Ein Regler zur Kollisionsvermeidung von Flugrobotern

Bernd Hügel, Jürgen Eckert, Reinhard German und Falko Dressler

Computer Networks and Communication Systems,
University of Erlangen, Germany

November 19th, Boppard

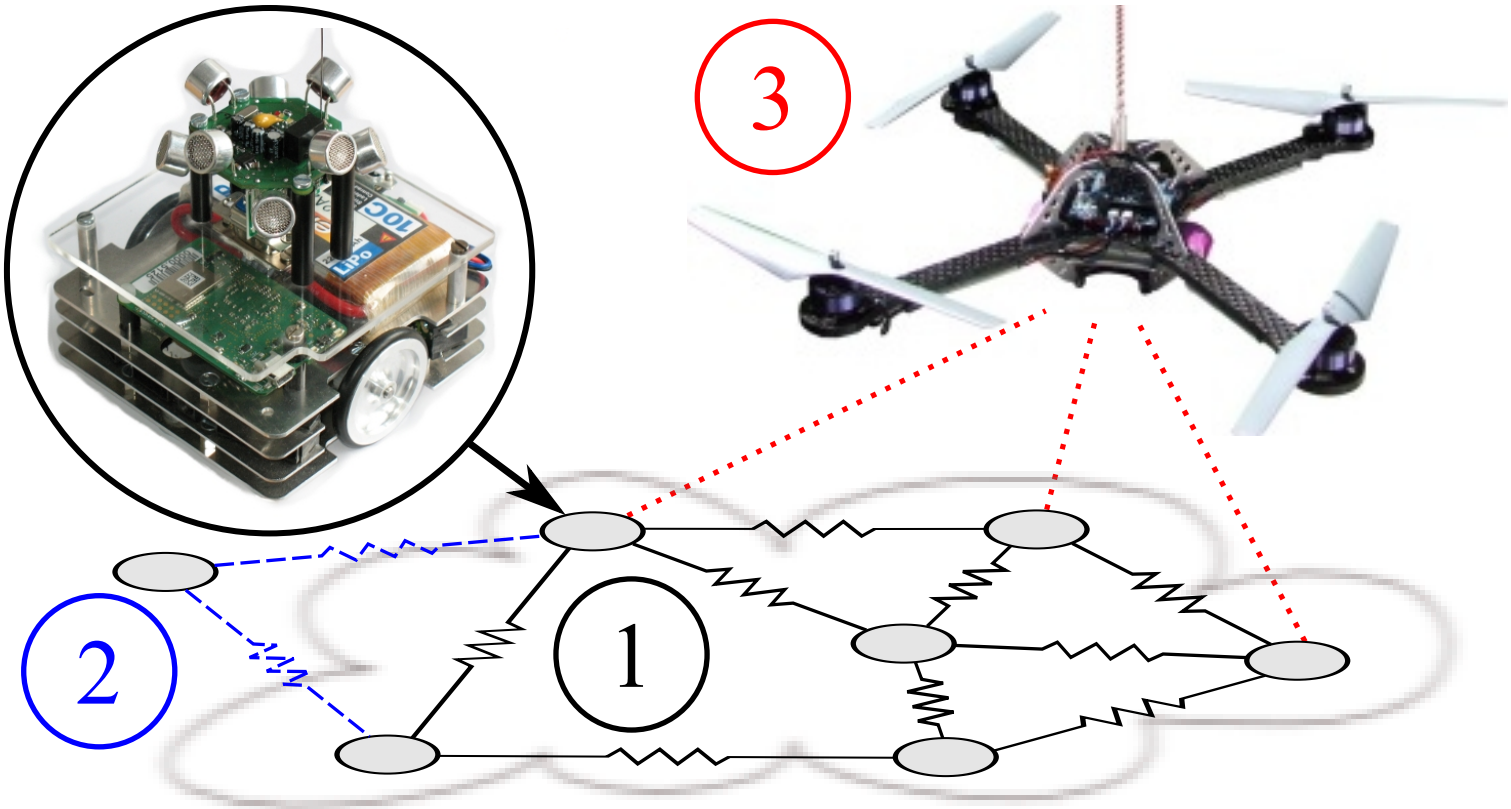
Outline

Motivation

Objective: **Local positioning system**

- Zero-configuration
- Fully distributed
- Self-organized
- Ubiquitous indoor environment

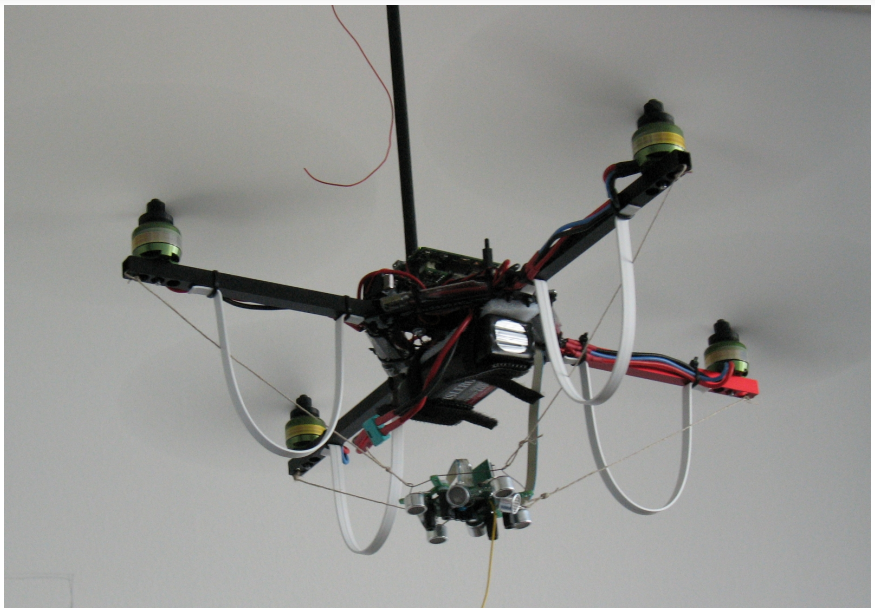
Demo: Autonomous flying four-rotor robot

