Flow based software design for embedded systems

Michael Guntli – Software Engineer
IMT Information Management Technology AG
Gewerbestrasse 8
9470 Buchs, Switzerland

imt.
making ideas work

Where does it come from..

- Ralf Westphal
  - «Design smart not hard»
  - «Zusammenstecken wie Lego»
  - www.ralfw.de
  - www.clean-code-advisors.com

- Actor model
  - Concurrent computation
- Smalltalk
  - Smalltalk object

Object-oriented programming

<<Software objects interact and communicate
with each other by sending messages to each other.>>

Smalltalk Object

- Hold state
- Receive a message from itself or another object
- In the course of processing a message, send messages to itself or another object

➔ Focus on behavior and interaction

C++ / C# / Java OO

- Classes and Methods
- Inheritance - Polymorphism
- Abstraction – Encapsulation

➔ Focus on structure

ActiveParts

- C++ framework and runtime for:
  - highly concurrent
  - independent and efficient message driven applications
  - focused for bare metal embedded systems

ActivePart

- Objects which encapsulate:
  - State
  - Behavior
  - Data
- Communicate to each other by exchanging messages

➔ Nice colleagues at work

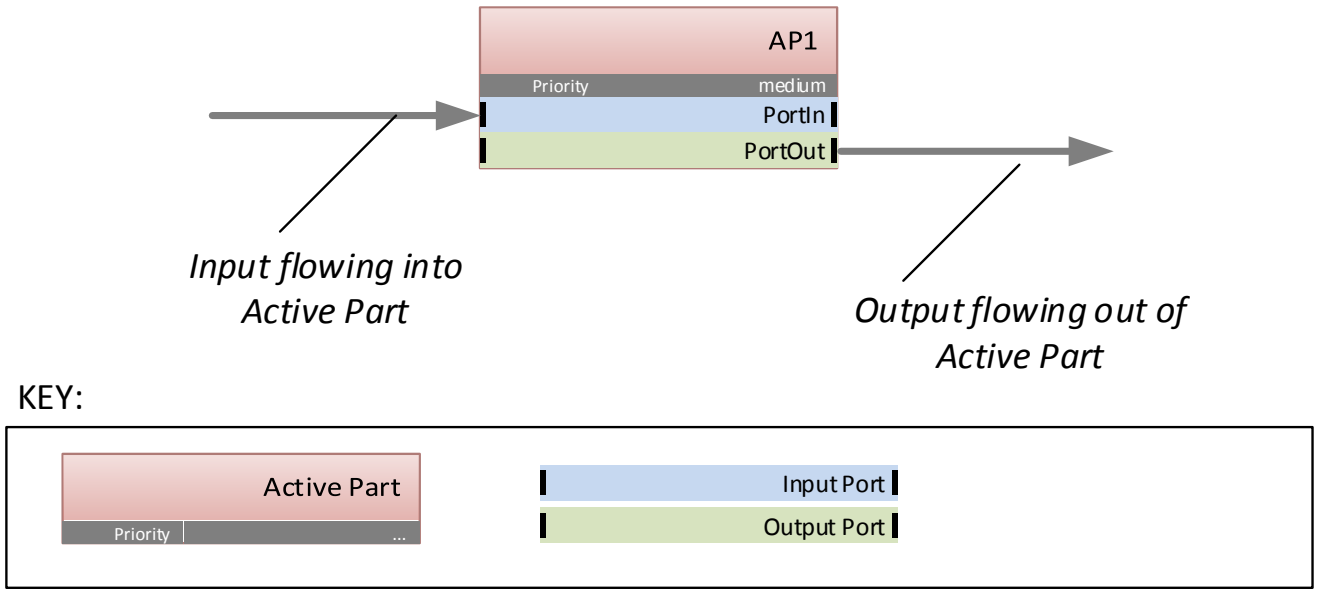
System philisophy

1. Structure across the main data flows
2. Split up by separation of concerns
3. Form hierarchies after separation based on belonging

