

Моделирование колесных роботов

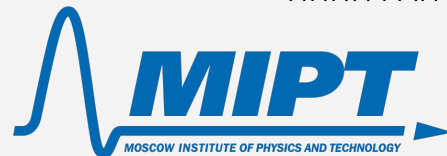
Лекция 8. Введение в ROS, базовые
концепции, первый запуск

Николай Жердев

Москва, 2023



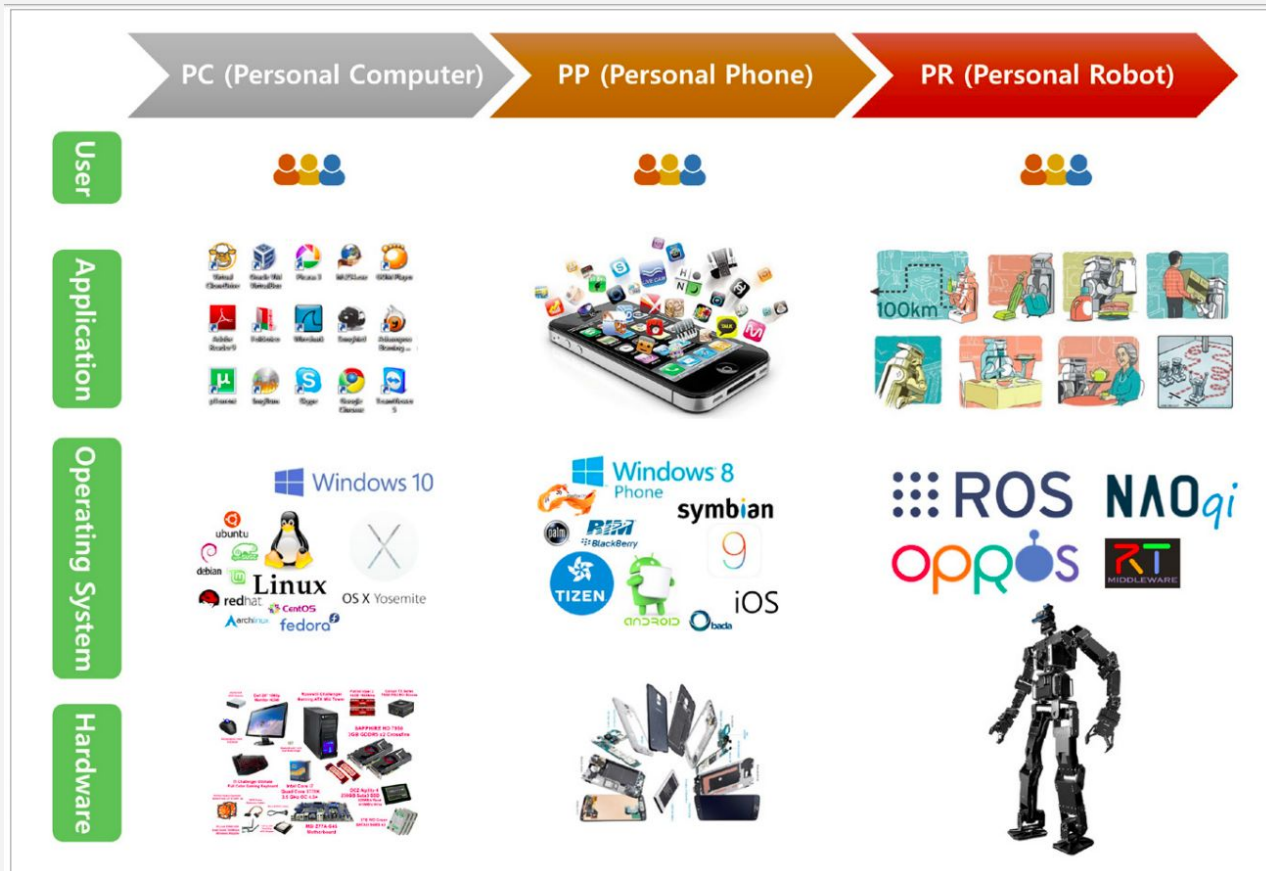
ИППИ РАН



СОДЕРЖАНИЕ ЛЕКЦИИ

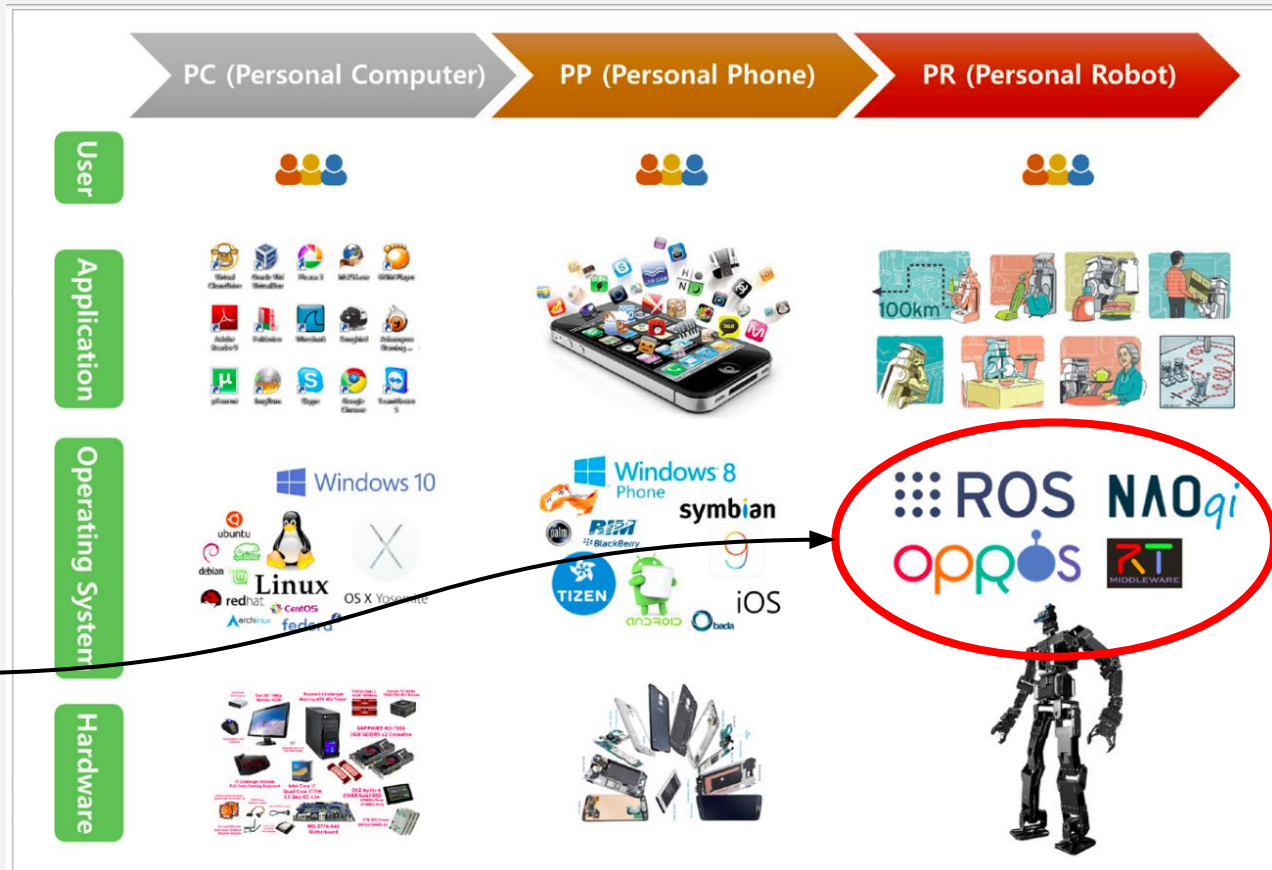
1. Зачем роботам операционная система?
2. Что такое ROS?
3. Место ROS в программировании роботов
4. История создания и существующие дистрибутивы ROS
5. Установка ROS
 - a. ROS из контейнера
6. Первый запуск ROS
 - a. source...
 - b. roscore
 - i. rosmaster
 - ii. Parameter Server
 - iii. rosout
7. Демонстрация запуска готовых модулей на примере модуля turtlesim

ЗАЧЕМ РОБОТАМ ОС?



