ОСуРВ лабораторијска вежба 8

ROS2 - Robot Operating System 2

верзија 1.0

Одсек РТ-РК

9. јануар 2026.

1 Увод

На претходној вежби смо имали пример дистрибуирани система са 2 чвора, један за контролу, други за моторе. Чворови су комуницирали путем `ZeroMQ` библиотеке. Ова библиотека захтева експлицитно адресирање чворова, провера грешака и декодовање порука.

`ROS2` окружење (енгл. Framework) аутоматизује горенаведене послове. Чворови се аутоматски проналазе и комуницирају преко теме (енгл. Topic), док је целокупно руковање порукама сакривено од корисника.

2 Радни процес

У овој вежби се чворови компајлирају и извршавају на 2 RPi 5. Као и у претходној вежби, један RPi служи као контролер који објављује поруке, док други RPi се претплаћује на поруке и на основу њих управља моторима.

Табела 1 приказује на којем RPi-ју и у којем радном простору (енгл. Workspace) које чворове треба компајлирати и покренути:

Табела 1: Чворови

hostname	`rpi-controls.local`	`rpi-motors.local`
workspace	`controls_ws`	`motors_ws`
чвор	joy	stepper
чвор	routine	wiper

Са hostname-ом смо се упознали у прошлој вежби. Сав ROS2 код се налази у `ROS2` директоријуму, а у њему су директоријуми `controls_ws` и `motors_ws` који одговарају workspace-у (отуда наставак `_ws`).

2.1 Монтирање

Да би сви студенти могли компајлирати и извршавати своје ROS2 чворове, користићемо SSH и NFS где ће `rpi-controls.local` и `rpi-motors.local` RPi-јеви бити сервери, а са Ubuntu PC-јева ће се студенти повезивати на њих. Да би преко NFS монтирали RPi-јеве:

```
mkdir rpi-controls
mkdir rpi-motors
sudo mount -t nfs rpi-controls.local:/home/pi/Public rpi-controls
sudo mount -t nfs rpi-motors.local:/home/pi/Public rpi-motors
```

Онда направити директоријум са рецимо бројем индекса, тако да сваки студент има свој директоријум у `Public` директоријуму на RPi-јевима, па након тога клонирати пројекат:

```
mkdir rpi-controls/RA_100
pushd rpi-controls/RA_100
git clone https://github.com/MilosSubotic/OSuRV_Labs
popd

mkdir rpi-motors/RA_100
pushd rpi-motors/RA_100
git clone https://github.com/MilosSubotic/OSuRV_Labs
popd
```

Због неслагања идентификатора корисника на PC-ју и RPi-јевима, да би компајлирање радило, потребно је са PC-ја дати право уписа за све кориснике са следећим командама:

```
chmod -R a+w rpi-controls/RA_100
chmod -R a+w rpi-motors/RA_100
```

2.2 Компајлирање

Отворити 2 конзоле за SSH повезивање на RPi-јеве и компајлирање.

RPi за контролу:

```
ssh pi@rpi-controls.local
cd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/ROS2/controls_ws/

source /opt/ros/jazzy/setup.bash

colcon build --symlink-install

source install/setup.sh
```

RPi за моторе:

```
ssh pi@rpi-motors.local
cd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/ROS2/motors_ws/

source /opt/ros/jazzy/setup.bash

colcon build --symlink-install

source install/setup.sh
```

`source /opt/ros/jazzy/setup.bash` ће учитати ROS2 окружење. `colcon build --symlink-install` компајлира код. `source install/setup.sh` учитава у окружење искомпајлиране чворове.

2.3 Контрола

Покренути joypad скрипту:
RPi за контролу:

```
ros2 launch joy_teleop teleop.launch.py
```

Све ROS2 команде за покретање и тестирање почињу са `ros2`. `launch` подкоманда служи да покрене скрипту са суфиксом `*.launch.py`, која покреће програме једног или више чворова са одговарајући аргументима. Овде је `joy_teleop` име пакета, док је `teleop.launch.py` име скрипте.

На другом RPi-у покренути команду за излиставање тема:
RPi за моторе:

```
ros2 topic list
```

Онда пратити како се мења `/joy` тема класе `Joy`:
RPi за моторе:

```
ros2 topic echo /joy
```

Када се притискају дугмићи, видеће се промене у вредностима `buttons`. `topic` подкоманда је везана за тестирање тема, са `list` се излиставају теме, а са `echo` се штампају поруке на теми.

Сада уместо joypad скрипте покренути `routine` чвор за рутине:
RPi за контролу:

```
ros2 run routine routine_node
```

`run` служи да се покрене програм чвора. Овде је `routine` име пакета, док је `routine_node` име чвора тј. програма чвора.