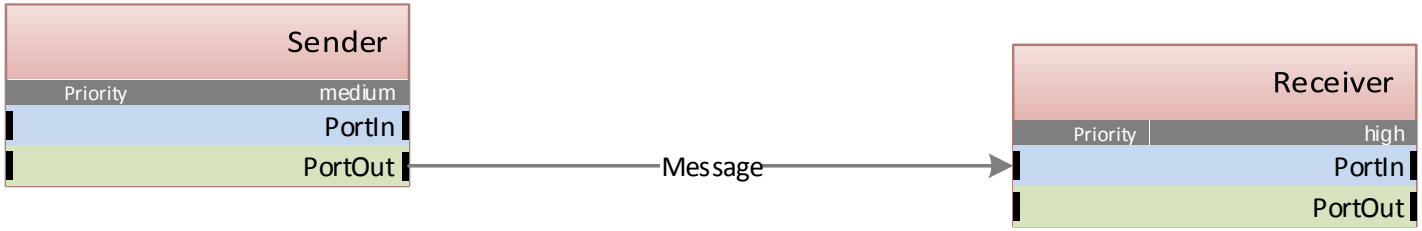
Implementing an ActivePart

- Simple procedural programming (sequential)
- Structured programming
- Functional programming
- Object-oriented programming
- Low-level hardware programming
- Multiparadigm-programming

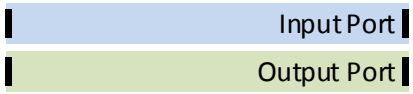
ActivePart - Notation



Message passing

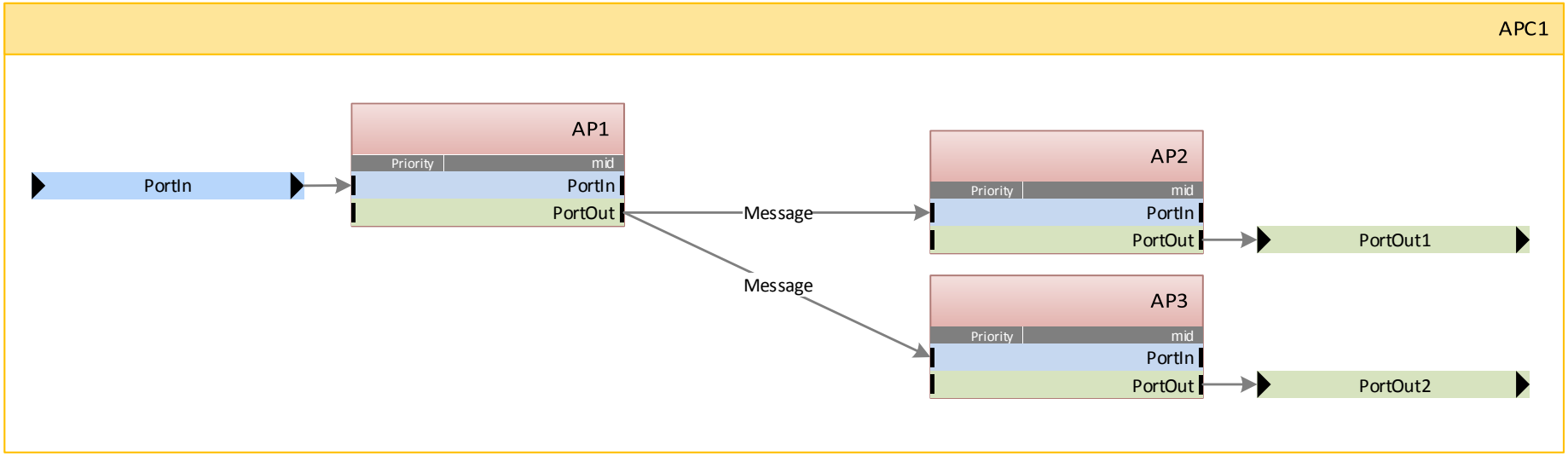


KEY:

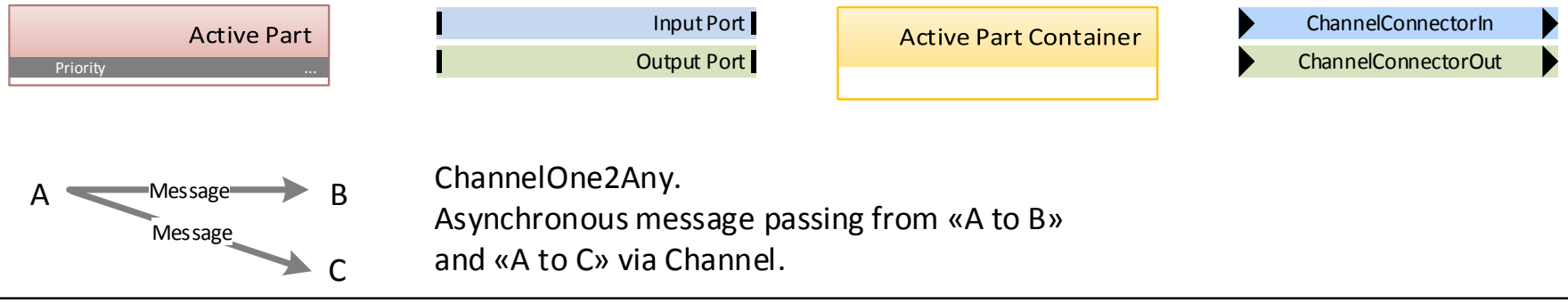


ChannelOne2One
Asynchronous message passing from A to B via Channel.

