Challenges in real-time application development – The I4*Copter* project

Invited talk

Computational Systems Group – University of Salzburg

13 May 2009

Peter Ulbrich

Chair in Distributed Systems and Operating Systems
Friedrich-Alexander University Erlangen-Nuremberg

`ulbrich@cs.fau.de`
`http://www4.informatik.uni-erlangen.de/~ulbrich`

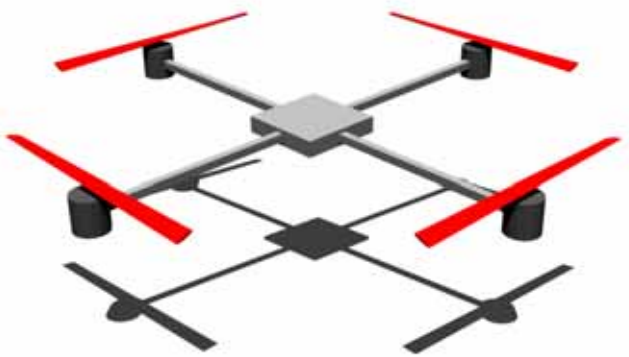
Chair in Distributed Systems and Operating Systems

Friedrich-Alexander-Universität Erlangen-Nürnberg

Motivation

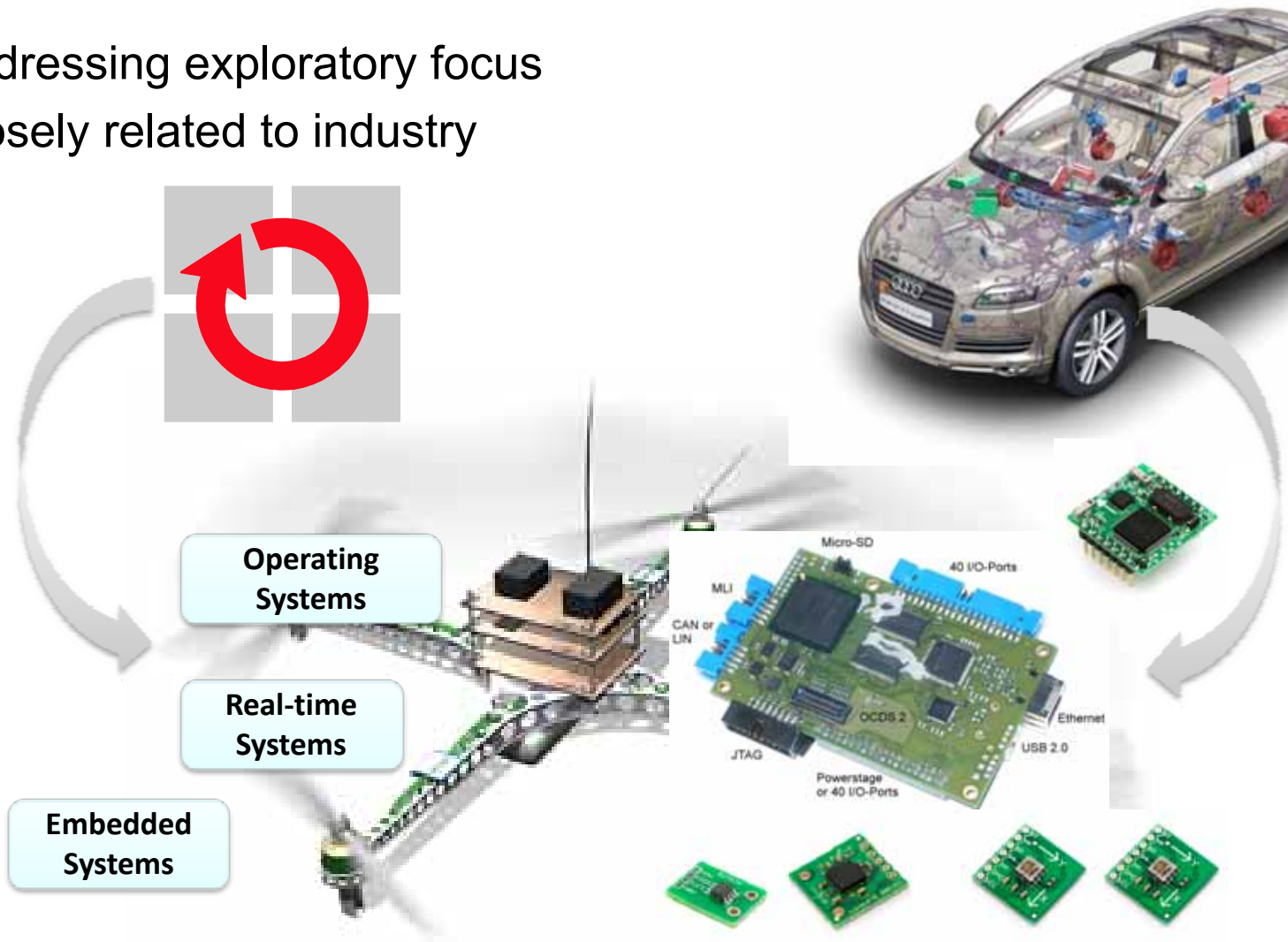
- **Showcase for embedded and real-time system software?**
- **Real-time system engineering**
  - Drawing conclusions from development process
- **System research and industry projects**
  - Creditable safety-critical application available
  - Research project evaluation
- **Teaching**
  - Comprehensive and demanding application
  - Cross-domain education

➔ **A quadrotor helicopter! (Quadrocopter)**



Requirements (1)

- Addressing exploratory focus
- Closely related to industry



Requirements (2)

- **Microcontroller → Infineon TriCore**
  - Widely used in automotive domain
  - Sufficient performance reserves (150MHz, 2MB Flash, 256KB RAM)
  - Substantial periphery support
- **Off-the-shelf sensors**
  - Heterogeneous communication type (analog, digital, bus)
  - Software signal processing and filtering

