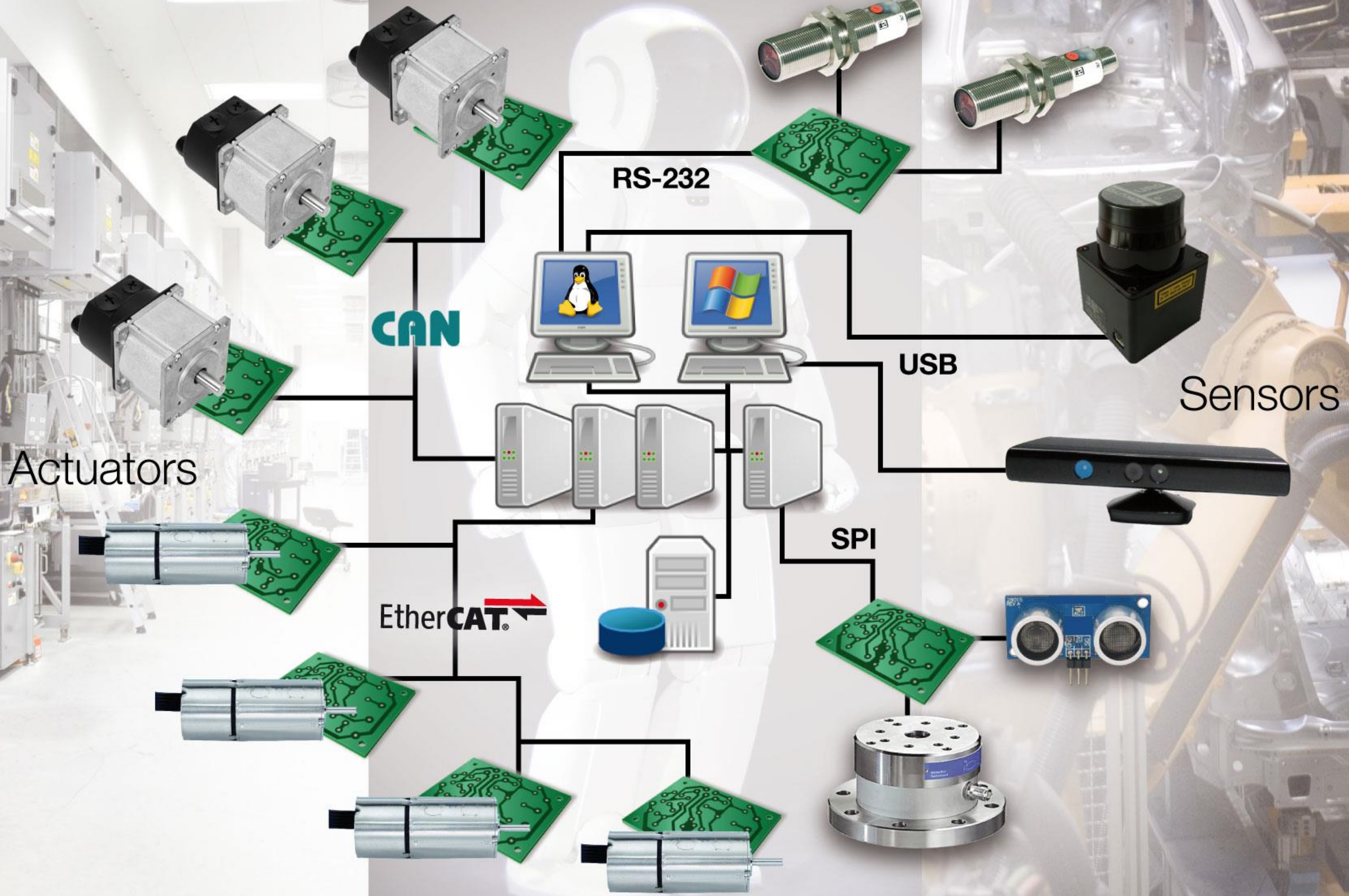
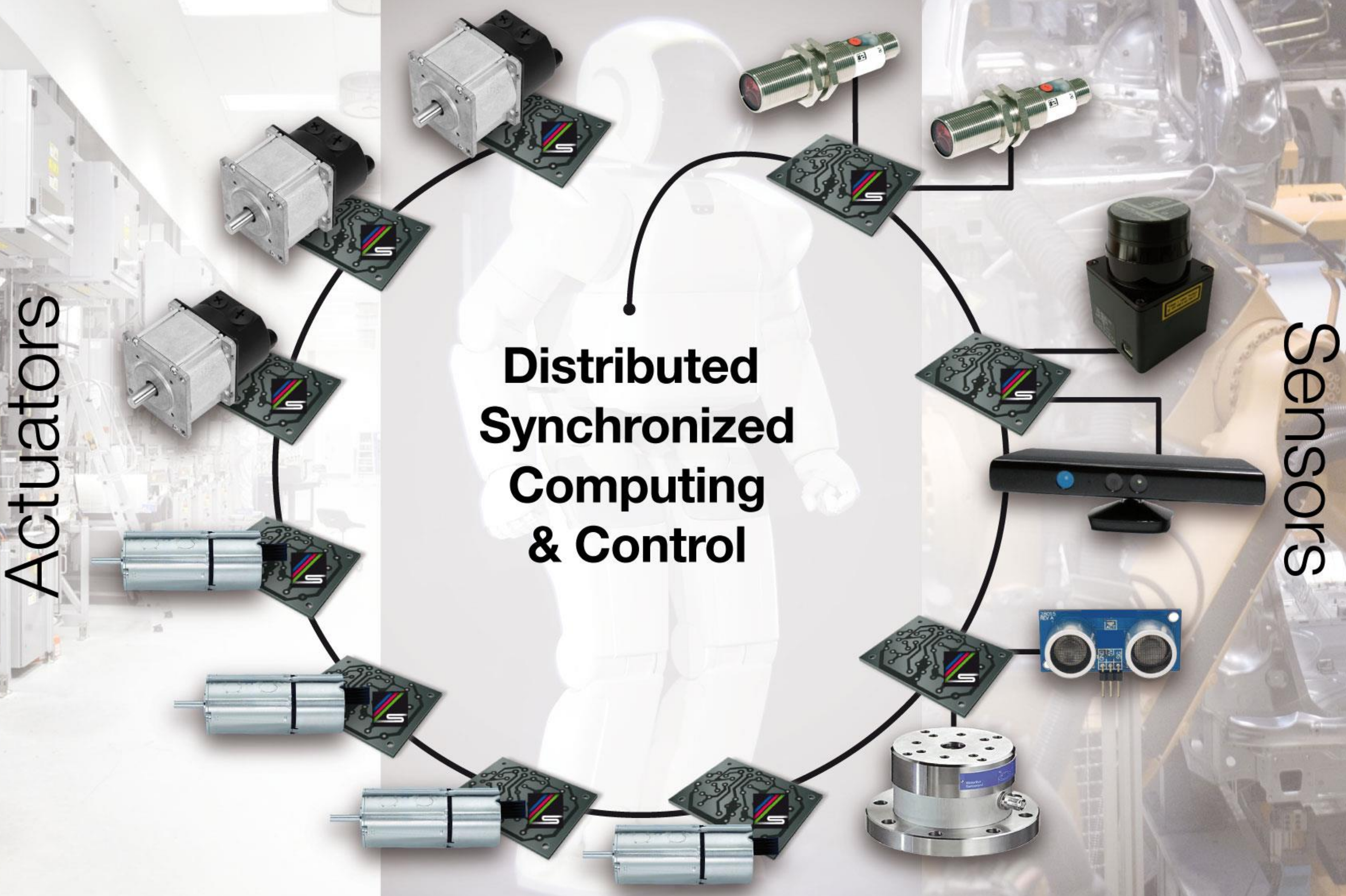
Introducing rosc

