Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Dr. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

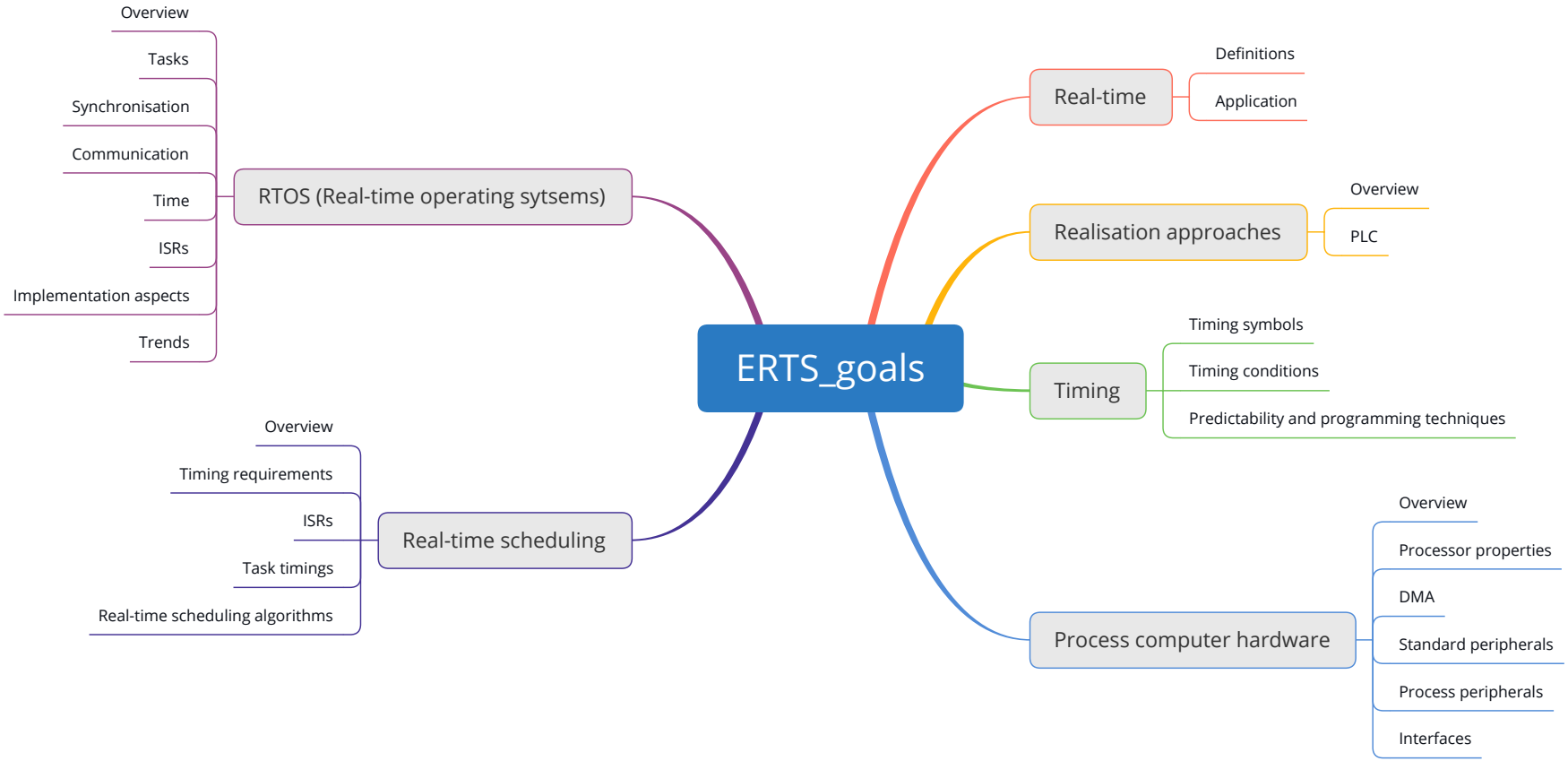
ERTS - Embedded real-time systems

ERTS 11 – ROS2

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

ERTS::ROS2

- ROS2 intro
- ROS2 communication
- ROS2 elements
- ROS2 code sample
- ROS2 workspace and build system
- Turtlebot3