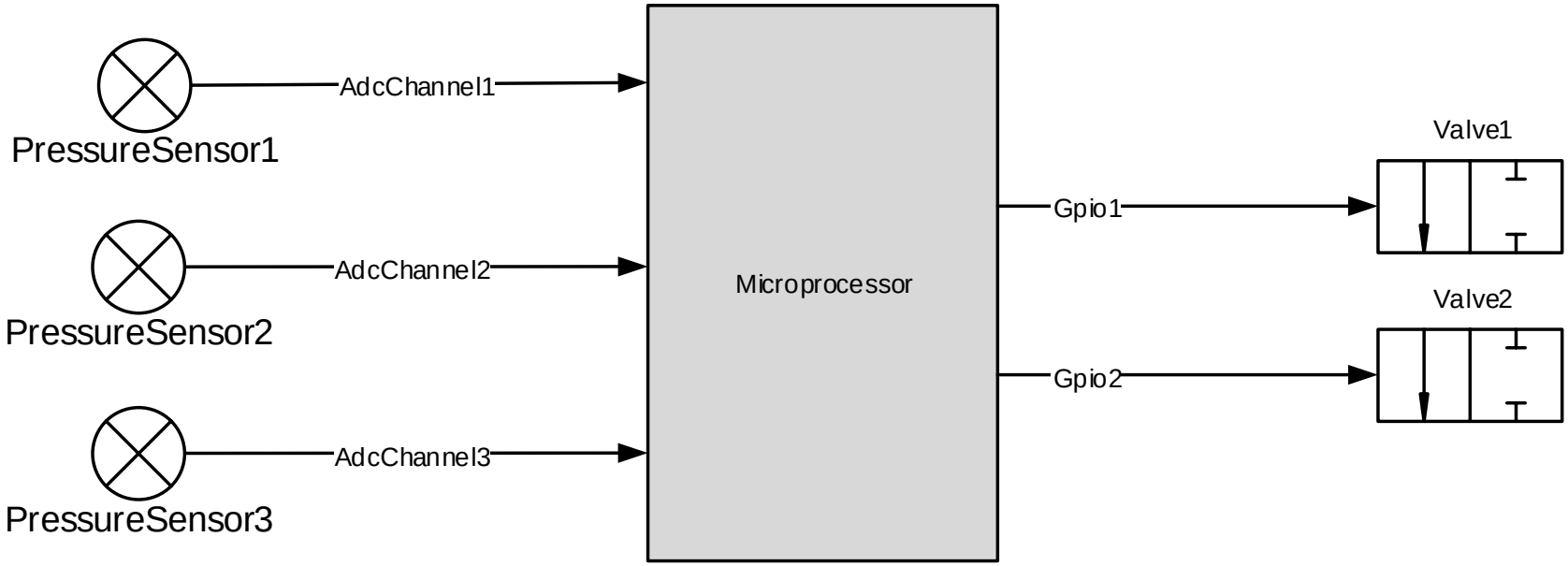
Hierarchical structure



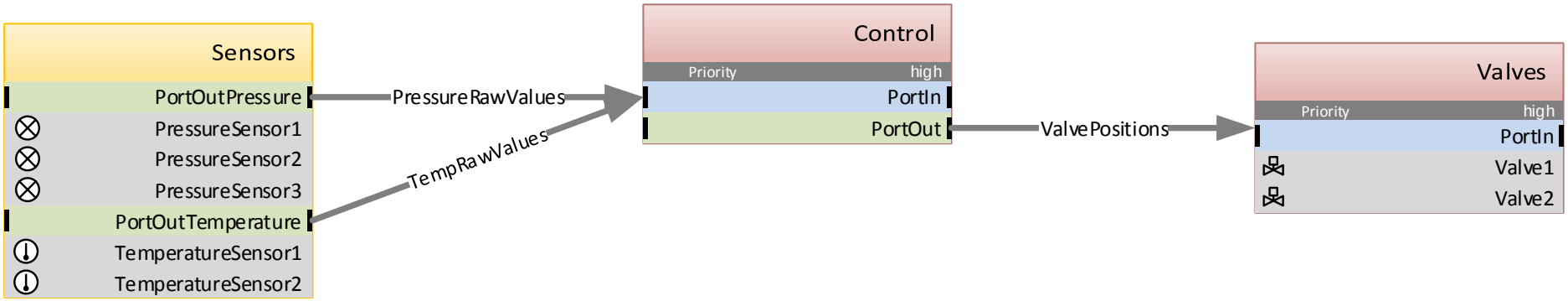
KEY:



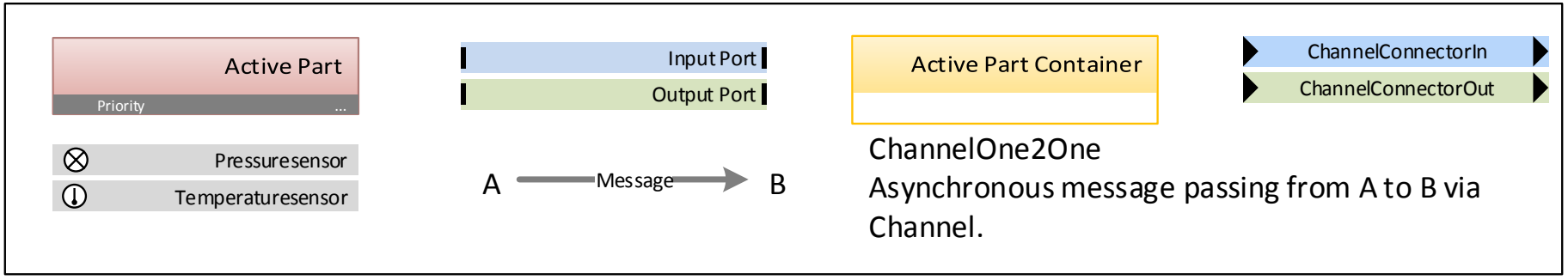
Example hardware block diagram



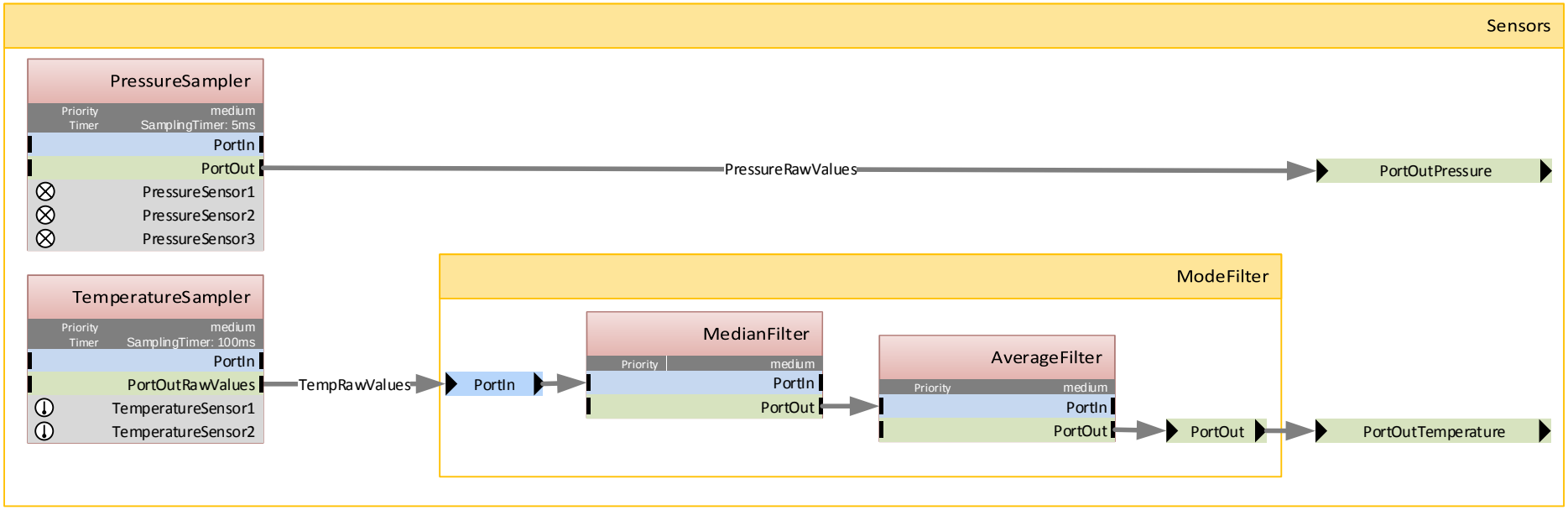
Example software design



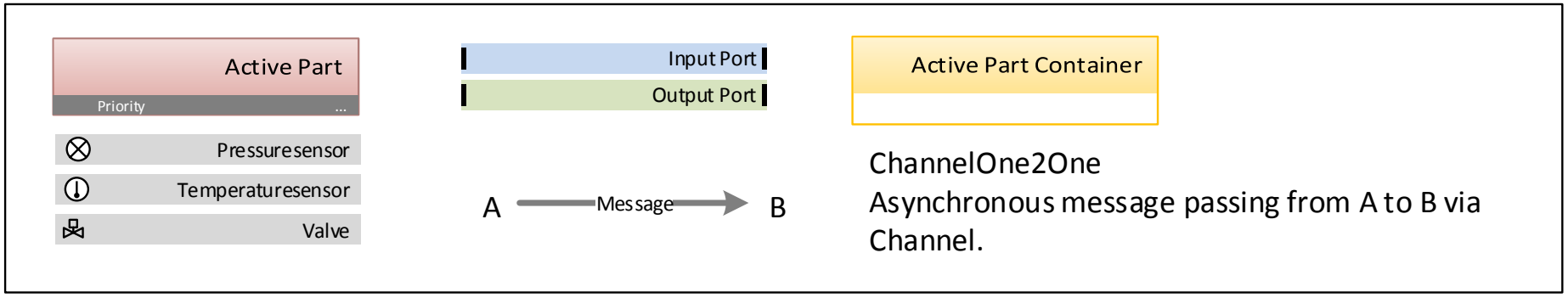
KEY:



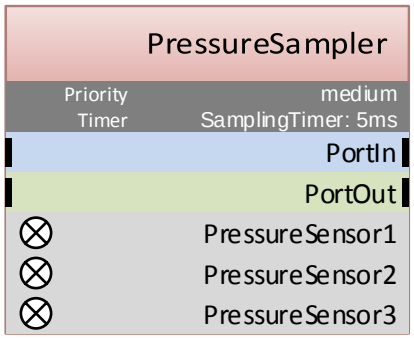
Sensors


Sensors
PressureSampler
Priority medium
Timer SamplingTimer: 5ms
PortIn
PortOut
PressureSensor1
PressureSensor2
PressureSensor3
PressureRawValues
PortOutPressure
TemperatureSampler
Priority medium
Timer SamplingTimer: 100ms
PortIn
PortOutRawValues
TemperatureSensor1
TemperatureSensor2
TempRawValues
ModeFilter
PortIn
MedianFilter
Priority medium
PortIn
PortOut
AverageFilter
Priority medium
PortIn
PortOut
PortOut
PortOutTemperature

KEY:


Active Part
Priority ...
Pressuresensor
Temperaturesensor
Valve
Input Port
Output Port
A
Message
B
Active Part Container
ChannelOne2One
Asynchronous message passing from A to B via Channel.