ROSCon 2013

Stuttgart, 12th May 2013

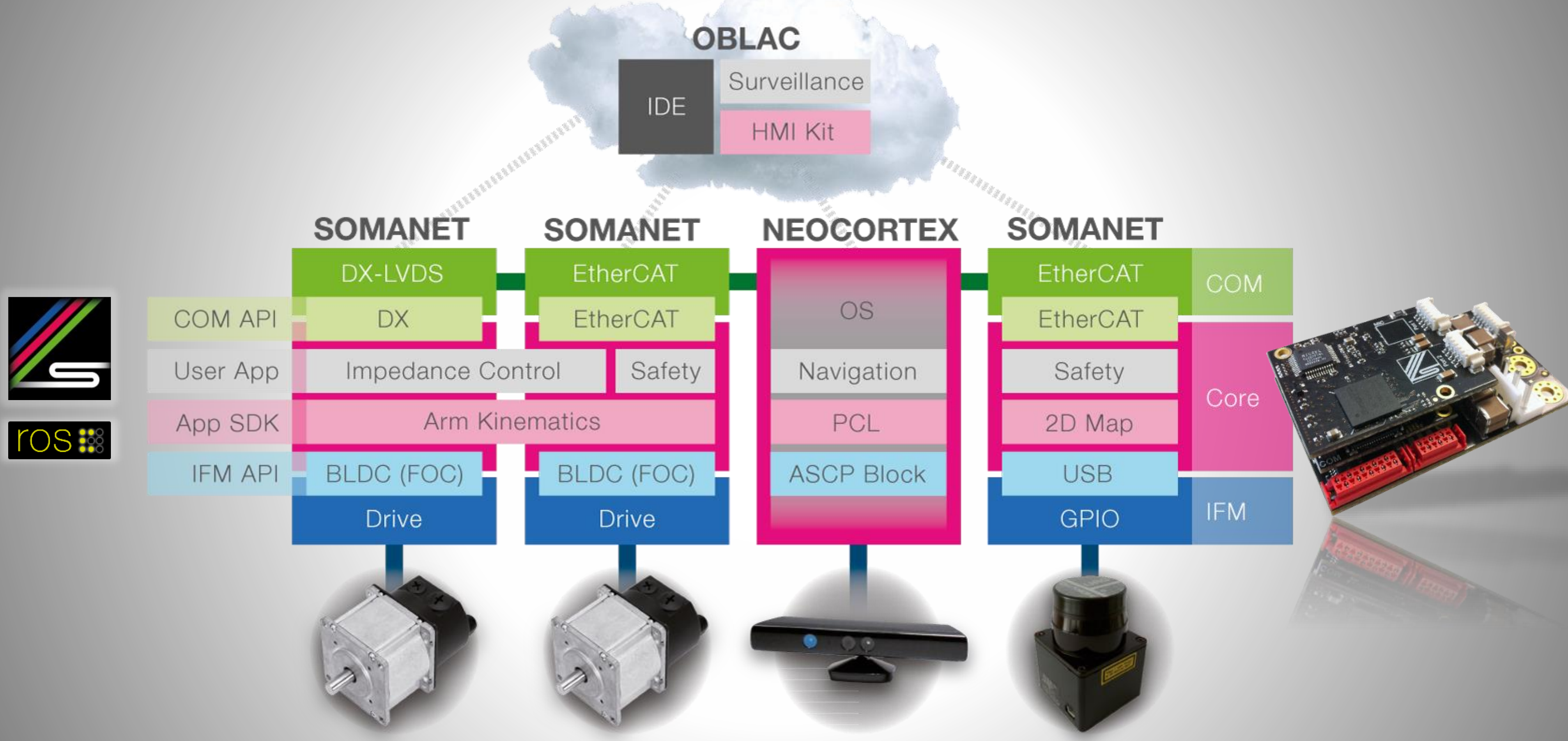
Conventional Control Systems





Distributed Synchronized Computing & Control

Synapticon DYNARC Platform



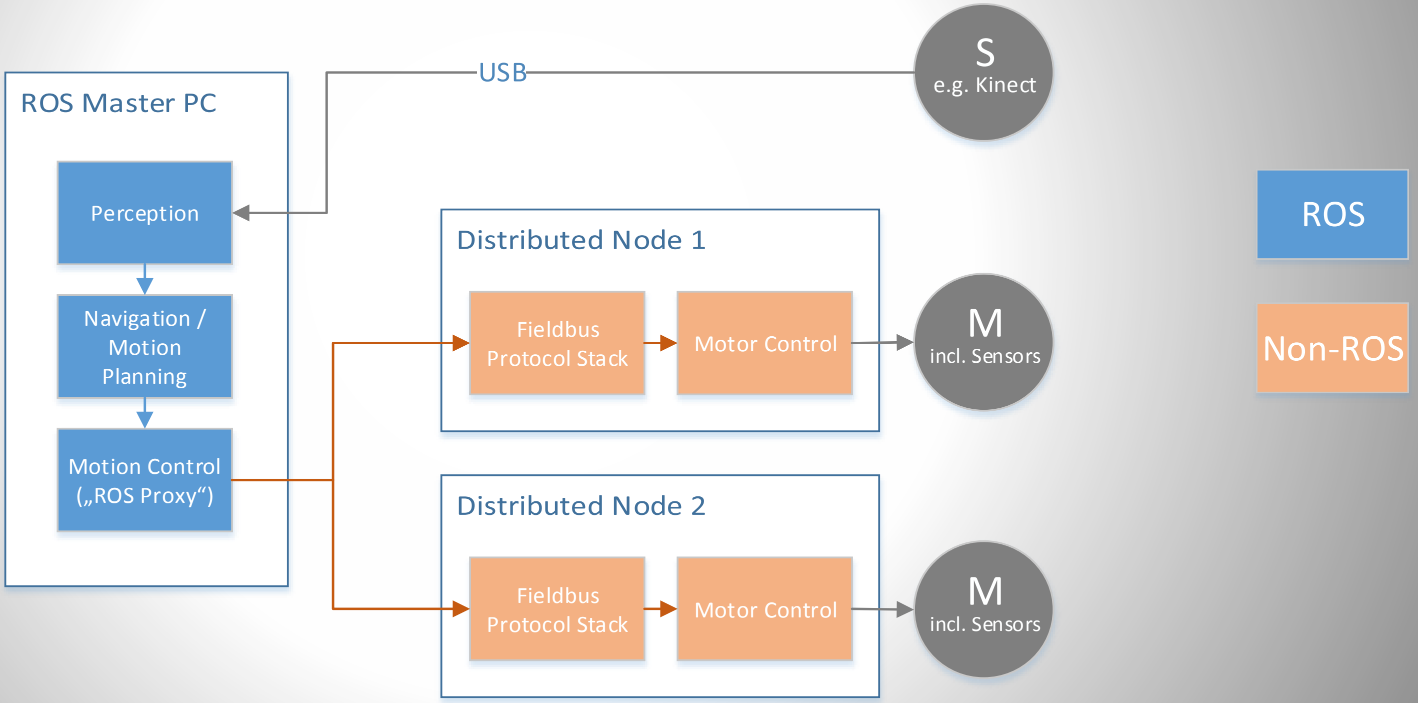
OBLAC

IDE	Surveillance
	HMI Kit

	SOMANET	SOMANET	NEOCORTEX	SOMANET	
	DX-LVDS	EtherCAT		EtherCAT	COM
COM API	DX	EtherCAT	OS	EtherCAT	
User App	Impedance Control	Safety	Navigation	Safety	Core
App SDK	Arm Kinematics		PCL	2D Map	
IFM API	BLDC (FOC)	BLDC (FOC)	ASCP Block	USB	
	Drive	Drive		GPIO	IFM