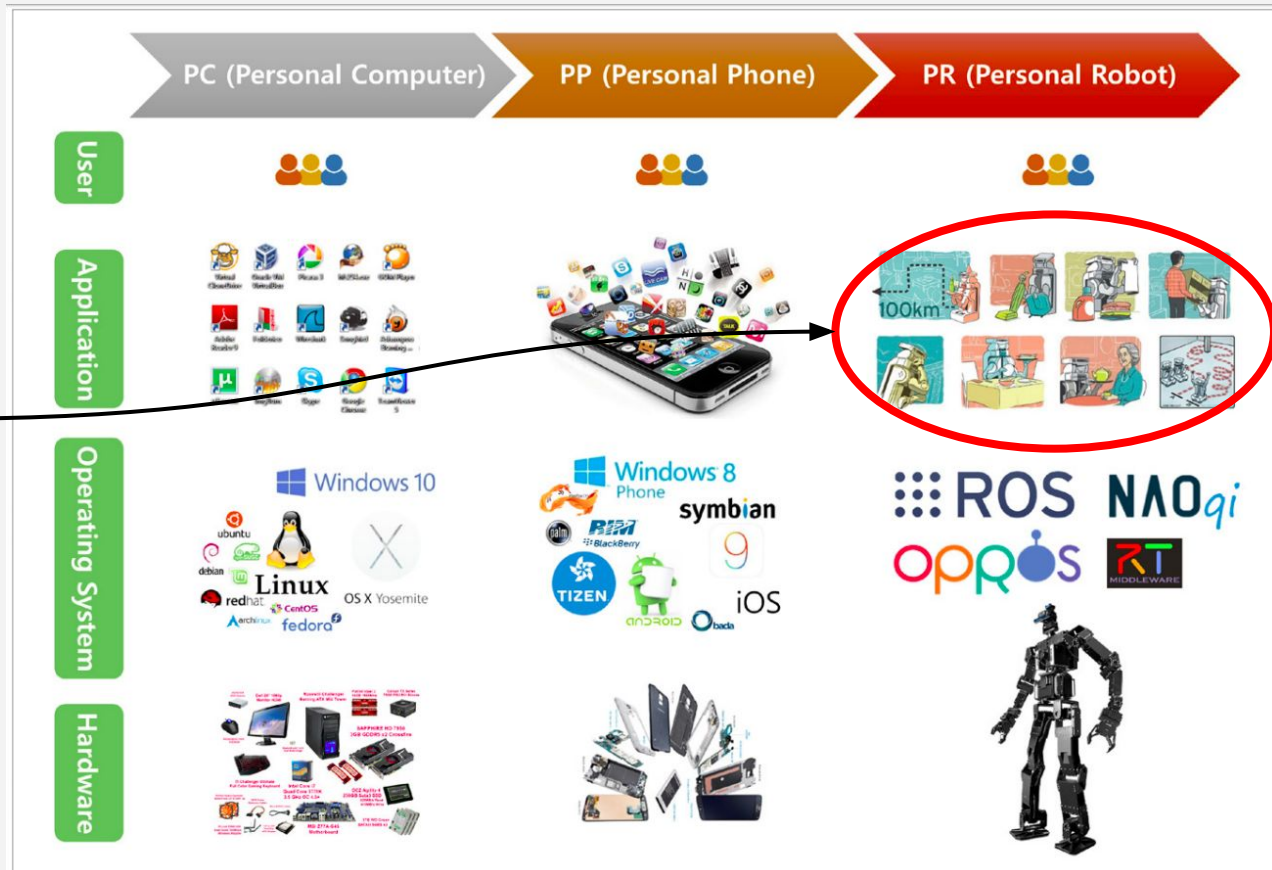
ЗАЧЕМ РОБОТАМ ОС?

ОС предоставляет абстракцию от “железа”, а также унифицирует процессы передачи данных между процессами (подсистемами)



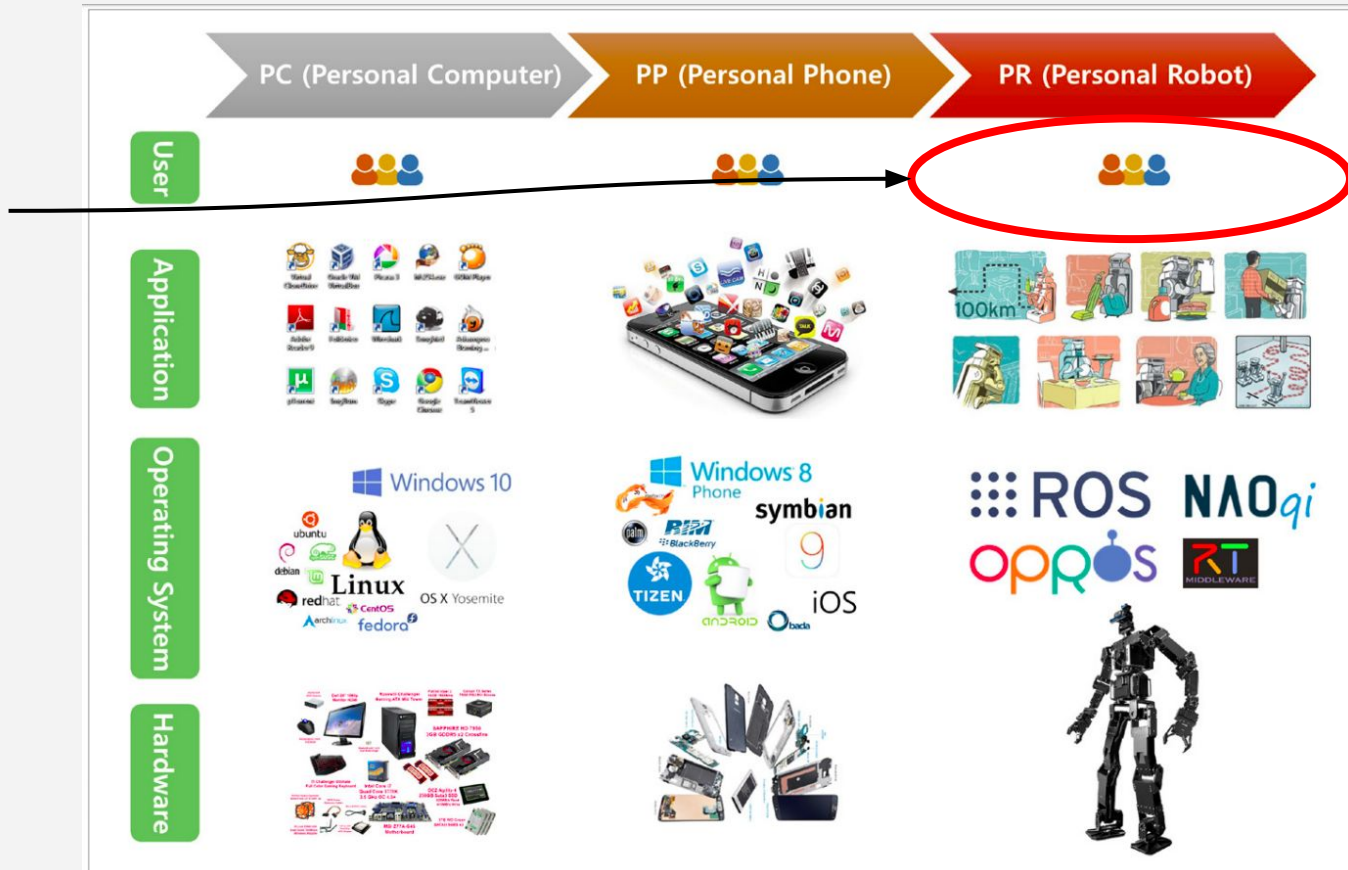
ЗАЧЕМ РОБОТАМ ОС?

Разработчики приложений
могут сосредоточиться на
написании ПО



ЗАЧЕМ РОБОТАМ ОС?

Пользователи получают большое разнообразие приложений, работающей на одном физическом устройстве



ПЛЮСЫ ОС В РОБОТОТЕХНИКЕ

- ❑ Абстракция от “железа”
- ❑ Переиспользование программ
- ❑ Модулярность за счет унифицированной парадигмы обмена данными между подпрограммами
- ❑ Инструменты разработки и отладки
- ❑ Сообщество

СУЩЕСТВУЮЩИЕ ОС/ФРЕЙМВОРКИ ДЛЯ РОБОТОВ

- ❑ **MSRDS**, Microsoft Robotics Developer Studio
- ❑ **ERSP**, Evolution Robotics Software Platform, Evolution Robotics
- ❑ **ROS**, Robot Operating System, Open Robotics
- ❑ **OpenRTM**, National Institute of Adv. Industrial Science and Technology (AIST)
- ❑ **NAOqi OS**, SoftBank and Aldebaran
- ❑ ...