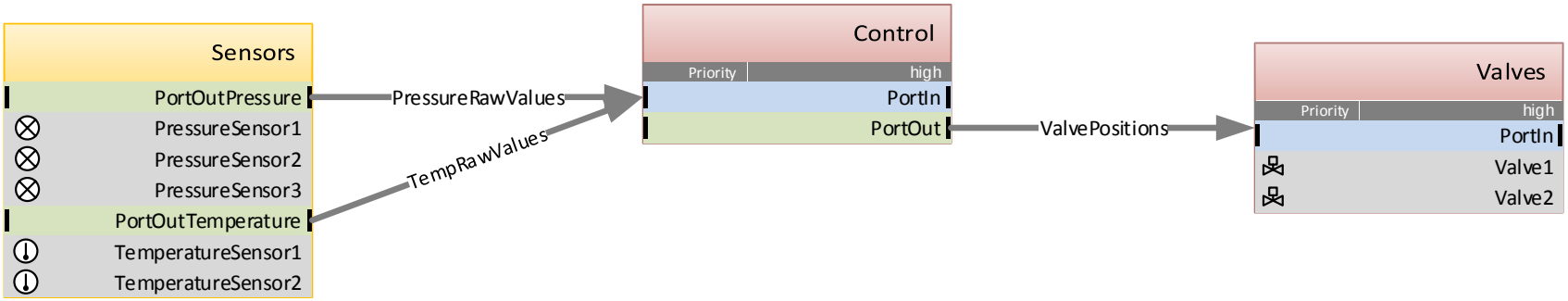
PressureSampler – C++ Code




```
void PressureSampler::receive(const MessageIfc& msg) {
    if (isSamplingTimer(msg)) {
        PressureRawValues values;
        values.setValue(PressureSensor1, AdcDriver::read(AdcChannel1));
        values.setValue(PressureSensor2, AdcDriver::read(AdcChannel2));
        values.setValue(PressureSensor3, AdcDriver::read(AdcChannel3));

        PortOut.send(values);
    }
}
```

Application – C++ Code



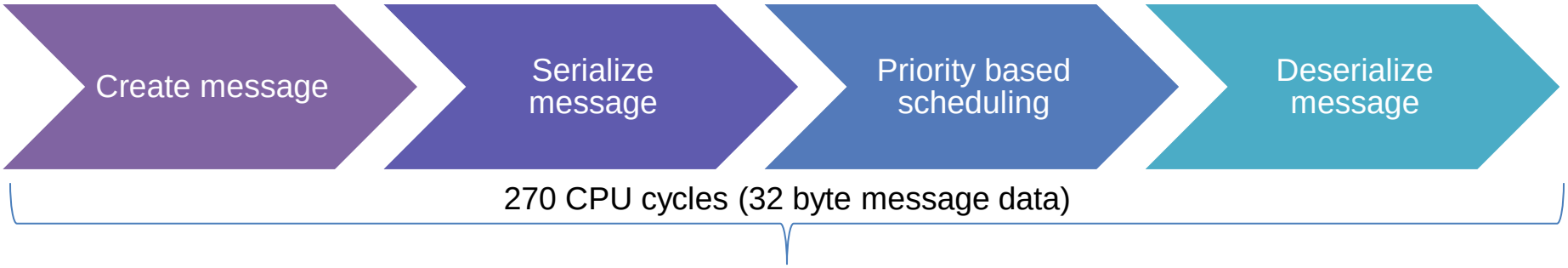

```
App::initialize() {
    // glue code to connect ports
    ChannelSensorsPressureToControl.connectPorts(Sensors.PortOutPressure, Control.PortIn);
    ChannelSensorsTempToControl.connectPorts(Sensors.PortOutTemperature, Control.PortIn);
    ChannelControlToValves.connectPorts(Control.PortOut, Valves.PortIn);
}
```

Challenges for embedded systems

- Usually no RTOS
- Small amount of memory available (kilobytes)
- Small amount of flash available (kilobytes)
- Hard-realtime requirements (microseconds)
- Used in safety criticital devices

Features

- MISRA C++ 2008 compliant (no heap)
- Very small code size (~2kByte ROM)
- Extremly fast execution time



CPU	Execution time [µs]	Execution time [ms]
Cortex-M3 @8MHz	34µs	0.034ms
Cortex-A8 @720MHz	0.6µs	0.0006ms

Scheduler

- Scheduler ported for multiple platforms:
  - STM32 ARM Cortex-M3 (8MHz up to 72MHz)
  - Texas Instruments AM335x ARM Cortex-A8 (up to 720MHz)
  - Windows Embedded Compact 7
  - Win32 (Threads & Slim Reader / Writer Locks)
  - QNX (pthread & mutex / condition)