Steps towards Embedded ROS

Common architecture supported by ROS today

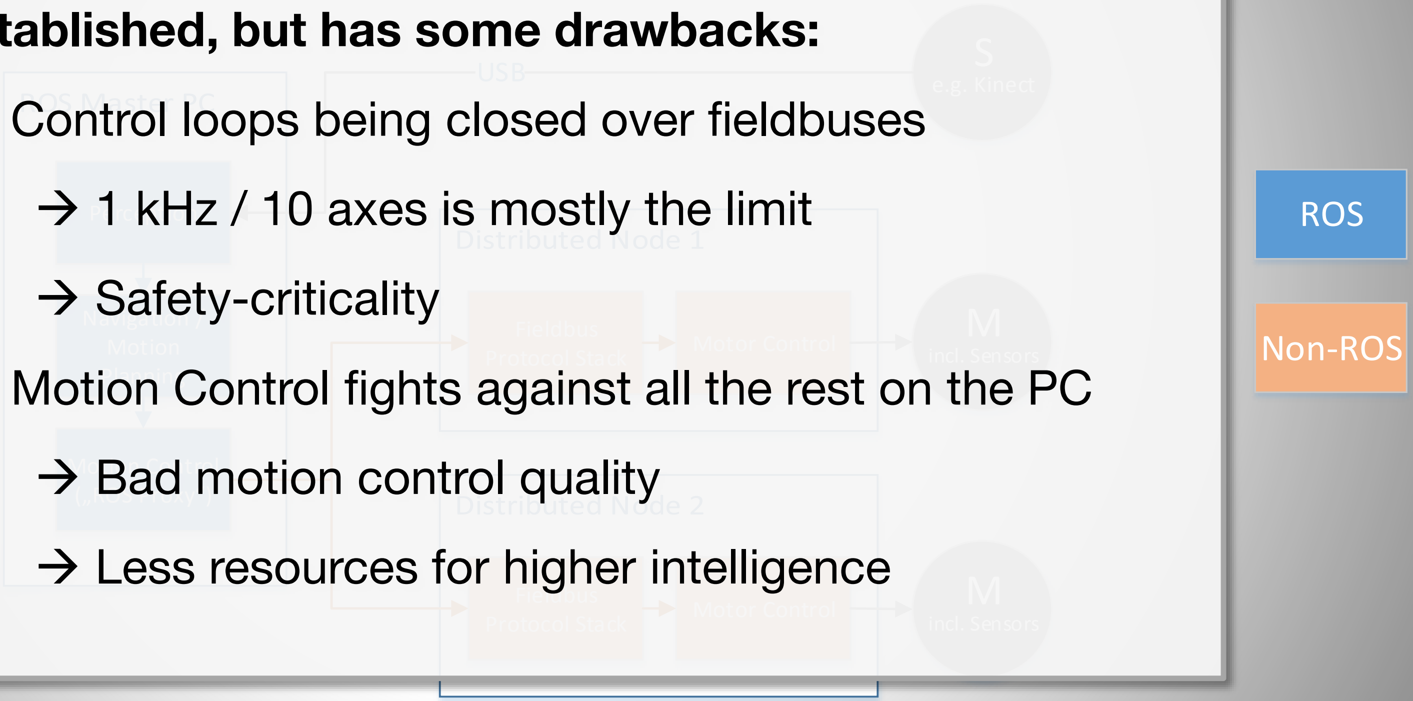


Steps towards Embedded ROS

Common architecture supported by ROS today

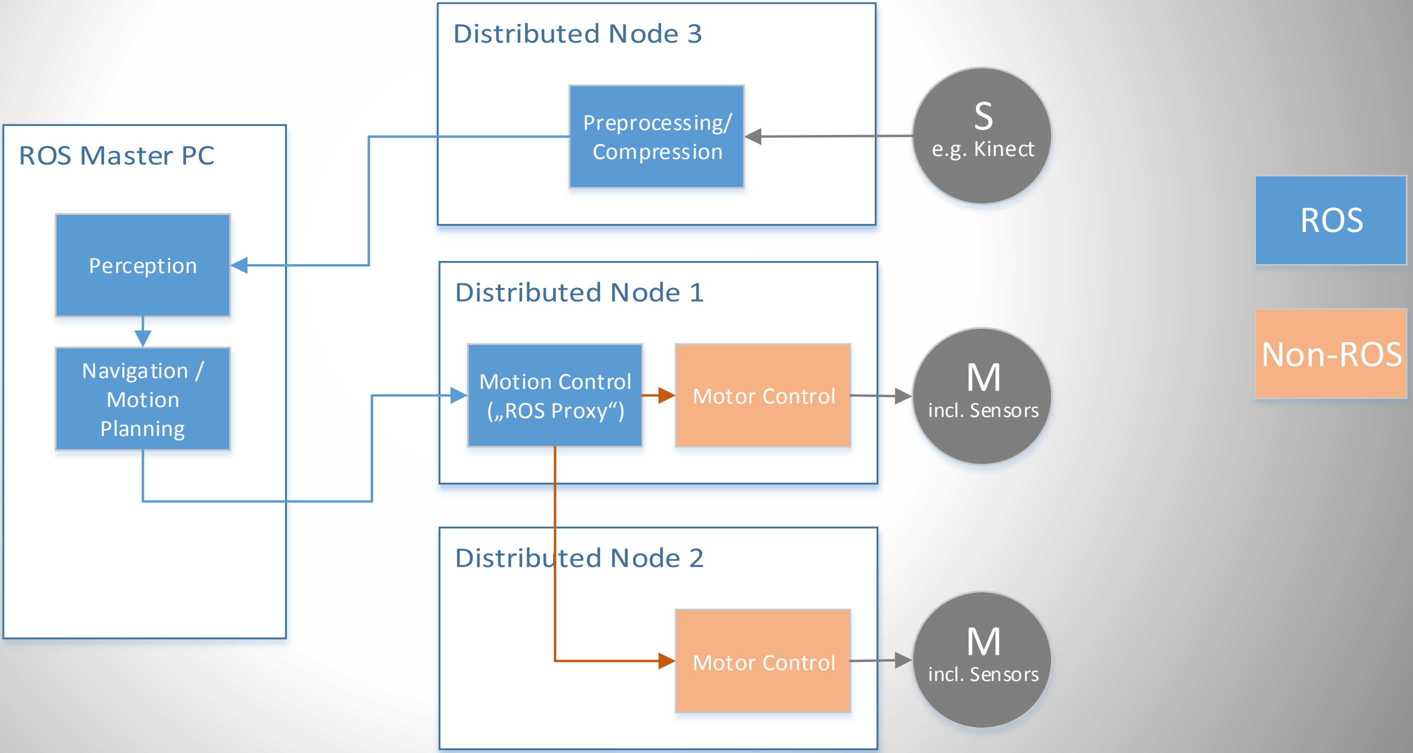
Established, but has some drawbacks:

- Control loops being closed over fieldbuses
  - → 1 kHz / 10 axes is mostly the limit
  - → Safety-criticality
- Motion Control fights against all the rest on the PC
  - → Bad motion control quality
  - → Less resources for higher intelligence



