# ANRO

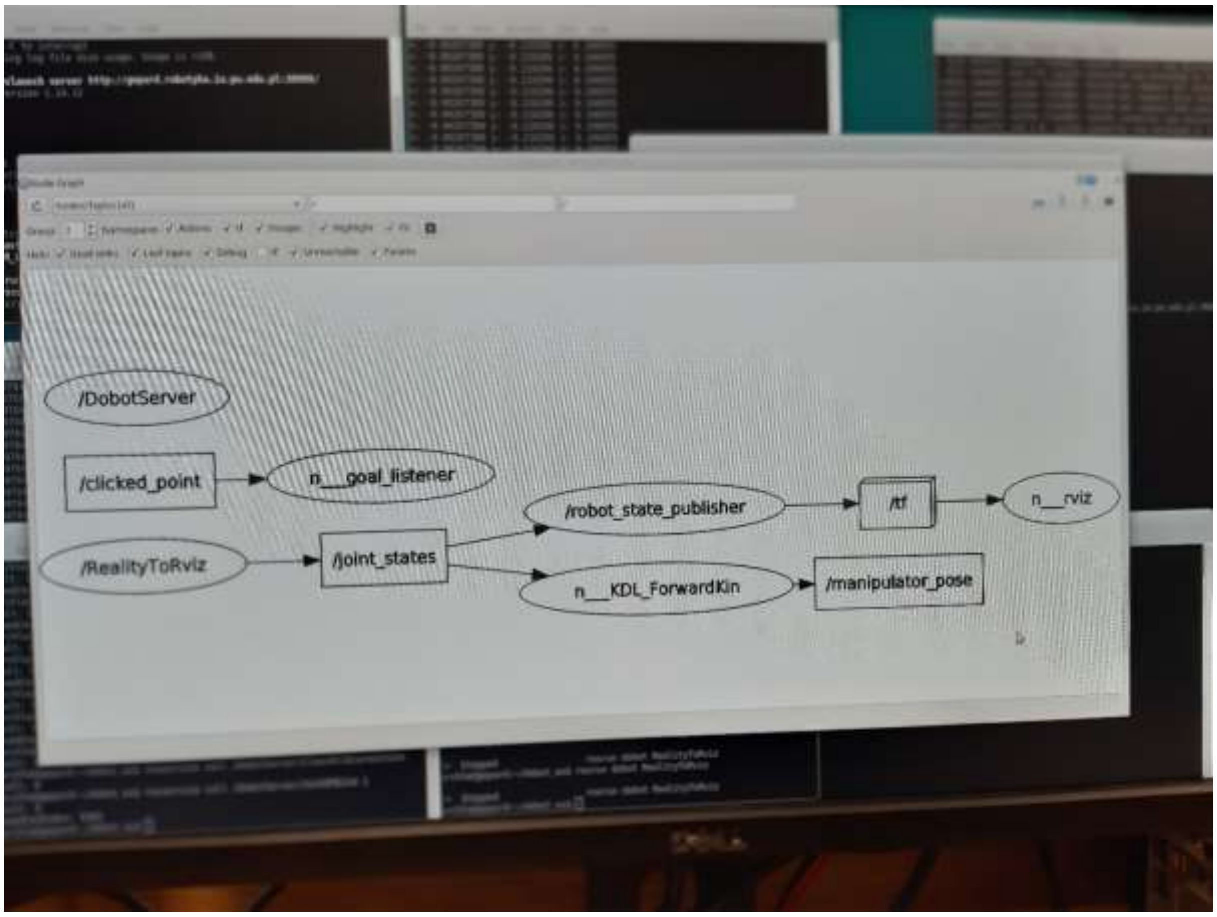
Laboratorium 4

Michał Kwarciński, Kacper Marchlewicz

## Zadawanie pozycji docelowej końcówki rzeczywistego manipulatora z poziomu RViz

Stworzyliśmy węzeł „goal_listener” który odbiera pozycję docelową z markera RViz i woła SetPTPCmd.request w celu poruszenia robota w miejsce wyznaczone przez marker. Na laboratoriach mieliśmy problem z robotem. Po około godzinie restartowania Dobota udało nam się uzyskać kontrolę. Niestety zaowocowało to rozkalibrowanym programem – nie mieliśmy wystarczająco czasu, aby odpowiednio skorygować współrzędne w RViz a w świecie rzeczywistym.

