Курс повышения квалификации

Введение в современную робототехнику на основе ROS

Орг вопросы

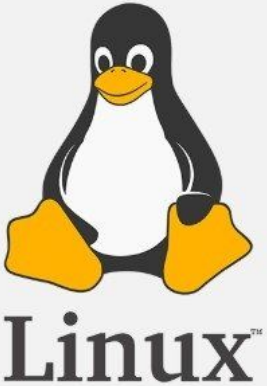
Wi-Fi

TurtleBro или TurtleBro_5G
turtlew001

Структура очной части курса

Работа с реальными роботами
Программирование робота
Работа с периферией
Телеуправление
Автономная навигация
Работа с камерой
Практикум: патрулирование
Практикум: Работа с удаленным роботом
Итоговая работа
Обсуждение работы, аттестация

Linux для роботов



Linux. Основы командной строки.

Для программистов, разработчиков робототехники, операторов роботов и инженеров Terminal это один из основных инструментов работы.

Терминал вызывается нажатием клавиш **Ctrl + Alt + T**

- **Tab** - автодополнение, наберите начало команды или пути и нажмите Tab, если такая команда только одна оболочка ее дополнит. Если доступно несколько вариантов, нажмите два раза Tab чтобы их увидеть
- **Стрелка вверх** - предыдущая команда в истории
- **Стрелка вниз** - следующая команда в истории
- **Ctrl + C** - прервать выполнение программы
- **Ctrl + Z** - приостановить выполнение программы
- **Ctrl + D** - завершить текущий сеанс связи
- **Ctrl + Shift + C** Скопировать
- **Ctrl + Shift + V** Вставить

Linux. Основы командной строки. Основные команды Linux.

ls (List - список) вывод списка файлов в текущей директории.

pwd (print working directory) выводит текущую директорию

cd (Change Directory) - сменить директорию

mkdir (Make Directory) - создать директорию

touch (Коснуться) можно создать пустой файл

mv (Move) - переместить файлы или директории

cp (Copy) скопировать файлы или директории

rm (Remove) - удалить файлы или директории

sudo (Substitute User & DO) - подменить пользователя
и выполнить

find (Найти) предназначена для поиска файлов

cat (Catenate - связывать) вывод содержимого файла на экран

nano - консольный текстовый редактор

| - конвейер перенаправления вывода

scp копирование файлов по сети

grep (global regular expression print) применение регулярных выражений

apt — менеджер пакетов

SSH— доступ к удаленному терминалу


$eat
$eat
$sudo
eat
Чилик

Python для роботов

Python для роботов

Основы и синтаксис

```
>>> print("Hello, robot")
Hello robot
>>>
```

Отступы - основа разметки

```
if (a > b):
    print("a > b")
else:
    print("b > a")
```

```
a = 2
b = 3
c = a + b
print(c)
```

Python для роботов

Ход выполнения программы

```
a = 2
b = 3
c = a + b
print(c)
```

Python для роботов

Числа и операции над ними

Целые числа - int

Действительные числа - float

Операции над числами:

+ сложение
- вычитание
* умножение
/ деление
** возведение в степень
// целочисленное деление с отбрасыванием остатка
% получение остатка от деления

Python для роботов

Операторы сравнения чисел

Истина - True Ложь - False

< “Меньше” — условие верно, если первый операнд меньше второго

> “Больше” — условие верно, если первый операнд больше второго

<= “Меньше или равно”

>= “Больше или равно”

== “Равенство” Условие верно, если два операнда равны

!= “Неравенство” Условие верно, если два операнда неравны

Python для роботов

Переменные и оператор присваивания

Оператор присваивания =

```
>>> moon_to_earth = 384400
>>> moon_to_earth = moon_to_earth + 1
```

Python для роботов

Ввод и вывод данных

Вывод print()

```
print(3 + 5)
print(2 * 9, (6 - 9) * 4)
print(3 ** 4)  # две звёздочки (**) - оператор возведения в степень
print(69 / 4)  # косая черта (/) - оператор деления
print(69 // 4)  # две косые черты (//) -
оператор возвращающий частное от деления нацело
print(57 % 7)  # процент (%) - оператор остатка от деления нацело
```

Ввод input()

```
a = input("Enter a")
b = input("Enter b")
s = a + b
print(s)
```

Python для роботов

Преобразование типов

Функция type() - возвращает тип аргумента

Для явного преобразования типов переменной используются функции с названиями типов

int() - целое число

float() - действительное число

str() - строка

и др.