CAMPUS Rosenheim
Computer Science

Technische Hochschule **Rosenheim**
Technical University of Applied Sciences

ROS2 - overview

The **Robot Operating System (ROS)** is a

- set of **software libraries** and **tools**
- for building **robot** applications

History:

- ROS 1 – 2007 (started)
  - Single point of failure (the roscore)
  - Lack of security
  - No real-time support
- ROS 2 – 2015
  - Multi-robot system
  - Multi-platform (Linux, Windows, MAC, RTOS, ...)
  - Real-time support
  - Network connection with QoS (quality of service)
  - Production environments

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 - releases

Release	Release date	Logo	EOL date
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023

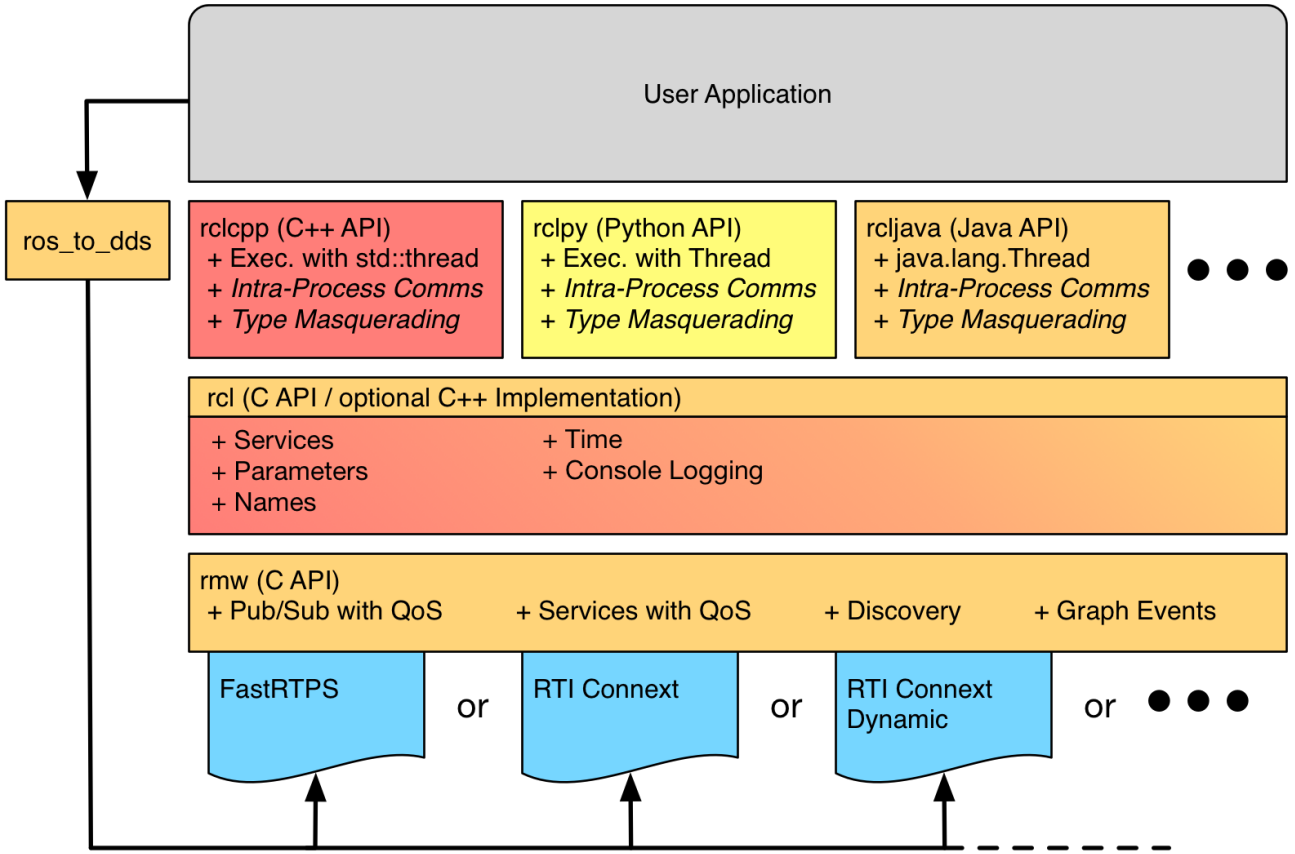
[source: https://docs.ros.org]

...

