Introduction to the Crazyflie

Lecture at Aerial Robotics Course (EPFL)

Kimberly McGuire

3rd of April 2023

Introduction to Bitcraze AB

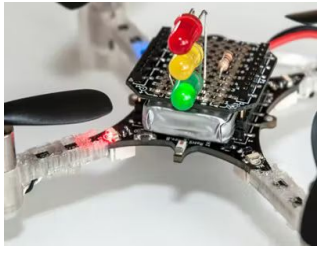
- Who are we?
  - Crazyflie
  - Hardware Development
- Where are we?
  - Malmö, Sweden
- All the team members?
  - Tobias
  - Marcus
  - Kristoffer
  - Arnaud
  - Barbara
  - Kimberly

History of Bitcraze

- Hobby project
- Crazyflie 1.0
- Company in 2011
- Crazyflie 2.X

Who uses the Crazyflie?

- Hobbyists
- Researchers
- Shows designers
- Educators
  - And their students :)



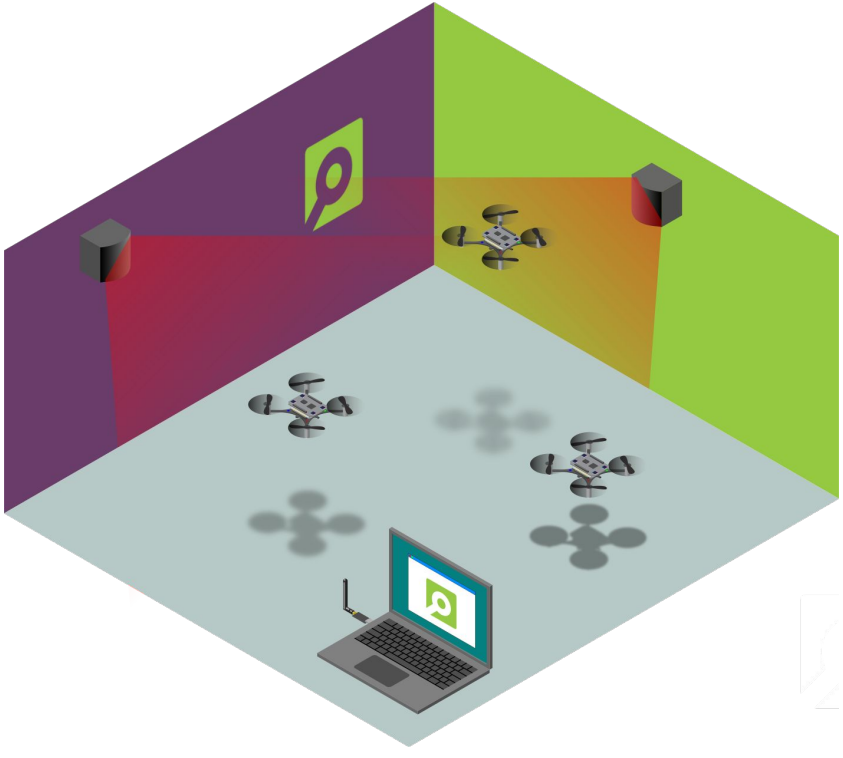
Research 2020-2022

https://youtu.be/iTe6-lLp5iM

Crazyflie show with 20+ CFs

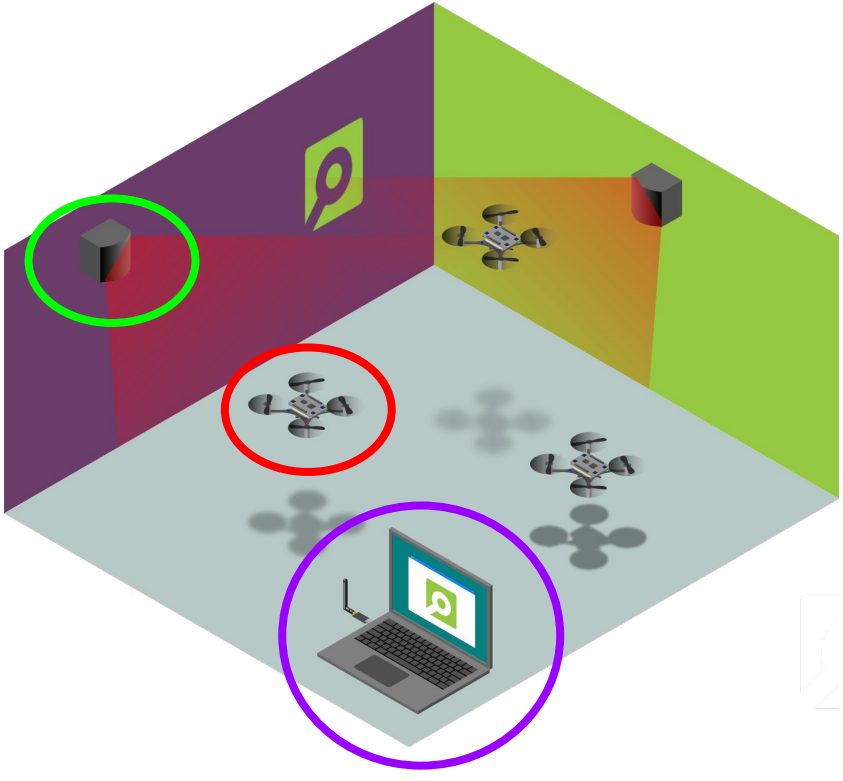
https://youtu.be/w8jAHYlcj7k

The Eco-system

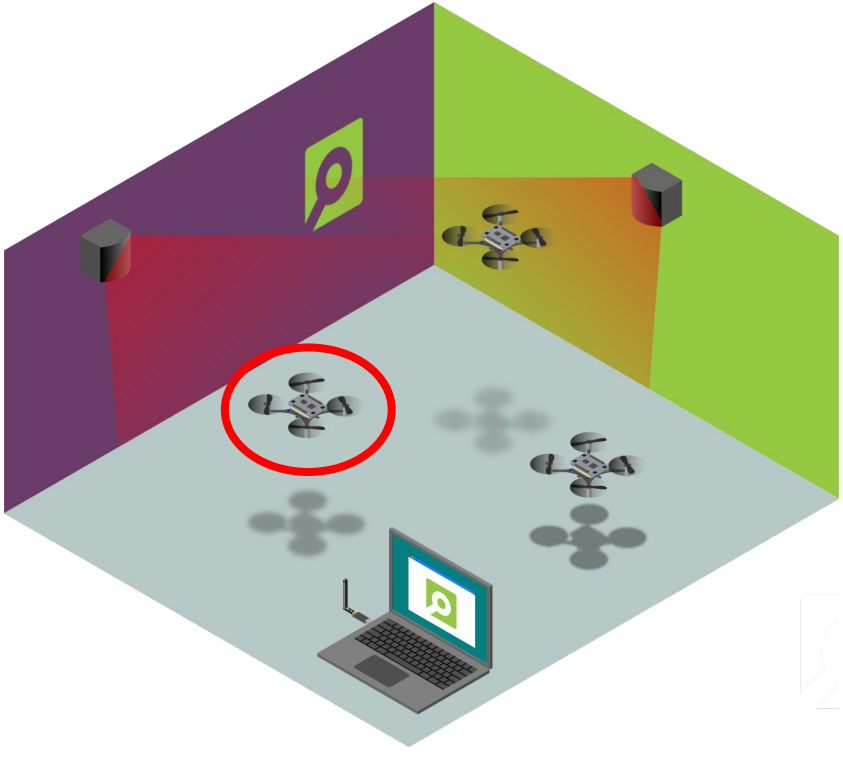


The Eco-system

- Quadcopter
- Positioning
- Communication



The Quadcopter

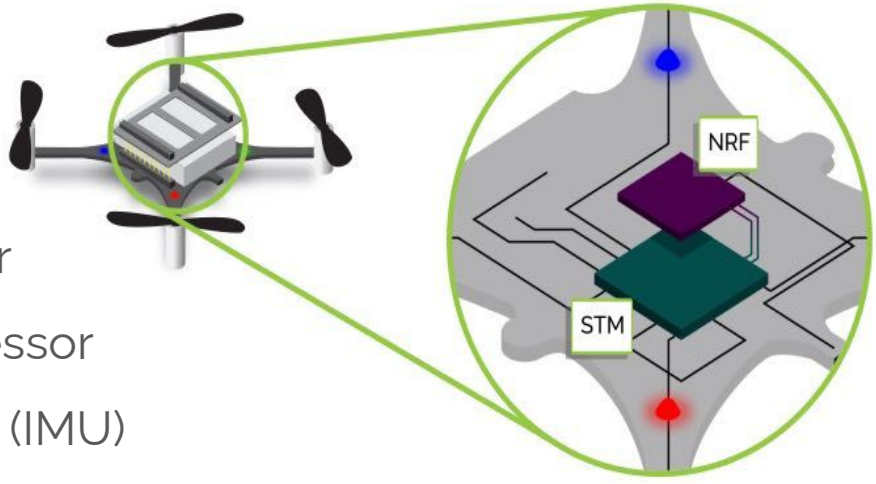


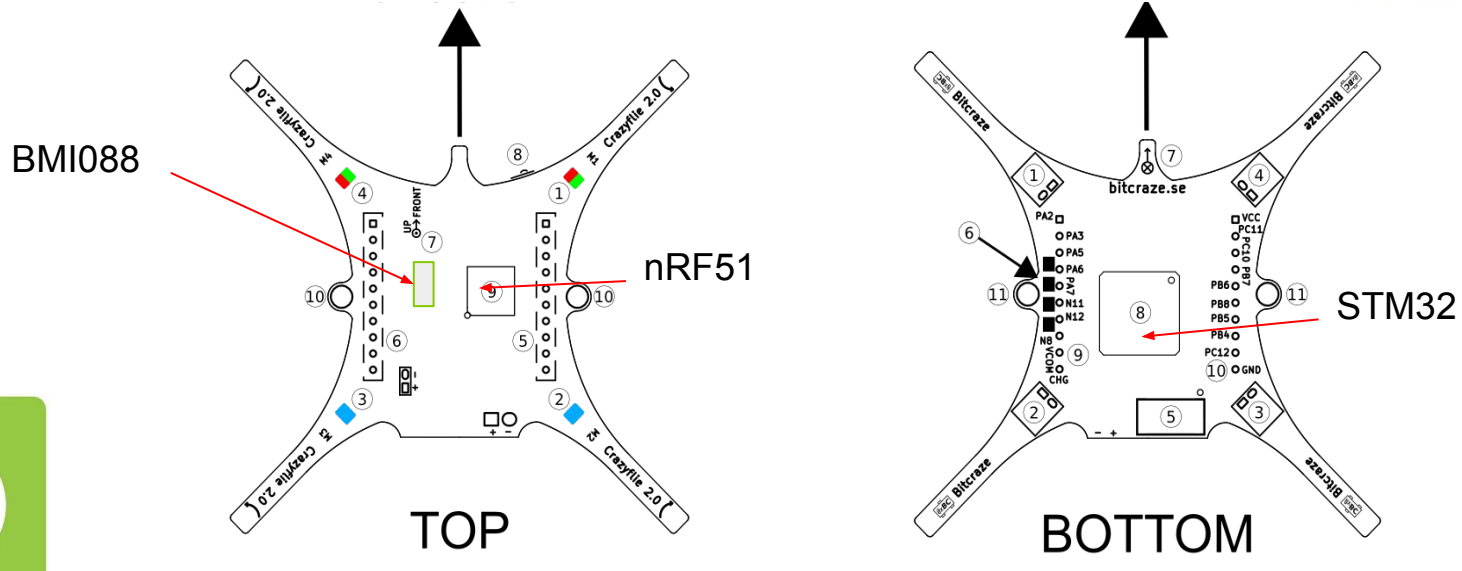
Crazyflie Hardware

- Quadrotor
- 4 DC **coreless** motors
  - Less strong than brushless, more efficient though, and safe :D
- Control board
- 24 grams
- 1 cell lipo battery (7 min flight time)
  - Do not deplete the battery! Land when you see **the red LED**
  - Firmware is open-source : crazyflie-firmware