Python для роботов

Списки и строки

Список

```
Primes = [2, 3, 5, 7, 11, 13]
```

```
Rainbow = ['Red', 'Orange', 'Yellow', 'Green']
```

Строка

```
s = "Hello robot"
```

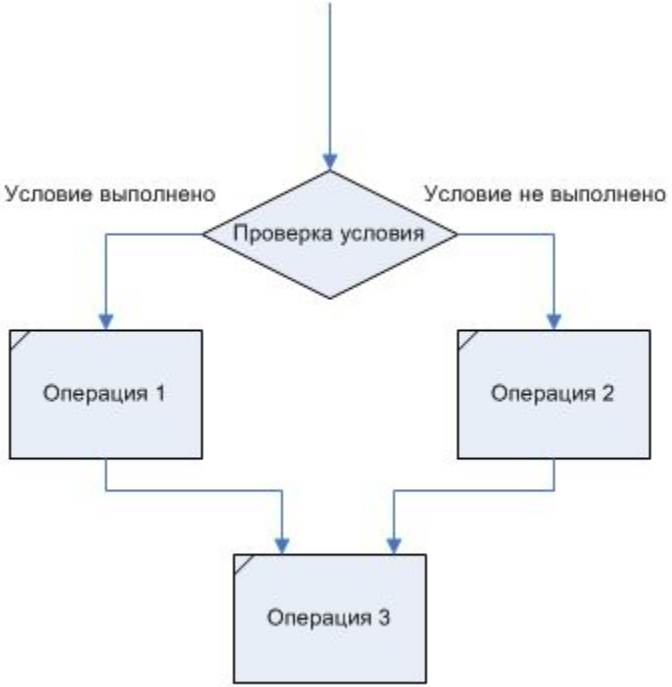
Доступ к элементу списка или строки

```
a = Primes[2]
```

Python для роботов

Ветвления хода программы

```
if (5 < 3):
    print("Условие == true")
    print("Основной путь ветвления")
else:
    print("Условие == false")
    print("Альтернативный путь ветвления")
```



Python для роботов

Логические операторы

Стандартные логические операторы: логическое И, логическое ИЛИ, логическое НЕ (отрицание)

логическое И - and	(a and b) == True, a == True and b == True
логическое ИЛИ - or	(a or b) == True, a == True or b == True
логическое НЕ - not	(not a) == True, a == False

Python для роботов

Циклы

```
while (условие == истина):
        код

for переменная in итерируемый объект:
      код с участием переменной

s = “Hello robot”
for i in s:
      print(i)
```

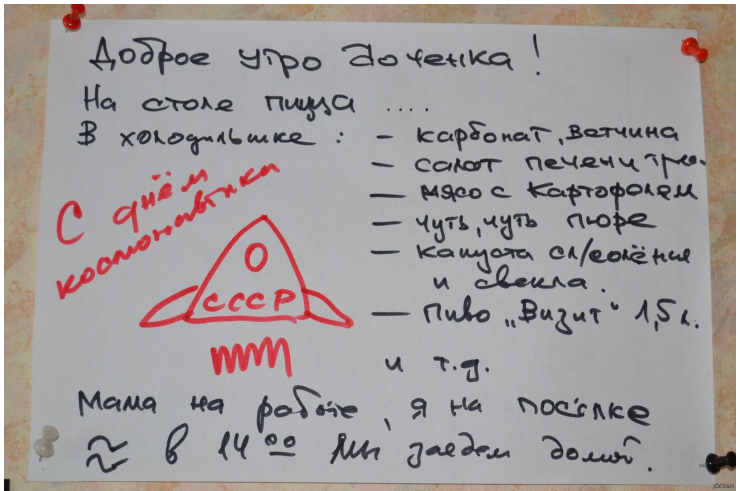
Python для роботов

Функции

Определение - Описание функции. Алгоритм работы.
Вызов - Выполнение функции. Непосредственно работа.
Возврат - Значение возвращаемое в точку вызова функции
Передача параметра в функцию - arg

```
def Foo(arg):
    return 1
```

```
a = Foo()
```



