Introduction to the Crazyflie

Lecture at Aerial Robotics Course (EPFL)

Kimberly McGuire

27th of April 2021

Introduction to Bitcraze AB

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

History of Bitcraze

- Hobby project
- Company in 2009
- Crazyflie 1.0
- Crazyflie 2.X

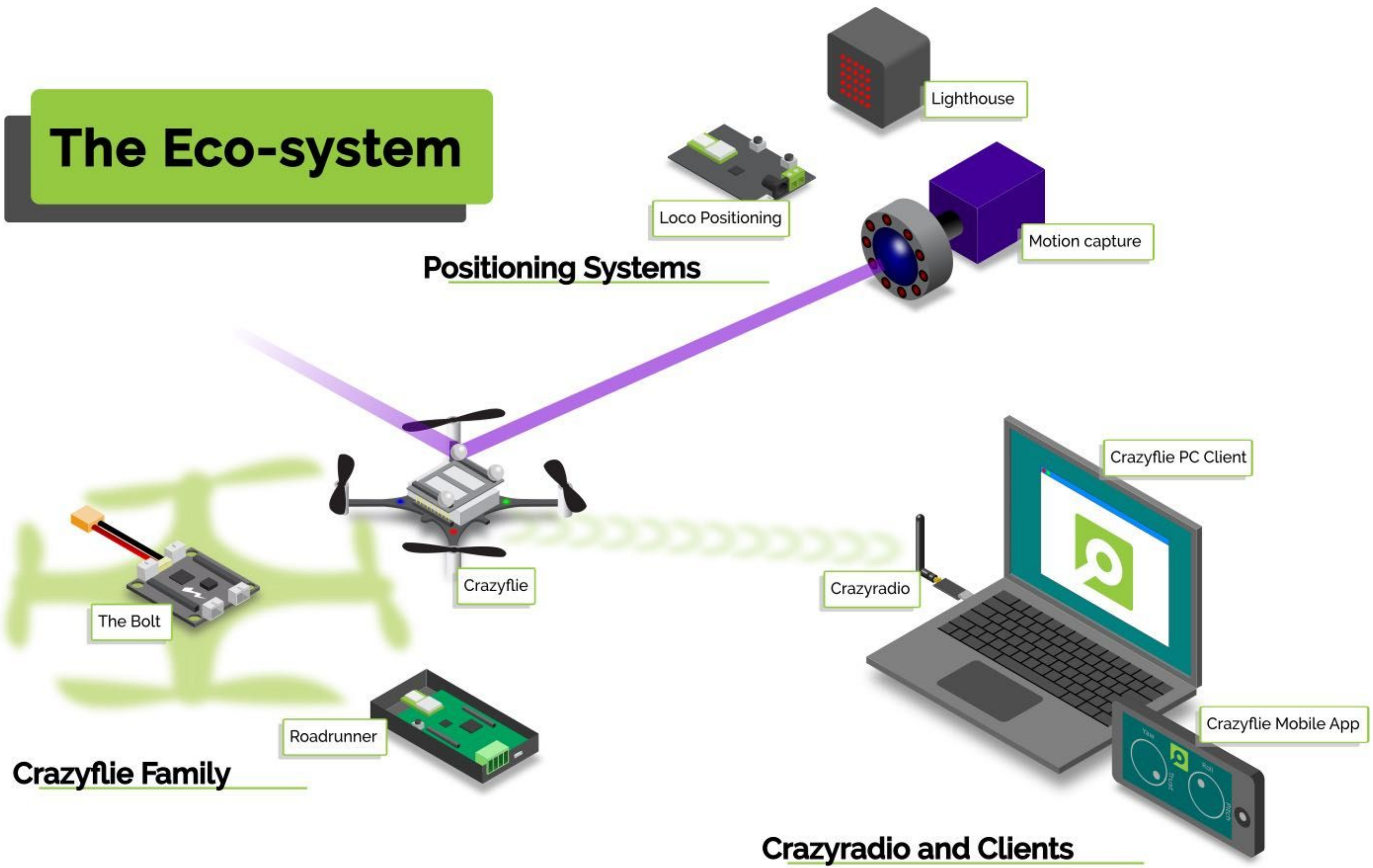
Who uses the Crazyflie?

- 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

The Eco-system

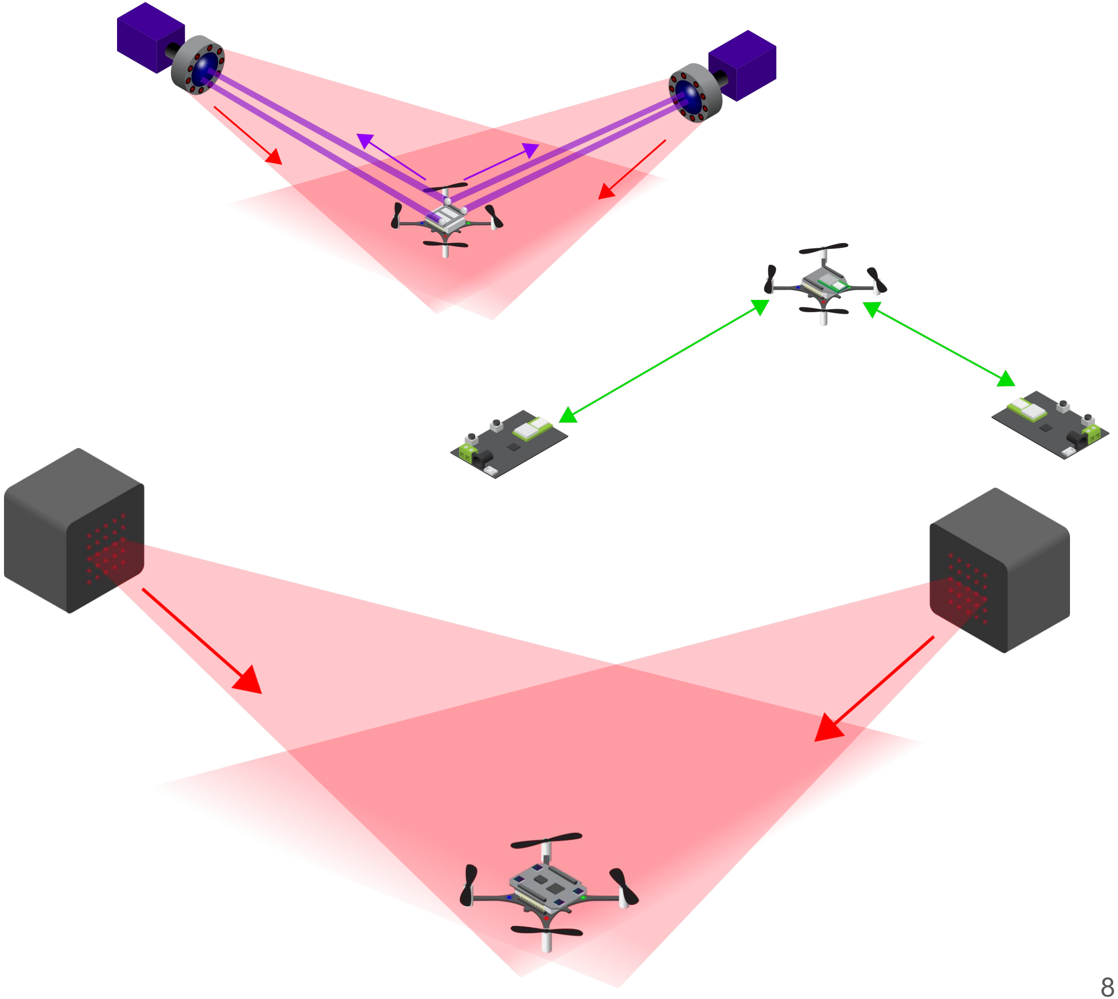


Crazyflie

- Quadrotor
- 4 DC coreless motors
- Battery

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*

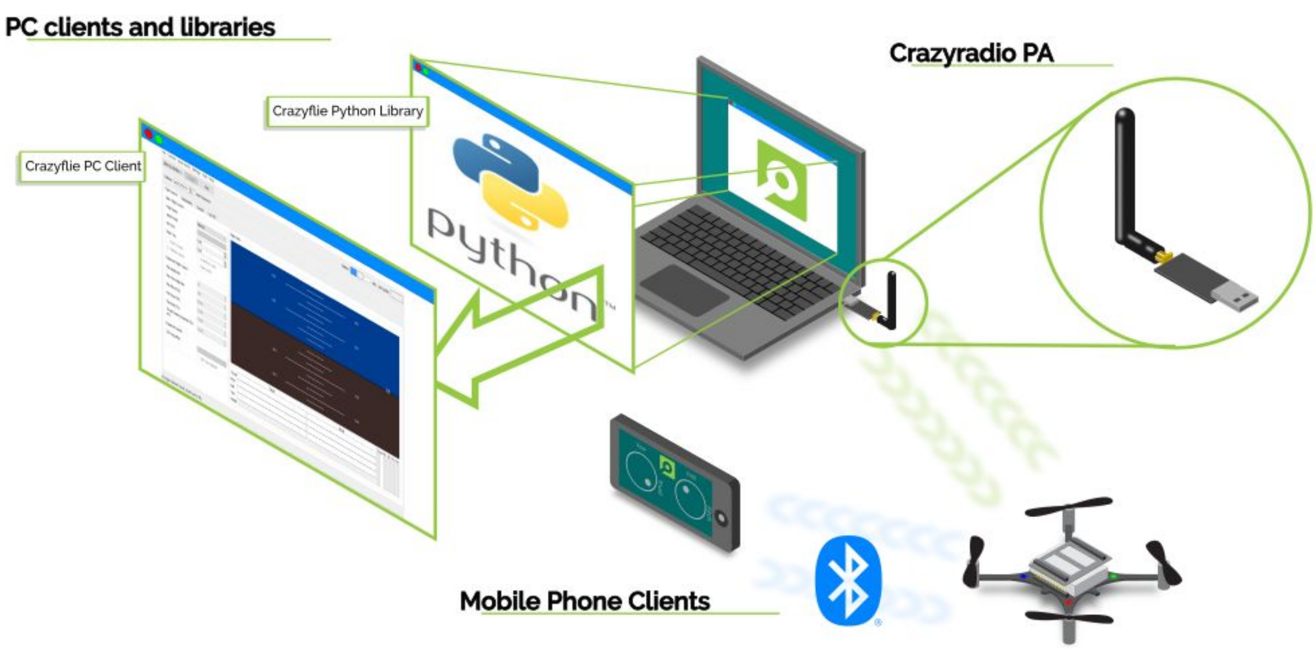


Demonstration Lighthouse

Show lighthouse positioning in action!

