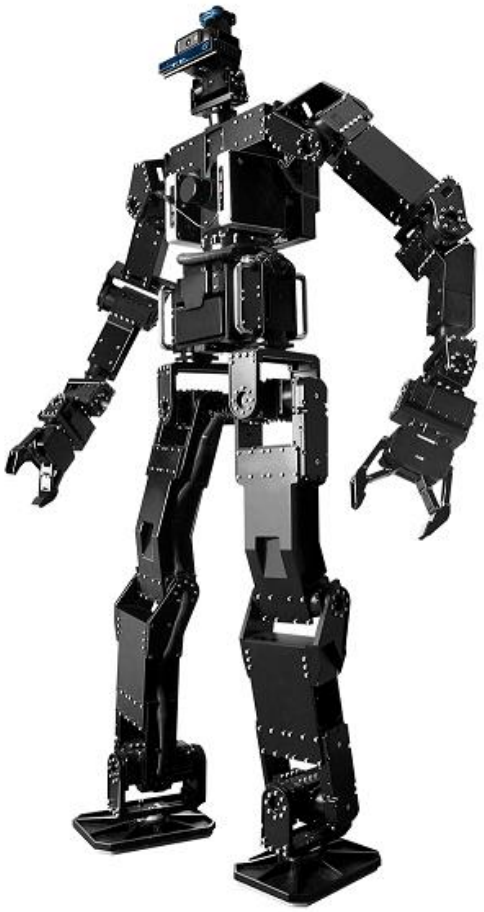


THORMANG3

THORMANG3 Tutorial

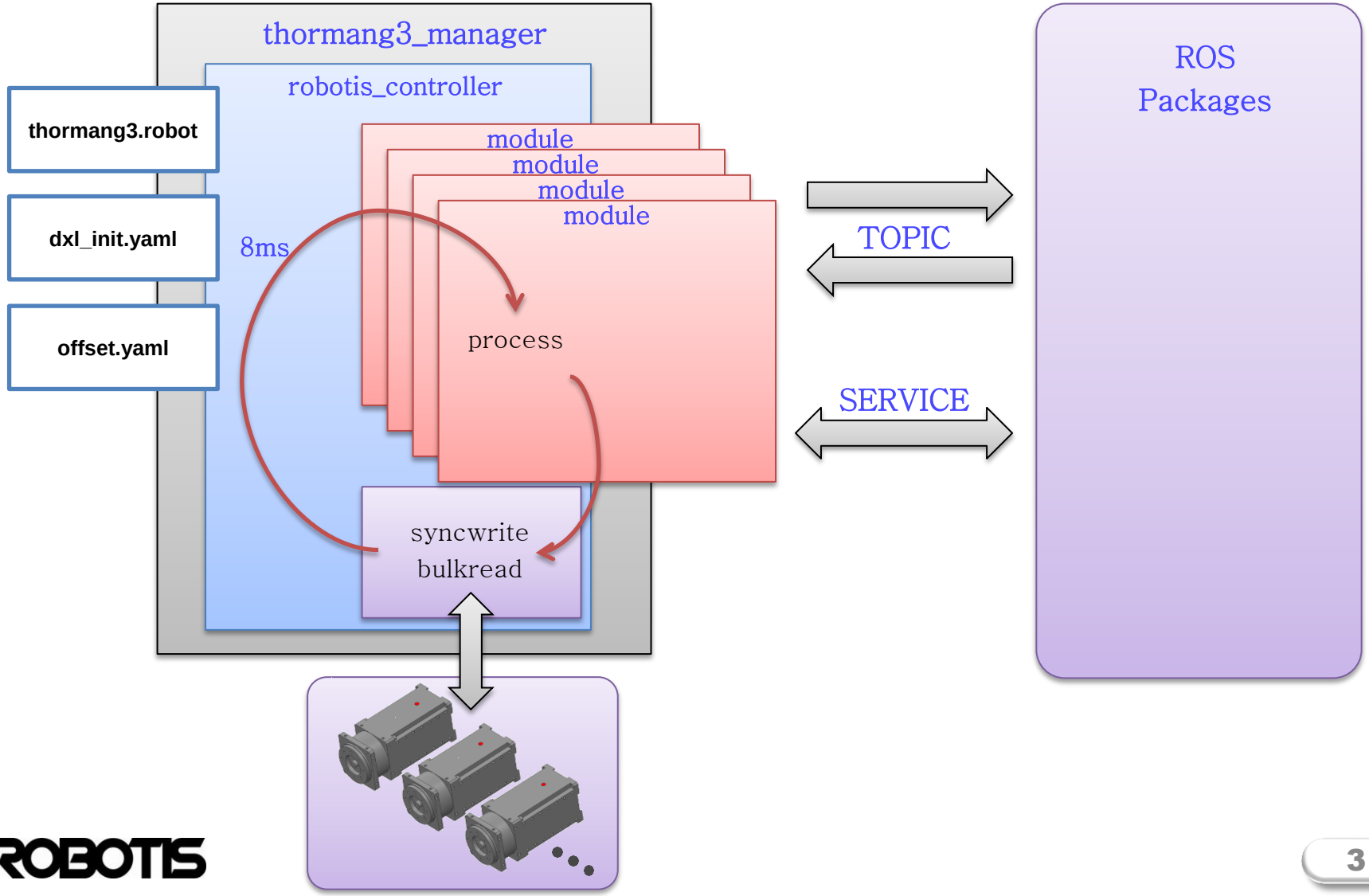
Framework

Agenda

- **What is the thormang3_manager**
  - robotis_controller
  - THORMANG3.Robot
  - dxl_init.yaml
  - offset.yaml
- **How to make a new robot manager**
- **How to make a Motion Module**
- **How to make a Sensor Module**

What is the thormang3_manager

- **System Diagram**



What is the thormang3_manager

- **robotis_controller**
  - ✓ **Initialization**
    - Load robot information file(.robot) and initialize the robot with the robotis_device package.
    - Configures initial value for each joint by loading initialization file(.yaml).
    - Gets ready to use sync write and bulk read for the joint control.
  - ✓ **Periodically call process() function by the timer (default cycle: 8 msec)**
    - The startTimer() creates a thread that calls process() function periodically.
  - ✓ **What process() does :**
    - Receives status packets with Bulk Read to get status of each sensors and joints.
    - Transfers the result value of the Motion Module with Sync Write.
    - Transfers instruction packet to each sensors and joints with Bulk Read.
    - Calls process() function of the Sensor Module in the list and saves the result value.
    - Calls process() function of the Motion Module in the list and saves the result value.
    - Publishes current value and target value in the form of ROS Topic.

What is the thormang3_manager

- **robotis_controller - Subscribed Topics**
  - ✓ **/robotis/write_control_table** (robotis_controller_msgs/WriteControlTable)
    - The message can write multiple item values to a specific joint by using Sync Write.
  - ✓ **/robotis/sync_write_item** (robotis_controller_msgs/SyncWriteItem)
    - The message can write a specific item value to multiple joints by using Sync Write.
  - ✓ **/robotis/set_joint_ctrl_modules** (robotis_controller_msgs/JointCtrlModule)
    - The message can configure motion modules that controls specific joints.
  - ✓ **/robotis/enable_ctrl_module** (std_msgs::String)
    - This message assigns a specific motion module that manages specific joints to a motion control module.
  - ✓ **/robotis/set_control_mode** (std_msgs::String)
    - The message sets the control mode of robotis_controller to ether DIRECT_CONTROL_MODE or MOTION_MODULE_MODE.
  - ✓ **/robotis/set_joint_states** (sensor_msgs::JointState)
    - This message includes status data for each joint. The data in this message is transmitted to each joint to control the joint.

