Monitoring environment-parameters for research towards energy-efficient buildings

Speaker's Intro

Speaker

- Bart Voet

Day Job

- Practice Lead Java Development at AXA

Evening and weekend

- Family
- Programming and hacking
- Learning electronics
- Skateboarding and snowboarding
- ...

Stakeholders

Division of Building Physics KUL
(Department Civil Engineering)

- System capturing data from digital sensors

- Scalable to different scenario's
  - Research facility
  - Educational environment
  - External research

Stakeholders

Primary stakeholder

- **Division of Building Physics KUL** (Department Civil Engineering)
  - System capturing data from digital sensors

  - Scalable to different scenario's
    - Research facility
    - Educational environment
    - External research

Stakeholders

Stakeholder

- **Groep T**
  - Evolution
    - Smart sensors
    - Digital interfaces
    - Sensor networking
  - New devices (and open source)
    - Rapsberry Pi, BeagleBoard, Cubieboard
    - AVR, Arduino, ...
    - ...

Stakeholders

Stakeholder

- **Author (and other hobbyist)**
  - Learning platform
    - Experienced Java Developer
    - Learning embedded development
  - Open source platform
    - Scalable to different devices
    - Focused on monitoring
    - Integrable in different scenario's

Startup requirements

Context

Department of Building Physics KUL

performing

research on energy **efficient buildings**

Startup requirements

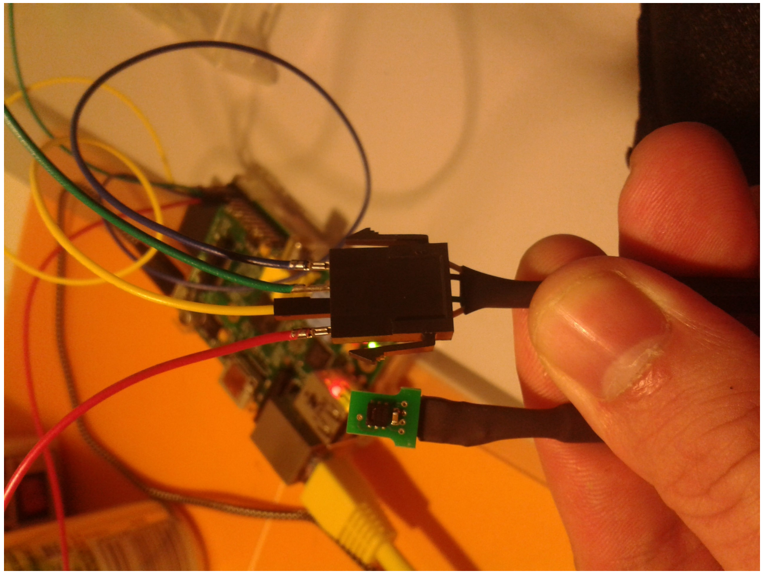
Demand

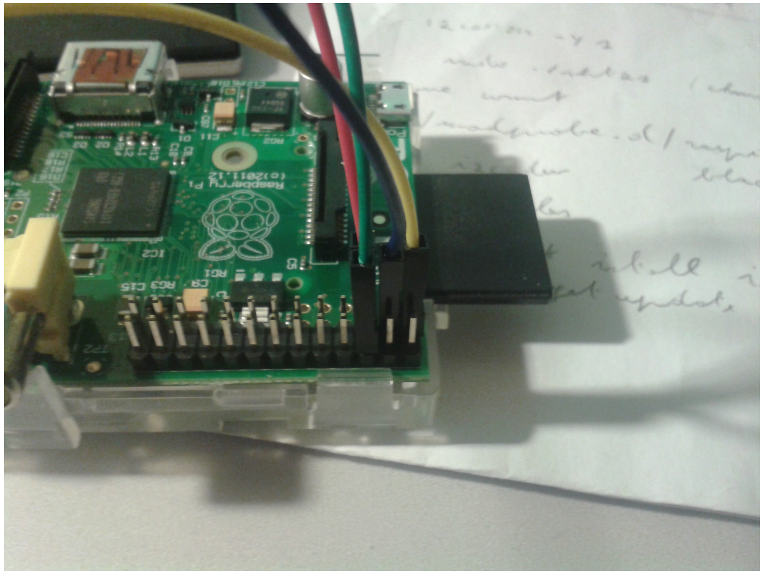
- System(s) for continuous measurement that is
  - Reliable
  - Inexpensive system(s)
  - Continuous measurement

ACE · GROEP T
ANTICIPATIVE CONTINUING EDUCATION

Startup requirements

Use **Raspberry Pi** as a device for **sensing**





Startup requirements

Use Raspberry Pi as a device for **sensing** environment **parameters**

- Important for
  - Indoor climate
  - Energy consumption
- Example given
  - Relative humidity
  - Temperature
  - Differential pressure
- … and **other** measurements in the future

Startup requirements

More specifically, use Raspberry Pi as a device to

- **Control** and **configure** sensors
- **Collect** data
- **Store** sensor measurements
  - (for later evaluation and analysis)
- **Correlate** stored measurements
  - configuration
  - timing

Startup requirements

Using sensors (Sensirion)

- SHT21 (STS21-SHT25)
  - temperature
  - relative humidity

- SDP600 (later phase)
  - differential pressure

Startup requirements

Taking into account profile of the users

- Students
- Researchers

Assuming only basic knowledge of

- Electronics
- Command line
- High level programming construct

(but not advanced)