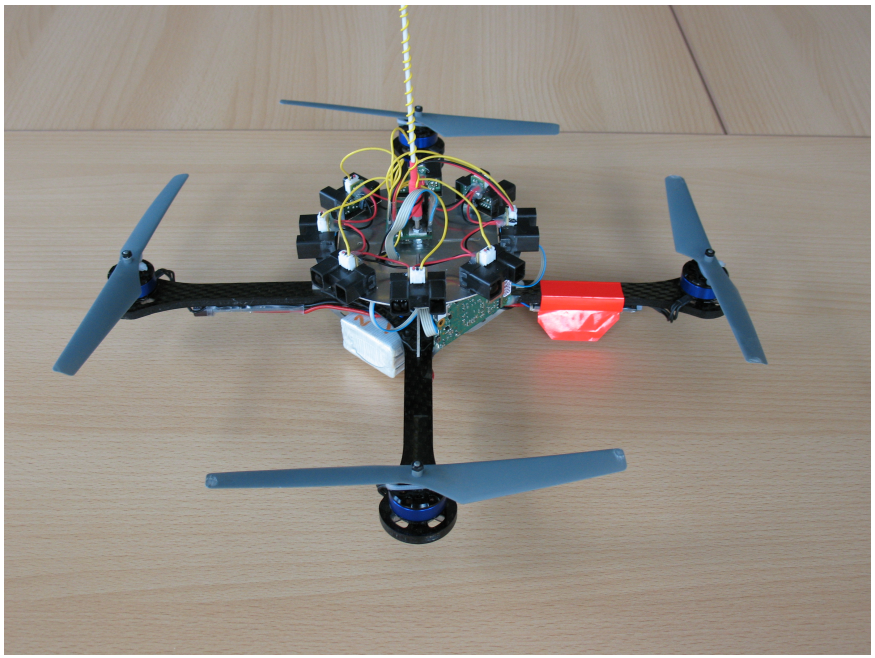


Approach:

1. Mobile robots span a coordinate system (anchor-free, but unique)
2. New robots join and extend the existing network
3. Localization support for clients (e.g. quadrocopters)