What is the thormang3_manager

- **robotis_controller - Published Topics**
  - ✓ **/robotis/goal_joint_states** (sensor_msgs::JointState)
    - The message publishes goal joint status value for each joint.
  - ✓ **/robotis/present_joint_states** (sensor_msgs::JointState)
    - The message publishes current joint status value read from each joint.
  - ✓ **/robotis/present_joint_ctrl_modules** (robotis_controller_msgs/JointCtrlModule)
    - The message publishes current status of motion module that controls each joint.
- **robotis_controller - Services**
  - ✓ **/robotis/get_present_joint_ctrl_modules** (robotis_controller_msgs/GetJointModule)
    - The service to get the configuration of motion module that controls each joint.

What is the thormang3_manager

- **THORMANG3.robot**
  - Default Path : /thormang3_manager/config/THORMANG3.robot
  - Description

```
[ control info ]
control_cycle = 8   # milliseconds

[ port info ]
# PORT NAME   | BAUDRATE | DEFAULT JOINT
/dev/ttyUSB0 | 2000000  | r_arm_sh_p1
/dev/ttyUSB1 | 2000000  | l_arm_sh_p1
/dev/ttyUSB2 | 2000000  | r_leg_hip_y
/dev/ttyUSB3 | 2000000  | l_leg_hip_y

[ device info ]
# TYPE     | PORT NAME     | ID  | MODEL          | PROTOCOL | DEV NAME      | BULK READ ITEMS
dynamixel | /dev/ttyUSB0 | 1   | H54-100-S500-R | 2.0      | r_arm_sh_p1   | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 2   | H54-100-S500-R | 2.0      | l_arm_sh_p1   | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 3   | H54-100-S500-R | 2.0      | r_arm_sh_r    | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 4   | H54-100-S500-R | 2.0      | l_arm_sh_r    | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 5   | H54-100-S500-R | 2.0      | r_arm_sh_p2   | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 6   | H54-100-S500-R | 2.0      | l_arm_sh_p2   | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 7   | H54-100-S500-R | 2.0      | r_arm_el_y    | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 8   | H54-100-S500-R | 2.0      | l_arm_el_y    | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 9   | H42-20-S300-R  | 2.0      | r_arm_wr_r    | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 10  | H42-20-S300-R  | 2.0      | l_arm_wr_r    | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 11  | H42-20-S300-R  | 2.0      | r_arm_wr_y    | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 12  | H42-20-S300-R  | 2.0      | l_arm_wr_y    | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 13  | H42-20-S300-R  | 2.0      | r_arm_wr_p    | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 14  | H42-20-S300-R  | 2.0      | l_arm_wr_p    | present_position, present_voltage
dynamixel | /dev/ttyUSB2 | 15  | H54-100-S500-R | 2.0      | r_leg_hip_y   | present_position, present_voltage
dynamixel | /dev/ttyUSB3 | 16  | H54-100-S500-R | 2.0      | l_leg_hip_y   | present_position, present_voltage
dynamixel | /dev/ttyUSB2 | 17  | H54-200-S500-R | 2.0      | r_leg_hip_r   | present_position, present_voltage
dynamixel | /dev/ttyUSB3 | 18  | H54-200-S500-R | 2.0      | l_leg_hip_r   | present_position, present_voltage
dynamixel | /dev/ttyUSB2 | 19  | H54-200-B500-R | 2.0      | r_leg_hip_p   | present_position, present_voltage
dynamixel | /dev/ttyUSB3 | 20  | H54-200-B500-R | 2.0      | l_leg_hip_p   | present_position, present_voltage
dynamixel | /dev/ttyUSB2 | 21  | H54-200-S500-R | 2.0      | r_leg_kn_p    | present_position, present_voltage
dynamixel | /dev/ttyUSB3 | 22  | H54-200-S500-R | 2.0      | l_leg_kn_p    | present_position, present_voltage
dynamixel | /dev/ttyUSB2 | 23  | H54-200-B500-R | 2.0      | r_leg_an_p    | present_position, present_voltage, external_port_data_1, e
dynamixel | /dev/ttyUSB3 | 24  | H54-200-B500-R | 2.0      | l_leg_an_p    | present_position, present_voltage, external_port_data_1, e
dynamixel | /dev/ttyUSB2 | 25  | H54-200-S500-R | 2.0      | r_leg_an_r    | present_position, present_voltage, external_port_data_1, e
dynamixel | /dev/ttyUSB3 | 26  | H54-200-S500-R | 2.0      | l_leg_an_r    | present_position, present_voltage, external_port_data_1, e
dynamixel | /dev/ttyUSB0 | 27  | H54-100-S500-R | 2.0      | torso_y       | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 28  | H42-20-S300-R  | 2.0      | head_y        | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 29  | H42-20-S300-R  | 2.0      | head_p        | present_position, present_voltage
dynamixel | /dev/ttyUSB1 | 30  | GRIPPER        | 2.0      | l_arm_grip    | present_position, present_voltage
dynamixel | /dev/ttyUSB0 | 31  | GRIPPER        | 2.0      | r_arm_grip    | present_position, present_voltage
```

