Introduction to the Crazyflie

Lecture at Aerial Robotics Course (EPFL)

bitcraze

Kimberly McGuire

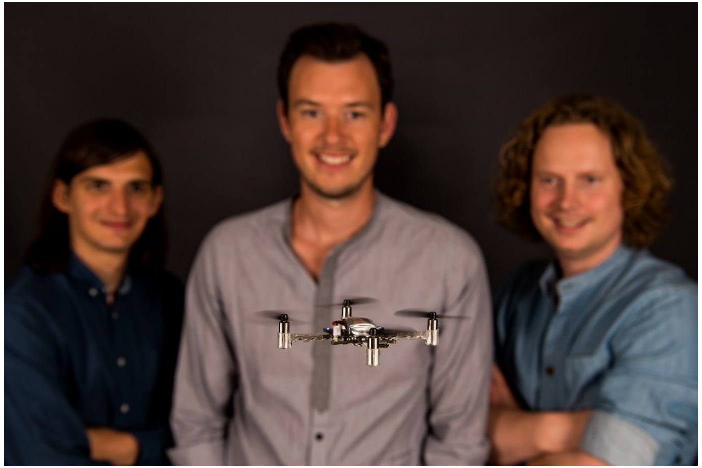
5th of April 2022

Bitcraze

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

Brief History of Bitcraze

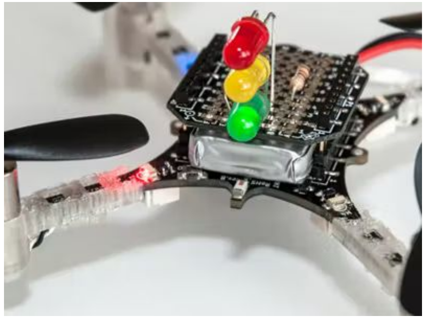
- Hobby project
- Crazyflie 1.0
- Company in 2011
- Crazyflie 2.X
- 10 year anniversary



bitcraze 10 Years

Who uses the Crazyflie?

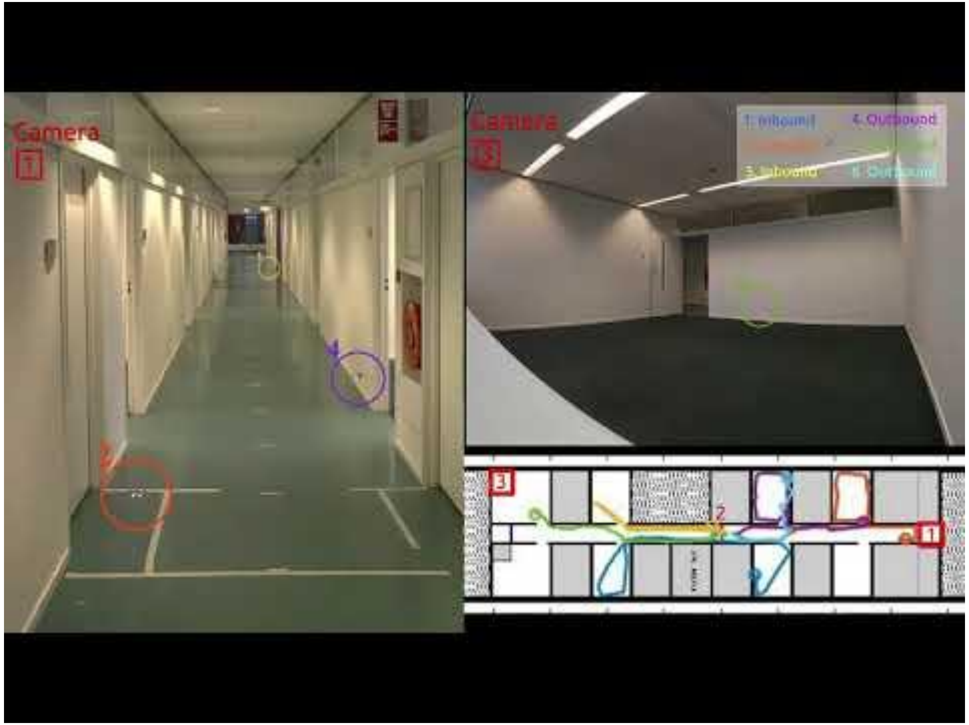
- Hobbyists
- Researchers
- Educators
- Shows designers

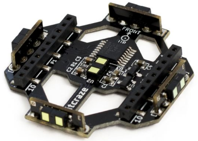


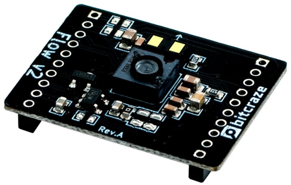
Ted-Talk

Raffaello d’Andrea: https://www.ted.com/talks/raffaello_d_andrea_meet_the_dazzling_flying_machines_of_the_future

PhD work



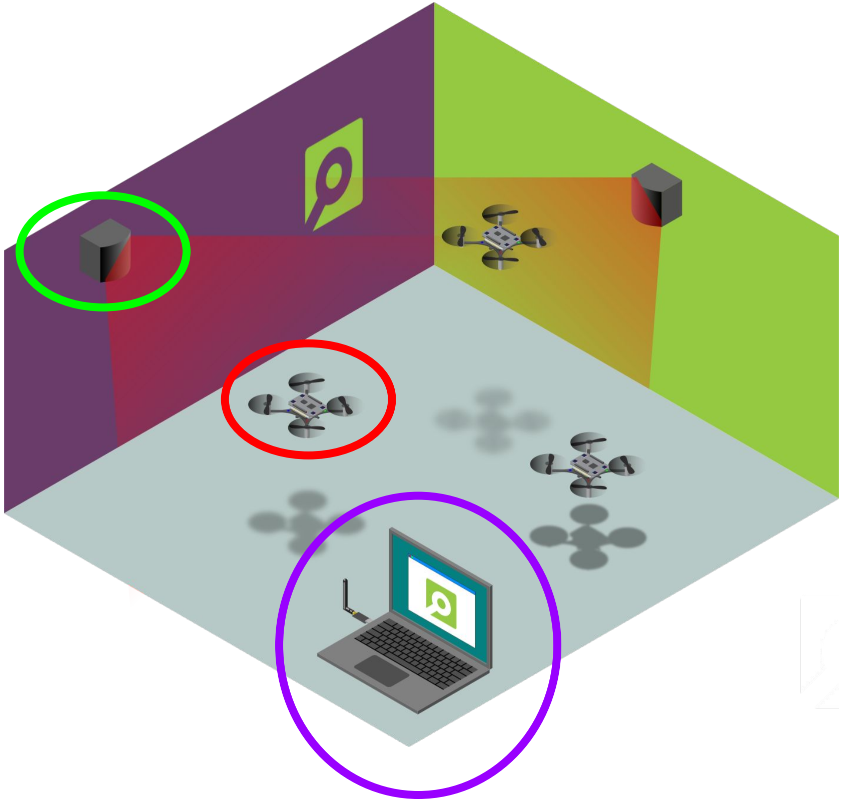




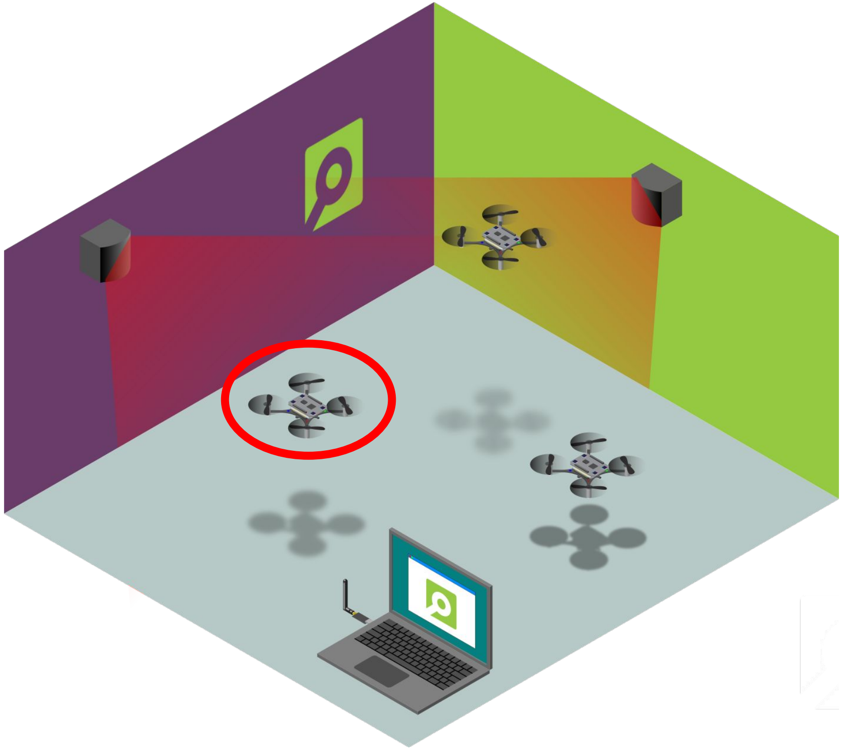
Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment (Science Robotics, 2019) K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen, https://youtu.be/jU4wsxwM1No

