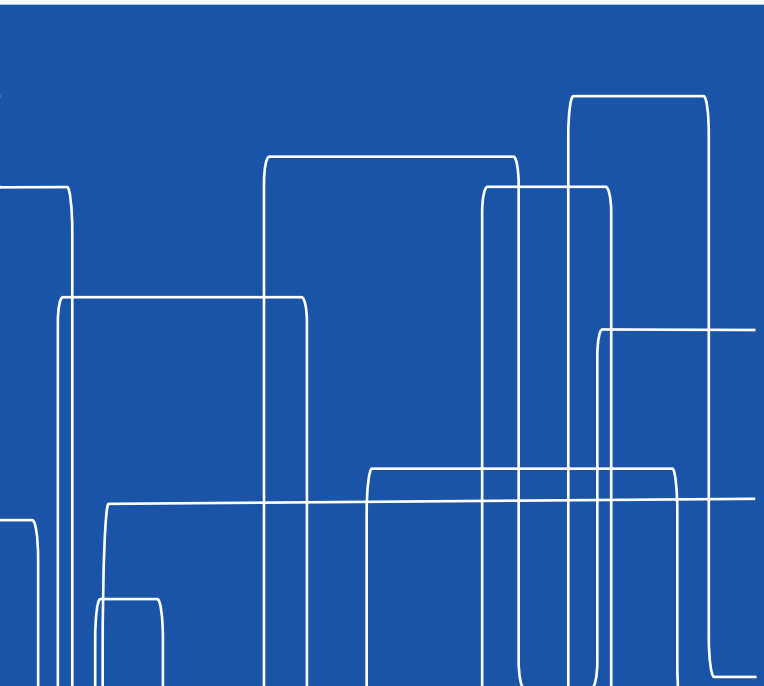


Coordination of Autonomous Aerial Vehicles

ACCESS Summer Project

June - August 2015

Project overview

Aim: Design a computer program to fly one/multiple quadcopters

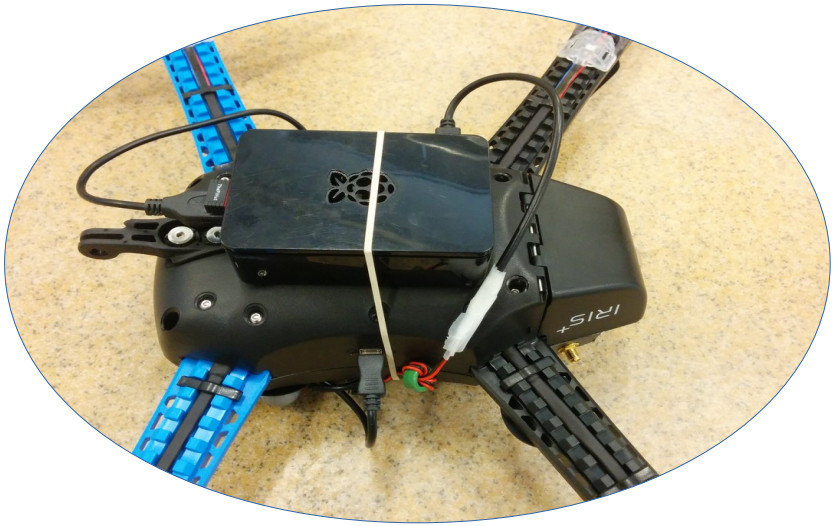
- Flexible software architecture
- Safe flying
- Simulator with same interface
- User friendly GUI
- Trajectory following
- More advanced tasks