What is the thormang3_manager

- **THORMANG3.robot**
  - Description
    - [control info]
      - control_cycle : loop control cycle, default 8ms
    - [port info]
      - PORT NAME : The deivce name used for communication
      - BAUDRATE : The baudrate used for communication
      - DEFAULT_JOINT : one of devices connected the PORT
    - [device info]
      - TYPE : type of device (dynamixel or sensor)
      - PORT NAME : the port name which device is connected
      - ID : dynamixel or sensor id
      - MODEL : model name of device, It has to be same with the device file.
        cf) reference : ROBOTIS-Framework/robotis_deivce/devices/
      - PROTOCOL : dynamixel protocol version
      - DEV NAME : device name using in ROS(robotis_controller) ex. joint name
      - BULK READ ITEM : items to read from the device, control table item name

What is the thormang3_manager

- **H54-100-S500-R.device**
  - Description

```
[device info]
model_name  = H54-100-S500-R
device_type = dynamixel

[type info]
torque_to_current_value_ratio   = 9.66026
velocity_to_value_ratio         = 4793.01226
value_of_0_radian_position      = 0
value_of_min_radian_position    = -250950
value_of_max_radian_position    =  250950
min_radian                      = -3.14159265
max_radian                      =  3.14159265

torque_enable_item_name         = torque_enable
present_position_item_name      = present_position
present_velocity_item_name      = present_velocity
present_current_item_name       = present_current
goal_position_item_name         = goal_position
goal_velocity_item_name         = goal_velocity
goal_current_item_name          = goal_torque
position_d_gain_item_name       =
position_i_gain_item_name       =
position_p_gain_item_name       = position_p_gain
velocity_d_gain_item_name       =
velocity_i_gain_item_name       = velocity_i_gain
velocity_p_gain_item_name       = velocity_p_gain

[control table]
# addr | item name                | length | access | memory |   min value    |  max value  | signed
   0   | model_number             |        |        |        |                | 65535       | N
   6   | version_of_firmwa        |        |        |        |                | 254         | N
   7   | ID                       | 1      | RW     | EEPROM | 0              | 252         | N
   8   | baudrate                 | 1      | RW     | EEPROM | 0              | 8           | N
   9   | return_delay_time        | 1      | RW     | EEPROM | 0              | 254         | N
   11  | operating_mode           | 1      | RW     | EEPROM | 0              | 4           | N
   13  | homing_offset            | 4      | RW     | EEPROM | -2147483648    | 2147483647  | Y
   17  | moving_threshold         | 4      | RW     | EEPROM | 0              | 2147483647  | N
   21  | max_temperature_limit    | 1      | RW     | EEPROM | 0              | 100         | N
   22  | max_voltage_limit        | 2      | RW     | EEPROM | 0              | 400         | N
   24  | min_voltage_limit        | 2      | RW     | EEPROM | 0              | 400         | N
```

Item of control table

Item used in ROS(robotis_controller)

