Spiraling Swarm Demo

Tutorial Lecture
4th of November, 18:00 CET

bitcraze

Technicalities

- Discord-only participants
  - Voice is disabled on the #tutorial_video
  - Use the #tutorial_chat to ask questions
- Mozilla hubs
  - The link to the *Mozilla Hubs room* can be found in #tutorial_chat in *Discord*
  - You need to sign in with your email address that you used for *Discord*
  - Make sure that you disconnect from #tutorial_video or mute it since it will cause an **echo**
  - Please press **mute** during the sessions
  - Use the chat box in *Mozilla hubs* to ask questions
  - During break & socializing you can **unmute** yourself
- **The full tutorial will be recorded and slides will become available.**

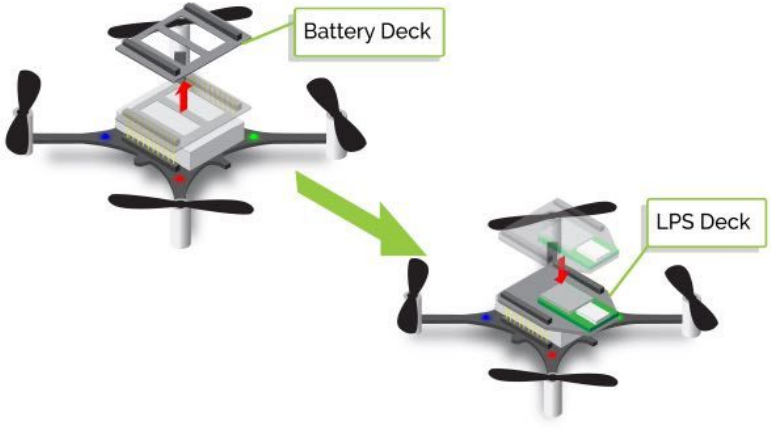
Introduction to Bitcraze AB

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

Crazyflie

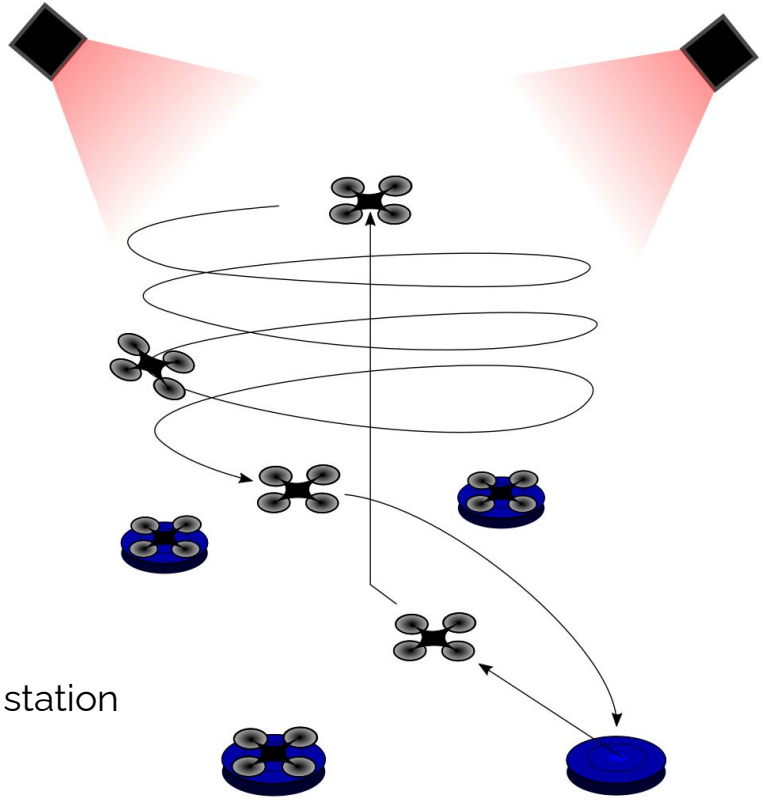
- Quadrotor
- Used by:
  - Hobbyist
  - Researchers
  - Educators in Aerial Robotics
  - Shows
- Open Source firmware
- Modular Design

