

ОСyРВ лабораторијска вежба 8

ROS2 - Robot Operating System 2

верзија 1.0

Одсек РТ-РК

8. 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, користимо 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:

```

1 mkdir rpi-controls
2 mkdir rpi-motors
3 sudo mount -t nfs rpi-controls.local:/home/pi/Public rpi-controls
4 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:

```

1 mkdir rpi-controls/RA_100
2 pushd rpi-controls/RA_100
3 git clone https://github.com/MilosSubotic/OSuRV_Labs
4 popd
5
6 mkdir rpi-motors/RA_100
7 pushd rpi-motors/RA_100
8 git clone https://github.com/MilosSubotic/OSuRV_Labs
9 popd

```

У случају да следећи корак компајлирања не ради, то је највероватније због тога што се разликују идентификатори корисника на PC-ју и RPi-јевима, па је најлакше са PC-ја дати право уписа за све кориснике са следећим командама:

```

1 chmod -R a+w rpi-controls/RA_100
2 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:

```

1 ssh pi@rpi-controls.local
2 cd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/ROS2/controls_ws/
3
4 source /opt/ros/jazzy/setup.bash
5
6 colcon build --symlink-install
7
8 source install/setup.sh

```

RPi за моторе:

```

1 ssh pi@rpi-motors.local
2 cd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/ROS2/motors_ws/
3
4 source /opt/ros/jazzy/setup.bash
5
6 colcon build --symlink-install
7
8 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 јоурад скрипту:

RPi за контролу:

```
1 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:

```
1 ros2 topic list
```

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

RPi за моторе:

```
1 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 јоурад скрипте покренути `routine` чвор за рутине:

RPi за контролу:

```
1 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:

```
1 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:

```
1 pushd ~/Public/RA_100/OSuRV_Labs/L8_ROS2/SW/Driver/gpio_ctrl_v2/  
2 make  
3 make start  
4 dmesg  
5 popd
```

Покренути чвор за руковање корачним мотором.

RPi за моторе:

```
1 ros2 run stepper stepper_node
```

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

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.

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

Путем чвора се даље прави издавач.

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

Аргументи су име теме и дубина реда чекања.

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

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

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

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

Следећи код објављује поруку.

```
1 RCLCPP_INFO_STREAM(node->get_logger(), "STOP");
2 joy_msg.buttons[BUTTON_STOP] = 1;
3 publisher->publish(joy_msg);
4 rclcpp::sleep_for(std::chrono::seconds(1));
5 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.

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

Када је порука примљена позива се следећи функција:

```

1 void joy__cb(const sensor_msgs::msg::Joy::SharedPtr joy_msg) {
2     if(joy_msg->buttons[BUTTON_CW]){
3         RCLCPP_INFO_STREAM(node->get_logger(), "CW");
4         rot_en = true;
5         rot_dir = -1;
6     }
7     if(joy_msg->buttons[BUTTON_CCW]){
8         RCLCPP_INFO_STREAM(node->get_logger(), "CCW");
9         rot_en = true;
10        rot_dir = +1;
11    }
12    if(joy_msg->buttons[BUTTON_STOP]){
13        RCLCPP_INFO_STREAM(node->get_logger(), "STOP");
14        rot_en = false;
15    }
16 }

```

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:

```

1 rclcpp::Subscription<sensor_msgs::msg::Joy>::SharedPtr subscription
2 = node->create_subscription<sensor_msgs::msg::Joy>(
3     "joy",
4     1,
5     &joy__cb
6 );

```

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:

```

1 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()`.