What is the thormang3_manager

- **H54-100-S500-R.device**
  - Description
    - device info
      - model_name : model name of the device
      - device_type : type of the device. Type can be either dynamixel or sensor.
    - type info
      - torque_to_current_value_ratio : the current ratio of torque to current control value.
        *current control value = torque(N) * torque_to_current_value_ratio*
      - velocity_to_value_ratio : the velocity ratio of velocity to control value.
        *velocity control value = velocity(Rad/s) * velocity_to_value_ratio*
      - min_radian : the minimum radian value of the position allowed for current dynamixel.
      - max_radian : the maximum radian value of the position allowed for current dynamixel.
      - value_of_0_radian_position : the position value at 0 radian position.
      - value_of_min_radian_position : the position value at min radian position.
      - value_of_max_radian_position : the position value at max radian position.
      - torque_enable_item_name : name of the torque enable item
      - present_position_item_name : name of the current position item
      - present_velocity_item_name : name of the current velocity item
      - present_current_item_name : name of the current current item
      - goal_position_item_name : name of the goal position item
      - goal_velocity_item_name : name of the goal velocity item
      - goal_current_item_name : name of the goal current item
      - position_d_gain_item_name : name of the D gain item of position PID control
      - position_i_gain_item_name : name of the I gain item of position PID control
      - position_p_gain_item_name : name of the P gain item of position PID control

What is the thormang3_manager

- **H54-100-S500-R.device**
  - Description
    - control table
      - address : Address of the Item. When transmitting control packets, items in the control packet are distinguished by the address.
      - item name : Name of the Item. The item name is used to distinguish items in the topic for control.
      - length : Data length of the Item.
      - access : Access permission of the Item. (R:Read, W:Write, RW:Read/Write)
      - memory : Type of memory that saves the Item.
        - » EEPROM : EEPROM keeps data without power supply. Some devices may not allow to write to EEPROM when torque is on.
        - » RAM : RAM loses its data and become initialized when the power is cut off.
      - min value : Minimum data value of the Item.
      - max value : Maximum data value of the Item.
      - signed : Y for signed Item data, N for unsigned Item data.

What is the thormang3_manager

- **dxl_init.yaml**
  - Default path : /thormang3_manager/config/dxl_init.yaml
  - Description

```
r_arm_sh_p1 :   # H54-100-S500-R
   return_delay_time      :  10    # item name : value
   operating_mode         :  3
   shutdown               :  58
   homing_offset          :  0
   torque_limit           :  310
   max_position_limit     :  250950
   min_position_limit     : -250950
   goal_torque            : 310
   goal_velocity          : 0
   goal_acceleration      : 0
   position_p_gain        : 32
   velocity_p_gain        : 180
   velocity_i_gain        : 452

l_arm_sh_p1 :   # H54-100-S500-R
   return_delay_time      :  10
   operating_mode         :  3
   shutdown               :  58
   homing_offset          :  0
   torque_limit           :  310
   max_position_limit     :  250950
   min_position_limit     : -250950
   goal_torque            : 310
   goal_velocity          : 0
   goal_acceleration      : 0
   position_p_gain        : 32
   velocity_p_gain        : 180
   velocity_i_gain        : 452

r_arm_sh_r  :   # H54-100-S500-R
   return_delay_time      :  10
   operating_mode         :  3
   shutdown               :  58
   homing_offset          :  0
```

What is the thormang3_manager

- **dxl_init.yaml**
  - Format

```
JOINT_NAME1 :     # comments
   CTRL_TABLE_ITEM_NAME1 : VALUE

JOINT_NAME2 :
   CTRL_TABLE_ITEM_NAME1 : VALUE
   CTRL_TABLE_ITEM_NAME2 : VALUE
```