Flying Roboter





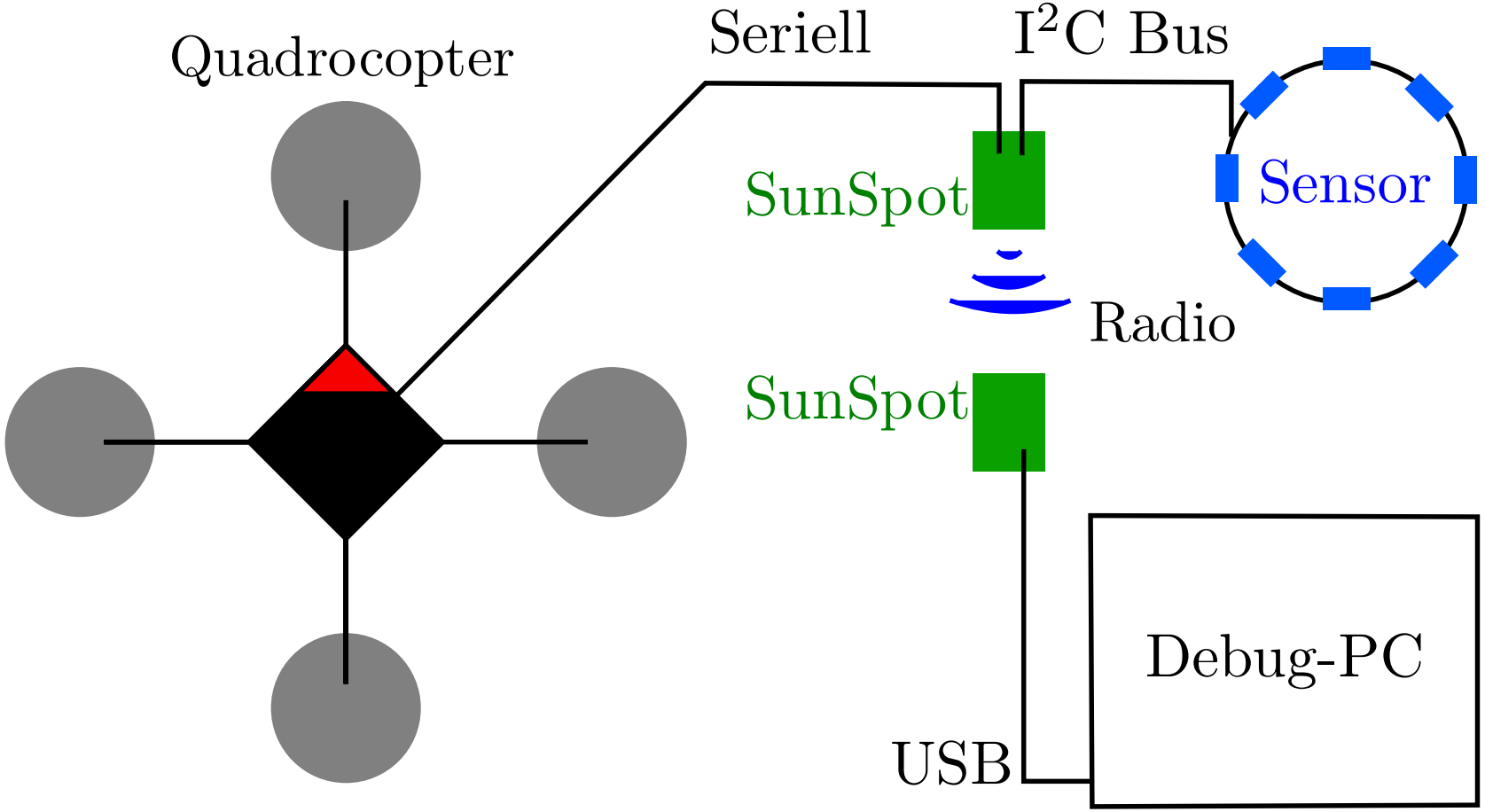
Off the shelf Quadrocopter

- Microkopter M2
- AscTec Hummingbird

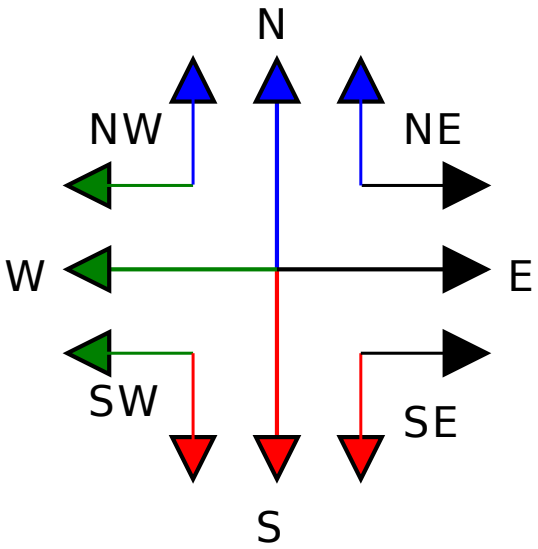
Equipped with

- Sensor Node (SunSpot)
- Localization Hardware
- Distance Sensors

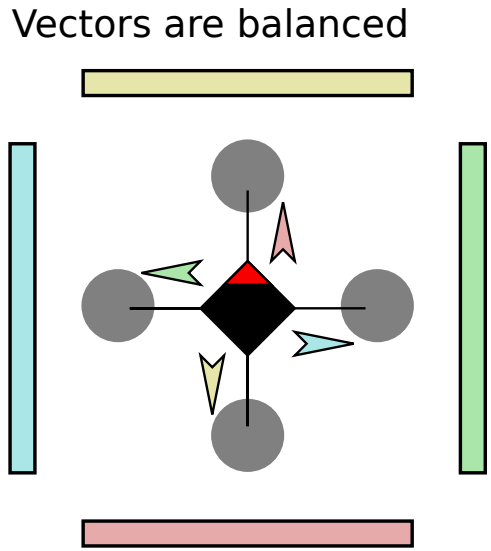
If no location service is present the robots start to drift and sooner or later they crash into an obstacle.

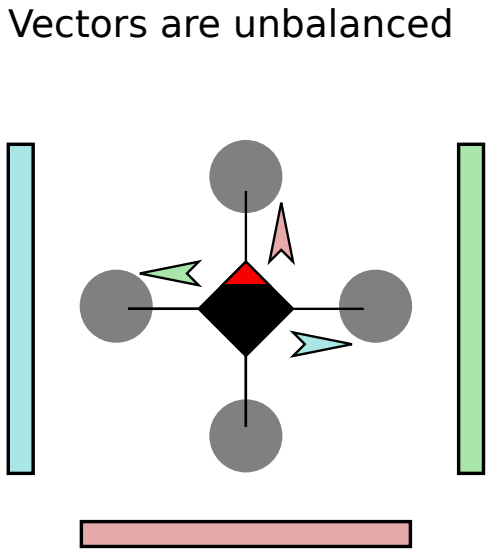

Quadrocopter
Seriell
I²C Bus
SunSpot
Sensor
Radio
SunSpot
Debug-PC
USB

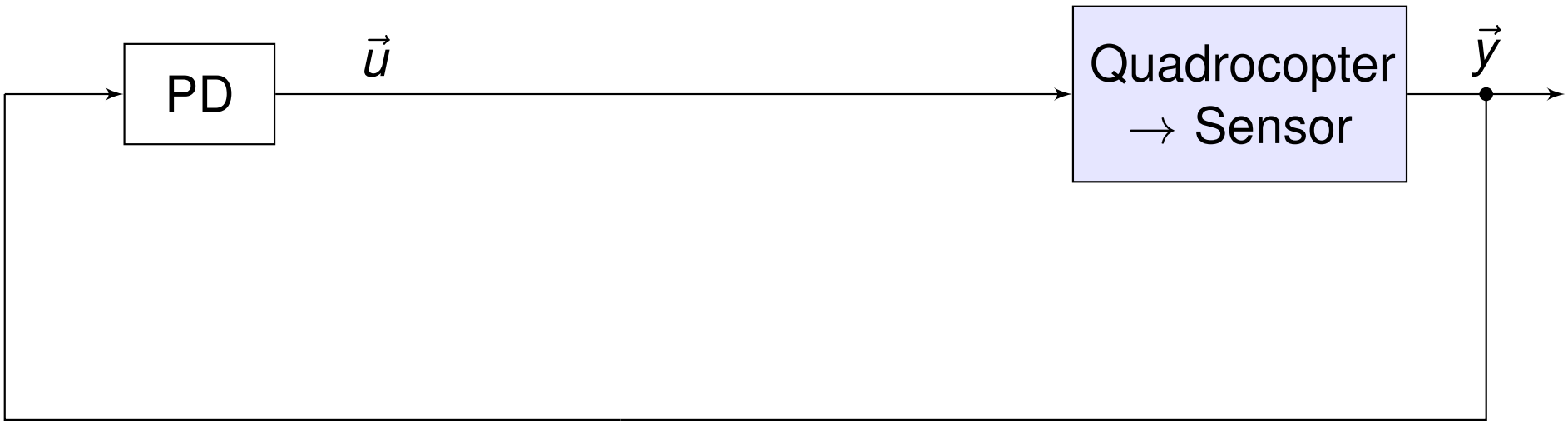
- Deriviation of an escape vector from many sensor readings
- Steering decision to avoid collision



