RRM

Úvod do ROS a C++

Michal Dobiš
michal.dobis@stuba.sk

Marek Čornák
marek.cornak@stuba.sk

Jakub Ivan
jakub.ivan@stuba.sk

Úvod do predmetu - osnova cvičení

1. Úvod do ROS
2. ROS komunikácia 1
3. ROS komunikácia 2
4. ROS komunikácia 3
5. Priama kinematika/DH parametre
6. Modelovanie v URDF
7. Plánovanie trajektórií v kĺbovom priestore
8. Plánovanie trajektórií v kartezianskom priestore
9. Plánovanie trajektórií pokračovanie
10. MoveIt - konfigurácia
11. MoveIt - navigácia robotického ramena
12. MoveIt pokračovanie

Úvod do predmetu - bodovanie

40 bodov cvičenia (3 bloky)

1. Úvod do ROS (12 b) mini zadania na každý týždeň jedno
2. Plánovanie trajektórií (20 b) 4 týždne na vypracovanie, odovzdáva sa v 10. týždni
3. MoveIt - (8 b) 3 týždne na vypracovanie, odovzdáva sa v 12. týždni

60 bodov skúška

Obsah cv1

1. Úvod do ROS2
2. Vytvorenie workspace a jeho štruktúra
3. Vytvorenie balíka a jeho štruktúra
4. Vytvorenie vlastnej node
5. Úvod do c++ - základy OOP (trieda, objekt)
6. Samostatná úloha

Robotické manipulátory a aplikácie

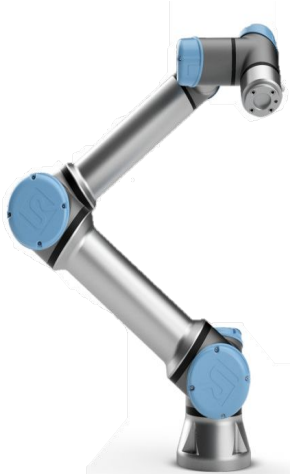
Inteligentné zváranie

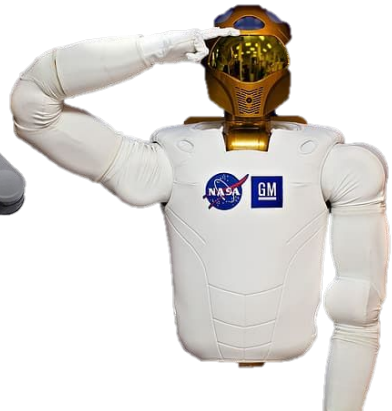
Binpicking

Čo je ROS

ROS = ROBOT OPERATING SYSTEM

- **Framework prostredie pre vývoj robotických aplikácii**
- **Funguje ako middleware na užívateľovom počítači**
- Prototyp - Stanford University
- Vyvinutý Willow Garage (2007)
- Spravovaný OSRF (Od 2013)