IRIS+

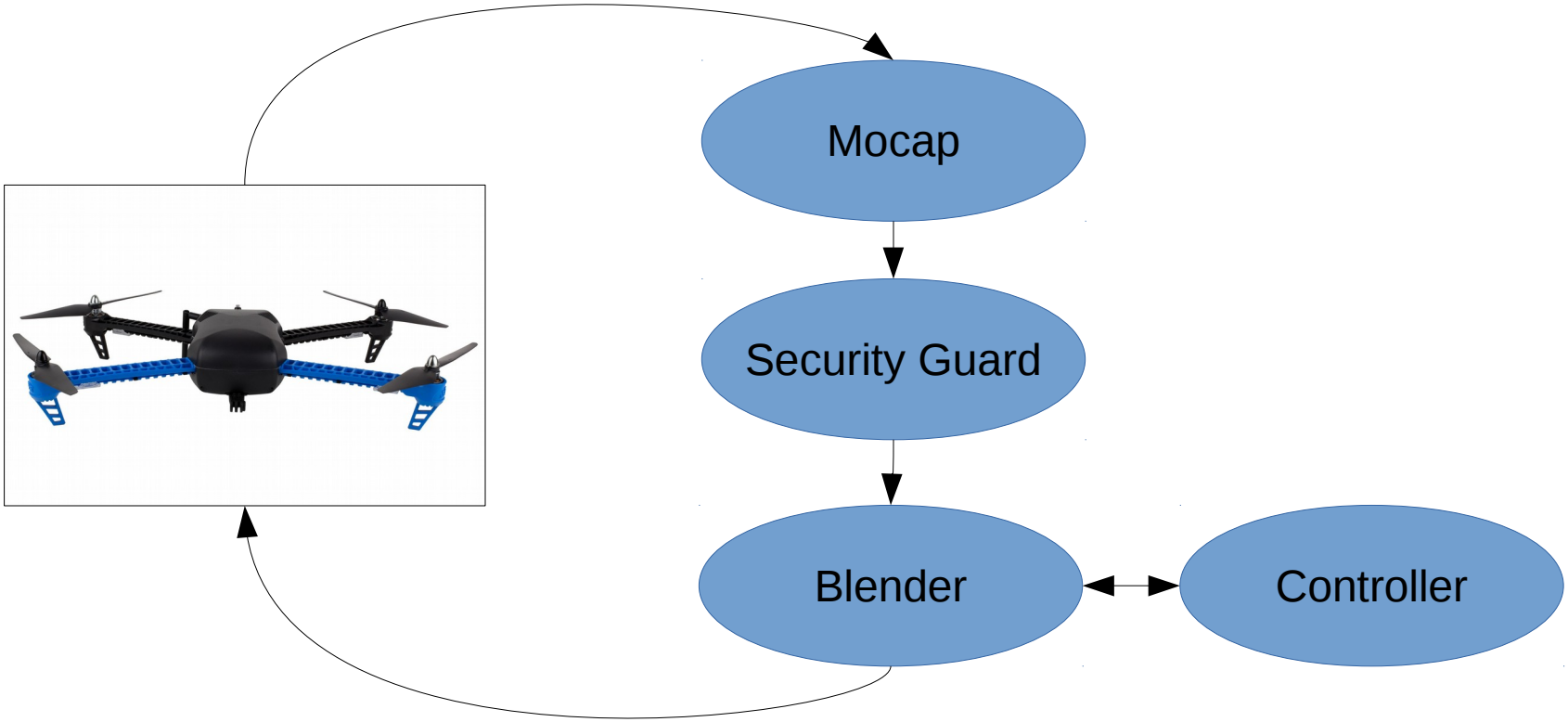
- Has internal software that transforms roll, pitch, throttle and yaw to motor speeds
- Can fly for 15 min
- Able to carry about 500 g