Modular Design



Our demo

Positioning system

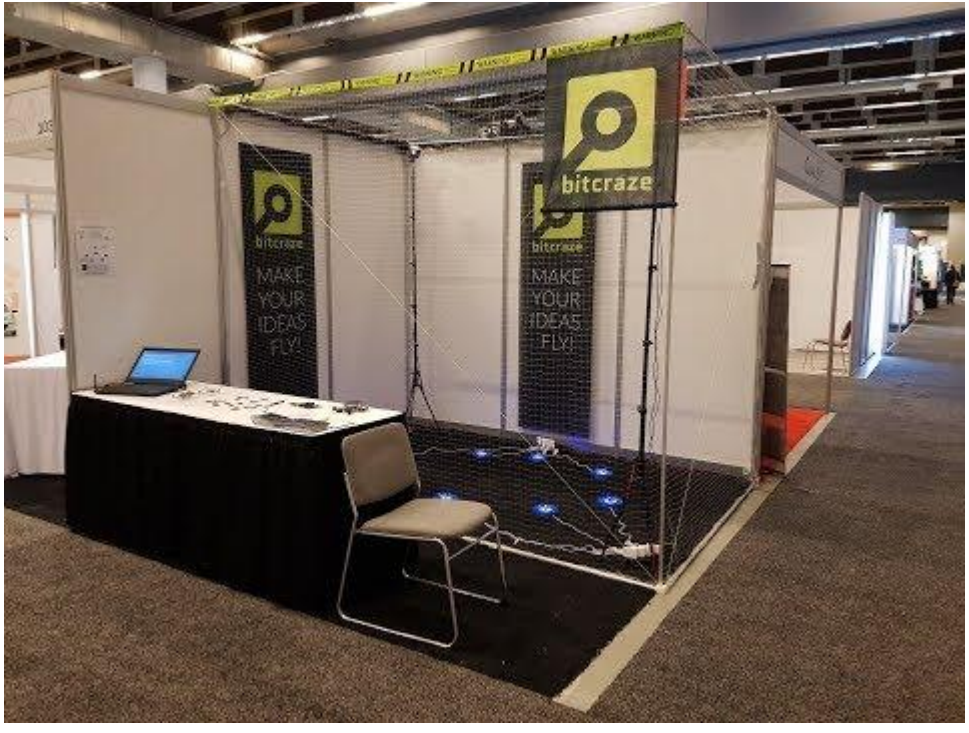


Crazyflies

Charging station

Control tower

Video

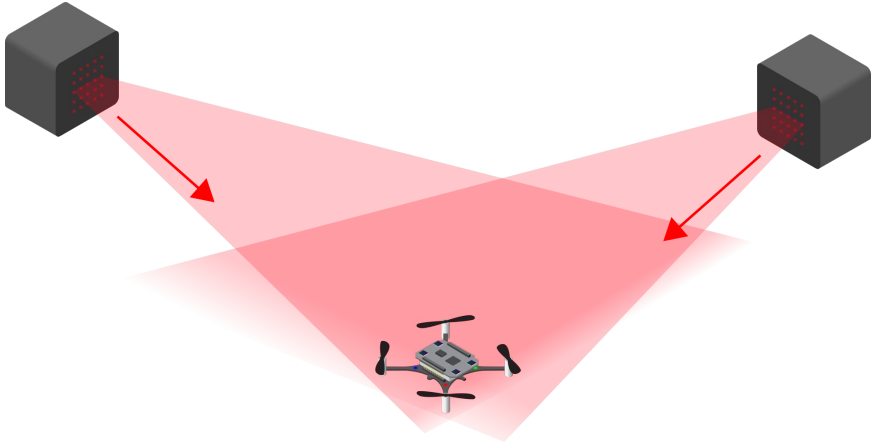

bitcraze
MAKE
YOUR
IDEAS
FLY!
bitcraze
MAKE
YOUR
IDEAS
FLY!
bitcraze

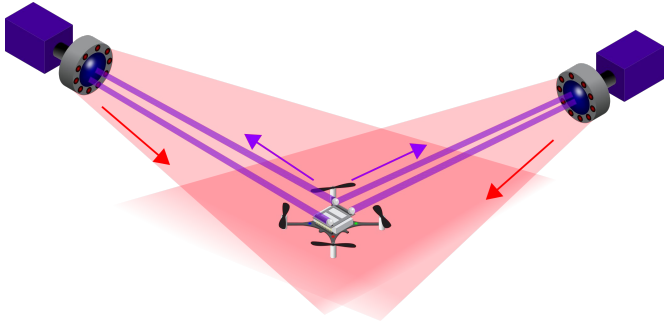
Topics of today

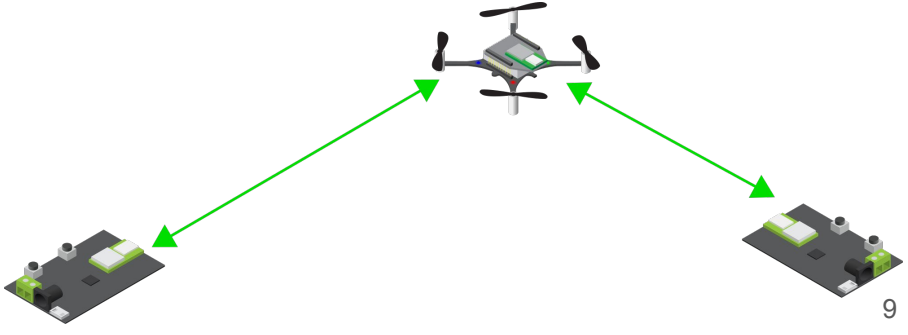
- 18:00
  - 35 min: Positioning - Lighthouse V2
  - 10 min: Questions
- 18:45: break
- 19:00:
  - 35 min: Communication - Swarm Autonomy
  - 10 min: Questions
- 19:45
  - Socializing :)

Positioning

- Crazyflie need to know where it is
- Types of positioning:
  - Relative positioning
  - Absolute positioning
    - Onboard estimate
    - Offboard estimate







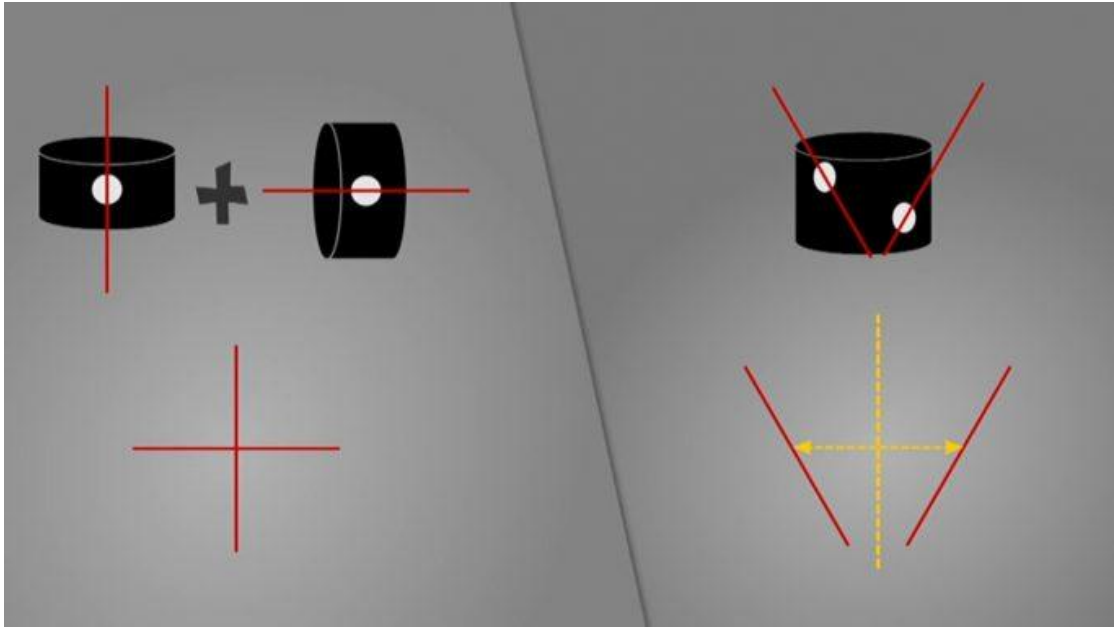
Lighthouse Positioning

- Valve Corporation - Basestations
- Virtual Reality
- Early access
- Mm precision

https://store.steampowered.com/app/1059570/Valve_Index_Base_Station/

Mechanics Basestations

V1 V2

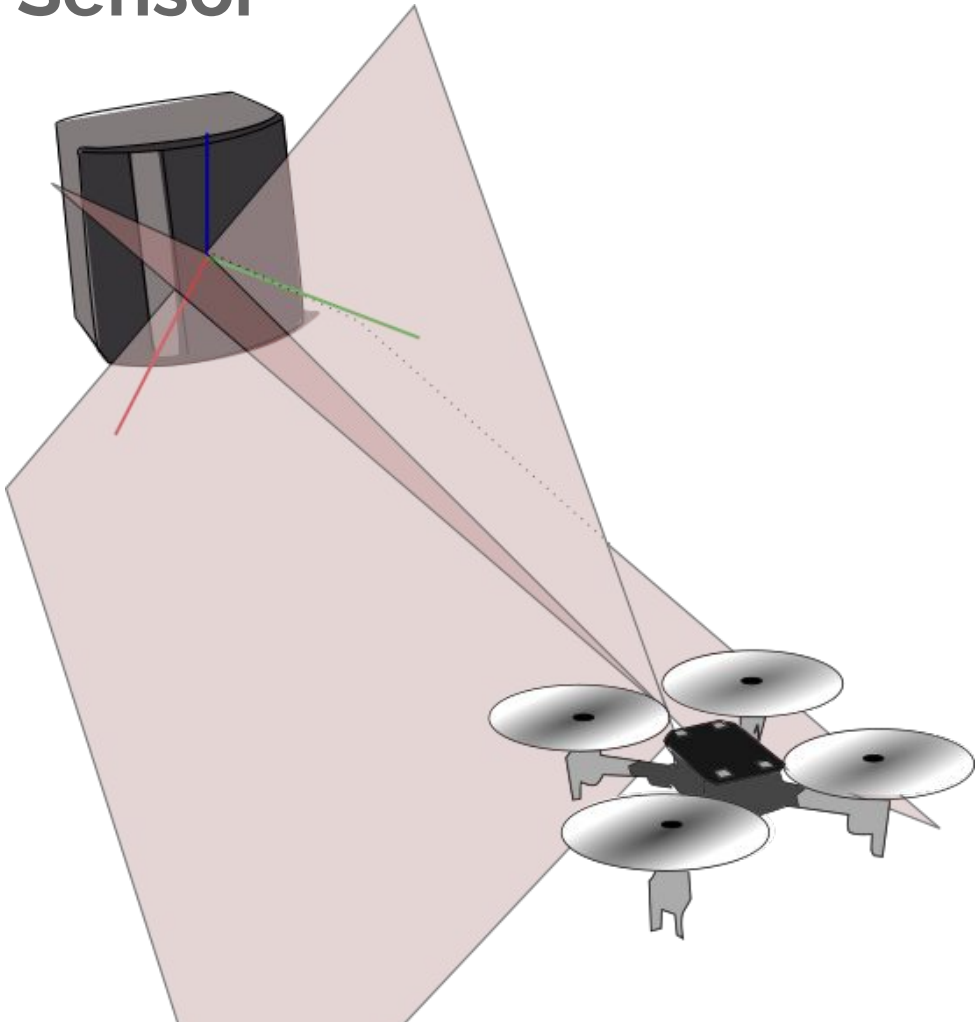


Lightsweeps

V1

V2

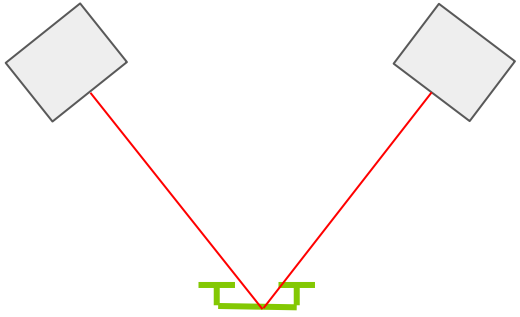
Lightsweeps to Sensor



Hands-on: Setting up the basestations

- Setting up the basestations
- Powering them
- Put a crazyflie in the middle
- Show the angles in the cfclient