Stanford University

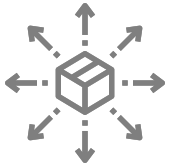
Willow Garage

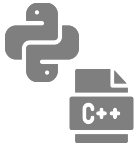
Open Source Robotics Foundation

ROS v praxi

Zdroj: https://rosindustrial.org/ric/current-members

ROS úvod

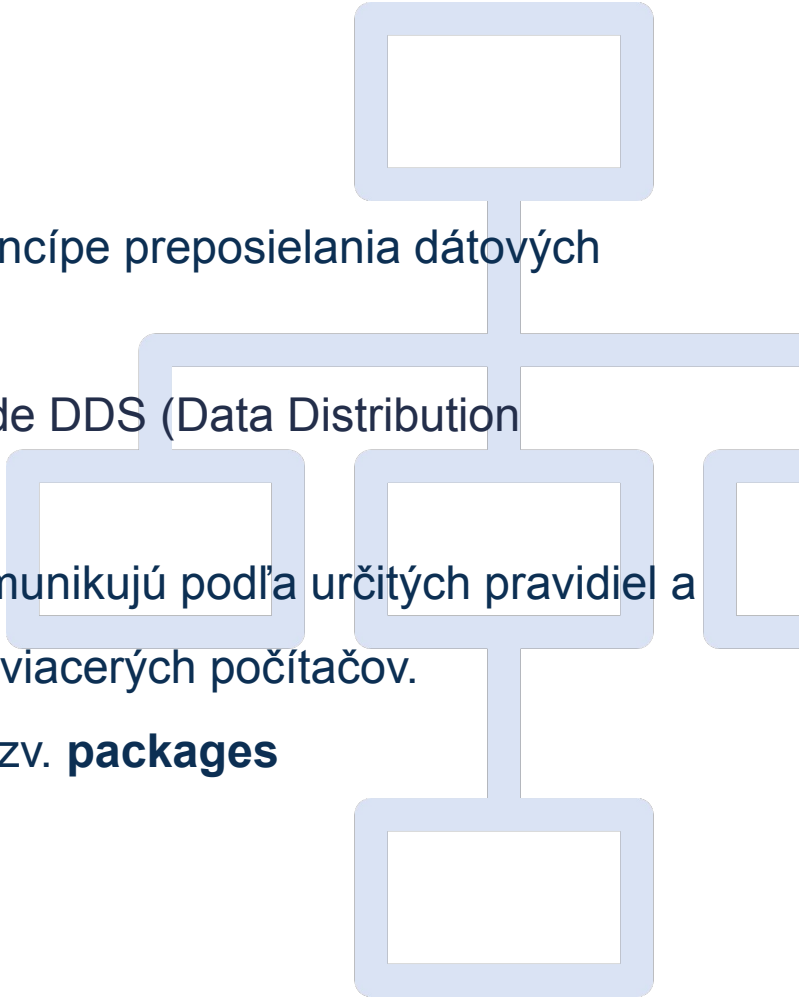




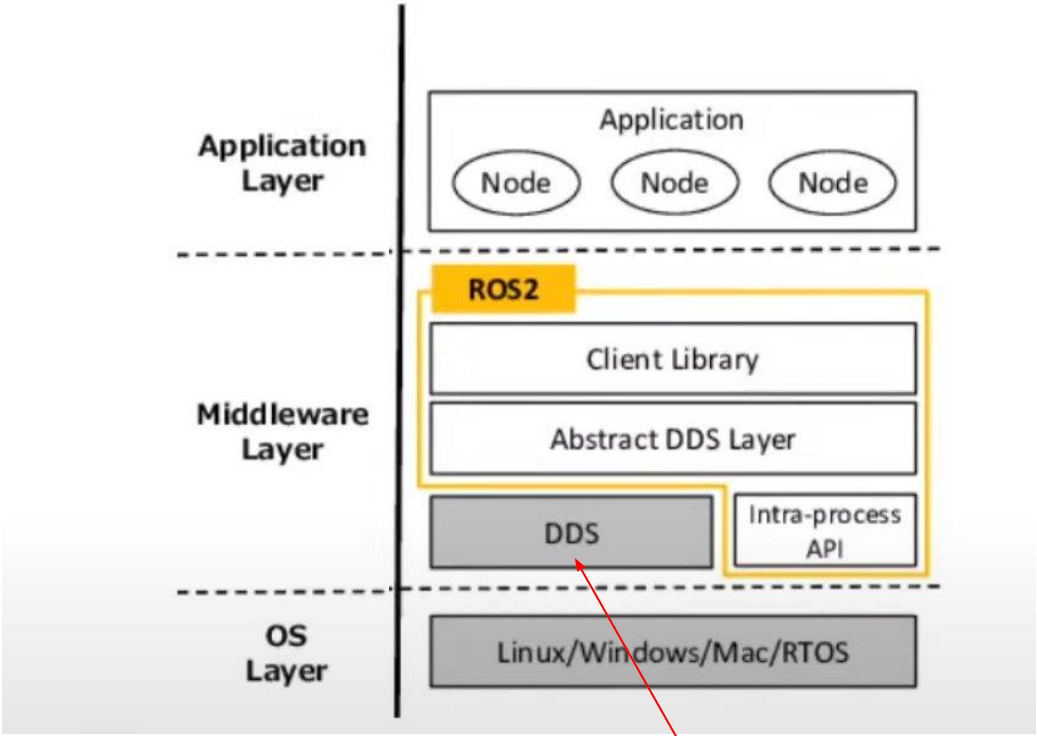
- **Peer to peer komunikácia**
  - Jednotlivé procesy spolu komunikujú cez definované API (ROS messages, services atď.)
- **Distribúcia**
  - Programy môžu bežať na viacerých PC na sieti
- **Podpora viacerých jazykov**
  - Najmä C++, Python (MATLAB, JAVA, atď.)
- **Open - Source podpora**
  - Väčšina softvéru voľne stiahnuteľná (balíčky)

Štruktúra ROS

- Základ ROSu je „middleware“ založený na princípe preposielania dátových správ medzi jednotlivými procesmi/uzlami
- Komunikácia v ROS2 je založená na štandarde DDS (Data Distribution Service)
- Sieť, v ktorej jednotlivé procesy navzájom komunikujú podľa určitých pravidiel a to synchrónne/asynchrónne, z jedného alebo viacerých počítačov.
- Softvér v ROSe je organizovaný do balíčkov tzv. **packages**