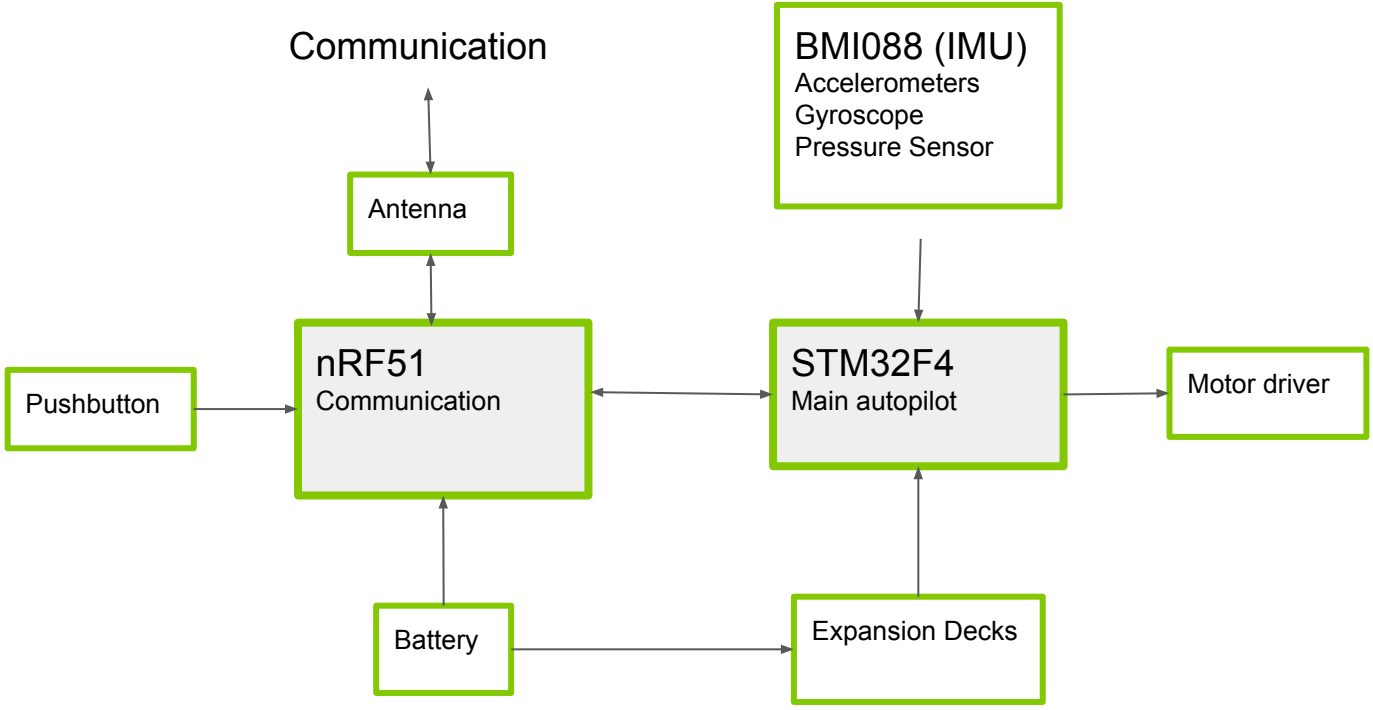
Back to the hardware

- STM32F4: Autopilot Microprocessor
- nRF51: Communication Microprocessor
- BMI088: Inertial Measurement Unit (IMU)



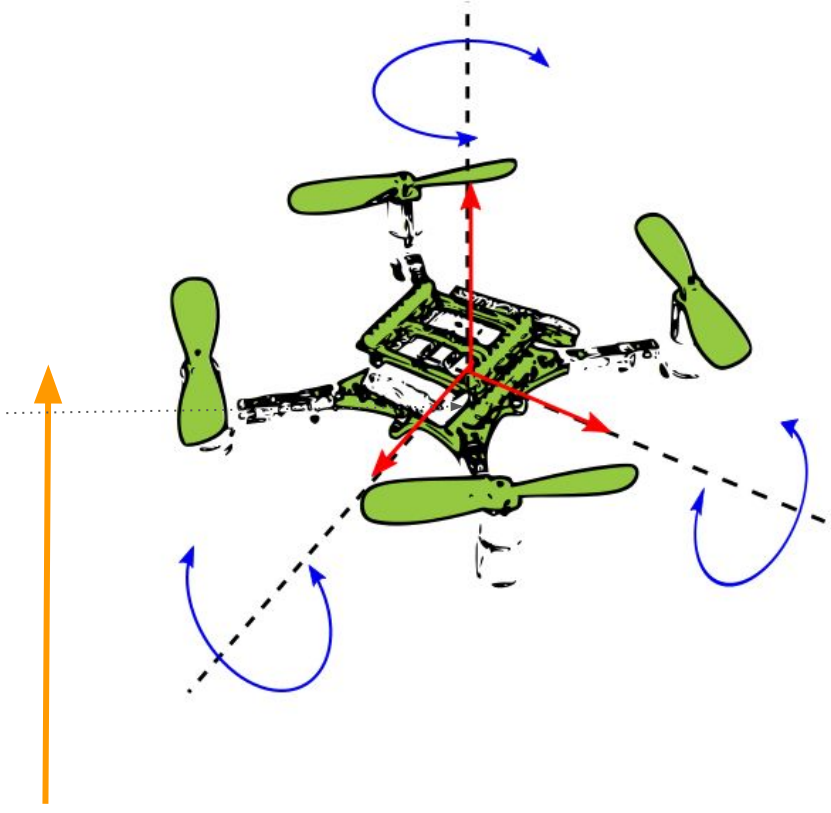


Hardware component connections

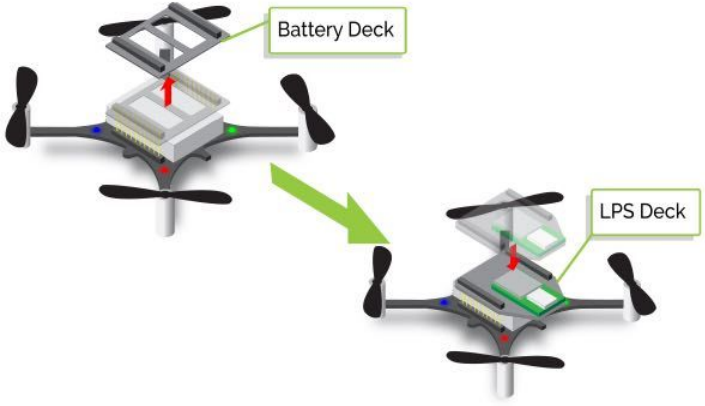


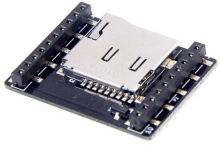
Inertial Measurement Unit (IMU)

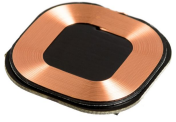
- Accelerometers
- Gyroscope
- *Pressure Sensor*