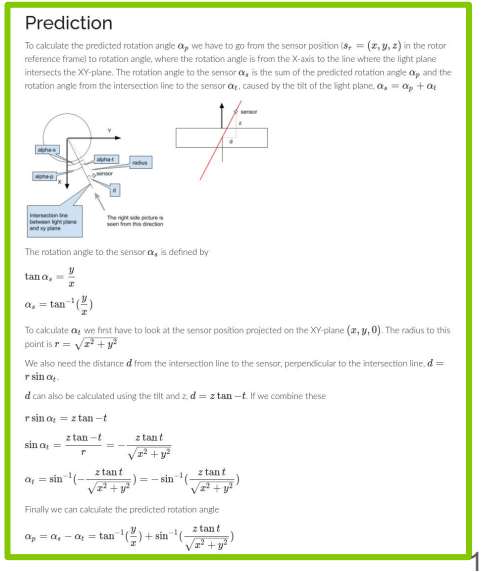
Lighthouse to position



Two methods:

- Crossing beams
  - Only with two basestations
- Raw sweeps
  - Kalman filter measurement model
  - Basestation are decoupled

www.bitcraze.io : Documentation -> system overview -> Positioning Systems -> Lighthouse positioning system



Prediction

To calculate the predicted rotation angle α_p we have to go from the sensor position ($s_r = (x, y, z)$ in the rotor reference frame) to rotation angle, where the rotation angle is from the X-axis to the line where the light plane intersects the XY-plane. The rotation angle to the sensor α_s is the sum of the predicted rotation angle α_p and the rotation angle from the intersection line to the sensor α_t, caused by the tilt of the light plane, $\alpha_s = \alpha_p + \alpha_t$

The rotation angle to the sensor α_s is defined by

$$\tan \alpha_s = \frac{y}{x}$$

$$\alpha_s = \tan^{-1}(\frac{y}{x})$$

To calculate α_t we first have to look at the sensor position projected on the XY-plane $(x, y, 0)$. The radius to this point is $r = \sqrt{x^2 + y^2}$

We also need the distance d from the intersection line to the sensor, perpendicular to the intersection line, $d = r \sin \alpha_t$.

d can also be calculated using the tilt and z, $d = z \tan -t$. If we combine these

$$r \sin \alpha_t = z \tan -t$$

$$\sin \alpha_t = \frac{z \tan -t}{r} = -\frac{z \tan t}{\sqrt{x^2 + y^2}}$$

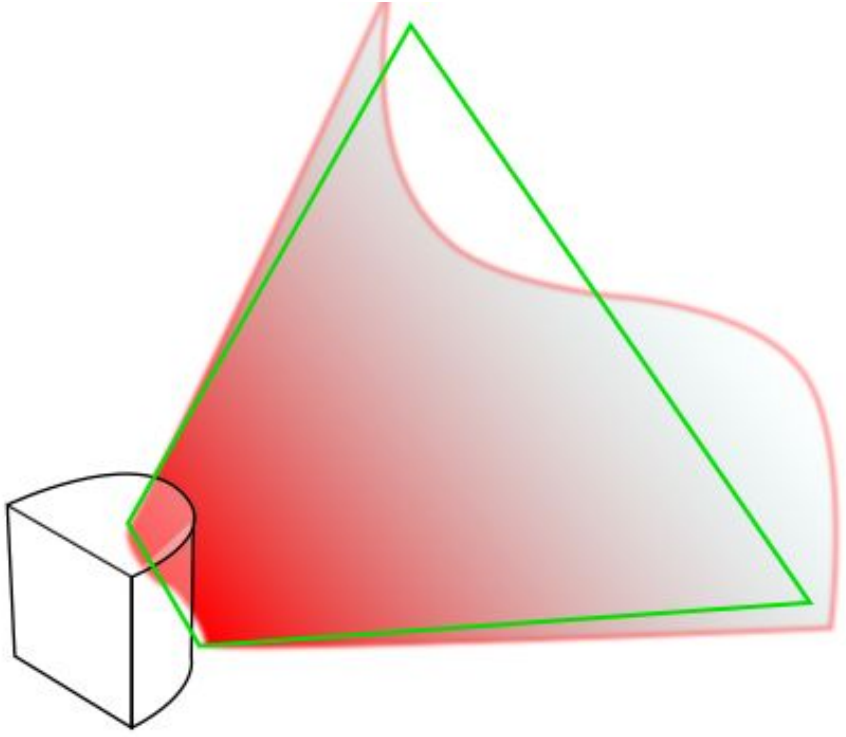
$$\alpha_t = \sin^{-1}(-\frac{z \tan t}{\sqrt{x^2 + y^2}}) = -\sin^{-1}(\frac{z \tan t}{\sqrt{x^2 + y^2}})$$

Finally we can calculate the predicted rotation angle

$$\alpha_p = \alpha_s - \alpha_t = \tan^{-1}(\frac{y}{x}) + \sin^{-1}(\frac{z \tan t}{\sqrt{x^2 + y^2}})$$

Calibration Lighthouse

- Light sweeps are not perfect
- Reflections of glass
- Motor calibration
- Curvation



Hands-on: Getting calibration data V2

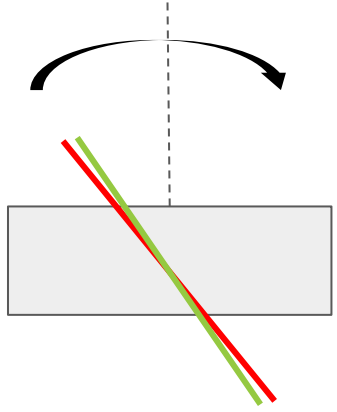
- Show the CFclient with console
  - Takes about 1 minute though...
- Alternative way with basestation and microusb
  - Find location of device with **dmesg**
  - **python3 get_lh2_calib_data.py --dev /dev/ttyACM0**
- How to change mode permanently
  - **picocom /dev/ACM0**
  - See which mode its in **mode**
  - Set new mode **mode 2**
  - Save parameters to make permanent **param save**

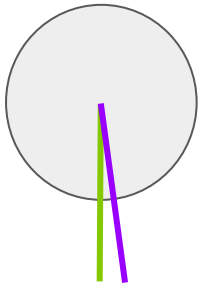
This is only necessary for Basestation V2!

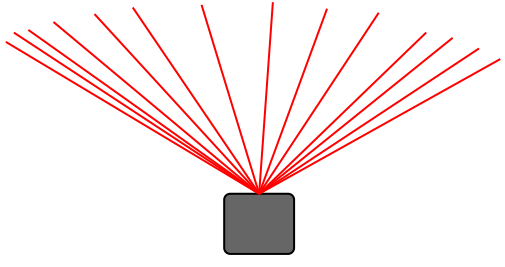
Basestation V1 receives the calibration data so quickly that you don't need to hardcode it at all!