Scope and position

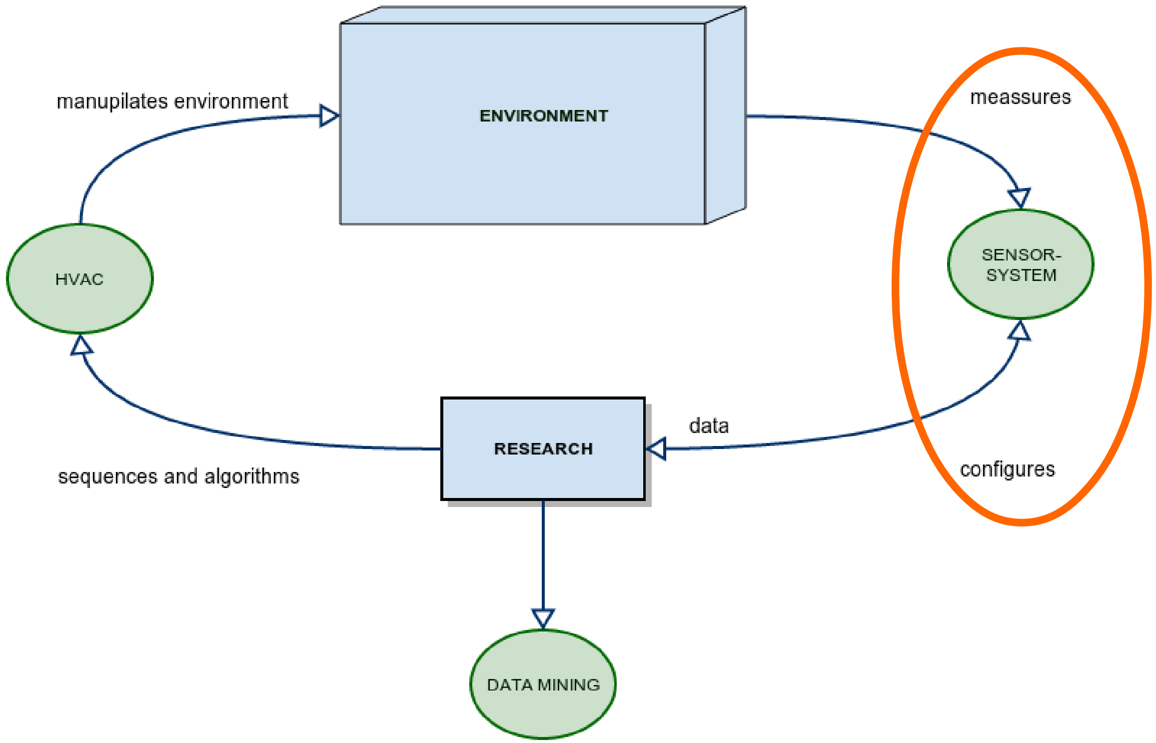
- Scope limited to **digital** (**smart**) sensors
  - Digital interface (i2c, spi or custom)
  - Integrated MCU performing
    - Calibration
    - Linearization
- No focus on **classic sensors**
  - Manual calibration
  - Precision resistors
  - ...

Scope and position

- Consequence
  - Focus on **system** (not hardware design)
    - Integration
    - Extensibility
    - Ease of use
    - Reliability
    - Documentation
    - ...

Context

Research-process and its components



Scenario's

Primary scenario: Research-facility in Gent

- HVAC-infrastructure deployed
- Electricity and ethernet

- Deploy, install and configure sensors
- Aggregate data

Scenario's

Scenario: Educational environment (students)