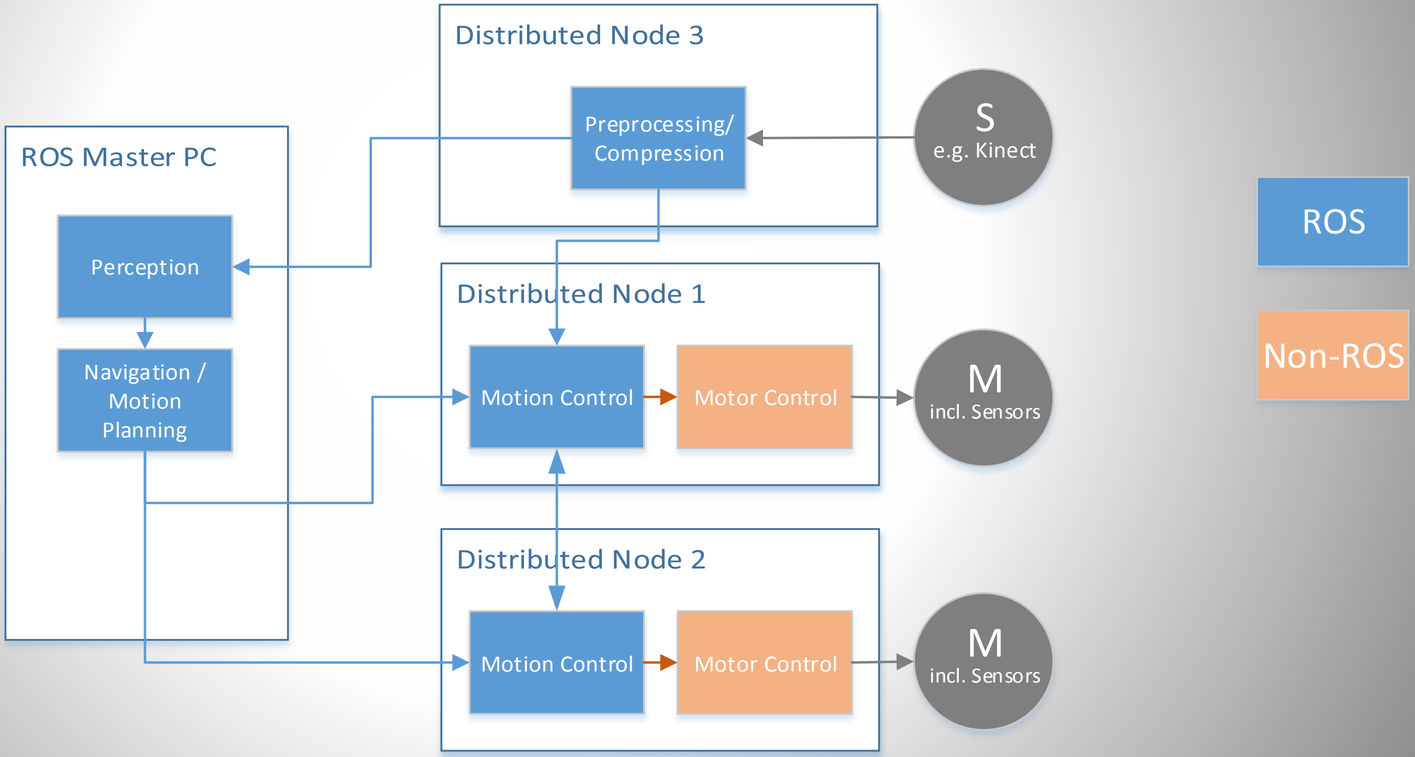
Steps towards Embedded ROS

Architecture requested (and delivered by rosc & µROSnode)

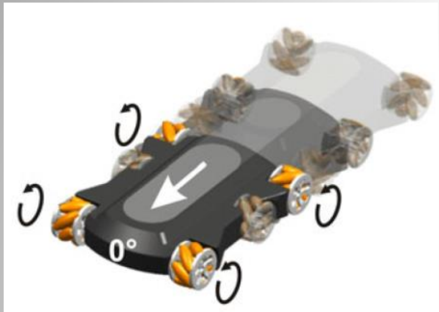


Steps towards Embedded ROS

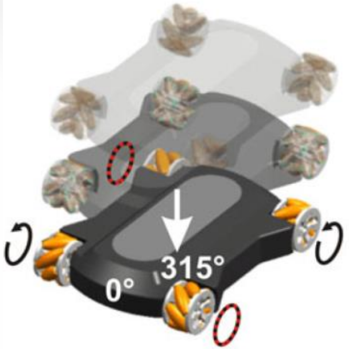
Architecture requested (and delivered by rosc & µROSnode)



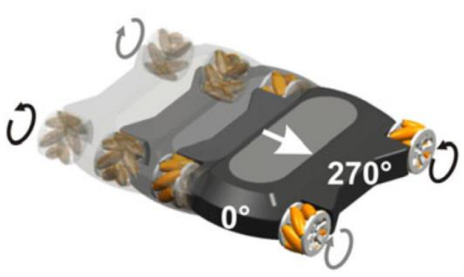
Instantaneous superposition of platform translation and rotation



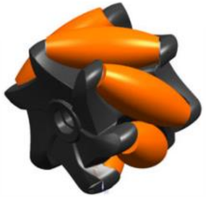
moving straight
wheels rotate in same direction



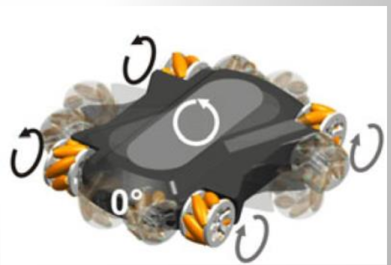
moving diagonally
two diagonally opposite wheels stand still
two others rotate in same direction



moving sideways
wheels rotate in opposite directions



Meccanum wheel
with passive rollers

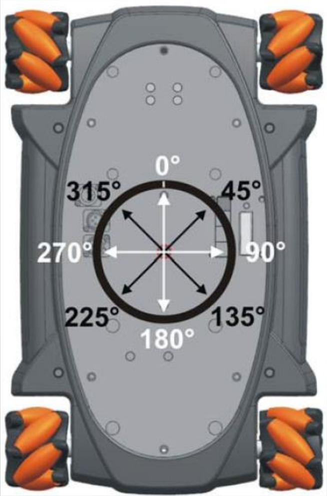


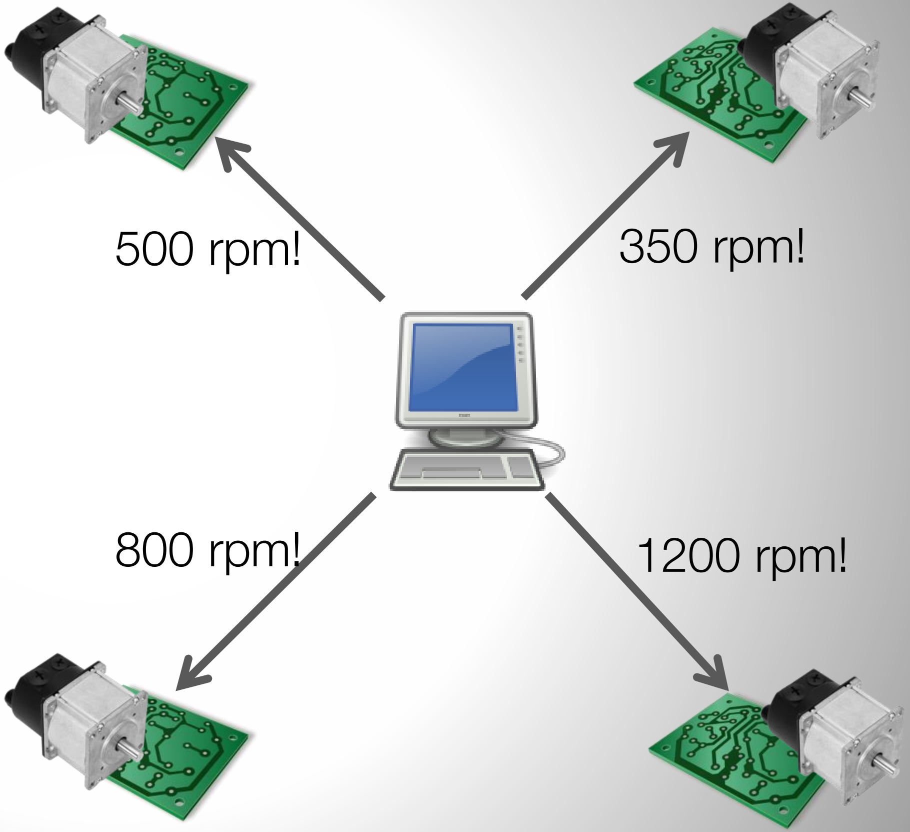
rotate around central axis
wheels on one side rotate in opposite direction to wheels on other side

Pictures courtesy of KUKA Laboratories

Industry Default: Centralized Motion Control Architecture

Centralized Dynamics Model & Control using a PC





500 rpm!

350 rpm!

800 rpm!

