.

Windows

Visual Studio Code (VSCode)

Prerequisites

- Visual Studio Code
- CMake Tools plugin
- Python
- Git Bash

First time setup

Start by cloning the Prusa-Firmware repository

```
git clone https://github.com/prusa3d/Prusa-Firmware
```

Open the `Prusa-Firmware` folder in VScode.

Open a new terminal in VScode (Terminal→New Terminal) and run

```
python .\utils\bootstrap.py
```

This will download all dependencies required to build the firmware. You should see a `.dependencies` folder in the Prusa-Firmware folder.

Reload VScode. If all works correctly you should see the VScode automatically configuring the CMake project for you. If this doesn't happen you likely need to set the CMake kit; This can be done in two ways:

1. Type `Ctrl+Shift+P` and search for `CMake: Select a Kit`. Select `avr-gcc`. If none appear, Scan for kits first.
2. If 1) does not work for some reason, as a last resort you can edit the CMake Tools settings. Search for "Additional Kits" and add `.vscode/cmake-kits.json` to the list.

After updating the kit, you may need to reload VScode.

Building

To start building a firmware, click the CMake Tools plugin icon on the far left side. You will get a very large list of targets to build. Find the firmware you'd like to build (like `MK3S_ENGLISH`) and select the small icon which shows "Build" when hovered over.

The built .hex file can then be found in folder `Prusa-Firmware/build`

Arduino IDE (deprecated)

Using Arduino IDE is still possible, but *no longer supported*. Prusa-Firmware requires a complex multi-step build process that cannot be done automatically with just the IDE. For a long time we provided instructions to use Arduino in combination with shell scripts, however starting with 3.13 the build system has been completely switched to `cmake`.

Building with Arduino IDE results in a *limited* firmware:

- Arduino IDE can only build a single, english-only variant at a time that you manually have to select
- The build will not be reproducible (meaning you will likely get a different binary every time you build the same sources)
- You need to download, patch and select the correct board definitions by hand

For these reasons, you should think twice before reporting issues for a firmware built with Arduino. If you find a bug in the firmware, building and testing using CMake should be your first thought. Issues regarding Arduino builds are answered by the community and are not officially supported.

Environment preparation

Install "Arduino Software IDE" from the official website https://www.arduino.cc -> Software -> Downloads. Version 1.8.19 or higher is required.

Setup Arduino to install and use the Prusa board definitions:

- Open Arduino and navigate to File -> Preferences -> Settings
- To the text field "Additional Boards Manager URLs" add `https://raw.githubusercontent.com/prusa3d/Arduino_Boards/master/IDE_Board_Manager/package_prusa3d_index.json`
- Open Board manager (Tools -> Board -> Board manager)
- Install "Prusa Research AVR Boards by Prusa Research"

Source code preparation

Clone or download this repository to your local drive.

In the subdirectory `Firmware/variants/` select the configuration file (.h) corresponding to your printer model and manually copy it to `Firmware/Configuration_prusa.h`

Run "Arduino IDE", then

- Open the file `Firmware/Firmware.ino`
- Select the target board with Tools -> Board -> "PrusaResearch Einsy RAMBo"
- Open `Firmware/config.h` and change `LANG_MODE` to 0.

Compilation and upload

- Run the compilation: Sketch -> Verify/Compile
- Upload the result code into the connected printer: Sketch -> Upload