- Class-room environment
- Labo

- Experiment and learn
- Explore sensors

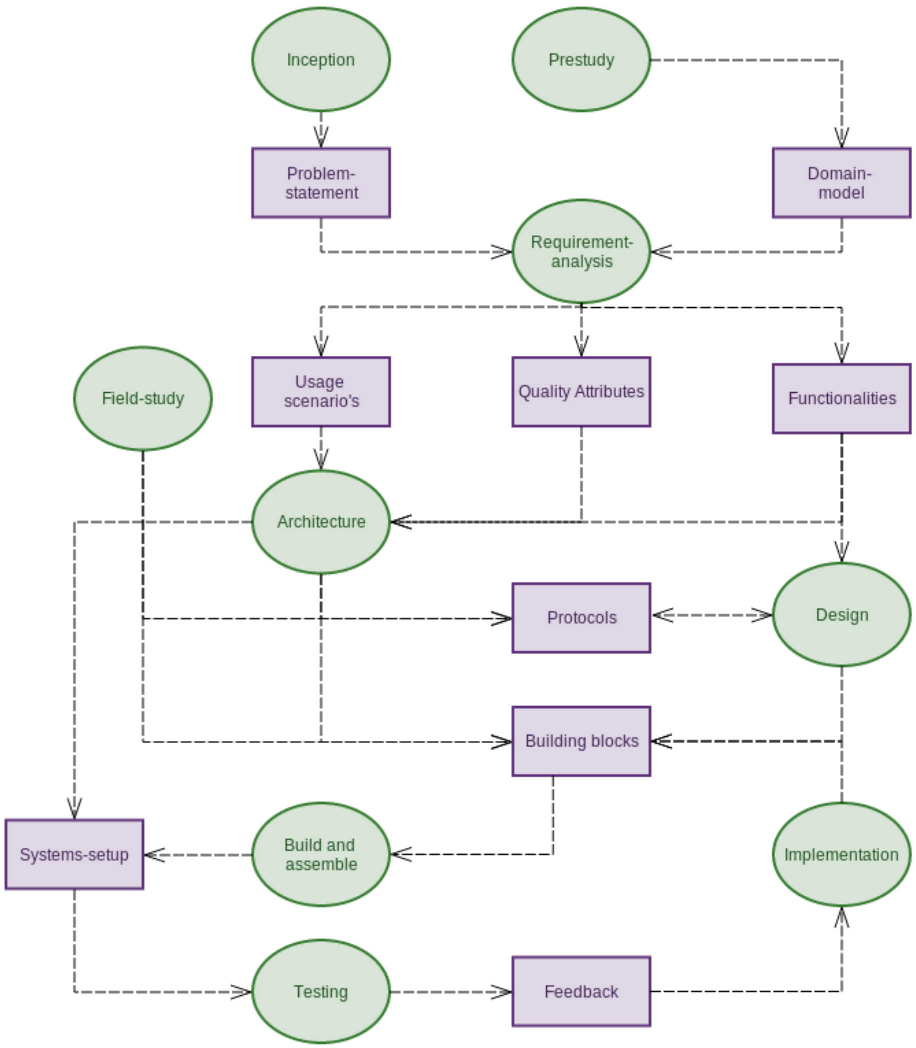
Scenario's

Scenario: Large buildings

- Mobile scenario
- No HVAC
- No assumptions on
  - Electricity
  - Network

- Similar to primary scenario
- More constrained environment

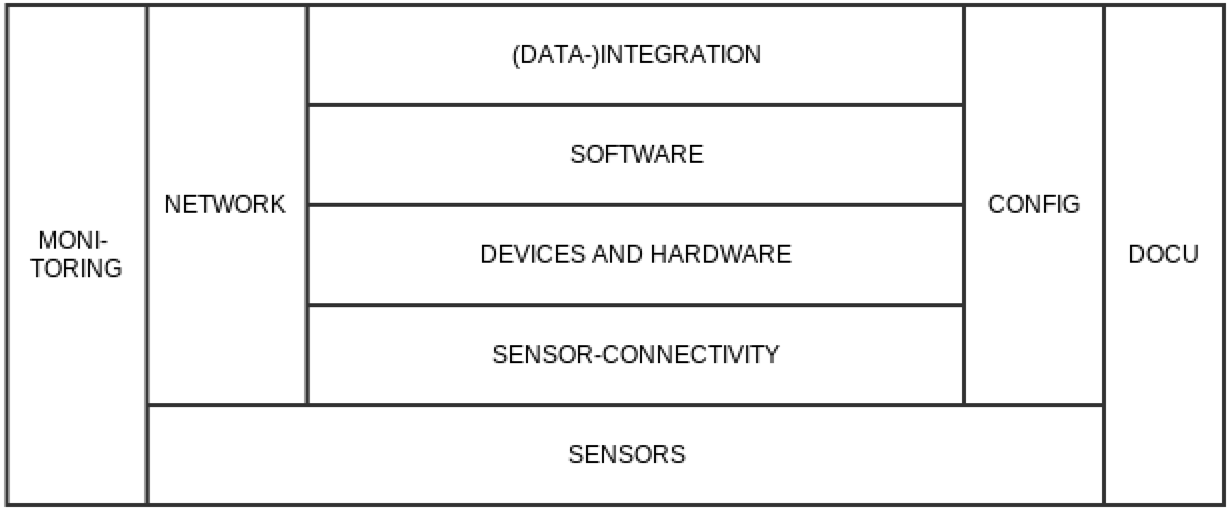
Approach


Inception
Prestudy
Problem-statement
Domain-model
Requirement-analysis
Field-study
Usage scenario's
Quality Attributes
Functionalities
Architecture
Protocols
Design
Building blocks
Systems-setup
Build and assemble
Implementation
Testing
Feedback

Approach

Category of building blocks (solutions)

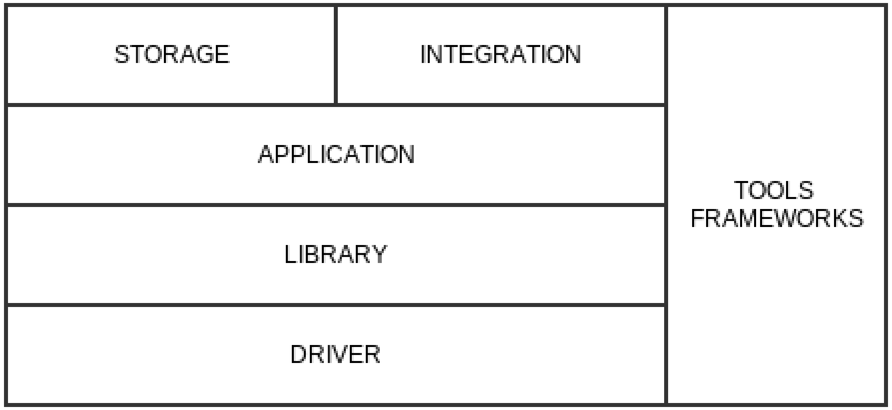
- Hardware, software, documentation
- Serve as annotation in thesis



Approach

Category of solutions

- Hardware, software, documentation
- Serve as annotation in thesis



System Challenges

TOP 5 Challenges

Multiple sensors

Concurrent access

e.g. Sensirion-sensors having same i2c-adress

Large area's

i2c and spi not developed
for long distance
(even if you lower the clock)

System Challenges

TOP 5 Challenges

- **Reliability**, **durability** and **resilience**
  - Ability to recover from
    - Power interruption
    - Network incidents **...**
  - Alerting-capability
    - Sensor goes down
    - Errors coming from sensors
    - Processing device not working
    - Heating **...**

System Challenges

TOP 5 Challenges

- **Usability**
  - Scalable to different scenario's
  - Users are no software- or hardware-engineers
  - Need an interface that's
    - Easy to integrate with other systems
    - Easy to integrate in personal computing
    - (structured txt-files)

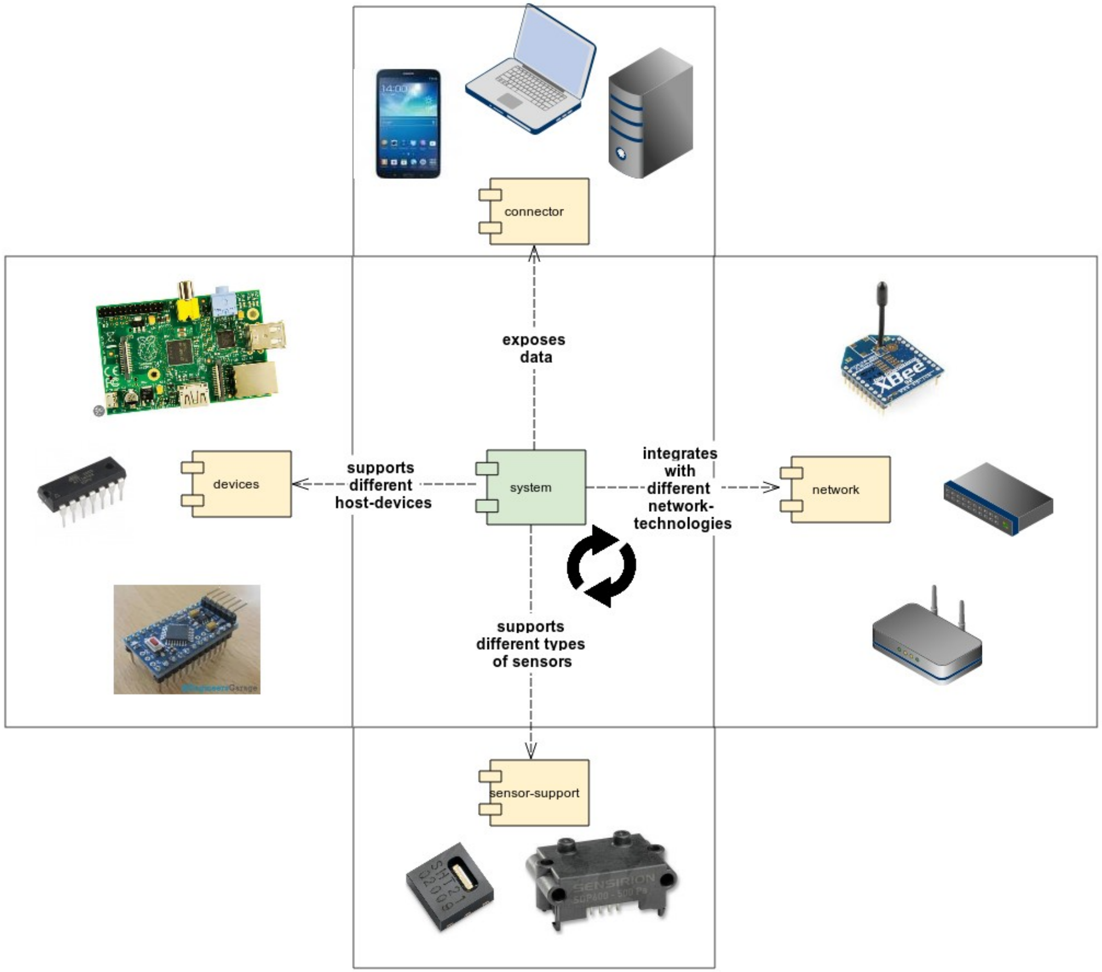
System Challenges

TOP 5 Challenges

- **Extensibility**
  - Adding new sensors and configure
  - Adding new sensor-type without changing the system (open-closed-system)
- **Configurability**
  - Changing sensor-parameters
  - Changing scheduling