→ **No adequate construction set available on the open market!***

*at that time

Timeline

Late 2007

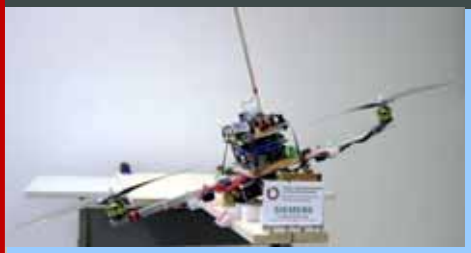
- A bagful of hardware
- First clumsy copter
- Incapable of flying

Timeline

Late 2007

Early 2008

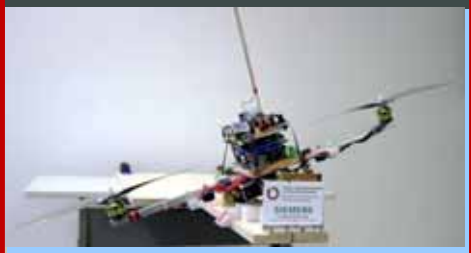
- Back to drawing-board
- 1-axis test rig
- Engine test rig



Timeline

Late 2007

Early 2008



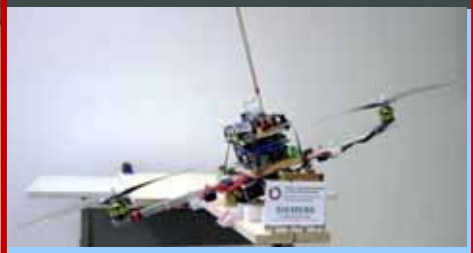
Mid 2008

- I4Copter Prototype V1.0
- First flight (Late 2008)
- I4Copter Protoype V1.1

Timeline

Late 2007

Early 2008



Mid 2008

Early 2009

- I4Copter Prototype V2.0
- Acceptable flight behaviour

Timeline

Late 2007 | **Early 2008** | **Mid 2008** | **Early 2009**

Why did it take so long?

Outline

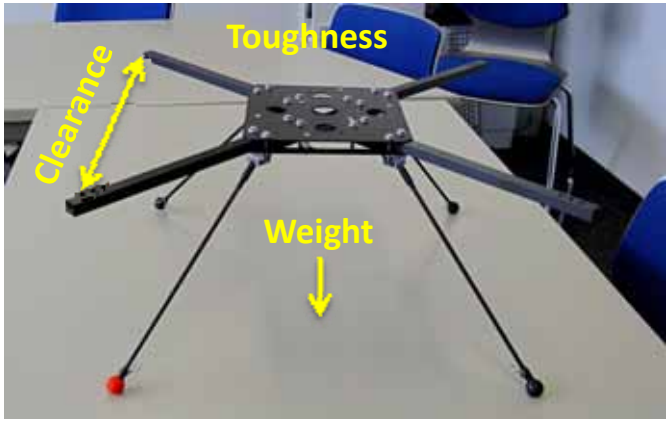
- **Building the quadrocopter**
  - Prototype development
- **Real-time application analysis and design**
  - Physical model
  - Real-time system
- **System implementation**
  - Component design
  - Loose coupling
- **Lessons learned and conclusion**

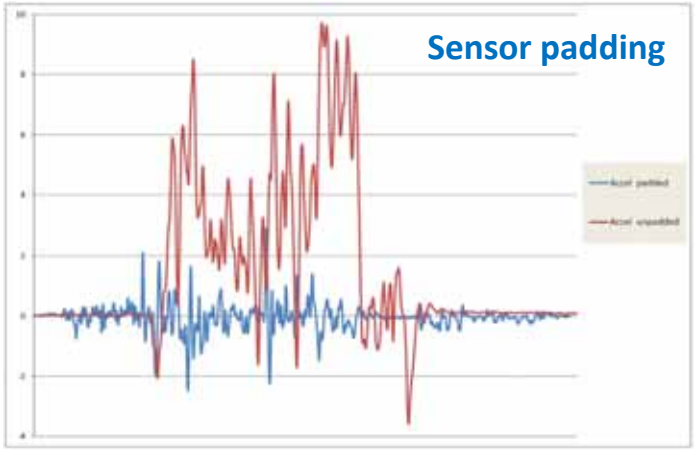
Building the quadrocopter

System complexity

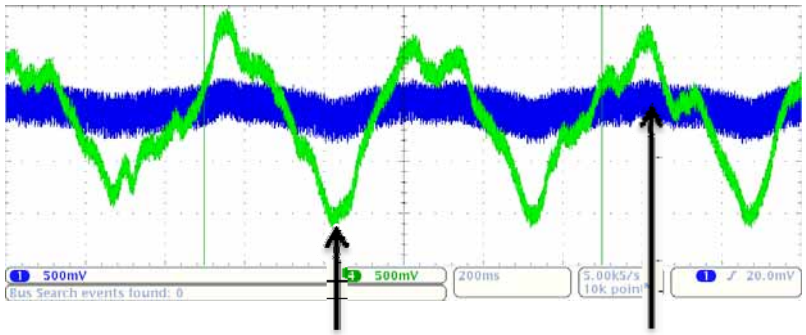
- A quadrocopter is highly complex system (in every sense)
  - Beyond the domain of computer science and automation control
- Simply the construction took months:

Manufacturing





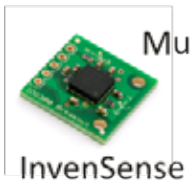
Electrical engineering



The I4*Copter* prototype v2

- 3rd Iteration: Prototype „Apollo“

Facts


Murata ENC-03

InvenSense IDG-300

2x

AnalogDevices ADXL202


Graupner R700

Gyroscope

Accelerometer

Remote

Lantronix WiPort

2x

Sharp GP2Y0A02

Proximity

GPTA

ADC

Microcontroller
TriCore TC1796

Ethernet

WLAN

GPIO

SPI


VTI SCP1000

Safety switch

&

Pressure

current

4x

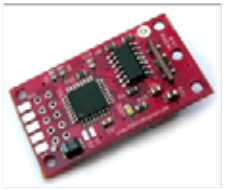
Mikrokopter BL-Ctrl

GPIO

SPI

I²C

Microcontroller
ATMega168

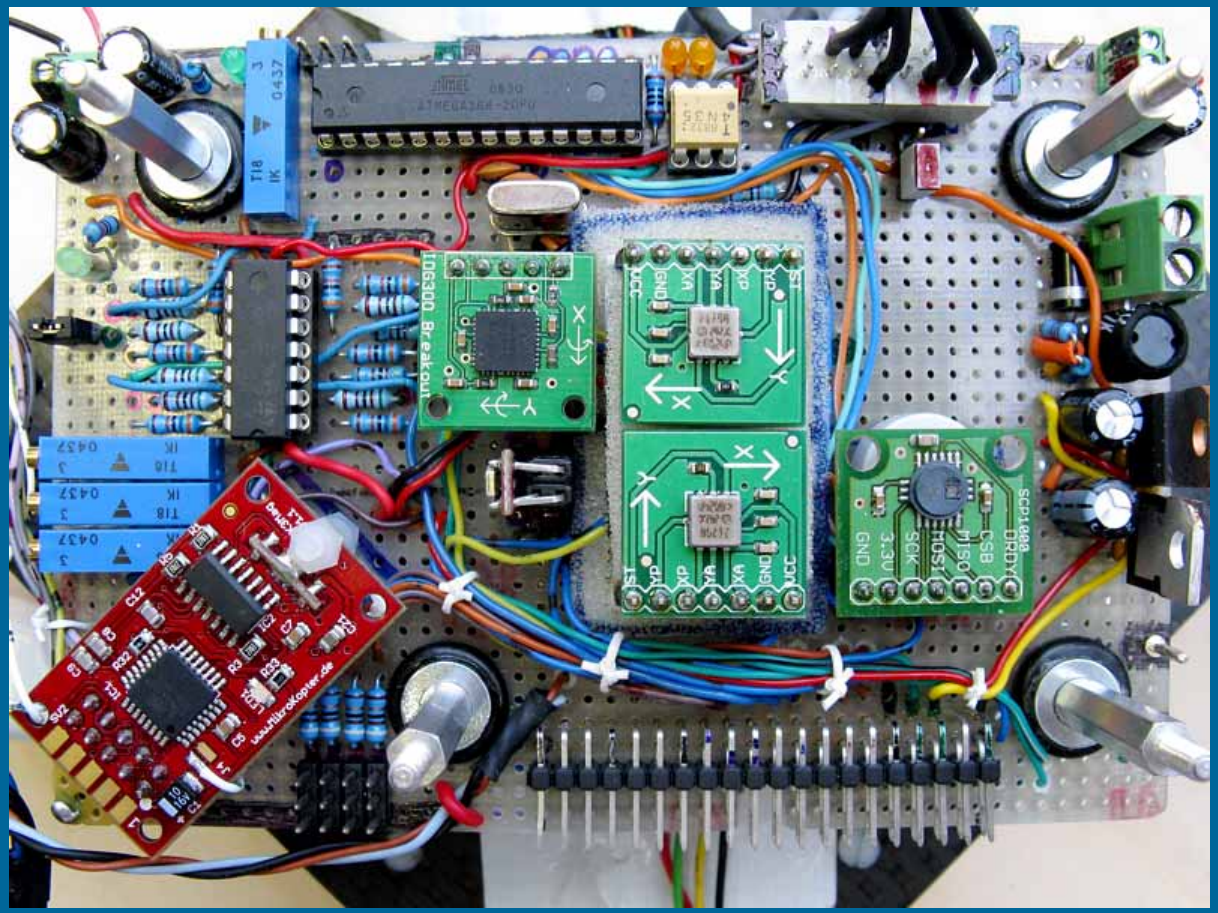

Mikrokopter MK3Mag

Engine controller

Compass

Facts

Prototype periphery board – Mark II



Real-time application analysis and design

Application Requirements (Excerpt)

- **Goal:** semi-autonomus flight
  - Safe **hovering** (maintain position, heading and height)
  - **Steering** by remote and/or WLAN
  - Support by automatic **take off** & **touch down**
  - Heading for waypoints – **Autopilot**

- **Allocation**
  - *Behaviour engine* – firm real-time
  - *Attitude control* – hard real-time

Application analysis

- **Relationship** between **Event** and **Result**
  - **Temporal** – Time allowed to pass → Deadline
  - **Physical** – Way of determing the result
- **Physical object**
  - Relevant parameters and their connection?
- **Real-time system**
  - Events to be handled? Deadlines?
  - Relationship: Deadline ↔ Physical object
- **Physical model**
  - Parameters to be mapped?
  - How to map parameters?