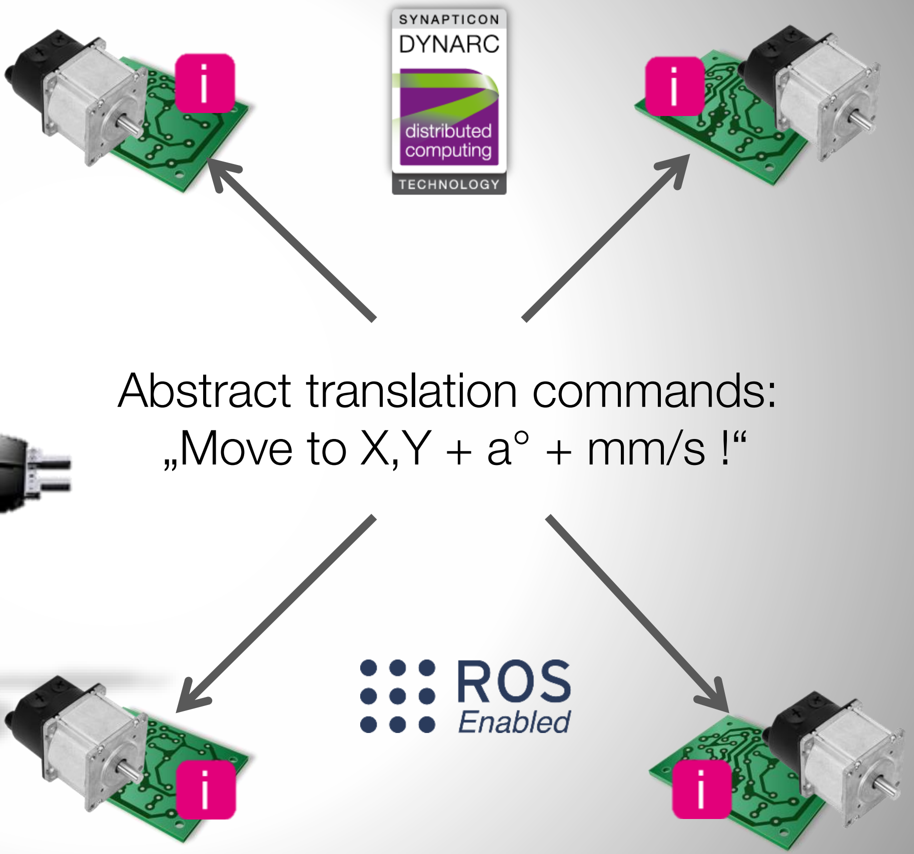
1200 rpm!

Pictures courtesy of KUKA Laboratories

Advanced Alternative: Distributed Motion Control Architecture

Dynamics Model & Control running on Motor Control Level

Same story for manipulators



Abstract translation commands:
„Move to X,Y + a° + mm/s !“

Pictures courtesy of KUKA Laboratories

Steps towards Embedded ROS

„Embedded“

- ARM Cortex A (32bit): Beagle, Panda, Raspi, ODROID, Phytec
- Intel Atom (64bit): Congatec, Kontron

Perception & Intelligence

„Small Embedded“

- ARM Cortex M (32bit): mbed
- AVR (8-32bit): Arduino
- XMOS (32bit): Synapticon SOMANET

Control & Data Acquisition

Steps towards Embedded ROS

rosc

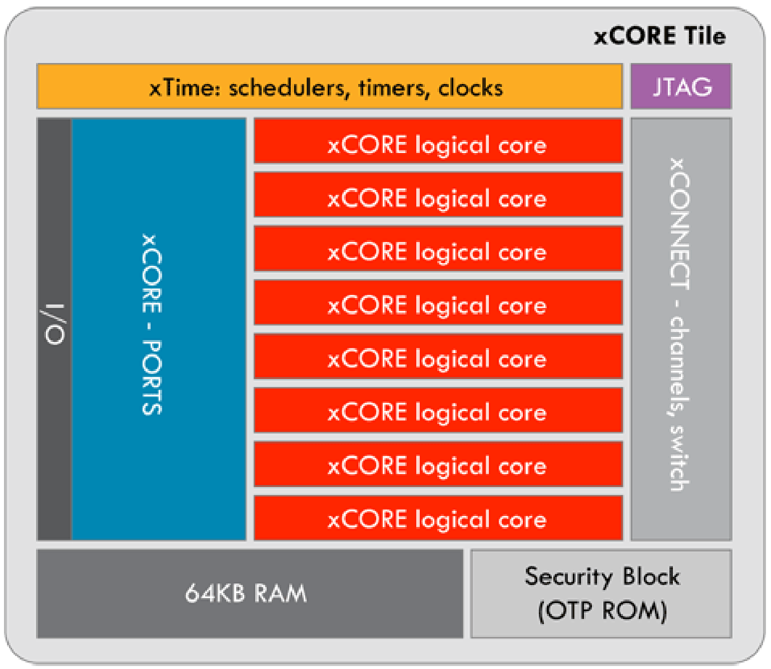
- ROS client library for Small Embedded devices
  - XML-RPC
  - TCPROS
- Light-weight (< 32 kB memory on XMOS*)
- No dependencies
- ANSI C (99)

Steps towards Embedded ROS

XMOS®

*** XMOS Arch:**

- 32bit
- Multicore (4-32)
- 125 Mhz/Core
- 64kB RAM/Tile
- Hard real-time



xCORE Tile

xTime: schedulers, timers, clocks

JTAG

I/O

xCORE - PORTS

xCORE logical core

xCORE logical core

xCORE logical core

xCORE logical core

xCORE logical core

xCORE logical core

xCORE logical core

xCORE logical core

xCONNECT - channels, switch

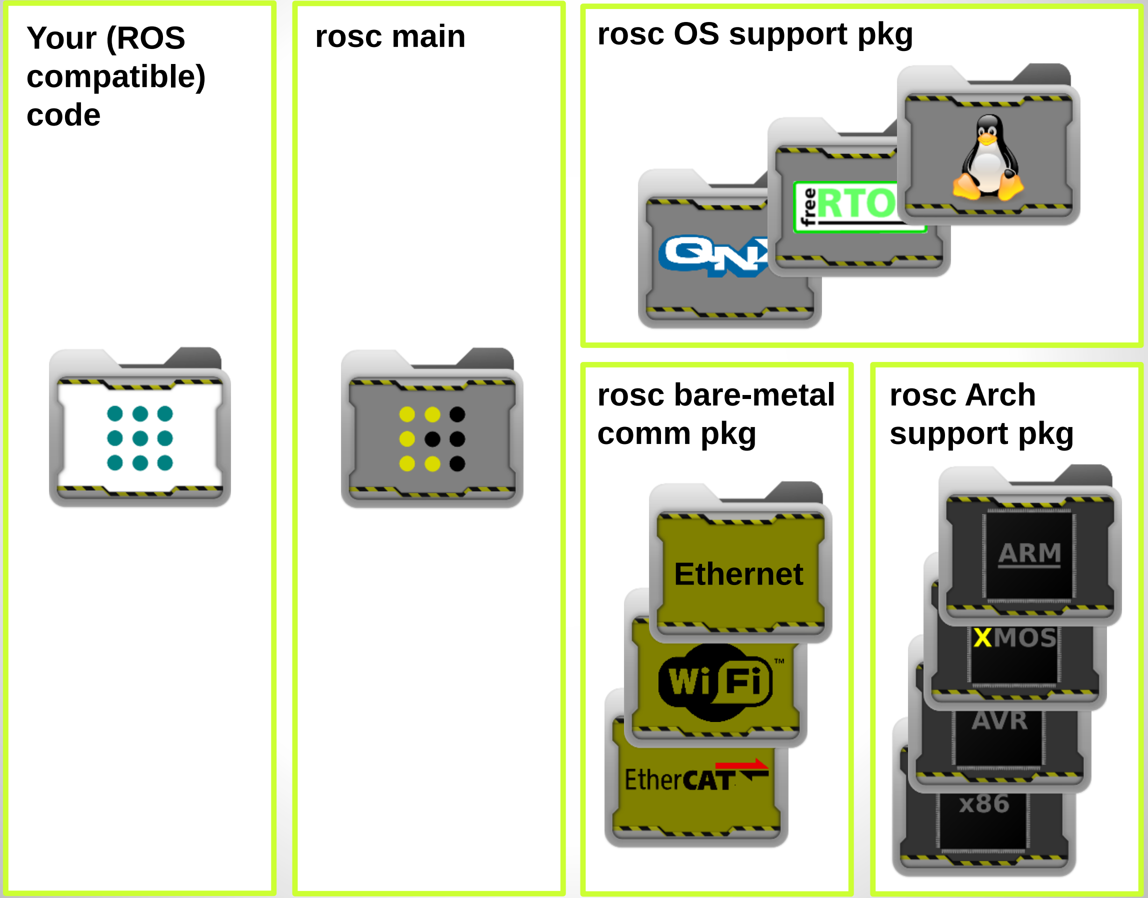
64KB RAM

Security Block (OTP ROM)

