EEROS with Gazebo

Agenda

1. Task
2. Toolchain
3. Changes in ROS-2
    - 3.1 Code Changes
    - 3.2 New Executor
4. Timedomain
5. Demo
6. Questions

1. Task

Eine bestehende Regelung für einen Motor soll wahlweise auf realer Hardware und in einer passenden Simulationsumgebung laufen gelassen werden können. Ein spezielles Augenmerk soll auf einer nahtlosen Integration in das bestehende EEROS Framework gelegt werden und mit passenden Werkzeugen soll alles möglichst einfach und automatisiert gestartet und demonstriert werden können. Eine komplexere Applikation mit einem kleinen Delta-Roboter soll als zweites Demonstrationsobjekt dienen.

2. Toolchain

ROS-1 Noetic

- Released 2020
- EOL 2025
- Last Release

- Ubuntu-20.04
  - EOL 2025
- Gazebo 11.0
- EEROS

ROS-2 Humble

- Released 2022
- EOL 2027
- Ongoing

- Ubuntu 22.04
  - EOL 2027
- Gazebo 11.0
- EEROS

ROS-1 actually is EOL therefore I have to use ROS-2

EEROS is not working with ROS-2

3. Changes in ROS-2

- Messages have moved into a Sub-Namespace (msg)
- Includes are snake_case and not CamelCase
- ROS-2 is more Object-Oriented
- C++ API is now RCLCPP and not ros.h anymore
- The whole Application spins and not a single node

3.1 Code Changes

```
#include <geometry_msgs/PointStamped.h>
#include <ros.h>


geometry_msgs::PointStamped point_stamped;
```

```
#include <geometry_msgs/msg/point_stamped.hpp>
#include <rclcpp/rclcpp.hpp>


geometry_msgs::msg::PointStamped point_stamped;
```

```
#include "ros/ros.h"
#include "std_msgs/String.h"


ros::init(argc, argv, "talker");
ros::NodeHandle node;
ros::Publisher chatter_pub =
  node.advertise<std_msgs::String>("chatter", 1000);
std_msgs::String msg("hello world");
while (ros::ok()) {
  chatter_pub.publish(msg);
  ros::spinOnce();
}
```

Type:
rclcpp::Node::SharedPtr

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"


rclcpp::init(argc, argv);
auto node = rclcpp::Node::make_shared("talker");
auto chatter_pub =
 node->create_publisher<std_msgs::msg::String>("chatter", 1000);
std_msgs::msg::String msg("hello world");
while (rclcpp::ok()) {
  chatter_pub->publish(msg);
  rclcpp::spin_some(node);
}
```

Name on the Node
Attention: Multiple nodes
with the same Name

Spinning on the node
and not the Application

3.2 New Executor

```
rclcpp::CallbackGroup::SharedPtr
Executor::registerSubscriber(rclcpp::Node::SharedPtr node, const bool sync) {

  syncWithRosTopicIsSet = sync;

  if (subscriberExecutor == nullptr) {

    subscriberExecutor =
rclcpp::executors::MultiThreadedExecutor::make_shared();

  }

  auto cb = node->create_callback_group(rclcpp::CallbackGroupType::Reentrant);

  subscriberExecutor->add_callback_group(cb, node->get_node_base_interface());

  return cb;

}
```

One Multithreaded ROS-Executor is used for all Subscribers

```
if (rclcpp::ok()) {

  rclcpp::SubscriptionOptionsWithAllocator<std::allocator<void>> options;

  options.callback_group = Executor::instance().registerSubscriber(handle,
syncWithTopic);

  subscriber = handle->create_subscription<TRosMsg>(topic, queueSize,
std::bind(&RosSubscriber::rosSubscriberCallback, this, _1), options);

  log.info() << "RosSubscriber, reading from topic: '" << topic << "' on
node '" << node->get_name() << "' created.";

}

.
```

Subscriber has to register in the EEROS-Executor

Every Subscriber needs a Callback-Group

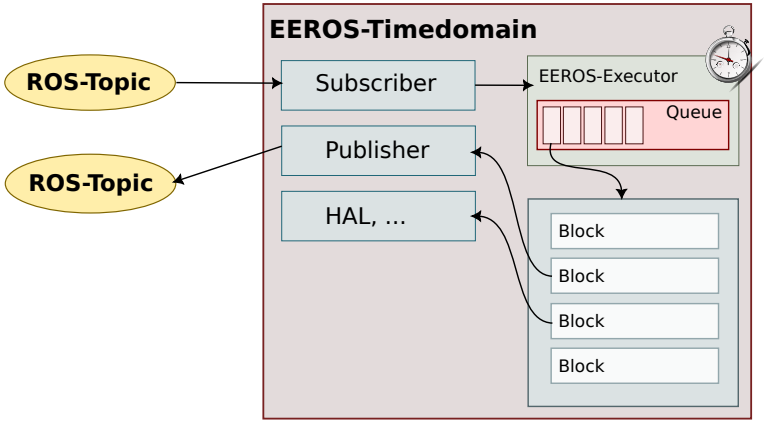
```
// Starts spinning subscribers in an own thread to not block the program

if (subscriberExecutor != nullptr) {

  subscriberThread = std::make_shared<std::thread>([this]() {

    log.info() << "Starting ROS-Executor for handling RosSubscription";

    subscriberExecutor->spin();

  });

}
```

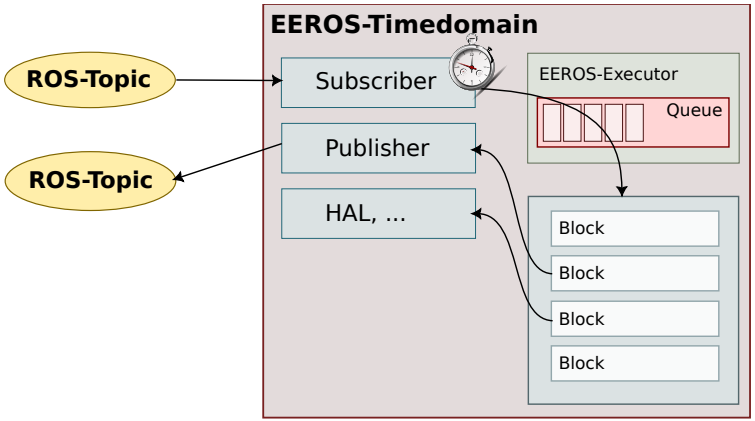
The ROS-Executor is starts spinning, not the node!

```
void rosSubscriberCallback(const TRosMsg& msg) {

  std::lock_guard<std::mutex> lock(queue_mutex);

  queue.push_back(std::move(msg));

  if (syncWithTopic) {

    Executor::instance().processTasks();

  }

}
```

Subscriber-Callback calls the EEROS-Executor on every message

4. Timedomain





5. Demo

6. Questions