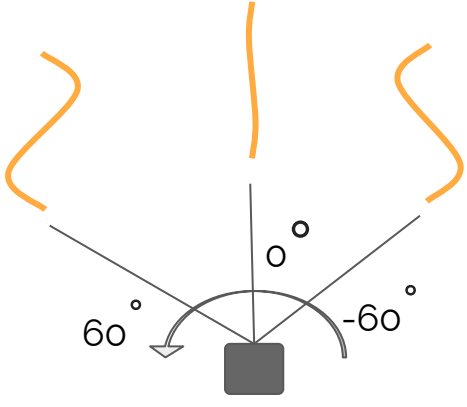
Calibration Values

- For each lightplane
  - Motor calibration
    - Tilt (error of lightplane tilt)
    - Phase (error rotating drum)
  - Distortion
    - Gibphase / -mag (Compression along the sweep)
    - Curve (Curvature of lightplane)
    - Ogeephase/ -mag (only V2) (Shape of lightplane)









Top view

All speculations !!

Implementation calibration values

- Libsurvive (survive_reproject_gen2.c)
- Ours (lighthouse_calibration.c):
  - V1

```
const float ax = atan2f(y, x);
const float ay = atan2f(z, x);
const float r = arm_sqrt(x * x + y * y);
const float compTilt = asinf(clip1(z * tanf(calib->tilt) / r));
const float compGib = -calib->gibmag * arm_sin_f32(ax + calib->gibphase);
const float compCurve = calib->curve * ay * ay;
return ax - (compTilt + calib->phase + compGib + compCurve);
```

  - V2

```
const float ax = atan2f(y, x);
const float r = arm_sqrt(x * x + y * y);
const float base = ax + asinf(clip1(z * tanf(t - calib->tilt) / r));
const float compGib = -calib->gibmag * arm_cos_f32(ax + calib->gibphase);
return base - (calib->phase + compGib);
```

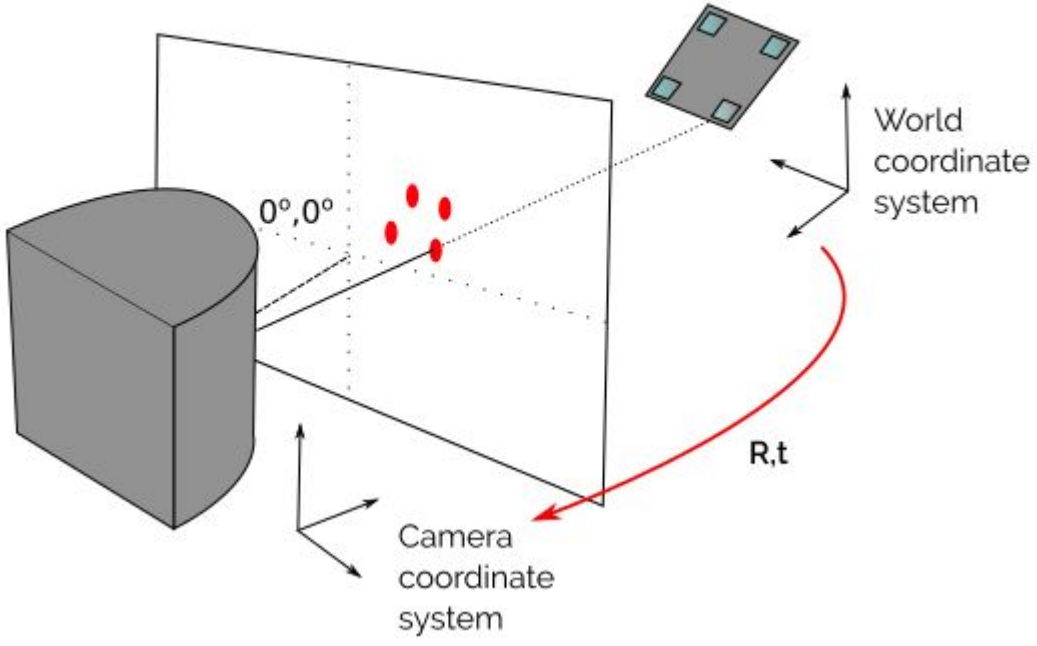
For V2: Curve and OGee are not used!

Hands-on: Implement calibration data

- Go to examples/app_api
- Check makefile if lighthouse deck is on
- Flash crazyflie
  - `python3 -m cfloader flash cf2.bin stm32-fw -w radio://0/10/2M/E7E7E7E709`
- Reboot for lighthouse deck
  - `python3 ../../tools/utils/reboot.py radio://0/10/2M/E7E7E7E709`
- Show the compensated angels in the logging tab cfclient

Find geometry basestations

- Translation and orientation
- Finds an object pose from 3D-2D point correspondences
  - Treats the basestation as a ‘camera’
- OpenCV
  - solvePnP



Inspired by solvepnp description (https://docs.opencv.org/)

Hands-on: Getting Geometry Data

- Go to tools/lighthouse
- Get geometry data:
  - `./get_bs_geometry.py --uri radio://0/10/2M/E7E7E7E709`
  - Always double check geometry if it looks okay!
- Flash firmware with geometry (or use flag --write)
- Show plotter for position estimate
- Show position estimate if calibration is turned on or off

Hands-on: Crazyflie flying in lighthouse

- Show spiral python script
  - Python3 fly_spiral.py

Lighthouse v2 : left to do

- Early release and subject to change
- Calibration implementation is not perfect
  - Maybe you want to help?? :)
- Add support for more than 2 basestations
- Adding support in tools in the client

Break and questions

We will start setting wireless chargers and more crazyflies in the meantime ;)

Swarm demo flying in the next hour!

See you back at 19:00 CET

Demo for Swarms

- Communication
- Autonomy
  - Highlevel commander
  - App layer

Communication

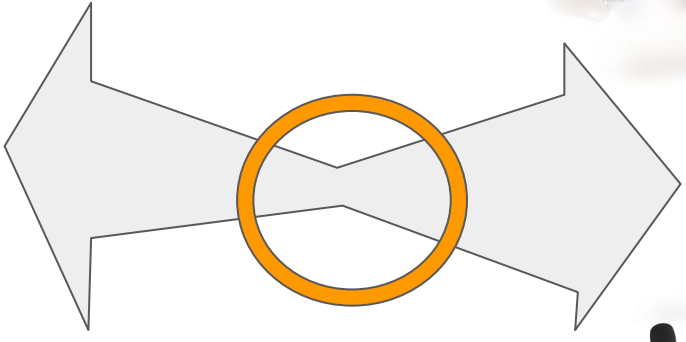
- Broadcasting
  - Messages with no return expected
- Multiple crazyflies per crazyradio
- Use same channel
- Can not send specific trajectories at each time step
- Crazyflies need to do more themselves

