Crazyflie Adventures: 2.0 Edition

Michael Saybolt, Joe Fabbo

5/4/2017

Contents

1 Introduction **4**
- 1.1 Motivation . . . 4
- 1.2 Development Background . . . 4

2 Setup **5**
- 2.1 Software Package Requirements . . . 5

3 Procedure for Basic Development **5**
- 3.1 Setting up Python Virtual Environment . . . 6
  - 3.1.1 Git Overview . . . 6
  - 3.1.2 Environment Management . . . 6
  - 3.1.3 Download Crazyflie Source . . . 7
- 3.2 The easy way . . . 8
  - 3.2.1 Tangent: brief discussion of virtual environments . . . 8
  - 3.2.2 Run the CF Client . . . 9
- 3.3 The slightly more complicated way . . . 9
  - 3.3.1 Create empty virtual environment . . . 9
  - 3.3.2 Install dependencies . . . 10
  - 3.3.3 For the curious onlooker . . . 10
- 3.4 Setting USB Device Permissions . . . 11
  - 3.4.1 Rules: Crazyradio PA . . . 11
  - 3.4.2 Rules: Crazyflie 2.0 . . . 12
  - 3.4.3 Additional Information . . . 12

4 I Broke It **12**

5 To Do **13**
- 5.1 To Do: In relation to ECE 813 Cyber Physical Systems project 13

6 Software Challenges Encountered in this Project **13**
- 6.1 Software Summary . . . 14

7 CPS Project Navigation **14**
- 7.1 Initial Approach . . . 14
- 7.2 Inertial Navigation . . . 14
- 7.3 Hovering . . . 15
- 7.4 Hardware Issues . . . 15

7.5 Troubleshooting 15
7.5.1 Channel Interference 15
7.5.2 Bluetooth Connection 15
7.5.3 Virtualbox Settings 16

1 Introduction

1.1 Motivation

- Why Crazyflie?
  - Modular, open source development platform
  - Very accurate sensors
  - Active development community
  - Scattered documentation, but this presentation and report aim to fix that

1.2 Development Background

- Crazyflie works on Linux, Windows, and Mac
- Most developers use Linux
- VM available for quick setup but native environment works better
  - Virtualbox USB driver issues
  - VM software introduces variables CF devs can't control
- This was tested on Ubuntu 16.04 LTS AMD64
- Crazyflie Client 2016.4 (though new versions should work)

This guide covers Linux setup for use and development. For Windows use, just download the compiled binary from the Bitcraze Github/website. Windows development is believed to happen through the Bitcraze VM, containing Linux. Running a VM introduces additional variables with the virtualization software out of control of the developers maintaining the VM. Therefore, running Linux natively on your machine is strongly suggested, even if you are new to Linux. If you would like to use a drone as advanced and sophisticated as the Crazyflie, you should have/learn the limited Linux proficiency required in this guide.

It is recommended that you skim the procedure before starting. It is targeted towards someone who may be new to or intimidated by Linux, so at times may be wordy, but to do any serious development, native Linux is strongly recommended by this guide, either dual booted or installed alone on your hard drive.

2 Setup

2.1 Software Package Requirements

- Git
  - To download and maintain Crazyflie programs
- Python virtual environment
  - Dependency management
    - Anaconda virtual environment
    - virtualenv
- Docker (optional, not covered in this guide)
  - Vitual environment outside of Python
  - Used for installation and use of Bitcraze toolbelt
  - Ensures same compilers are used for compiling Windows binaries, potentially Android/iOS, or firmware
  - Shouldn't be needed for development of CF Client
  - For more information see:
  - https://github.com/bitcraze/toolbelt

3 Procedure for Basic Development

The following procedure assumes you have Git installed and working (comes on most Linux distros) and will proceed as follows:

- Fork/clone crazyflie-clients-python from GitHub
- Set up Python virtual environment
- Install necessary dependencies
- Set udev permissions
- Run Crazyflie client, you can modify the source code
- (Optional) Install Docker/toolbelt

3.1 Setting up Python Virtual Environment

3.1.1 Git Overview

All version control for this project is done with Git. It is accessible from the command line and some proficiency with basic linux commands such as *cd* to change directory, *ls*, to list the contents, and *pwd* to print the working directory are assumed. If this is your first time using Github repositories, it is recommended that you create a Github folder in your home folder, containing the usernames of repo owners, containing the individual repositories.