Github: crazyflie-lib-python/examples/autonomous_sequence_high_level.py

Client Software



Communication

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

- Crazyradio PA
  - Crazyradio Real-Time Protocol (CRTP)
- Unique URI
  - Medium
  - Channel
  - Communication Speed
  - Address
- Broadcast to multiple Crazyflies
  - Sure, as long as you are on the same channel

0xE7E7E7E701

0xE7E7E7E702

0xE7E7E7E703

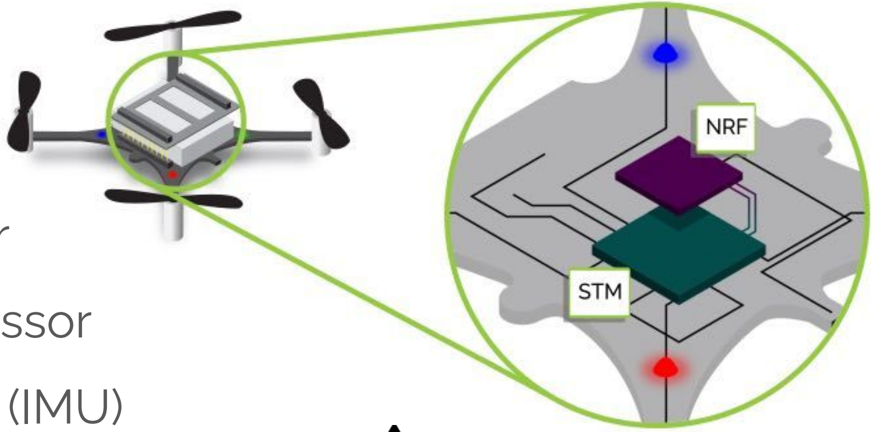
HANDS-ON

Connect to the Crazyflie

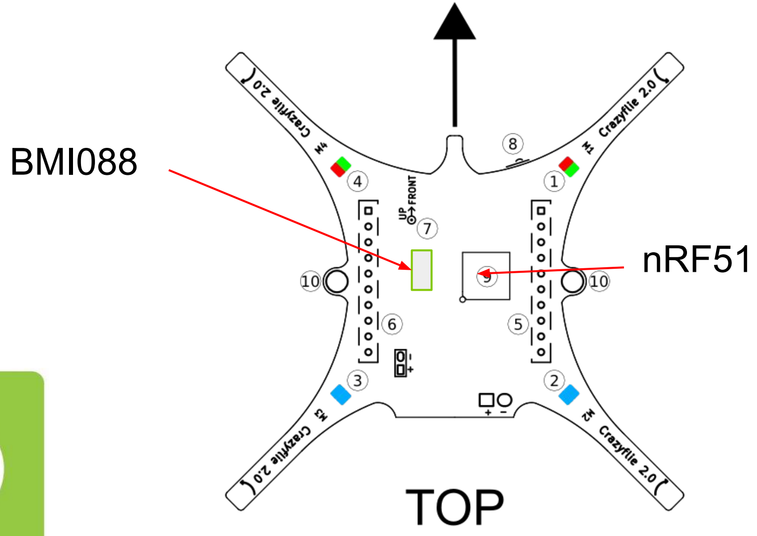
Show the CF client

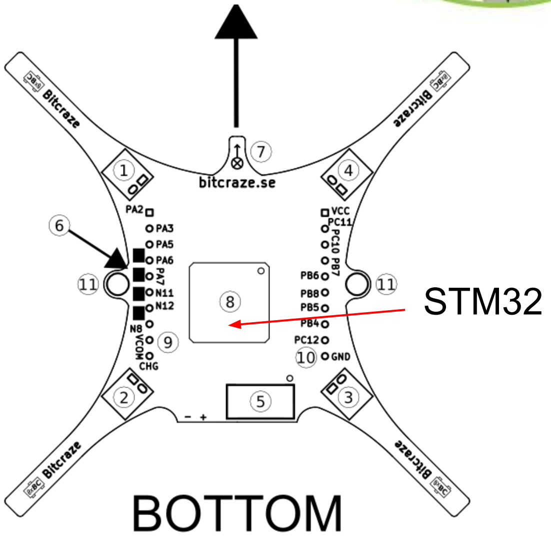
https://github.com/bitcraze/crazyflie-clients-python

Back to the hardware

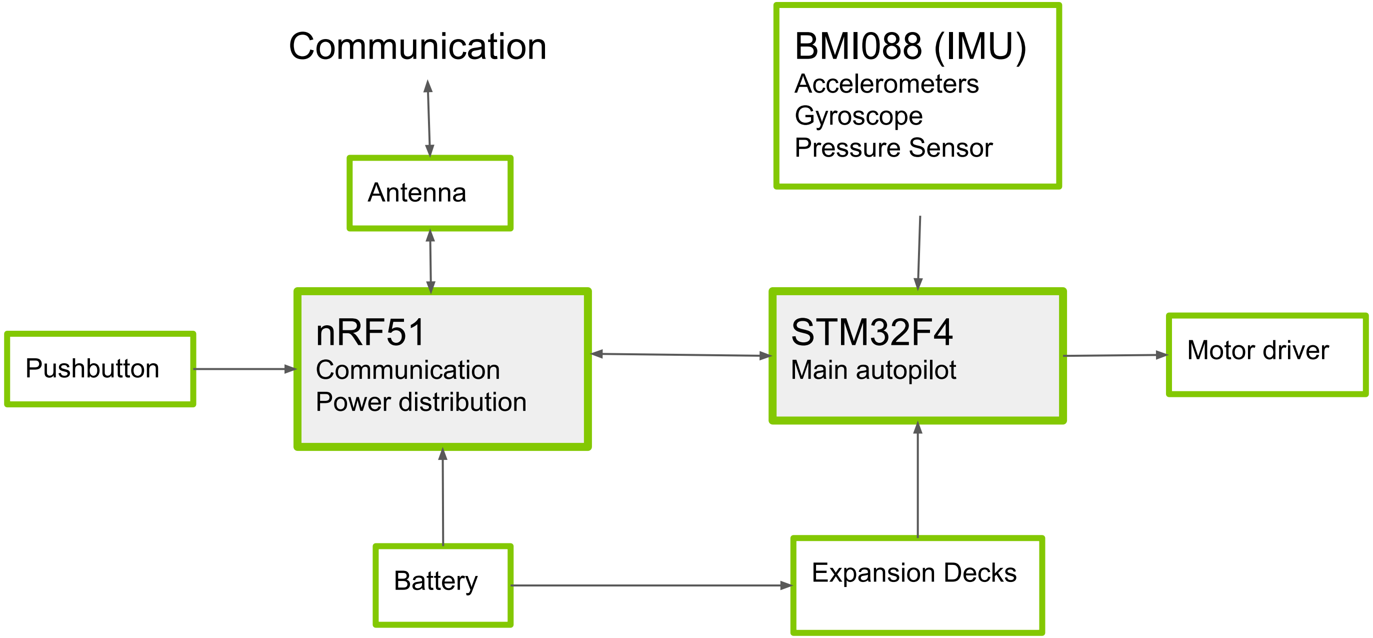


- STM32F4: Autopilot Microprocessor
- nRF51: Communication Microprocessor
- BMI088: Inertial Measurement Unit (IMU)



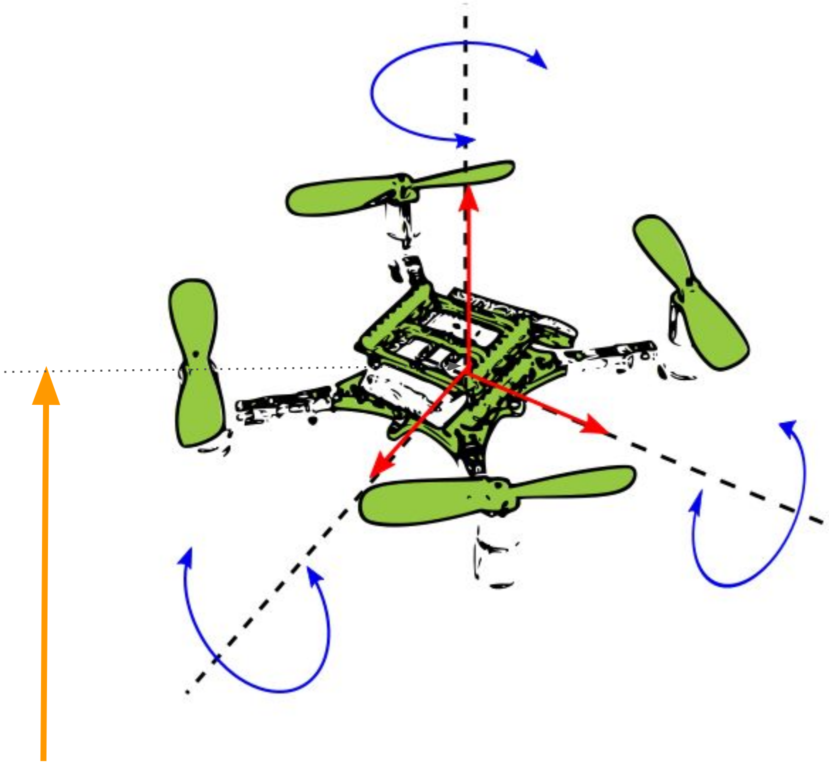


Hardware component connections



Inertial Measurement Unit (IMU)

- Accelerometers
- Gyroscope
- *Pressure Sensor*



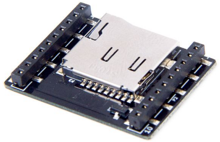
HANDS-ON

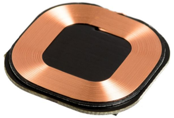
Show how to setup logging configuration

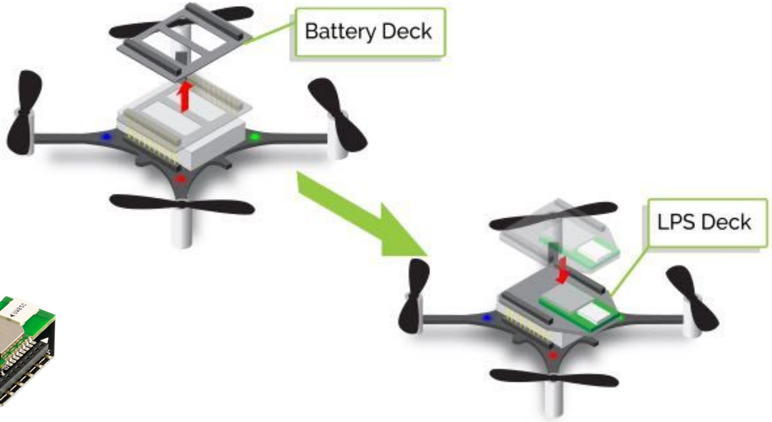
Plotting tab in CFclient to show raw IMU values

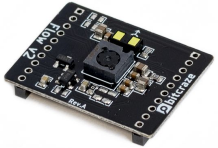
https://github.com/bitcraze/crazyflie-clients-python