ROBOT OPERATING SYSTEM

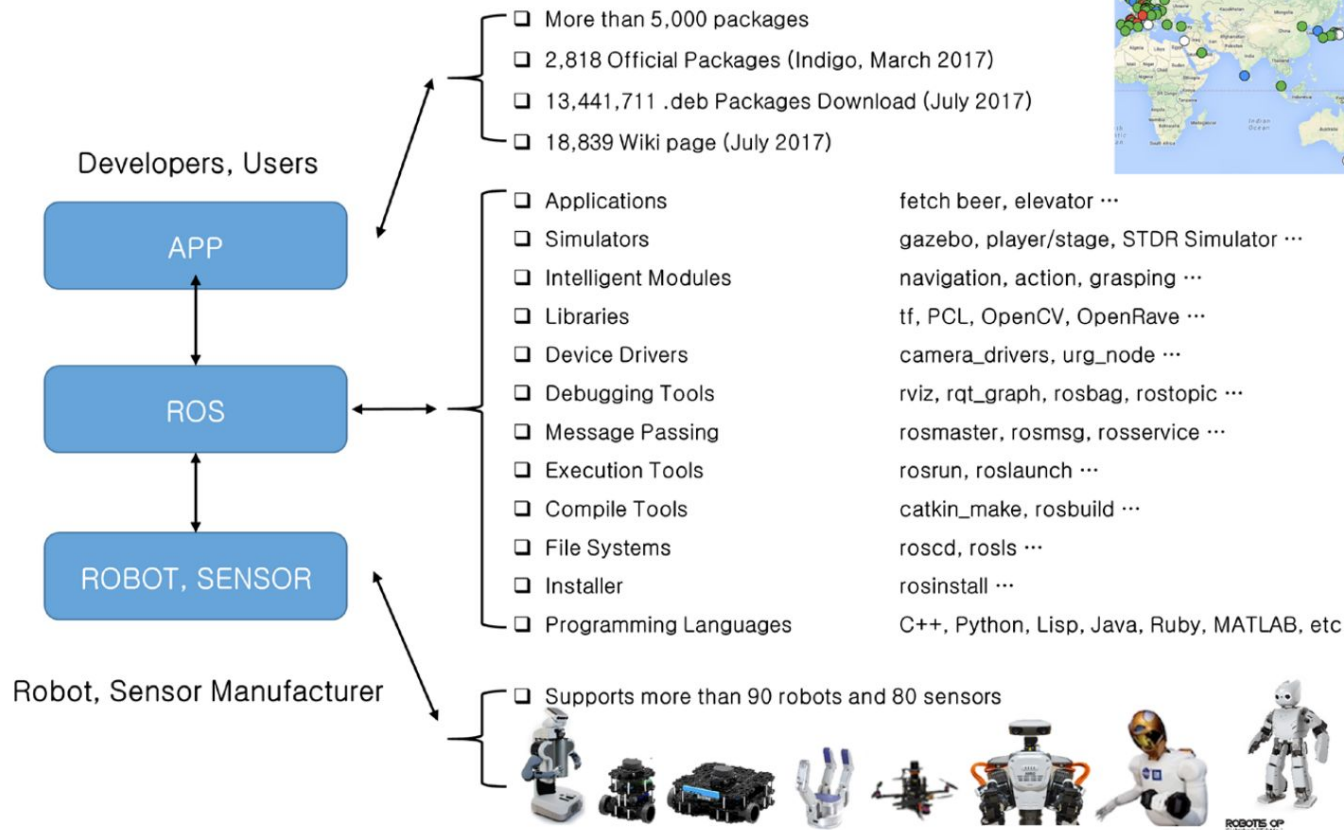
Robot Operating System, ROS — свободно распространяемая мета-операционная система для роботов. ROS обеспечивает стандартные службы операционной системы:

- ❑ аппаратную абстракцию
- ❑ низкоуровневое управление устройствами
- ❑ готовые реализации часто используемых функций
- ❑ передачу информации между процессами
- ❑ управление пакетами