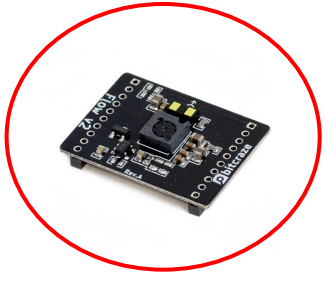


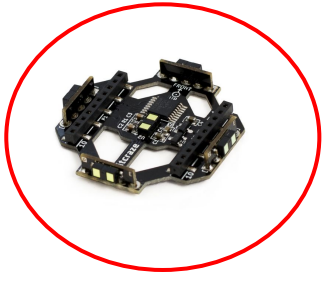
Expansion Decks



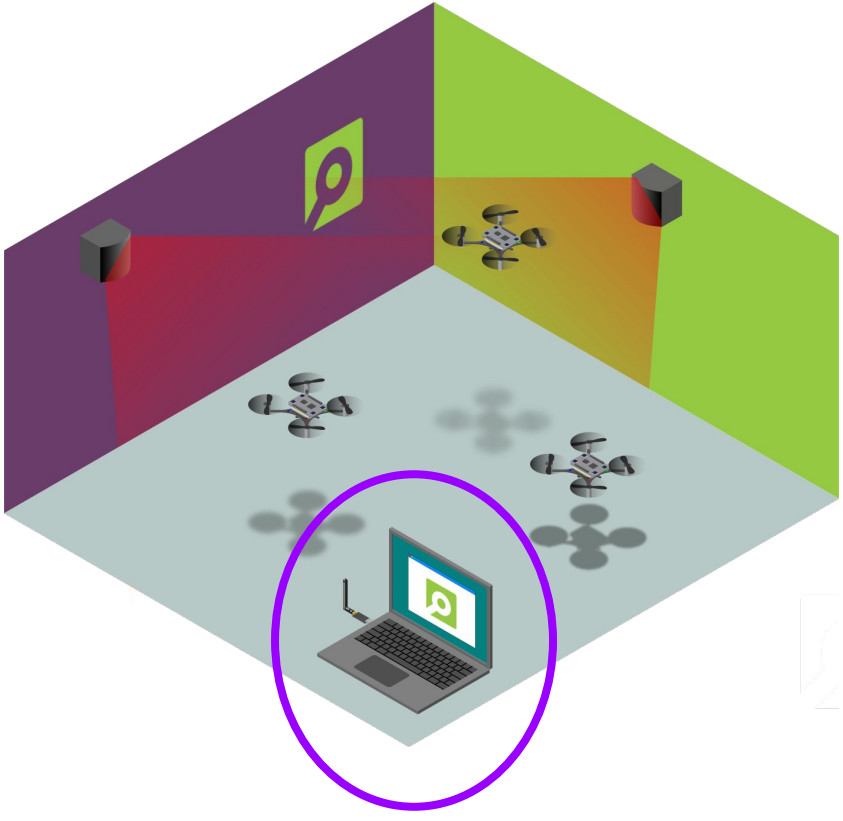




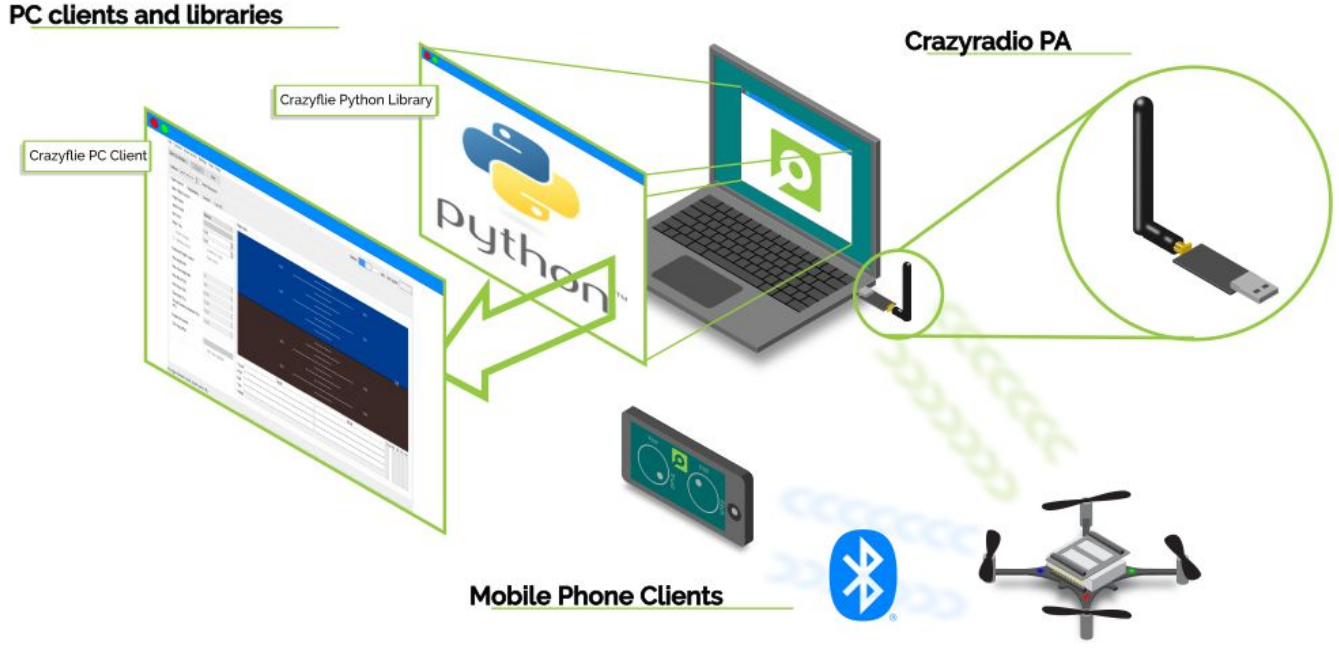




Communication



Client Software



All open-source software

Communication

- Crazyradio PA
  - Crazyradio Real-Time Protocol (CRTP)
- Unique URI

0xE7E7E7E701

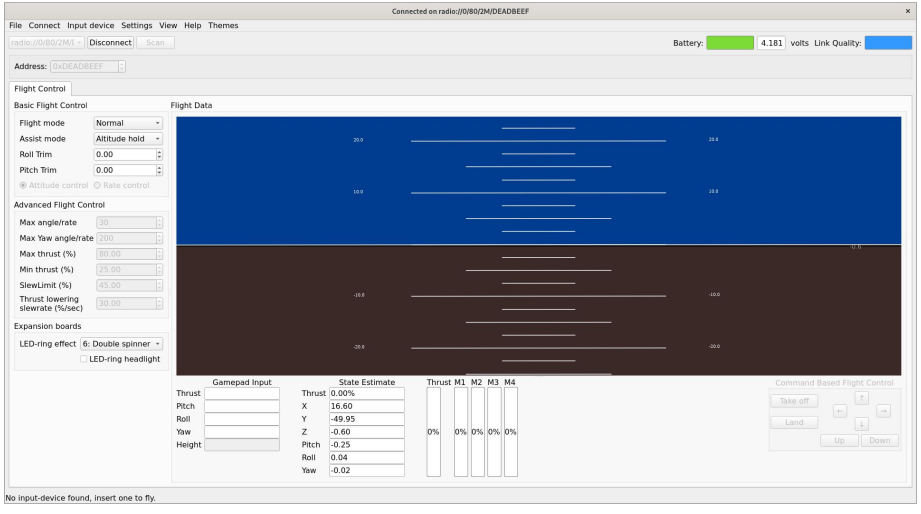
- Multiple Crazyflies

0xE7E7E7E702

0xE7E7E7E703

Crazyflie Python Client (CFclient)

- Python 3.7>
- `pip3 install cfclient`
- USB devices access differ for win/linux/mac



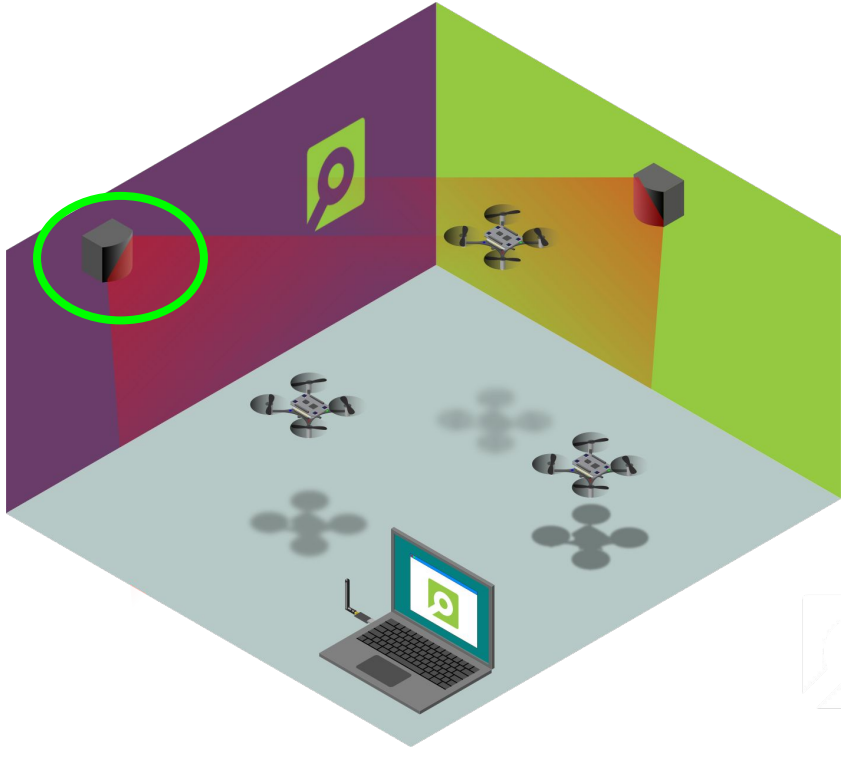
https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/installation/install/

HANDS-ON

Connect to the Crazyflie

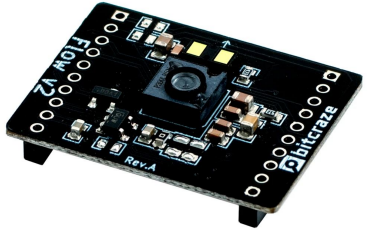
Show the CF client flight tab

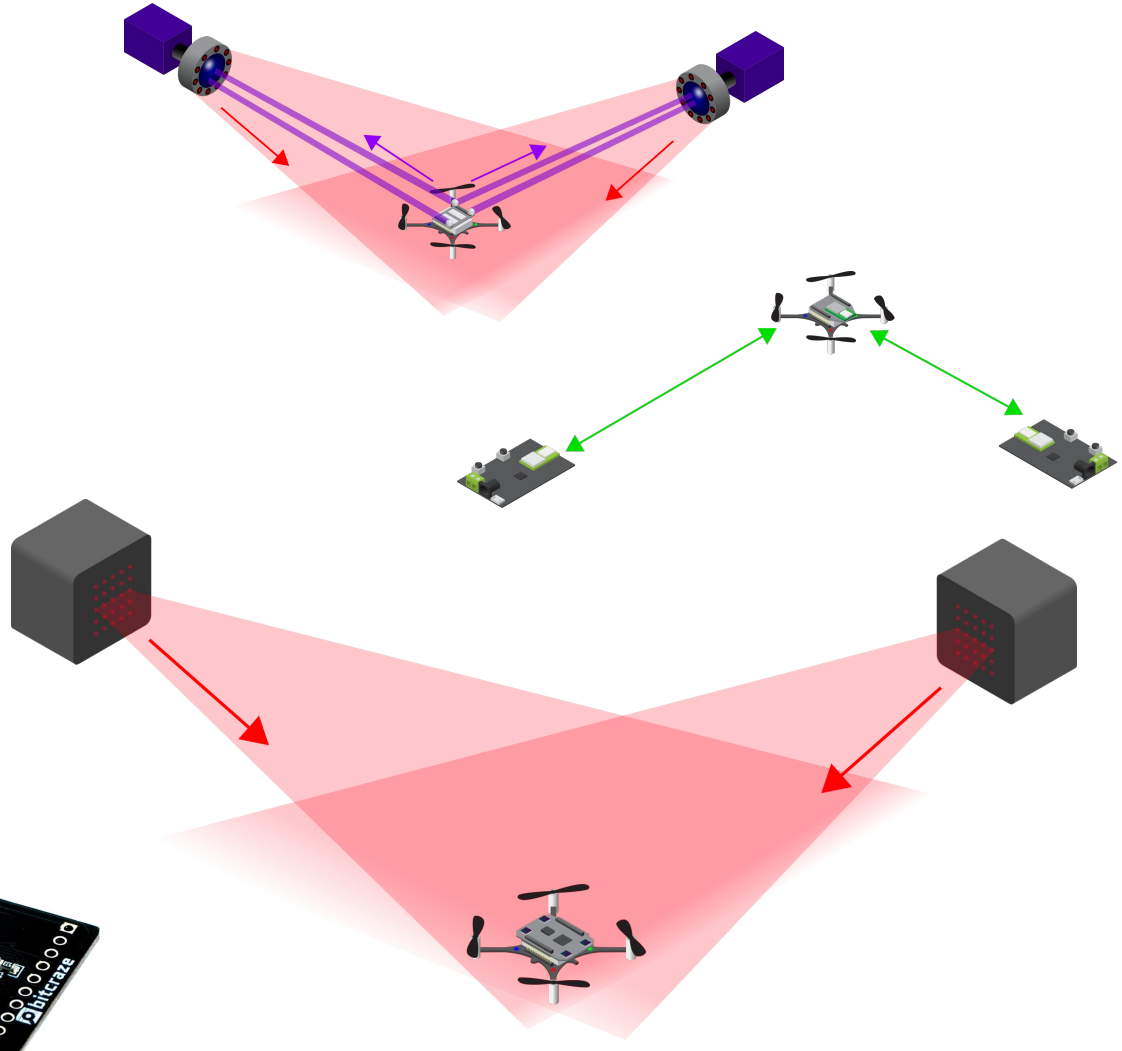
Positioning



Positioning types

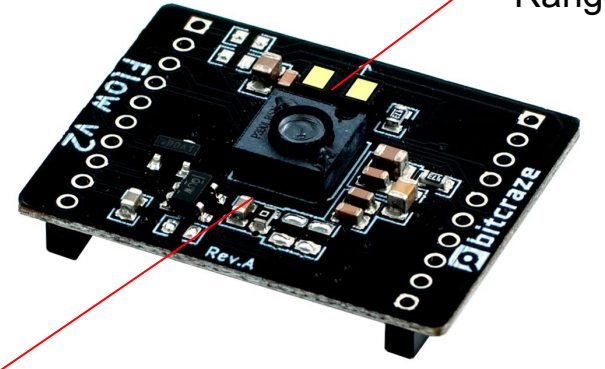
- Motion Capture Systems
  - Markers
- Loco positioning systems
  - Ultra wide band
  - Like in the show video
- Lighthouse system
  - HTC vive VR system
- *Relative positioning*
  - *Flow-deck*