Ако се вратимо на
RPi за моторе:

```
ros2 topic echo /joy
```

видећемо како се на сваку секунду смењује порука за окретање мотора `CW` и порука за заустављање мотора `STOP`.

2.4 Мотори

Један студент или асистент нек покрене руковалац за GPIO:
RPi за моторе:

```
pushd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/SW/Driver/gpio_ctrl_v2/
make
make start
dmesg
popd
```

Покренути чвор за руковање корачним мотором.
RPi за моторе:

```
ros2 run stepper stepper_node
```

Мотор треба да се окретаће и светле LED-овке, зависности од команде чвора рутина или joypad-a.

3 Објашњење кода

3.1 Чвор издавач за рутине

На PC-ју отворити датотеку `rpi-controls/RA_100/OSuRV_Labs/L8_ROS2/ROS2/controls_ws/src/routine/src/routine_node.cpp`.

Пре свега је потребно инстанцирати чвор.

```
rclcpp::init(argc, argv);
auto node = std::make_shared<rclcpp::Node>("routine_node");
```

Путем чвора се даље прави издавач.

```
rclcpp::Publisher<sensor_msgs::msg::Joy>::SharedPtr publisher
    = node->create_publisher<sensor_msgs::msg::Joy>("joy", 10);
```

Аргументи су име теме и дубина реда чекања.

ROS2 има предефинисану поруку за joypad, па је њу потребно иницијализовати.

```
sensor_msgs::msg::Joy joy_msg;
joy_msg.header.frame_id = "joy";
joy_msg.buttons.resize(N_BUTTONS, 0);
```

Да би се могло изаћи из чвора на `Ctrl+C` потребно је следећи услов ставити у петљу:

```
while(rclcpp::ok()){
```

Следећи код објављује поруку.

```
RCLCPP_INFO_STREAM(node->get_logger(), "STOP");
joy_msg.buttons[BUTTON_STOP] = 1;
publisher->publish(joy_msg);
rclcpp::sleep_for(std::chrono::seconds(1));
joy_msg.buttons[BUTTON_STOP] = 0;
```

Линије обављају следеће.

- Испис на информационом (INFO) нивоу. Корисни су и упозорење (WARN) и грешка (ERROR).
- Поставимо одговарајуће дугме да је притиснуто.
- Објавимо поруку.
- Сачекамо секунду.
- Отпустимо дугме.

3.2 Чвор претплатник за корачни мотор

На PC-ју отворити датотеку `rpi-motors/RA_100/OSuRV_Labs/L8_ROS2/ROS2/motors_ws/src/wiper/src/wiper_node.cpp`.

Следеће променљиве служе за управљање стањем мотора.

```
int rot_dir = -1;
bool rot_en = false;
```

Када је порука примљена позива се следећи функција:

```
void joy__cb(const sensor_msgs::msg::Joy::SharedPtr joy_msg) {
    if(joy_msg->buttons[BUTTON_CW]){
        RCLCPP_INFO_STREAM(node->get_logger(), "CW");
        rot_en = true;
        rot_dir = -1;
    }
    if(joy_msg->buttons[BUTTON_CCW]){
        RCLCPP_INFO_STREAM(node->get_logger(), "CCW");
        rot_en = true;
        rot_dir = +1;
    }
    if(joy_msg->buttons[BUTTON_STOP]){
        RCLCPP_INFO_STREAM(node->get_logger(), "STOP");
        rot_en = false;
    }
}
```

У њој се проверава које је дугме притиснуто, а на основу тога врши испис у команде моторима `rot_en` и `rot_dir`.

Код за претплату се прави путем чвора:

```
rclcpp::Subscription<sensor_msgs::msg::Joy>::SharedPtr subscription
    = node->create_subscription<sensor_msgs::msg::Joy>(
    "joy",
    1,
    &joy__cb
);
```

Аргументи су име теме, дубина реда чекања и функција која ће се позвати при пријему поруке (енгл. Callback).

Остатак код је већ познато отварање руковаоца и управљање корачним мотором у бесконачној петљи на основу `rot_en` и `rot_dir`.

Потребно је с времена на време у петљи позвати:

```
rclcpp::spin_some(node);
```

да се провери да ли је нека порука стигла.

4 Задатак

Довршити TODO-ове за сложенију рутину у `routine_node.cpp`.

Реализовати `wiper_node.cpp`. Напомена да се је управљање мотором брисача једноставније од корачних, па се упис у GPIO пинове може извршити у самом `joy__cb()`.