Transmission

- 433 MHz – interference problems
- Wi-Fi through a Raspberry Pi



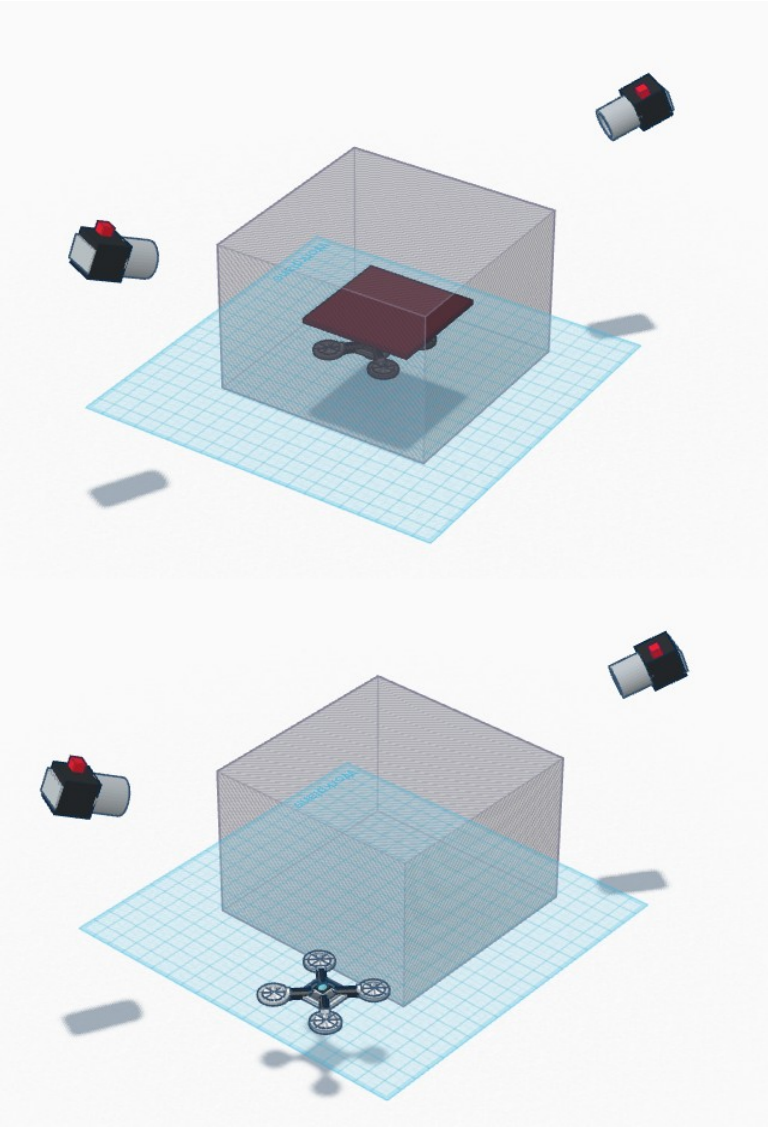
Architecture overview



Security features

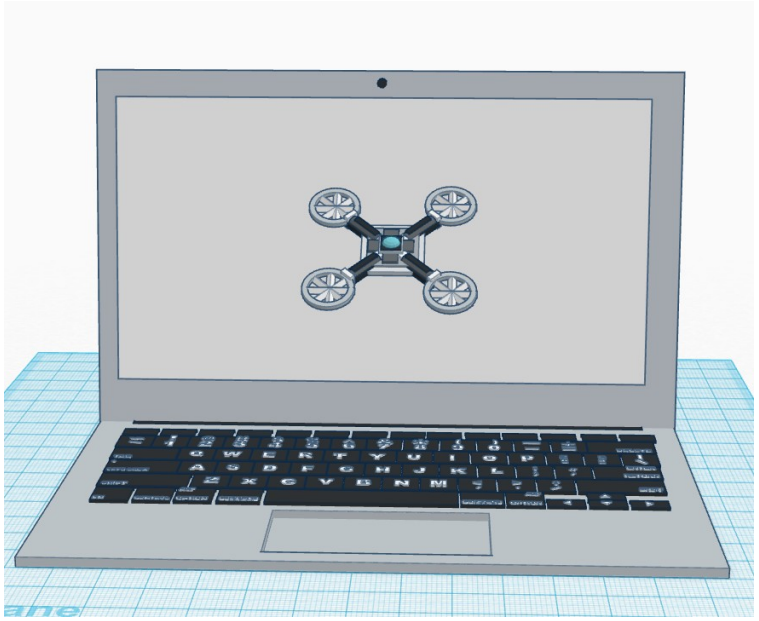
Will land if

- Motion caption signal is lost
- Out of safety zone
- Landing requested



Simulator

- Same interface as real quadcopter
- Test code in simulation before real world
- Can handle both one and multiple quadcopters