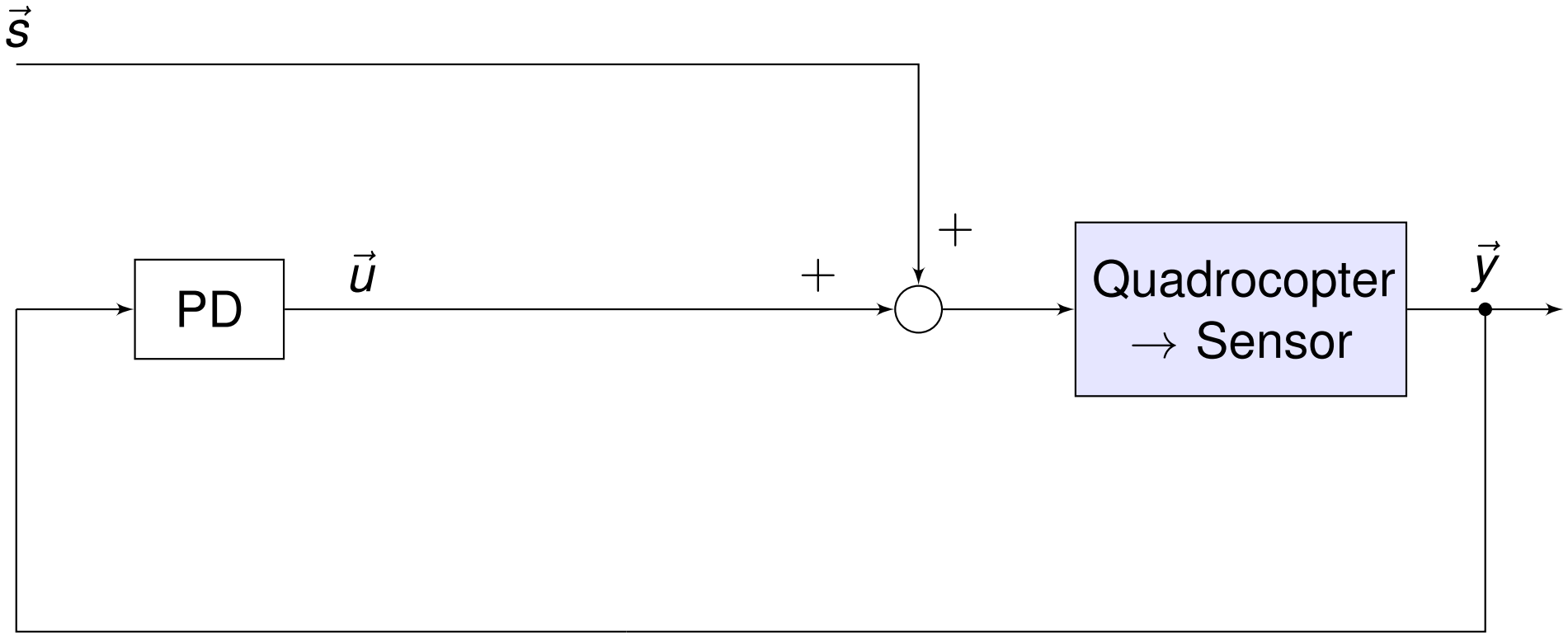
+Pitch
-Pitch
+Roll
-Roll



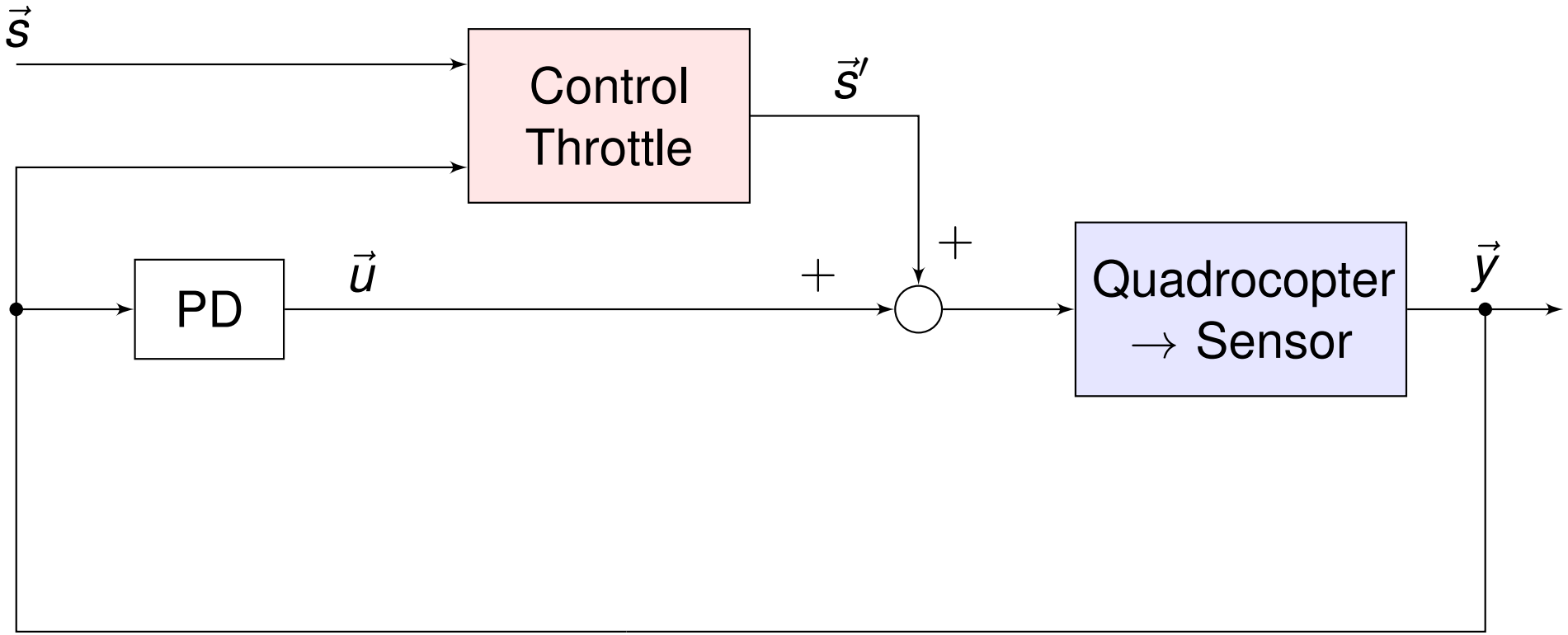




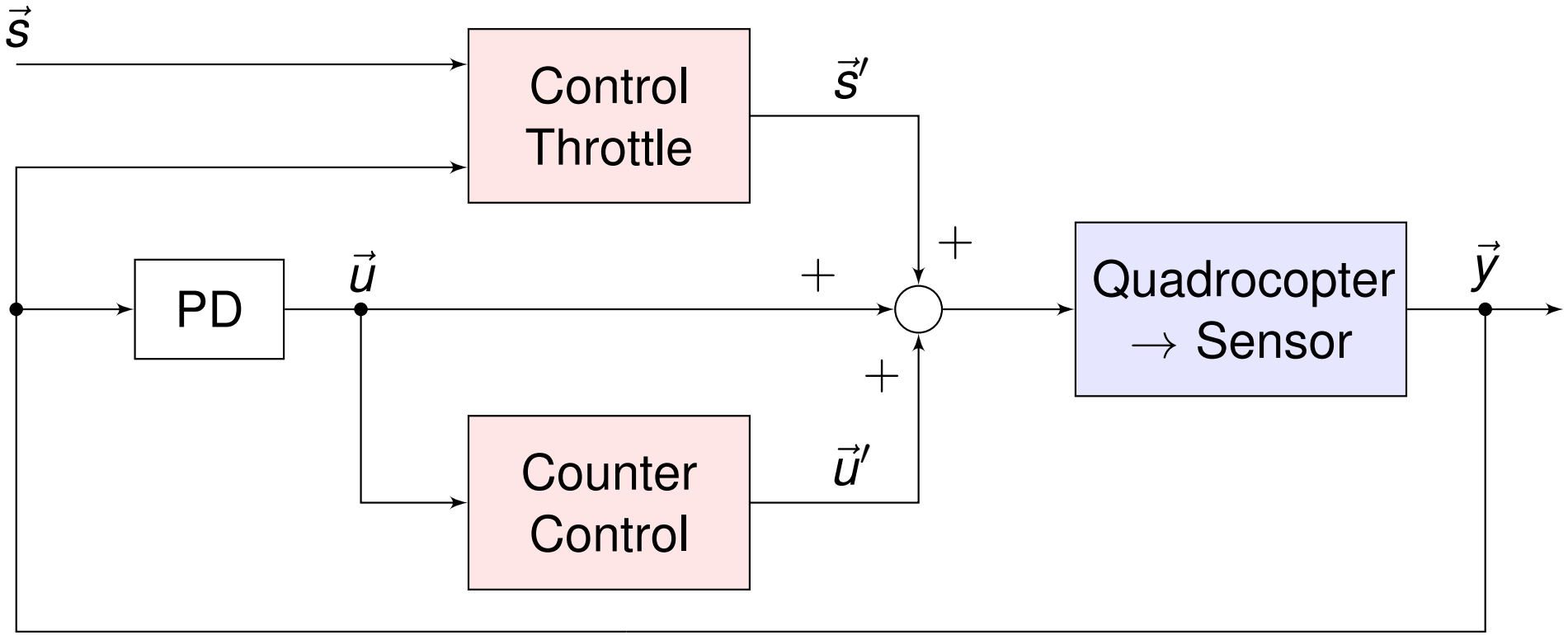
- Collision avoidance but non-maneuverable