Python для роботов

Область видимости

Разделение доступа к переменным

```
def function1():
    a = 1
    print(a)

def function2():
    print(a)

function1()
function2()
```

NameError: name 'a' is not defined

Python для роботов

Глобальные переменные

```
def function1():
    global a
    a = 5
    print(a)

def function2():
    print(a)

function1()
function2()
```

```
>>> 5
>>> 5
```

Python для роботов

Стандартные функции

print(), input(), len(), type(), str(), int(), float(), list(), dict(), tuple(), range(), sum(), min(), max() ….

Python с библиотеками

Python без них

Python для роботов

Библиотека math

```
import math
```

`round(x)` Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
`abs(x)` Модуль (абсолютная величина). Это — стандартная функция.
`sqrt(x)` Квадратный корень. Использование: `sqrt(x)`
`log(x)` Натуральный логарифм. При вызове в виде `log(x, b)` возвращает логарифм по основанию `b`.
`e` Основание натуральных логарифмов e = 2,71828… `pi` Константа π = 3.1415...
`sin(x)` Синус угла, задаваемого в радианах
`cos(x)` Косинус угла, задаваемого в радианах
`tan(x)` Тангенс угла, задаваемого в радианах
`asin(x)` Арксинус, возвращает значение в радианах
`acos(x)` Арккосинус, возвращает значение в радианах
`atan(x)` Арктангенс, возвращает значение в радианах
`atan2(y, x)` Полярный угол (в радианах) точки с координатами (x, y).
`degrees(x)` Преобразует угол, заданный в радианах, в градусы.
`radians(x)` Преобразует угол, заданный в градусах, в радианы.

Python для роботов

Библиотека math

```
import math
```

`round(x)` Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
`abs(x)` Модуль (абсолютная величина). Это — стандартная функция.
`sqrt(x)` Квадратный корень. Использование: `sqrt(x)`
`log(x)` Натуральный логарифм. При вызове в виде `log(x, b)` возвращает логарифм по основанию `b`.
`e` Основание натуральных логарифмов e = 2,71828… `pi` Константа π = 3.1415...
`sin(x)` Синус угла, задаваемого в радианах
`cos(x)` Косинус угла, задаваемого в радианах
`tan(x)` Тангенс угла, задаваемого в радианах
`asin(x)` Арксинус, возвращает значение в радианах
`acos(x)` Арккосинус, возвращает значение в радианах
`atan(x)` Арктангенс, возвращает значение в радианах
`atan2(y, x)` Полярный угол (в радианах) точки с координатами (x, y).
`degrees(x)` Преобразует угол, заданный в радианах, в градусы.
`radians(x)` Преобразует угол, заданный в градусах, в радианы.

$$Z = \frac{9\pi t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^{x}$$

Python для роботов

Разбор задач

$$Z = \frac{9\pi t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^x$$

```
import math

t = 10
x = 5

z = (9*math.pi*t + 10 * math.cos(x)) * math.e**x / (math.sqrt(t) -
abs(math.sin(t)))

print(round(z,2))
```

Python для роботов

ООП

Объектно-ориентированное программирование (ООП) — это парадигма программирования, где различные компоненты компьютерной программы моделируются на основе реальных объектов. **Объект** — это что-либо, у чего есть какие-либо характеристики и то, что может выполнить какую-либо функцию.

Python для роботов

Класс и его экземпляр

Каждый **объект** является экземпляром некоторого **класса**.

Класс это некий чертеж будущего объекта. Как план строительства дома, или чертеж детали. А экземпляр - это конкретная реализация чертежа т.е. построенный дом или выточенная деталь.

```
class Book():
    title = ""
    author = ""

    def print_book(self):
        print(self.title, self.author)

b = Book()
b.print_book()
```

`title, author` - поле, переменная, атрибут класса

`print_book()` - метод класса

Python для роботов

Разбор задач

```
def rotate_left(triple):
    old_triple = list(triple)
    new_triple = [old_triple[1],old_triple[2],old_triple[0]]
    print(tuple(new_triple))

def rotate_right(triple):
    old_triple = list(triple)
    new_triple = [old_triple[2],old_triple[0],old_triple[1]]
    print(tuple(new_triple))

triple = ('A','B','C')

rotate_left(triple)
rotate_right(triple)
```

