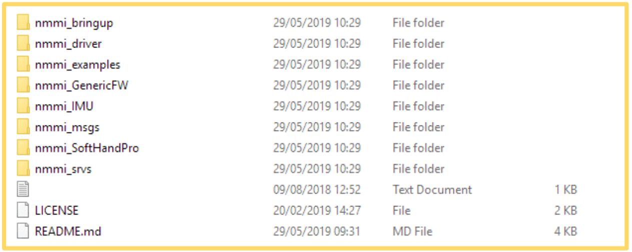
ROS node

Preliminary considerations

New ROS node that interfaces and derives from new QB ROS nodes

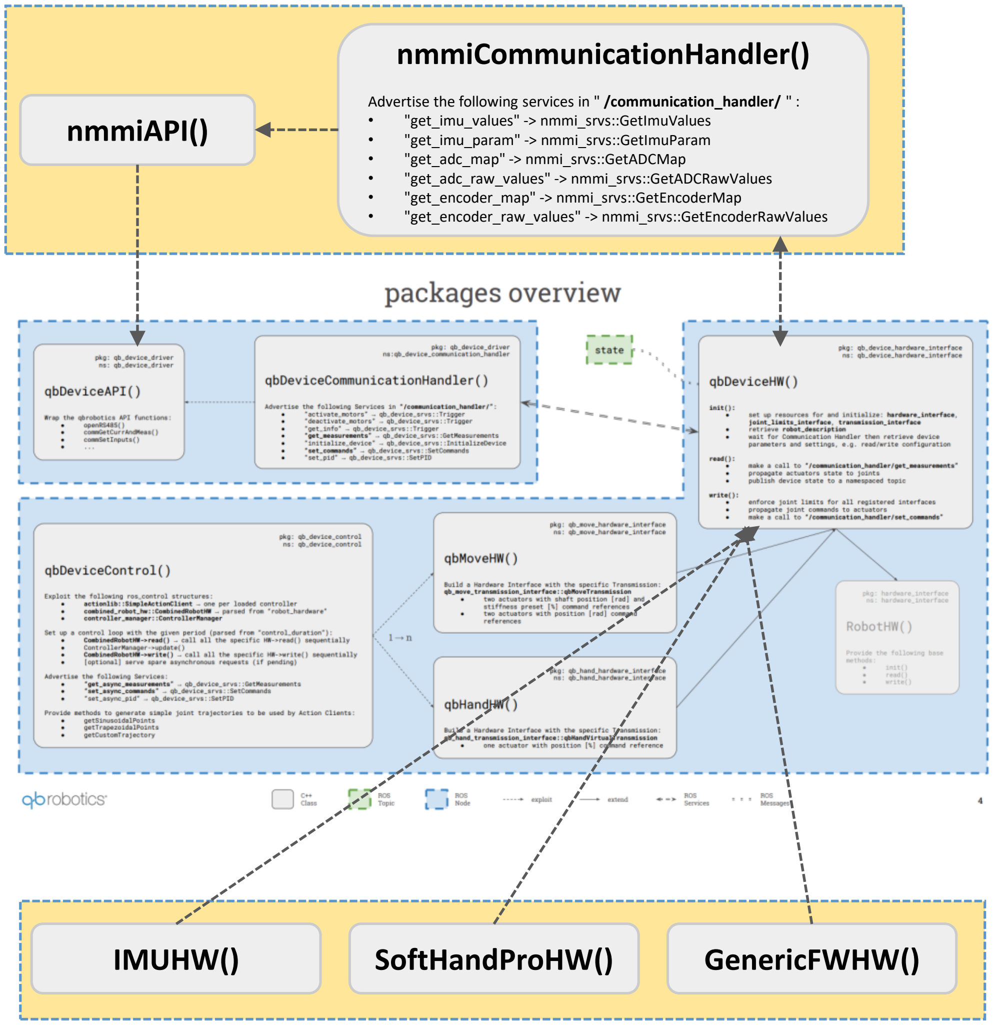
Since the need of handling IMU and generic FW features, new nodes have been implemented: 🟨