- Collision avoidance
- Maneuverable but steering could draw out control loop



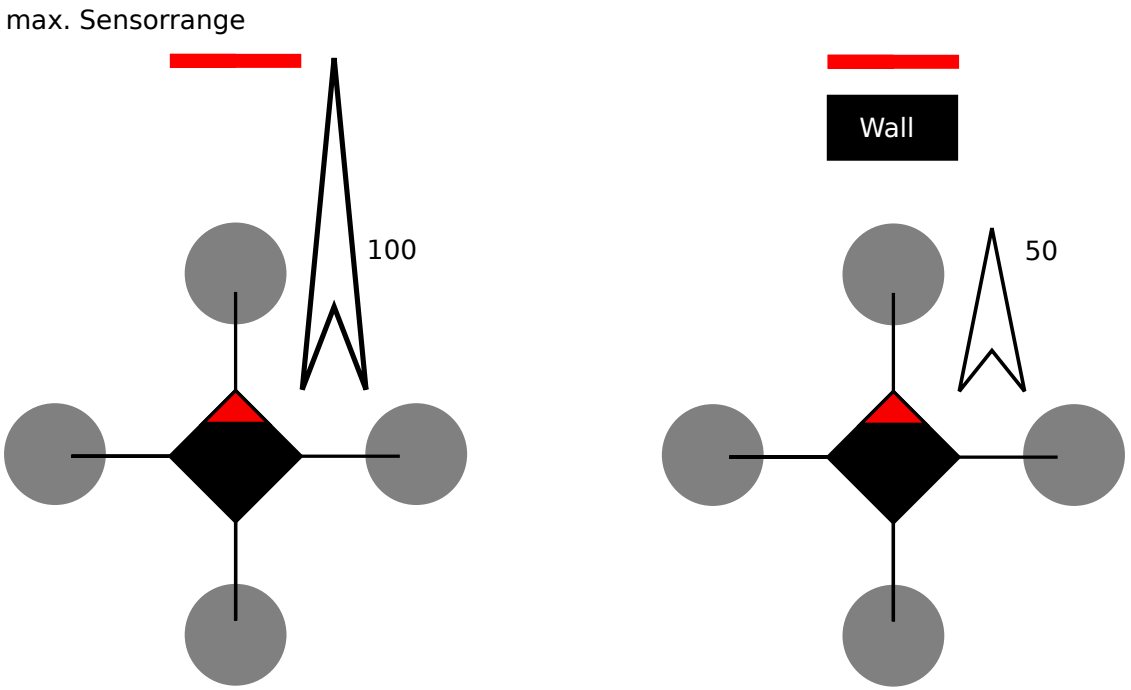
- Collision avoidance
- Maneuverable
- High velocity after the evasion