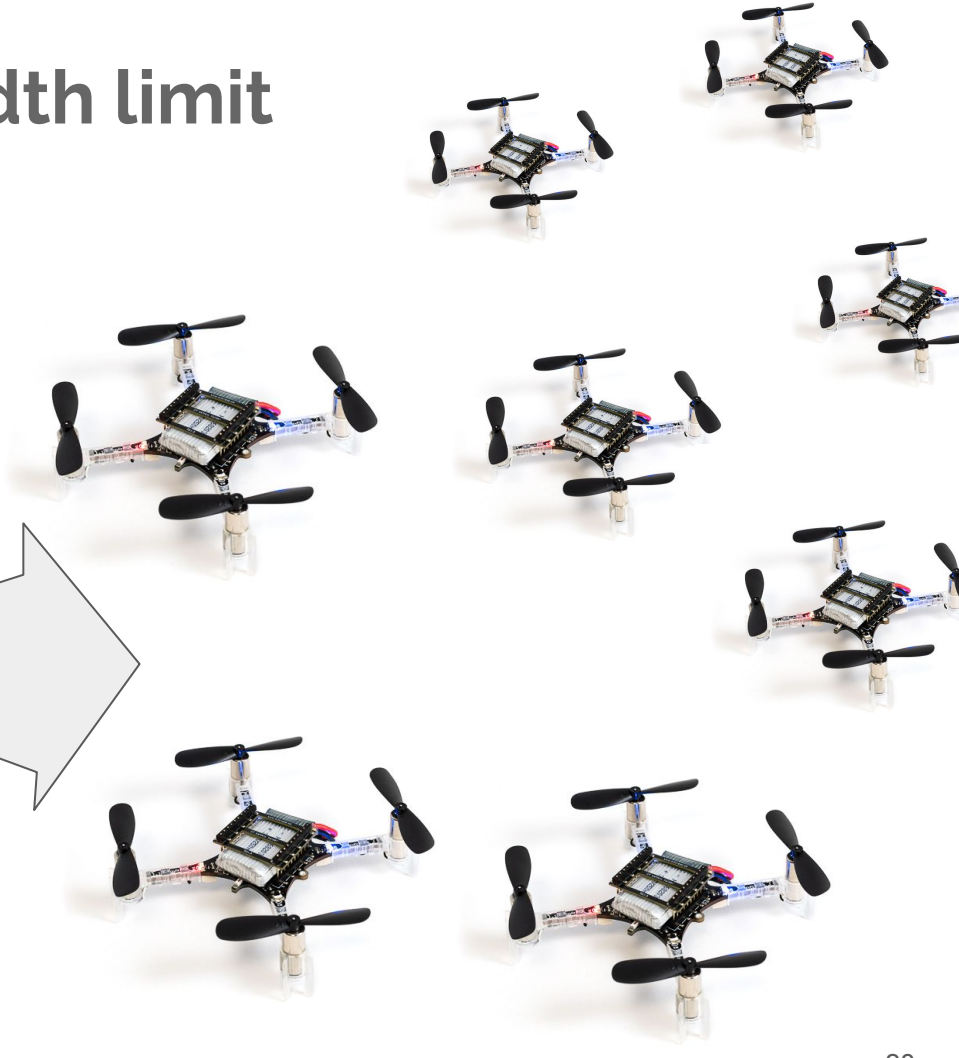
0xE7E7E7E701

0xE7E7E7E702

0xE7E7E7E703

Communication bandwidth limit

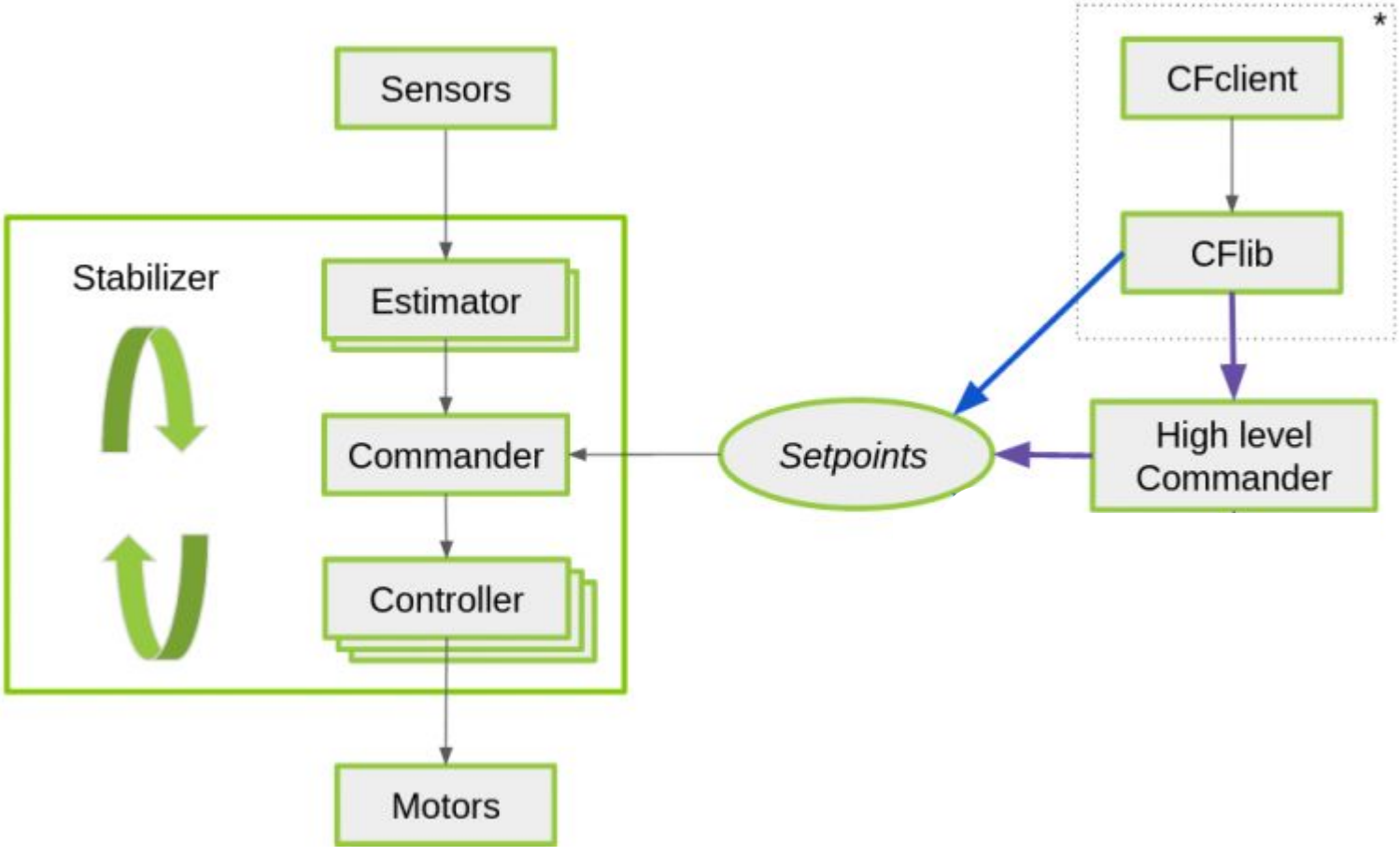




Hands-on: Flashing multiple crazyflies

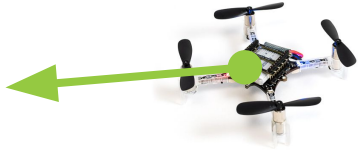
- Bit off-topic but this will need to be done in advance.
- Go to examples/demos/swarm_demo
- Don't forget to put in geometry!
  - Maybe this shifted during setting up
- Use the .sh script for flashing all of them
  - `./cload_all.sh`