The Eco system

- Quadcopter
- Positioning
- Communication



The Quadcopter

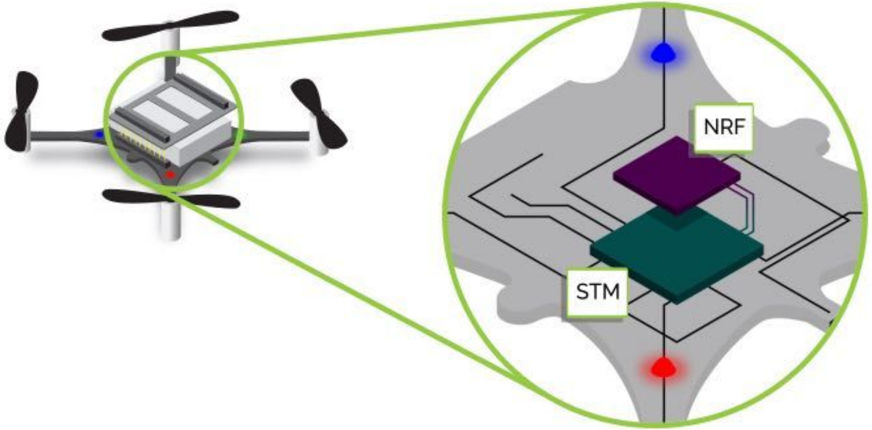


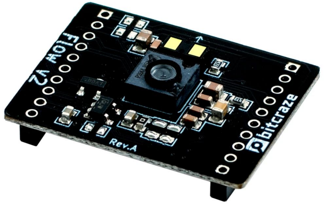
Crazyflie 2.1

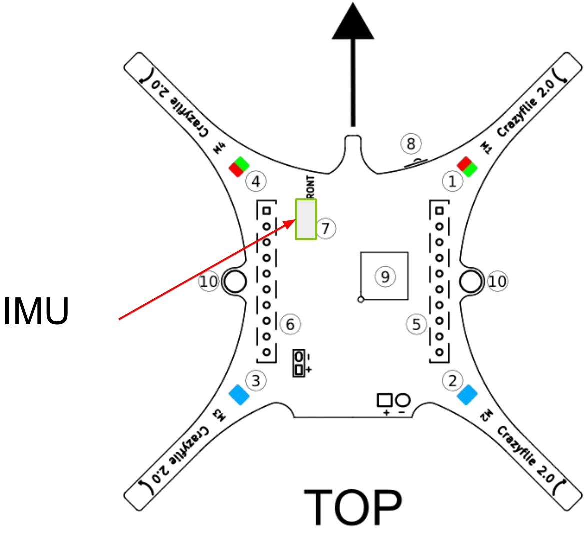
- Quadrotor
- 4 DC coreless motors
- Control board
- 1 cell lipo battery

Control board

- Two ARM processors
  - Autopilot Microprocessor
  - Communication Microprocessor
- Inertial Measurement Unit (IMU)
  - 3 axis accelerometer/ gyroscope
  - Pressure sensors
- Expansion decks



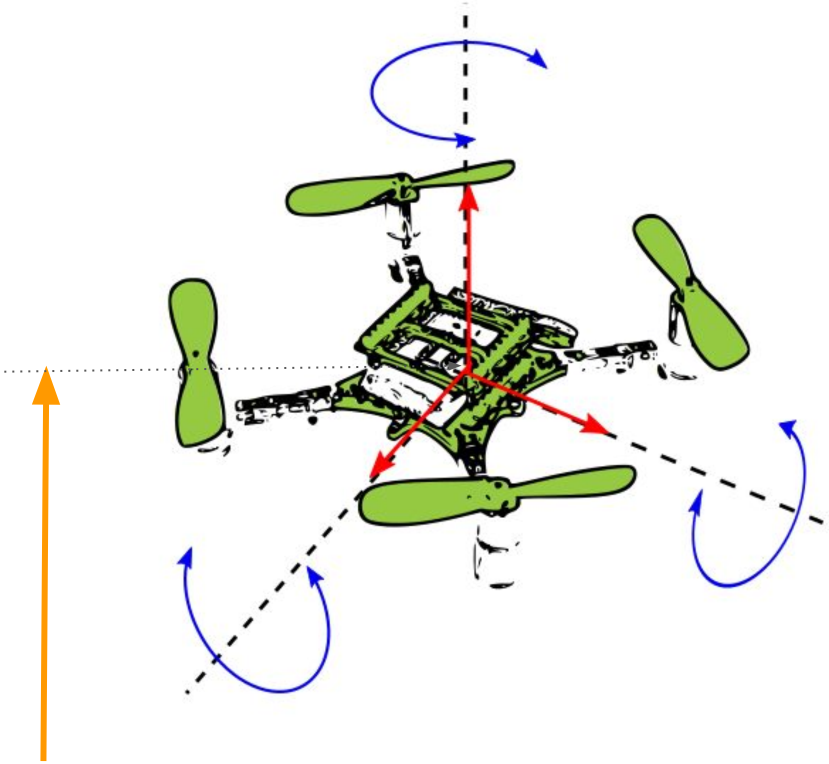




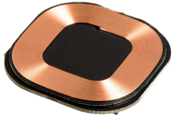
https://www.bitcraze.io/products/crazyflie-2-1/

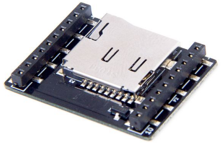
Inertial Measurement Unit (IMU)

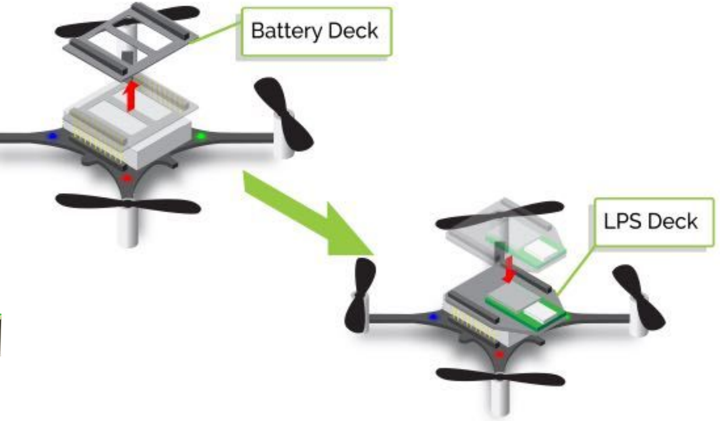
- Accelerometers
- Gyroscope
- *Pressure Sensor*
- Important for control
  - I will talk about that later!