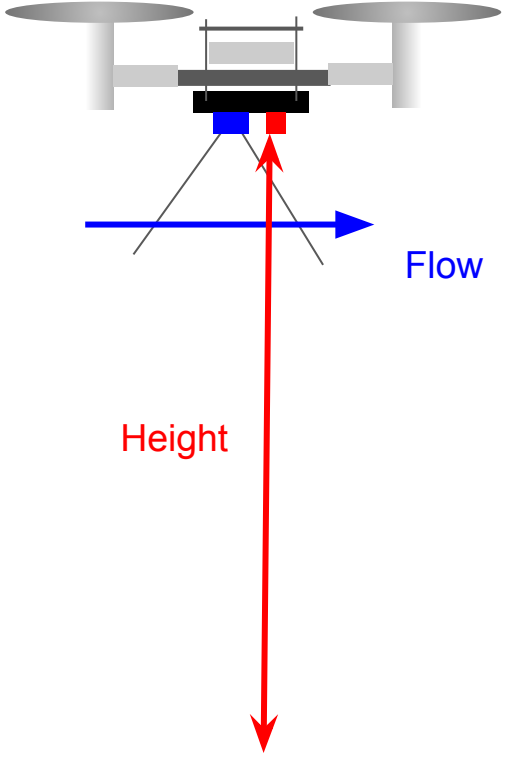


Flowdeck hardware

VL53L1x
Range sensor



PMW3901
Optic Flow Sensor

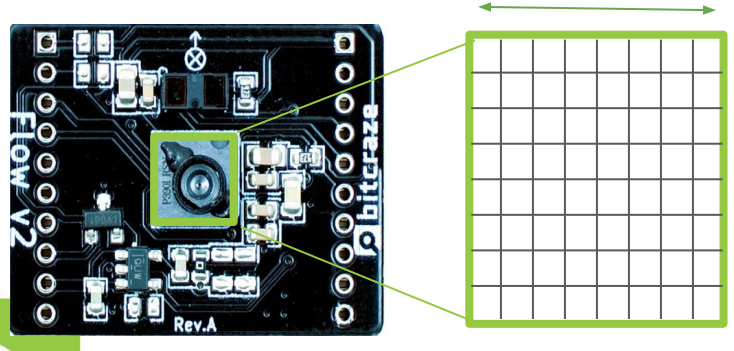


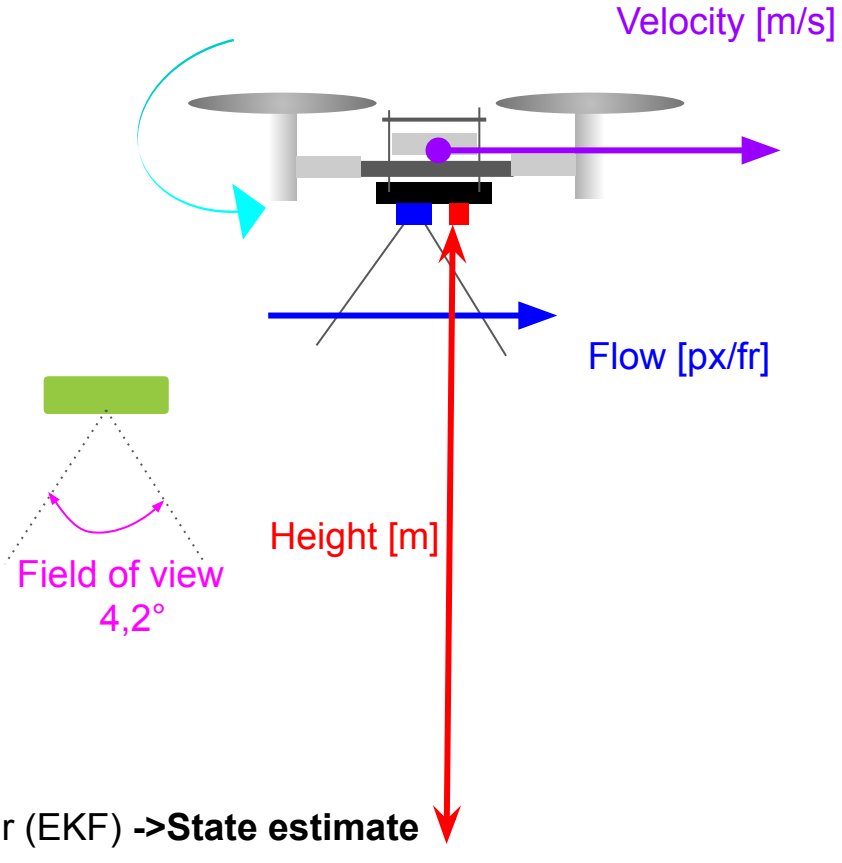
Relative vs global position!

Velocity Flowdeck

$$velocity = \frac{height * FOV * flow}{\Delta t * pixel\_width} + \dots$$

Sample time





Measurement model -> Extended Kalman Filter (EKF) **->State estimate**

https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/

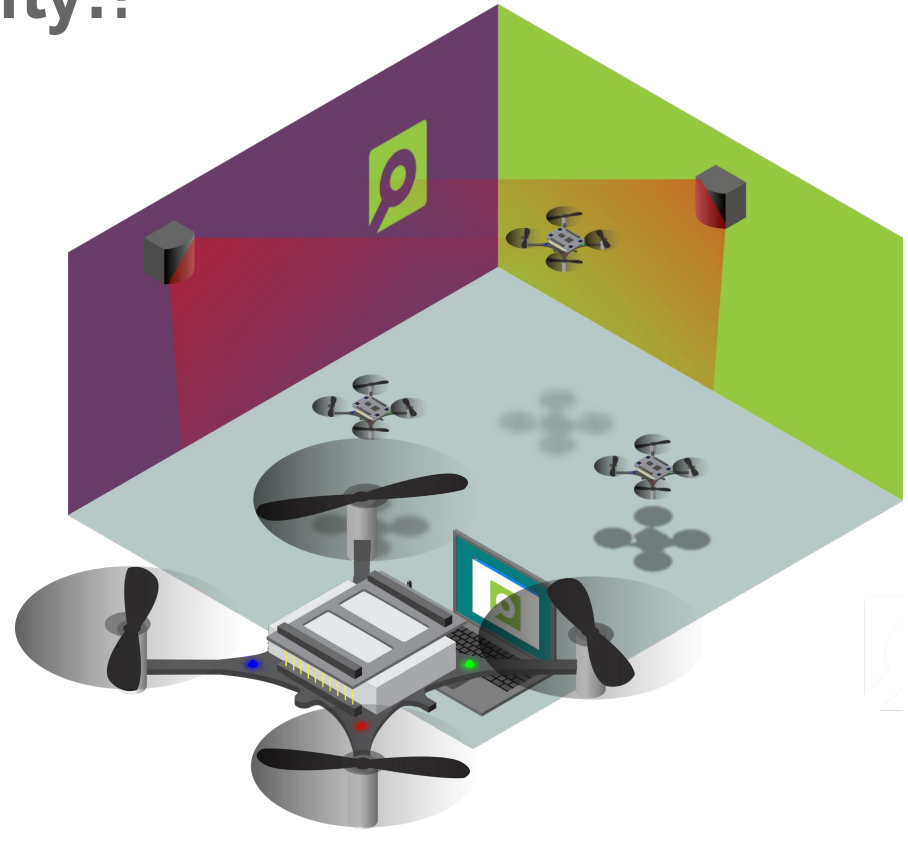
HANDS-ON

Introduction to console-tab

CFclient logging with flowdeck measurements

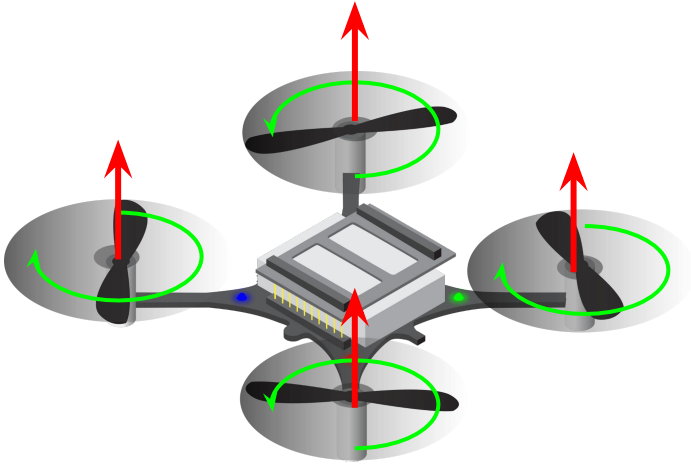
https://github.com/bitcraze/crazyflie-clients-python

Ready to fly!?