System Architecture

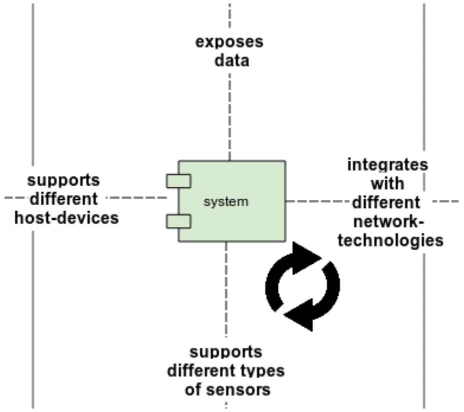
System-concept: **runtime and dependencies**



System Architecture

System-concept **Runtime**

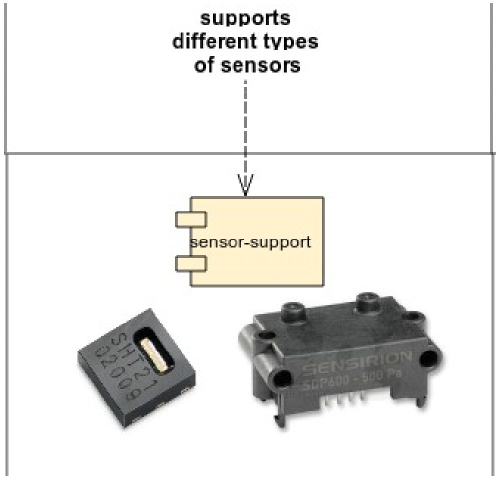
- Scheduling measurements
- Relying on system abstractions
  - Integration
  - Storage
  - Device-abstractions
  - More sensors
  - Of different types



System Architecture

System-concept: **Support different sensors**

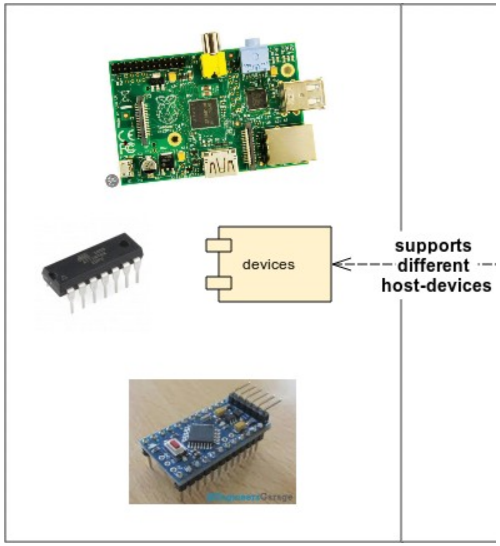
- System supports
  - Extracting data from multiple Sensors
  - Multiple types via Sensor-abstraction



System Architecture

System-concept: **Support different devices**

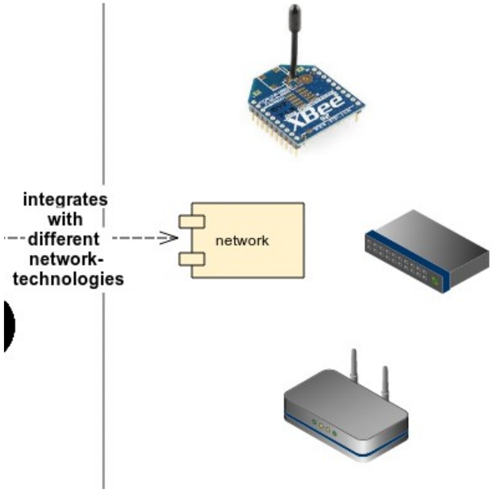
- Isolate system-dependencies
    - Scheduling
    - Digital interfaces
    - ...
- Support for
    - Low level (c-api)
    - High level (java)



System Architecture

System-concept: **Network independence**

- Integration-capability isolated
  - Local integration
  - Zigbee
  - WIFI
  - MQTT
  - ...

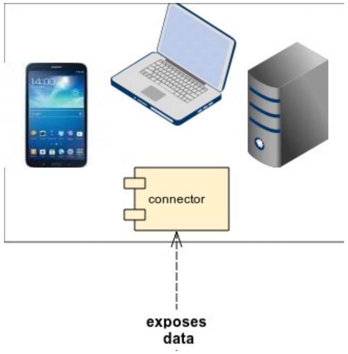


!! System provides abstraction and pluggability to adapt, not all implementations exists !!

System Architecture

System-concept: **Data exposure**

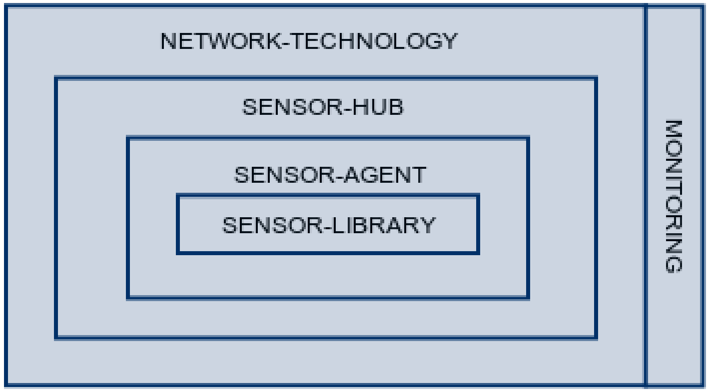
- Connectors for clients
  - Open protocols to integrate with various kind of devices



- Current provided protocol
  - REST exposing
    - JSON
    - CSV (under construction)

System Architecture

Design-concept: **Layering**

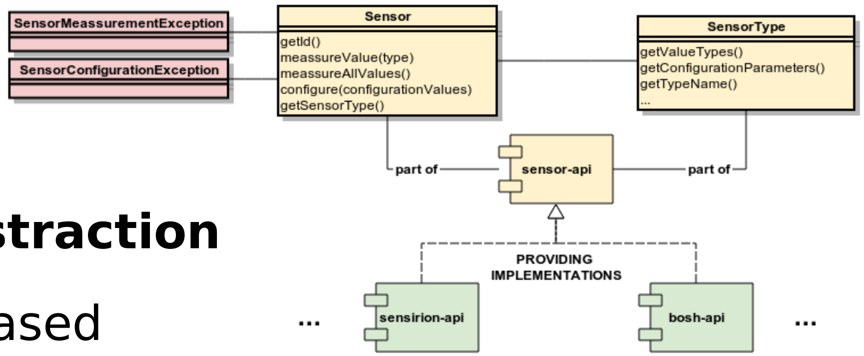


Different building blocks

- Built on top
- Inner layers can be used independently
- Segregation by interface