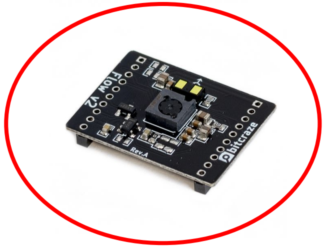


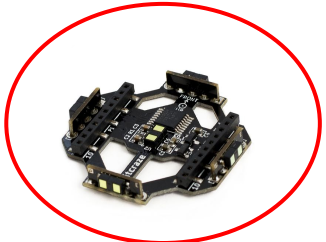
Expansion Decks

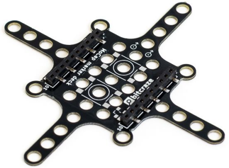




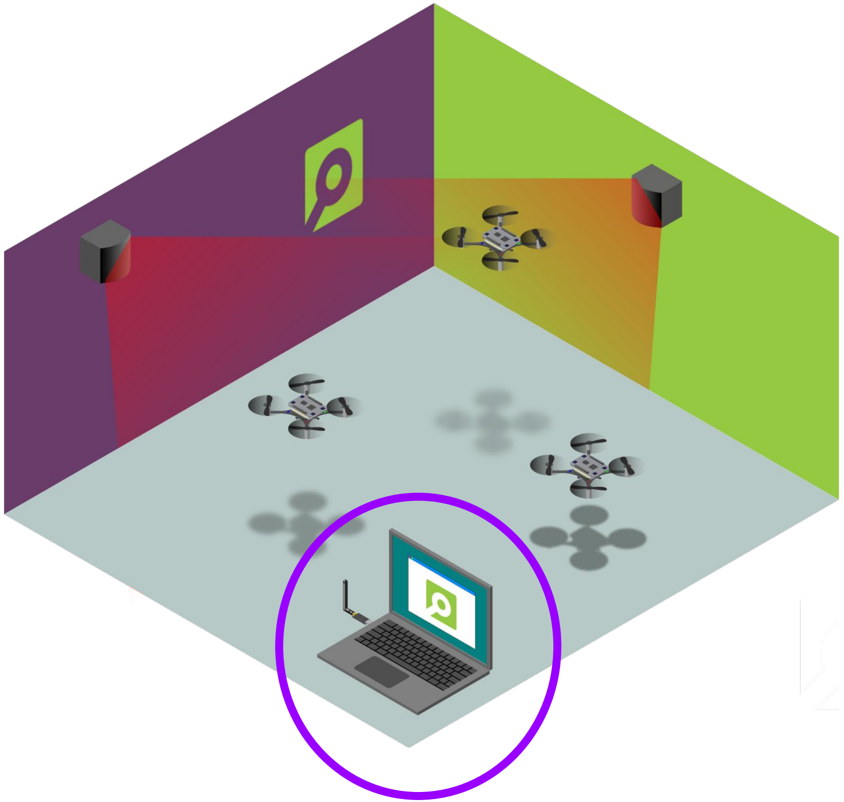




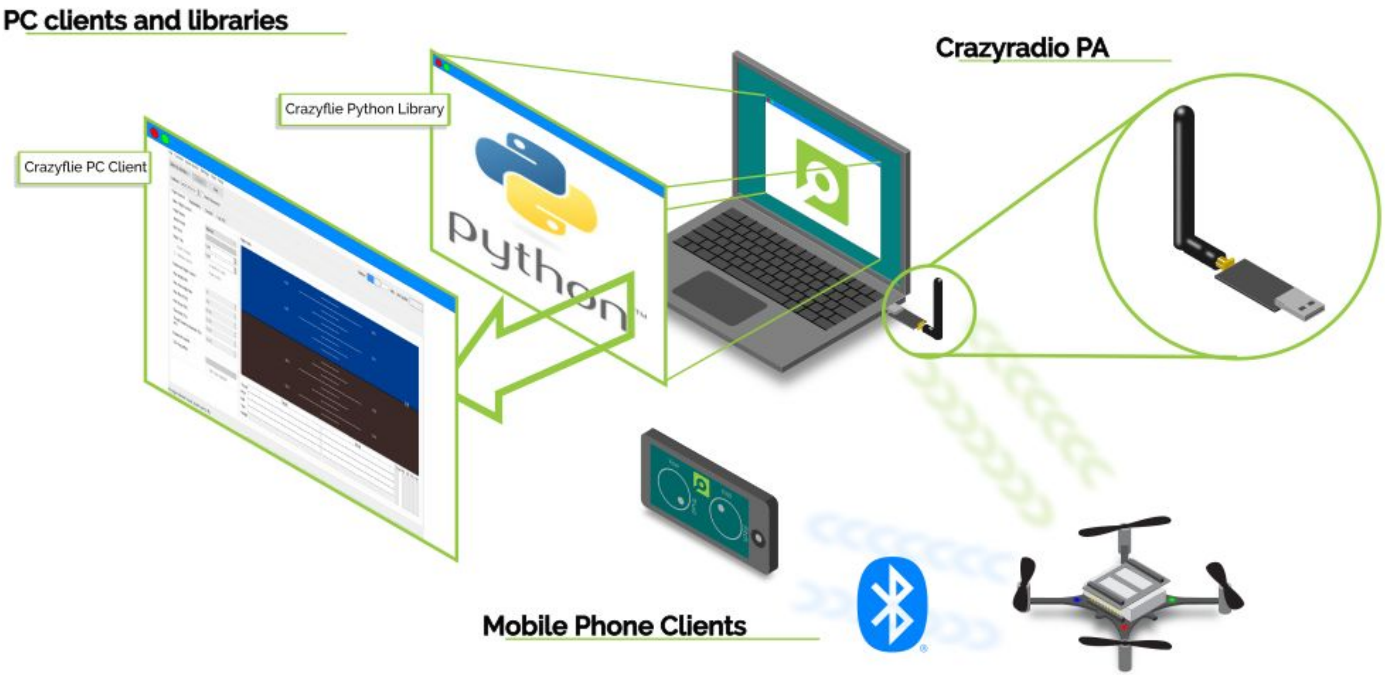




Communication



Client Software



Communication

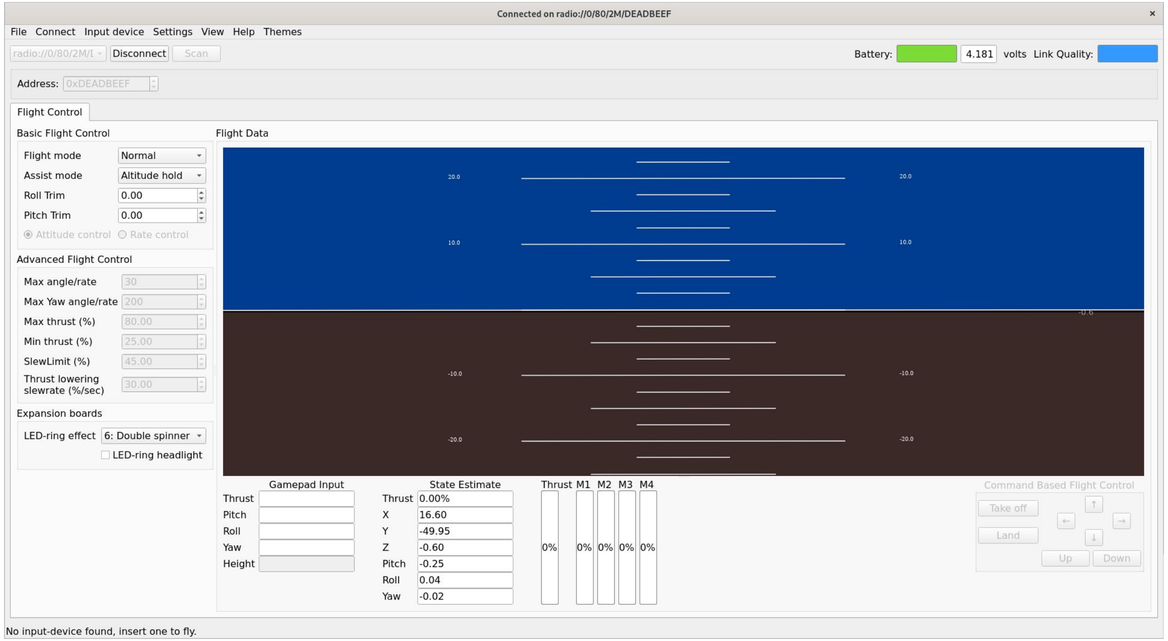
radio://0/80/2M/E7E7E7E7E7

- Crazyradio PA
  - Crazyradio Real-Time Protocol (CRTP)
- Unique URI
  - Medium
  - Channel
  - Communication Speed
  - Address
- Multiple Crazyflies
  - Different addresses

E7E7E7E701

E7E7E7E703

E7E7E7E702



Connected on radio://0/80/2M/DEADBEEF

radio://0/80/2M/I

Battery: 🟩 4.181 volts Link Quality: 🟦

Address: 0xDEADBEEF

Basic Flight Control

Flight mode	Normal
Assist mode	Altitude hold
Roll Trim	0.00
Pitch Trim	0.00

● Attitude control ○ Rate control

Advanced Flight Control

Max angle/rate	30
Max Yaw angle/rate	200
Max thrust (%)	80.00
Min thrust (%)	25.00
SlewLimit (%)	45.00
Thrust lowering slewrate (%/sec)	30.00

Expansion boards

LED-ring effect: 6: Double spinner

□ LED-ring headlight

Flight Data

Gamepad Input

Thrust	
Pitch	
Roll	
Yaw	
Height	

State Estimate

Thrust	0.00%
X	16.60
Y	-49.95
Z	-0.60
Pitch	-0.25
Roll	0.04
Yaw	-0.02

Thrust	M1	M2	M3	M4
0%	0%	0%	0%	0%

Command Based Flight Control

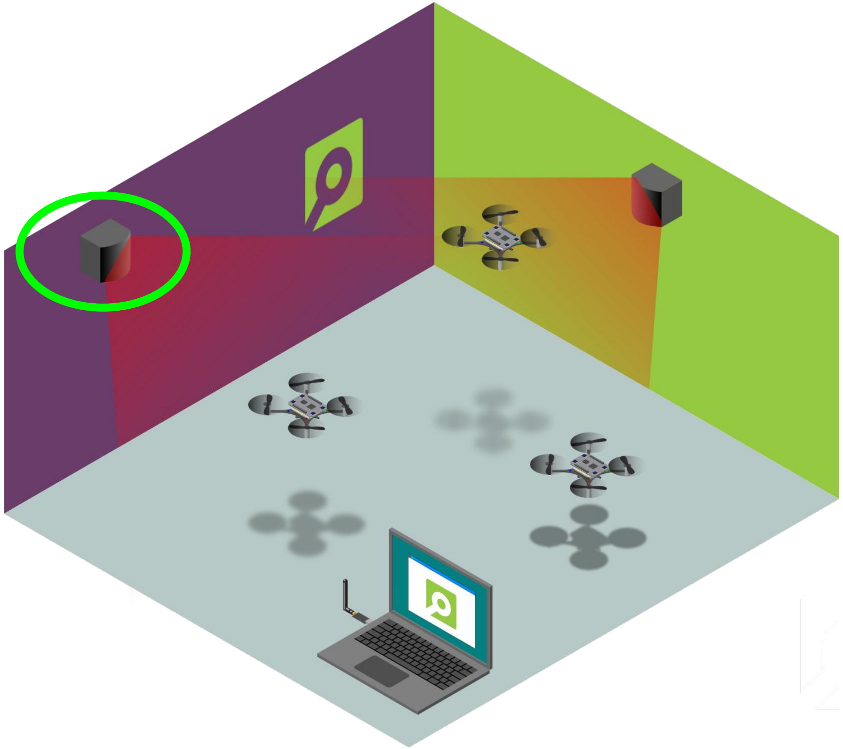
No input-device found, insert one to fly.