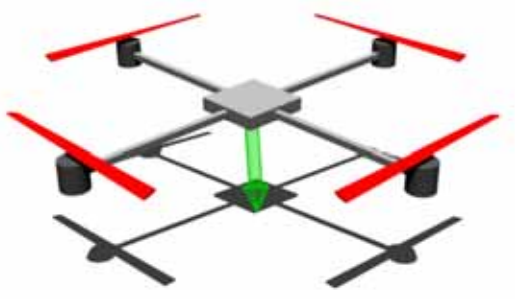
➔ **Is it possible to reduce the model to simple state observance?**

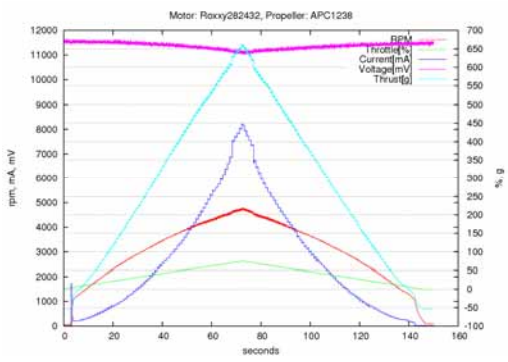
Quadrocopter analysis

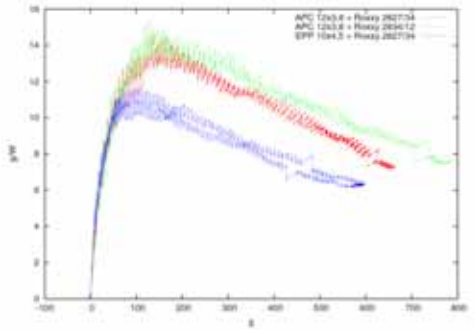
- State is **not** fully observable but calculable → **control engineering**
- **Observation**
  - **Angular rate** ω and **angle** φ of X,Y and Z-axis
- **Manipulation**
  - **Thrust** generated by the engines
- **Response**
  - **Change of position**, depending on the objects momenta (mass, inertia)
  - and the engine / airscrew (friction, inertia, efficiency)
- System model describes the correlation between **observable**, **calculable** and **manipulable** parameters

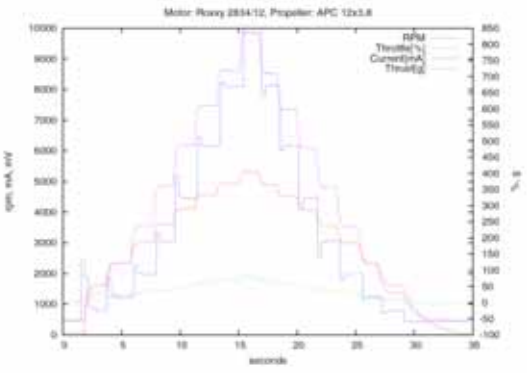
Physical parameters

- **Determining by measurement**
  - e.g. thrust, power consumption, voltage, weight
- **Derivation of parameters**
  - e.g. inertia, efficiency
- **Examples:**
  - Moment of inertia: 37,74 m²g
  - Engine response time: ~160ms (66% nominal)