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 - architecture



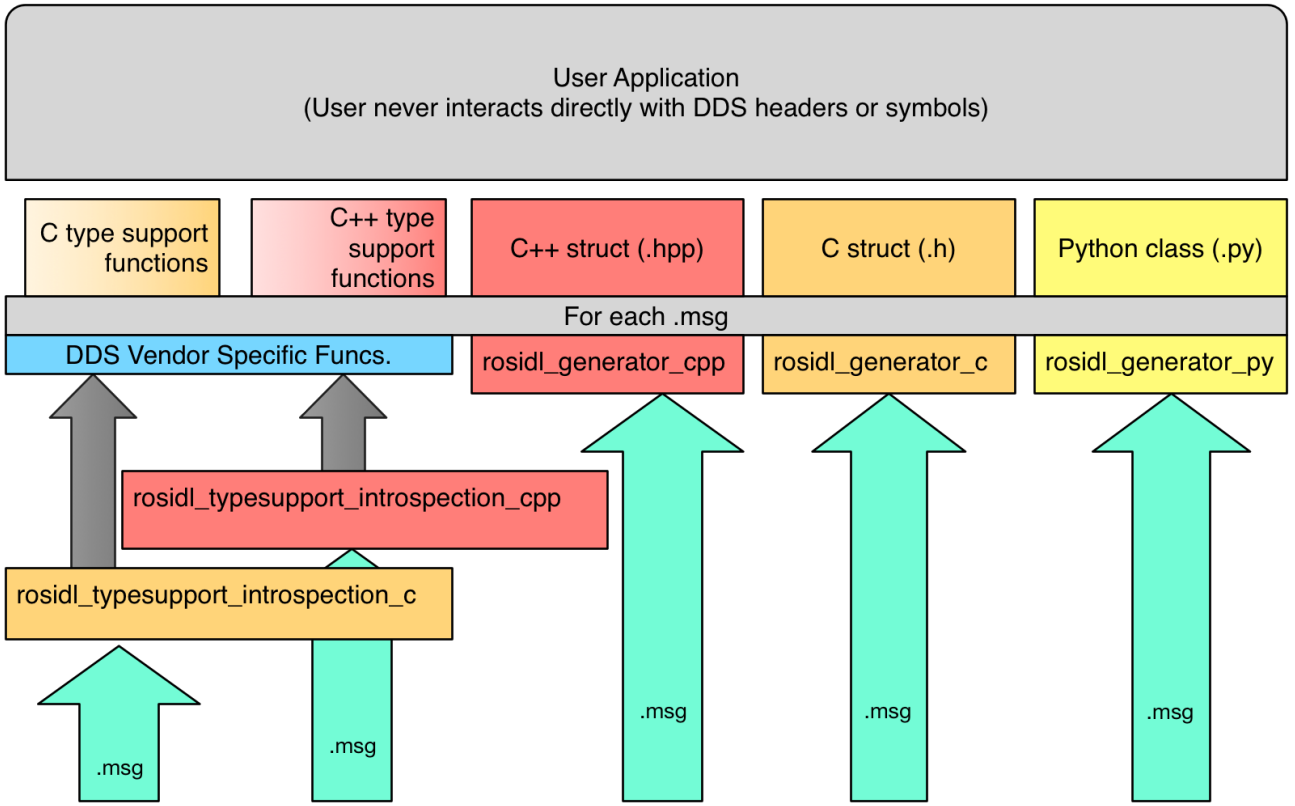
* *Intra-Process Comms* and *Type Masquerading* could be implemented in the client library, but may not currently exist.