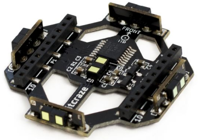
Expansion Decks

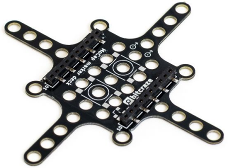






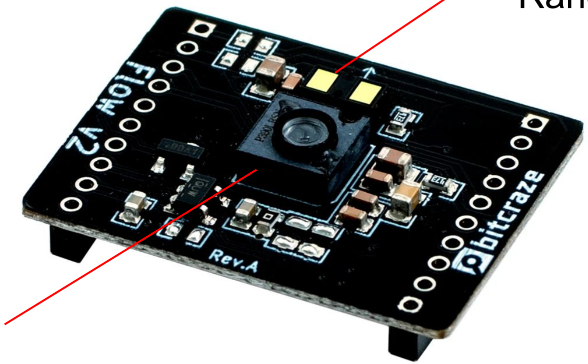




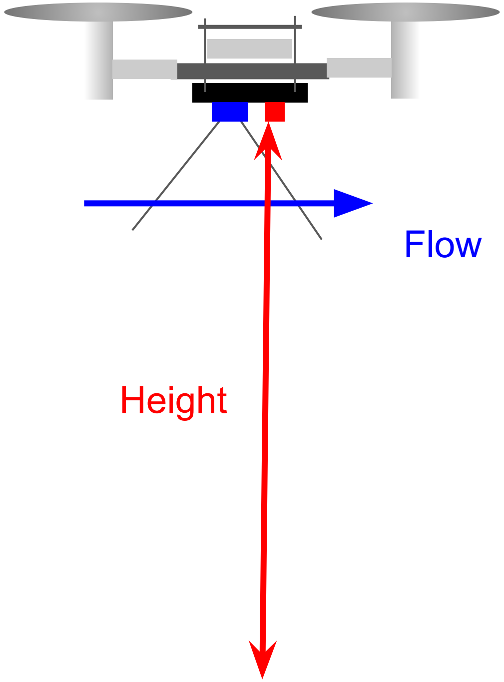


Flowdeck

VL53L1x
Range sensor



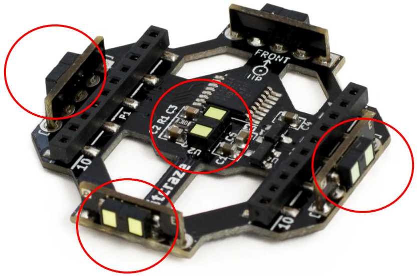
PMW3901
Optic Flow Sensor

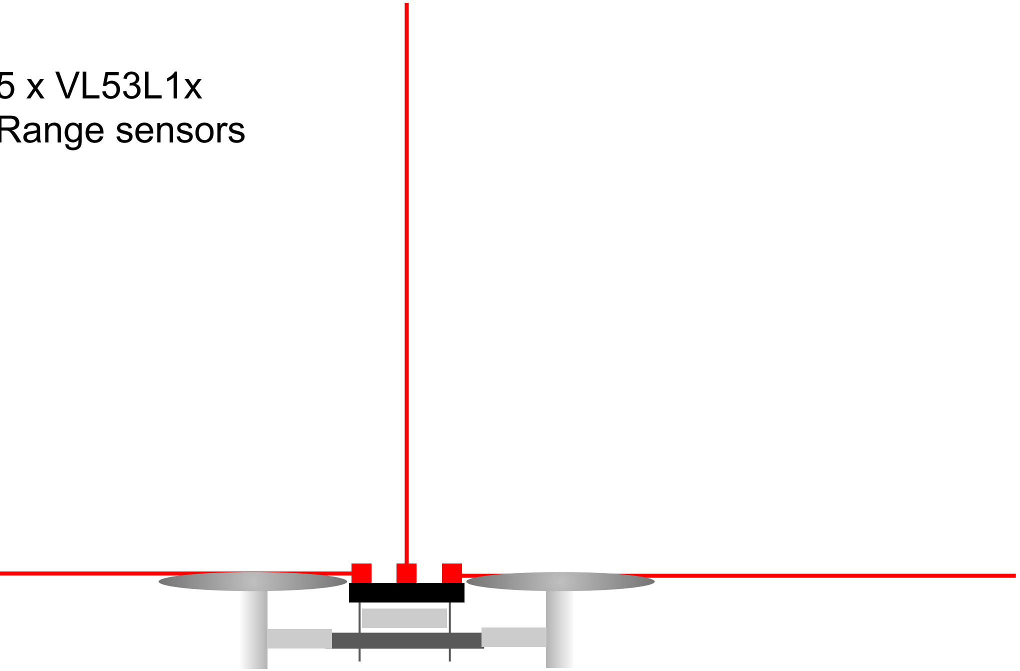


Relative vs global position!

Multiranger

5 x VL53L1x
Range sensors





HANDS-ON

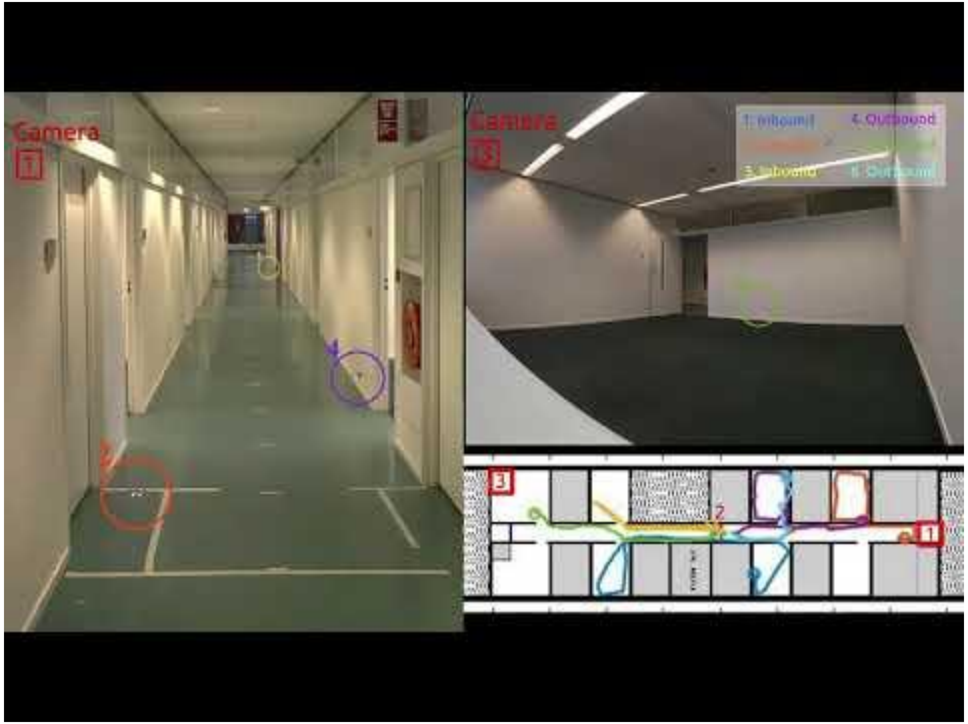
Introduction to console-tab

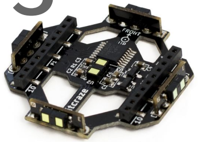
CFclient logging with flowdeck measurements

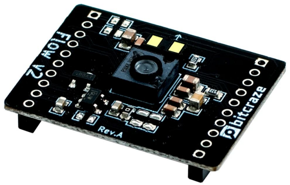
Also show multiranger measurements

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

Example with the Flowdeck + Multiranger



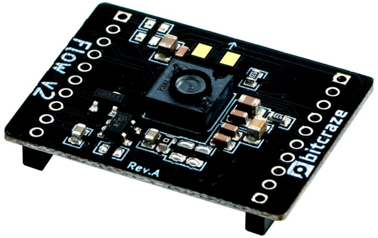


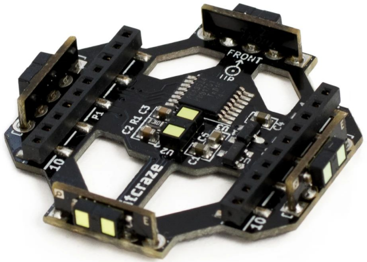


Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment (Science Robotics, 2019) K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen, https://youtu.be/jU4wsxwM1No

Recap of the last hour

- Crazyflie
- CFclient and logging
- Flowdeck + Multiranger

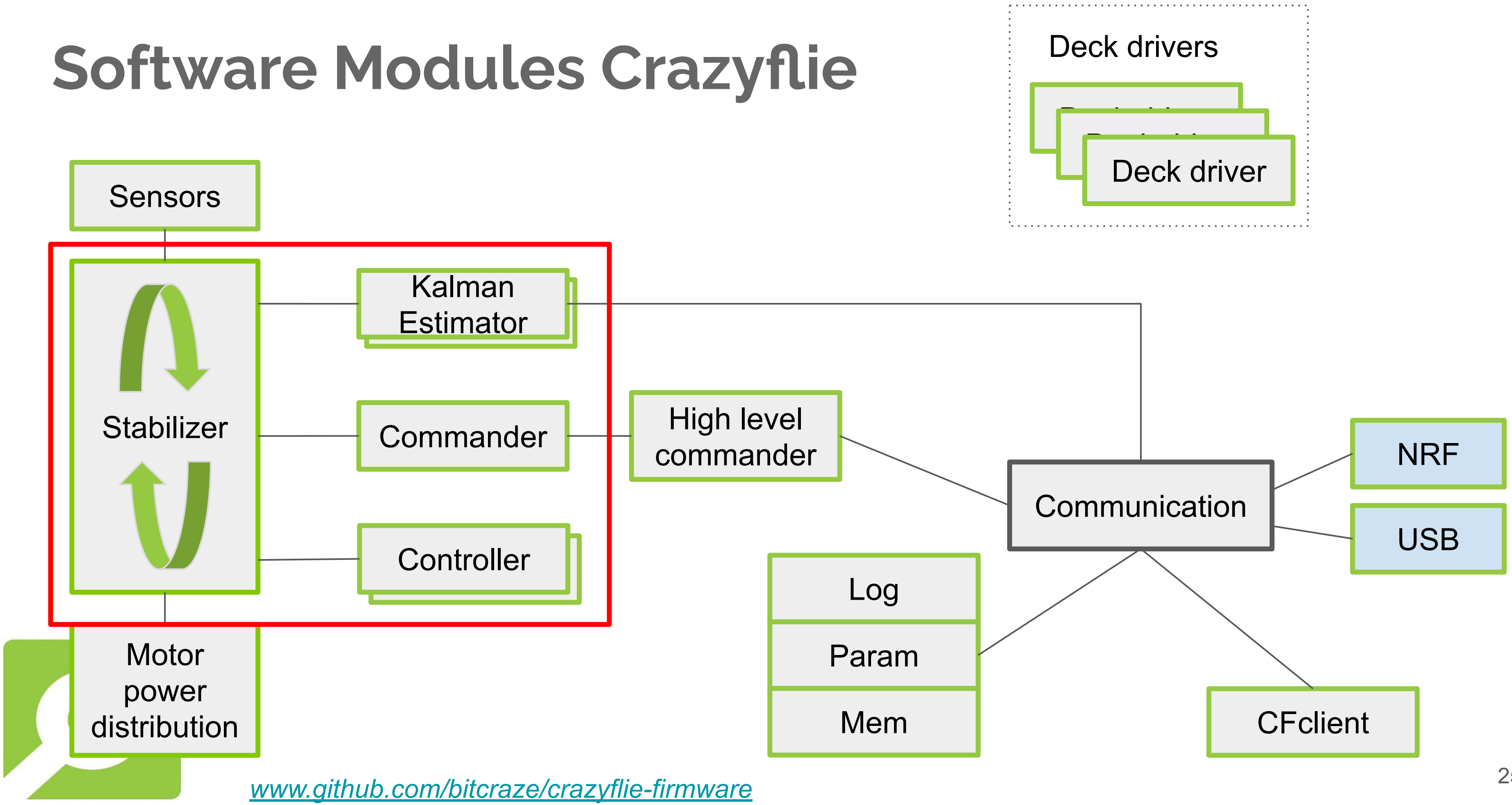




What do you need to fly?