    - Starting with # : comments

What is the thormang3_manager

- **offset.yaml**
  - Default path : /thormang3_manager/config/offset.yaml
  - Description

```
offset:
  head_p: 0
  head_y: 0
  l_arm_el_y: 0
  l_arm_grip: 0
  l_arm_sh_p1: 0
  l_arm_sh_p2: 0
  l_arm_sh_r: 0
  l_arm_wr_p: 0
  l_arm_wr_r: 0
  l_arm_wr_y: 0
  l_leg_an_p: 0
  l_leg_an_r: 0
  l_leg_hip_p: 0
  l_leg_hip_r: 0
  l_leg_hip_y: 0
  l_leg_kn_p: 0
  r_arm_el_y: 0
  r_arm_grip: 0
  r_arm_sh_p1: 0
  r_arm_sh_p2: 0
  r_arm_sh_r: 0
  r_arm_wr_p: 0
  r_arm_wr_r: 0
  r_arm_wr_y: 0
  r_leg_an_p: 0
  r_leg_an_r: 0
  r_leg_hip_p: 0
  r_leg_hip_r: 0
  r_leg_hip_y: 0
  r_leg_kn_p: 0
  torso_y: -0
init_pose_for_offset_tuner:
  head_p: 0
  head_y: 0
```

1. Create the Robot Manager Node

Go to the directory where Robot Manager package will be created, then create a manager package:

```
$ cd ~/catkin_ws/src/ROBOTIS-THORMANG-MPC
$ catkin_create_pkg thormang3_manager std_msgs roscpp
```

1. The Code

Create below cpp file in the `thormang3_manager` package.

```
#include "robotis_controller/robotis_controller.h"

/* Sensor Module Header */
#include "thormang3_feet_ft_module/feet_force_torque_sensor_module.h"

/* Motion Module Header */
#include "thormang3_base_module/base_module.h"
#include "thormang3_action_module/action_module.h"
#include "thormang3_head_control_module/head_control_module.h"
#include "thormang3_manipulation_module/manipulation_module.h"
#include "thormang3_walking_module/walking_module.h"

using namespace thormang3;
```

How to make a new robot manager

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, "THORMANG3_Manager");
    ros::NodeHandle nh;

    ROS_INFO("manager->init");
    robotis_framework::RobotisController *controller = robotis_framework::RobotisController::getInstance();

    /* Load ROS Parameter */
    std::string offset_file = nh.param<std::string>("offset_file_path", "");
    std::string robot_file  = nh.param<std::string>("robot_file_path", "");

    std::string init_file   = nh.param<std::string>("init_file_path", "");

    /* gazebo simulation */
    controller->gazebo_mode_ = nh.param<bool>("gazebo", false);
    if(controller->gazebo_mode_ == true)
    {
        ROS_WARN("SET TO GAZEBO MODE!");
        std::string robot_name = nh.param<std::string>("gazebo_robot_name", "");
        if(robot_name != "")
            controller->gazebo_robot_name_ = robot_name;
    }

    if(robot_file == "")
    {
        ROS_ERROR("NO robot file path in the ROS parameters.");
        return -1;
    }
```

How to make a new robot manager

```
    if(controller->initialize(robot_file, init_file) == false)
    {
        ROS_ERROR("ROBOTIS Controller Initialize Fail!");
        return -1;
    }

    if(offset_file != "")
        controller->loadOffset(offset_file);

    sleep(1);

    /* Add Sensor Module */
    controller->addSensorModule((robotis_framework::SensorModule*)FeetForceTorqueSensor::getInstance());

    /* Add Motion Module */
    controller->addMotionModule((robotis_framework::MotionModule*)BaseModule::getInstance());
    controller->addMotionModule((robotis_framework::MotionModule*)ActionModule::getInstance());
    controller->addMotionModule((robotis_framework::MotionModule*)ManipulationModule::getInstance());
    controller->addMotionModule((robotis_framework::MotionModule*)HeadControlModule::getInstance());
    controller->addMotionModule((robotis_framework::MotionModule*)WalkingMotionModule::getInstance());

    controller->startTimer();

    while(ros::ok())
    {
      usleep(1000*1000);
    }

    return 0;
}
```

2. Building the Robot Manager Node

If a package is created with the `catkin_create_pkg` command, `package.xml` and `CmakeLists.txt` files are automatically generated.

1. package.xml (modified)