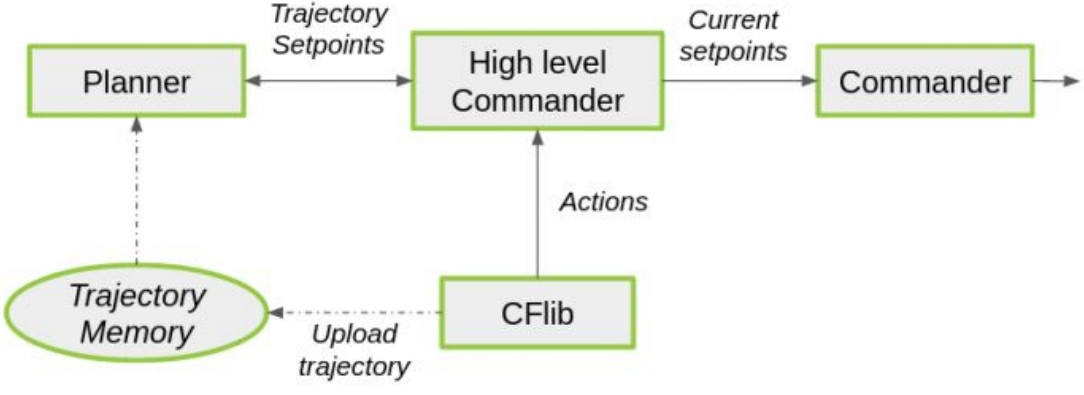
High level commander



Principle of HL commander

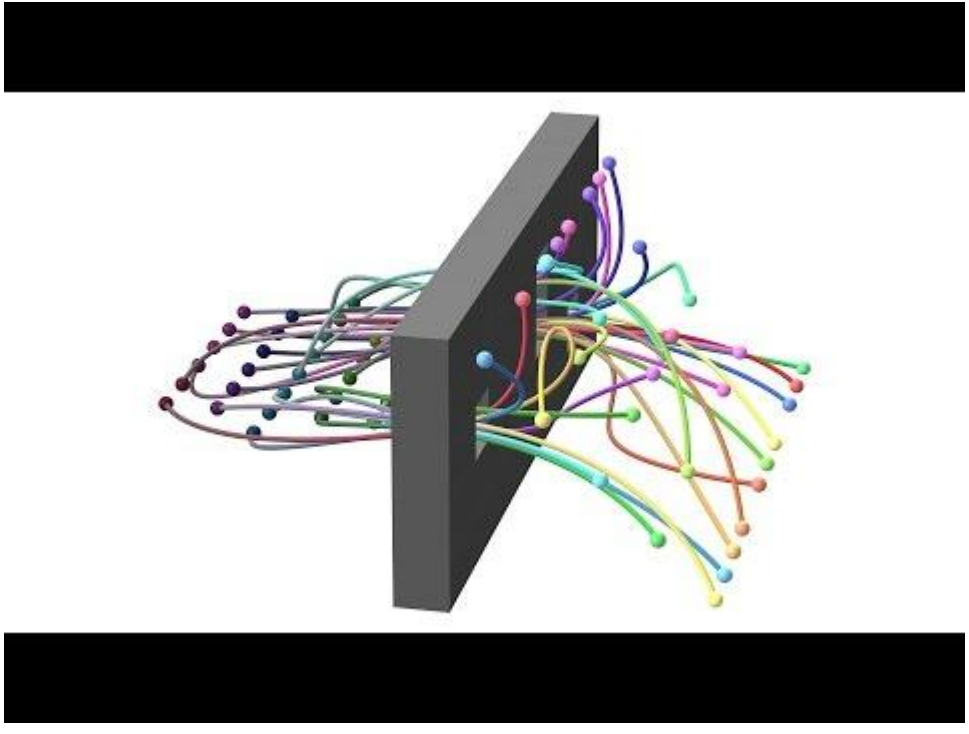
- Attitude commander
- Position/velocity commander
- High level commander





High level commander

- Implemented for Crazyswarm

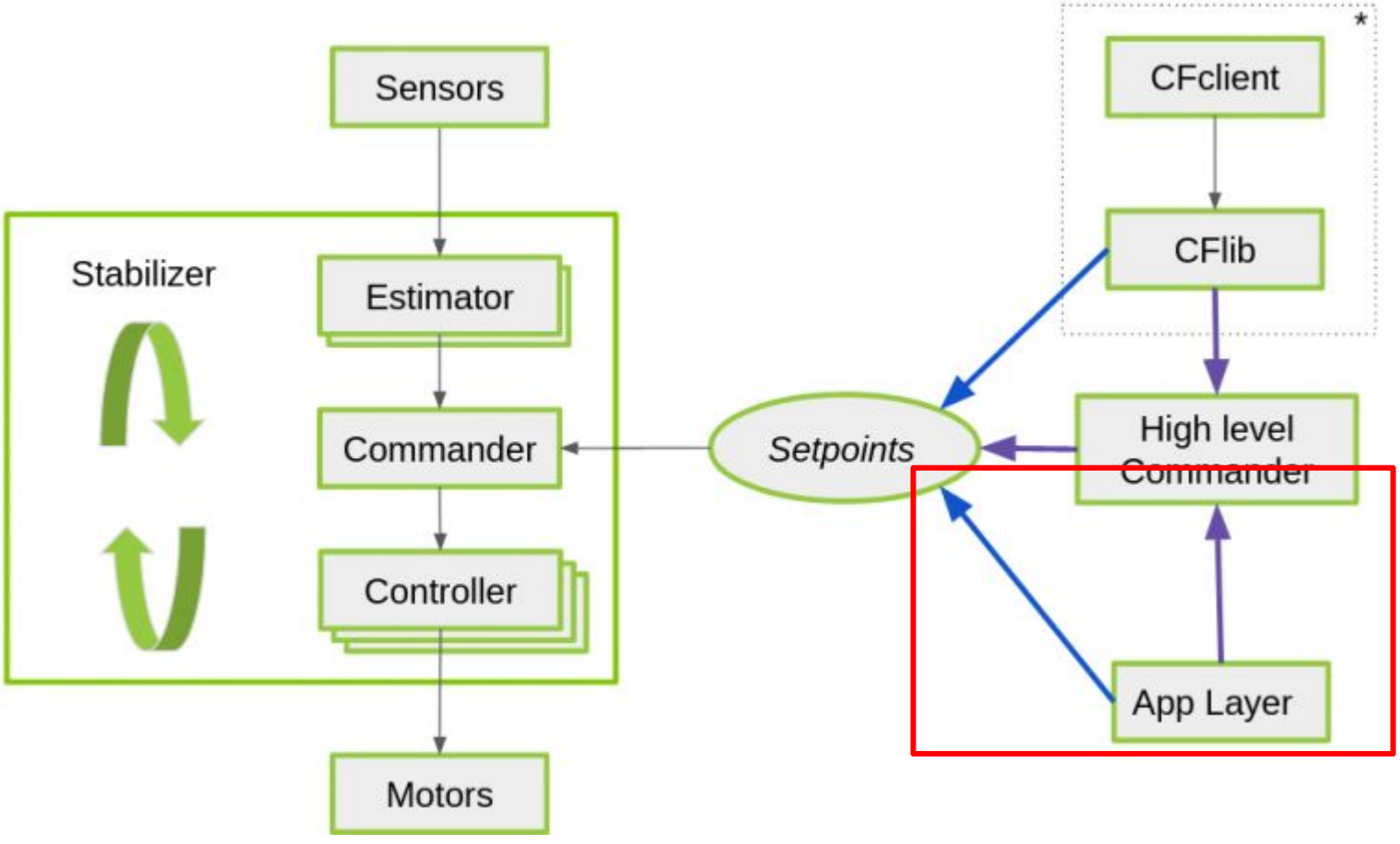


Preiss, James A., et al. "Downwash-aware trajectory planning for large quadrotor teams." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

Hands-on: Swarm with HL Commander

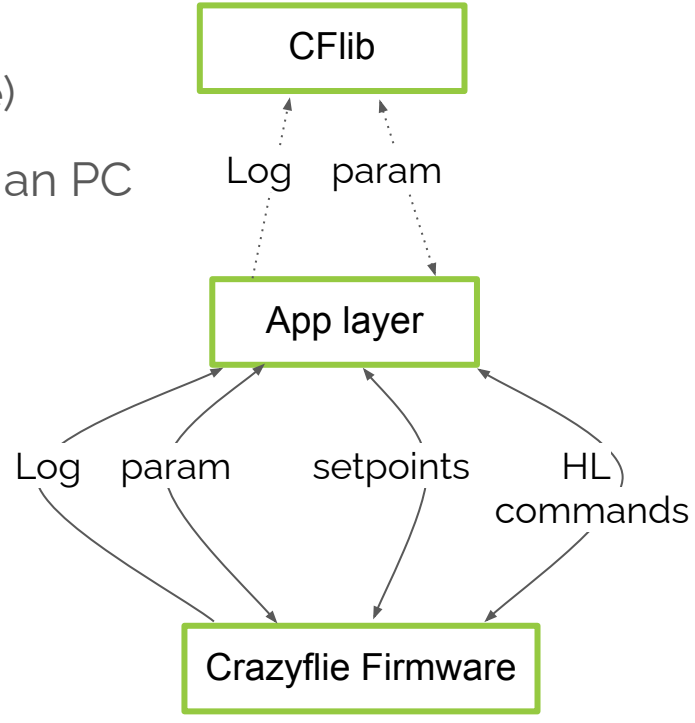
- Go to swarm script
- Fly first with two
- Then with all

More autonomy?