Hands-on

Start cfclient

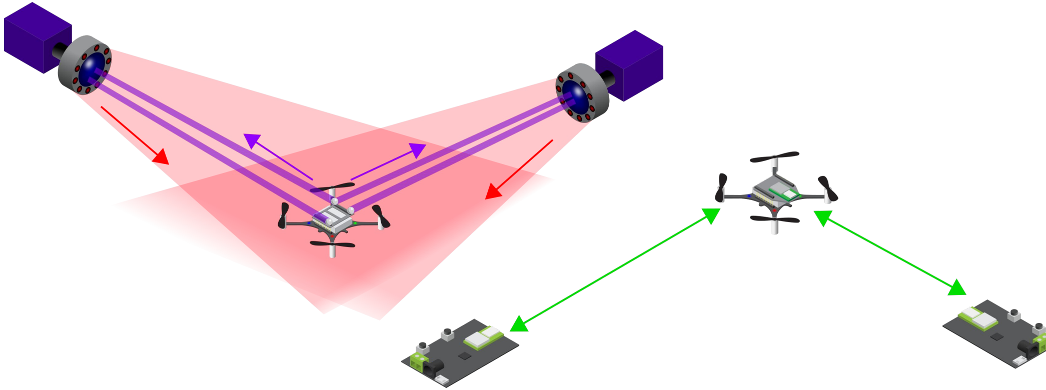
Show IMU values

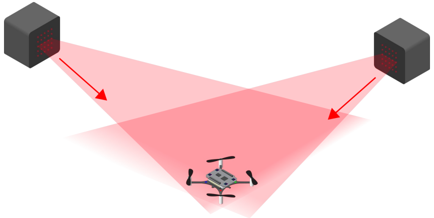
Positioning

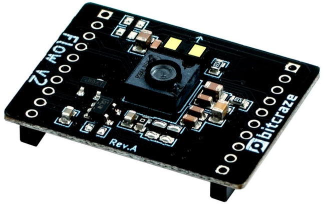


Positioning

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

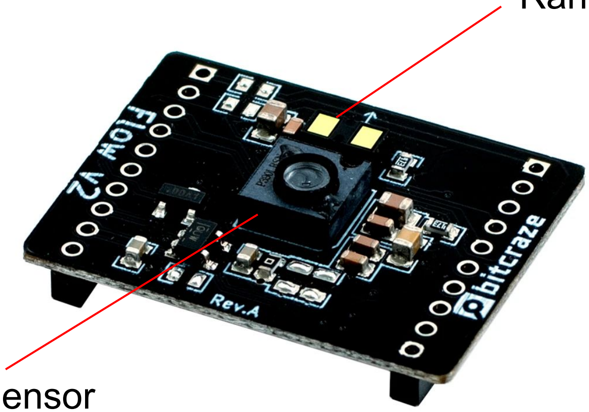






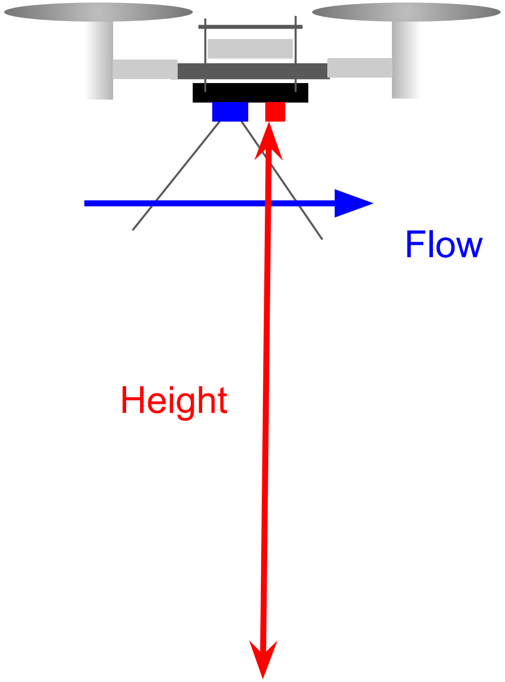
https://www.bitcraze.io/documentation/system/positioning/

Flowdeck



Range sensor

Optic Flow Sensor

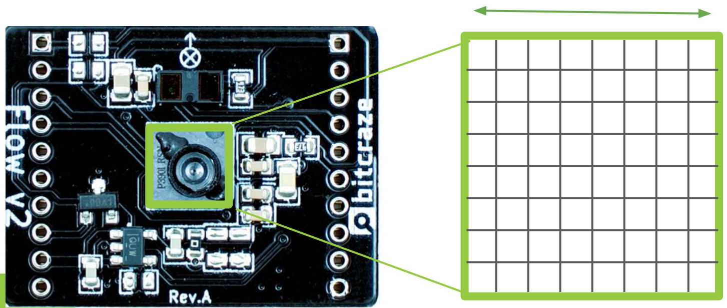


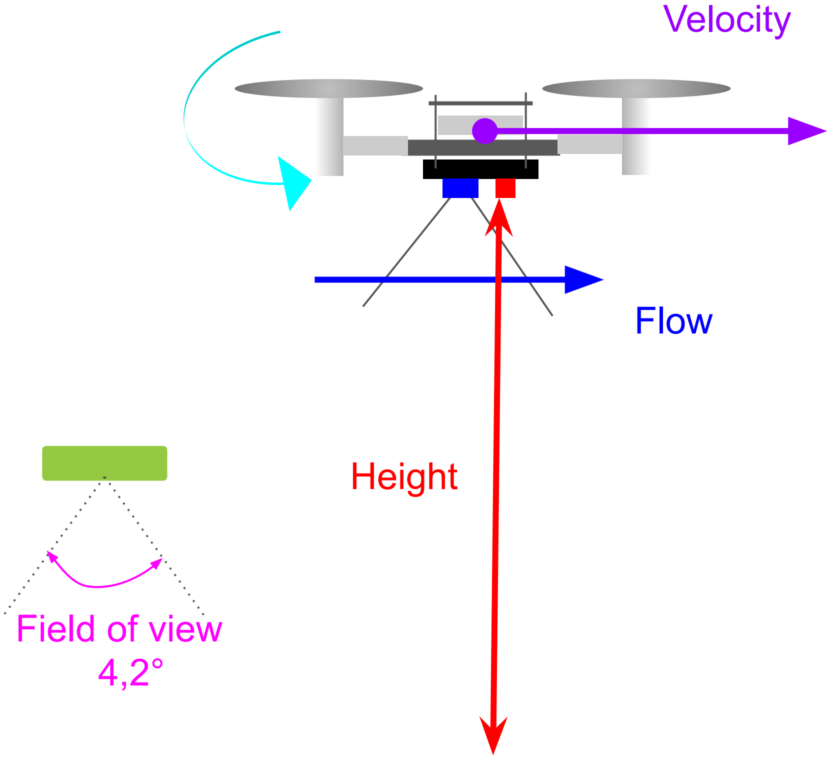
https://www.bitcraze.io/products/flow-deck-v2/

Velocity Flowdeck (EKF)

$$\boxed{\dot{x}} = \frac{\boxed{h} \cdot \boxed{\theta_{px}} \cdot \boxed{\Delta n_x}}{\Delta t \cdot \boxed{N_x}}$$

Sample time





Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation, M. Greiff, Master's thesis, Lund University, 2017

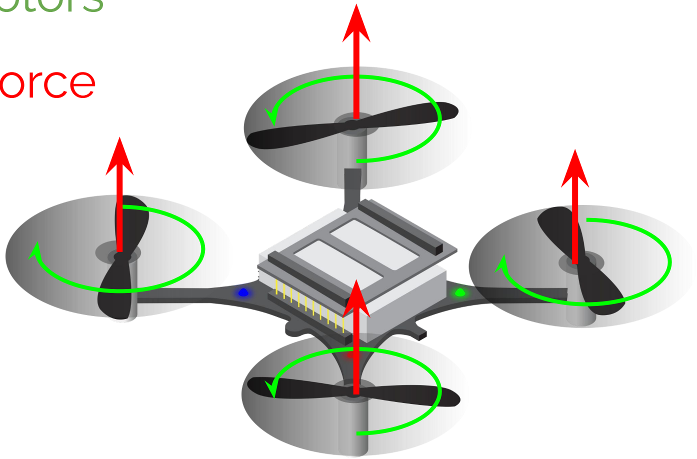
Hands-on

Show state estimate with the flowdeck on.