Вам предстоит реализовать две функции, которые "вращают" тройку влево и вправо. Как это выглядит, вы можете понять из пары примеров:

```
>>> triple = ('A', 'B', 'C')
>>> rotate_left(triple)
('B', 'C', 'A')
>>> rotate_right(triple)
('C', 'A', 'B')
```

Python для роботов

Разбор задач

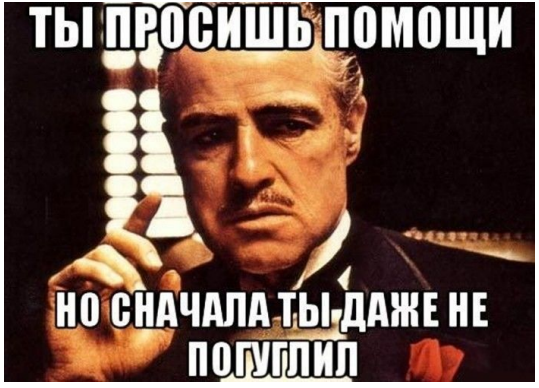
```
def rotate_left(triple):
    new_triple = [triple[1],triple[2],triple[0]]
    print(new_triple)

def rotate_right(triple):
    new_triple = [triple[2],triple[0],triple[1]]
    print(new_triple)

triple = ['A','B','C']

rotate_left(triple)
rotate_right(triple)
```

Вам предстоит реализовать две функции, которые "вращают" тройку влево и вправо. Как это выглядит, вы можете понять из пары примеров:

```
>>> triple = ('A', 'B', 'C')
>>> rotate_left(triple)
('B', 'C', 'A')
>>> rotate_right(triple)
('C', 'A', 'B')
```

Python для роботов

Разбор задач



Основы ROS

Основы ROS

Установка и запуск ROS

Инструкция установки ROS для Ubuntu
http://wiki.ros.org/noetic/Installation/Ubuntu

Настройка рабочего окружения
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

Запуск мастер-ноды ROS: roscore

Основы ROS

Базовые понятия ROS

Мастер (Master), Мастер-Нода

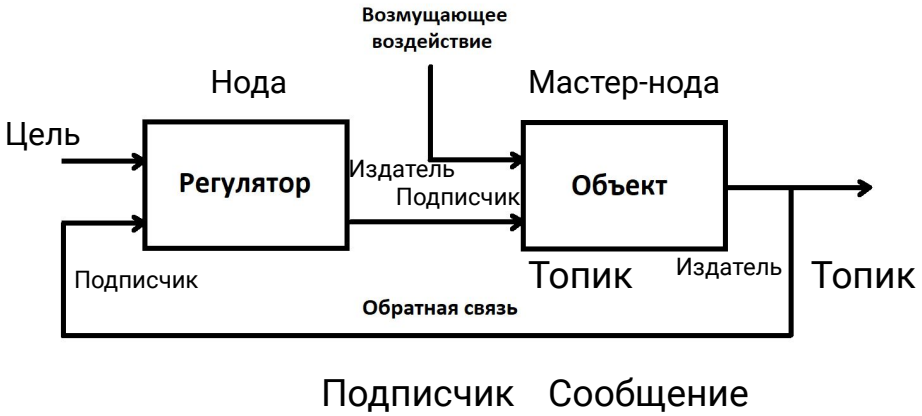
Нода (Node)

Сообщение (Message)

Топик (Topic)

Издатель (Publisher)

Подписчик (Subscriber)