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



Video example

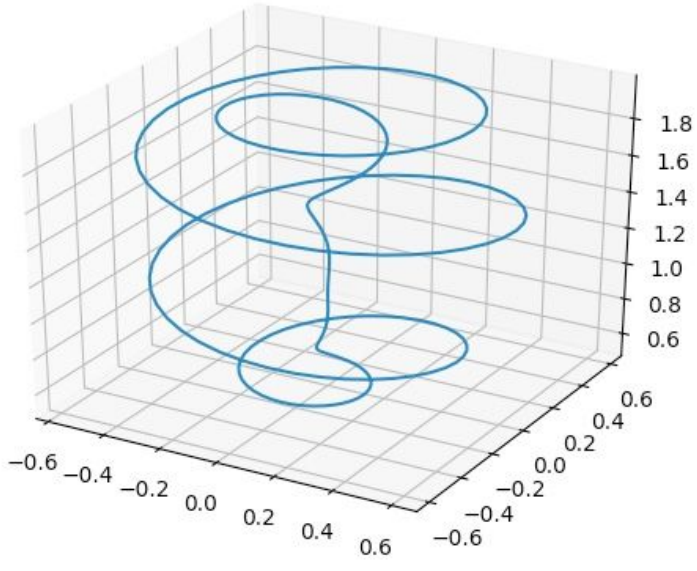
Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment, K.N. McGuire, C. De Wagter, K. Tuyls, H. Kappen, G.C.H.E. de Croon. Science Robotics (2019)

Hands-on: simple hello world

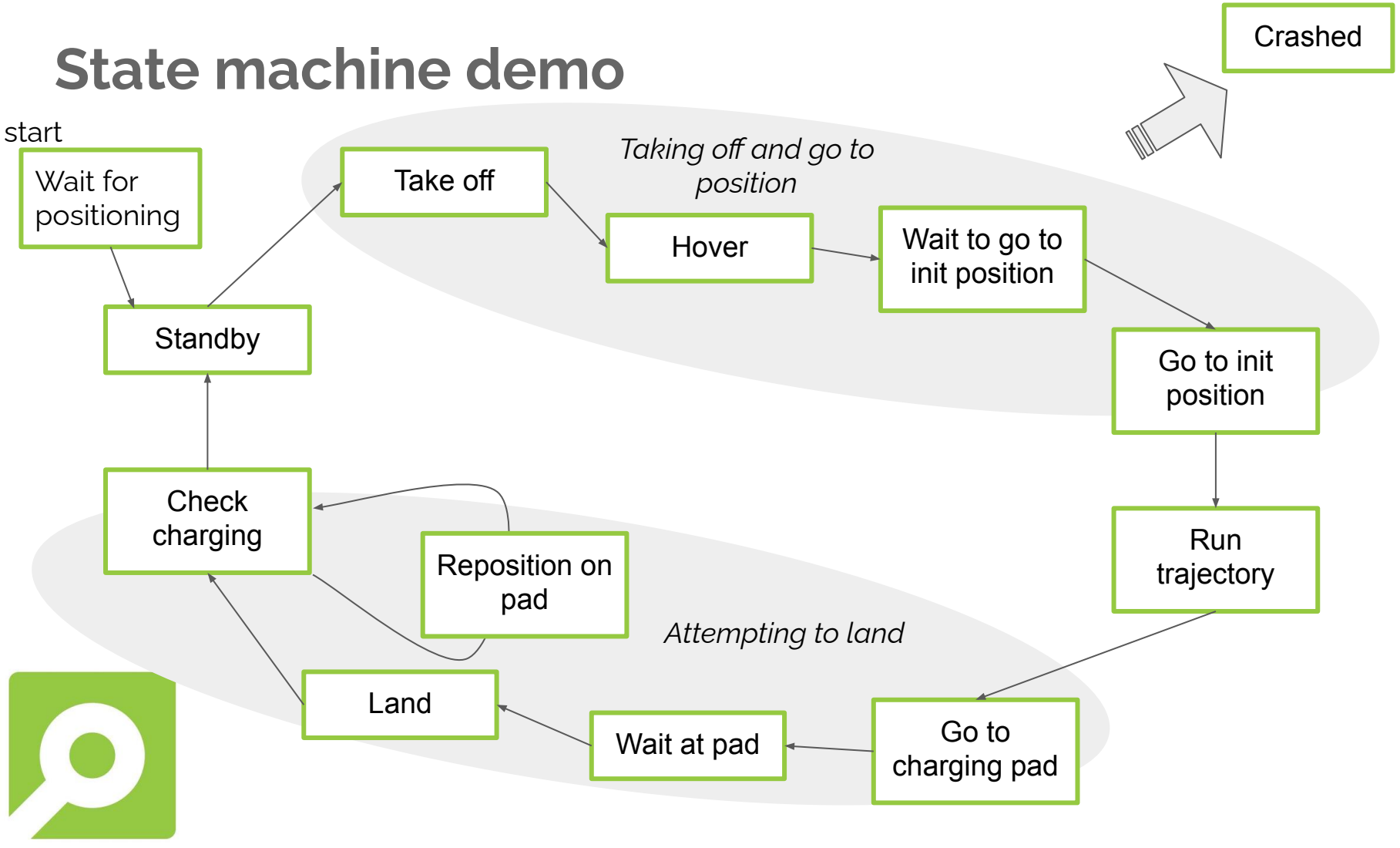
- App layer examples
  - Go to examples/app_helloworld
- Show the makefile