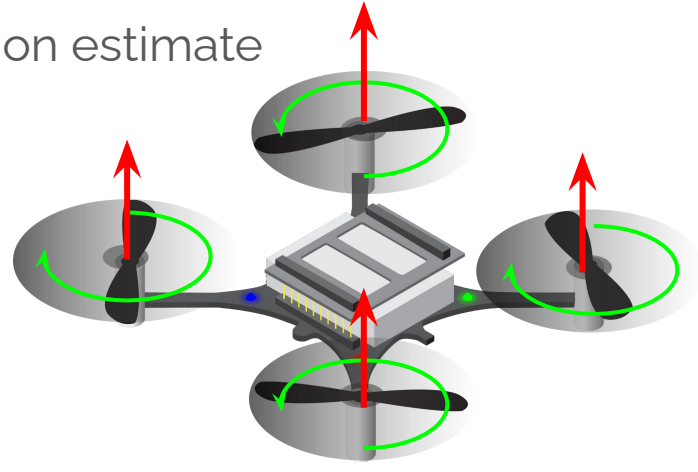
Quadcopter Dynamics

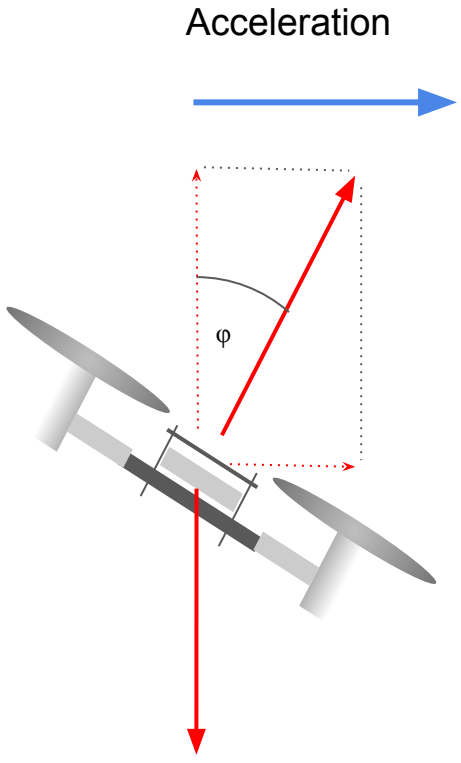
- Rotating motors
- Resulting Force



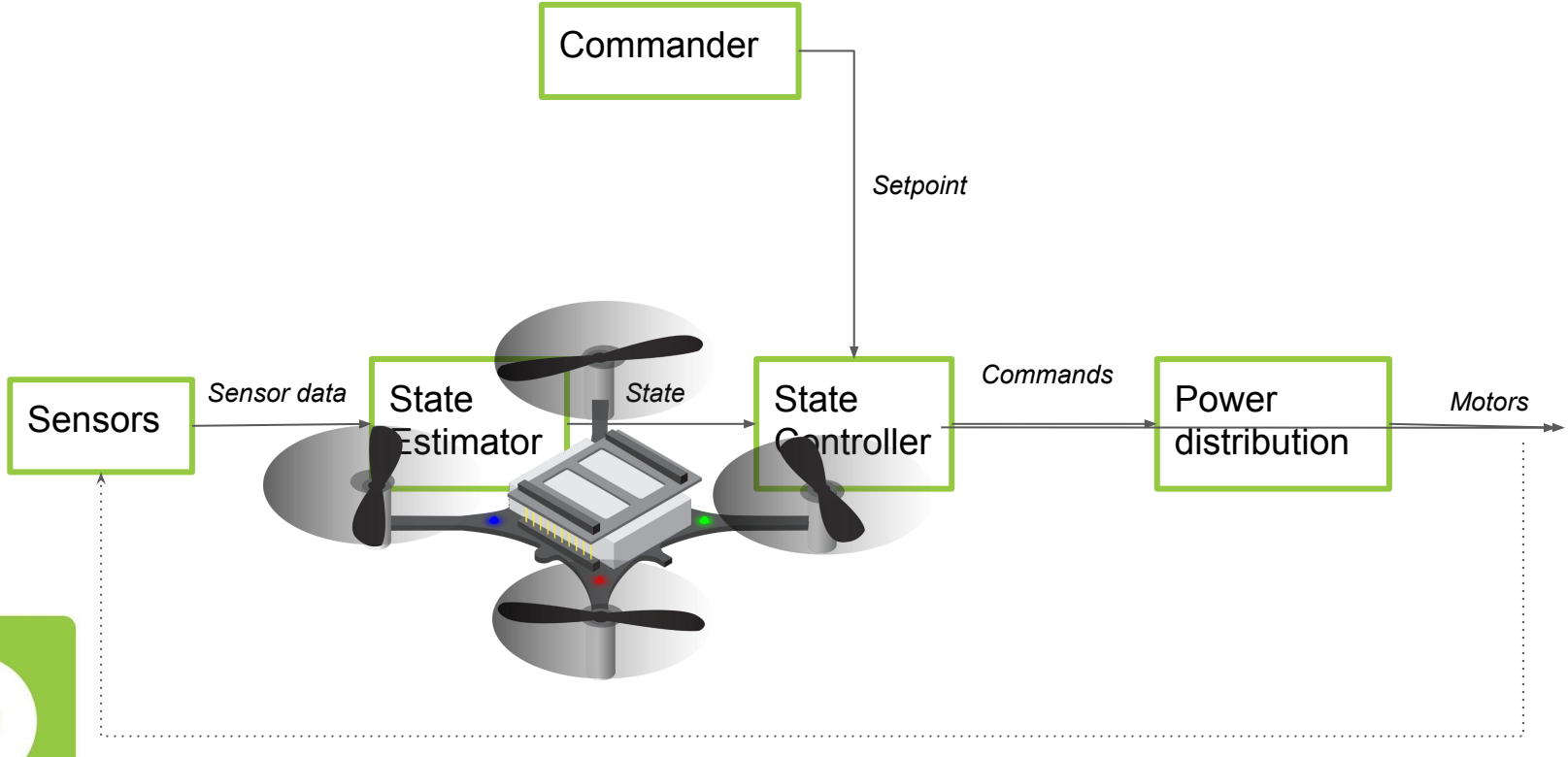
Quadcopter Dynamics

- Moments
- Linear acceleration
- Unstable = drift
- Need a position estimate





Flow from sensors to motors

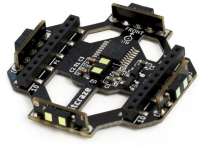


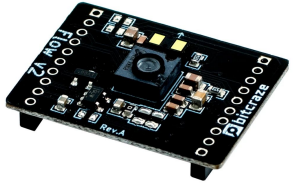
https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/

Hands-on

- Show the crazyflie flying with flight command
- CFclient show:
  - Position estimation
  - Control commands

Autonomy?



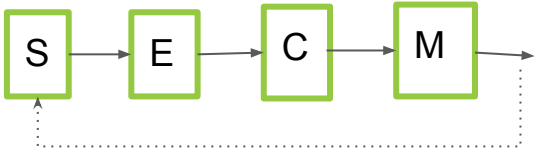


Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment (Science Robotics, 2019) K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen, https://youtu.be/jU4wsxwM1No

Break!

What is a robot?

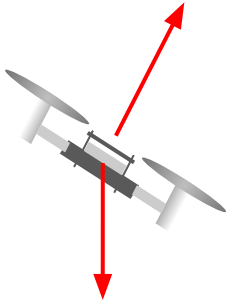
- Sensors
- Estimate
- Controllers
- Motors
- **Behavior**
- Is a quadcopter a robot?

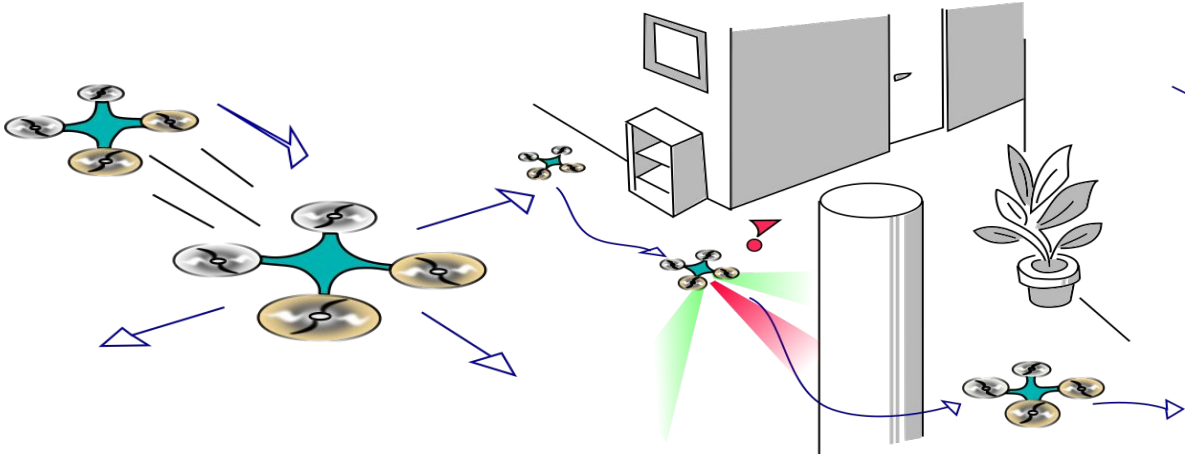


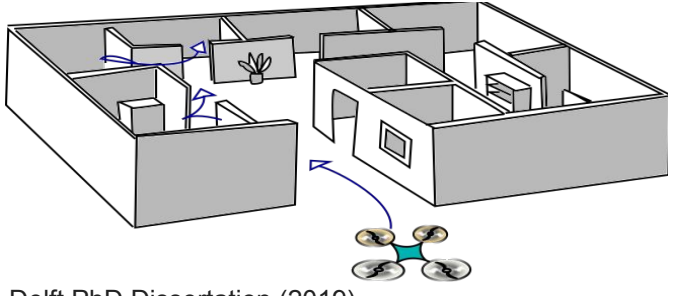
Nature

Autonomy levels

- Car autonomy levels?

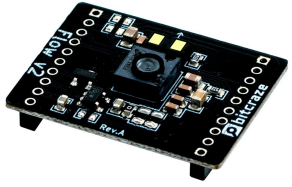




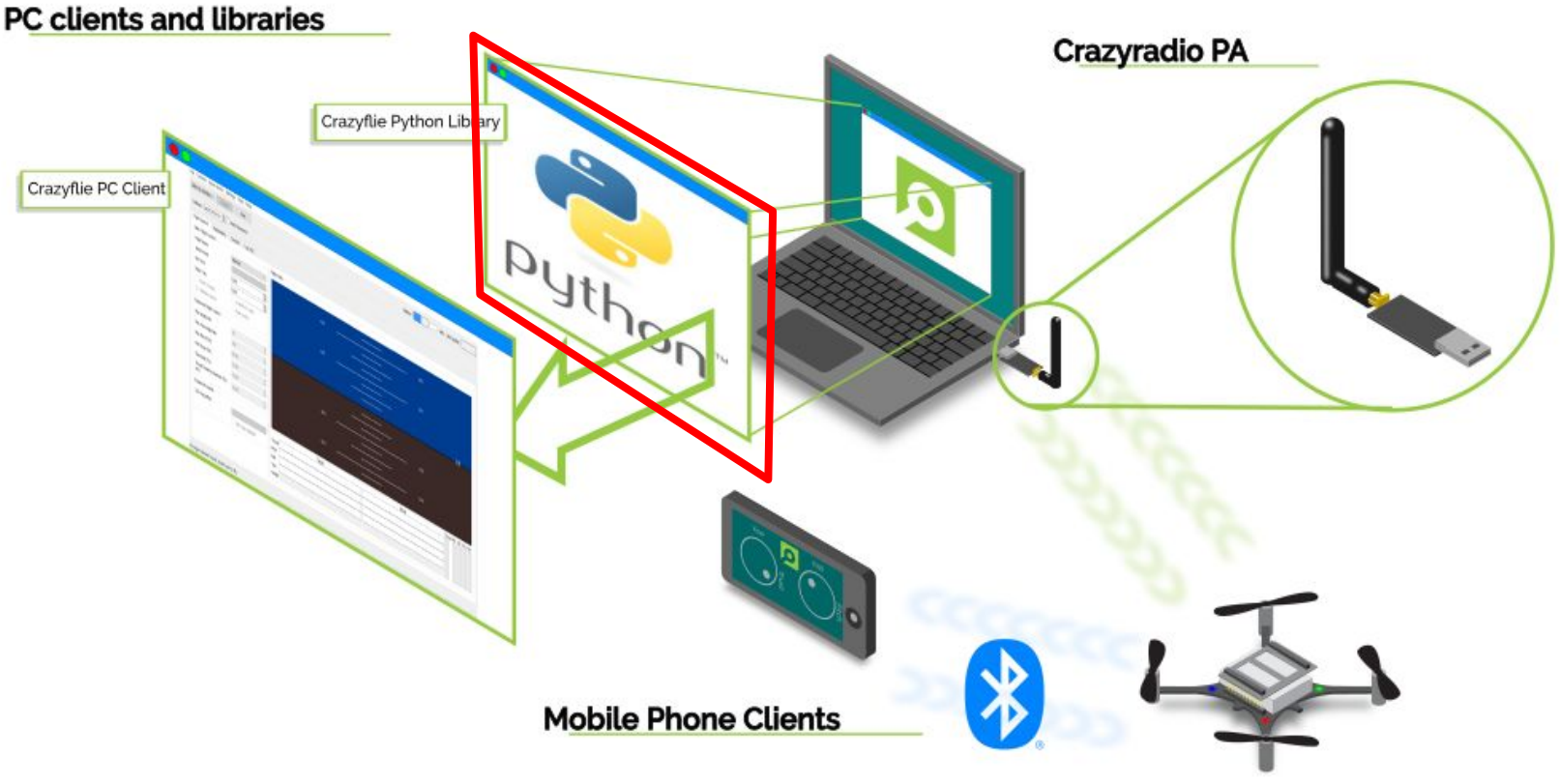


McGuire, Kimberly. "Indoor swarm exploration with Pocket Drones." TU Delft PhD Dissertation (2019).