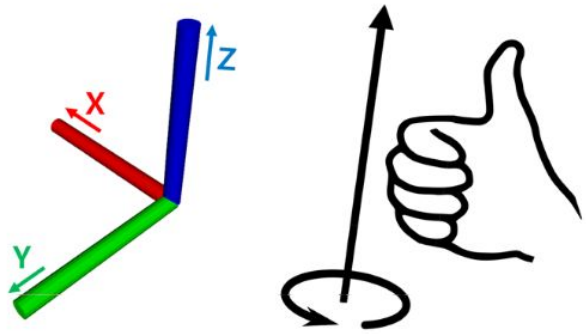


Основы ROS

Стандарты ROS Список всех стандартов https://ros.org/reps/rep-0000.html

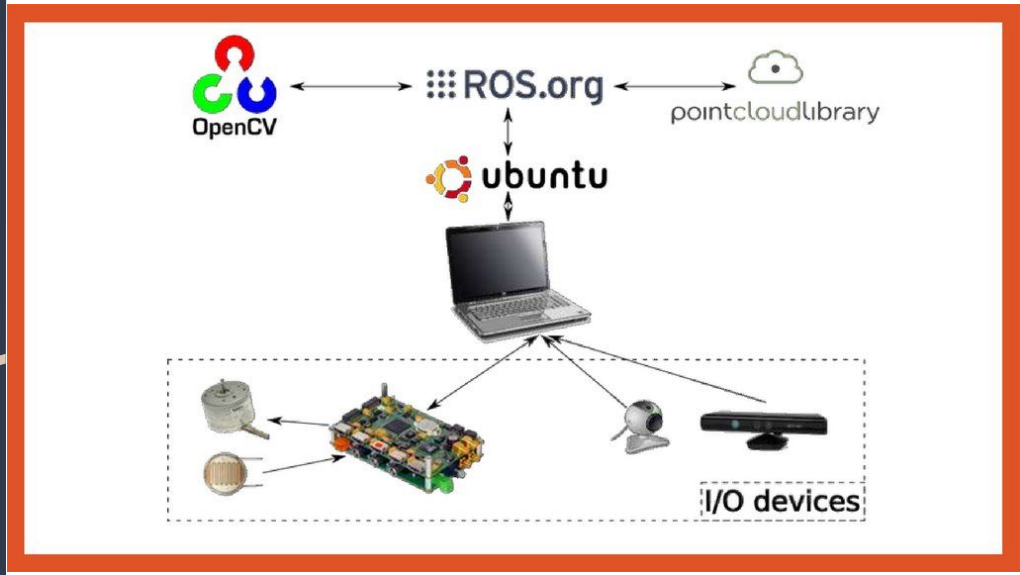
Объект	Правило	Пример
Пакет (Package)	under_scored	rst_ros_package
Topic, Service	under_scored	raw_image
File	under_scored	turtlebot3_fake.cpp
Variable	under_scored	string table_name;
Type	CamelCased	typedef int32_t PropertiesNumber;
Class	CamelCased	class UrlTable
Structure	CamelCased	struct UrlTableProperties
Function	camelCased	addTableEntry();
Method	camelCased	void setNumEntries(int32_t num_entries)
Constant	ALL_CAPITALS	const uint8_t DAYS_IN_A_WEEK = 7;
Macro	ALL_CAPITALS	#define PI_ROUNDED 3.0

Единицы измерений используемые в ROS, должны соответствовать единицам СИ



Оси x, y и z в ROS используют правило правой руки

Разработка в ROS



Разработка в ROS

Python для ROS

rospy - это клиентская библиотека Python для ROS. Библиотека позволяет разработчикам быстро взаимодействовать с Топиками и Сервисами.

Архитектура **rospy** отдает предпочтение скорости реализации (то есть времени, затраченному разработчиком) по сравнению с производительностью во время выполнения. Библиотека отлично подходит для быстрого прототипирования и тестирования в ROS. Многие инструменты ROS написаны на rospy, например, уже известные нам утилиты rostopic и rosservice.

Библиотека **rospy** устанавливается при установке ROS, поэтому отдельно устанавливать нам ничего не нужно.

