FLYING THE CRAZYFLIE

This tutorial is meant to cover **the specific configuration of the Crazyflie 2.1 drone.** In doing so, the firmware is explored to make the best use of this drone in the lab.

TODO (Please ignore)

1. **First flight with a keyboard**
2. **Various python application libraries**
3. **Platform replication**

Contents

A research drone

The Crazyflie 2.1 is a **durable**, **open-hardware** nano **quadcopter** that targets hobbyists and researchers. Its small size (92 mm diagonal rotor-to-rotor) and weight (27 g) make it ideal for indoor swarming applications. The firmware is open source and the flexibility of the platform makes it ideal for research, education or other applications where **openness** and **full control** is important.

As a fist-sized, lightweight and **re-assemblable** piece of equipment, it offers functional hardware for drone autonomy in a field marred by **hardware constraints**. Its 6 to 7 minutes of flight time make it practical for testing **autonomous flight algorithms**, and it has become the drone of choice for research laboratories. It is particularly adept to autonomous control and coordination of **multi-robot systems**, since its small size allows for **dense formations with low air turbulence**. It also offers agility in research pertaining to **control optimisation** for **aggressive manoeuvres**.

Starting point for the Crazyflie 2.1

https://www.bitcraze.io/documentation/start/

External projects based on the Crazyflie

https://www.bitcraze.io/support/external-projects/

For **drone maintenance**, please refer to the drone tips tutorial.

Drone components

The architecture is as follows (for 2.0 and 2.1 versions):

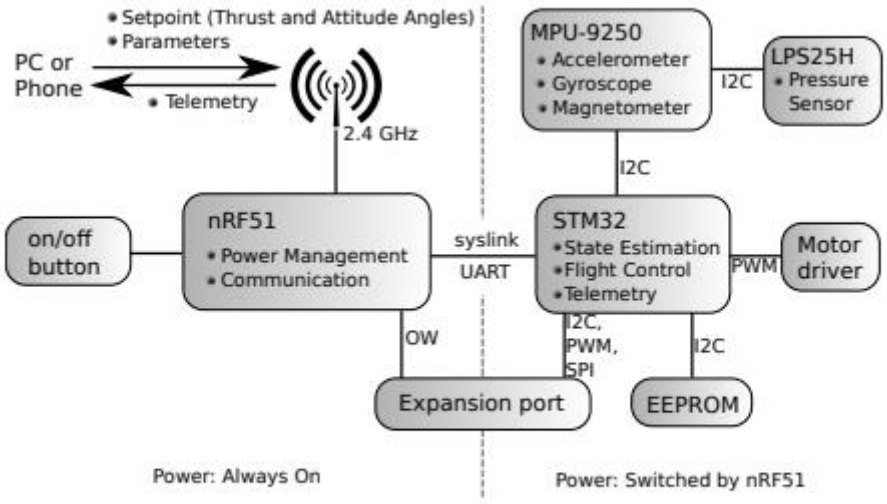


Fig. 2. Components and architecture of the Crazyflie 2.0 quadcopter. Based on images by Bitcraze AB.

- STM32F405: main microcontroller, used for state-estimation, control, and handling of extensions. (Cortex-M4, 168 MHz, 192 kB SRAM, 1 MB flash).
- nRF51822: radio and power management microcontroller. (Cortex-M0, 32 MHz, 16 kB SRAM, 128 kB flash).
- MPU-9250: 9-axis inertial measurement unit.
- LPS25H: pressure sensor.
- 8 kB EEPROM.
- uUSB: charging and wired communication.
- Expansion port (I2C, UART, SPI, GPIO).
- Debug port for STM32. An optional debug-kit can be used to convert to a standard JTAG-connector and to debug the nRF51 as well.