ROS1/ROS2



Industry standard
niekoľko implementácii

Prerekvizity:

- Linux (Debian based) - Ideálne Dual Boot (alebo WSL2)
  - videonávod DB: https://www.youtube.com/watch?v=qq-7X8zLP7g
  - videonávod WSL2 https://www.youtube.com/watch?v=F3n0SMAFheM&t=413s
- ROS2 inštalácia: https://docs.ros.org/en/jazzy/Installation.html
  - Ubuntu 24 - Jazzy Jalisco (najnovšia)
  - Ubuntu 22 - Iron Irwini
- Programovacie prostredie IDE
  - CLion - študentská verzia
  - VS Code - (ROS extension, dobrá podpora WSL)
  - Vi/Vim (iba pre profíkov ;D)

Vytvorenie colcon workspace:

Colcon - nástroj na kompiláciu/buildovnie (ROS2) priestorov

- generuje bin. spustiteľné súbory do ‘install’ adresára
- vytvára ‘build’ súbor pre dočasné súbory a objekty

Prerekvizita: `sudo apt install python3-colcon-common-extensions`

Návod:

1. Vytvorte si adresár: **`mkdir -p ~/<nazov_vasho_ws>/src`** (napr. `ros2_ws`)
2. **`cd ~/ros2_ws`**
3. Buildovanie ws: **`colcon build`**
4. Zdrojovanie prostredia: **`source install/setup.bash`**
5. Automatické sourcovanie pracovného priestoru pri spustení terminálu:
   a. **`nano ~/.bashrc`**
   b. Pridať do .bashrc: **`source ~/<nazov_vasho_ws>/install/setup.bash`**
   c. **`source ~/.bashrc`** alebo otvoriť a znova otvoriť terminál

návod: https://docs.ros.org/en/iron/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html#create-a-workspace

```
marek@marek-ASUS-TUF-Gaming-A15-FA507NV-FA507NV:~/xcornak_rrm$ ls
build  install  log  src
```

ROS Package

- Reprezentuje zvyčajne určitú funkcionalitu
  - driver - riadenie robota, snímače, …
  - algoritmy a aplikácie - navigácia, plánovanie trajektórií, …
- Obsahuje
  - **Zdrojový kód** – src
  - **Hlavičkové súbory** - include
  - **Súbor pravidiel pre kompiláciu**: CMakeLists.txt
  - **Manifest (metaúdaje o balíčku)** - package.xml
  - **Ďalšie** – konfigurácie a parametre, spúšťacie skripty, správy

```
my_package/
├── CMakeLists.txt
├── package.xml
├── src/
│   └── my_node.cpp (alebo iný zdrojový súbor)
├── include/
│   └── my_package/
│       └── my_header.hpp (alebo iný hlavičkový súbor)
├── launch/
│   └── my_launch_file.launch.py (alebo iný launch súbor)
├── config/
│   └── my_config.yaml (alebo iné konfiguračné súbory)
├── msg/
│   └── MyMessage.msg (definície správ)
├── srv/
│   └── MyService.srv (definície služieb)
└── test/
    └── test_my_node.cpp (alebo iné testovacie súbory)
```

ROS Package - vytvorenie

- Vytvorenie balíčka:
  Navigujte sa do /src adresara vasho ws: `cd ros2_ws/src>`

```
ros2 pkg create <package_name> --build-type ament_cmake --dependencies <dep_1> <dep_2> --node-name <node_name>
```

  (--build-type ament_python pre Python)

- **Vytvorte:** ros2 pkg create cv1 --build-type ament_cmake --dependencies rclcpp --node-name cv1_node

- Kompilácia balíčka (z úrovne workspace-ového priečinku ~/ros2_ws)

```
cd ..
colcon build
```

https://docs.ros.org/en/iron/Tutorials/Beginner-Client-Libraries/Creating-Your-First-ROS2-Package.html#

https://docs.ros.org/en/jazzy/How-To-Guides/Developing-a-ROS-2-Package.html

Štruktúra projektu po príkaze pkg

```
cv1/
├── CMakeLists.txt
├── package.xml
├── include/
├   └── cv1/
├── src/
├   └── cv1_node.cpp
```

Hello World Program!