ROS

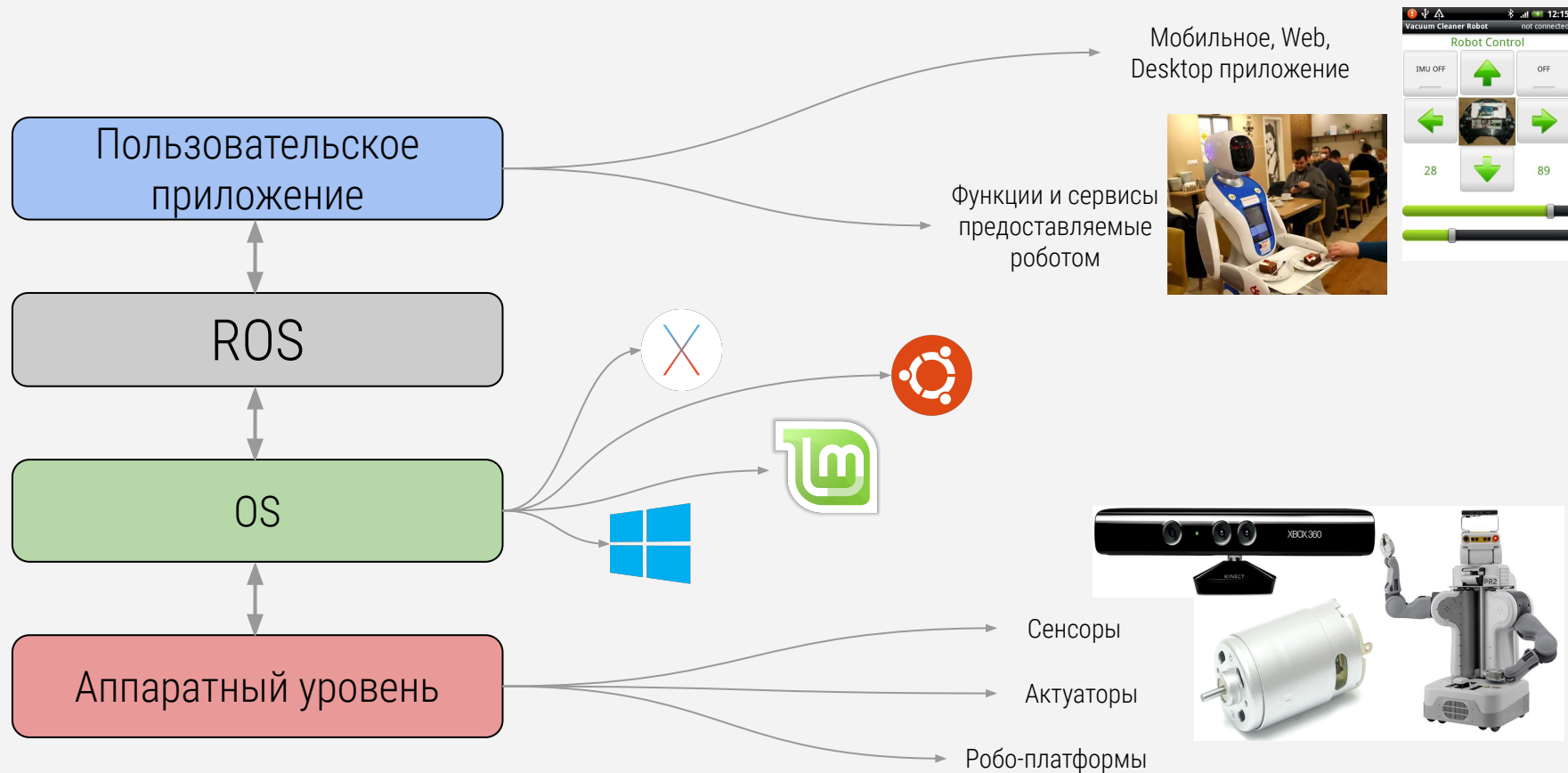


ROBOT OPERATING SYSTEM. ЭКОСИСТЕМА



<https://robots.ros.org/>

МЕТА-ОПЕРАЦИОННАЯ СИСТЕМА

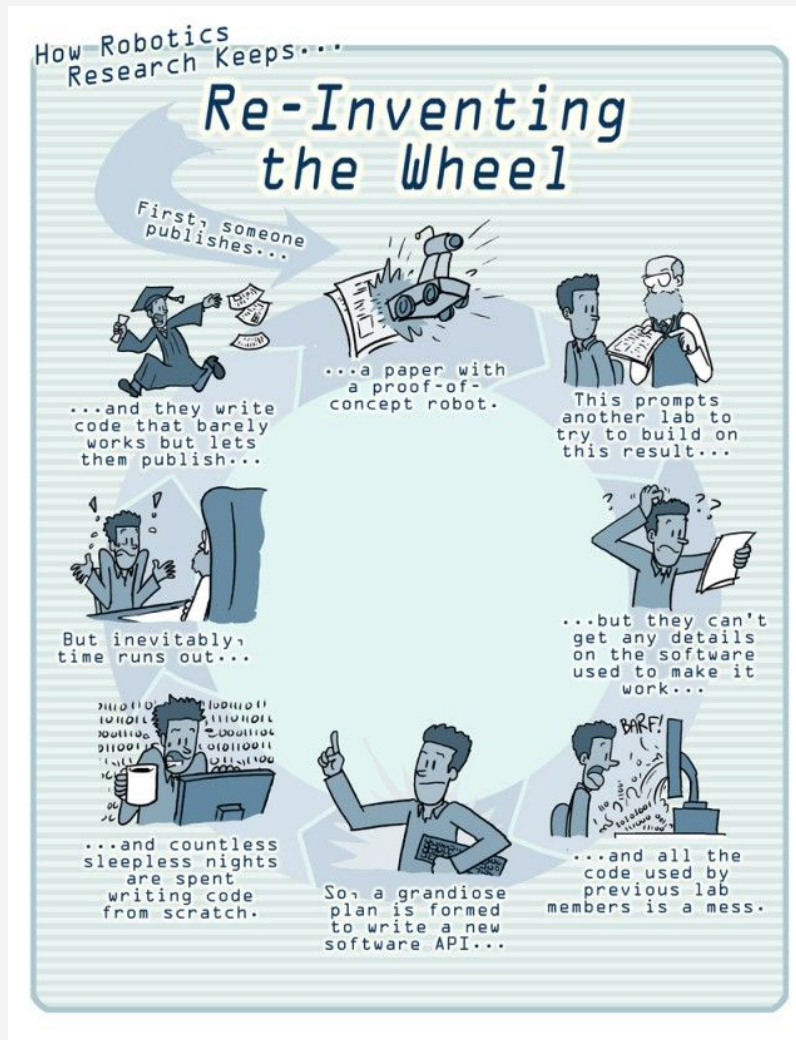


КОМПОНЕНТЫ ROS

Client Layer	roscpp	rospy	roslisp	rosjava	roslibjs	
Robotics Application	Movelt!	navigatioin	executive smach	descartes	rospex	
	teleop pkgs	rocon	mapviz	people	ar track	
Robotics Application Framework	dynamic reconfigure	robot localization	robot pose ekf	Industrial core	robot web tools	ros realtime
	tf	robot state publisher	robot model	ros control	calibration	octomap mapping
	vision opencv	image pipeline	laser pipeline	perception pcl	laser filters	ecto
Communication Layer	common msgs	rosbag	actionlib	pluginlib	rostopic	rosservice
	roswode	roslaunch	rosparm	rosmaster	rosout	ros console
Hardware Interface Layer	camera drivers	GPS/IMU drivers	joystick drivers	range finder drivers	3d sensor drivers	diagnostics
	audio common	force/torque sensor drivers	power supply drivers	rosterial	ethernet drivers	ros canopen
Software Development Tools	RViz	rqt	wstool	rospack	catkin	roscdep
Simulation	gazebo ros pkgs	stage ros				

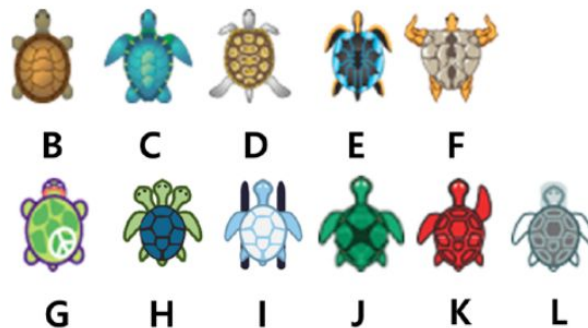
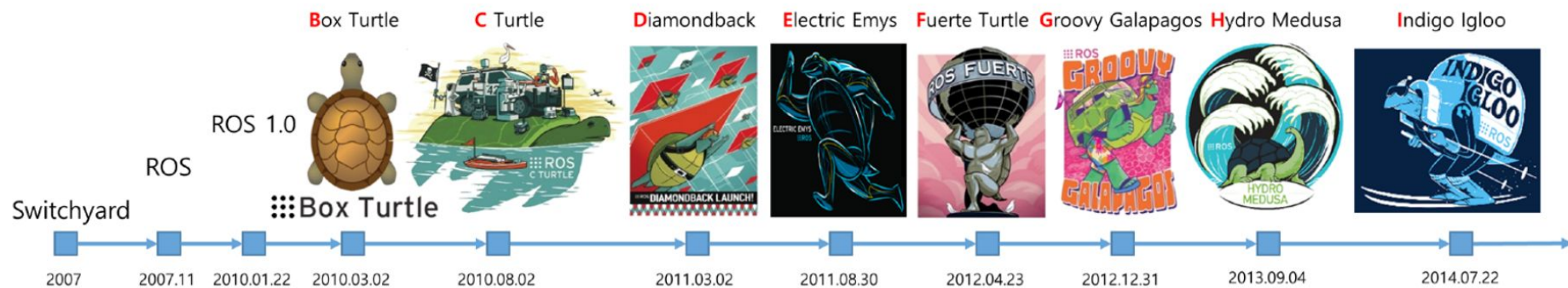
ИСТОРИЯ ROS

- ❑ 2007, Switchyard в Stanford. Еще до появления ROS в Стэнфорде было разработано несколько прототипов фреймворков для роботов. Для экспериментов использовались STanford Artificial Intelligence Robot (STAIR) и Personal Robotics (PR) program.
- ❑ 2007, Willow Garage — робототехнический инкубатор поддерживает разработку ROS.
- ❑ 2010, ROS 1.0



ДИСТРИБУТИВЫ ROS

<http://wiki.ros.org/Distributions>



ДИСТРИБУТИВЫ ROS








- ❑ Ubuntu 18.04 LTS «Bionic Beaver»
- ❑ Выпущена в мае 2018
- ❑ Поддерживается (как и Ubuntu 18.04) до мая 2023 года

ROS 2

https://design.ros2.org/articles/why_ros2.html

Почему ROS 2?

- ❑ Новые задачи, которые не стояли перед разработчиками ROS 1:
 - ❑ Группы роботов
 - ❑ Встраиваемые платформы (микроконтроллеры)
 - ❑ Системы реального времени
 - ❑ Проблемы передачи данных
 - ❑ Использование в коммерческих продуктах
 - ❑ Защита данных
 - ❑ ...
- ❑ Появление новых технологий
- ❑ Чтобы учесть прошлые ошибки
- ❑ ...

Distro	Release date	Logo	EOL date
Eloquent Elusor	Nov 22nd, 2019		Nov 2020
Dashing Diademata	May 31st, 2019		May 2021
Crystal Clemmys	December 14th, 2018		Dec 2019
Bouncy Bolson	July 2nd, 2018		Jul 2019
Ardent Apalone	December 8th, 2017		Dec 2018
beta3	September 13th, 2017		Dec 2017
beta2	July 5th, 2017		Sep 2017
beta1	December 19th, 2016		Jul 2017
alpha1 - alpha8	August 31th, 2015		Dec 2016

НАВИГАЦИЯ НА САЙТЕ ROS

The image shows a screenshot of the ROS.org website with several elements highlighted by red boxes and connected to descriptive text by red lines. The elements include the ROS.org logo, a navigation bar with links for About, Mailing Lists, and code.ros.org, a search box with Text and Titles buttons, and three main navigation buttons: Documentation, Browse Software, and News. Each button is accompanied by a text box explaining its function.

ROS.org

[About](#) | [Mailing Lists](#) | [code.ros.org](#)

Search:

Documentation

The Documentation and the ROS.org links will take you to the ROS landing page with the list of relevant links.

Browse Software

The Browse Software link will display a list of all the software packages in ROS, along with a brief description.

News

The News link will keep you informed about the latest happening with ROS.