For example, open a terminal and change to your home directory:

```
cd /home/yourUsername/
```

You should not see any change because by default, your teminal should open to your home directory.

```
yourUsername@yourHostname:~$
```

The tilde is a shortcut for home.

3.1.2 Environment Management

Most of the documentation states to use virtualenv. Virtualenv Contains dependencies installed with pip, a Python package manager in an isolated environment. Pip struggles to install some dependencies needed to run the CF client.

Solution: use Anaconda/Miniconda

Anaconda is a Python suite including Python, conda, a python package manager, and many useful Python packages. Conda package manager handles finnicky package installs better than pip. Conda also has an implementation of virtual environments that can handle packages installed with conda as well as pip. Some preconfigured packages in Anaconda don't play nice with the required ones for cfclient. Miniconda contains just Python and conda

Miniconda can be used to setup 'naked' conda virtual environments and install only what is needed and works. Download and install latest Miniconda setup. The installer should add an alias to the 'conda' command in your .bashrc. This lets you run conda from any folder by typing 'conda'

```
username@Hostname:~$ conda
```

It should return:

```
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications,
environments and packages.
```

with some additonal text, if it is installed correctly.

3.1.3 Download Crazyflie Source

Download the crazyflie source by navigating to your Github folder:

```
cd ~/Github
```

If it does not exist, create it:

```
mkdir ~/Github
```

(you can leave off the

```
~/
```

if you are already in home folder).

Following suggested heirarchy, make a bitcraze folder inside:

```
mkdir bitcraze
```

Clone (download/sync up to) the repository:

```
git clone https://github.com/bitcraze/crazyflie-clients-python.git
```

3.2 The easy way

Since all the heavy lifting of sifting through documentation and figuring out what works and what doesn't has been done already, the installation is extremely straightforward. Using conda is so nice, because what works and what doesn't can be summed up into a nice bulleted list... **which can be read by conda and used to create a new environment and download/install all dependencies in one line!** Simply clone this GitHub repository into a suitable folder:

```
git clone https://github.com/sayboltm/ECE813.git
```

Navigate to your cloned folder:

```
cd /path/to/GitHub/sayboltm/ECE813/environments/
```

It contains a file, crazyflie.yml. Feel free to open it and see the neat bulleted list that will be interpreted by conda to build your environment.

Create a conda virtual environment from the file (-f), install the cflib and other dependencies:

```
conda env create -f crazyflie.yml
```

Install to conda virtual environment 'crazyflie' is complete. This is not the latest version, but it is known to work, and should always work because it pulls specific versions of only what it needs. If an update is needed, it is recommended to simply create a new environment and test the new version in there.

3.2.1 Tangent: brief discussion of virtual environments

To activate an environment, in this case one called 'crazyflie', type:

```
 source activate crazyflie
```

Your terminal will go from:

```
 yourUsername@yourHostname:~$
```

To:

```
 (crazyflie)yourUsername@yourHostname:~$
```

to signify that the environment is active. Typing

```
conda list
```

will list all python packages installed in the current environment.

To deactivate:

```
source deactivate
```

but be sure to be in the environment when using or developing things that depend on these packages.

3.2.2 Run the CF Client

You can now run the Crazyflie Client by navigating to the cloned repo:

```
cd /path/to/Githubfolder/bitcraze/crazyflie-clients-python/
```

and running the binary inside the bin folder:

```
python bin/cfclient
```

3.3 The slightly more complicated way

Instead of creating a conda virtual environment from a file, this will create an empty conda virtual environment, and then what is needed is installed.