µROSnode & rosc

Aspect	µROSnode	rosc
Background	University project	Commercial open-source
Status	Working demos available	70% of first release
Long-term	No certain plans	Aims to be major ROS Industrial/Products client lib
Memory	Compact (1 MB reference device)	As small as possible (64 kB reference device)
OS	ChibiOS & POSIX reference	Bare-metal reference
Transport	OS driver (LWIP)	Modular transport layer concept

Components of rosc



Your (ROS compatible) code

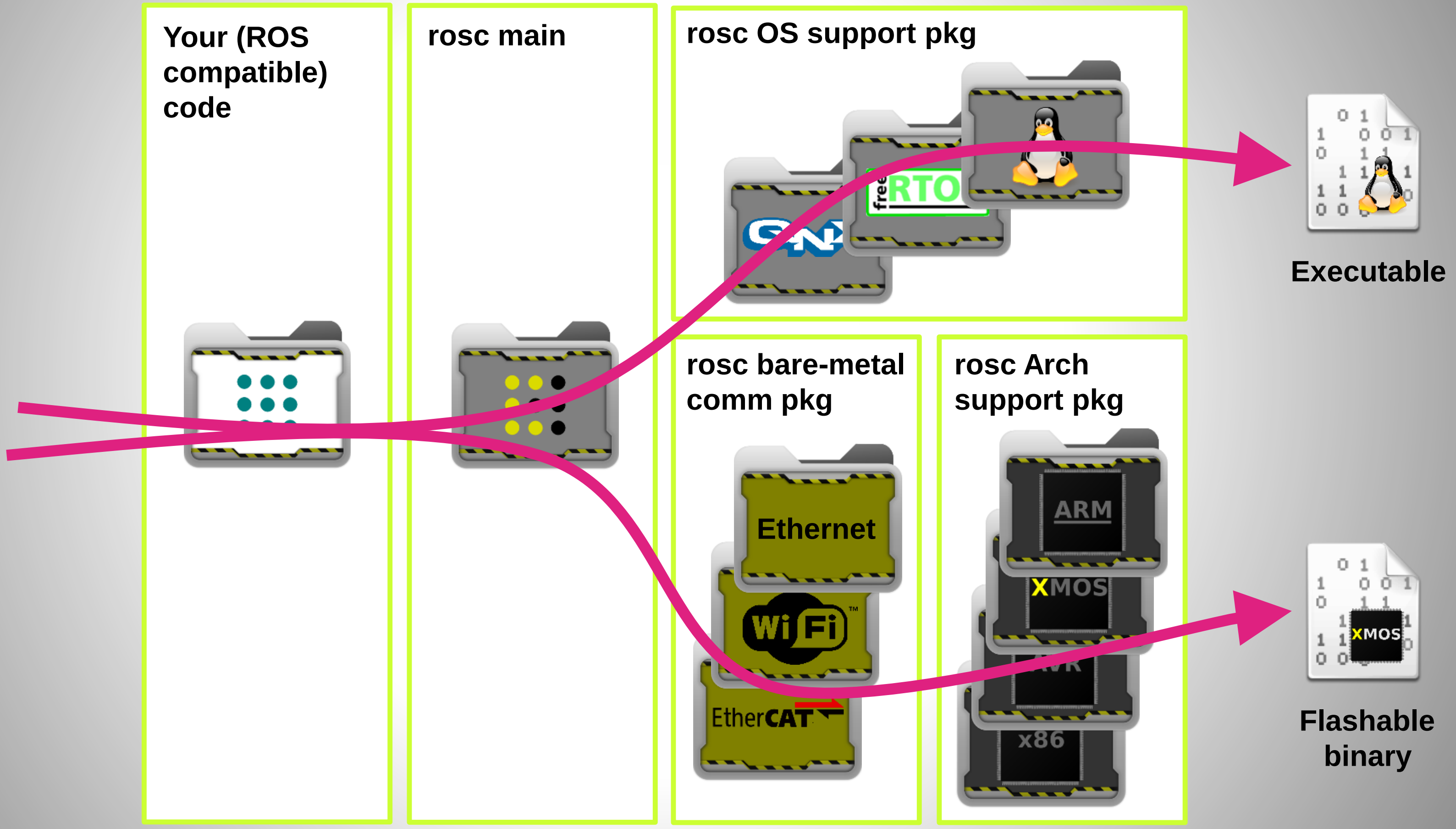
rosc main

rosc OS support pkg

rosc bare-metal comm pkg

rosc Arch support pkg

Building a node with rosc



Your (ROS compatible) code

rosc main

rosc OS support pkg

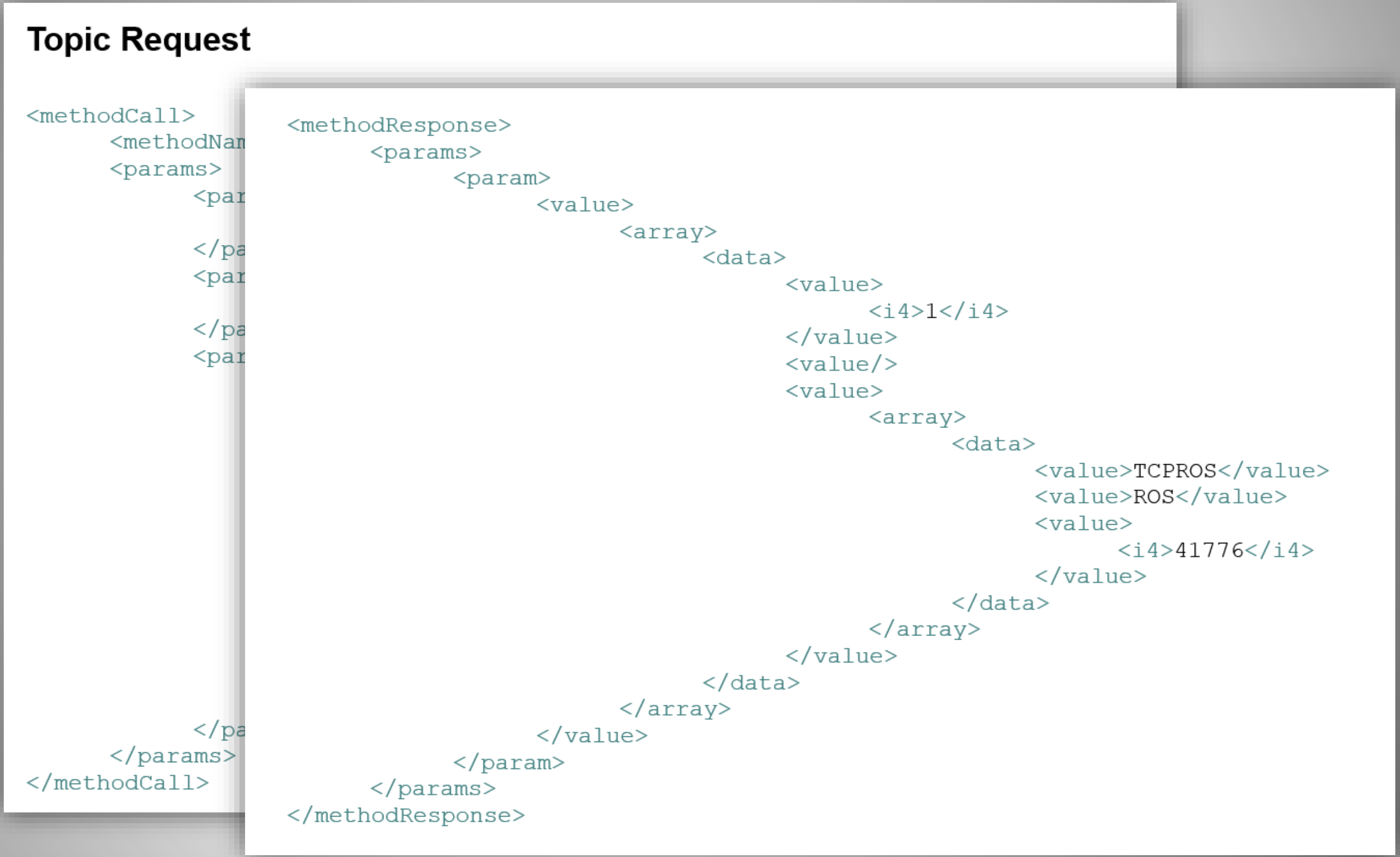
rosc bare-metal comm pkg

rosc Arch support pkg

Executable

Flashable binary

The XML-RPC pain



Topic Request

```
<methodCall>
    <methodNam
    <params>
        <par

        </pa
        <par

        </pa
        <par








        </pa
    </params>
</methodCall>
```

```
<methodResponse>
    <params>
        <param>
            <value>
                <array>
                    <data>
                        <value>
                            <i4>1</i4>
                        </value>
                        <value/>
                        <value>
                            <array>
                                <data>
                                    <value>TCPROS</value>
                                    <value>ROS</value>
                                    <value>
                                        <i4>41776</i4>
                                    </value>
                                </data>
                            </array>
                        </value>
                    </data>
                </array>
            </value>
        </param>
    </params>
</methodResponse>
```