Ready to fly!?

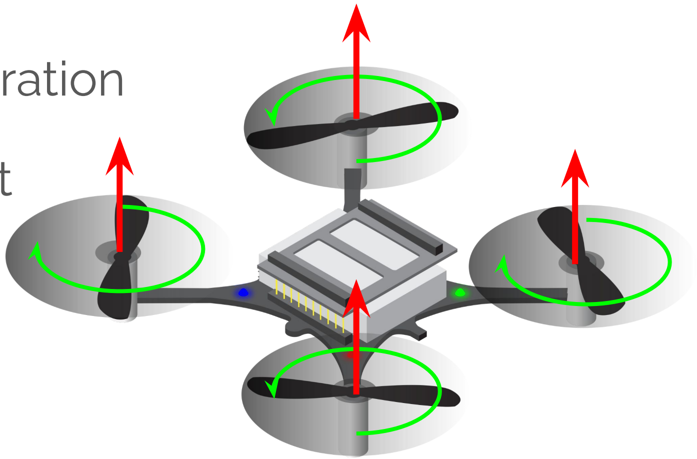
Quadcopter Dynamics

- Rotating motors
- Resulting Force

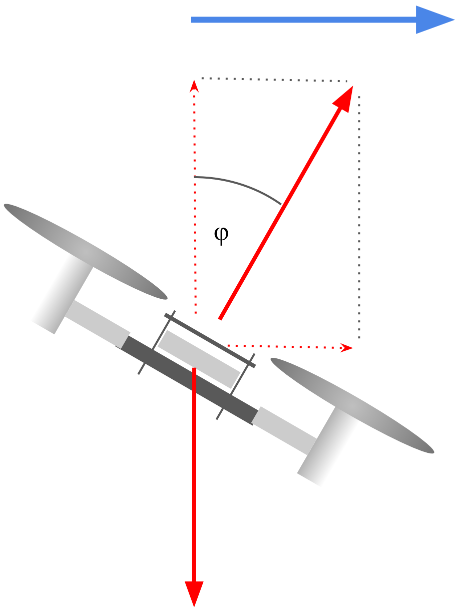


Quadcopter Dynamics

- Moments
- Linear acceleration
- Instable + drift

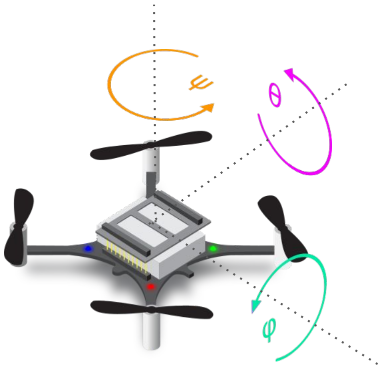


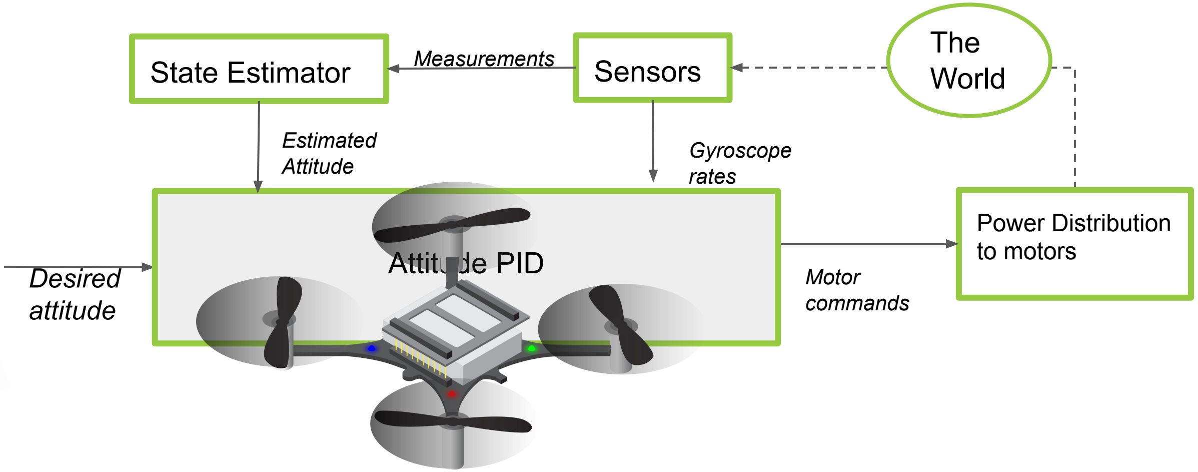
Linear acceleration



Attitude control

- Roll pitch Yaw
- Thrust





State Estimator

Measurements

Sensors

The World

Estimated Attitude

Gyroscope rates

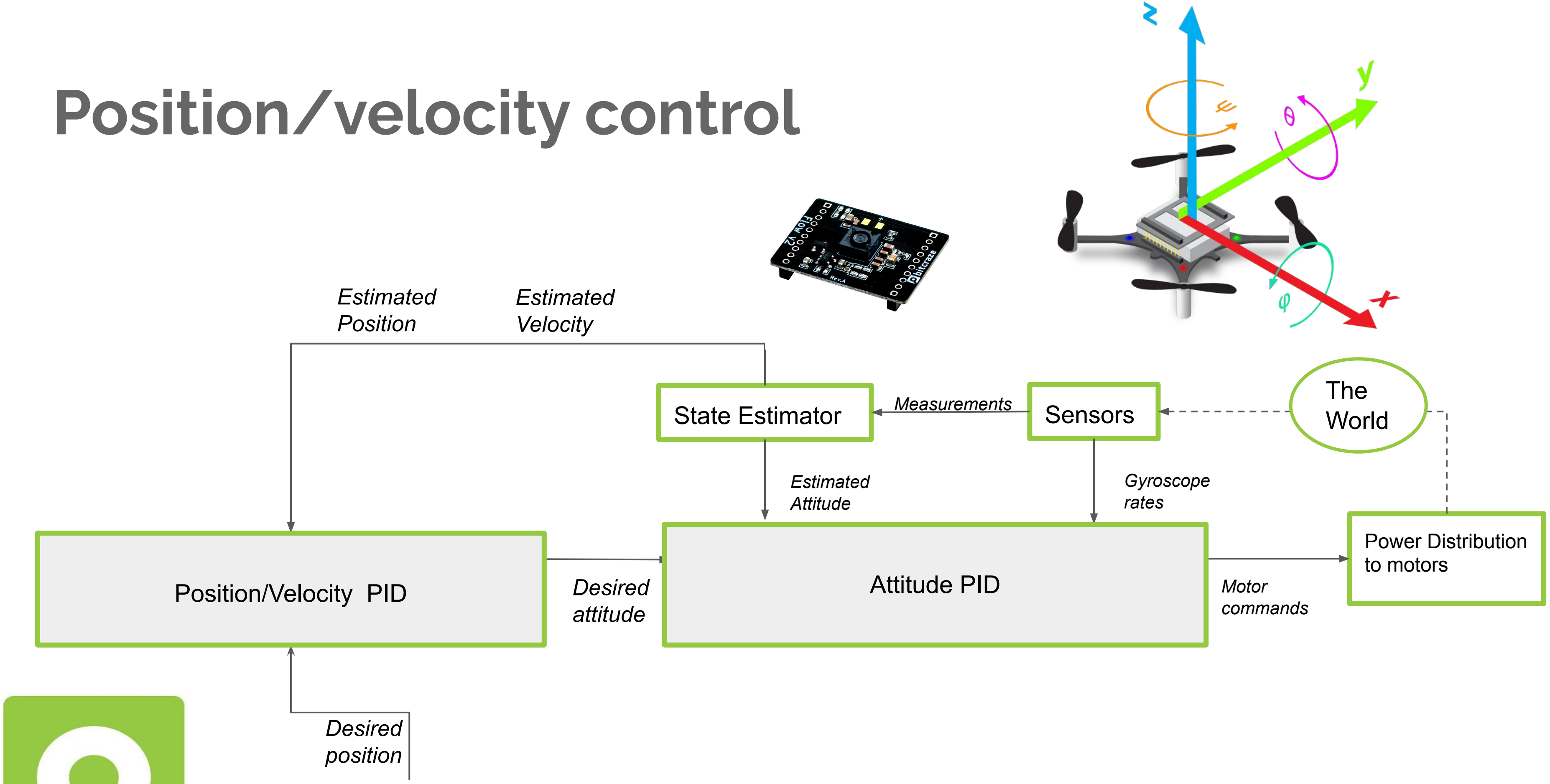
Desired attitude

Attitude PID

Motor commands

Power Distribution to motors

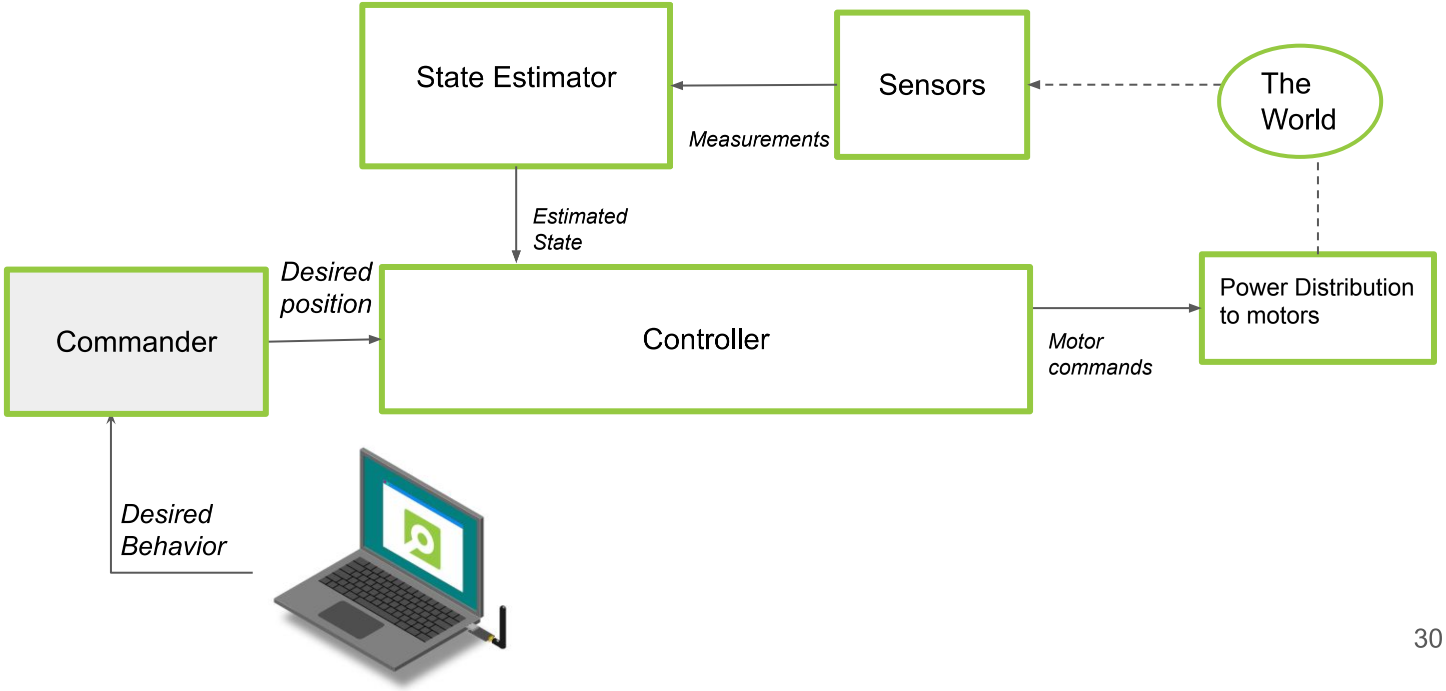