НАВИГАЦИЯ НА САЙТЕ ROS

This is the tf package page

tf

This is a list of all packages in the geometry stack

geometry: [angles](#) / [bullet](#) / [eigen](#) / [kdl](#) / [tf](#) / [tf_conversions](#)

tf is part of the geometry stack

Package Summary

TF is a package that lets the user keep track of multiple coordinate frames over time. TF maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

Author: Tully Foote

License: BSD

This is the tf package header that is auto generated from the package manifest.xml.

Package Links:

[Code API](#)

[Msg/Srv API](#)

[Tutorials](#)

[Troubleshooting](#)

[Reviews](#) (API cleared)

[Dependency Tree](#)

The package sidebar contains links to API documentation, tutorials, troubleshooting, and reviews specific to each package.

УСТАНОВКА ROS

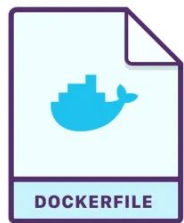
<https://www.ros.org/>

<http://wiki.ros.org/melodic/Installation/Ubuntu>

1. Добавление <http://packages.ros.org> в список “источников” из которых могут быть скачаны пакеты (в source.list)
2. Добавление ключей безопасности
3. Установка пакетов:
 - ❑ **Desktop-Full Install:** ROS, rqt, rviz, robot-generic libraries, 2D/3D симуляторы и 2D/3D пакеты обработки сенсорных данных
 - ❑ `$ sudo apt install ros-<имя дистрибутива>-desktop-full`
 - ❑ **Desktop Install:** ROS, rqt, rviz, и robot-generic libraries
 - ❑ `$ sudo apt install ros-<имя дистрибутива>-desktop`
 - ❑ **ROS-Base:** ROS package, build, и communication libraries. Без графических инструментов
 - ❑ `$ sudo apt install ros-<имя дистрибутива>-base`
 - ❑ **Отдельный пакет**
 - ❑ `$ sudo apt install ros-<имя дистрибутива>-<имя пакета>`

Жизненный цикл контейнера

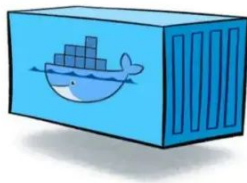
Докерфайл - набор инструкций для создания докер образа.



Docker file



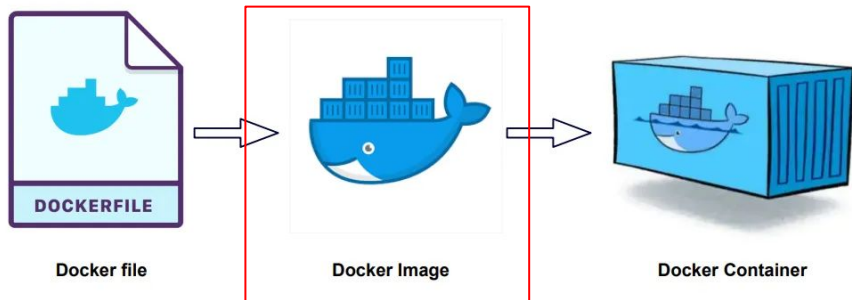
Docker Image



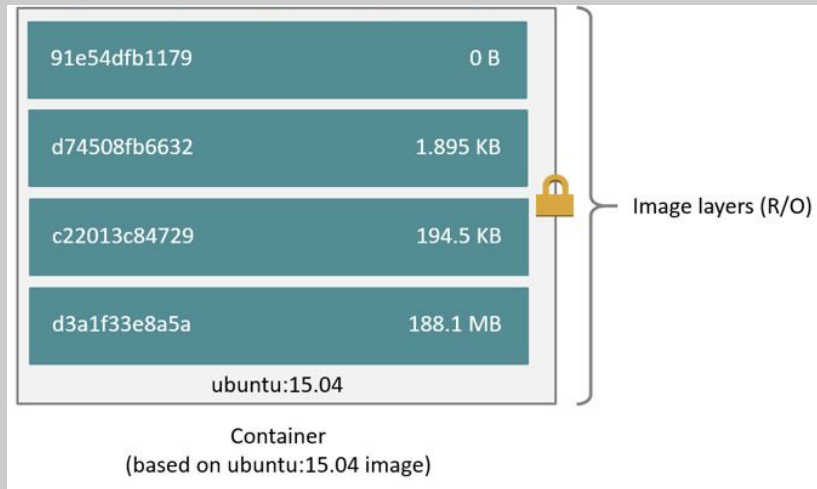
Docker Container

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
LABEL org.opencontainers.image.authors="org@example.com"
COPY . /app
RUN make /app
RUN rm -r $HOME/.cache
CMD python /app/app.py
```

Жизненный цикл контейнера

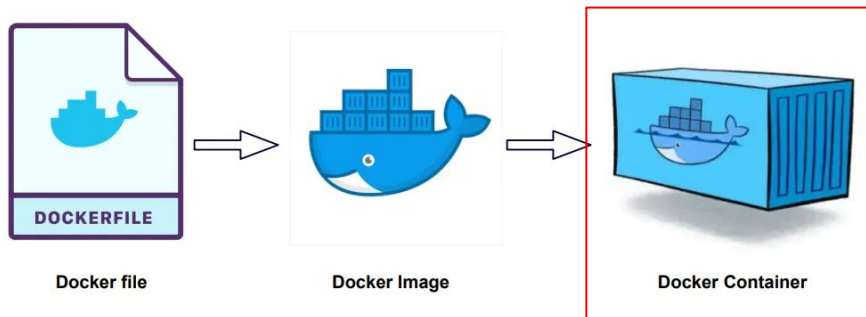


Докер образ - это файл-шаблон с инструкциями для создания контейнеров.

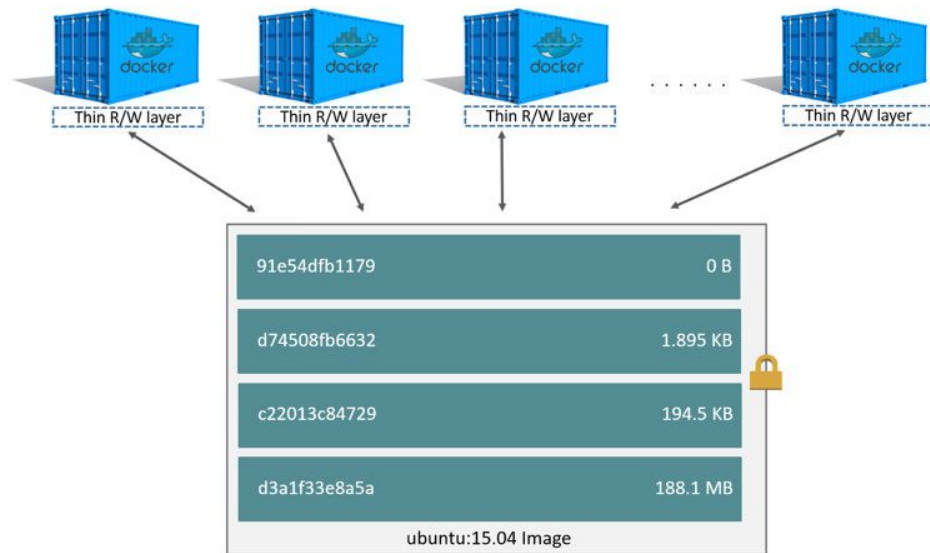


Состоит из слоёв. Одна директива RUN образует один слой.

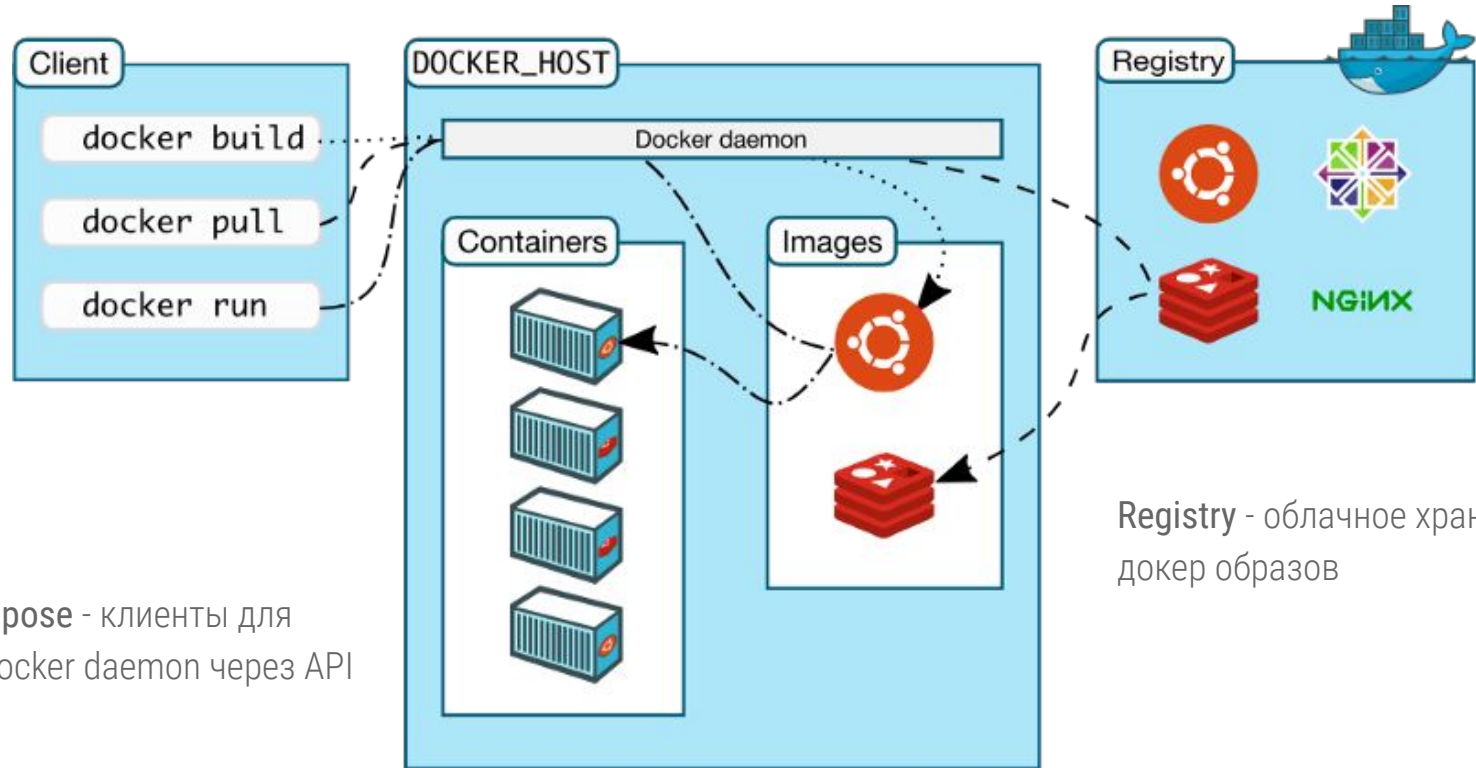
Жизненный цикл контейнера



Докер контейнер - экземпляр докер образа.
Имеет дополнительный слой для записи.



Жизненный цикл контейнера



docker,
docker compose - клиенты для
работы с Docker daemon через API

Registry - облачное хранилище для
докер образов

ЗАПУСК DOCKER КОНТЕЙНЕРА С ROS

<http://wiki.ros.org/docker/Tutorials/Docker>

https://hub.docker.com/_/ros

1. **Создаем Dockerfile** и описываем в нем установку и настройку необходимых пакетов
2. **Создаем** из Dockerfile **образ** (Image):
 - ❑ `$ docker build --tag <имя образа> .` ← в конце точка! Это значит собирать Докерфайл в текущей директории
1. **Запускаем контейнер** из созданного образа:
 - ❑ `$ docker run -it <имя образа>`

Чтобы получить доступ к экрану из контейнера (необходимо для запуска графических приложений из контейнера), можно запустить следующим образом:

```
$ docker run -e DISPLAY=unix$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -it <имя образа>
```

- ❑ Если при запуске возникает ошибка вида:
`No protocol specified`
`Error: cannot open display: unix:0.0`
- ❑ Необходимо разрешить локальные сетевые подключения на компьютере-хосте:
`$ xhost +local:`

ОСНОВНЫЕ ТЕРМИНЫ

<http://wiki.ros.org/ROS/Concepts>

Нод — единица программного кода, исполняемая в отдельном потоке и выполняющий определенную вычислительную функцию. Ноды коммуницируют друг с другом через **топики**. Совокупность всех запущенных нодов образует **ROS граф**.

Топик — система передачи данных с механизмом публикации / подписки. Нод отправляет данные, публикуя их в топик, и читает, подписываясь на топик. Множество нодов может публиковать данные в один топик и читать данные из одного топика. Один нод может как подписываться, так и публиковать множество топиков.

Тип сообщения — описание формата сообщения, публикующегося в топике. Каждый топик публикует сообщения определенного типа.

ROS граф — совокупность всех нодов и топиков, публикующихся в системе.

Пакет — единица организации ПО в ROS. Пакет может содержать программный код нодов, описание сообщений топиков, конфигурационные и другие файлы, связанные общим смыслом.

SOURCE А.К.А СТРАШНЫЙ СОН НОВИЧКА

- ❑ Необходимый шаг перед **каждым** сеансом работы с ROS (выполняется в каждом вновь открытом терминале):

`$ source /opt/ros/<имя дистрибутива>/setup.bash`

В нашем случае <имя дистрибутива> = melodic

- ❑ Что делает `$ source ...` ?
 - ❑ Устанавливает переменные окружения, необходимые для работы ROS
- ❑ Чтобы каждый раз не выполнять команду выше, ее часто добавляют в файл `~/.bashrc`, что приводит к ее автоматическому запуску для каждого нового терминала

`$ echo "source /opt/ros/<имя дистрибутива>/setup.bash" >> ~/.bashrc`

SOURCE A.K.A СТРАШНЫЙ СОН НОВИЧКА

```
→ ~ env | grep ROS
→ ~ source /opt/ros/kinetic/setup.zsh
→ ~ env | grep ROS
ROS_DISTRO=kinetic
ROS_ETC_DIR=/opt/ros/kinetic/etc/ros
ROS_PACKAGE_PATH=/opt/ros/kinetic/share
ROS_VERSION=1
ROS_ROOT=/opt/ros/kinetic/share/ros
ROS_MASTER_URI=http://localhost:11311
ROSLISP_PACKAGE_DIRECTORIES=
→ ~ █
```

- ❑ ROS дистрибутив
- ❑ Путь к установленным ROS-пакетам
- ❑ Версия ROS
- ❑ URI мастера
- ❑ ...

ЗАПУСК ROSCORE

<http://wiki.ros.org/roscore>

roscore — это набор нодов и программ, необходимых для запуска любого ROS-приложения. **RoScore** должен быть запущен для того, чтобы ноды могли коммуницировать. **RoScore** запускается командой:

- ❑ `$ roscore`

Также можно указать порт, на котором будет запущен **master**:

- ❑ `$ roscore -p 1234`

roscore запускает:

- ❑ [rosmaster](#)

- ❑ [Parameter Server](#)

- ❑ [rosout](#) — нод логирования

roscore всегда запускается автоматически, если запуск производится через **roslaunch**, и **roscore** не был запущен до этого.

Ноды, которые запускаются при запуске **roscore** определены в **roslaunch/roscore.xml** (**внимание** не рекомендуется менять roscore.xml т.к. это повлияет на все последующие запуски ROS).

ЗАПУСК ROSCORE

- ❑ Куда осуществляется логирование текущей сессии ROS

- ❑ Где запущен roslaunch server

- ❑ Какие параметры доступны на **ROS Parameter Server**

- ❑ Где запущен **rosmaster**

- ❑ Запуск нода **rosout**

```
roscore http://devel-Latitude-5491:11311/
roscore http://devel-Latitude-5491:11311/ 71x40
~ roscore
... logging to /home/shipitko/.ros/log/97e47afc-7cda-11ea-9a3e-185680878a02/roslaunch-devel-Latitude-5491-28870.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://devel-Latitude-5491:38865/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [28880]
ROS_MASTER_URI=http://devel-Latitude-5491:11311/