[source: https://docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 - dynamic type support

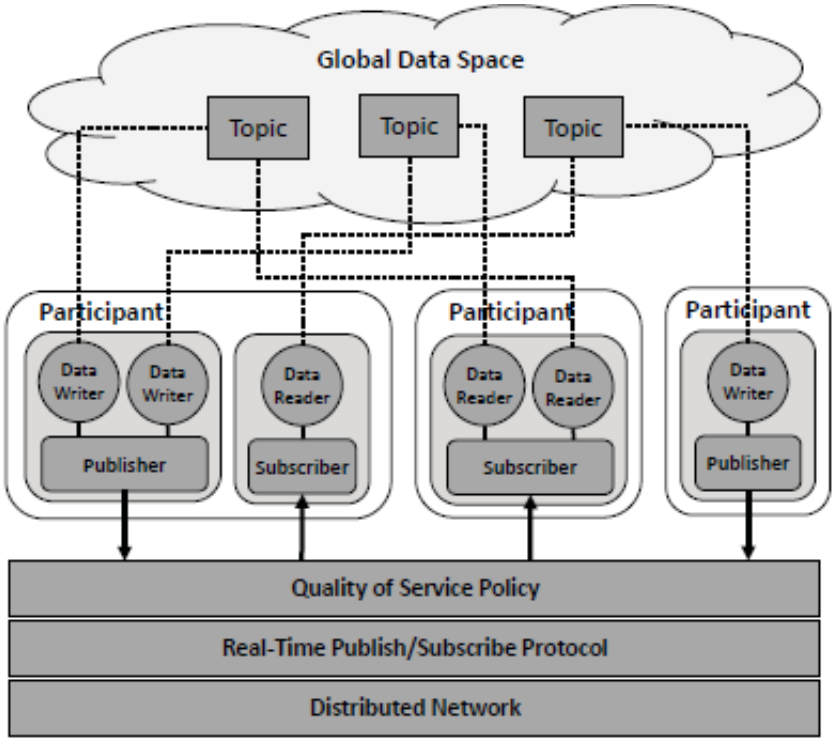


[source: https://docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 DDS communication



[source: etn-sas.eu]

The **Data Distribution Service (DDS)**

- is a **middleware**
- for **machine-to-machine**
- **communication**, using the
- **publish-subscribe** pattern

Properties:

- dependable
- high-performance
- interoperable
- real-time
- scalable data

ROS ID:

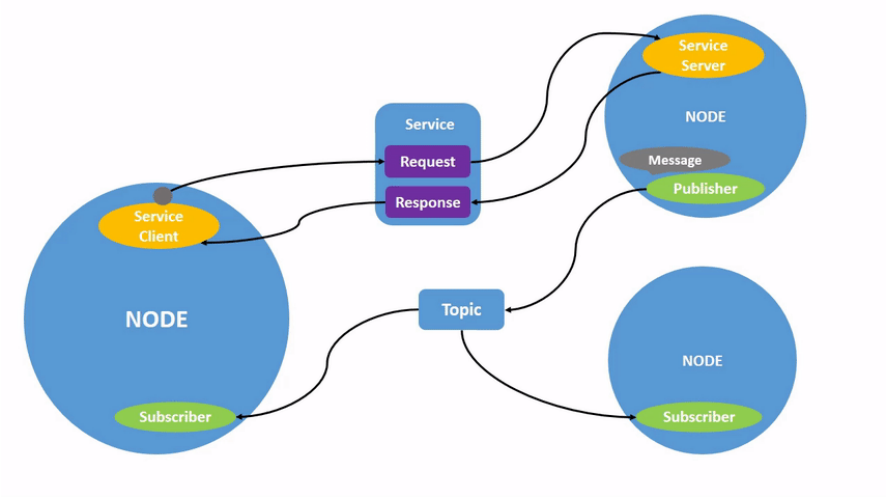
- ROS domain id
- Used for discovery and communication
- Automatic discovery mechanism

[source: docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 elements – node



[source: https://docs.ros.org]

Nodes:

- Send/receive data to/from
  - Topics
  - Services
  - Actions
  - Parameters
- Executable: can contain one or more nodes

Some useful commands:

- `ros2 node list`
- `ros2 node info <node_name>`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 elements – node