Position/velocity control



Hands-on

Let it fly with the cfclient

Commander framework



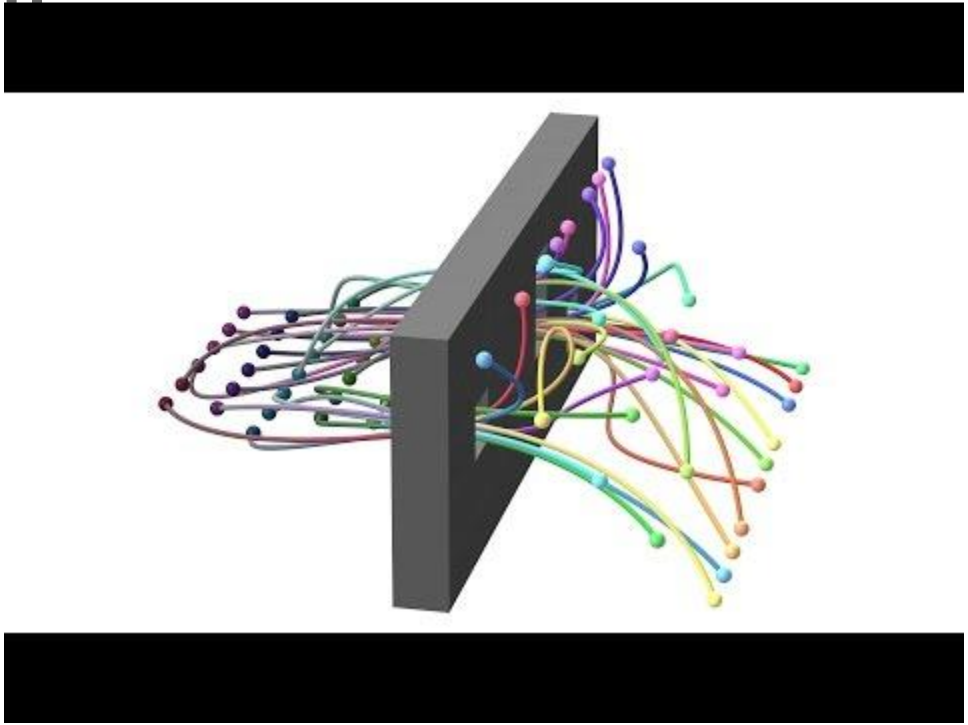
Type of Commanders

- Attitude commander
- Position/velocity commander
- Trajectory commander (planner)

LIS lab

Soria, E., Schiano, F. & Floreano, D. Predictive control of aerial swarms in cluttered environments. Nat Mach Intell (2021). https://youtu.be/cAXUKNGpMG4

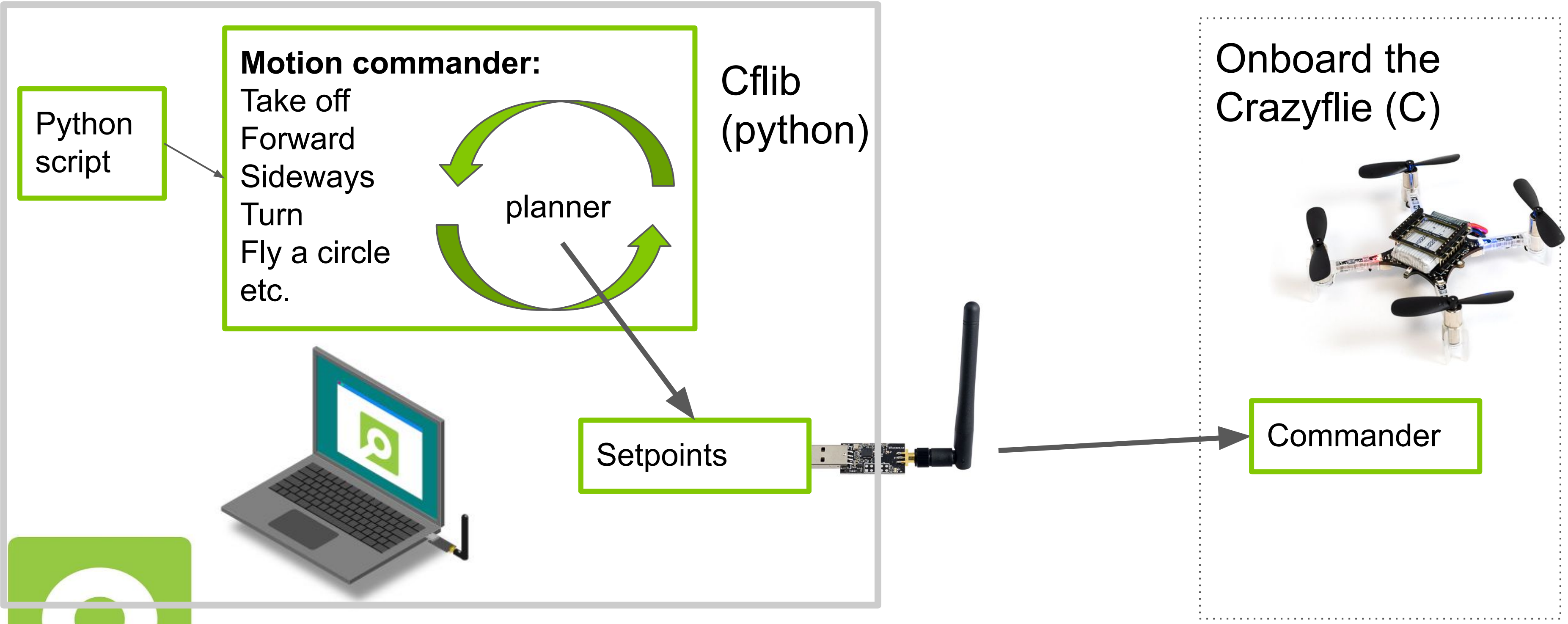
Crazyswarm



Preiss, James A., et al. "Downwash-aware trajectory planning for large quadrotor teams." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017 https://youtu.be/YnGZ-arUwgc

.