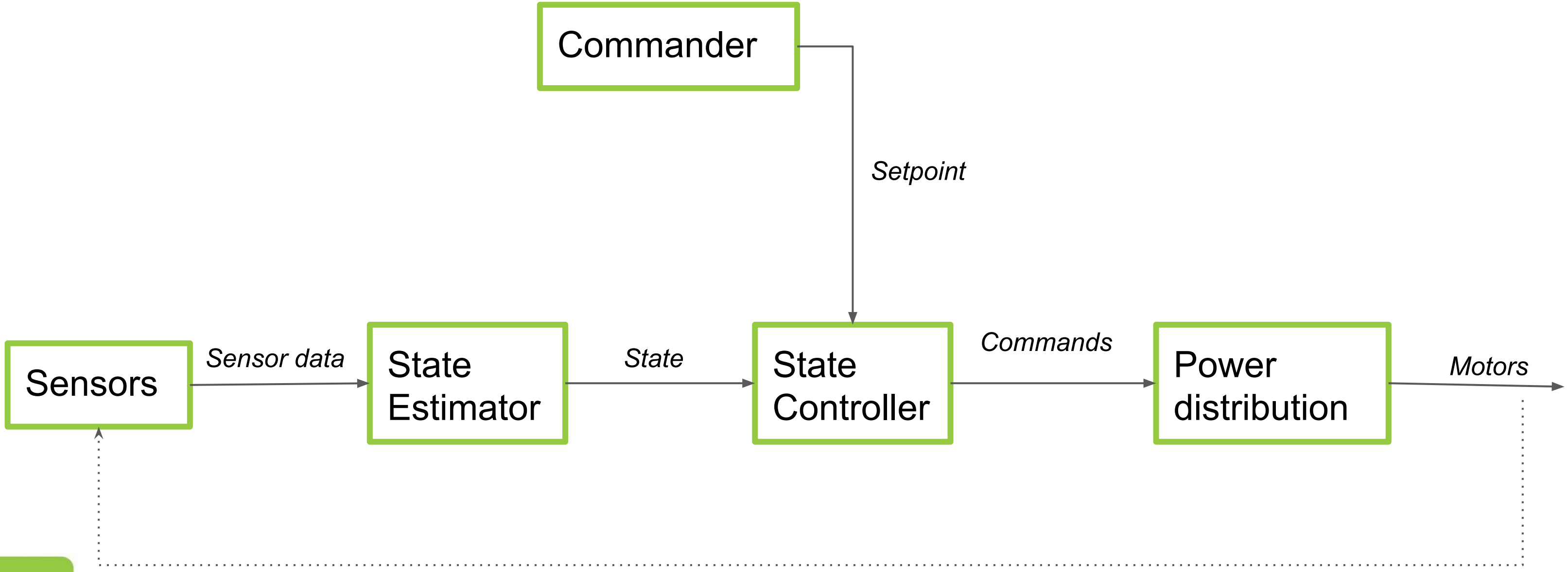
- Hardware (last hour)
- Software (firmware)

Software Modules Crazyflie



www.github.com/bitcraze/crazyflie-firmware

Flow from sensors to motors

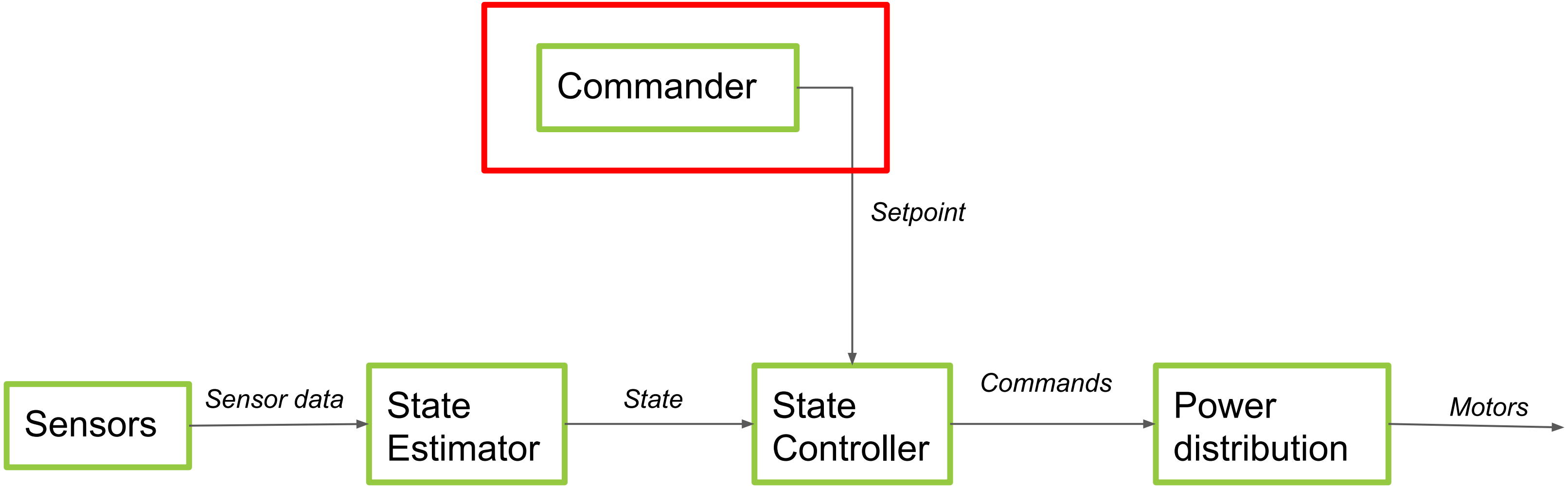


Documentation:
https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/

Hands-on

- Show the crazyflie flying
- CFclient show:
  - position estimation
  - Control commands
- Emphasis on setpoints

Flow from sensors to motors

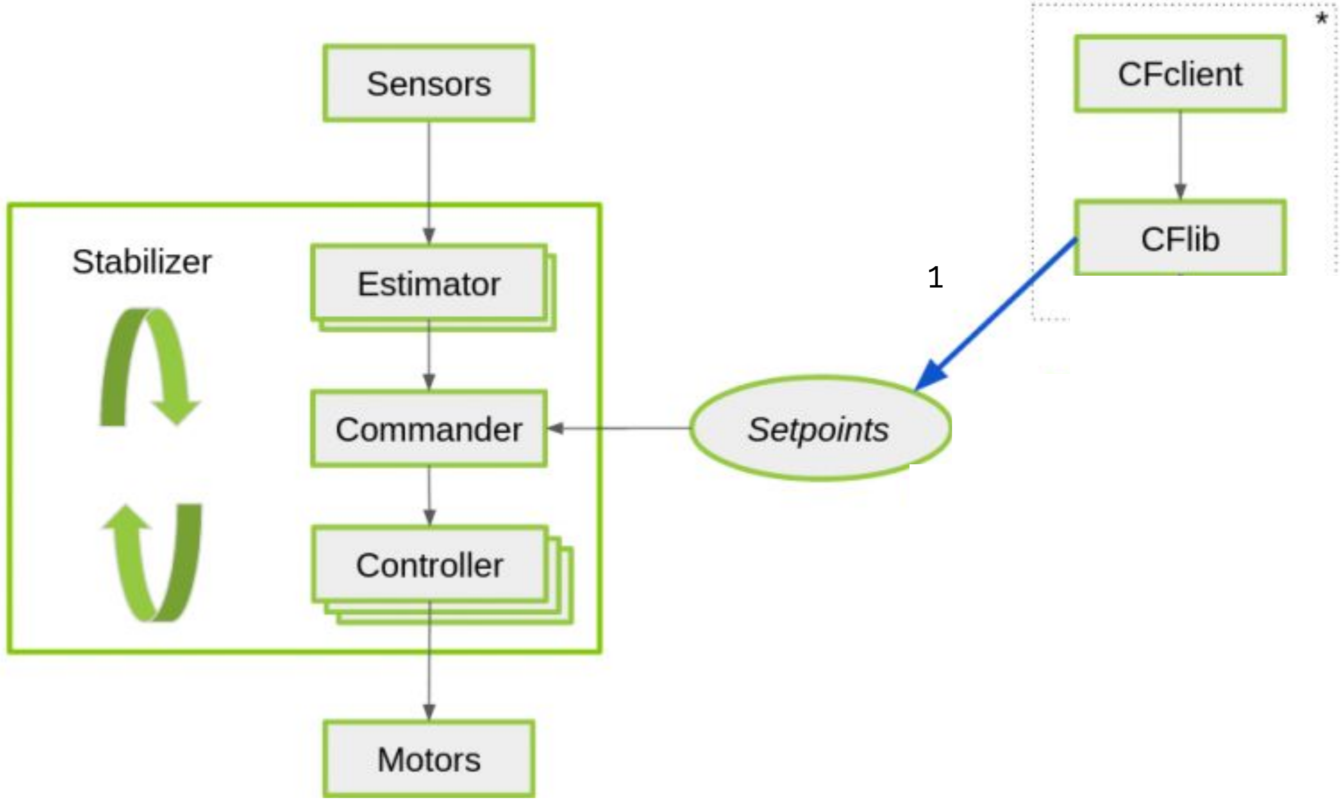


Documentation:
https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/

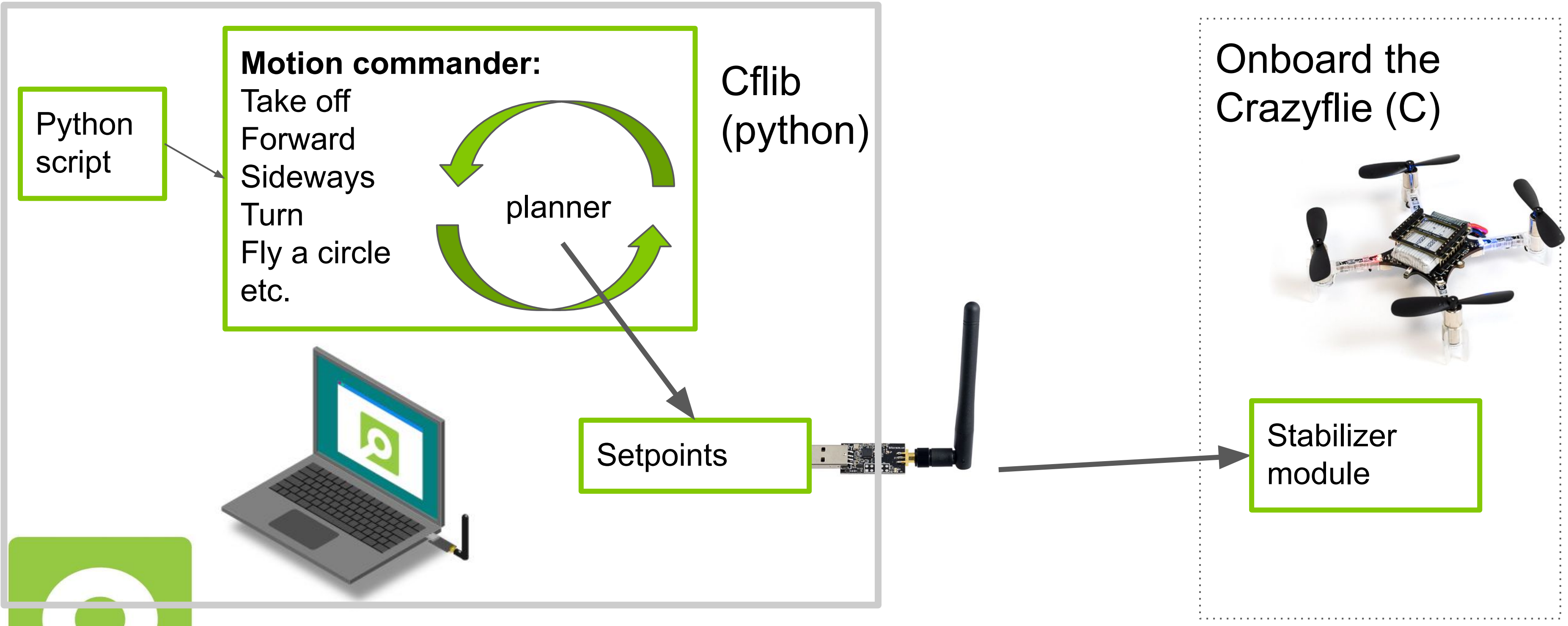
Type of Commanders

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

Setpoints through the CFlib



Motion Commander



Hands-on

Go through the Motion Commander Demo

Push Demo with crazyflie and multiranger

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

motion_commander_demo.py

```python
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()
```