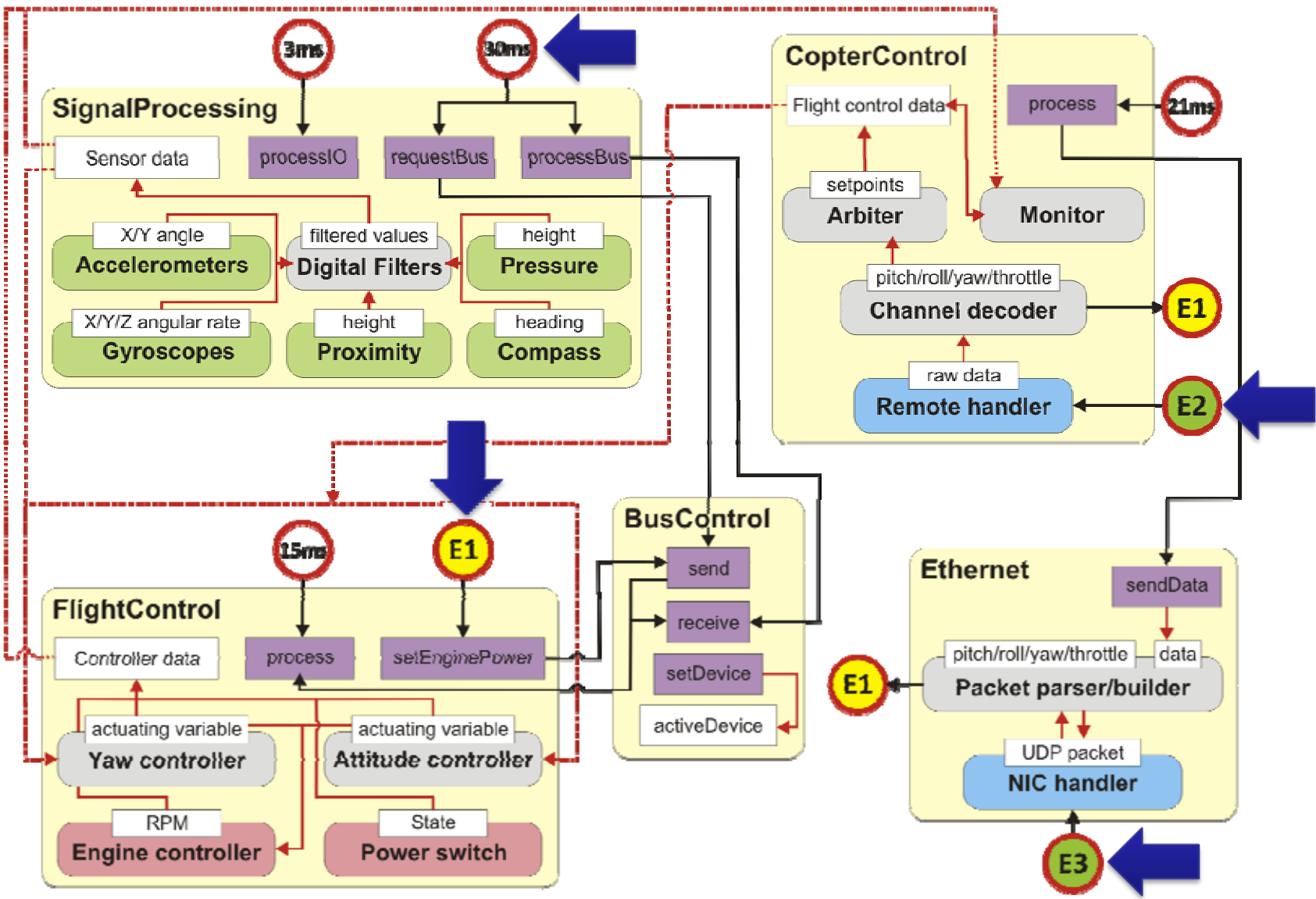
Real-time system - Events

- **Signal processing** → **periodical** – 3ms / 30ms
  - 2x oversampling (sampling theorem)
- **Flight control** → **periodical** – 15ms
  - 10x compared to engine response time (school of thought)
- **Monitoring** → **periodical** – 25ms
  - 10x compared to object inerta (school of thought)
- **Command** → **aperiodical** – 20..250ms
  - 2x oversampling, depending on human response time and object inertia

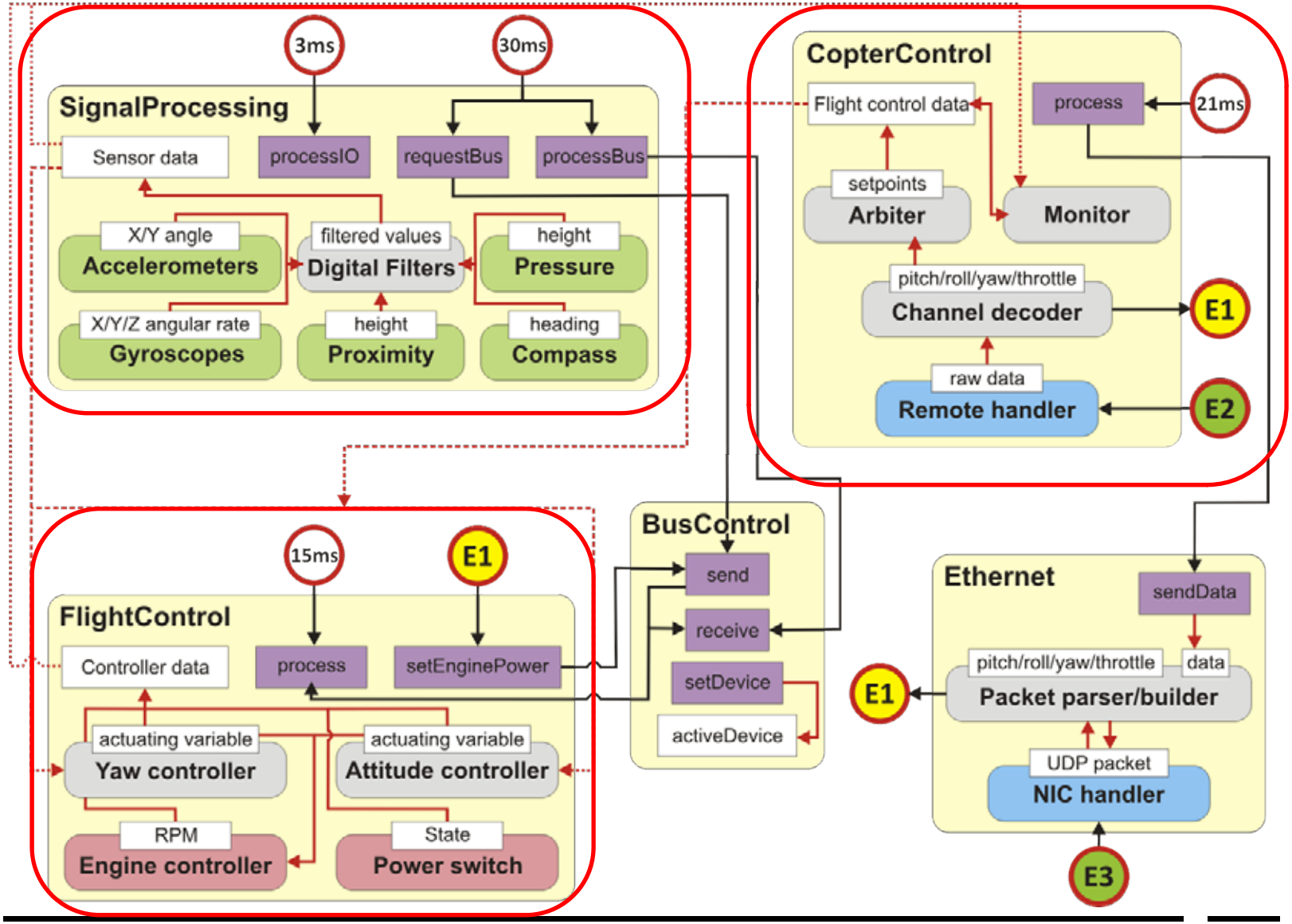
→ **50% of events depend on physical properties**