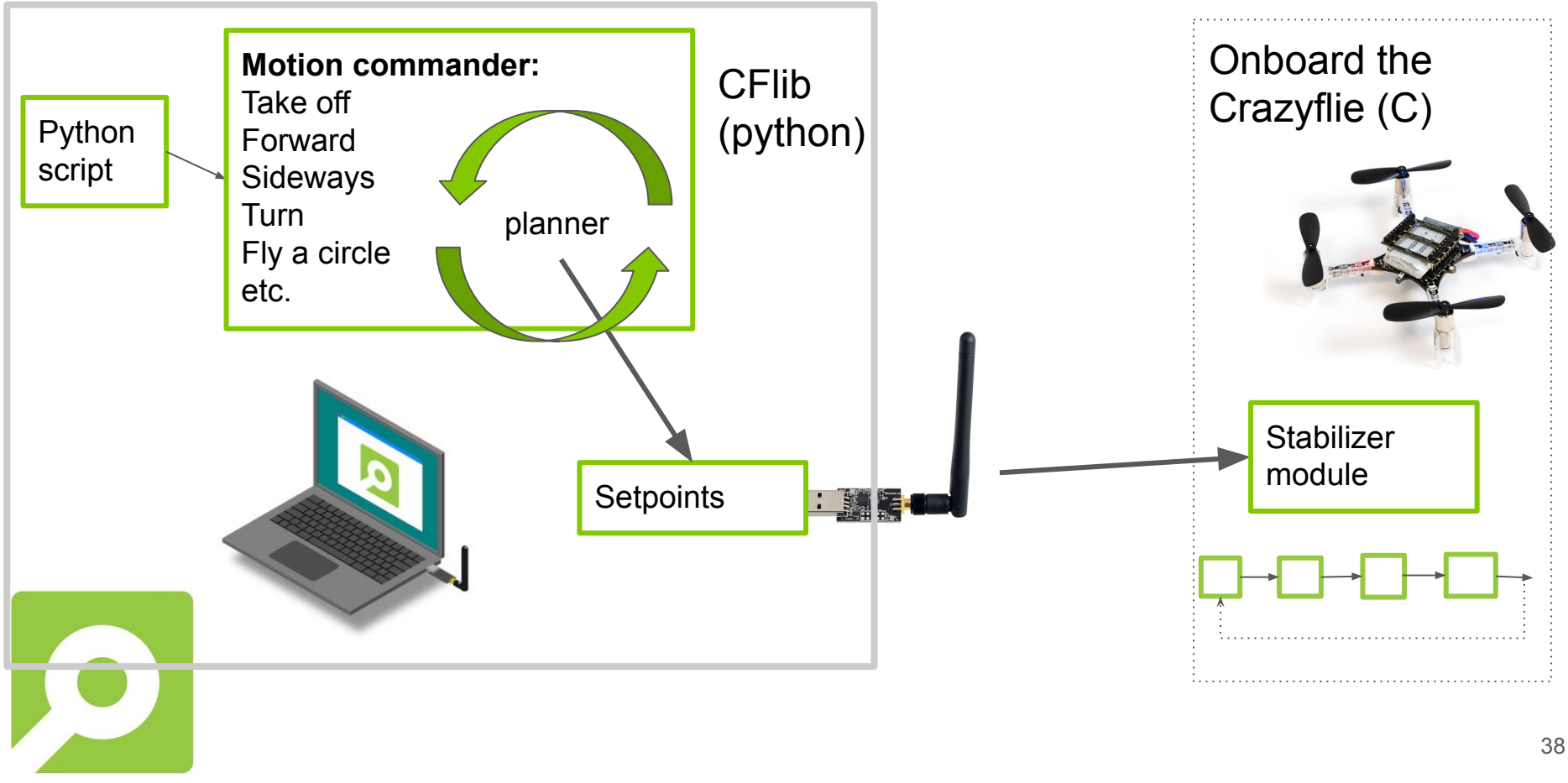
Translation



Crazyflie python library (CFlib)



Translation with CFlib



Motion Commander Class

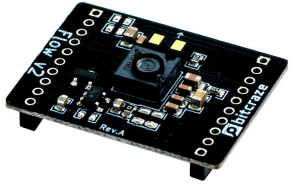
- Class MotionCommander(crazyflie, default_height = 0.3)
  - Translation: front(). back() left() right() down() up()
  - Turning: turn_left(), turn_right(), circle_left(), circle_right()
  - Start functions: start_*()
    - * = any of above
  - start_linear_motion(vx, vy, vz, rate_yaw)

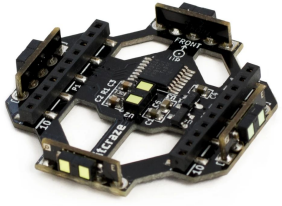
https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/positioning/motion_commander/

Video motion commander

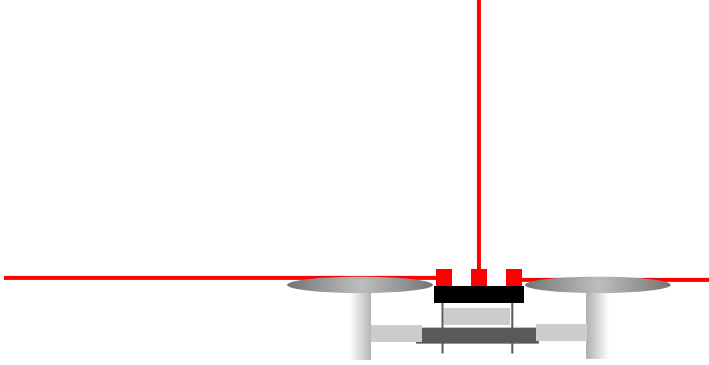
https://youtu.be/qKGjWWvjRt0

Avoidance

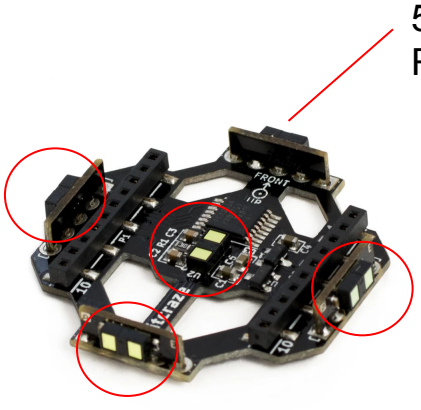




Expansion deck: Multiranger



- Getting values ranges of Multiranger



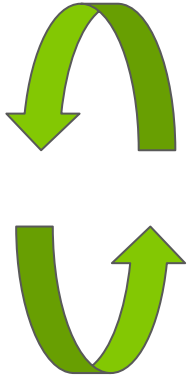
- Multiranger(crazyflie, rate_ms, zranger)
  - Down/back/front/left/right/up

- Up to 4 meters
  - Depended on environment
  - Difficult with direct sunlight

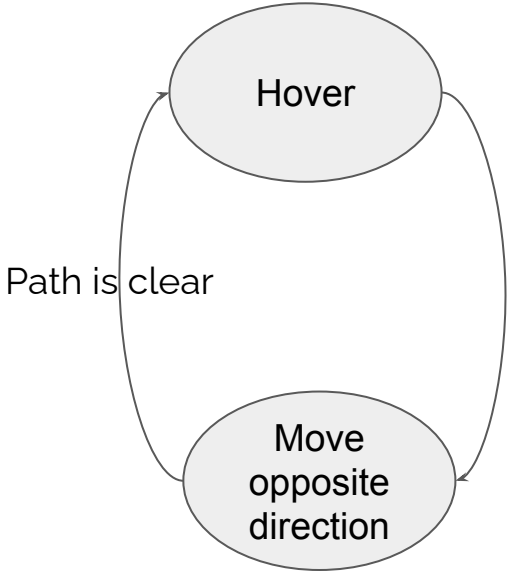
Multiranger_push.py (on the crazyflie-lib-python repository)

```python
if __name__ == '__main__':
    # Initialize the low-level drivers
    cflib.crtp.init_drivers()

    cf = Crazyflie(rw_cache='./cache')
    with SyncCrazyflie(URI, cf=cf) as scf:
        with MotionCommander(scf) as motion_commander:
            with Multiranger(scf) as multiranger:
                keep_flying = True

                while keep_flying:
                    VELOCITY = 0.5
                    velocity_x = 0.0
                    velocity_y = 0.0

                    if is_close(multiranger.front):
                        velocity_x -= VELOCITY
                    if is_close(multiranger.back):
                        velocity_x += VELOCITY

                    if is_close(multiranger.left):
                        velocity_y -= VELOCITY
                    if is_close(multiranger.right):
                        velocity_y += VELOCITY

                    if is_close(multiranger.up):
                        keep_flying = False

                    motion_commander.start_linear_motion(
                        velocity_x, velocity_y, 0)

                    time.sleep(0.1)

            print('Demo terminated!')
```

```python
with MotionCommander(scf) as motion_commander:
    with Multiranger(scf) as multiranger:
```



State machine




```python
motion_commander.start_linear_motion(
    velocity_x, velocity_y, 0)
```