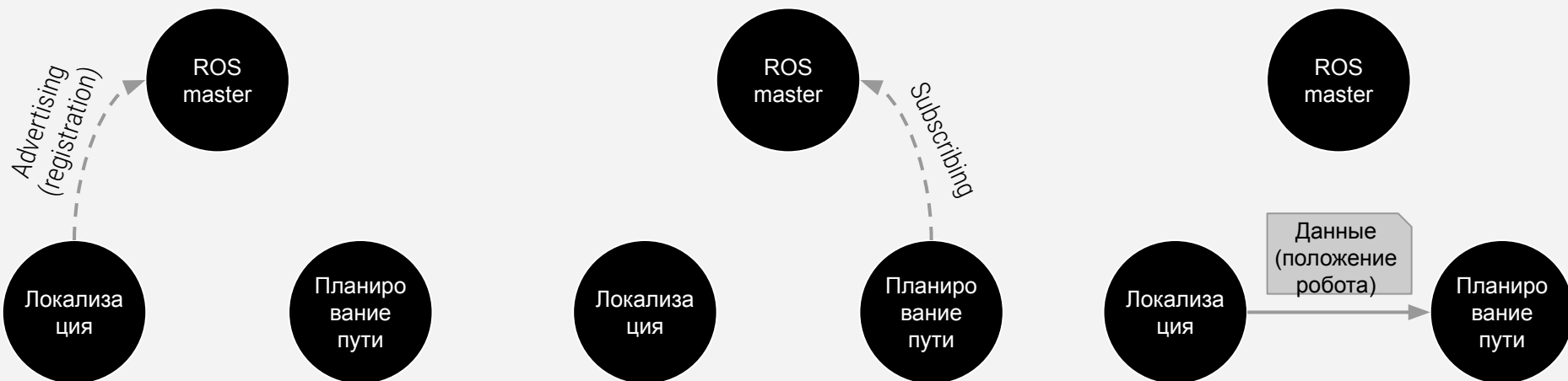
setting /run_id to 97e47afc-7cda-11ea-9a3e-185680878a02
process[rosout-1]: started with pid [28893]
started core service [/rosout]
```

```
$ ps axjfw
```

ROS MASTER

<http://wiki.ros.org/Master>

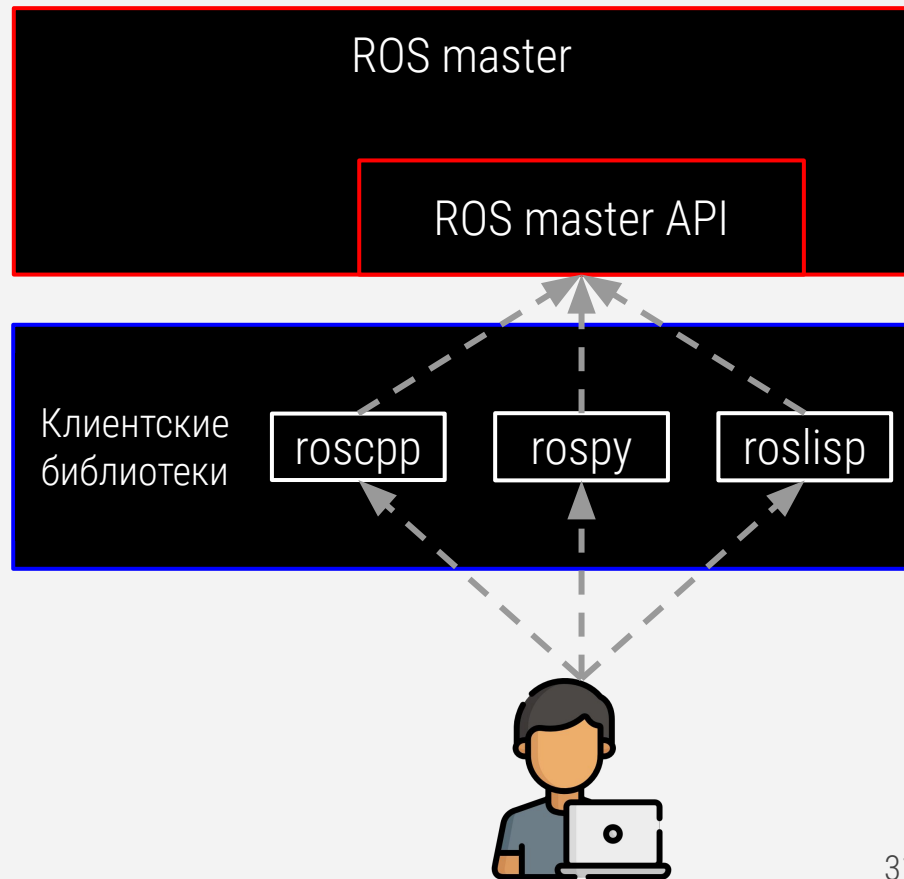
- ❑ Обеспечивает регистрацию топиков и сервисов. Помогает нодам находить друг друга. Можно сравнить с DNS-сервером – по имени топика / сервиса, предоставляет его URI.
- ❑ Предоставляет сервер параметров (**Parameter Server**)



ROS MASTER

http://wiki.ros.org/ROS/Master_API

- ❏ **rosmaster** предоставляет API для регистрации/отмены регистрации топиков и сервисов, а также для получения списка запущенных нодов, зарегистрированных топиков и т.д.
- ❏ **Клиентские библиотеки** используют **rosmaster API** и предоставляют разработчику простой механизм создания нодов, публикации топиков и т.д.
- ❏ **XML-RPC** - remote procedure call (RPC) протокол использующий формат XML для создания запросов и механизм HTTP для их передачи.



PARAMETER SERVER

<http://wiki.ros.org/Parameter%20Server>

Parameter Server — словарь параметров, доступный всем нодам в системе. Используется для хранения различных параметров и доступа к ним в режиме реального времени. Запускается внутри **rosmaster**.

Параметры на сервере:

/camera/left/name: leftcamera
/camera/left/exposure: 1
/camera/right/name: rightcamera
/camera/right/exposure: 1.1

Запрос: /camera/left/name

Ответ:
leftcamera

```
~$ rosparam get /camera/left
```

Запрос: /camera/left

Ответ:
name: leftcamera
exposure: 1