System Architecture

Layer: **Sensor-API**



Sensor- and SensorType-abstraction

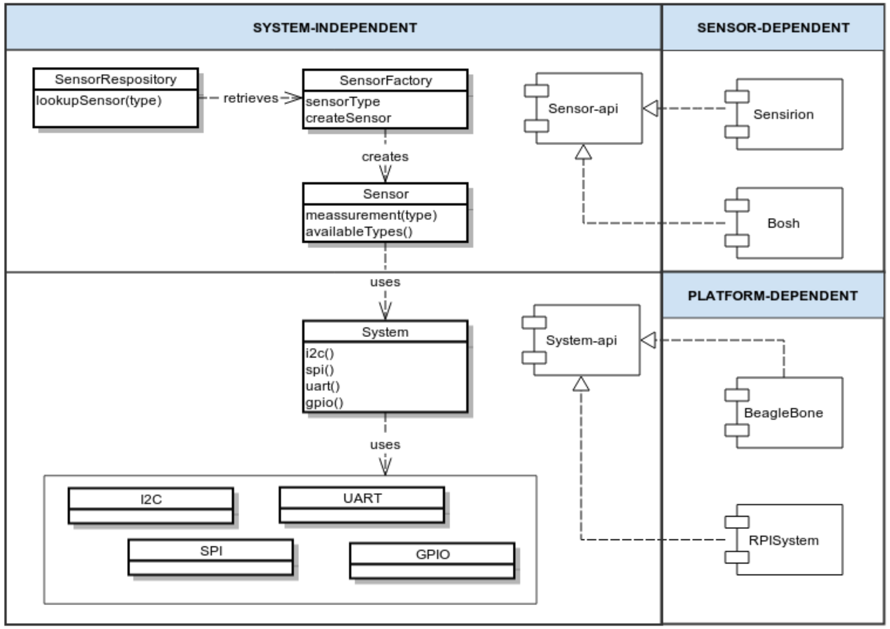
- Interfaces key-value pair based
- Modules containing concrete implementations
- Standardized exceptions

Goal

- Provide a repository for reuse (Github-project)
- Isolate the processing logic
- Provide an abstraction layer for Sensor-agent

System Architecture

Layer: **Sensor-API**



System-abstraction of

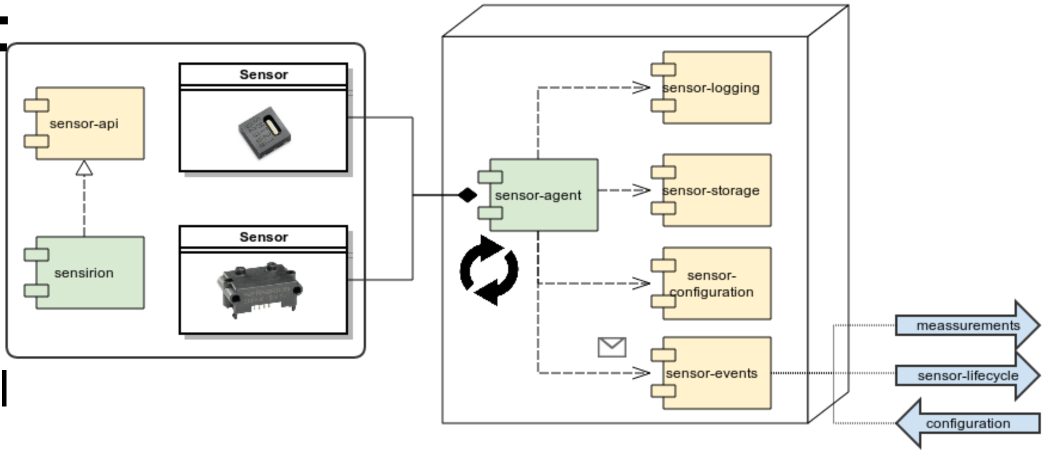
- Digital interfaces (i2c, spi, uart)
- Pin-access
- Timing

Goal

- Portability (vs scenario's)
- Choice of libraries (e.g. RPi can work JME or Pi4j)

System Architecture

Layer: **Sensor-Agent**



Runtime or application

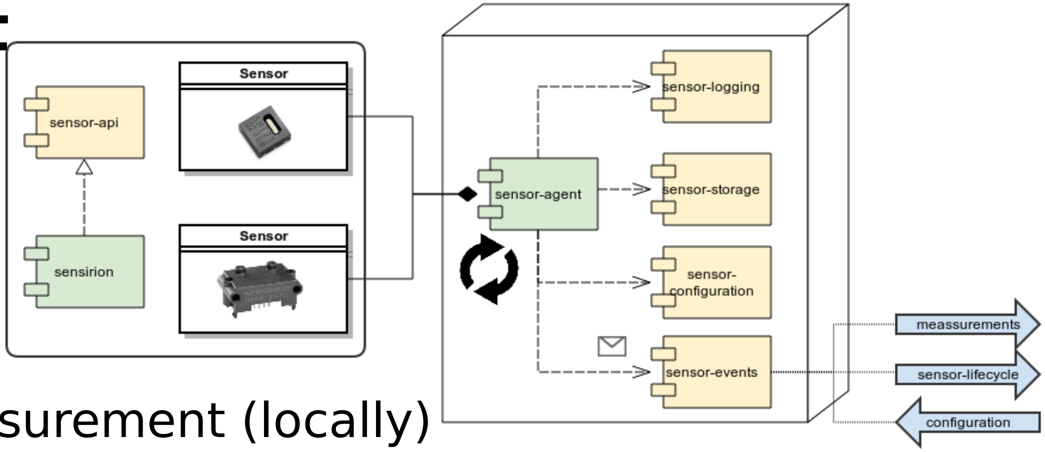
- Captures data at interval
- Manages sensors via sensor-api-abstraction
- Notifies and communicates via sensor-events

Goal

- Use the sensor-api without low-level coding
- Set up a measurement system based on configuration
- Plug-in architecture for cross-cutting-concerns

System Architecture

Layer: **Sensor-Agent**



Depends on abstractions

Logging

Storing the sensor-measurement (locally)

Storing the sensor-**configuration**
(might be another storage-medium than measurement)