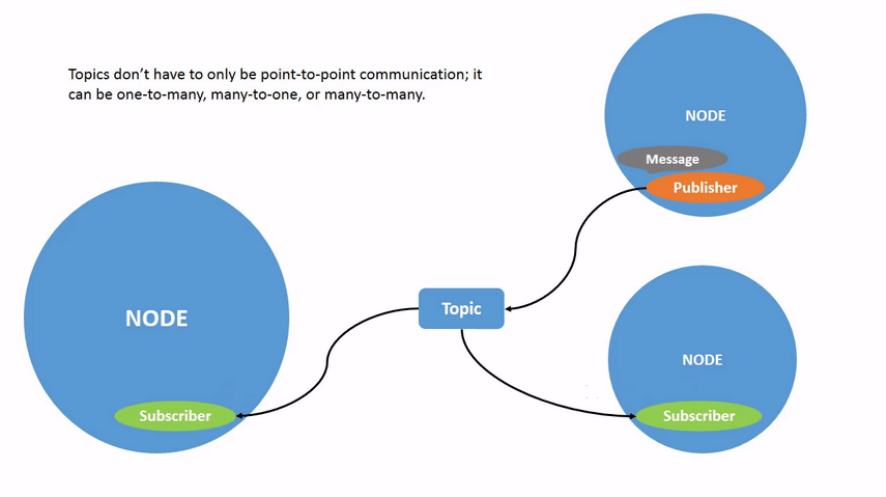
Example code:

```
1  #include <memory>
2  #include <string>
3
4  #include "rclcpp/rclcpp.hpp"
5
6  class ExampleNode : public rclcpp::Node
7  {
8  public:
9      ExampleNode()
10         : Node("example_node")
11     {
12
13     }
14 };
15
16 int main(int argc, char * argv[])
17 {
18     rclcpp::init(argc, argv);
19     rclcpp::spin(std::make_shared<ExampleNode>());
20     rclcpp::shutdown();
21
22     return 0;
23 }
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 elements – topic



[source: https://docs.ros.org]

Topics:

- Act as a bus for nodes to exchange messages
- **Publishers send** messages
- **Subscribers receive** messages
- Multiple publishers possible
- Multiple subscribers possible

Some useful commands:

- `ros2 topic list`
- `ros2 topic list -t`
- `ros2 topic echo <topic_name> #live`
- `ros2 topic info <topic_name>`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 elements – message

Example: `geometry_msgs/msg/Twist.msg`

```
# This expresses velocity in free space
# broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

Basic field types:

Type name	C++	Python	DDS type
bool	**bool**	builtins.bool	boolean
byte	**uint8_t**	builtins.bytes*	octet
char	**char**	builtins.str*	char
float32	**float**	builtins.float*	float
float64	**double**	builtins.float*	double
int8	**int8_t**	builtins.int*	octet
uint8	**uint8_t**	builtins.int*	octet
int16	**int16_t**	builtins.int*	short
uint16	**uint16_t**	builtins.int*	unsigned short
int32	**int32_t**	builtins.int*	long
uint32	**uint32_t**	builtins.int*	unsigned long
int64	**int64_t**	builtins.int*	long long
uint64	**uint64_t**	builtins.int*	unsigned long long
string	std::string	builtins.str	string
wstring	std::u16string	builtins.str	wstring

[source: https://docs.ros.org]

Message:

- Contains one or more members
- Language independent definition
- File: `message.msg`
- Compiled into the target language: C++, python, java

Some useful commands:

- `ros2 interface show <message>`
- `ros2 interface list`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 elements – parameters

Example code:

```
1  #include <memory>
2  #include <string>
3
4  #include "rclcpp/rclcpp.hpp"
5
6  class ExampleNode : public rclcpp::Node
7  {
8  public:
9      ExampleNode()
0          : Node("example_node")
1      {
2          this->declare_parameter("velocity", 0.9);
3      }
4
5      void some_function()
6      {
7          double velocity = this->get_parameter("velocity").as_double();
8          //work with velocity
9      }
0  };
```

Parameters can be set:

- from a launch file during startup
- from command line on startup
  `ros2 run ... --ros-args --params-file <path_to_yaml>`
- during runtime through `ros2 param...`

Some useful commands:

- `ros2 param set /<node> <param> <value>`
- `ros2 param get /<node> <param>`
- `ros2 param list <node>`
- `ros2 param dump /<node> --print`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 code sample