```
<?xml version="1.0"?>
<package>
  <name>thormang3_manager</name>
  <version>0.1.0</version>
  <description>The thormang3_manager
package</description>

  <maintainer
email="zerom@robotis.com">ROBOTIS</maintainer>
  <license>GPLv2</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>dynamixel_sdk</build_depend>
  <build_depend>robotis_framework_common</build_depend>
  <build_depend>robotis_device</build_depend>
  <build_depend>robotis_controller</build_depend>
  <build_depend>robotis_controller_msgs</build_depend>
  <build_depend>cmake_modules</build_depend>
  <build_depend>thormang3_feet_ft_module</build_depend>
  <build_depend>thormang3_head_control_module</build_depend>
  <build_depend>thormang3_manipulation_module</build_depend>
  <build_depend>thormang3_walking_module</build_depend>
  <build_depend>thormang3_base_module</build_depend>
  <build_depend>thormang3_action_module</build_depend>

  <run_depend>roscpp</run_depend>
  <run_depend>robotis_controller</run_depend>

</package>
```

2. Building the Robot Manager Node

2. CMakeLists.txt (modified)

```
cmake_minimum_required(VERSION 2.8.3)
project(thormang3_manager)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  dynamixel_sdk
  robotis_framework_common
  robotis_device
  robotis_controller
  robotis_controller_msgs
  robotis_math

  cmake_modules
  ati_ft_sensor
  thormang3_kinematics_dynamics

  thormang3_feet_ft_module

  thormang3_head_control_module
  thormang3_manipulation_module
  thormang3_walking_module
  thormang3_base_module
)
```

```
find_package(Eigen REQUIRED)

catkin_package(
)

include_directories(
  include
  ${Eigen_INCLUDE_DIRS}
  ${catkin_INCLUDE_DIRS}
)

add_executable(thormang3_manager
src/thormang3_manager.cpp)

target_link_libraries(thormang3_manager
  ${catkin_LIBRARIES}
)
```

2. Building the Robot Manager Node

3. Build

Now, run the catkin_make within the catkin_workspace

```
$ cd ~/catkin_ws
$ catkin_make
```

How to make a new robot manager

3. Launch

Create the .launch file to pass the configuration file paths as parameters when running the Robot Manager.

1. .launch file

```
<?xml version="1.0" ?>

<launch>
    <arg name="use_imu" default="true"/>
    <arg name="use_lidar" default="true" />

    <param name="gazebo"                    value="false"    type="bool"/>
    <param name="gazebo_robot_name"         value="thormang3"/>

    <param name="offset_file_path"          value="$(find thormang3_manager)/config/offset.yaml"/>
    <param name="robot_file_path"           value="$(find thormang3_manager)/config/THORMANG3.robot"/>
    <param name="init_file_path"            value="$(find thormang3_manager)/config/dxl_init.yaml"/>

    <param name="ft_data_path"              value="$(find thormang3_manager)/config/ft_data.yaml"/>
    <param name="ft_calibration_data_path" value="$(find thormang3_manager)/config/ft_calibration_data.yaml"/>

    <!-- imu sensor package -->
    <include file="$(find imu_3dm_gx4)/launch/imu.launch" if="$(arg use_imu)"/>

    <!-- lidar -->
    <include file="$(find thormang3_description)/launch/thor_laserscan.launch" if="$(arg use_lidar)"/>

    <!-- THORMANG3 Manager -->
    <node name="thormang3_manager" pkg="thormang3_manager" type="thormang3_manager" output="screen"/>

    <!-- Robot Model & TF -->
    <include file="$(find thormang3_description)/launch/thormang3_mpc.launch"/>
</launch>
```

3. Launch

2. Run

Execute the .launch file with the roslaunch command.