HANDS-ON

- Look through the code
- Push demo

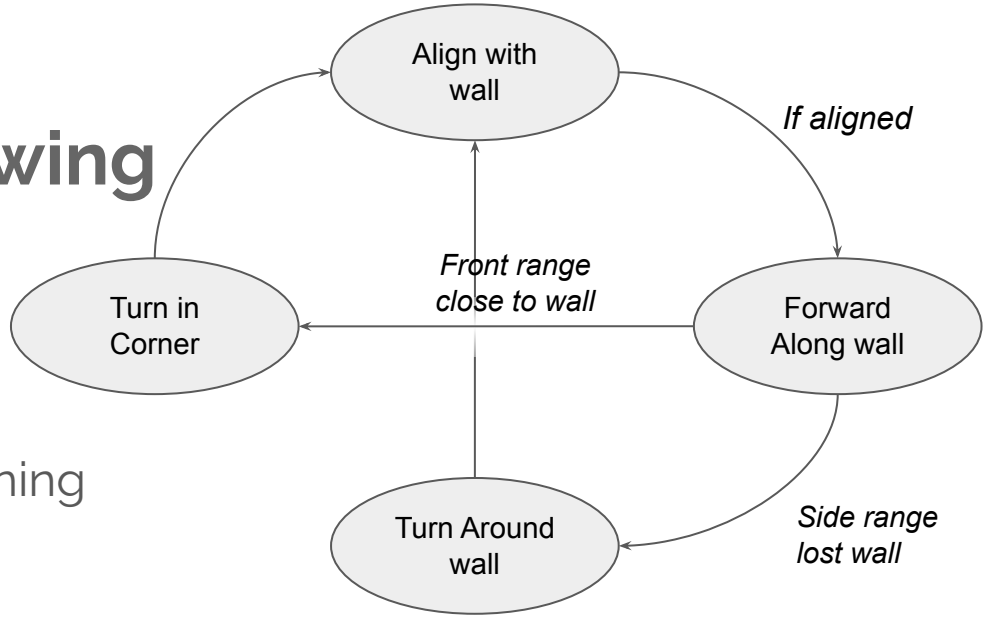
https://youtu.be/tQ9ygfUFOz8

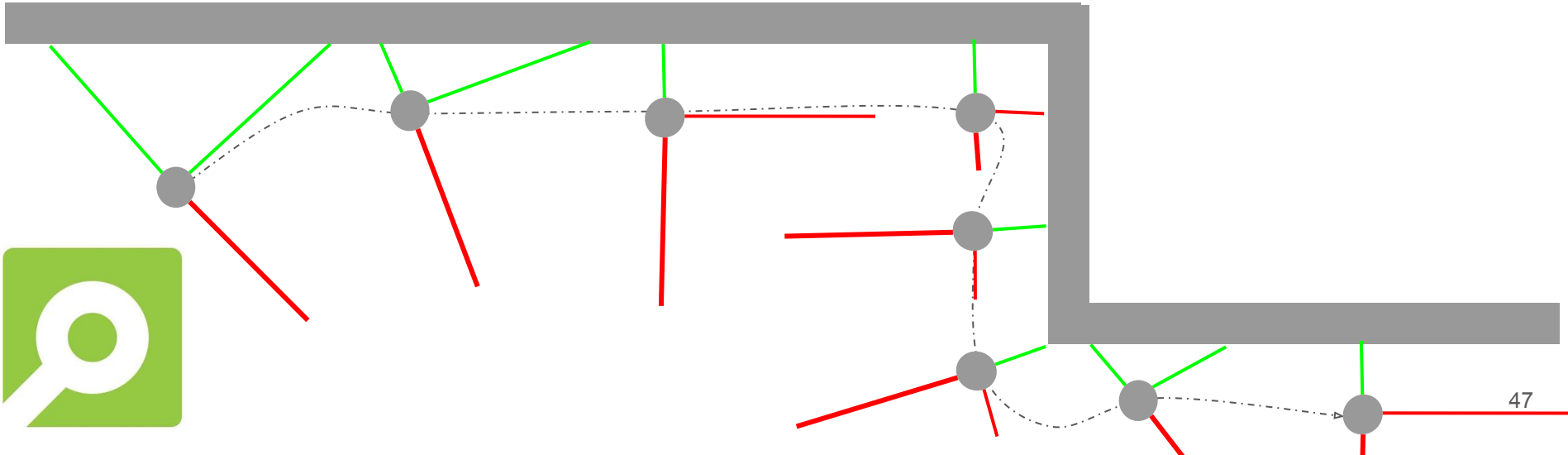
More complex behavior?

- Crazyflie is small = safe
- BUT, still annoying if it constantly crashes
- Might break props, motor guards, or worse!

Case study: Wall following

- State machine
- Need straight walls
- Wall following, turning and aligning
- Simulation help

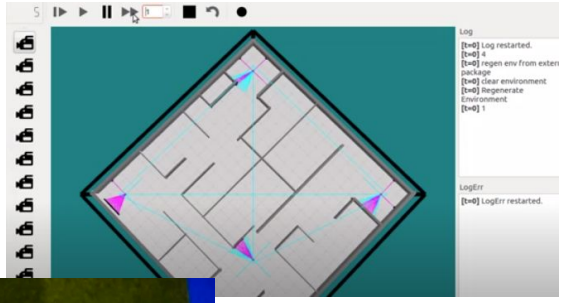


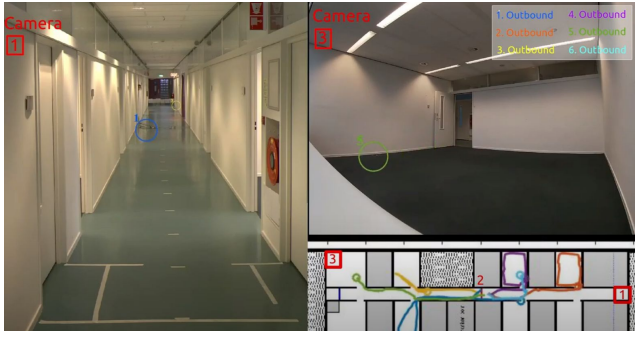


Process of developing

- PhD work: SGBA
- Steps:
  - 1- Python + ArGos**
  - 2- Python + Gazebo
  - 3- Python CFlib
  - 4- C + Gazebo
  - 5- C + On the drone

Simulation



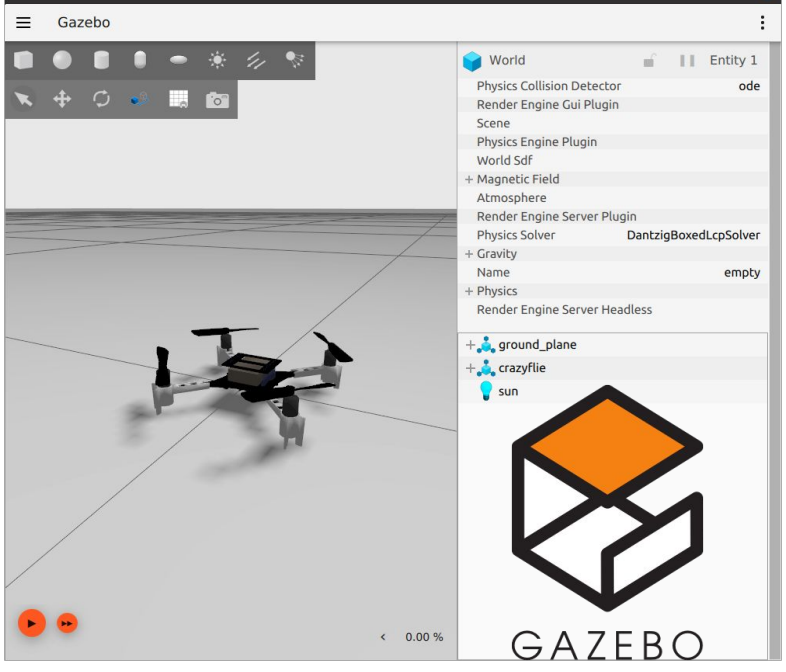


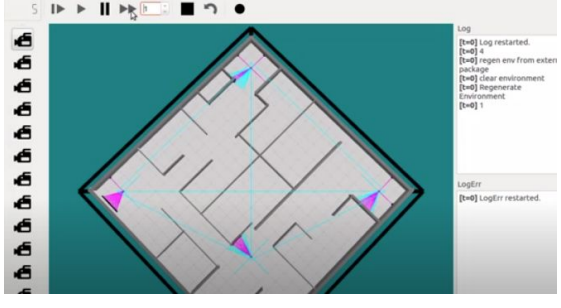
* Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment (Science Robotics) K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen,

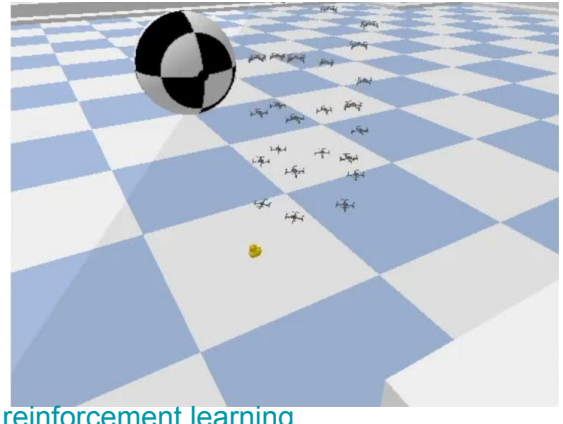
** McGuire, Kimberly N., G. C. H. E. de Croon, and Karl Tuyls. "A comparative study of bug algorithms for robot navigation." *Robotics and Autonomous Systems* 121 (2019): 103261.

Simulation

- General robotics sim
  - Gazebo
  - Webots
- Realistic Vision
  - Nvidia Isaac
  - AIRsim
- Swarms
  - ArGos
  - Pybullet gym*



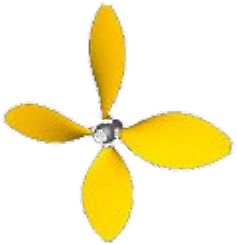


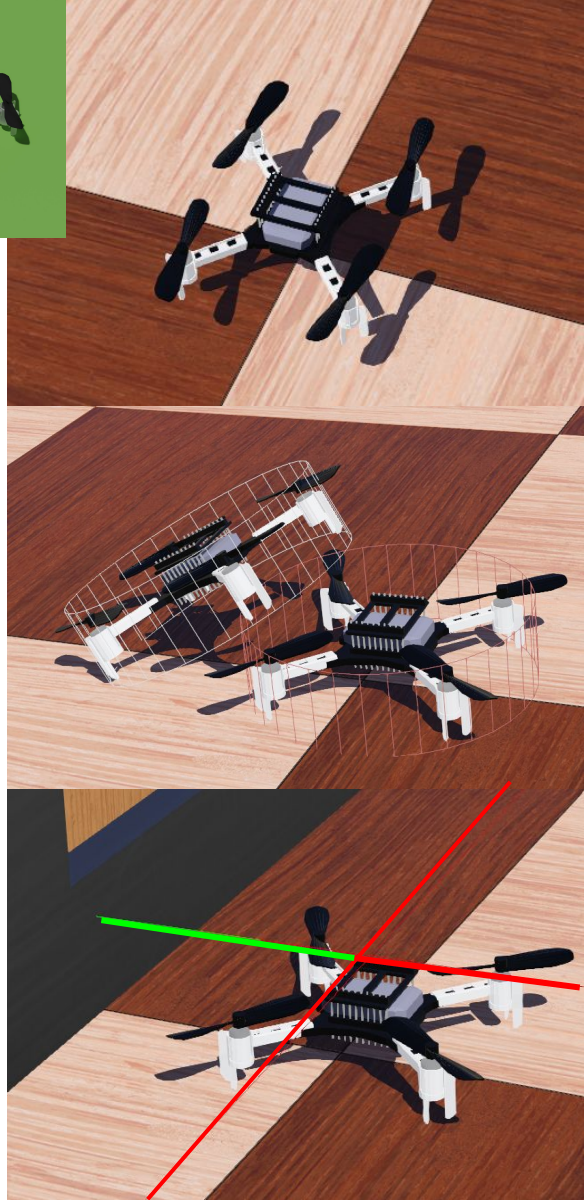


* gym-pybullet-drones | PyBullet-based Gym for single and multi-agent reinforcement learning with nano-quadcopters (utiasdsl.github.io)