Demo implementation app layer

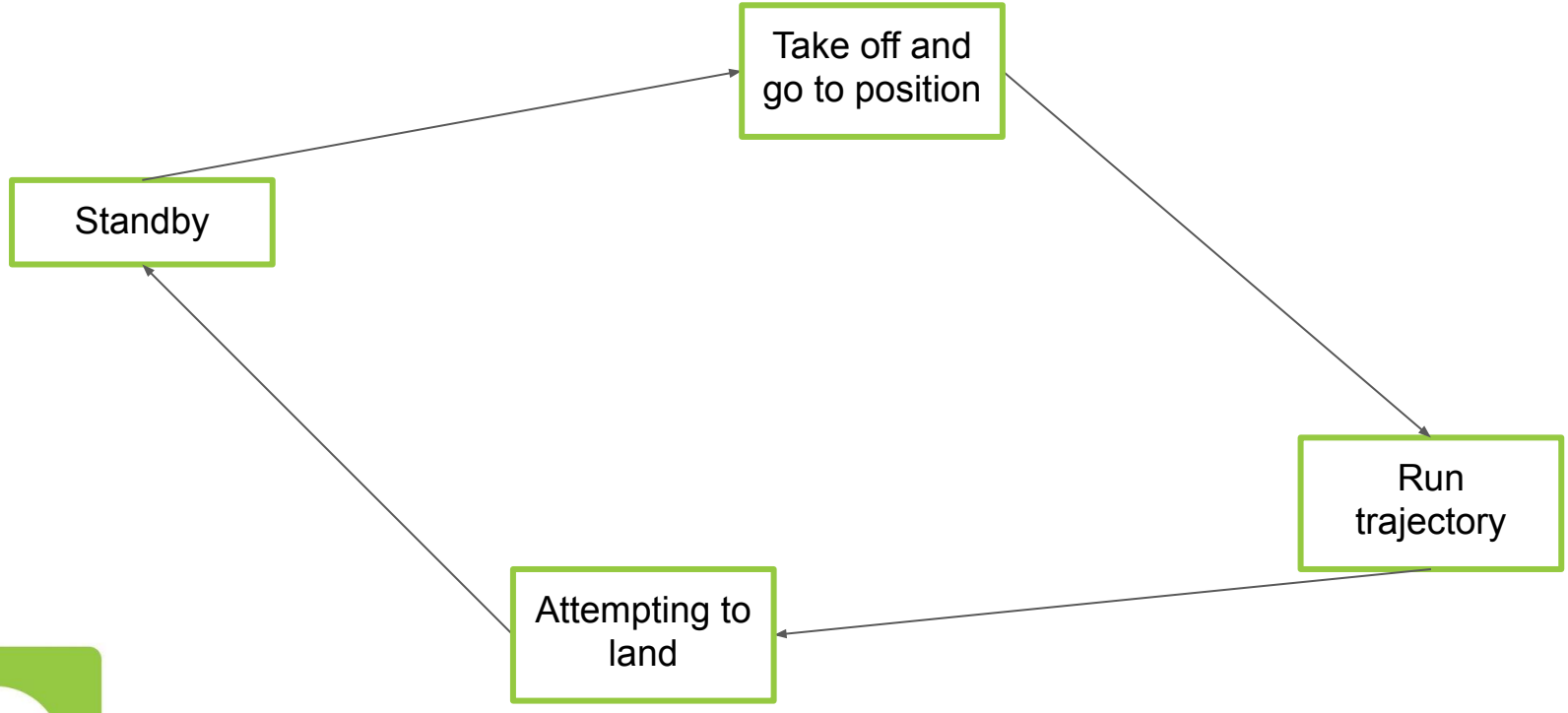
- Spiral trajectory
- Auto landing
- State machine
- Connection with cfclient control tower
  - Only indicates which crazyflie should ‘start’



State machine demo



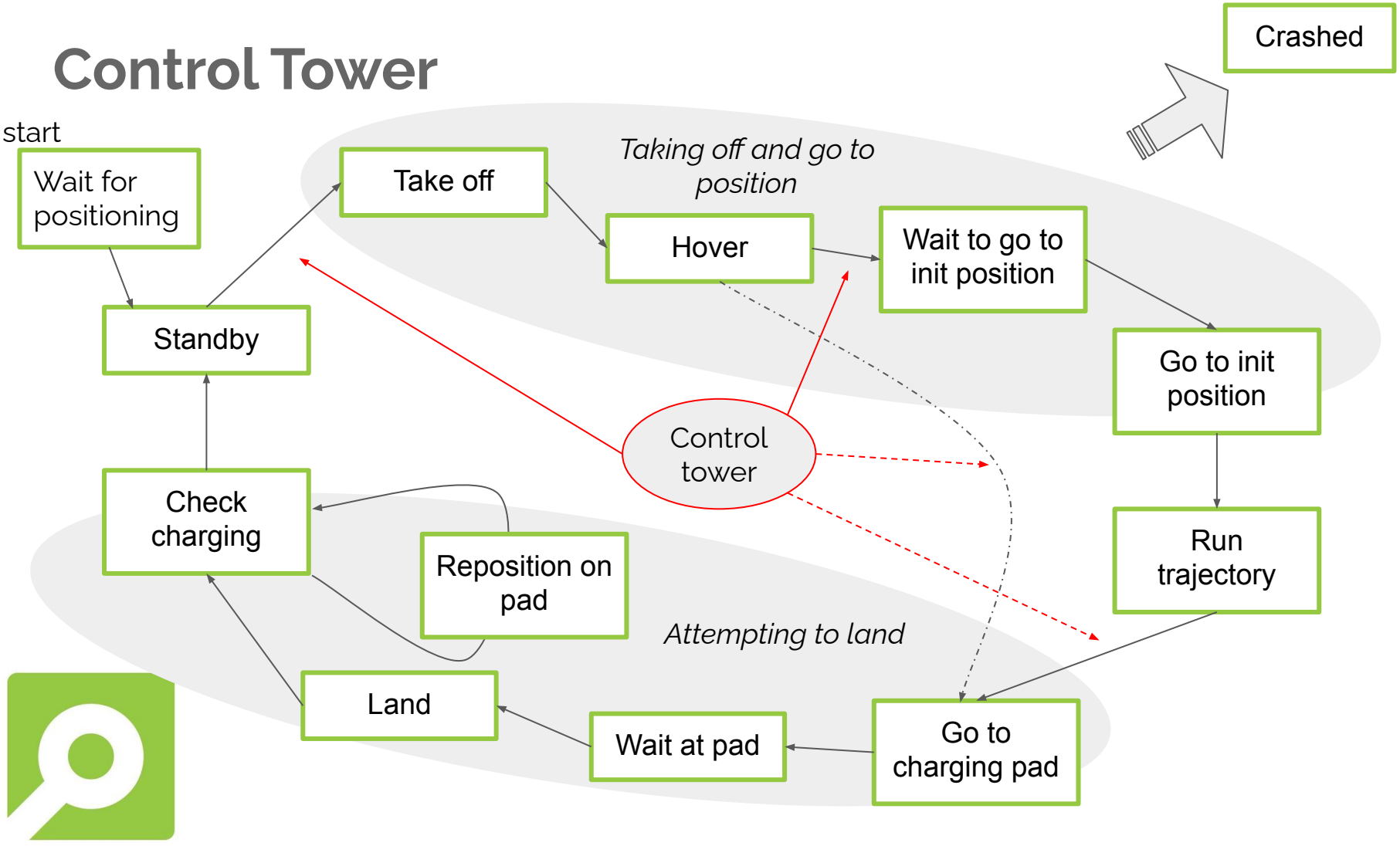
State machine demo



Show implementation code

- Go to examples/demos/swarm_demo
- Show app.c
  - Indicate where trajectories are
- (This has already been flashed before)

Control Tower



Control tower

- Communicates params
  - When to hover and wait
  - When to start trajectory
  - *When to terminate trajectory* (optional)
- Monitors state
  - Battery level
  - Crashed
  - State machine
- Very limited communication

Hands-on: Show the control tower script

- Show the control tower script
- Connect all crazyflies without flying
  - `python3 control_tower/control_tower.py 0`
- Show the gui
  - Separate terminal: `python3 control_tower/tower_gui.py`

Hands-on: Let's look at the demo

- One crazyflie flying
- 2 in one spiral
- Multiple at the same time

Video of 9 CFs flying

Recap steps to setup demo

1. Get calibration data basestations
2. Set-up basestations
3. Put one crazyflie with calibration data flashed on the floor
4. Get geometry basestations
5. Setup wireless chargers
6. Flash all crazyflies
7. Start the control tower script and the tower GUI script
8. Done!

What to improve in the future?

- Lighthouse 2 improvement
- Fully independent of control tower
  - Maybe with Peer to Peer ?
- Suggestions from you?

Questions and let's socialize!

- Slide and videos will be on *www.bitcraze.io/events/bct12020*