Setting up for Development of Crazyflie Client

Procedure and analysis of challenges encountered

Michael Saybolt, Joe Fabbo

Michigan State University

5/4/2017

Outline

Introduction.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

Introduction.DevelopmentBackground

- 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)

Software Package Requirments

- Git
  - To download and maintain Crazyflie programs
- Python virtual environment
  - Dependency management
    - Anaconda virtual environment
    - virtualenv
- Docker
  - 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

Procedure for Basic Development

- (Optional) Install Docker/toolbelt
- Fork/clone crazyflie-clients-python from GitHub

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

- Set up Python virtual environment
- Install necessary dependencies
- Set udev permissions
- Run Crazyflie client, you can modify the source code

Setting up Python Virtual Environment

- Most of the documentation states to use 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

Setting up Python Virtual environment (cont.)

- Download and install latest Miniconda setup
  - It should add an alias to the 'conda' command in your .bashrc
  - This lets you run conda from any folder by typing 'conda'
- Create a conda virtual environment for the CF Client install and development
  - Name it something useful, like 'crazyfliedev'

```
conda create -n crazyfliedev
```

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

Installing Dependencies

- Hard work of solving proper dependencies can be summed up
- .. Into a conda virtual environment configuration file!
- Simply clone (or view/search) this github repository:

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

  - Locate File:
  - ECE813/environments/crazyflie.yml
  - Create a new conda virtual environment from file!

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

  - Activate it:

    ```
    source activate crazyflie
    ```

Installing Dependencies (cont.)

- Else, (say, in case of failure to run CF client with those dependencies included in crazyflie.yml)
  - Activate previously created 'naked' environment

    ```
    source activate crazyfliedev
    ```
  - Run setup file

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

    ```
    python setup.py
    ```
- Install any dependencies inside your conda virtual environment
- Deactivate the virtual environment when not needed

  ```
  source deactivate
  ```
- Any modification or use of the CFC will require the environment to be active

Setting USB Device (udev) permissions

- Setting udev permissions allows access to usb devices without root permissions
- It is bad practice to use root unless absolutely necessary
- Crazyflie development is not a good reason
- Add yourUsername to group plugdev if it is not already
  - Check to see what groups you are in

    ```
    groups yourUsername
    ```
  - Add yourself or create and add if needed

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

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 ./
```

To Do

- 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

Challenges: Sounds easy right?

- 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