Initializing

```python
    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)
```

Position control

```python
            # 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)
```

Velocity control

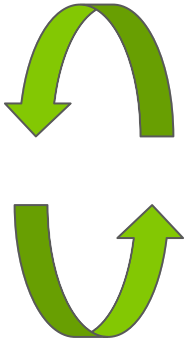
```python
            # We can do circles or parts of circles
            mc.circle_right(0.5, velocity=0.5, angle_degrees=180)

            # 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)
```

Non-blocking functions

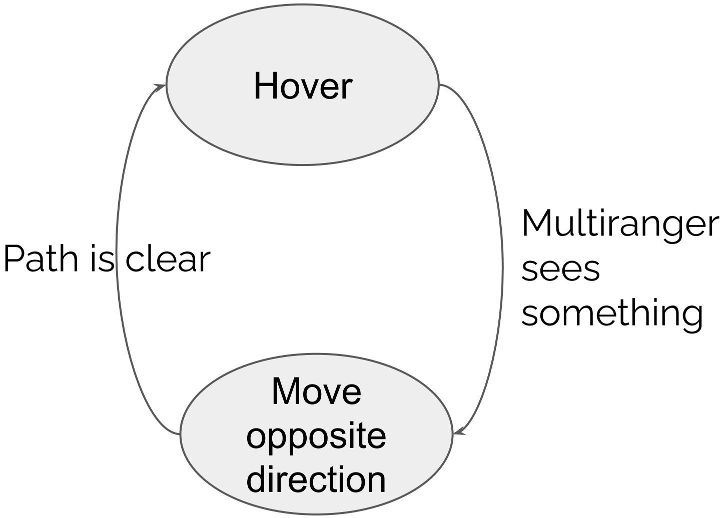
```python
            # And we can stop
            mc.stop()

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

multiranger_push.py

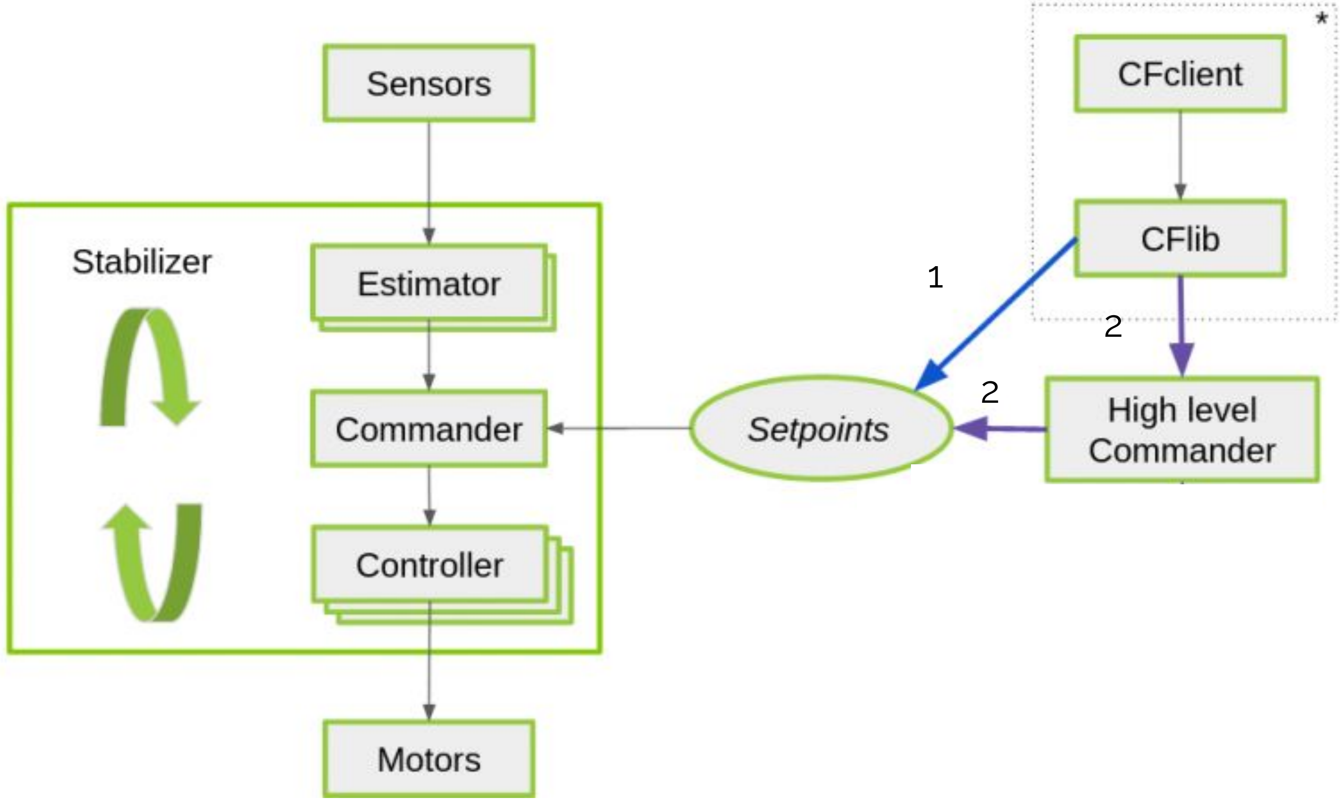
```python
if __name__ == '__main__':
    # Initialize the low-level drivers
    cflib.crtp.init_drivers()

    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

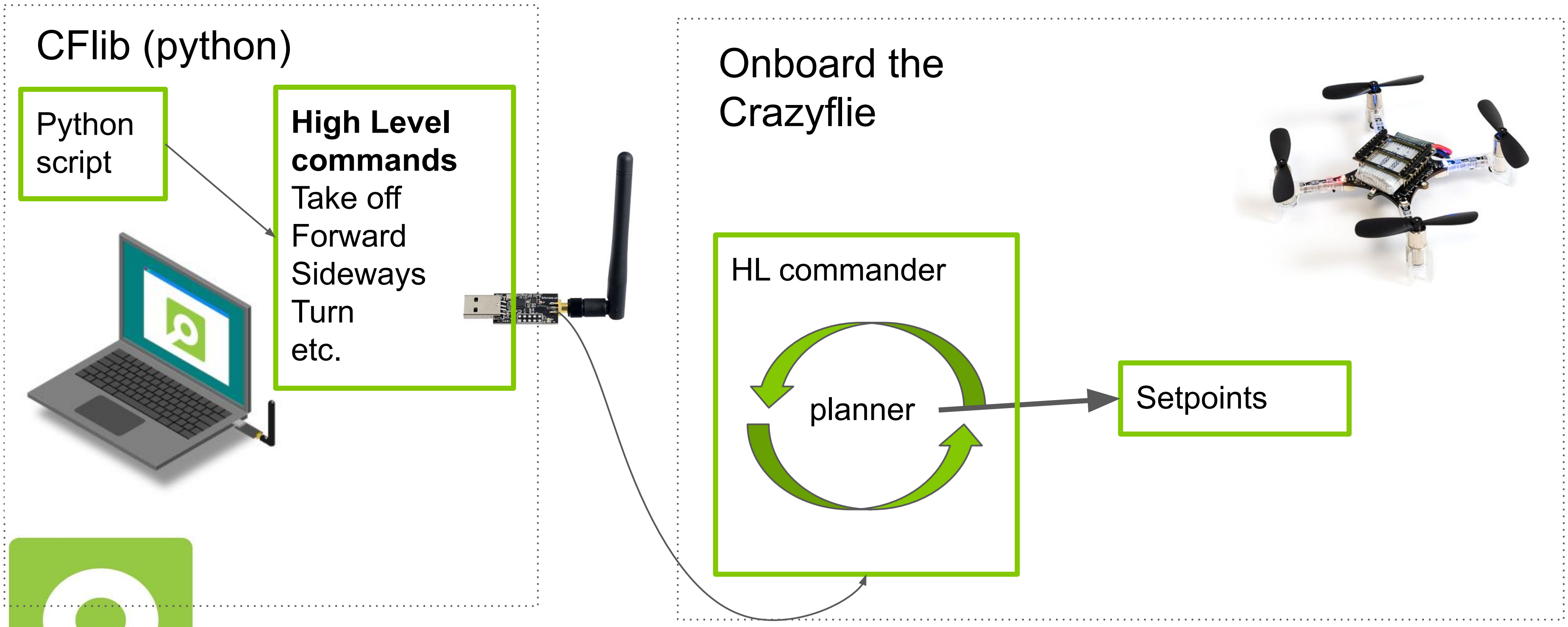


More autonomy onboard?