Webots

- Cyberbotics
  - Spinoff EPFL
- Render model
- Collision Model
- Sensors
- Propeller physics
- Controller





https://www.cyberbotics.com/

Hands-on

Webots start up

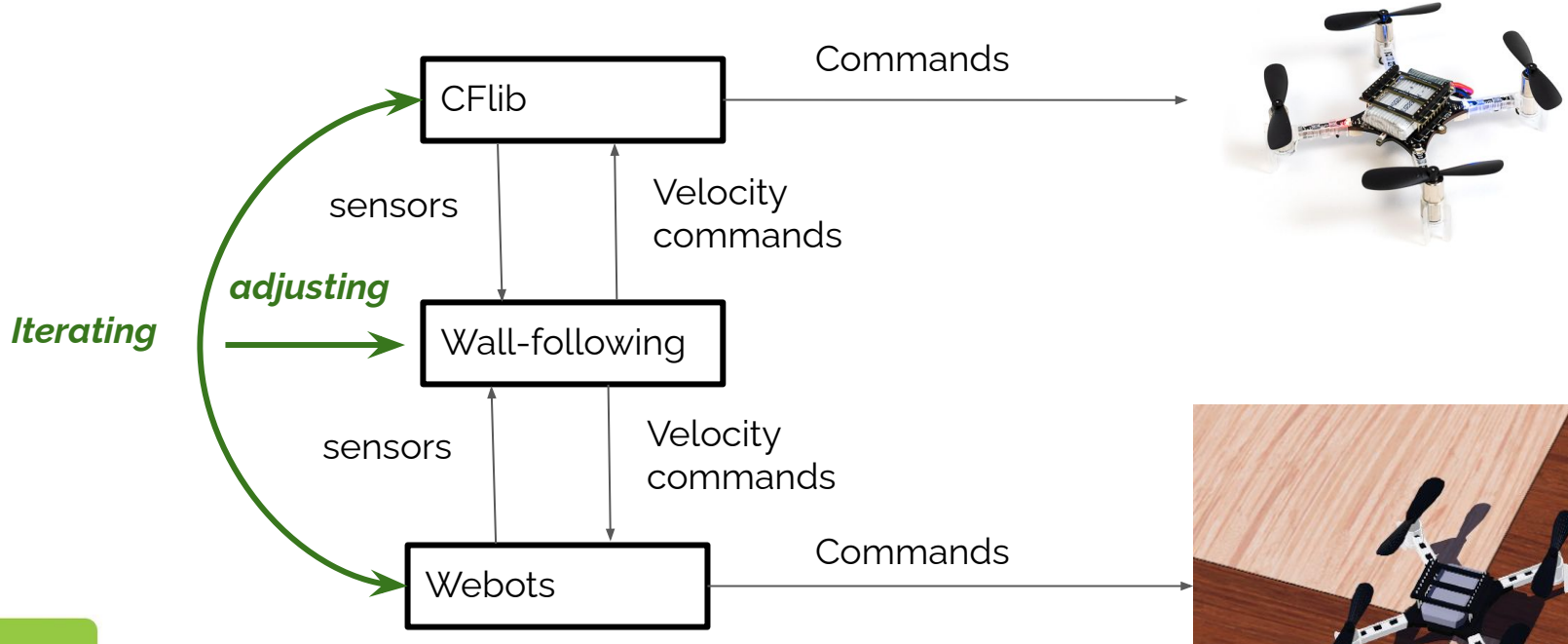
Select controller

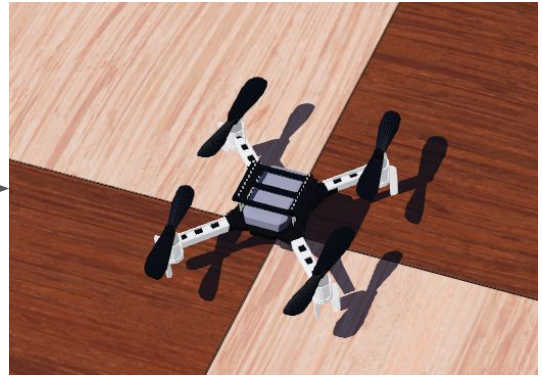
See crazyflie flying

Change controller to wall following,

https://youtu.be/es69Nf0Wlwc

Simulation to to real drone





Hands-on

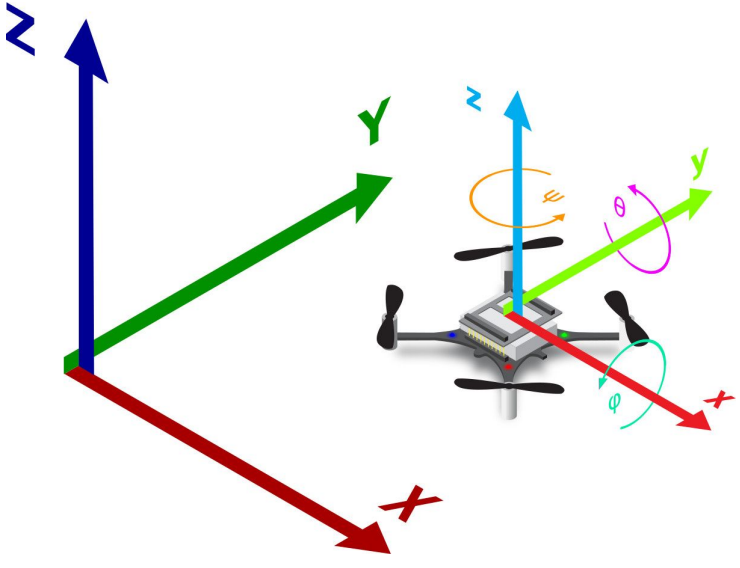
Cardboard walls on stage

Cflib wall following

https://youtu.be/es69Nf0Wlwc

Parameters Wall following Simulation vs Real

- Angle buffer
- Wall distance reference
- Special attention to axes (pitch negative)
- Mind the units (range meters, millimeters)

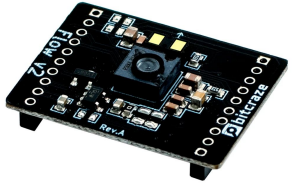


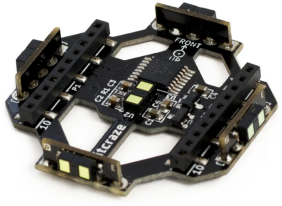
Needed to negate the yaw (bug)

Pros and Cons of off-board autonomy?

- Pros
  - Quick development
  - Easy switch between simulation and real drone
  - Easier language
  - Connect with other packages: opencv, ROS etc
- Cons
  - Communication bandwidth = swarms
  - Delay and speed
  - Need of an external computer

Avoidance (on-board)





Onboard autonomy

- App layer
  - Simulates ‘companion’ computer
- Port state machine to C code
- Micropython?
  - Not yet, but maybe next year :D

Video demo (in case of trouble)

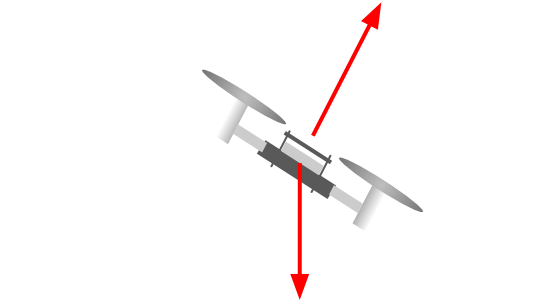
Onboard app layer wall following

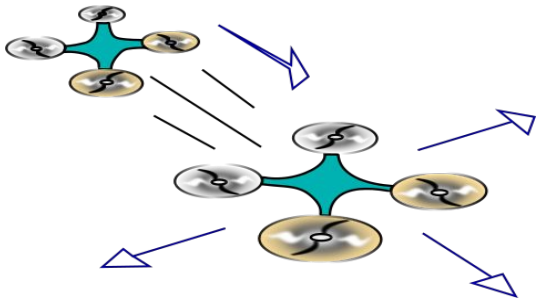
https://youtu.be/es69Nf0Wlwc

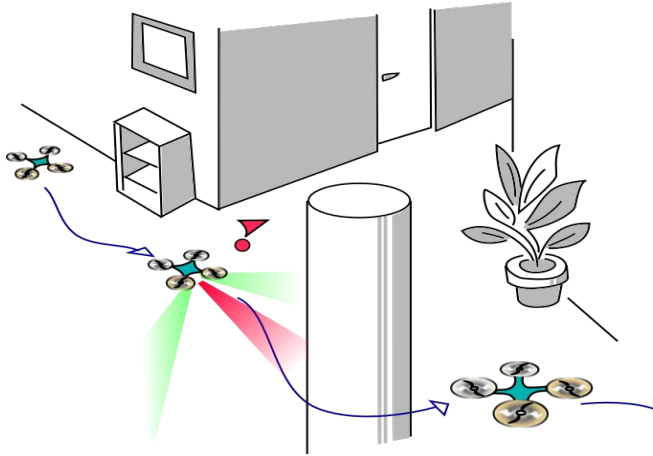
Autonomy levels

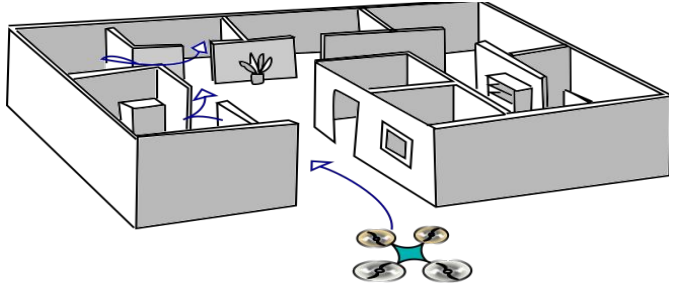
STAP

- **S**tability
- **T**ranslation
- **A**voidance
- **P**urpose





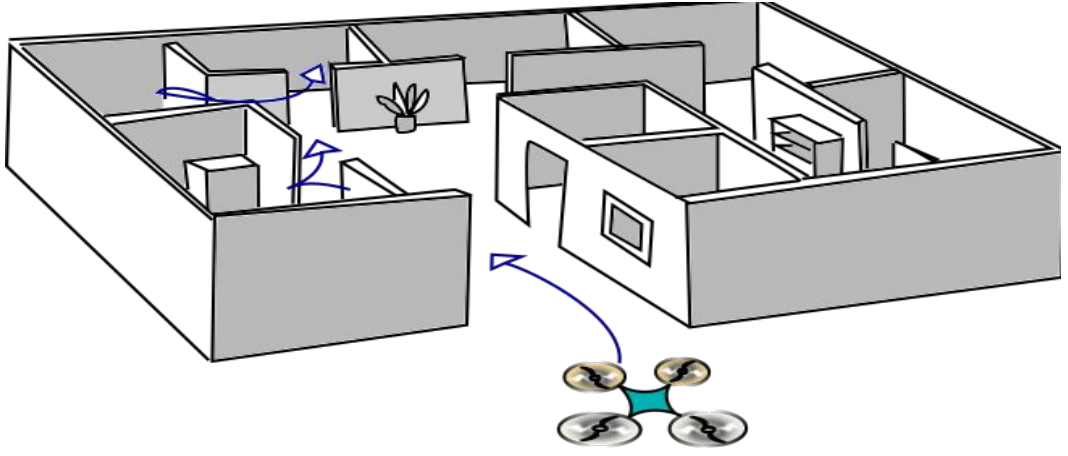




McGuire, Kimberly. "Indoor swarm exploration with Pocket Drones."
TU Delft PhD Dissertation (2019).

Purpose

- Up to you!



Thank you!

Website:	https://www.bitcraze.io/
Github repos:	https://github.com/bitcraze/
Support:	https://discussions.bitcraze.io/
Email:	contact@bitcraze.io
	kimberly@bitcraze.io