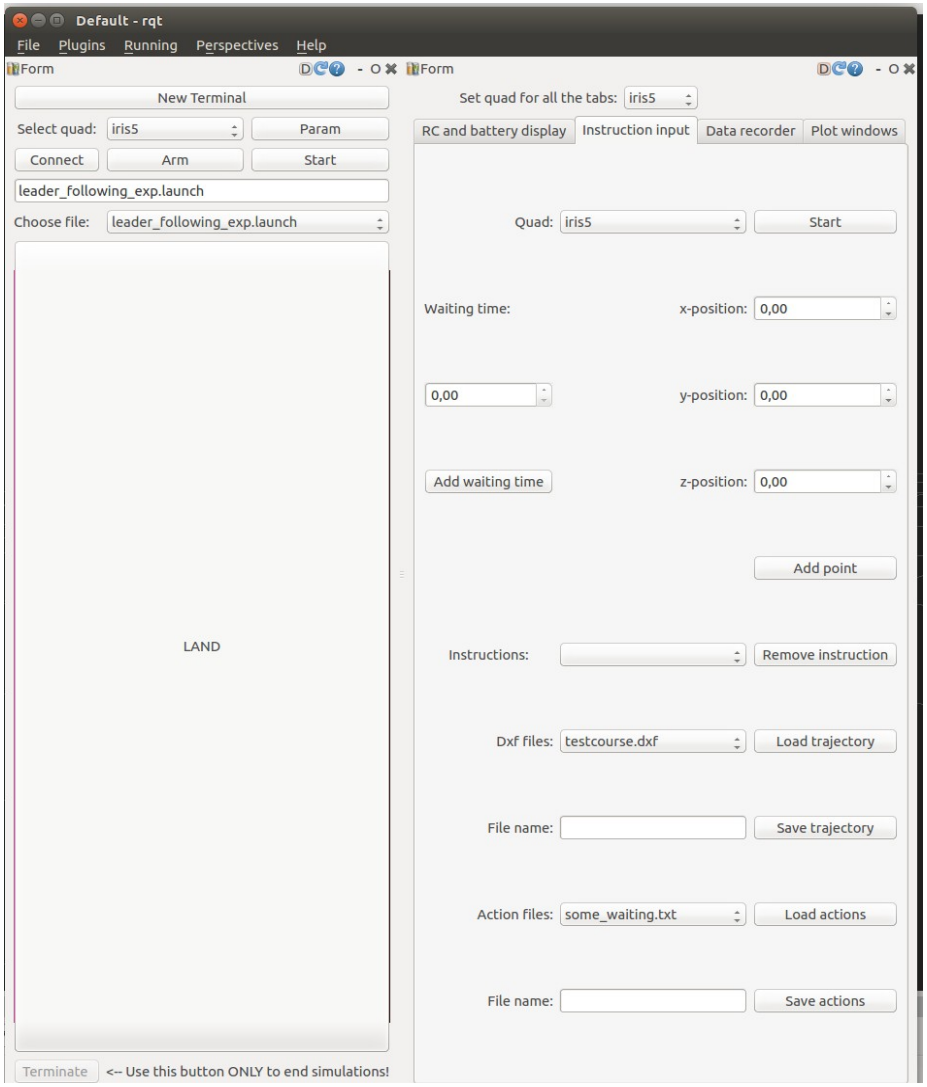
GUI

- Different plugins for different things – more flexible

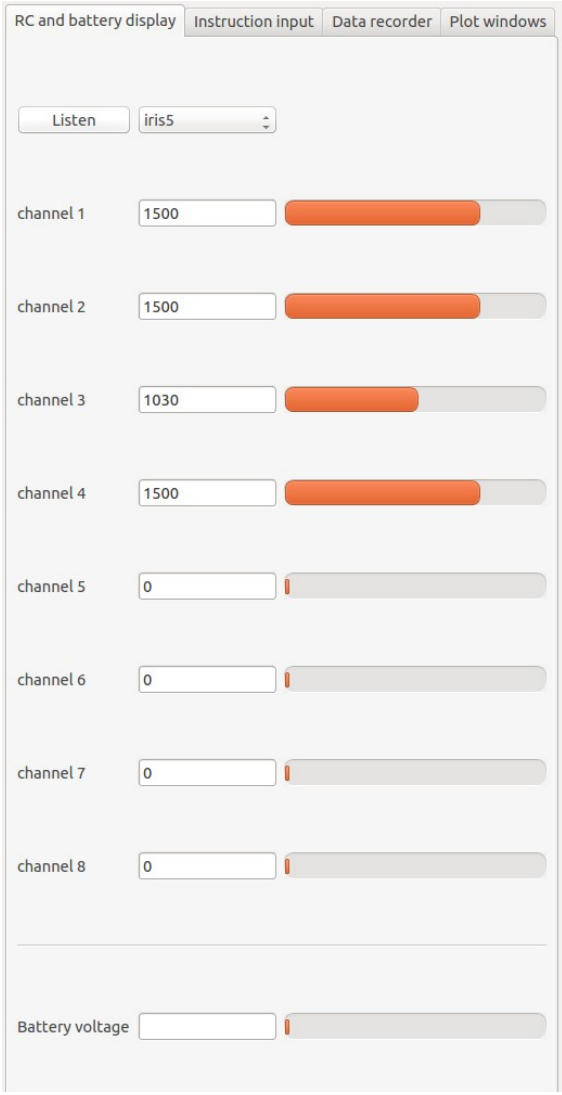
The GUI should be able to

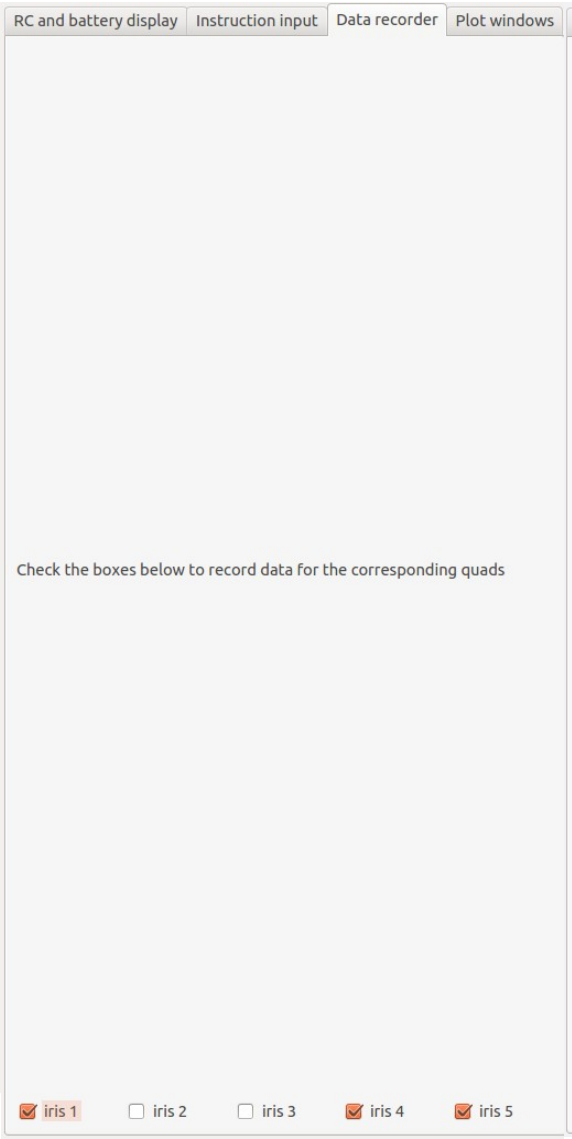
- Take inputs from user
- Show outputs to the quadcopter
- Show status of quadcopter