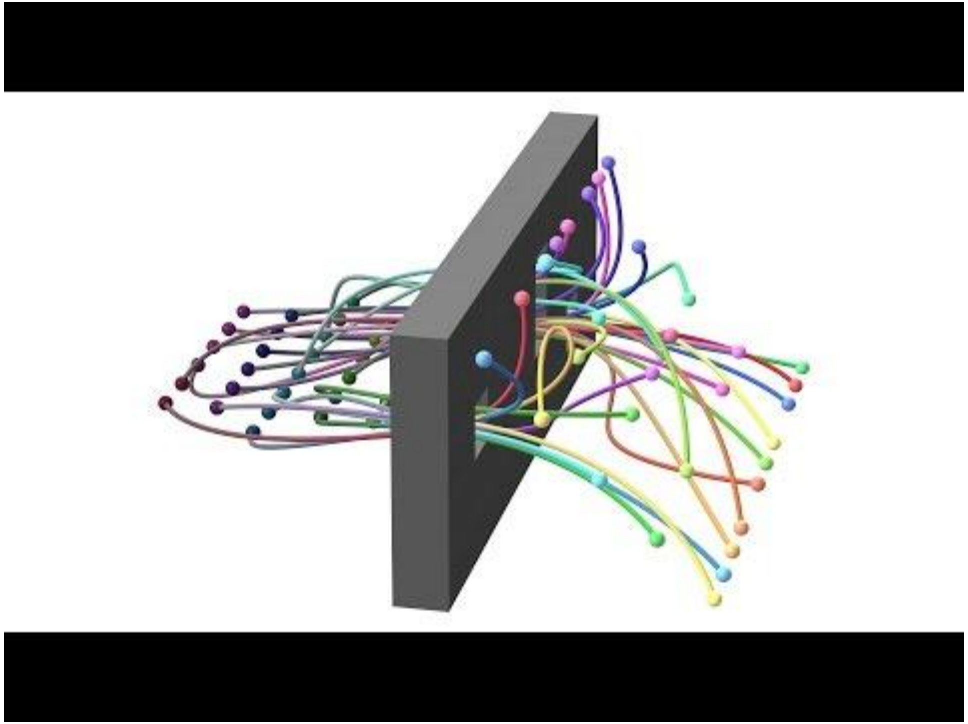


High level (HL) commander

Velocity commands not implemented- maybe not great for the relative positioning

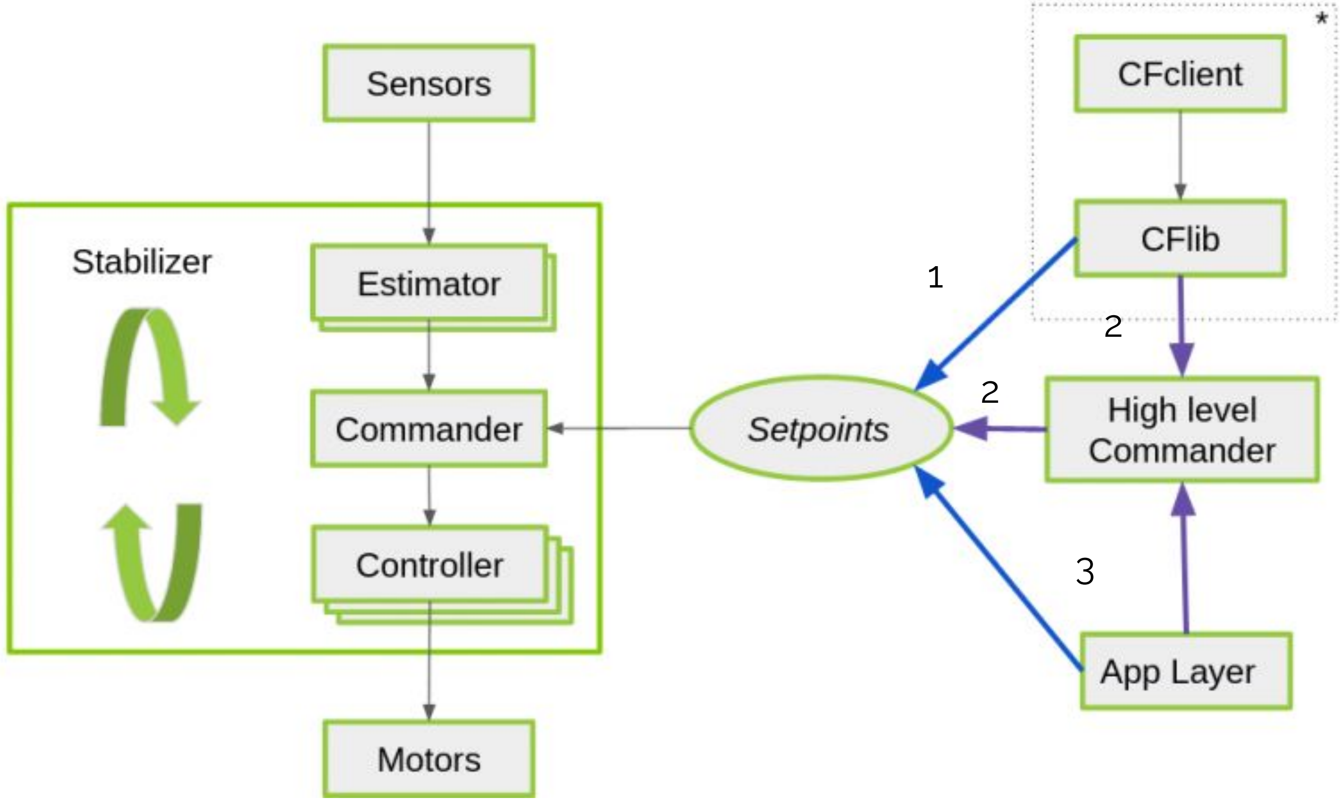


Example of the high level commander

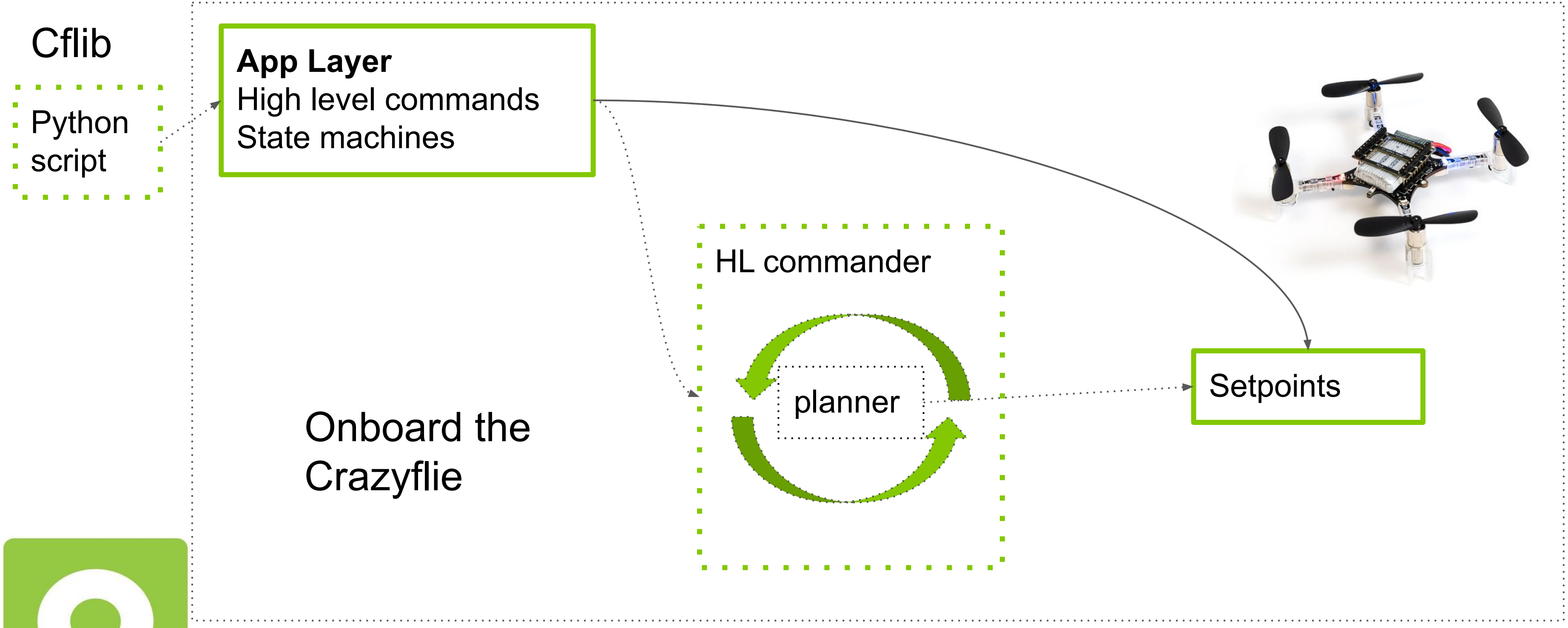


Preiss, James A., et al. "Downwash-aware trajectory planning for large quadrotor teams." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017. https://youtu.be/YnGZ-arUwgc

How about even more autonomy??



The App layer



The App layer

- User / research specified application
- Easier to maintain (seperate from firmware)
- More onboard autonomy without needing an PC
- Similar to library but then onboard

Hands-on

- Flash it
- Show code the app layer version of the Push demo
- Demo

App Layer: Demo can be found in [crazyflie-firmware/examples/demos]

demos/app_push_demo/

Makefile

```
# enable app support
APP=1
APP_STACKSIZE=300

VPATH += src/
PROJ_OBJ += push.o

CRAZYFLIE_BASE=../../..
include $(CRAZYFLIE_BASE)/Makefile
```

push.c

```
void appMain()
{
  static setpoint_t setpoint;

  vTaskDelay(M2T(3000));

  logVarId_t idUp = logGetVarId("range", "up");
  logVarId_t idLeft = logGetVarId("range", "left");
  logVarId_t idRight = logGetVarId("range", "right");
  logVarId_t idFront = logGetVarId("range", "front");
  logVarId_t idBack = logGetVarId("range", "back");

  paramVarId_t idPositioningDeck = paramGetVarId("deck", "bcFlow2");
  paramVarId_t idMultiranger = paramGetVarId("deck", "bcMultiranger");
```

Get Logs

Get parameters

```
while(1) {
  vTaskDelay(M2T(10));
  //DEBUG_PRINT(".");

  uint8_t positioningInit = paramGetUint(idPositioningDeck);
  uint8_t multirangerInit = paramGetUint(idMultiranger);

  uint16_t up = logGetUint(idUp);

  if (state == unlocked) {
    uint16_t left = logGetUint(idLeft);
    uint16_t right = logGetUint(idRight);
    uint16_t front = logGetUint(idFront);
    uint16_t back = logGetUint(idBack);

    uint16_t left_o = radius - MIN(left, radius);
```

.
.
.

Send Setpoints

```
    if (1) {
      setHoverSetpoint(&setpoint, velFront, velSide, height, 0);
      commanderSetSetpoint(&setpoint, 3);
    }

    if (height < 0.1f) {
      state = stopping;
      DEBUG_PRINT("X\n");
    }
```

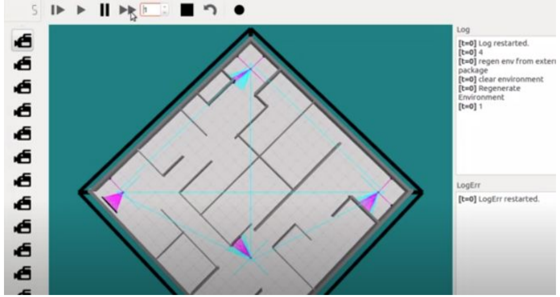
Motion Commander vs App Layer

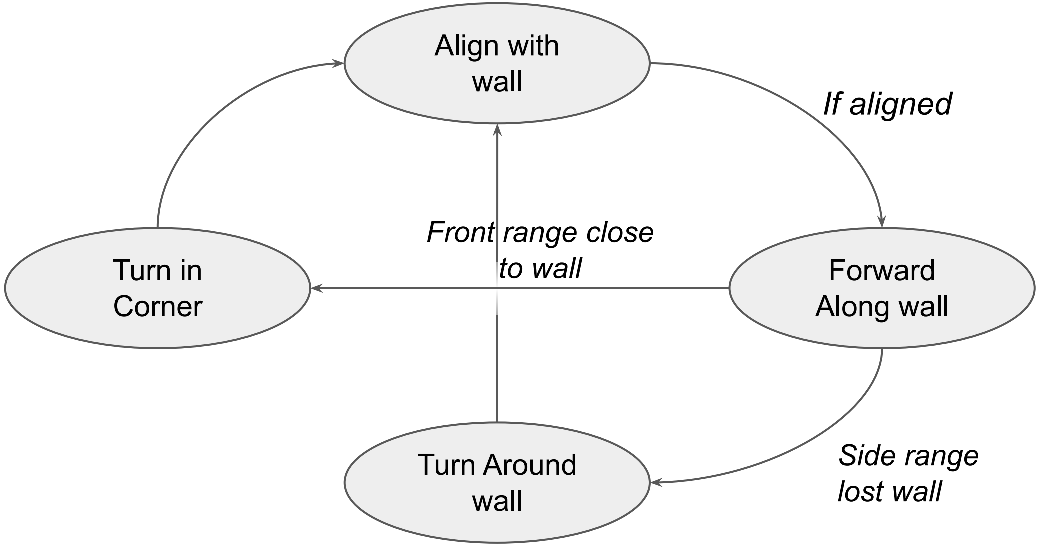
- Python
- Computer has the state machine
- Need a crazyradio for communication
- Small delay measurements-commands
- Good for trying out