3.3.1 Create empty virtual environment

Create a conda virtual environment for the CF Client install and development. Name it something unambiguious, like 'crazyfliedev'.

```
conda create -n crazyfliedev
```

Your conda empty virtual environment is now ready for dependency installation.

3.3.2 Install dependencies

Activate the virtual environment if not already active:

```
source activate crazyflie_dev
```

Download the python package manager, pip:

```
conda install pip
```

Note that if you use alternate channels for conda, like conda-forge or something, there have been errors when using packages from there. Use standard conda channels. If using conda-forge, remove with:
`conda config --remove channels conda-forge`
Else if this is your first time using conda/git, don't worry about this.

Navigate to the cloned CF Client repo if not already there:

```
cd /path/to/Github/bitcraze/crazyflie-clients-python/
```

Use pip to install the dependencies from the cloned repo, in editable mode (-e flag) by running pip install in the current directory (./):

```
pip install -e ./
```

Pip doesn't like to install pyqt, a python wrapped C-based library for making cool GUIs. But conda can.

```
conda install pyqt
```

Run the CF Client by navigating to its directory and running it as described in section *Run the CF Client.*

3.3.3 For the curious onlooker

To recap, the easy version has conda install all dependencies from the crazyflie.yml markup file that happens to be nice and human readable. To ensure future compatibility, this installs specific versions of the dependencies, but was modified to pull the cflib from github at a spefic commit, because that was the version that was installed when the environment was created. Essentially, it creates an environment with a specific version of crazyflie client and cflib, and installs those dependencies that work with it. For more information, learn more about Github version control.

3.4 Setting USB Device Permissions

In Linux, access to USB devices requires root (admin) privledges. It is bad practice to use root unless absolutely necessary. Crazyflie development is not a good reason. Setting udev permissions allows access to usb devices for your username without root permissions.

You can view what groups your user is a part of by typing:

```
groups yourUsername
```

Create group plugdev if it is not already created, and **a**ppend yourUsername to that **G**roup:

```
sudo groupadd plugdev
sudo usermod -a -G plugdev yourUsername
```

Note: Using superuser to do (sudo) this is ok here, and generally required to change system level settings like this.

3.4.1 Rules: Crazyradio PA

Now you must tell the system how to identify a specific USB device (the Crazyradio PA, and Crazyflie 2.0 if you want to plug it in) for access by the users in group plugdev. This is done by creating the following rules. If the file or folder does not exist, create it. For the Crazyradio PA, there must be a file in:

```
/etc/udev/rules.d/
```

called

```
99-crazyradio.rules
```

Make folders and files as necessary, cd into the rules.d folder, and open a text editor like nano or vim in the terminal. Paste the following into the file:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="1915", ATTRS{idProduct}=="7777",
MODE="0664", GROUP="plugdev"
```

Save it. Note to save in this folder you will have to open the text editor as root, prefixing the command with sudo. This is the same as dragging a text file into that folder with that text, but it is easier to just open the text editor while in that directory and paste in the text.

3.4.2 Rules: Crazyflie 2.0

There also must be a rule to uniquely identify the Crazyflie. Make a file, again in:

```
/etc/udev/rules.d/
```

but this time called:

```
99-crazyflie.rules
```

containing the following text:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
MODE="0664", GROUP="plugdev"
```

3.4.3 Additional Information

If the crazyradio is not detected by the CF Client, look here and make sure everything is typed correctly and in the correct location. These rules are unique to these USB devices so they will not work with crazyflie 1.0.

4 I Broke It

- Adding or upgrading dependencies can break others
- Importance in using virtual Python environments
- Easy to save, recreate old environment
- Recommended that you create a new virtual environment when upgrading
- Pull the latest Crazyflie client, then use pip to install any new dependencies

  ```
  cd /path/to/crazyflie-clients-python/

  git pull

  pip install -e ./
  ```

5 To Do

There is a Toolbelt and Docker install necessary for compiling Windows binaries, but this was difficult to follow and did not seem necessary for Linux use and dev.