Integrates with the outside-world via sensor-**events**

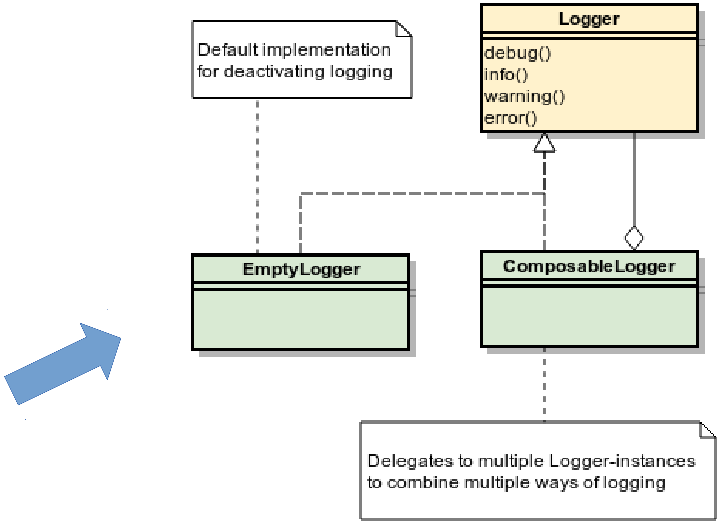
New measurement (out)
Sensor activated or reconfigured (out)
Instructions for reconfiguration (in)

System Architecture

Layer: **Sensor-Agent**

Abstractions

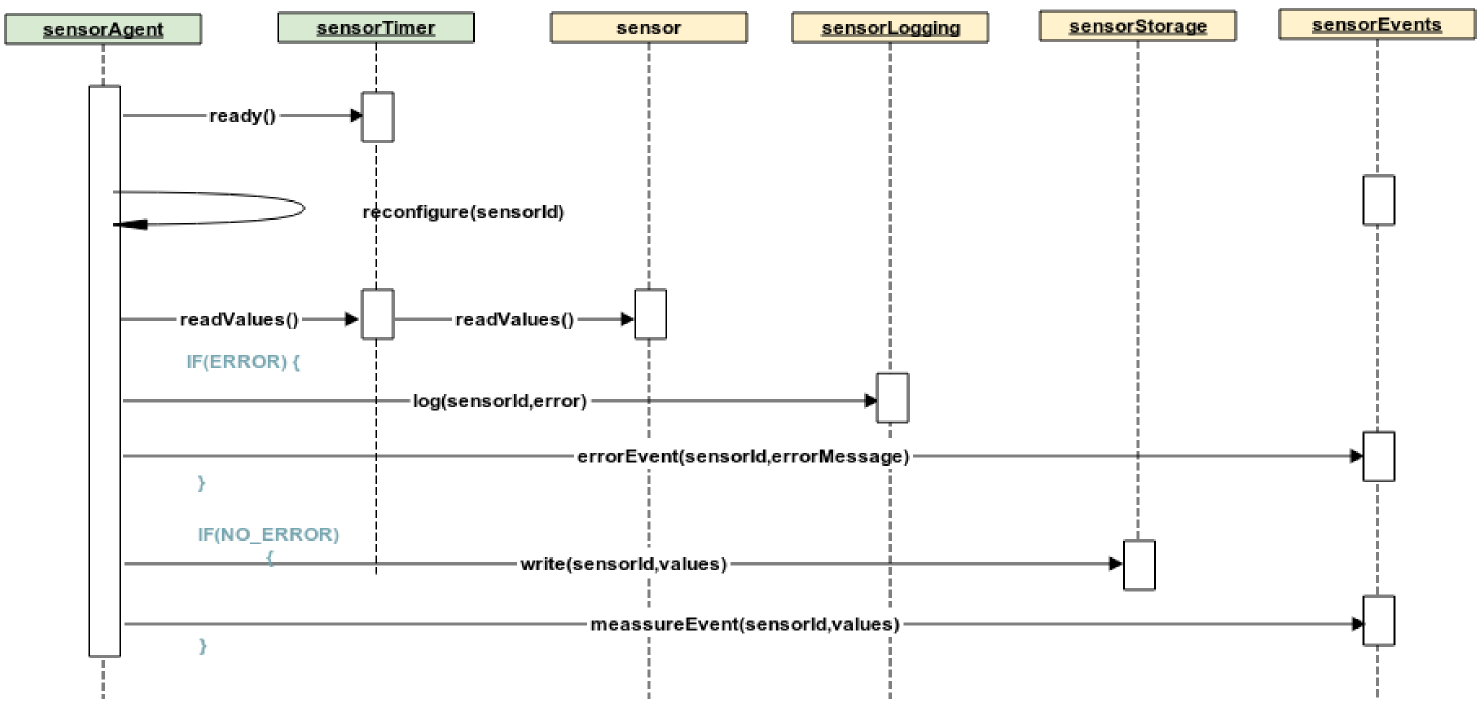
- are interchangeable
    - interface segregation
    - dependency injection
- can be combined (or composed)
- are deactivated by default
    - by default empty implementations