- IMU, SoftHandPro and GenericFW hardware interfaces
- NMMI communication handler to expand the QB one with the additional features
- NMMI custom messages, services and bringup files
- New states topic in addition to old ones



nmmi_bringup	29/05/2019 10:29	File folder	
nmmi_driver	29/05/2019 10:29	File folder	
nmmi_examples	29/05/2019 10:29	File folder	
nmmi_GenericFW	29/05/2019 10:29	File folder	
nmmi_IMU	29/05/2019 10:29	File folder	
nmmi_msgs	29/05/2019 10:29	File folder	
nmmi_SoftHandPro	29/05/2019 10:29	File folder	
nmmi_srvs	29/05/2019 10:29	File folder	
	09/08/2018 12:52	Text Document	1 KB
LICENSE	20/02/2019 14:27	File	2 KB
README.md	29/05/2019 09:31	MD File	4 KB

new developed files

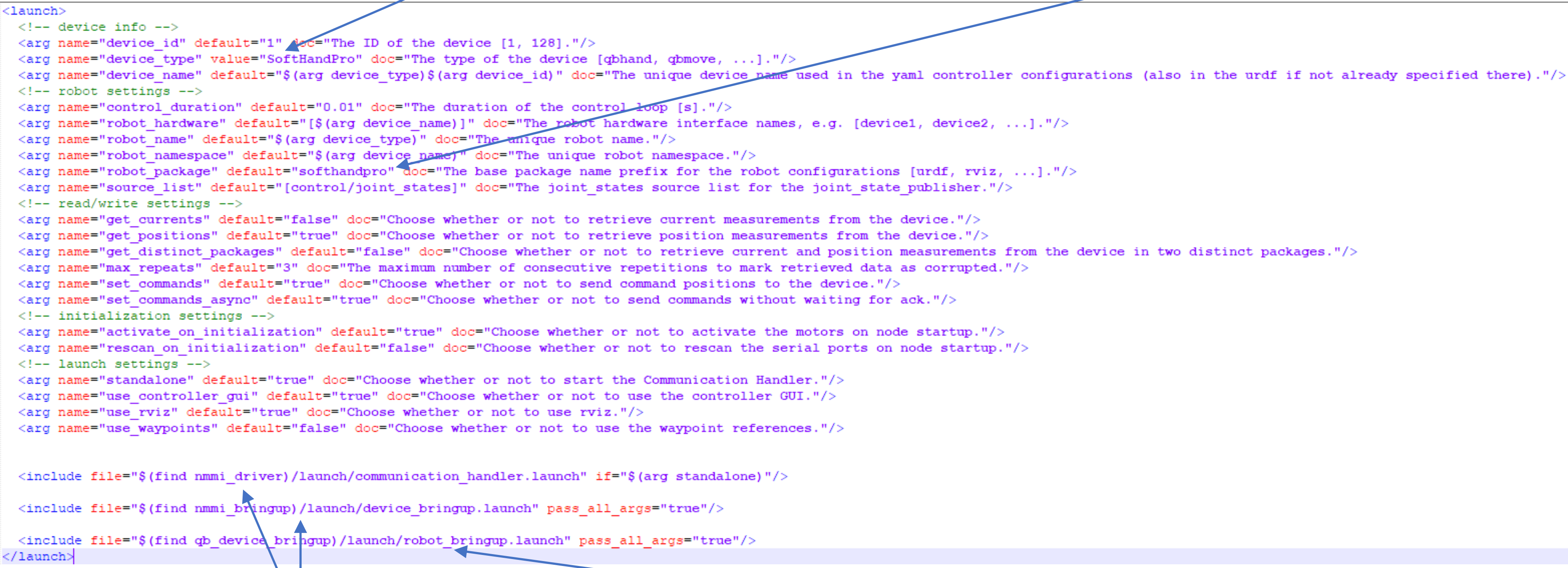

nmmiCommunicationHandler()
Advertise the following services in " /communication_handler/ " :
"get_imu_values" -> nmmi_srvs::GetImuValues
"get_imu_param" -> nmmi_srvs::GetImuParam
"get_adc_map" -> nmmi_srvs::GetADCMap
"get_adc_raw_values" -> nmmi_srvs::GetADCRawValues
"get_encoder_map" -> nmmi_srvs::GetEncoderMap
"get_encoder_raw_values" -> nmmi_srvs::GetEncoderRawValues
nmmiAPI()
packages overview
qbDeviceAPI()
qbDeviceCommunicationHandler()
state
qbDeviceHW()
qbDeviceControl()
qbMoveHW()
qbHandHW()
RobotHW()
1 → n
qbrobotics
IMUHW()
SoftHandProHW()
GenericFWHW()