The XML-RPC pain

XML-RPC

What common developers think about it

What embedded developers think about it

Current features of rosc (1)

HTTP/XML Parser

- Size: < 10 kB (on XMOS)
- Time for parsing: 139µs
  (publisherUpdate msg on XMOS, 125MHz, 100 byte buffer)
- Features/Limitations:
  - **Streaming Parser**
    (parsing on demand, fully variable buffersize 1 Byte - XXX Bytes)
  - **Almost fully validating**
    (unknown tags can not be validated)
  - **Does not support any encoding**
    (e.g. gzip, due to it's streaming nature)
  - **Maximum depth for tags: 20**
    (can be set to any value by #define)

Current features of rosc (2)

XML-RPC Message Generator

- Size: < 7 kB (on XMOS)

Port Interface Handling (Services, Topics, XML-RPC)

- Limitations on bare-metal systems:
  Fixed amount of ports and thus a limited amount
  of possible connections

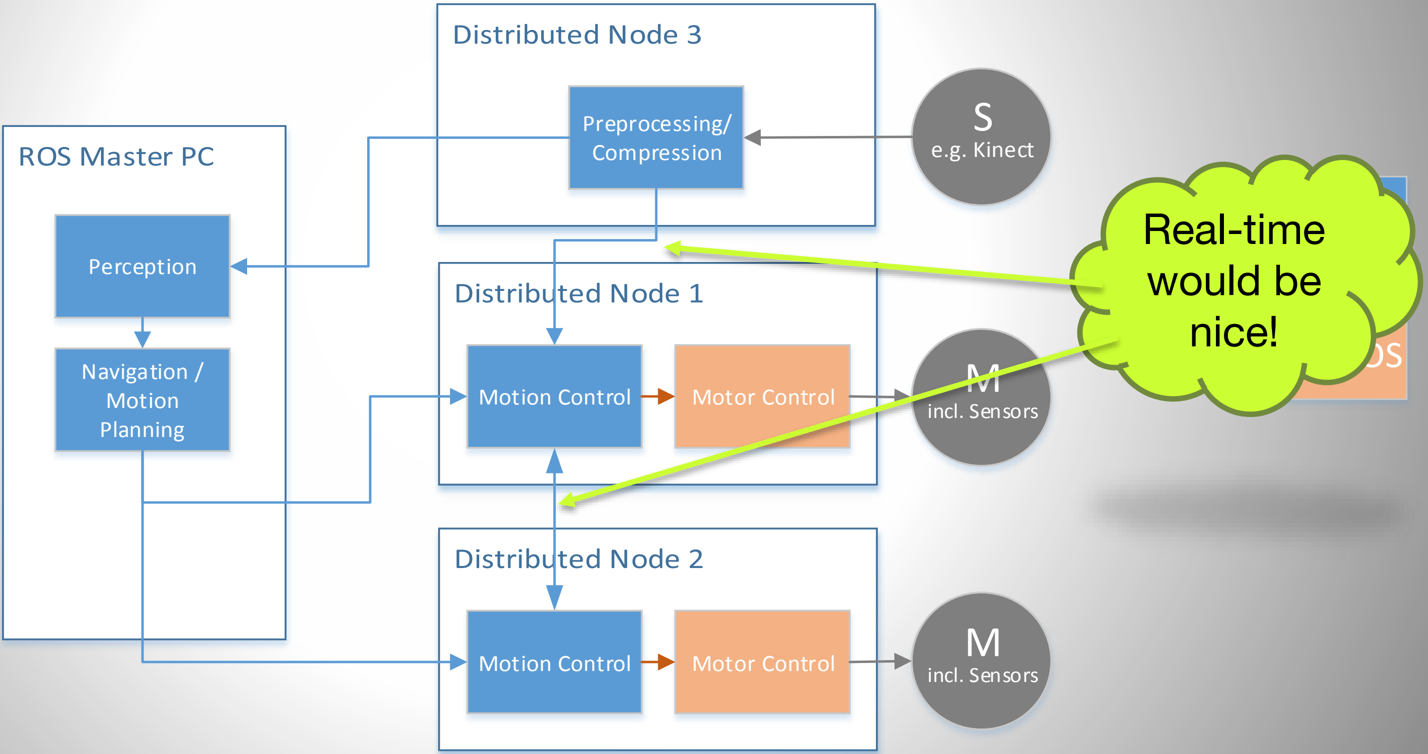
ROS msg Header/Source Generation

TCPROS (Un-)Marshalling

+ *Future dev towards ROS Industrial/Products*

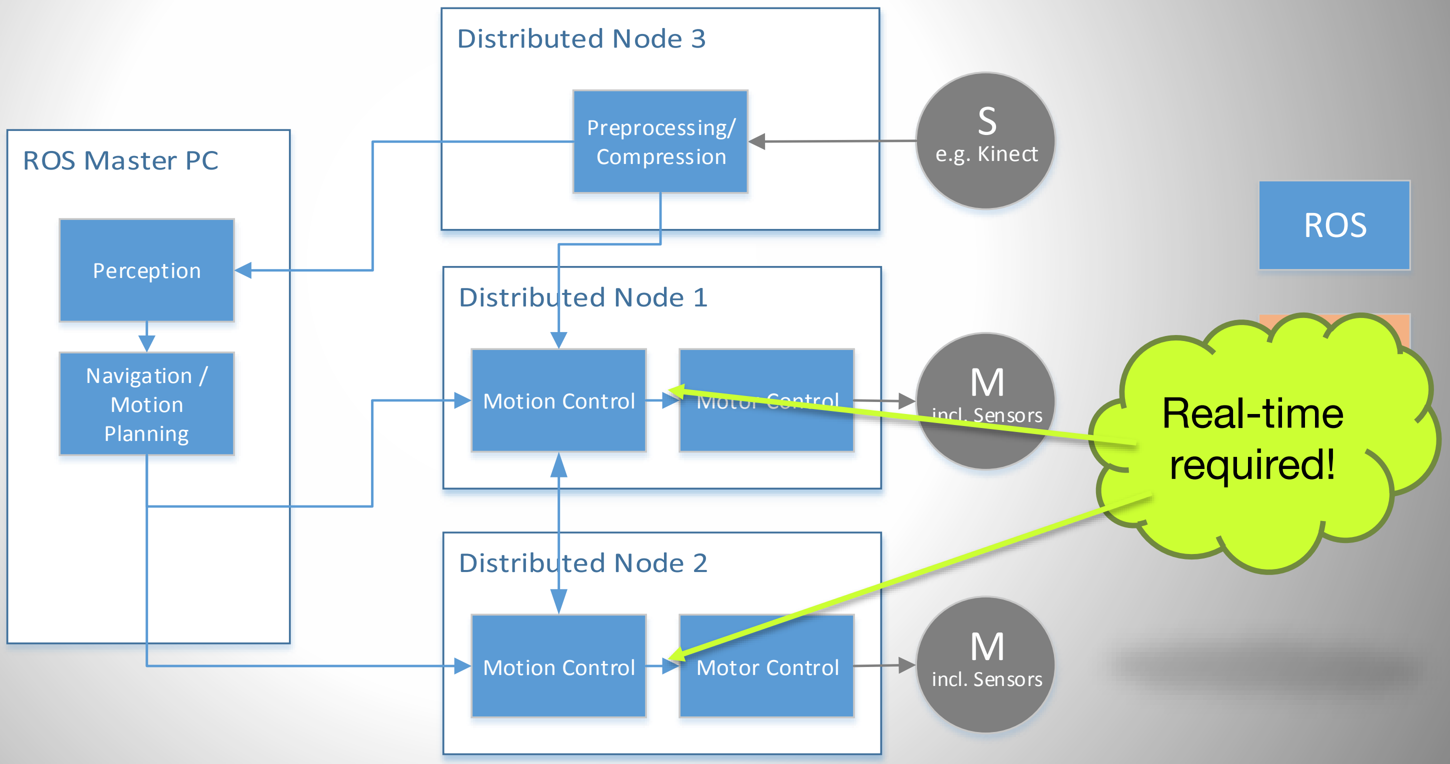
Steps towards Industrial/Product ROS

Solution requested (and delivered by rosc & µROSnode)