System Architecture

Layer: **Sensor-Agent**

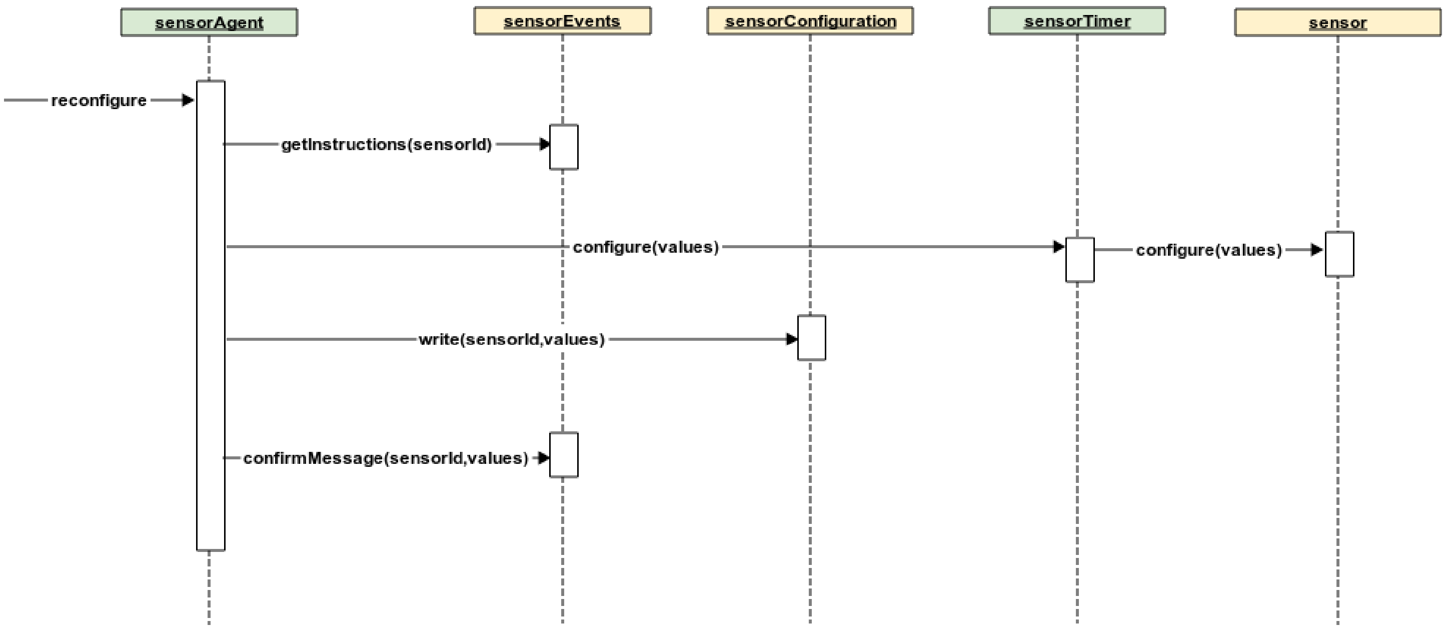
Runtime sequence



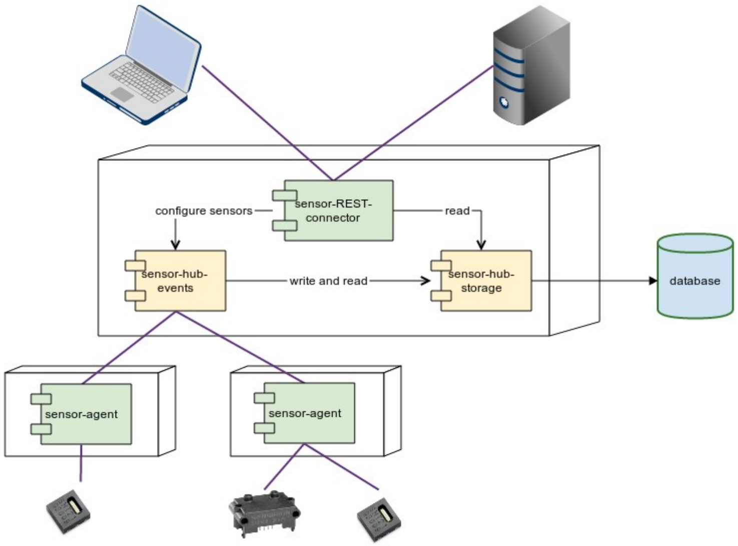
System Architecture

Layer: **Sensor-Agent**

Configuration sequence



System Architecture



Layer: **Sensor-Hub**

Runtime or (web-)application

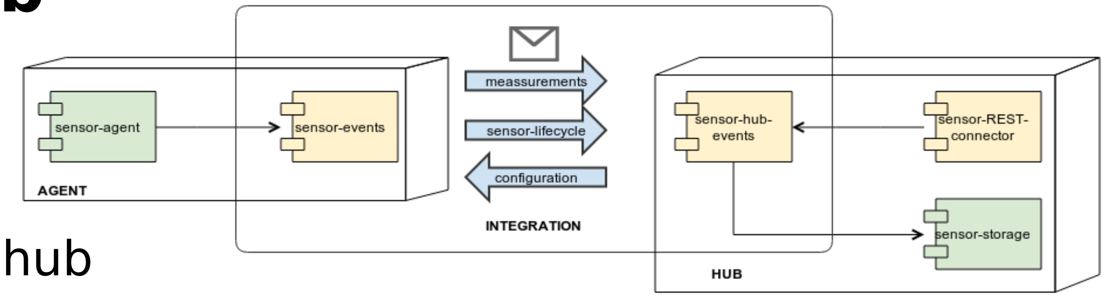
- Communicating with agents
- Centralizing data-storage
- Exposing data to users (and other devices/servers)

Goal

- Collecting and storing data from different sensors
- Enable user to query the data
- Interface for configuring remotely the sensors

System Architecture

Layer: **Sensor-Hub**



Link between agent and hub

- Measurements are pushed
- Instructions are forwarded to
- Events
  - Confirmation of configuration
  - Errors