5.1 To Do: In relation to ECE 813 Cyber Physical Systems project

- Fix Toolbelt setup
- Do some development with the CF Client
  - ZMQ backend allows for external input to the CF Client without needing to understand much of the CF Client source code
  - Source code is Python so easy to read anyways
  - Implement supplemental reactive learning
  - Implement local positioning project

6 Software Challenges Encountered in this Project

- Python wrapped C libraries are finnicky
  - E.g. QT
  - Updating QT for latest CFClient causes your Python-based IDE to segfault (underlying C implementation issue)
- Different versions of Python can also create issues
  - py2exe not supported on Python $>$ 3.4
  - QT5 requires Python $>$ 3.4
  - Latest CFClient requires QT5
  - Conflict breaks environment used for other projects/classes
  - Must fix Crazyflie dev environment and tools for other coursework, find way for them to coexist
- Hence need for easy to use virtual environments

    - conda virtual environments > virtualenv

- Documentation is scattered, lots of trial and error
    - Still recommends virtualenv
    - Conda virtual environments are superior
    - Conda > pip for package management
    - Conda env handles conda and pip managed packages

6.1 Software Summary

At the very end, hours of research, trial and error were condensed into a conda environment configuration markup file leading to near instant deployment of crazyflie development environments without needing the finnicky VM setup. Although not much actual development was actually done, solving these problems encountered and using this containment system is hugely important for future development, and to lower the barrier of access for prospective developers, keeping the Crazyflie development community active and growing.

This was done while dodging many perceived hardware bugs, none of which were encountered after switching to Linux.

7 CPS Project Navigation

7.1 Initial Approach

The initial goal for this project was to achieve stable autonomous flight using a proprietary local positioning system. Two potential options that were considered were using the proprietary Loco positioning system and the use of acoustic beacons. The Loco positioning was too expensive and was abandoned. The CrazyflieâĂŹs limited battery life made acoustic positioning impractical.

7.2 Inertial Navigation

An attempt was made to navigate using only the drones onboard sensors; however, progress developing this approach was impeded by communication issues between the control PC and the drone. The project goals were reformulated to achieving autonomous hovering.

7.3 Hovering

The Crazyflie firmware has provisions for PID controllers. During controlled flight the user has the ability to activate an altitude hold mode via a controller input. Attempts were made to discover and emulate these control signals; however, connection issues continued to impede progress.

7.4 Hardware Issues

Throughout the project, connection issues made the connection between the drone and the radio dongle unreliable. The exact behavior of this issue varied, but the core issue was that excess packet loss would eventually break the connection. To reconnect the radio dongle would have be unplugged then reconnected to the control PC.

7.5 Troubleshooting

7.5.1 Channel Interference

The active communication channel and bandwidth were changed to avoid any potential interference. The channel was set too channel 49 and the bandwidth was set to 2Mbit/s While this did improve connection latency, it did not resolve the core issue.

7.5.2 Bluetooth Connection

Posts on the developerâĂŹs forum suggested that the CrazyflieâĂŹs Bluetooth connection could interfere with the connection to the radio dongle. An attempt was made to disable the Bluetooth connection. This was accomplished by editing make file parameters and rebuilding and re-flashing the communication MCUâĂŹs firmware. While this did disable the Bluetooth connection it did not solve the connection issues. The Bluetooth connection was eventually re-enabled using the terminal to navigate to the firmware directory and running the following commands: ake BLE=1 all, make clean, and make cload.

7.5.3 Virtualbox Settings

After an update, a setting in Virtualbox to support USB 3.0 was found. The PC that we were developing on had one USB 2.0 and two USB 3.0 ports. While the radio was plugged into the USB 2.0 port, changing the setting to USB 3.0 resolved the connection issue.