```
$ roslaunch thormang3_manager thormang3_manager.launch
```

1. Create the Motion Module

Go to the directory where motion_module package will be created, then create the package:

```
$ cd ~/catkin_ws/src/ROBOTIS-THORMANG-MPC
$ catkin_create_pkg motion_module_tutorial std_msgs roscpp
```

1. The Code

Create below header and cpp files in the `motion_module_tutorial` package.

- *robotis_framework_common/include/robotis_framework_common/motion_module.h*

- *motion_module_tutorial/include/motion_module_tutorial/motion_module_tutorial.h*
- *motion_module_tutorial/src/motion_module_tutorial/motion_module_tutorial.cpp*

2. Building your package

If a package is created with the `catkin_create_pkg` command, `package.xml` and `CmakeLists.txt` files are automatically generated.

1. package.xml (modified)

- *motion_module_tutorial/package.xml*

```
<?xml version="1.0"?>
<package>
  <name>motion_module_tutorial</name>
  <version>0.1.0</version>
  <description>The motion_module_tutorial package</description>

  <maintainer email="you@yourdomain.com">Your Name</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>robotis_device</build_depend>
  <build_depend>robotis_framework_common</build_depend>

  <run_depend>roscpp</run_depend>

</package>
```

2. Building your package

2. CMakeLists.txt (modified)

- *motion_module_tutorial/CMakeLists.txt*

```
cmake_minimum_required(VERSION 2.8.3)
project(motion_module_tutorial)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  dynamixel_sdk
  robotis_device
  robotis_framework_common
)

catkin_package(
  INCLUDE_DIRS include
  LIBRARIES motion_module_tutorial
)

include_directories(
  include
  ${catkin_INCLUDE_DIRS}
)

add_library(motion_module_tutorial
  src/${PROJECT_NAME}/motion_module_tutorial.cpp
)
```

2. Building your package

3. Build

Now, run the catkin_make within the catkin_workspace

```
$ cd ~/catkin_ws
$ catkin_make
```

3. Add Created Motion Module to robotis_controller

1. Create the Sensor Module

Go to the directory where sensor_module package will be created, then create the package:

```
$ cd ~/catkin_ws/src/ROBOTIS-THORMANG-MPC
$ catkin_create_pkg sensor_module_tutorial std_msgs roscpp
```

1. The Code

Create below header and cpp files in the `sensor_module_tutorial` package.

- *robotis_framework_common/include/robotis_framework_common/sensor_module.h*

- *sensor_module_tutorial/include/sensor_module_tutorial/sensor_module_tutorial.h*
- *sensor_module_tutorial/src/sensor_module_tutorial/sensor_module_tutorial.cpp*

2. Building your package

If a package is created with the `catkin_create_pkg` command, `package.xml` and `CmakeLists.txt` files are automatically generated.

1. package.xml (modified)

- *sensor_module_tutorial/package.xml*

```
<?xml version="1.0"?>
<package>
  <name>sensor_module_tutorial</name>
  <version>0.1.0</version>
  <description>The sensor_module_tutorial package</description>

  <maintainer email="you@yourdomain.com">Your Name</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>robotis_device</build_depend>
  <build_depend>robotis_framework_common</build_depend>

  <run_depend>roscpp</run_depend>

</package>
```

2. Building your package

2. CMakeLists.txt (modified)

- *sensor_module_tutorial/CMakeLists.txt*

```
cmake_minimum_required(VERSION 2.8.3)
project(sensor_module_tutorial)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  dynamixel_sdk
  robotis_device
  robotis_framework_common
)

catkin_package(
  INCLUDE_DIRS include
  LIBRARIES sensor_module_tutorial
)

include_directories(
  include
  ${catkin_INCLUDE_DIRS}
)

add_library(sensor_module_tutorial
  src/${PROJECT_NAME}/sensor_module_tutorial.cpp
)
```

2. Building your package

3. Build

Now, run the catkin_make within the catkin_workspace

```
$ cd ~/catkin_ws
$ catkin_make
```

3. Add Created Sensor Module to robotis_controller