- ros2 pkg create vytvoril balíček, v ktorom sú súbory **CMakeLists.txt**, package.xml - Package je v podstate CMake projekt (ament rozšírenie CMake pre ROS2)
- V CmakeLists.txt je konfiguračný súbor používaný nástrojom CMake na definovanie projektu a jeho build procesu
- V package.xml doplniť závislé ROS balíčky

Hello World Program!

```cpp
#include "rclcpp/rclcpp.hpp" // Zahrnutie hlavného ROS2 C++ API

int main(int argc, char * argv[])
{
   rclcpp::init(argc, argv); // Inicializácia ROS2 prostredia
   RCLCPP_INFO(rclcpp::get_logger("rclcpp"), "Hello ROS2!"); // Výpis správy

     return 0; // Ukončenie programu
}
```

Nezabudnite skompilovať: colcon build

Spustenie: ros2 run cv1 cv1_node

(nezabudnite sourcnut váš ws)

```
michaldobis@michaldobis:~/ros2_ws$ colcon build
Starting >>> cv1
Finished <<< cv1 [4.19s]

Summary: 1 package finished [4.35s]
michaldobis@michaldobis:~/ros2_ws$
```

```
michaldobis@michaldobis:~/ros2_ws$ ros2 run cv1 cv1_node
[INFO] [1726655343.036199669] [rclcpp]: Hello ROS2!
michaldobis@michaldobis:~/ros2_ws$
```

Úvod do c++

Základné pojmy

class - definícia triedy

public - verejné funkcie a premenné, prístupné zvonku

private - funkcie a premenné, ktoré sa môžu použiť iba v rámci triedy

Setter - mení stav objektu (funkcia *void move(double)*)

Getter - vracia nejakú hodnotu/stav, nemení stav vo vnútri objektu, preto je označený ako **const** (funkcia *double getCurrentPosition() const*)

Úvod do c++, štruktúra projektu

```
cv1/                          - package cv1
├── CMakeLists.txt
├── package.xml
├── include/
├   └── my_package/
├       └── robot.hpp  - definícia hlavnej triedy
├── src/
├   ├── robot.cpp      - implementácia metód triedy, konštruktor, deštruktor….
├   └── cv1_node.cpp               - funkcia main inicializácia triedy/node….
```

Úvod do c++, definovanie triedy

Vytvorenie hlavičkového súboru **robot.hpp** v **include/cv1**

```
class Robot {
public:
  Robot();
  void move(double position);
  double getCurrentPosition() const;
private:
  double position_;
};
```

vytvorenie balíčku: **`ros2 pkg create cv1 --build-type ament_cmake --dependencies rclcpp --node-name cv1_node`**

Úvod do c++, implementácia triedy

Vytvorenie zdrojového súboru **robot.cpp** do **src**

```
#include "cv1/robot.hpp"
#include "rclcpp/rclcpp.hpp"

Robot::Robot() : position_(0.0) {
RCLCPP_INFO(rclcpp::get_logger("rclcpp"),"Hello I'm robot");
}
void Robot::move(const double position){
this->position_ = position;
}
double Robot::getCurrentPosition() const {
return position_;
}
```

Scope resolution operator ::

- Prístup k premenným alebo funkciám, ktoré patria do určitého “namespace”, triedy alebo na určenie globálneho rozsahu

```
namespace MyNamespace {
    int myVar = 42;
}

int main() {
    int x = MyNamespace::myVar;
    return 0;
}
```

Odkaz na “namespace”

```
class MyClass {
public:
    void myMethod();
};

void MyClass::myMethod() {
    // Telo metódy
}
```

Definícia metódy

```
int x = 10;

int main() {
    int x = 5;
    int y = ::x;
    return 0;
}
```

Odkaz na globálnu premennú

Úvod do c++, použitie triedy a vytvorenie objektu

```
#include "cv1/robot.hpp"
#include "rclcpp/rclcpp.hpp"

int main(int argc, char **argv) {
  rclcpp::init(argc, argv);

Robot robot;
RCLCPP_INFO(rclcpp::get_logger("rclcpp"), "Position: %f", robot.getCurrentPosition());
  robot.move(1.0);
  RCLCPP_INFO(rclcpp::get_logger("rclcpp"), "Position: %f", robot.getCurrentPosition());
  return 0;
}
```

Úvod do c++ - úprava CMakeLists.txt, kompilácia, spustenie

- V CMakeLists.txt treba upraviť:

```
add_executable(cv1_node src/cv1_node.cpp src/robot.cpp)
# Vytvorí spustiteľný súbor 'cv1_node' z kódu v súboroch 'cv1_node.cpp' a 'robot.cpp'.
```

- cd ~/<vas_ws>
  colcon build
- ros2 run cv1 cv1_node