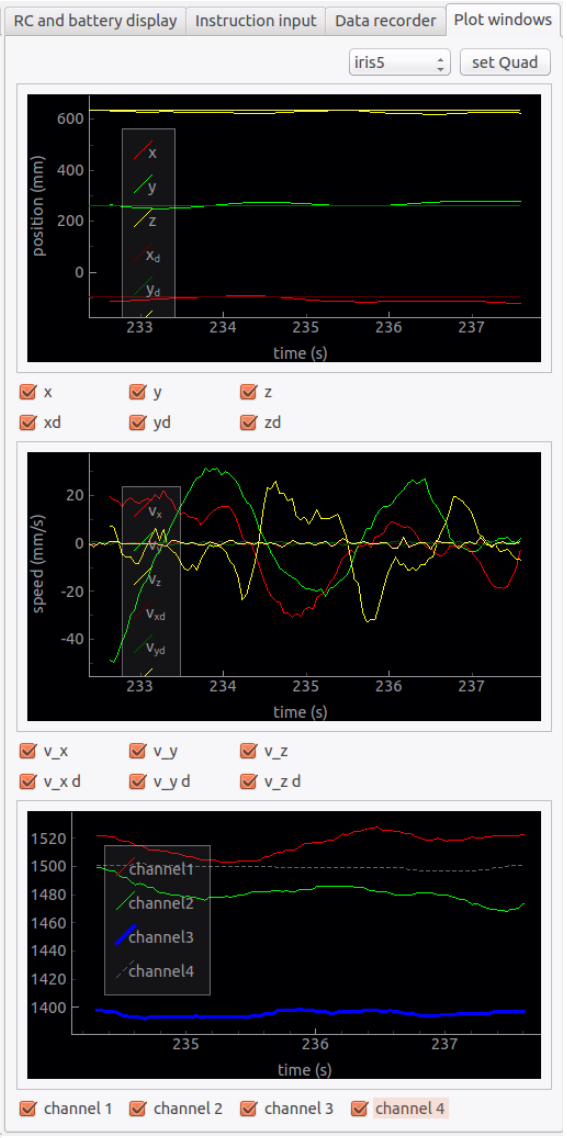
GUI


Default - rqt
File Plugins Running Perspectives Help
Form
New Terminal
Select quad: iris5
Param
Connect
Arm
Start
leader_following_exp.launch
Choose file: leader_following_exp.launch
LAND
Terminate
<-- Use this button ONLY to end simulations!
Form
Set quad for all the tabs: iris5
RC and battery display
Instruction input
Data recorder
Plot windows
Quad: iris5
Start
Waiting time:
x-position: 0,00
0,00
y-position: 0,00
Add waiting time
z-position: 0,00
Add point
Instructions:
Remove instruction
Dxf files: testcourse.dxf
Load trajectory
File name:
Save trajectory
Action files: some_waiting.txt
Load actions
File name:
Save actions

GUI


RC and battery display
Instruction input
Data recorder
Plot windows
Listen
iris5
channel 1
1500
channel 2
1500
channel 3
1030
channel 4
1500
channel 5
0
channel 6
0
channel 7
0
channel 8
0
Battery voltage

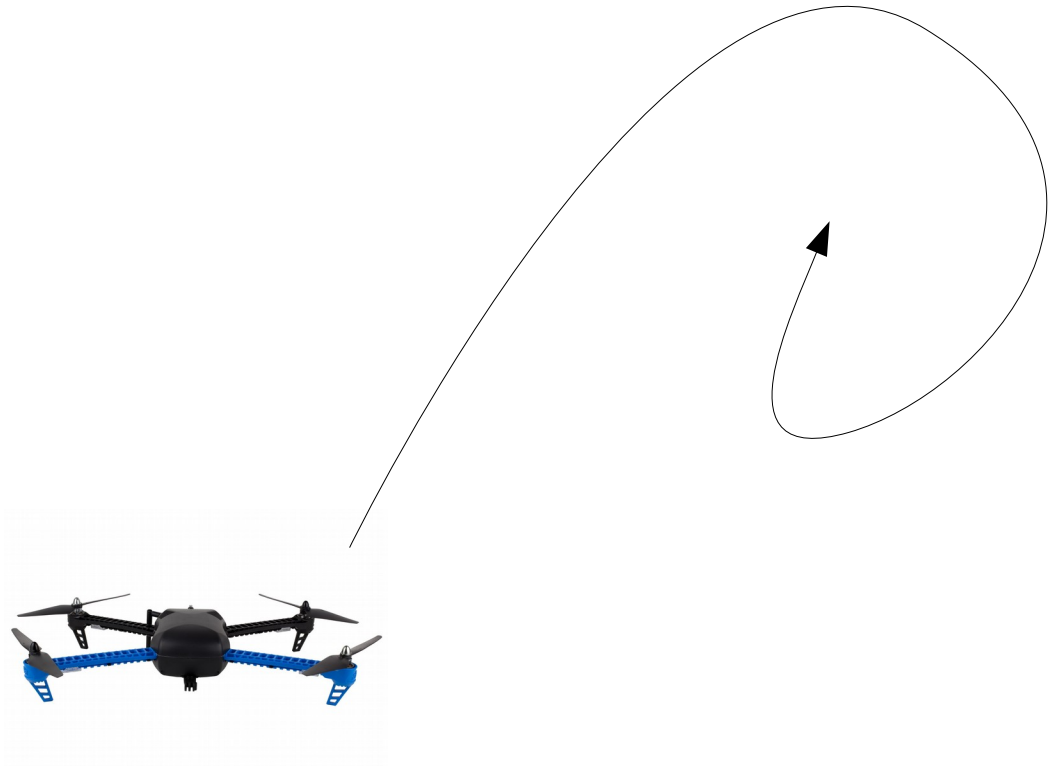

RC and battery display
Instruction input
Data recorder
Plot windows
Check the boxes below to record data for the corresponding quads
iris 1
iris 2
iris 3
iris 4
iris 5