System implementation

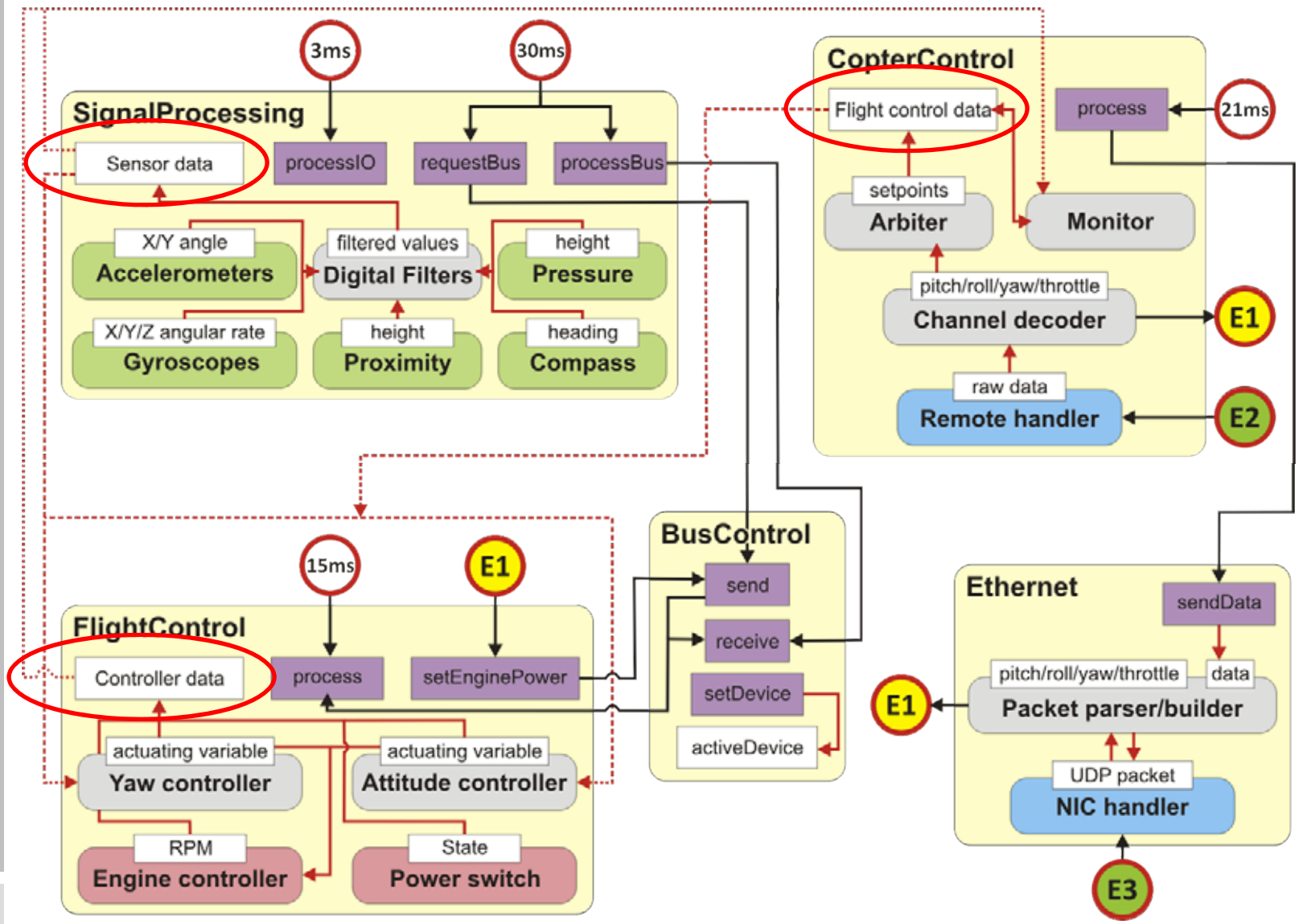
System overview



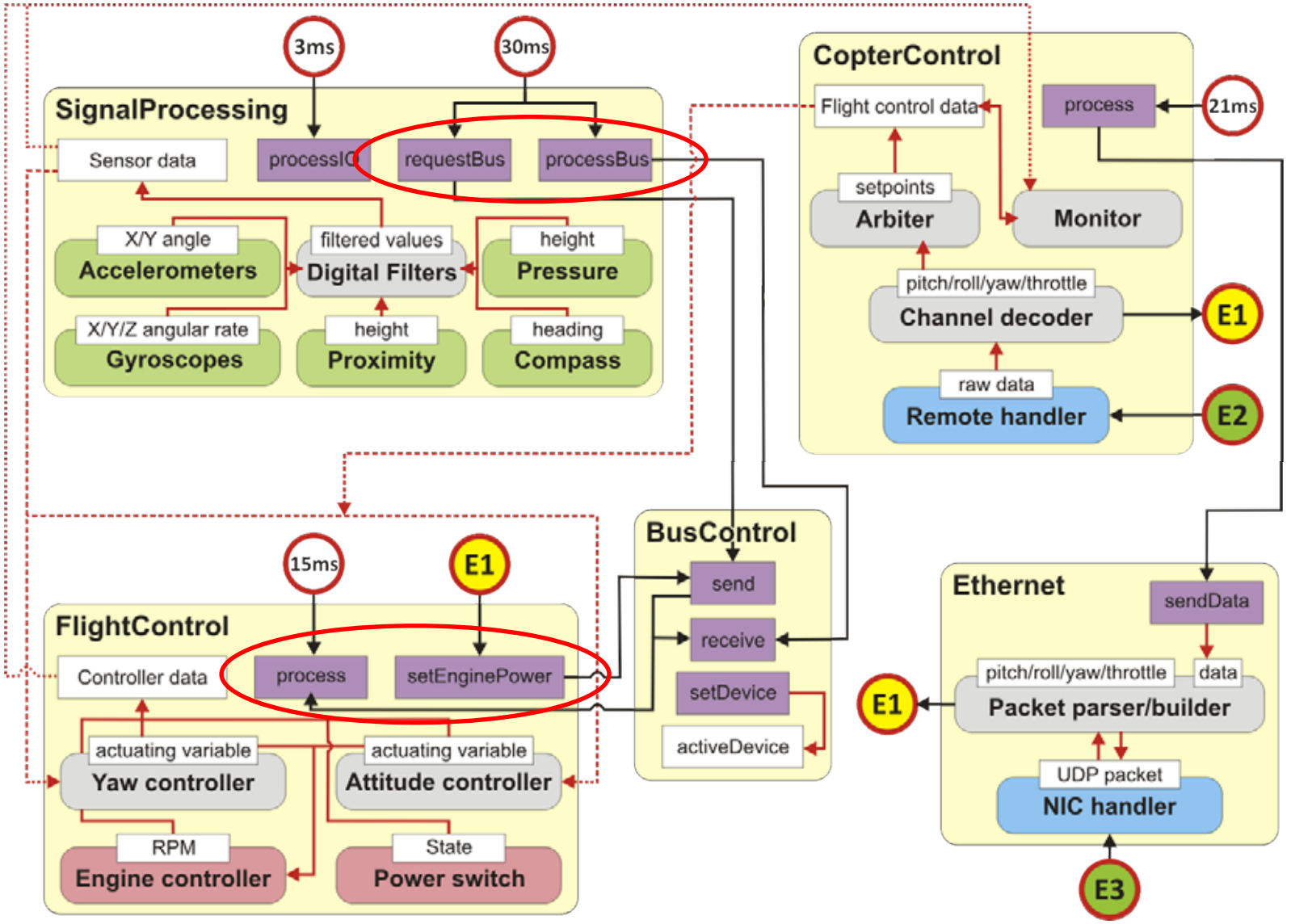
System overview



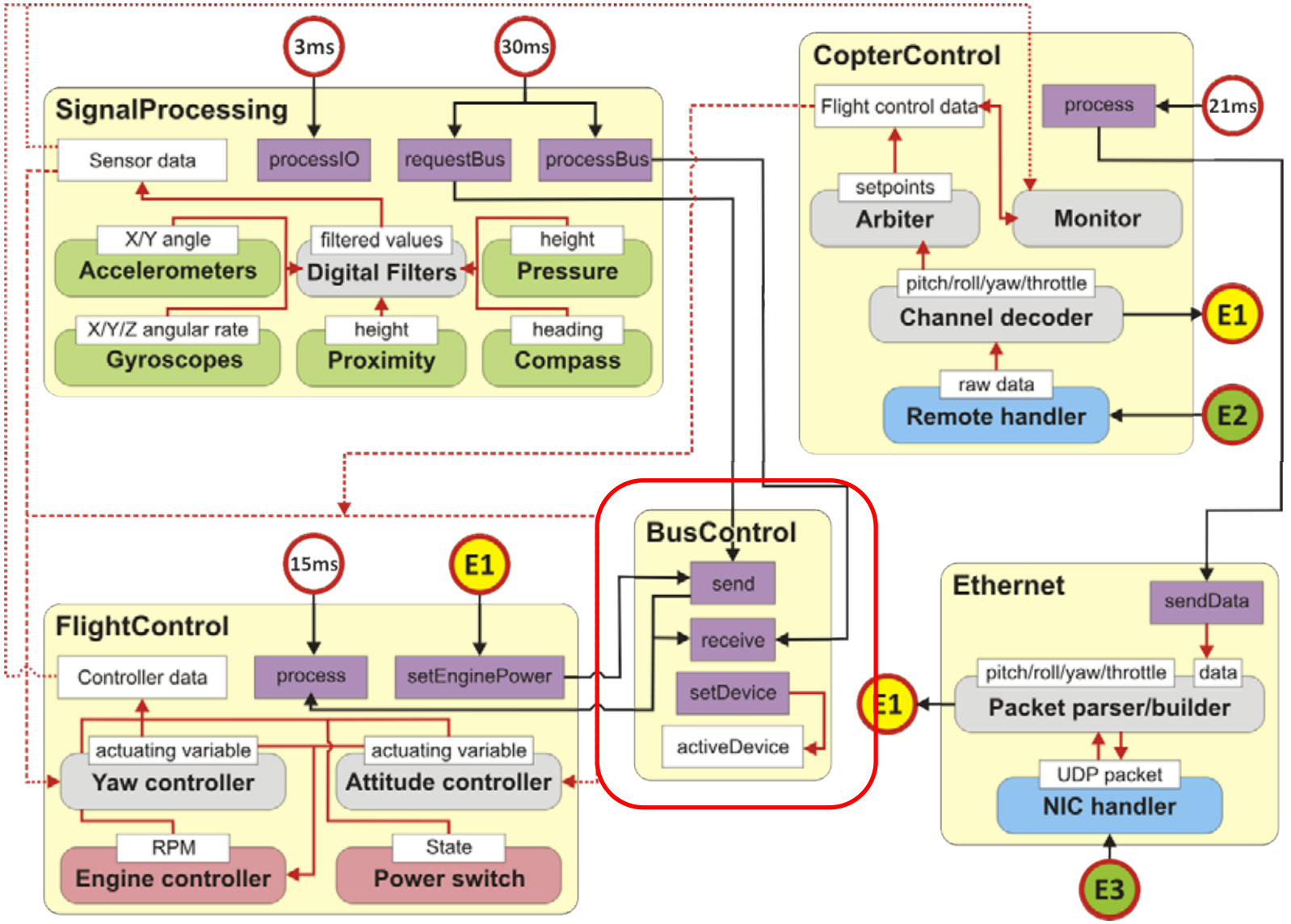
System overview



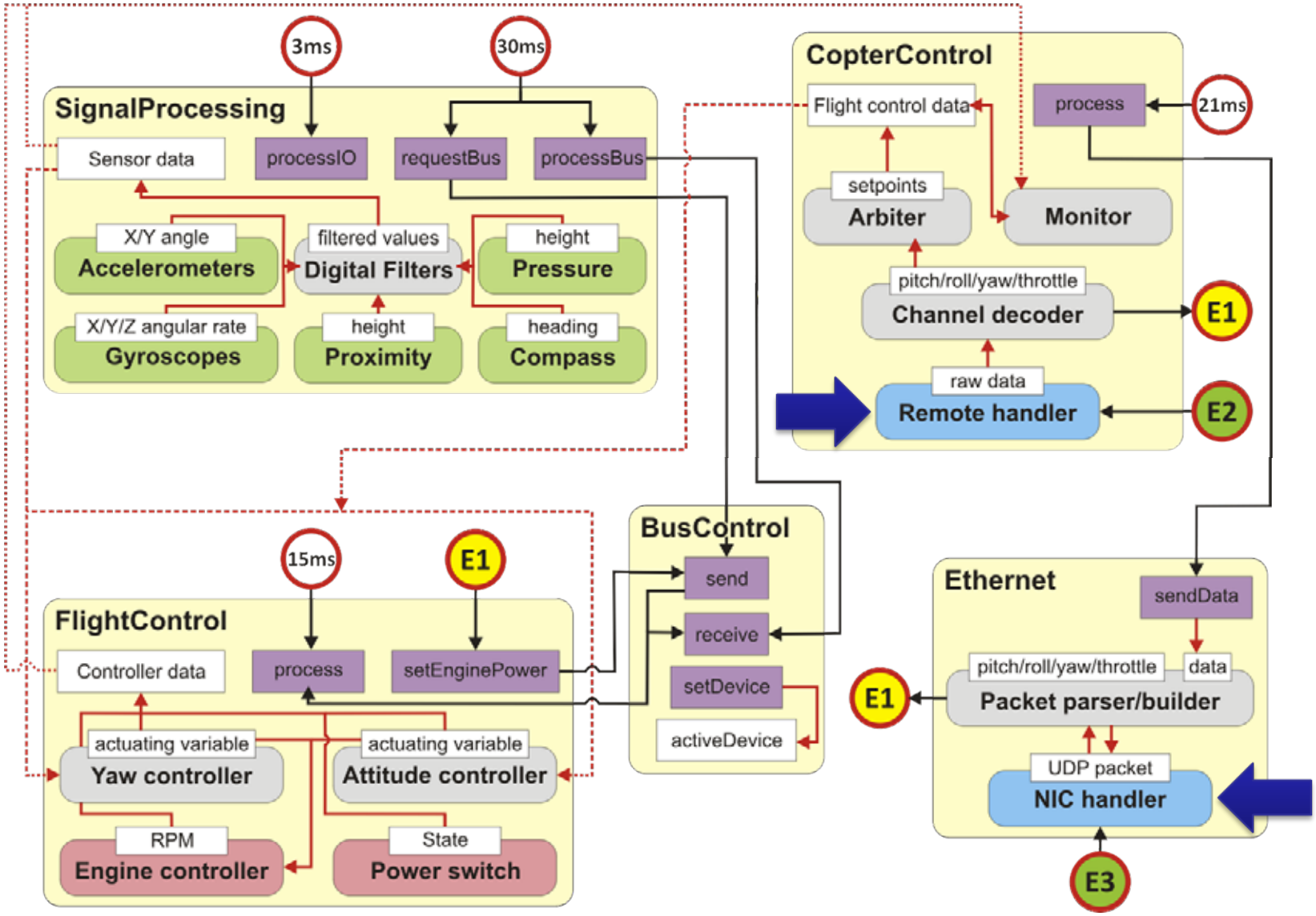
System overview



System overview

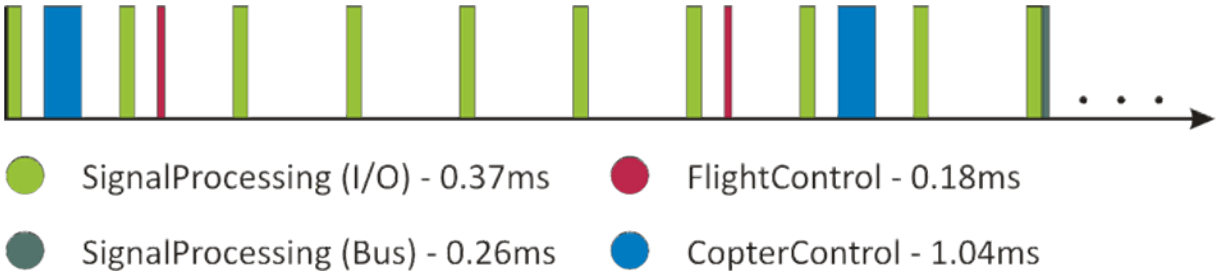


System overview



Facts

- **Static schedule**
  - Interrupts: min. interarrival time known
  - Based on application and WCET analysis



- **Using PxROS-HR**
  - Priority based RTOS
  - Implemented using programmable timer

Lessons learned and conclusion

Lessons learned

- **A quadrocopter is a unforgiving system**
  - Apparent procedures are physically complex
  - Unobservable parameters have severe impact on the system
  - Control engineering necessary
- **Implementing a real-time application requires precise analysis**
  - Modularisation depending on application design
  - Aim loose coupling (data flow vs. control flow)
- **Building a real-time system requires familiarity with physical object**
  - Physical parameters have impact on **events** and **deadlines**
  - One has to see beyond the own domain

Conclusion

- Designing and building a quadrocopter from scratch is challenging
  - Beyond the domain of computer science
  - Electrical engineering, manufacturing, control engineering
  - Real interdisciplinary project

Conclusion

- Designing and building a quadrocopter from scratch is challenging
  - Beyond the domain of computer science
  - Electrical engineering, manufacturing, control engineering
  - Real interdisciplinary project
- The I4*Copter* is a creditable demonstrator for safety-critical mission scenarios
  - A hard real-time system
  - Demanding application for the underlying system software
- It is perfectly suited for teaching and attracting students
  - Various theses
  - „Real-time system lab“ experiment

Thank you for your attention!

Questions?