Steps towards Industrial/Product ROS

ROS on all levels (not possible using today‘s ROS!)



Requirements for a future-proof (Embedded) ROS

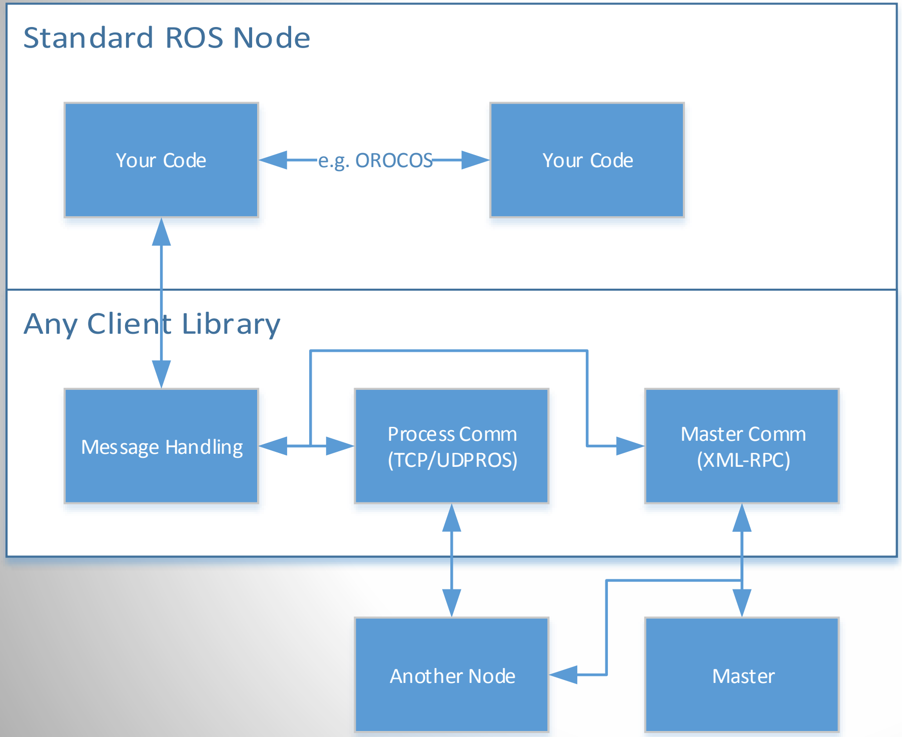
- Transport-independent (TCP/IP not required)
- Standard application protocols & reference architectures (Compatibility)
- Real-time capable (msg queues, transport)
- Replacement of XML-RPC (by JSON, or even ROSRPC?…)
- No master anymore (at least multi-master support)
- Model-based toolchain support
- Long-term support by foundation & suppliers
- Quality metrics & automated QA process

Requirements for a future-proof (Embedded) ROS

- Transport-independent (TCP/IP not required)
- Standard application protocols & reference archi (Compatibility)
- Real-time capable (msg queues, transport)
- Replacement of XML-RPC (by JSON, or even ROSRPC?…)
- No master anymore (at least multi-master support)
- Model-based toolchain support
- Long-term support by foundation & suppliers
- Quality metrics & automated QA process

Long-term targets of rosc

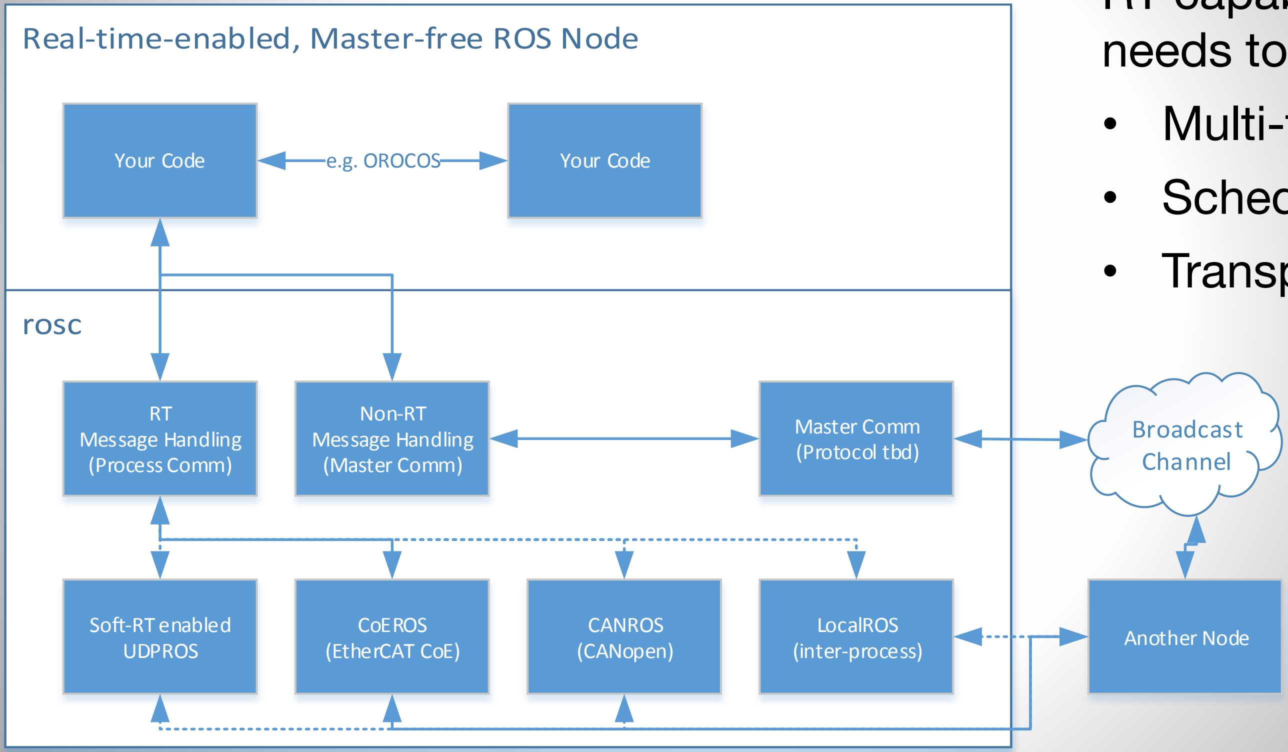
Draft for a RT-capable, master-free rosc



Mainly ROS is not RT-capable because of:

- IP transport
- Client implementations not supporting RTOS mechanisms

Draft for a RT-capable, master-free rosc



RT-capable ROS MW needs to support RT

- Multi-threading
- Scheduling
- Transport

www.ros.org/wiki/rosc

Free beer!

ROS Industrial & Products Meet-up @ Columbus, Sunday 7 pm

ros

synapticon.com

Synapticon GmbH, Hohlbachweg 2, 73344 Gruibingen, Germany

Contacts: Nikolai Ensslen, nensslen@synapticon.com, +49 7335 186 999 16
Christian Holl, choll@synapticon.com, +49 7335 186 999 11