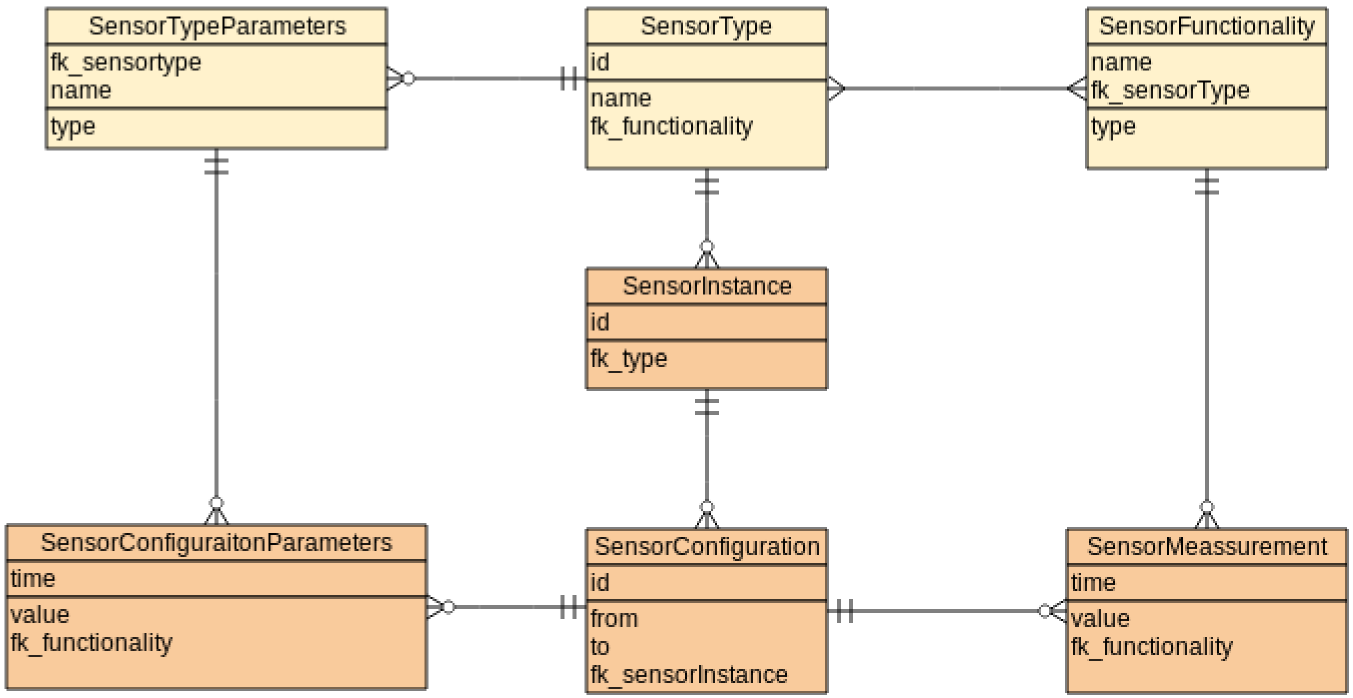
Sensor-hub-events and sensor-events

- Should integrate with same protocol
- Code is message based

System Architecture

Layer: **Sensor-Hub**

Datamodel



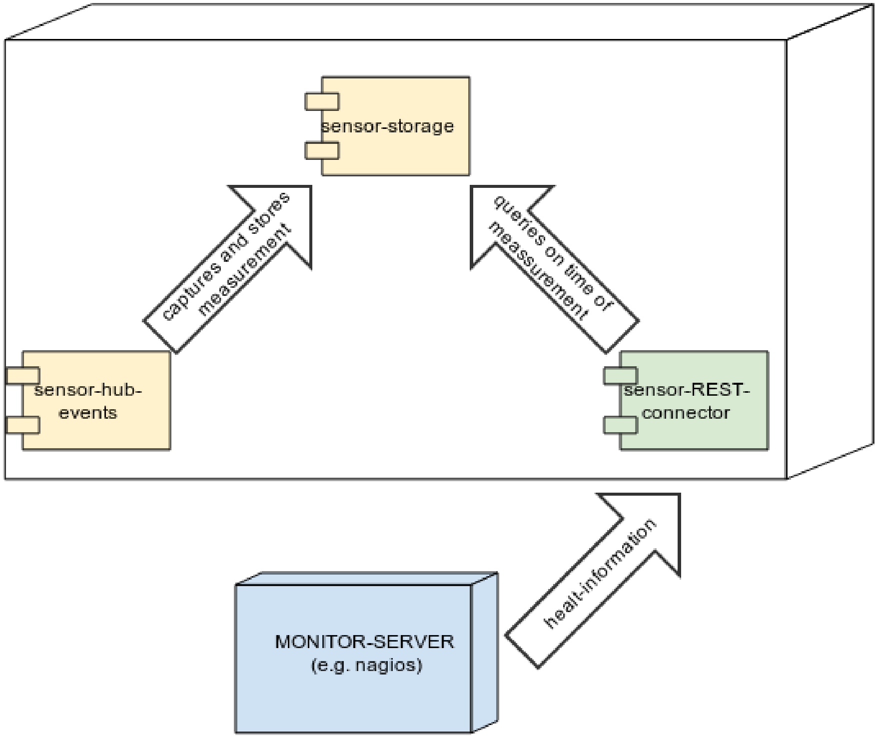
System Architecture

Layer: **Monitoring**

Monitor tool can query for

- Error events
- Deactivated agents or sensors (activity-monitoring)

REST-interface



Development principles

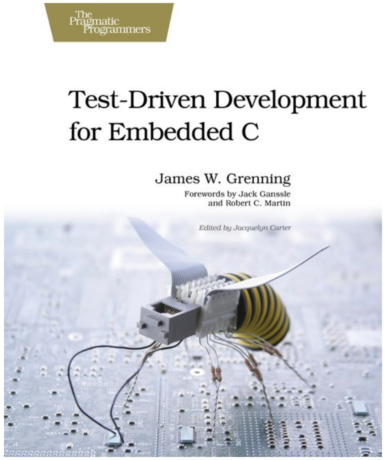
S
Single Responsibility Principle

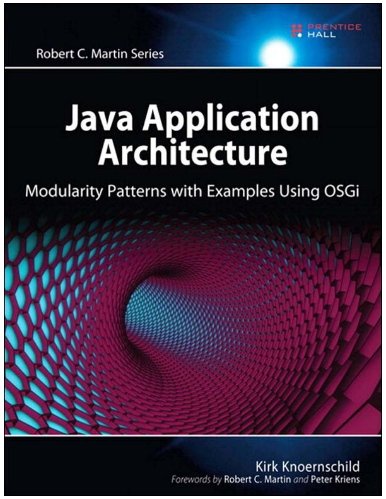
O
Open Closed Principle

L
Liskov Substitution Principle

I
Interface Segregation Principle

D
Dependency Inversion Principle





Development principles

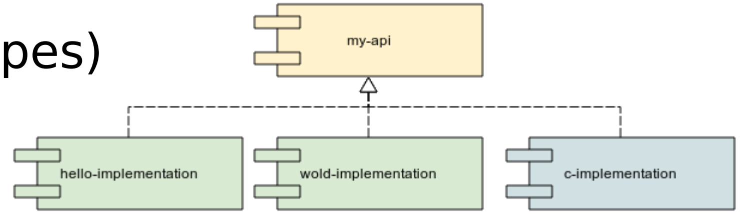
Modularity

Package code and classes into reusable and composable package

Provide

api-components

(interfaces and types)



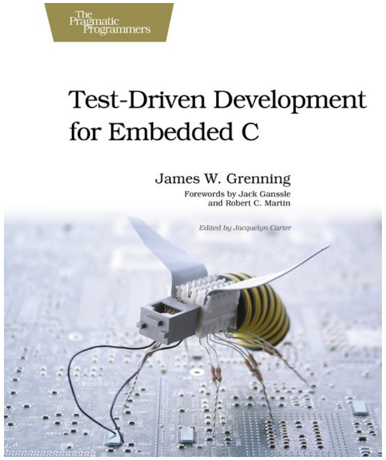
concrete implementations

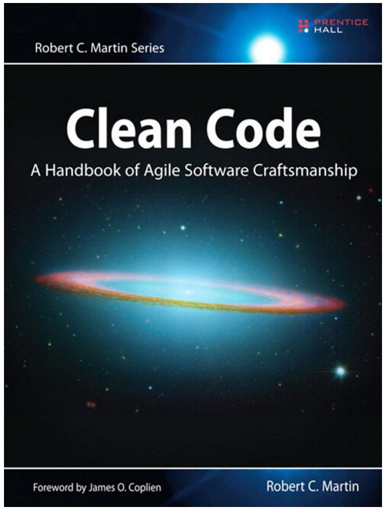
Code needs to be **SOLID**

Development principles

Test Driven Development (**TDD**)

- Drive your code trough tests
- Just enough
- Isolate dependencies





Technology positioning

Netwerk and IOT

Application
Presentation
Session
Transport
Network
Data link
Physical

Conclusion

Conclusion