SoftHand Pro device example

roslaunch nmmi_examples SoftHandPro_control.launch

device_type should be «*SoftHandPro*»

robot_package should be «*softhandpro*» in order to load correct urdf models



```
<launch>
  <!-- device info -->
  <arg name="device_id" default="1" doc="The ID of the device [1, 128]."/>
  <arg name="device_type" value="SoftHandPro" doc="The type of the device [qbhand, qbmove, ...]."/>
  <arg name="device_name" default="$(arg device_type)$(arg device_id)" doc="The unique device name used in the yaml controller configurations (also in the urdf if not already specified there)."/>
  <!-- robot settings -->
  <arg name="control_duration" default="0.01" doc="The duration of the control loop [s]."/>
  <arg name="robot_hardware" default="[$(arg device_name)]" doc="The robot hardware interface names, e.g. [device1, device2, ...]."/>
  <arg name="robot_name" default="$(arg device_type)" doc="The unique robot name."/>
  <arg name="robot_namespace" default="$(arg device_name)" doc="The unique robot namespace."/>
  <arg name="robot_package" default="softhandpro" doc="The base package name prefix for the robot configurations [urdf, rviz, ...]."/>
  <arg name="source_list" default="[control/joint_states]" doc="The joint_states source list for the joint_state_publisher."/>
  <!-- read/write settings -->
  <arg name="get_currents" default="false" doc="Choose whether or not to retrieve current measurements from the device."/>
  <arg name="get_positions" default="true" doc="Choose whether or not to retrieve position measurements from the device."/>
  <arg name="get_distinct_packages" default="false" doc="Choose whether or not to retrieve current and position measurements from the device in two distinct packages."/>
  <arg name="max_repeats" default="3" doc="The maximum number of consecutive repetitions to mark retrieved data as corrupted."/>
  <arg name="set_commands" default="true" doc="Choose whether or not to send command positions to the device."/>
  <arg name="set_commands_async" default="true" doc="Choose whether or not to send commands without waiting for ack."/>
  <!-- initialization settings -->
  <arg name="activate_on_initialization" default="true" doc="Choose whether or not to activate the motors on node startup."/>
  <arg name="rescan_on_initialization" default="false" doc="Choose whether or not to rescan the serial ports on node startup."/>
  <!-- launch settings -->
  <arg name="standalone" default="true" doc="Choose whether or not to start the Communication Handler."/>
  <arg name="use_controller_gui" default="true" doc="Choose whether or not to use the controller GUI."/>
  <arg name="use_rviz" default="true" doc="Choose whether or not to use rviz."/>
  <arg name="use_waypoints" default="false" doc="Choose whether or not to use the waypoint references."/>


  <include file="$(find nmmi_driver)/launch/communication_handler.launch" if="$(arg standalone)"/>

  <include file="$(find nmmi_bringup)/launch/device_bringup.launch" pass_all_args="true"/>

  <include file="$(find qb_device_bringup)/launch/robot_bringup.launch" pass_all_args="true"/>
</launch>
```