The Robotic Stack

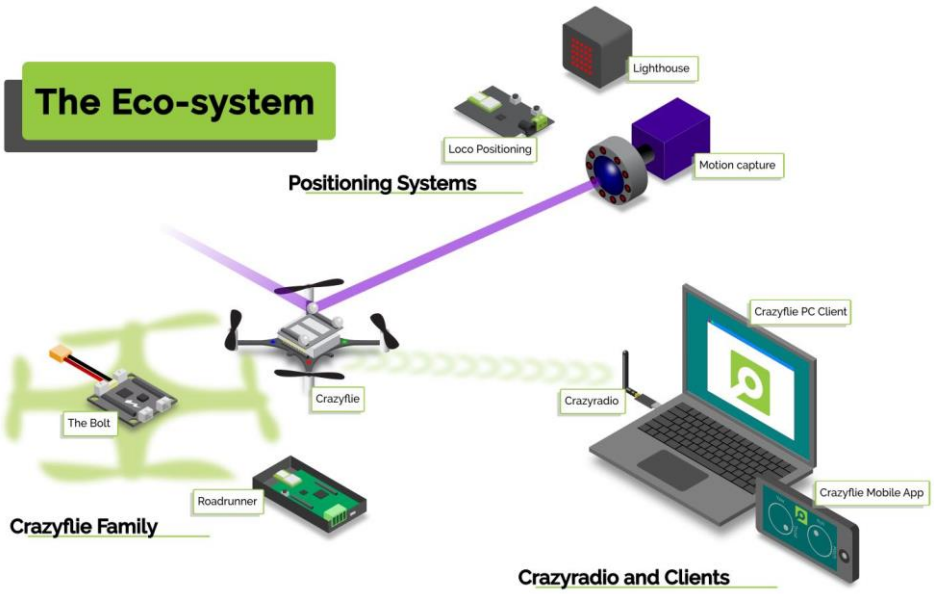
- SEE THEIR WEBSITE HERE FOR MORE INFORMATION

In terms of robotic functionality, the Crazyflie packages the **full robotic stack**. After all, the firmware, built on the **FreeRTOS operating system**, is opensource and modifiable. FreeRTOS handles the scheduling of processes and control the flight calculations. This robotic stack includes its own **state estimator**, **control architecture** and **trajectory follower**, which work out of the box. These algorithms are detailed on the Bitcraze website, and they can be customised at compile time. Note however the **lack of localization framework**, which calls for added modules, or an external localization system. There are two in-house position systems in the Crazyflie ecosystem: the Ultra Wide Band based **Loco Positioning System** and the **Lighthouse positioning system**, based on HTC Vive. It is also possible to integrate with external positioning systems, for instance Motion Capture systems.

Understanding an Autonomous Drone - bird's eye view
https://www.youtube.com/watch?v=1FG0wuAW1hM

Understanding an
Autonomous Drone (bird's...

Connecting and compiling



The compilation requires a wired connected with a computer, with the quadrotor in the bootloader mode. A Crazyflie can communicate with a phone or PC using BLE, when **flying manually with the Crazyflie companion app** or **custom designs**. Additionally, a custom USB dongle called Crazyradio PA, or Crazyradio for short, allows **lower latency communication**. The radio communication use a proprietary protocol, the **Crazy RealTime Protocol** (CRTP).

Github repositories

Python, C-based firmware and bootloaders

https://www.bitcraze.io/documentation/repository/

#Flight scripts

Procedure to be found here:

Trajectory Generation (/3)

less than 1 minute read

Developing trajectories for robots to follow.

A proprietary Python library can get the Crazyflie up and going and is also used to control swarms of Crazyflies. The python client runs on a PC and communicates via CRTP, where it can monitor robot battery voltage level, flight state and aid with bootloading especially as large swarms become more difficult to manage.

- FLYING THE CRAZYFLIE PROJECT

The crazyflie_ros framework helps wrap CRTP within a ROS framework, which is useful for scenarios of **hovering** and **waypoint following** from a **single robot** to the more complex multi-UAV case.

Theoretical underpinnings

crazyflie_ros: Flying Multiple UAVs using ROS

http://act.usc.edu/publications/Hoenig_Springer_ROS2017.pdf

The external CrazySwarm project is a **lower latency framework** that has gradually been extended to **incorporate external localization systems** such as VICON or Optitrack.

Link to tutorial:

Crazyswarm framework for synchronised flight

less than 1 minute read

Getting up and started with drone swarms.

The **modularity** of this platform gives the Crazyflie extra capabilities in sensing, positioning or visualization. See website for more on the LED ring, an AI chip coupled to a small camera, an optic flow deck, etc.

Implementation at the DVIC (as of August 2020)

Getting up to speed with the Crazyflie isn't too difficult, as its functionalities are neatly packaged. A first flight of the platform makes use of a keyboard control algorithm, which we detail in our Crazyflie flight project. We use this to check that the drone is in a working state. Its python API is useful for prototyping applications.

A video gives a quick overview of why as a lab we chose this drone over any other.

- FLYING THE CRAZYFLIE PROJECT