RC and battery display
Instruction input
Data recorder
Plot windows
iris5
set Quad
position (mm)
time (s)
x
y
z
xd
yd
zd
speed (mm/s)
time (s)
v_x
v_y
v_z
v_x d
v_y d
v_z d
channel1
channel2
channel3
channel4
time (s)
channel 1
channel 2
channel 3
channel 4

Trajectory following

Generate points for the quad with

- Position
- Velocity
- Acceleration
- “Virtual leader”

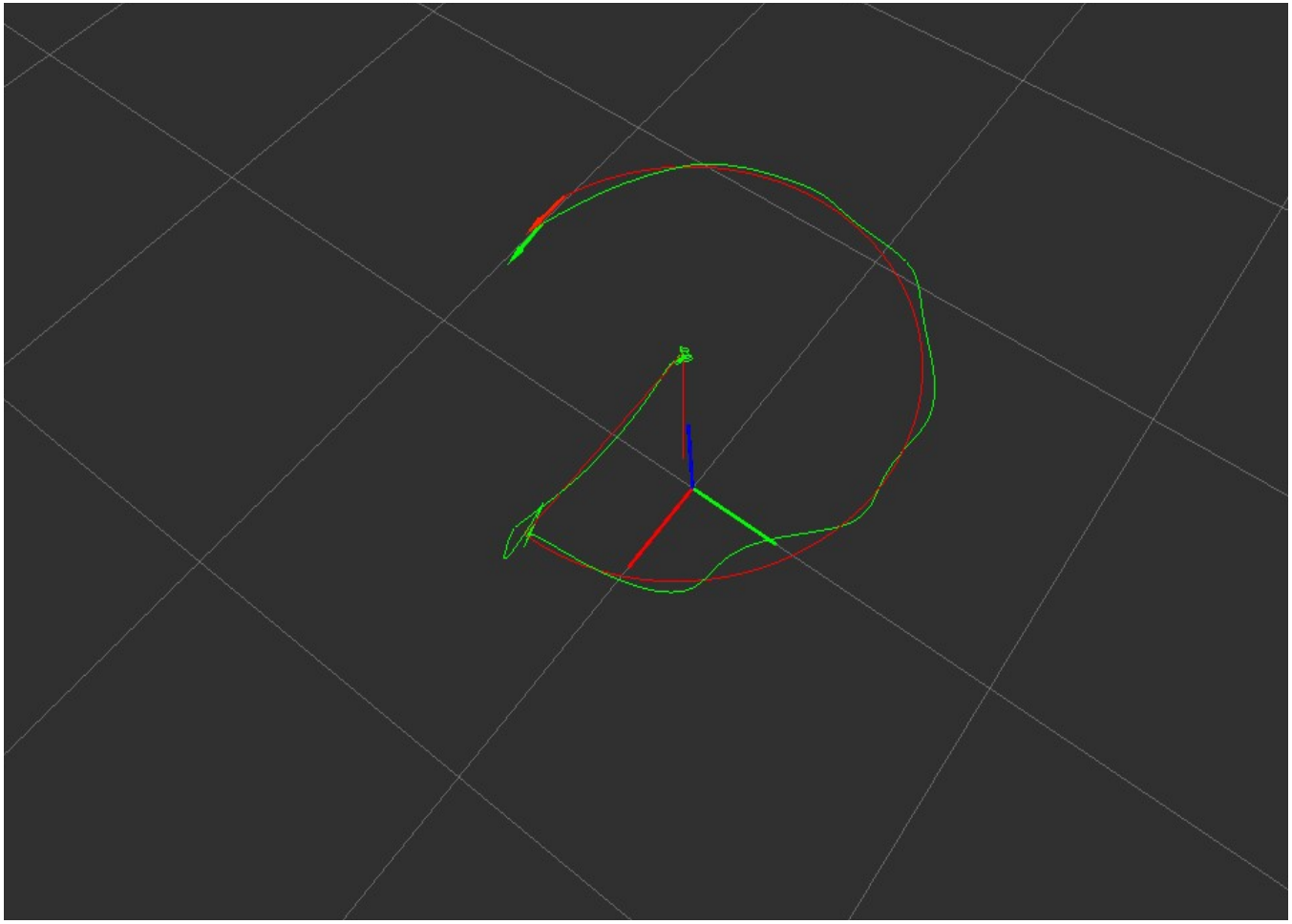


PID

- Control outputs are based on errors
- Model-based optimal tuning
- Quadcopter largely affected by battery voltage
- Error around a couple of centimeters are required for finer tasks

PID

- Error visualization is important



Obstacle avoidance

- Potential between quadcopter and obstacle
- Pushes the quadcopter away from the obstacle, to a lower potential