Code sample for publisher and subscriber

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Example: subscriber

```cpp
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
using std::placeholders::_1;

class MinimalSubscriber : public rclcpp::Node
{
public:
    MinimalSubscriber()
        : Node("minimal_subscriber")
    {
        subscription_ = this->create_subscription<std_msgs::msg::String>(
            "topic", 10, std::bind(&MinimalSubscriber::topic_callback, this, _1));
    }

private:
    void topic_callback(const std_msgs::msg::String::SharedPtr msg) const
    {
        RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg->data.c_str());
    }

private:
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
};
```

[source: https://docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Example: publisher

```
1  using namespace std::chrono_literals;
2
3  class MinimalPublisher : public rclcpp::Node
4  {
5  public:
6      MinimalPublisher()
7          : Node("minimal_publisher"), count_(0)
8      {
9      publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
10     timer_ = this->create_wall_timer(500ms, std::bind(&MinimalPublisher::timer_callback, this
11     }
12
13 private:
14     void timer_callback()
15     {
16         auto message = std_msgs::msg::String();
17         message.data = "Hello, world! " + std::to_string(count_++);
18         RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
19         publisher_->publish(message);
20     }
21 private:
22     rclcpp::TimerBase::SharedPtr timer_;
23     rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
24     size_t count_;
25 };
```

[source: https://docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 workspace

```
1 workspace_dir/
2               /build
3               /install
4               /log
5               /src
6                   /package1
7                   /package2
8                   /...
```

Workspace:

- Contains all files
- `/build`: temporary files for build
- `/install`: installed binaries, libraries, resources
- `/log`: log files
- `/src`: source packages

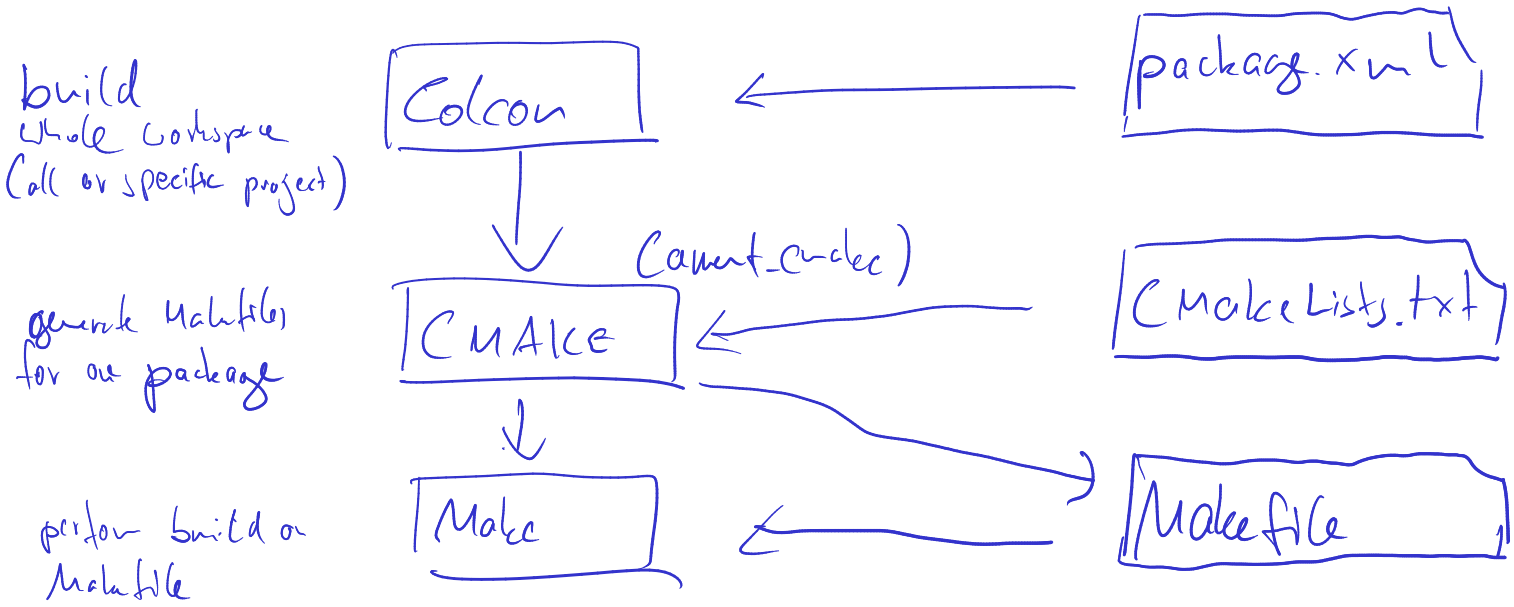
Common commands

- Build all: `colcon build --symlink-install`
- Run all tests: `colcon test`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 build system