Crazyflie python library (CFlib)



Python script

Motion commander:
Take off
Forward
Sideways
Turn
Fly a circle
etc.

planner

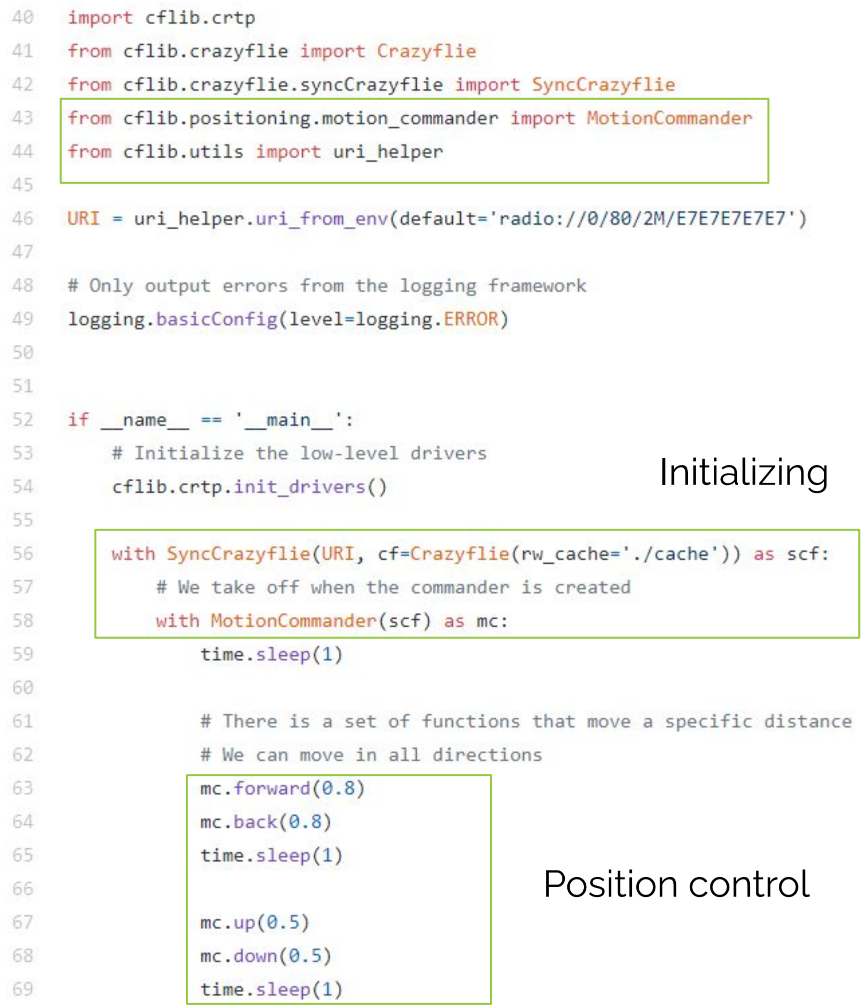
Cflib (python)

Setpoints

Onboard the Crazyflie (C)

Commander

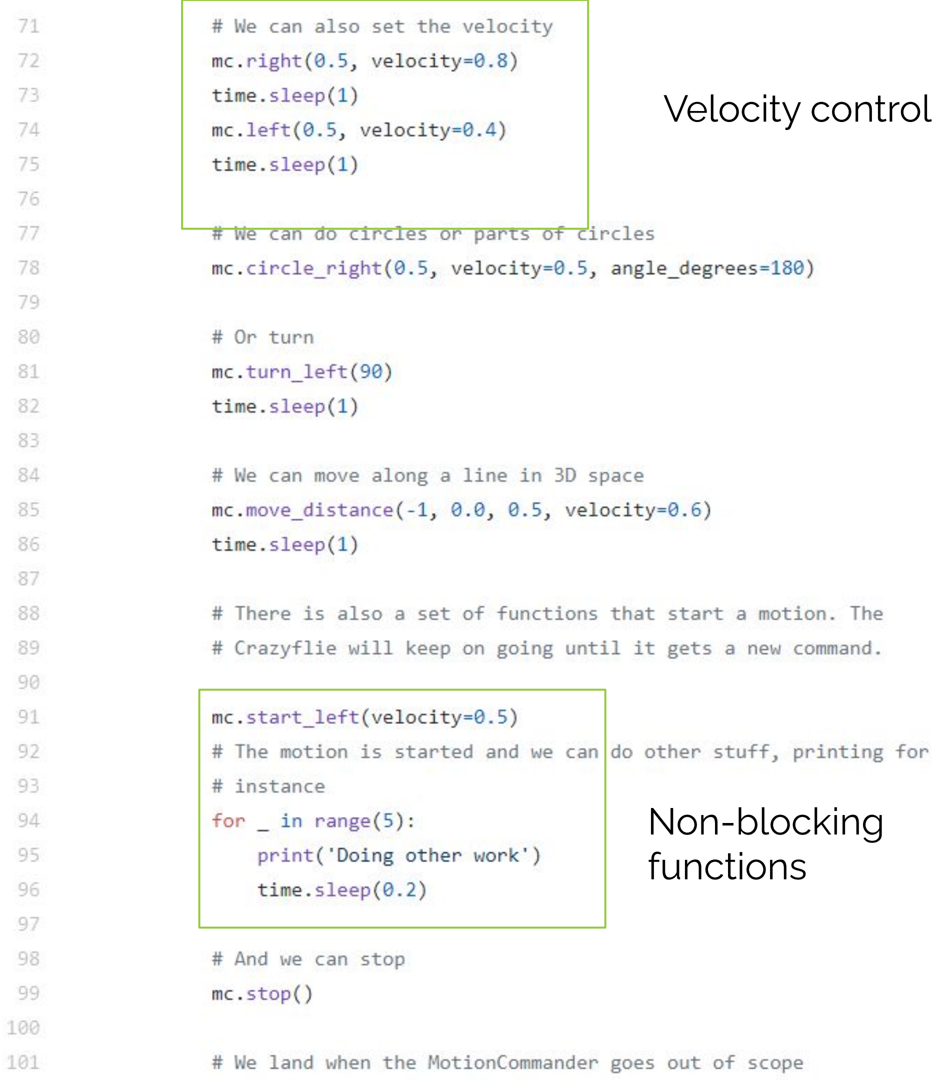
motion_commander_demo.py



```
import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.positioning.motion_commander import MotionCommander
from cflib.utils import uri_helper

URI = uri_helper.uri_from_env(default='radio://0/80/2M/E7E7E7E7E7')

# Only output errors from the logging framework
logging.basicConfig(level=logging.ERROR)


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

    with SyncCrazyflie(URI, cf=Crazyflie(rw_cache='./cache')) as scf:
        # We take off when the commander is created
        with MotionCommander(scf) as mc:
            time.sleep(1)

            # There is a set of functions that move a specific distance
            # We can move in all directions
            mc.forward(0.8)
            mc.back(0.8)
            time.sleep(1)

            mc.up(0.5)
            mc.down(0.5)
            time.sleep(1)
```

Initializing

Position control



```
            # We can also set the velocity
            mc.right(0.5, velocity=0.8)
            time.sleep(1)
            mc.left(0.5, velocity=0.4)
            time.sleep(1)

            # We can do circles or parts of circles
            mc.circle_right(0.5, velocity=0.5, angle_degrees=180)

            # Or turn
            mc.turn_left(90)
            time.sleep(1)

            # We can move along a line in 3D space
            mc.move_distance(-1, 0.0, 0.5, velocity=0.6)
            time.sleep(1)

            # There is also a set of functions that start a motion. The
            # Crazyflie will keep on going until it gets a new command.

            mc.start_left(velocity=0.5)
            # The motion is started and we can do other stuff, printing for
            # instance
            for _ in range(5):
                print('Doing other work')
                time.sleep(0.2)

            # And we can stop
            mc.stop()

            # We land when the MotionCommander goes out of scope
```

Velocity control

Non-blocking functions

Hands-on

Simple behavior motion commander script (`motion_commander_demo.py`)

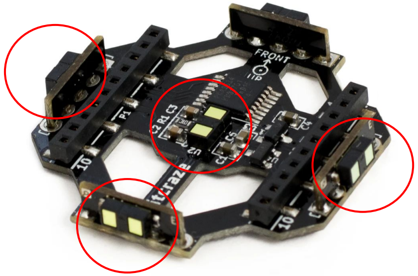
Demos can be found in crazyflie-lib-python/examples

API documentation:

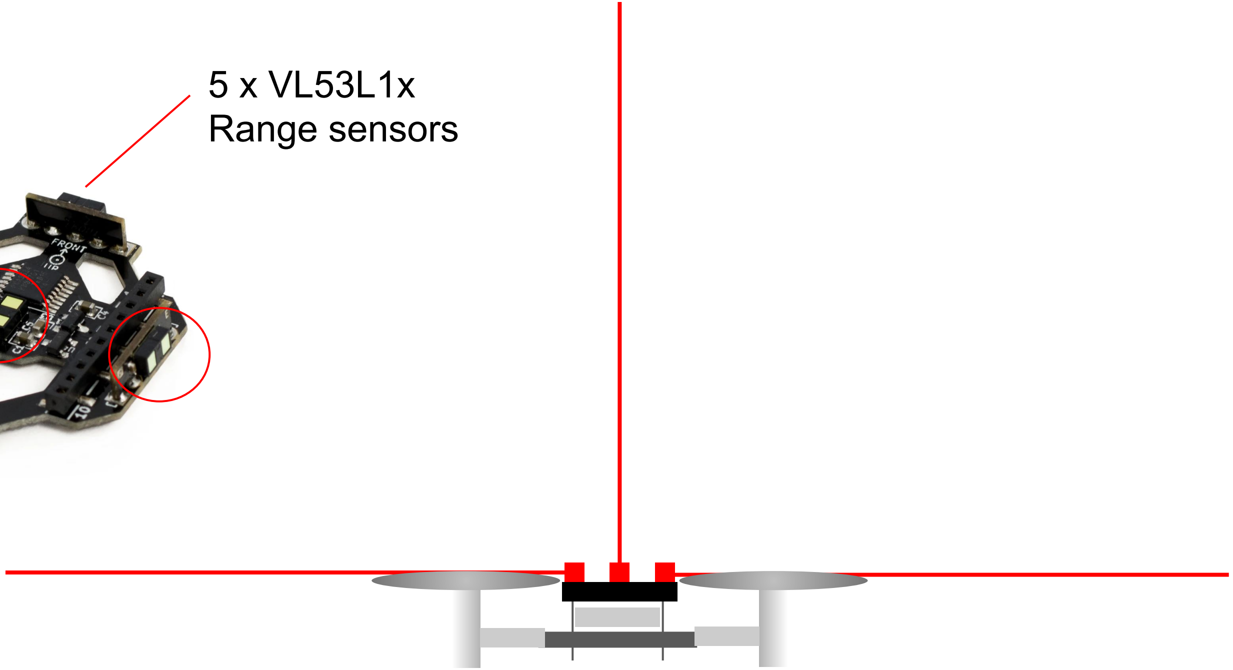
https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/positioning/motion_commander/

External Sensing

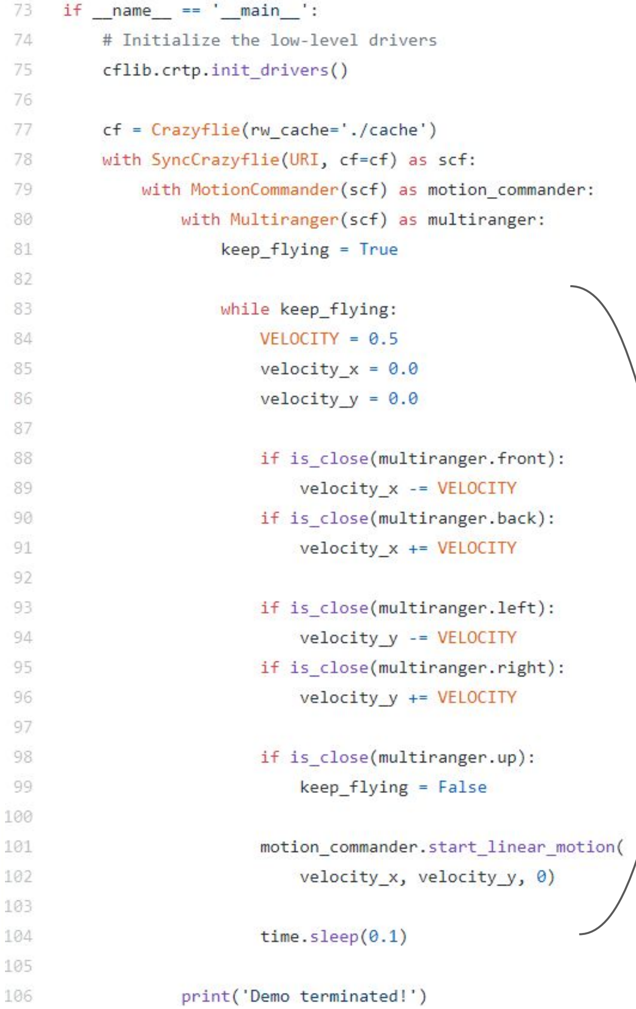
Multiranger deck



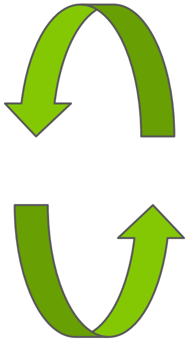
5 x VL53L1x
Range sensors



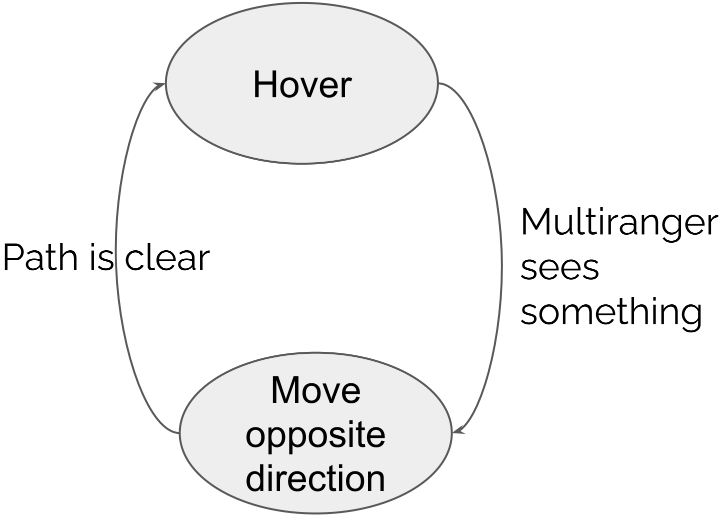
multiranger_push.py



```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!')
```



State machine



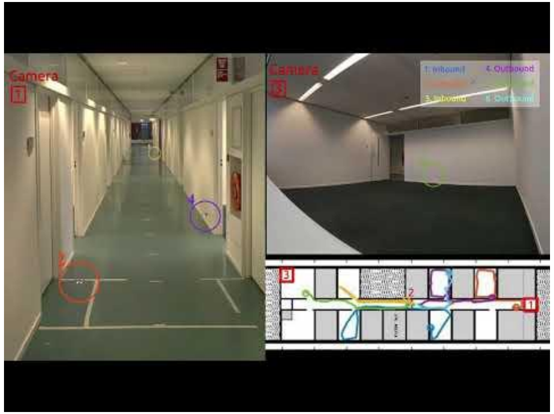
Hands-on

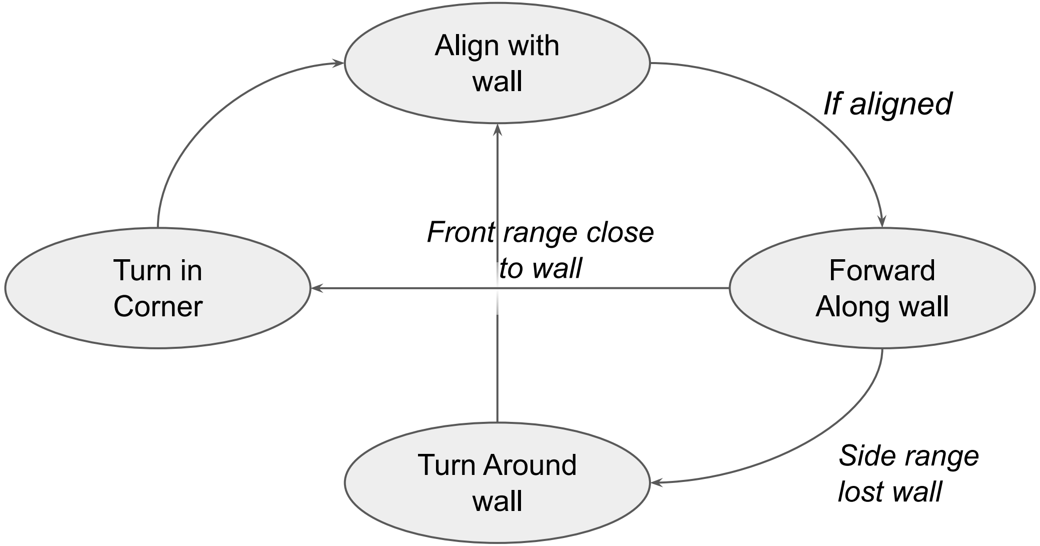
Push Demo with crazyflie and multiranger (multiranger_push.py)

Demos can be found in crazyflie-lib-python/examples

Wall following

-





Align with wall

If aligned

Forward Along wall

Front range close to wall

Turn in Corner

Side range lost wall

Turn Around wall

* 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.

Tips & Tricks

Lipo battery: Do not deplete your battery completely! Red LED on means landing!

Before flight checklist: Make sure to first check if flow + multiranger is attached before take off. Easier to do this programmatically!

The distance sensors are based on infrared light, so don’t fly in direct sunlight or else it detects objects that are not there

Thank you for listening!

Website: www.bitcraze.io

Forum: forum.bitcraze.io

Email: contact@bitcraze.io

kimberly@bitcraze.io