load both nmmi communication handler and nmmi device bringup file

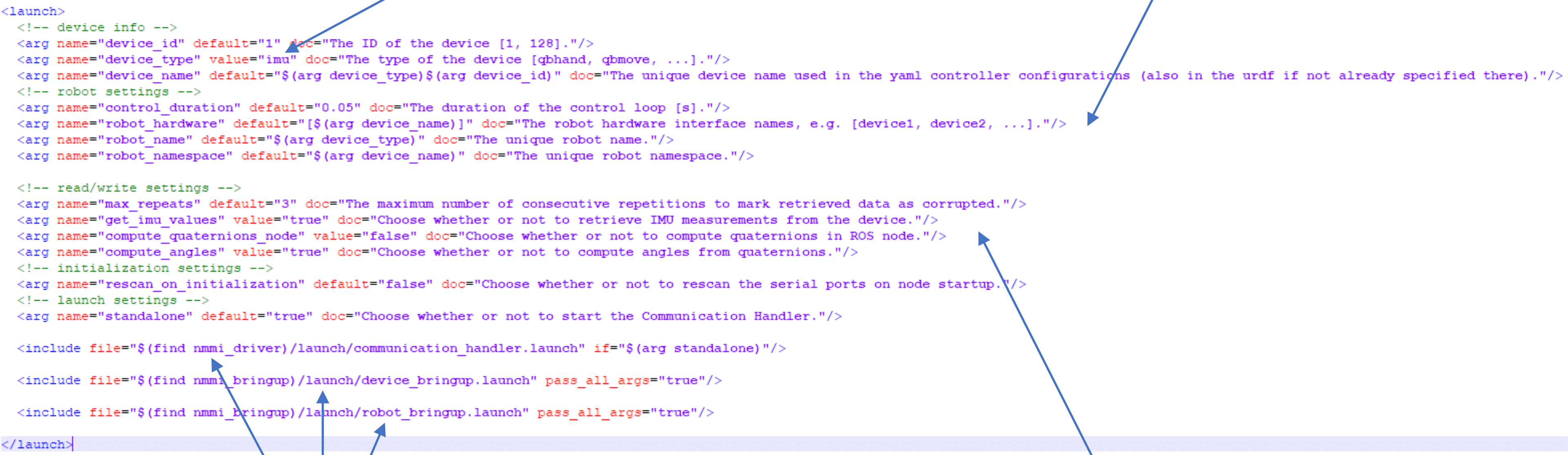
load robot bringup file from qb_device

IMU HW example

roslaunch nmmi_examples IMU_board_single.launch

device_type should be «*imu*»

robot_package is not needed since there are no description models and associated controls



```
<launch>
  <!-- device info -->
  <arg name="device_id" default="1" doc="The ID of the device [1, 128]."/>
  <arg name="device_type" value="imu" doc="The type of the device [qbhand, qbmove, ...]."/>
  <arg name="device_name" default="$(arg device_type)$(arg device_id)" doc="The unique device name used in the yaml controller configurations (also in the urdf if not already specified there)."/>
  <!-- robot settings -->
  <arg name="control_duration" default="0.05" doc="The duration of the control loop [s]."/>
  <arg name="robot_hardware" default="[$(arg device_name)]" doc="The robot hardware interface names, e.g. [device1, device2, ...]."/>
  <arg name="robot_name" default="$(arg device_type)" doc="The unique robot name."/>
  <arg name="robot_namespace" default="$(arg device_name)" doc="The unique robot namespace."/>

  <!-- read/write settings -->
  <arg name="max_repeats" default="3" doc="The maximum number of consecutive repetitions to mark retrieved data as corrupted."/>
  <arg name="get_imu_values" value="true" doc="Choose whether or not to retrieve IMU measurements from the device."/>
  <arg name="compute_quaternions_node" value="false" doc="Choose whether or not to compute quaternions in ROS node."/>
  <arg name="compute_angles" value="true" doc="Choose whether or not to compute angles from quaternions."/>
  <!-- initialization settings -->
  <arg name="rescan_on_initialization" default="false" doc="Choose whether or not to rescan the serial ports on node startup."/>
  <!-- launch settings -->
  <arg name="standalone" default="true" doc="Choose whether or not to start the Communication Handler."/>

  <include file="$(find nmmi_driver)/launch/communication_handler.launch" if="$(arg standalone)"/>

  <include file="$(find nmmi_bringup)/launch/device_bringup.launch" pass_all_args="true"/>

  <include file="$(find nmmi_bringup)/launch/robot_bringup.launch" pass_all_args="true"/>

</launch>
```

load all communication handler, device bringup file and robot bringup from nmmi side

Configure the node to read IMU values, compute quaternions by ROS on the node or not and compute also angles if needed

Ready to use examples

roslaunch nmmi_examples file.launch

- *IMU_board_single.launch*:
  configured for a board or device able to read IMU (see slide 14)
- *2_IMU_boards_chain.launch*:
  configuration with 2 IMU boards with different IDs
- *SoftHandPro_control.launch*:
  configured for using with a SoftHand Pro NMMI device (see slide 13)
- *generic_board_single.launch*:
  used together with generic FW to read raw ADC values and/or Encoders
- *SoftHandPro_IMU_chain.launch*:
  chain of a SoftHand Pro and a IMU board with different IDs
- *qbhand_IMU_chain.launch*:
  chain of a qbHand and a IMU board with different IDs
- *SoftHandPro_with_IMU_reading.launch*:
  configured to be used with a SoftHand Pro NMMI device with its own on board IMU reading active
- *SoftHandPro_with_generic_features.launch*:
  Used together with generic FW but with device type parameter set to "SOFTHAND PRO", so it's a SoftHand Pro NMMI device but with generic FW features