[source: https://docs.ros.org]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ROS2 C++ package

Package in: `workspace_dir/src/`

```
1  my_package/
2            /package.xml
3            /CMakeLists.txt
4
5            /include/
6                /class1.hpp
7            /src/
8                /class1.cpp
9                /node_main.cpp
10           /msg/
11               /CustomMsg.msg
12           /action/
13               /CustomAction.action
14           /srv/
15               /CustomSrv.srv
16           /param/
17           /launch/
18           /map/
19           /rviz/
```

[source: https://docs.ros.org]

C++ cmake based package

- `/include`: Headers and sources
- `/src`: Headers and sources
- `/msg`: Messages
- `/action`: Actions
- `/srv`: Services
- `/param`: Parameters
- `/launch`: Launch scripts
- `/map`: Maps
- `/rviz`: 3D visualization

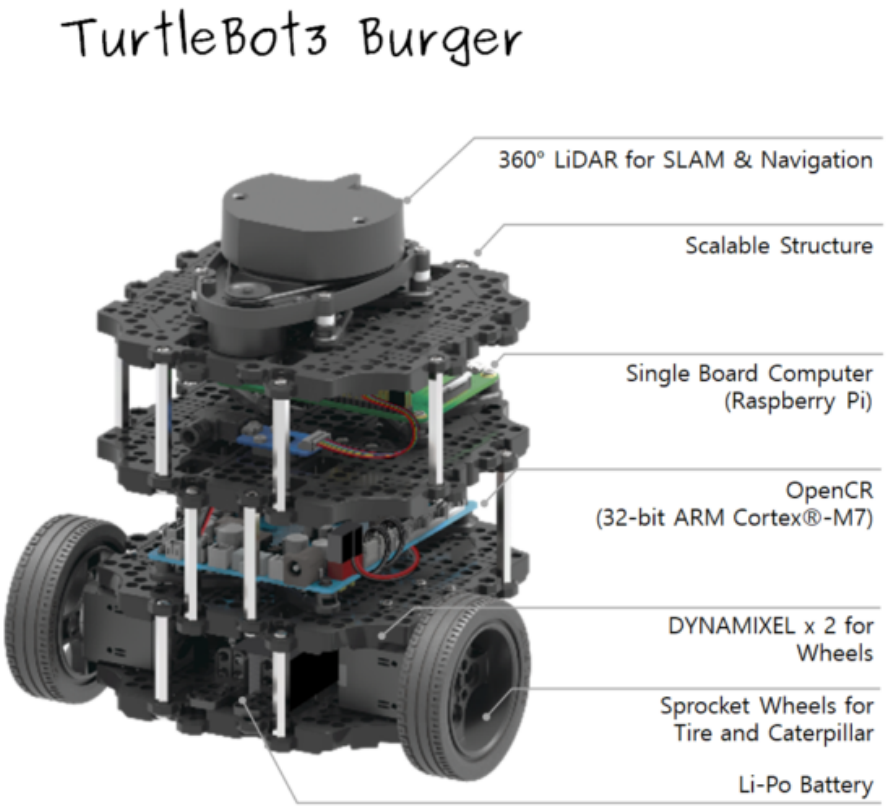
Common commands

- Create package: `ros2 pkg create --build-type ament_cmake <pkg_name> --dependencies ...`
- Build package: `colcon build --packages-select my_package --symlink-install`
- Run executable: `ros2 run my_package my_package_executable1`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Turtlebot3 - overview