- C
- Crazyflie contains the state machine
- Don't need a crazyradio or a computer
- Very little delay measurements -> commands
- Good for extra credit ;)

Case Study: Wall following

- Most important element of SGBA*
- Initially for survey**



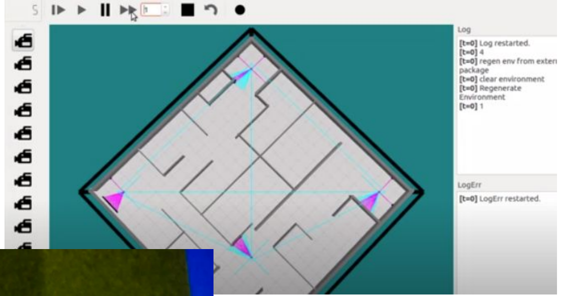


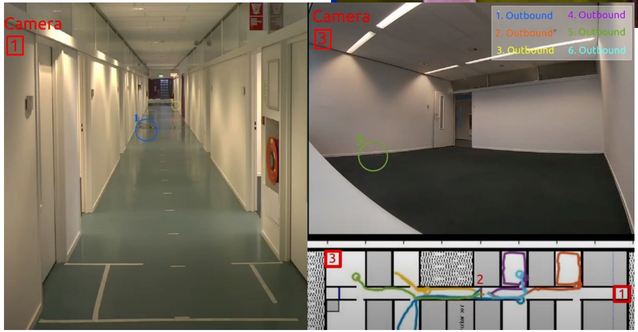
* Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment (Science Robotics) K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen,

** 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.

Case Study: Wall following

- Most important element of SGBA*
- Initially for survey**
- Steps:
  - 1- Python + ArGos**
  - 2- Python + Gazebo
  - 3- Python CFlib
  - 4- C + Gazebo
  - 5- C + On the drone





* 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.

Hands-on

- Show wall following app layer
- Flash and upload wall following code
- Show wall following

App Layer: Demo can be found in crazyflie-firmware/examples/demos

Python version: crazyflie-lib-python/examples/demos/ [wall_following_demo branch]

Recap

- Stabilizer module
- Commanders
- Different levels of autonomy

Thank you for listening!

Contact

Website: www.bitcraze.io

Forum: forum.bitcraze.io

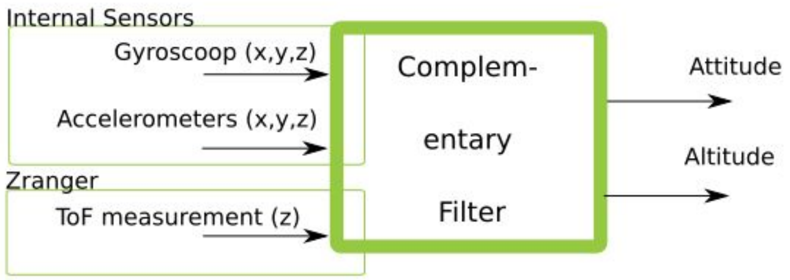
Email: contact@bitcraze.io

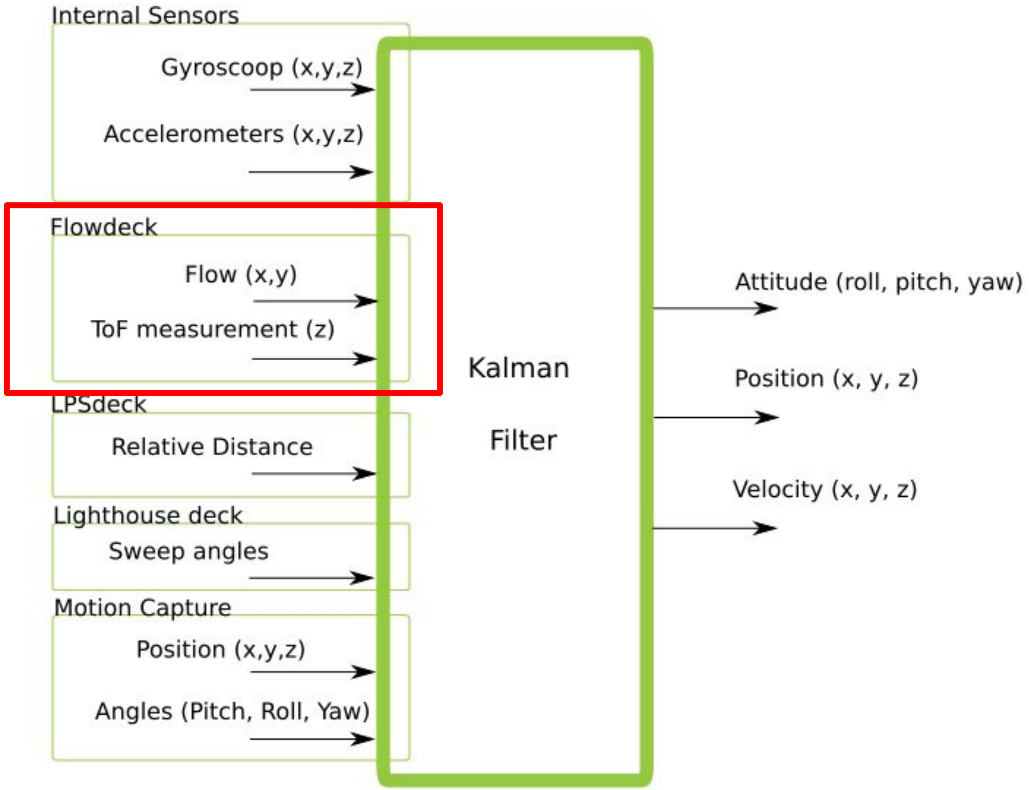
kimberly@bitcraze.io

Slides 2020

State estimation

- Complementary Filter
- Extended Kalman Filter





Extended Kalman Filter

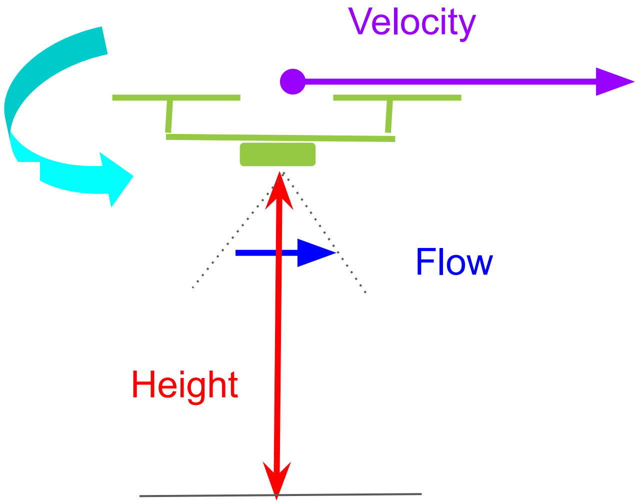
- Originally implemented by ETH Zurich*
- Quadrotor Motion Model*
- Measurement Models**
  - UWB lps system
  - Lighthouse system
  - Flowdeck

*Mueller, Mark W., Michael Hamer, and Raffaello D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation." *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

*Mueller, Mark W., Markus Hehn, and Raffaello D'Andrea. "Covariance correction step for kalman filtering with an attitude." *Journal of Guidance, Control, and Dynamics* 40.9 (2016): 2301-2306.

***crazyflie-firmware/src/modules/src/estimator/estimator_kalman, .../kalman_core.c*

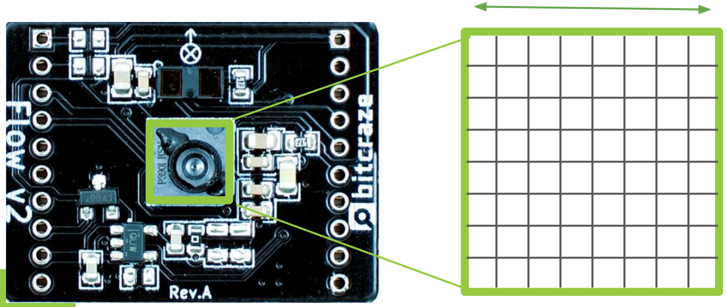
Velocity calculation Flowdeck

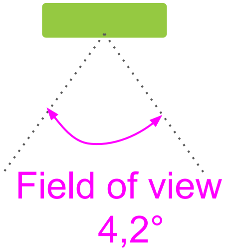


$$\boxed{\dot{x}} = \frac{\boxed{h} \cdot \boxed{\theta_{px}} \cdot \boxed{\Delta n_x}}{\Delta t \cdot \boxed{N_x}}.$$

Sample time

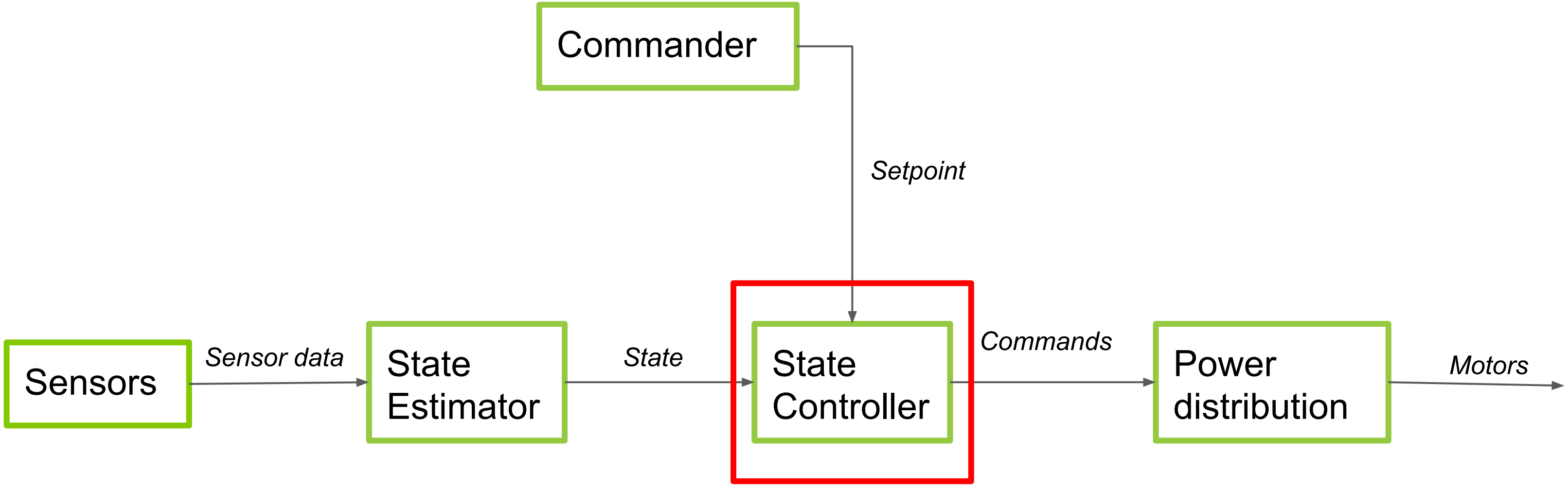
Pixel width (30 px)