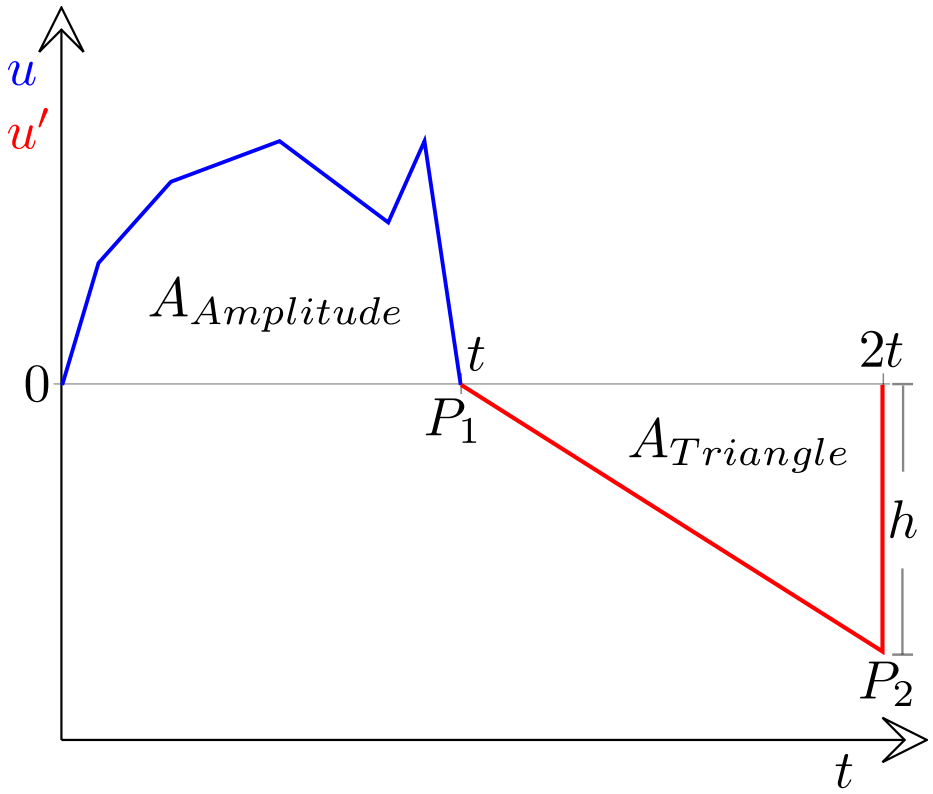


- Collision avoidance
- Maneuverable
- High velocity after the evasion prevented

Creation of an attenuation factor

$$\mu_{attenuation} = \frac{d_{measurement} - d_{min}}{d_{reaction} - d_{min}} \quad (1)$$





Area of the triangle:

$$A_{Triangle} = \frac{1}{2} t \cdot h \qquad (2)$$

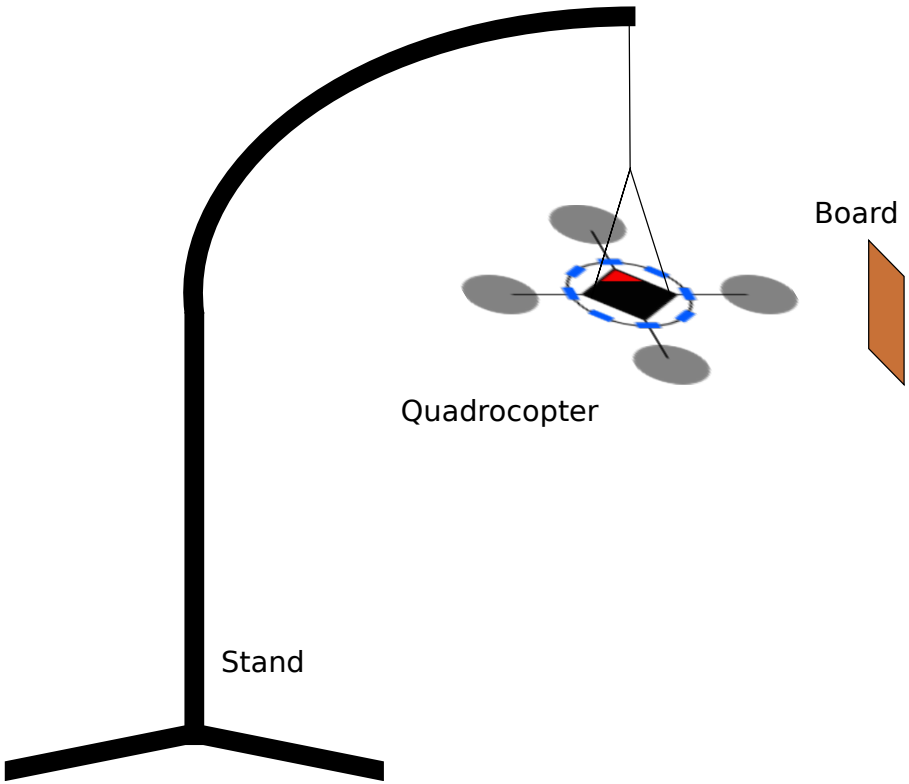
Area of the amplitude:

$$A_{Amplitude} = \int_0^t mx\,dx \qquad (3)$$

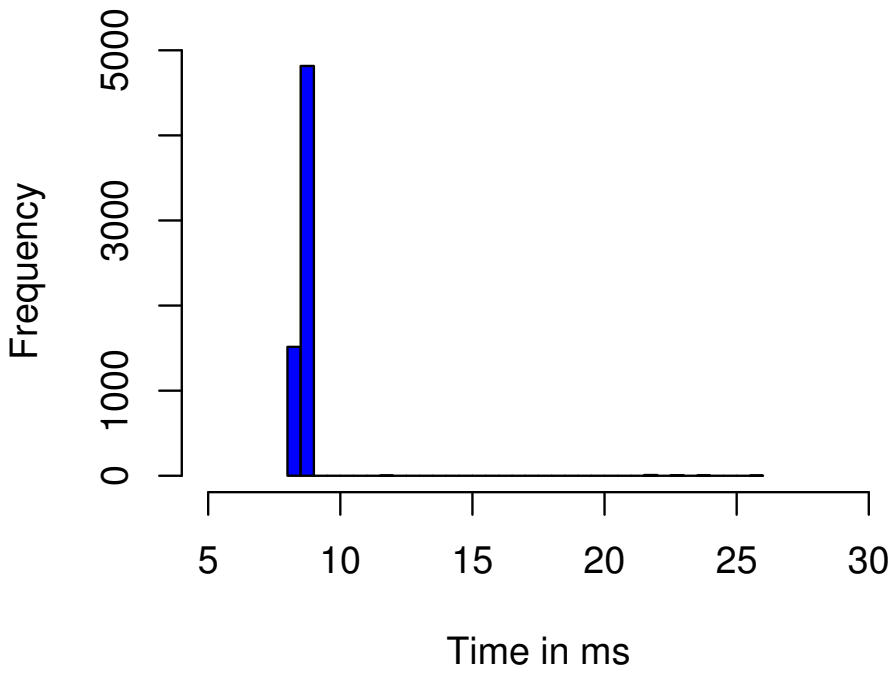
Velocity reduction:

$$f(x) = \frac{h}{t} \cdot x \qquad (4)$$

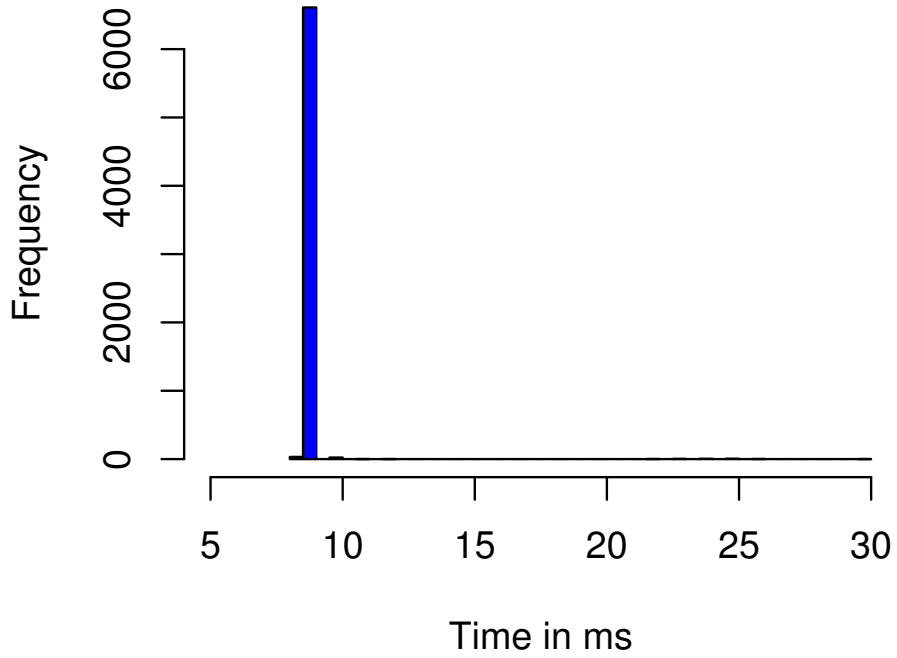
- Test stand: Observe controller behavior in different collision cases
- UfoView: Swing application to monitor I/O values of the robot



- TimerTask starts every 30 ms (scheduling precision 3ms)
- Cumputing duration per cycle 8 ms in general
- Up to 30 ms including a Java VM Garbage Collector cycle



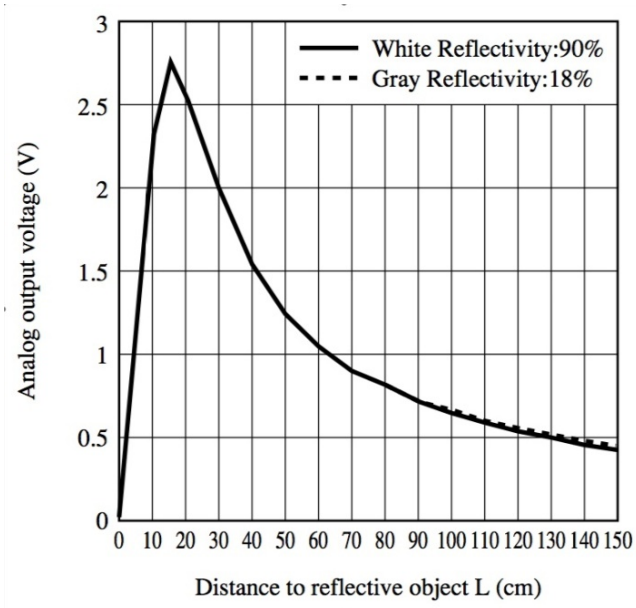
w/o Load

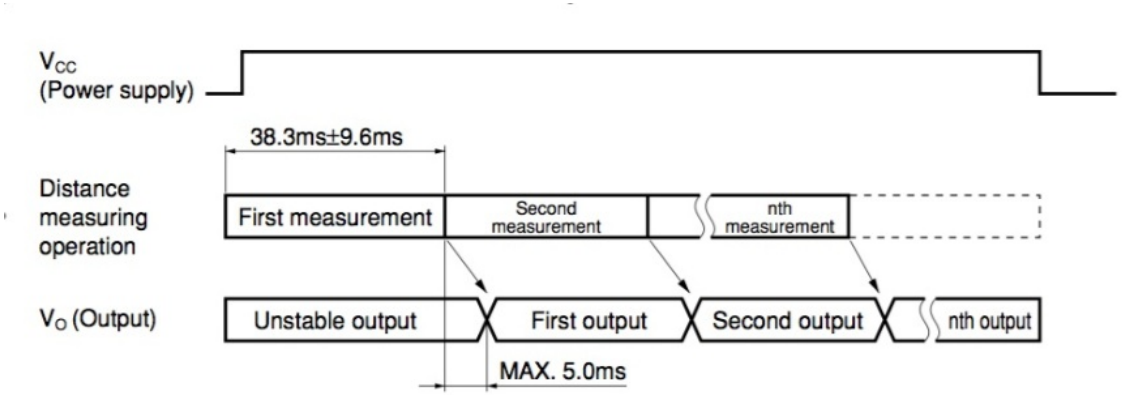


w/ Load

Characteristics of the Sensors:

- Actually too slow
- Distance greater than 70 cm result in impractical readings

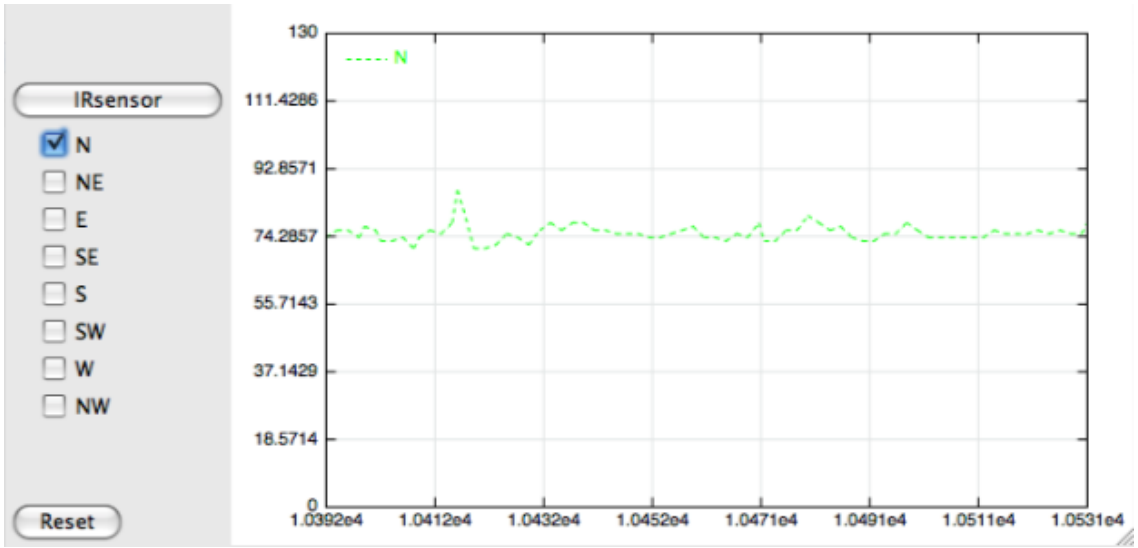




Characteristics of the Sensors:

- Actually too slow
- Distance greater than 70 cm result in impractical readings
- Different obstacle surfaces lead to wrong distance measurements (e.g. 40 cm are read as 75 cm)

⇒ Quadrocopter can only avoid collisions if its velocity is small

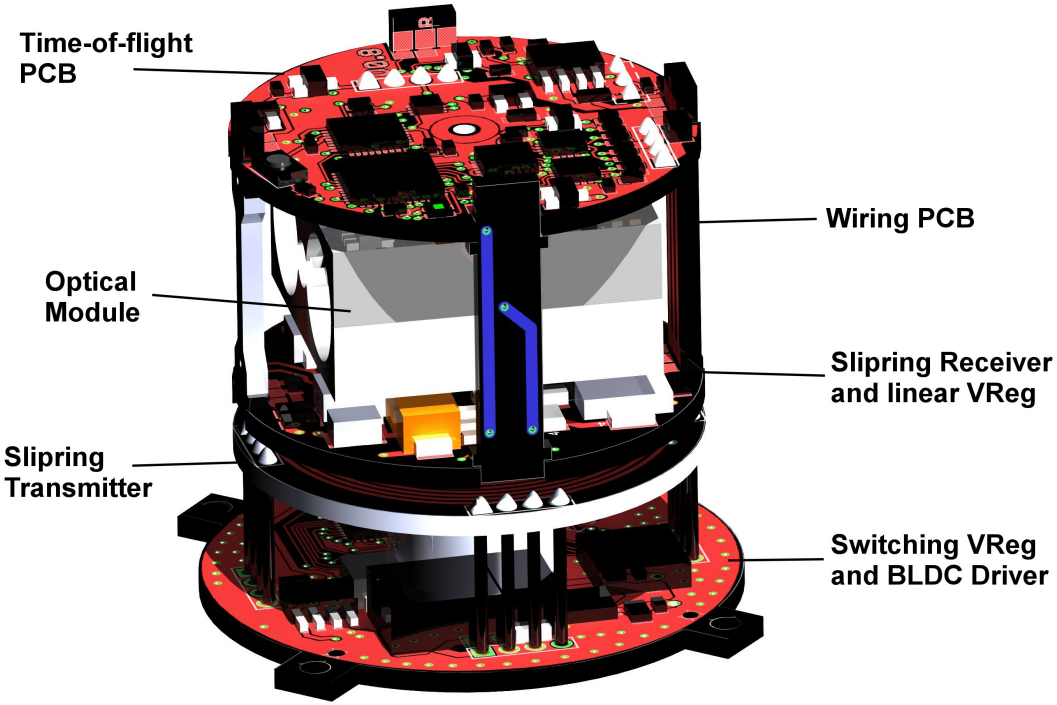


Watch out! – Ufos attacking

Live demo during the coffee break!
Now: Video

- Controller can be implemented on (almost) any low-cost platform
- $t_{Calculatation} + t_{GarbageCollector} < t_{SensorSample}$
  $\rightarrow$ So far Java VM is not a big problem (yet).
- Dynamic control intervals (by correlating measurements with the time) should solve future real-time issues
- Sensor unsuitable for this application. Should be faster and/or have a greater range

Laser Scanner



So far achieved properties:

- True 360° scanning
- Sampling rate $20kHz$
- Weight $\approx 40g$
- Distance range $\approx 10m$
- Accuracy $\pm 10cm$

Ein Regler zur Kollisionsvermeidung von Flugrobotern

Bernd Hügel, Jürgen Eckert, Reinhard German und Falko Dressler

Thanks!

juergen.eckert@informatik.uni-erlangen.de