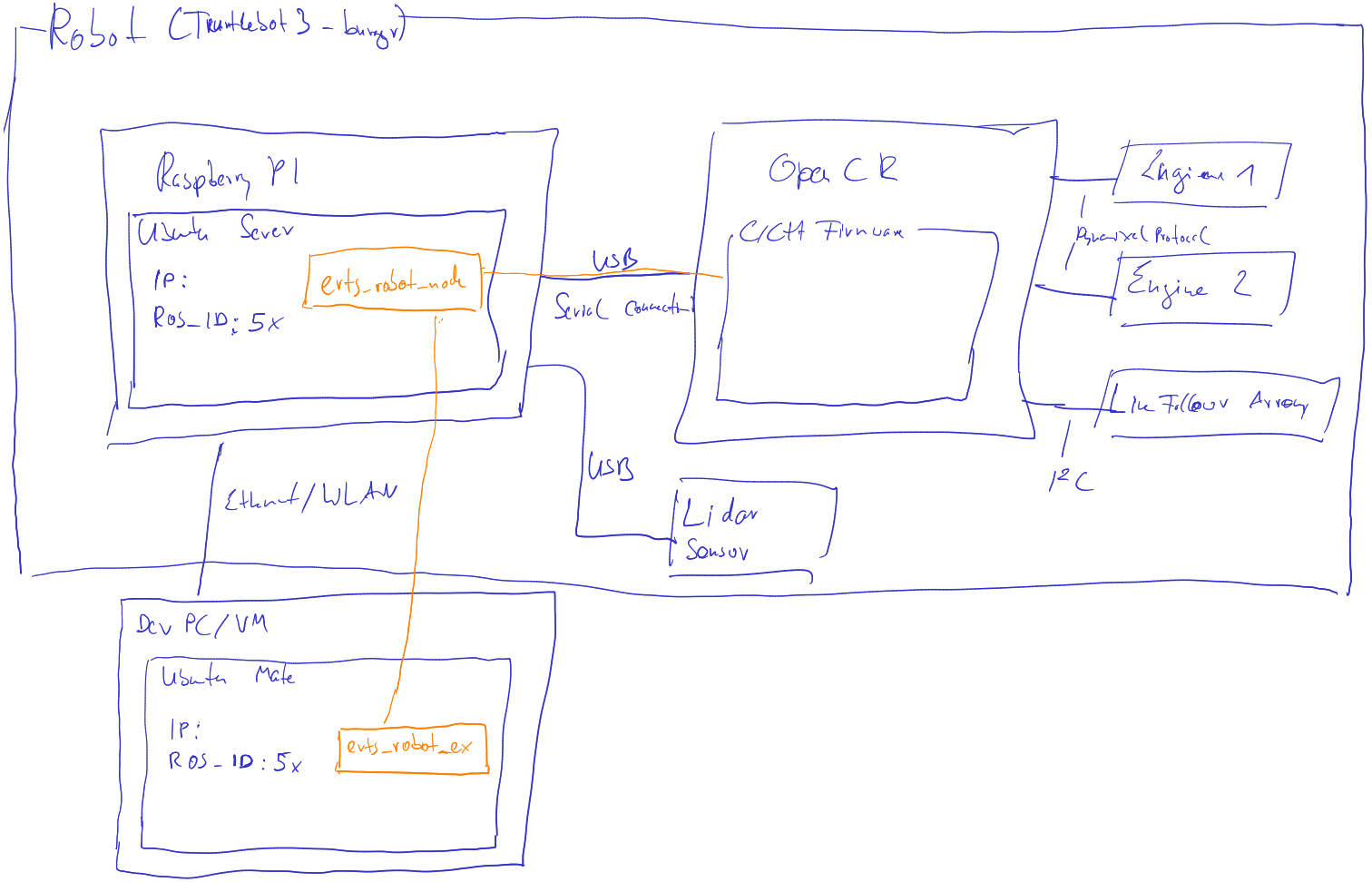
Components:

- Boards:
  - Raspberry Pi 3b plus
  - HW Driver: OpenCR 1.0
- Sensors:
  - Lidar: LDS-01 (HLS-LFCD2)
  - Line follower Array
- Motors: DYNAMIXEL (XL430-W250-T)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Turtlebot3 - details



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- ROS2 intro
- ROS2 communication
- ROS2 elements
- ROS2 code sample
- ROS2 workspace and build system
- Turtlebot3