Schemat stworzonego systemu:



Link do filmu z laboratorium i zaległego z pracy domowej:

https://1drv.ms/u/s!AkMYzt8ywVnohlJRvgpKI-Ki08RR?e=lONWbx

Kod programu:

```
#include <ros/ros.h>
#include "sensor_msgs/JointState.h"
#include <stdio.h>
#include <geometry_msgs/PointStamped.h>
#include <cmath>
```

```
#include <iostream>
#include <math.h>
#include "dobot/SetJOGCmd.h"
#include "dobot/SetPTPCmd.h"
#include "std_msgs/String.h"
#include "dobot/SetCmdTimeout.h"
#include "dobot/SetQueuedCmdClear.h"
#include "dobot/SetQueuedCmdStartExec.h"
#include "dobot/SetQueuedCmdForceStopExec.h"
#include "dobot/GetDeviceVersion.h"
#include "dobot/SetPTPJointParams.h"
#include "dobot/SetPTPCoordinateParams.h"
#include "dobot/SetPTPJumpParams.h"
#include "dobot/SetPTPCommonParams.h"

using namespace std;

double coordinates [3];
void callback(const geometry_msgs::PointStamped::ConstPtr& msg) {
    double x = msg->point.x;
    double y = msg->point.y;
    double z = msg->point.z;
    coordinates[0] = x;
    coordinates[1] = y;
    coordinates[2] = z;
}
int main( int argc, char** argv )
{
    ros::init(argc, argv, "goal_listener");
    ros::NodeHandle n("~");
    ros::ServiceClient client;
    ros::Subscriber subscriber = n.subscribe("/clicked_point", 1000, callback);

 client = n.serviceClient<dobot::SetCmdTimeout>("/DobotServer/SetCmdTimeout");

    dobot::SetCmdTimeout srv1;
    srv1.request.timeout = 3000;
    if (client.call(srv1) == false) {
        ROS_ERROR("Failed to call SetCmdTimeout. Maybe DobotServer isn't started
yet!");
        return -1;

    // Clear the command queue
    client =
n.serviceClient<dobot::SetQueuedCmdClear>("/DobotServer/SetQueuedCmdClear");
    dobot::SetQueuedCmdClear srv2;
    client.call(srv2);

        // Start running the command queue
    client =
n.serviceClient<dobot::SetQueuedCmdStartExec>("/DobotServer/SetQueuedCmdStartExec")
;
    dobot::SetQueuedCmdStartExec srv3;
    client.call(srv3);
    }


    // Set PTP joint parameters
    do {
        client =
n.serviceClient<dobot::SetPTPJointParams>("/DobotServer/SetPTPJointParams");
        dobot::SetPTPJointParams srv;

        for (int i = 0; i < 4; i++) {
            srv.request.velocity.push_back(200);
        }
        for (int i = 0; i < 4; i++) {
            srv.request.acceleration.push_back(200);
        }
        client.call(srv);
    } while (0);

    // Set PTP coordinate parameters
    do {
        client =
n.serviceClient<dobot::SetPTPCoordinateParams>("/DobotServer/SetPTPCoordinateParams
");
        dobot::SetPTPCoordinateParams srv;

        srv.request.xyzVelocity = 200;
        srv.request.xyzAcceleration = 200;
        srv.request.rVelocity = 100;
        srv.request.rAcceleration = 100;
        client.call(srv);
    } while (0);

        // Set PTP jump parameters
    do {
        client =
n.serviceClient<dobot::SetPTPJumpParams>("/DobotServer/SetPTPJumpParams");
        dobot::SetPTPJumpParams srv;

        srv.request.jumpHeight = 50;
        srv.request.zLimit = 200;
        client.call(srv);
    } while (0);

        // Set PTP common parameters
    do {
        client =
n.serviceClient<dobot::SetPTPCommonParams>("/DobotServer/SetPTPCommonParams");
        dobot::SetPTPCommonParams srv;

        srv.request.velocityRatio = 75;
        srv.request.accelerationRatio = 75;
        client.call(srv);
    } while (0);
    coordinates[0] = 0.3;
    coordinates[1] = 0.0;
    coordinates[2] = -0.005;
client = n.serviceClient<dobot::SetPTPCmd>("/DobotServer/SetPTPCmd");
    dobot::SetPTPCmd srvpoints;
    while(ros::ok())
{do {
            srvpoints.request.ptpMode = 1;
            srvpoints.request.x = coordinates[0]*1000;
            srvpoints.request.y = coordinates[1]*1000;
            srvpoints.request.z = coordinates[2]*1000;
            srvpoints.request.r = 0;
            cout << coordinates[0] << coordinates[1] << coordinates[2] << endl;;
            client.call(srvpoints);
            ros::spinOnce();
            if (ros::ok() == false) {
                break;
            }
        } while (1);
    ros::spinOnce();

}
    return 0;
}
```