Obstacle avoidance

- PID input and potential are combined, “blended”, with a convex combination

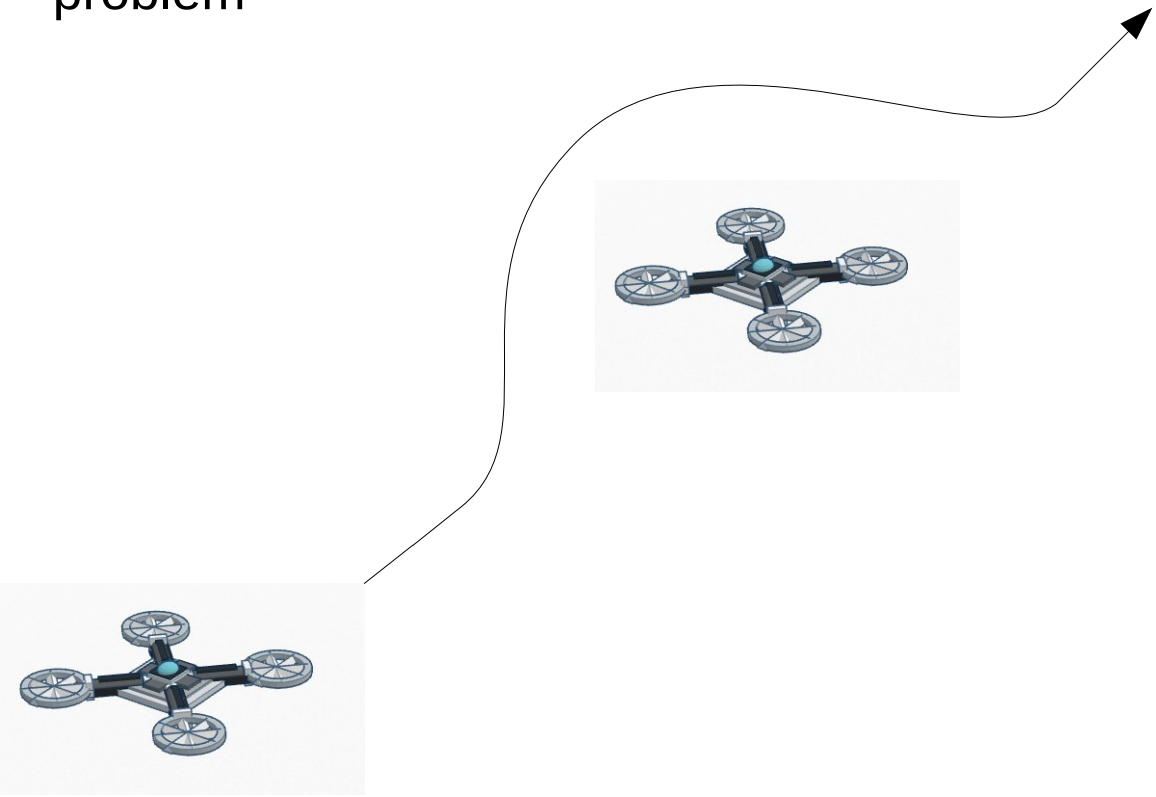
$$\text{output} = \alpha_{PID} \cdot \text{PID-input} + \alpha_{pot} \cdot \text{potential}$$

$$\alpha_{PID} + \alpha_{pot} = 1$$

- α_{PID} and α_{pot} can be dependent on distance to obstacle

Obstacle avoidance

- Simple version: Ignoring z-direction, resulting in a 2D problem



Leader following

- First quadcopter: Follow trajectory
- Second quadcopter: Keep constant offset to first quadcopter
- **Demo!**

Leader following

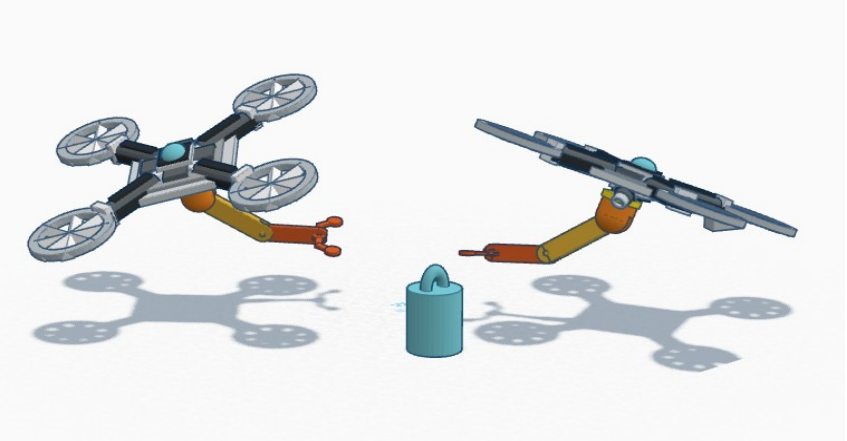
Possible continuation:

- Attractive potential, mimicking flocking behavior

Load lifting

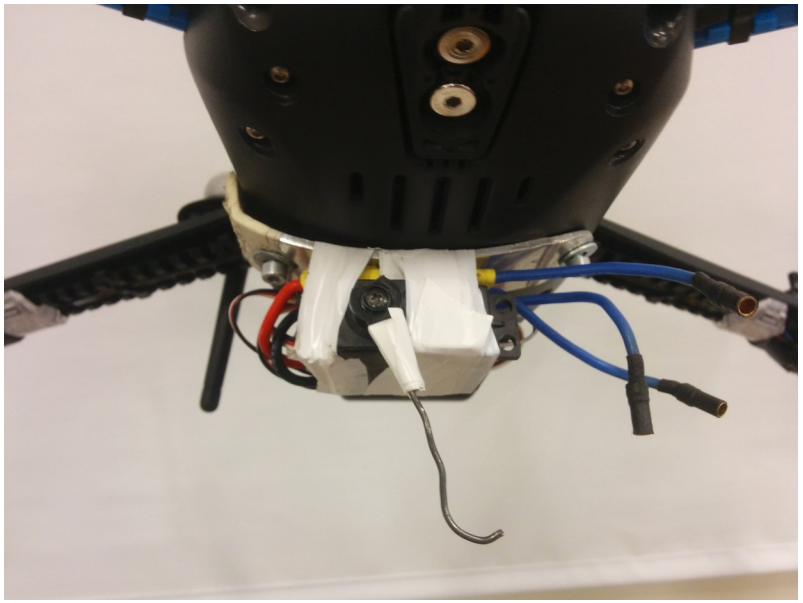
A quadcopter with a gripper can transport a load by

- Read and go to position of load
- Grip load
- Lift and transport load through a given trajectory
- Release load

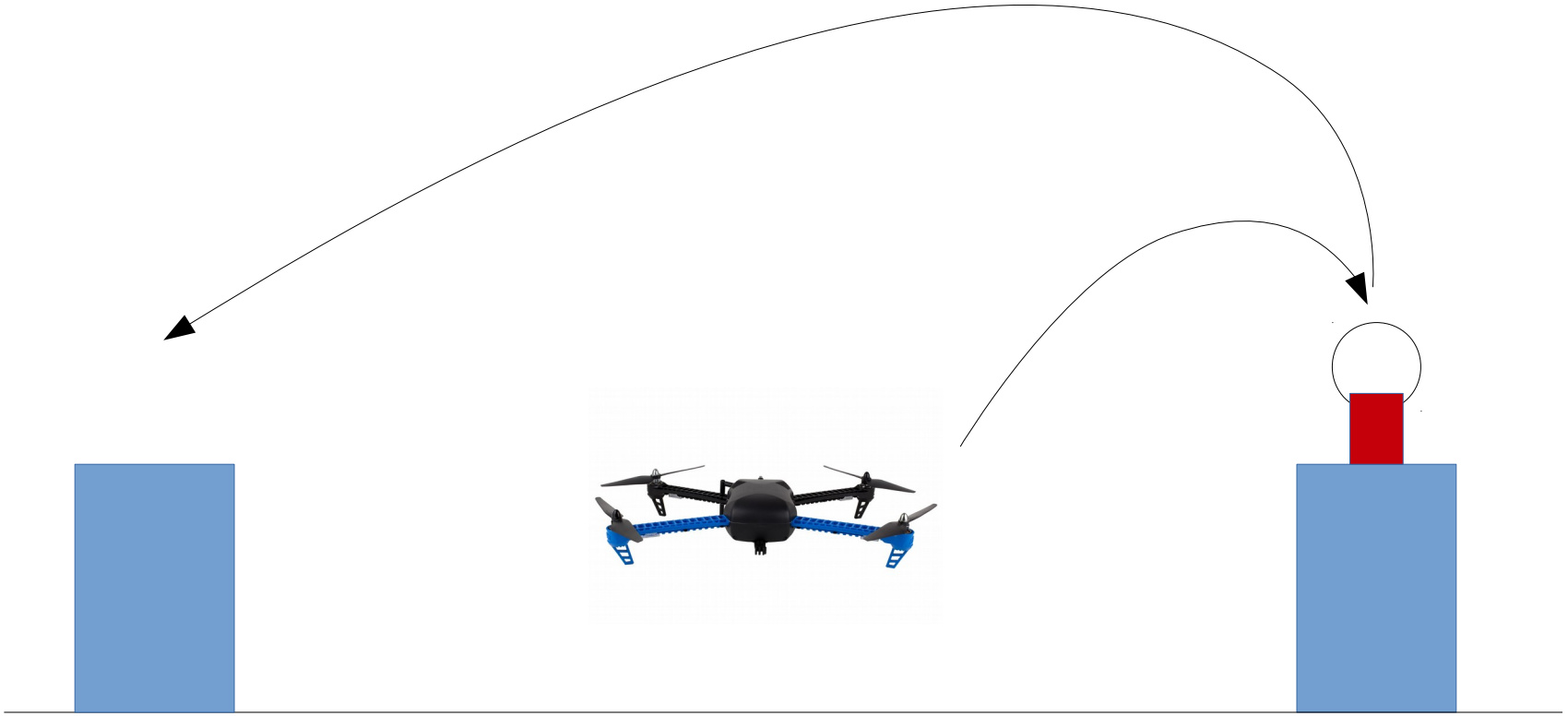


Load lifting

- Simple gripper: A hook on a servo



Load lifting



Load lifting

Possible continuation:

- Pick up objects in a warehouse
- Attach a spray can instead, and let it paint a wall
- Controller to minimize the swing

Thanks for your attention!

Questions?