```
name = rospy.get_param("/camera/left")
```

Запрос: /camera

Ответ:
left: { name: leftcamera, exposure: 1 }
right: { name: rightcamera, exposure: 1.1 }

PARAMETER SERVER

<http://wiki.ros.org/Parameter%20Server>

Типы данных, поддерживаемые ***Parameter Server*** :

- ❑ 32-bit integers
- ❑ booleans
- ❑ strings
- ❑ doubles
- ❑ iso8601 dates
- ❑ lists
- ❑ base64-encoded binary data

Доступ к параметрам осуществляется через клиентские библиотеки (***roscpp***, ***rospy***, ...), а также инструмент командной строки ***rosparam***. Оба способа будут рассмотрены в курсе позднее.

ROSOUT

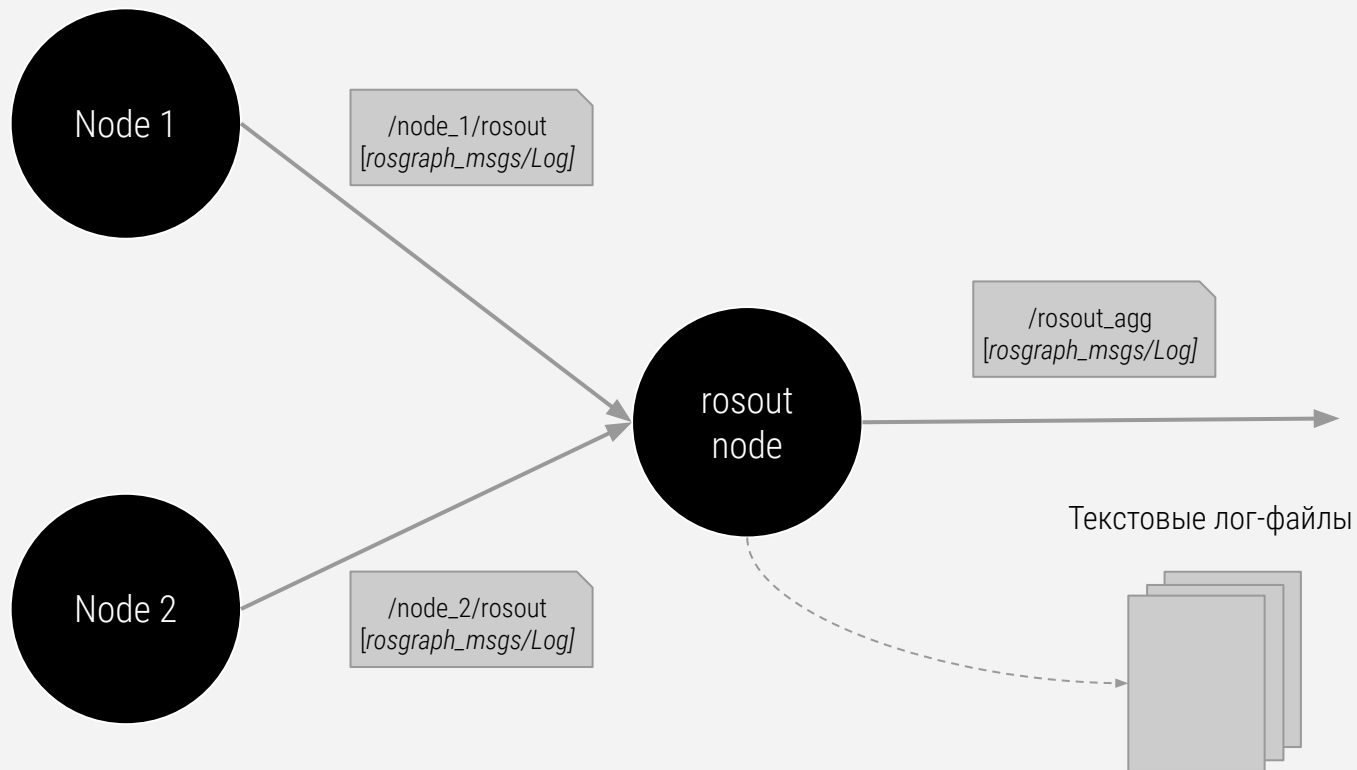
<http://wiki.ros.org/rosout>

Под термином **rosout** понимают несколько сущностей:

1. **Нод rosout**. Подписывается на лог-сообщения каждого нода (<node namespace>/rosout), сохраняет их в текстовый лог-файл, а также дублирует в топик /rosout_agg.
2. **Топик /rosout**. Стандартный топик для публикации лог-сообщений.
3. **Топик /rosout_agg**. Содержит агрегированные лог-сообщения от всех нодов.
4. **Тип сообщения rosgraph_msgs/Log**. Используется топиками /rosout и /rosout_agg.
5. **API клиентских библиотек** для работы с rosout логами в разных языках программирования.

ROSOUT

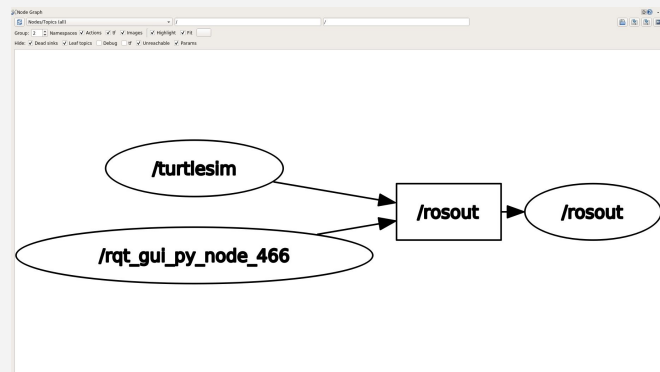
<http://wiki.ros.org/rosout>



ROS “HELLO (TURTLE) WORLD!”

<http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>

- ❑ Устанавливаем переменные окружения
`$ source /opt/ros/<дистрибутив>/setup.bash`
- ❑ Запускаем **roscore** в фоновом режиме
`$ roscore &`
- ❑ Запускаем **turtlesim_node** из пакета **turtlesim**
`$ rosrn turtlesim turtlesim_node`
- ❑ Смотрим получившийся **ROS граф**
`$ rqt_graph`
- ❑ Запускаем **turtle_teleop_key** нод
`$ rosrn turtlesim turtle_teleop_key`
- ❑ Управляем симуляцией нажатием “стрелок” и снова смотрим получившийся **ROS граф**



ROS “HELLO (TURTLE) WORLD!”

<http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>

- ❑ Проверяем запущенные ноды и существующие топики

`$ rostopic list`

`$ rosnodetree`

- ❑ Проверяем тип сообщения, публикуемого в топик

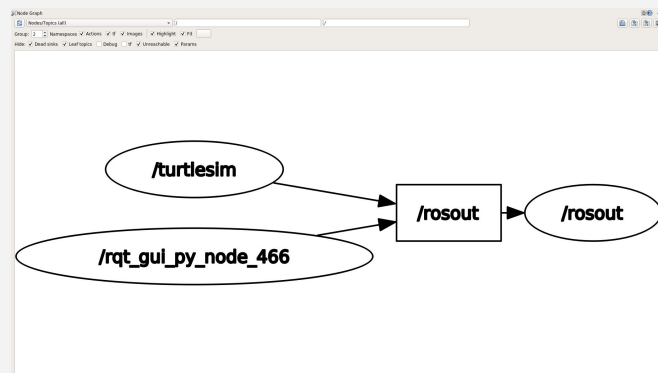
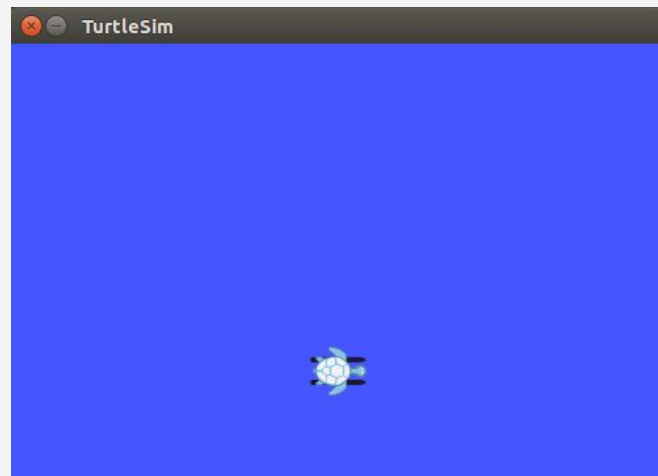
`/turtle1/pose`

`$ rostopic info /turtle1/pose`

`$ rostopic type /turtle1/pose`

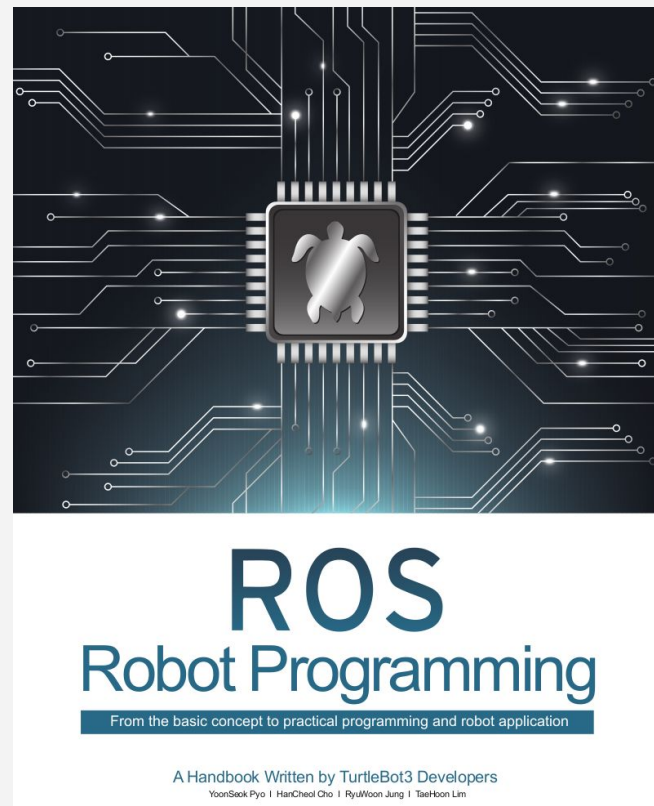
- ❑ Смотрим на данные, публикуемые в топик **`/turtle1/pose`**

`$ rostopic echo /turtle1/pose`



ДОПОЛНИТЕЛЬНЫЕ ИСТОЧНИКИ

1. Книга: ROS Robot Programming.
YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim (Eng) — основа этой лекции
2. Обучающие инструкции ROS (Eng)
3. Введение в ROS от Voltbro (Rus)
4. Clearpath Robotics ROS Tutorial (Eng)
5. История создания ROS (Eng)



ИНФОРМАЦИЯ О ПРЕЗЕНТАЦИИ

Эта презентация была подготовлена Олегом Шипитько в рамках курса “Моделирование колесных роботов” кафедры когнитивных технологий Московского физико-технического института (МФТИ). Автор выражает благодарность, авторам, чьи материалы были использованы в презентации. В случае, если вы обнаружили в презентации свои материалы, свяжитесь со мной, для включения в список авторов заимствованных материалов.

This presentation was prepared by Oleg Shipitko as part of the “Mobile Robotics” course at the Department of Cognitive Technologies, Moscow Institute of Physics and Technology. The author is grateful to the authors whose materials were used in the presentation. If you find your materials in a presentation, contact me to be included in the list of contributing authors.