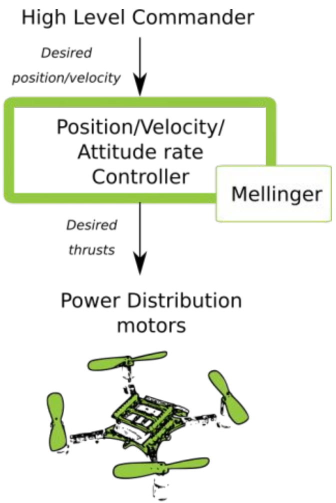
Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation, M. Greiff, Master's thesis, Lund University, 2017

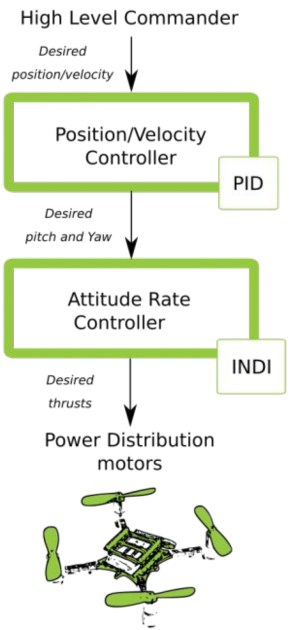
Flow from sensors to motors

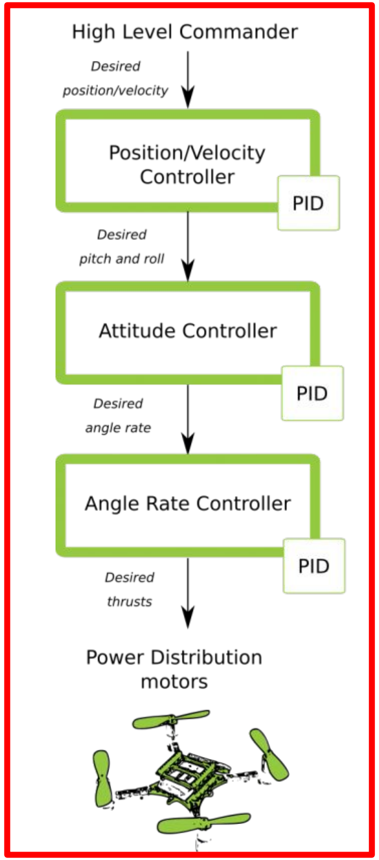


Controllers

- Levels of control
  - Position/velocity
  - Attitude
  - Attitude rate
- Types
  - PID
  - Incremental nonlinear dynamic inversion (INDI) *
  - Mellinger **



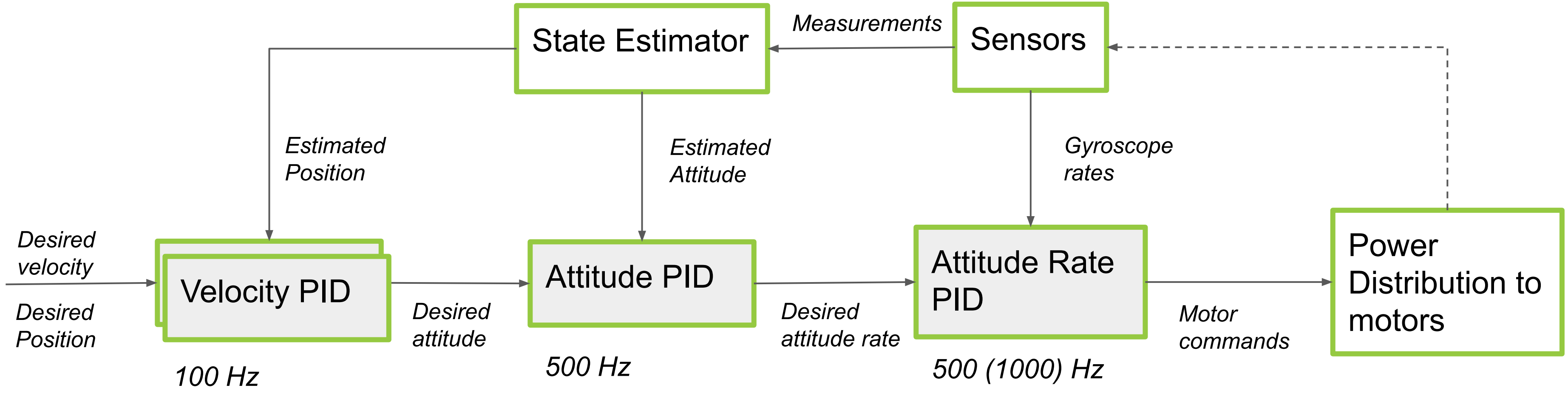




* E. de Smeur et al. "Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles." *Journal of Guidance, Control, and Dynamics* 38.12 (2016): 450-461.
* *Implemented by: E.Smeur and A.L.O. Paraense: crazyflie-firmware/src/modules/src/controller_indi.c (2019)*

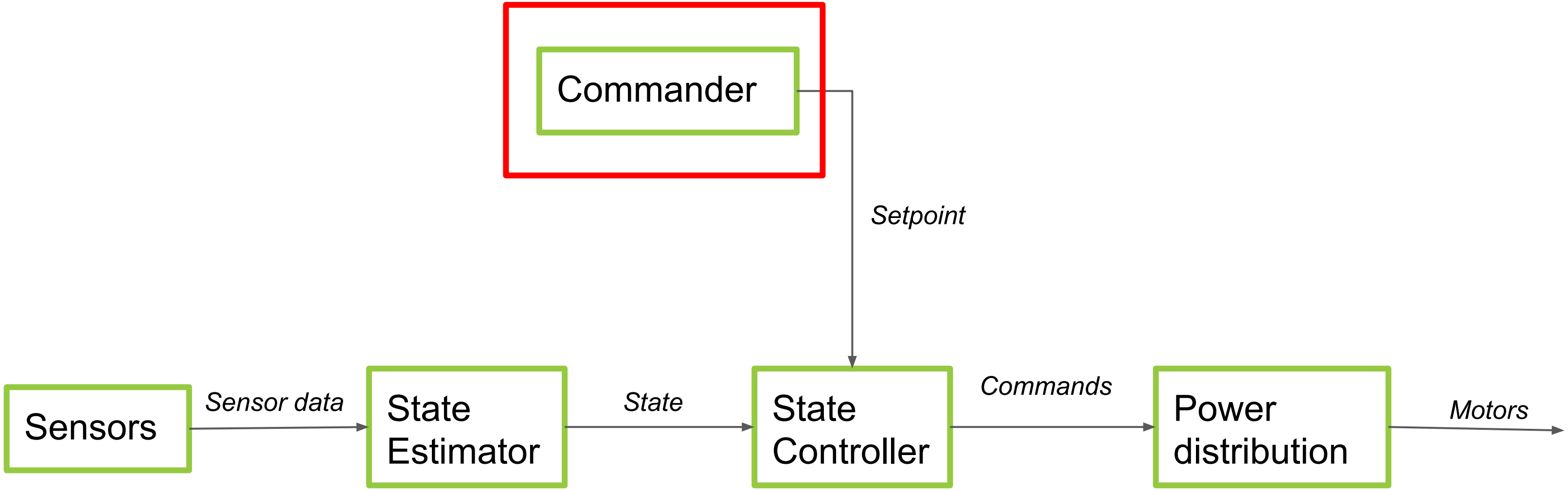
** Daniel Mellinger, Vijay Kumar:Minimum snap trajectory generation and control for quadrotors. IEEE International Conference on Robotics and Automation (ICRA), 2011.
** *Implemented W. Hönig & J. A. Preiss: crazyflie-firmware/src/modules/src/controller_mellinger.c*

Cascaded PID control



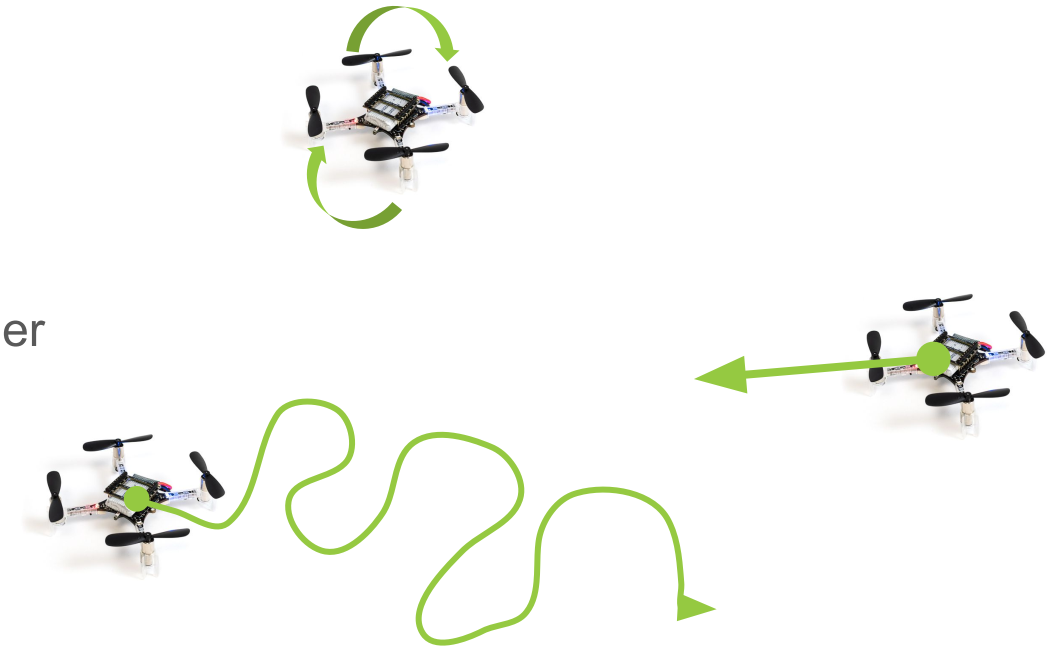
crazyflie-firmware/src/modules/src/controller_pid.c
crazyflie-firmware/src/modules/src/attitude_pid_controller.c
crazyflie-firmware/src/modules/src/position_controller_pid.c

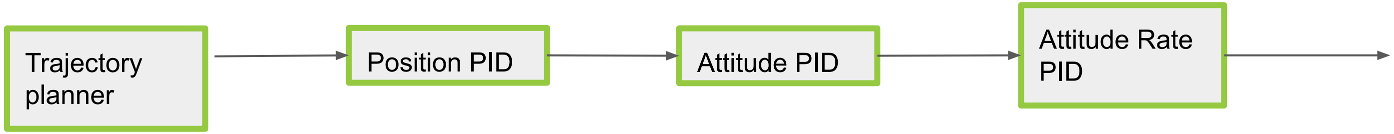
Flow from sensors to motors



Commanders

- Attitude commander
- Position/velocity commander
- High Level commander





Motion Commander vs High Level commander

Motion Commander:

- \+ More control from computer
- \+ Easier to see what's going on
- \- Lot of communication
- \- Not great for Swarms

Easy scripts for one crazyflie

If script ends or fails, or connection is lost, the crazyflie will land

High Level commander:

- \+ More autonomy on Crazyflie
- \+ Less communication necessary
- \- Less visibility of what is going on

More complicated scripts for one or multiple crazyflies

If script ends or fails, or connection is lost, the crazyflie will keep on hovering.