You can find these instructions on the official web page.

Overview

TyTools is a collection of **independent tools** and you only need one executable to use any of them. The Qt-based GUI tools are statically compiled to make that possible.

Tool	Type	Description
TyCommander	Qt GUI (static)	Upload, monitor and communicate with multiple boards
TyUpdater	Qt GUI (static)	Simple firmware / sketch uploader

Tool	Type	Description
tycmd	Command-line *No Qt !*	Command-line tool to manage Teensy boards

Download the latest release from GitHub. You can find development builds on koromix.dev.

All the code related to these programs is under **public domain**, you can do whatever you want with it. See the LICENSE file or unlicense.org for more information.

Using tycmd

You can manage multiple devices connected simultaneously, tycmd (and the other tools) uniquely identifies each device by its position in the host USB topology. Meaning if it stays on the same USB port, it is handled as the same device. That's necessary because across reboots and resets, Teensies look completely different to the host.

To target a specific device, use `tycmd <command> --board "[<serial>][-<family>][@<location>]"` . *serial* is the USB serial number, *family* is the board family name and *location* can be the virtual path computed by tycmd (see `tycmd list`) or an OS device path (e.g. /dev/hidraw1 or COM1). Any of them can be omitted. See the examples in the table below.

Tag filter	Effect
714230	Select board with serial number 714230
-Teensy	Select board with family name 'Teensy'
@usb-1-2-2	Select board plugged in USB port 'usb-1-2-2'
@COM1	Select board linked to the OS-specific device 'COM1'
714230@usb-1-2-2	Select board plugged in 'usb-1-2-2' and with serial number is 714230

You can learn about the various commands using `tycmd help` . Get specific help for them using `tycmd help <command>` .

List devices

`tycmd list` lists plugged Teensy devices. Here is how it looks:

```
add 34130@usb-1-2 Teensy 3.1
add 29460@usb-4-2 Teensy
add 32250@usb-4-3 Teensy 3.0
```

Use `--verbose` if you want detailed information about available devices:

```
add 32250@usb-4-3 Teensy 3.0
  + capabilities:
    - upload
    - reset
  + interfaces:
    - HalfKay Bootloader: /dev/hidraw2
```

If you need to read structured information in your scripts, you can set the output to JSON with `--output json`:

```
{"action": "add", "tag": "714230@usb-6-3", "serial": 714230, "location": "usb-6-3", "model": "Teensy", "capabilities": ["reboot", "serial"], "interfaces": [["Seremu", "/dev/hidraw4"]]}
{"action": "add", "tag": "1126140@usb-6-2", "serial": 1126140, "location": "usb-6-2", "model": "Teensy LC", "capabilities": ["upload", "reset"], "interfaces": [["HalfKay Bootloader", "/dev/hidraw3"]]}
```

You can also watch device changes with `--watch`, both in plain and JSON mode.

Action	Meaning
add	This board was plugged in or was already there
change	Something changed, maybe the board rebooted
miss	This board is missing, either it was unplugged (remove) or it is changing mode
remove	This board has been missing for some time, consider it removed

Upload firmware

Use `tycmd upload <filename.hex>` to upload a specific firmware to your device. It is checked for compatibility with your model before being uploaded.

By default, a reboot is triggered but you can use `--wait` to wait for the bootloader to show up, meaning tycmd will wait for you to press the button on your board.

Serial monitor

`tycmd monitor` opens a text connection with your Teensy. It is either done through the serial device (/dev/ttyACM*) or through the HID serial emulation (SEREMU) in other USB modes. tycmd uses the correct mode automatically.

You can use the `--reconnect` option to detect I/O errors (such as a reset, or after ab rerief unplugging) and reconnect immediately. Other errors will exit the program.

The `--raw` option will disable line-buffering/editing and immediately send everything you type in the terminal.

See `tycmd help monitor` for other options. Note that Teensy being a USB device, serial settings are ignored. They are provided in case your application uses them for specific purposes.

Reset and reboot

`tycmd reset` will restart your device. Since Teensy devices (at least the ARM ones) do not provide a way to trigger a reset, tycmd will instead start the bootloader first and then issue a reset without programming anything.

You can also use `tycmd reset -b` to start the bootloader. This is the same as pushing the button on your Teensy.

Hacking TyTools

Build on Windows

You can use MSVC (≥ 2015) or MinGW-w64. I have not tested Clang on Windows yet. The historical MinGW toolchain is not supported.

You need to install CMake to build the project files before you can use Visual Studio or MinGW.

If **you don't have Qt, only the libraries and command-line tools** will be compiled. There are two options to enable Qt in TyTools:

- *dynamic Qt build*: Easiest option, you only need to install the pre-built Qt DLLs for your compiler with the official Qt installer. The binaries need the Qt DLLs to work.
- *static Qt build*: Harder, but will produce autonomous binaries. Read the README in the lib/qt5 directory for instructions.

To build TyTools with MSVC 2015 32-bit, launch *VS2015 x86 Native Tools Command Prompt*, navigate to the project directory and execute CMake:

```
REM You can of course use another build directory if you prefer.
mkdir build
mkdir build/win32
cd build/win32
cmake ../..
```

This will create Visual Studio project and solution files in *build/win32*. You can then open the *solution TyTools.sln*.

To build 64-bit binaries, you should use *VS2015 x64 Native Tools Command Prompt* instead.

Build on Mac OS X

Install Xcode, the developer command-line tools and CMake. The native Clang compiler can build TyTools.

If **you don't have Qt, only the libraries and command-line tools** will be compiled. There are two options to enable Qt in TyTools:

- *dynamic Qt build*: Easiest option, you only need to install the pre-built Qt DLLs for your compiler with the official Qt installer. The binaries need the Qt DLLs to work.
- *static Qt build*: Harder, but will produce autonomous binaries. Read the README in the lib/qt5 directory for instructions.

After Qt is ready, you can build TyTools by executing the following commands in the project directory:

```
# You can of course use another directory if you prefer.
mkdir -p build/darwin && cd build/darwin
cd build/darwin
cmake ../..
```

If you want to build debug binaries instead, you should specify the build type:

```
cmake -DCMAKE_BUILD_TYPE=Debug ../..
```

Build on Linux

TyTools can be built with GCC or Clang.

Prerequisites

To install the dependencies on Debian or Ubuntu execute:

```
sudo apt-get install build-essential cmake libudev-dev qtbase5-dev pkg-config
```

On Arch Linux you can do so (as root):

```
pacman -S --needed base-devel cmake udev qt5-base
```

Compilation

Open the project directory in a terminal and execute:

```
# You can of course use another directory if you prefer.
mkdir -p build/linux && cd build/linux
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../..
make
```

If you want to build debug binaries instead, you should specify the build type:

```
cmake -DCMAKE_BUILD_TYPE=Debug ../..
```

The compiled binaries can be used directly from the build directory. Follow through the next section if you want to install the application.

Installation

You can deploy TyTools to your system with the following commands:

```
sudo make install
```

By default this will copy the files to `/usr/local`. To change this directory you need to change the `CMAKE_INSTALL_PREFIX` value in the Compilation section above.