Разработка в ROS

Программа Издатель

```
import rospy
from std_msgs.msg import String

rospy.init_node("nasha_noda")

pub = rospy.Publisher("welcome_topic", String)

s = String()
s.data = "Hello, robot"

rospy.sleep(0.5)
pub.publish(s)
```

Разработка в ROS

Программа Подписчик

```
import rospy
from std_msgs.msg import String

rospy.init_node("nasha_sub_node")

def callback(dannye):
    print(dannye)

rospy.Subscriber("welcome_topic", String, callback)

rospy.spin()
```

Разработка в ROS

Взаимодействие Подписчика и Издателя в рамках одной ноды

```
import rospy
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
rospy.init_node('one_meter_node')
pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
def callback_regulator(msg):
    vel = Twist()
    if msg.x >= 7:
        vel.linear.x = 0
    else:
        vel.linear.x = 0.1
    pub.publish(vel)
rospy.Subscriber('/turtle1/pose', Pose, callback_regulator)
rospy.spin()
```

Разработка в ROS

Взаимодействие Подписчика и Издателя в рамках одной ноды 2

```
import rospy
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
class RobotMover():
    def __init__(self):
        self.vel = Twist()
        self.distance_passed = 0
        self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
        rospy.Subscriber('/odom', Odometry, self.callback_handler)
```

Разработка в ROS

Взаимодействие Подписчика и Издателя в рамках одной ноды 2

```
    def callback_handler(self, msg):
        self.distance_passed =
msg.pose.pose.position.x

    def regulator(self):
        print(self.distance_passed)
        if self.distance_passed >= 1:
            self.vel_publisher(0)
        else:
            self.vel_publisher(0.1)

    def vel_publisher(self, vel_value):
        self.vel.linear.x = vel_value
        self.pub.publish(self.vel)
```

```
rospy.init_node('one_meter_node')
r = RobotMover()
while not rospy.is_shutdown():
    rospy.sleep(0.2)
    r.regulator()
```

Разработка в ROS

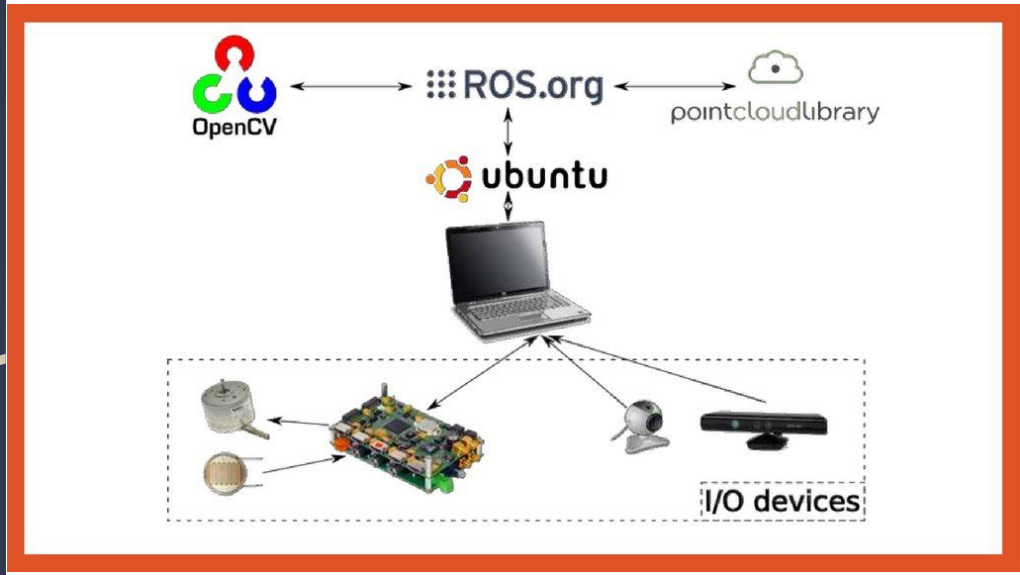
Разбор задач

```python
import rospy
import math

from geometry_msgs.msg import Twist
from turtlesim.msg import Pose

rospy.init_node("mover")

first_run = True

current_pose = Pose()
init_pose = Pose()
side_len = 1

pub = rospy.Publisher("/turtle1/cmd_vel", Twist, queue_size=10)

def odom_cb(pose):
    global current_pose
    current_pose = pose

rospy.Subscriber("/turtle1/pose", Pose, odom_cb)

def forward(des_dist):
    global init_pose, current_pose, first_run, side_len
    init_pose = current_pose
    dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
    while dist < des_dist:
        dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
        publish_vel(1,0)
    else:
        publish_vel(0,0)

def turn(des_angle):
    global init_pose, current_pose, first_run, side_len
    angle_turn = True
    while angle_turn:
        if abs(init_pose.theta - current_pose.theta) < des_angle:
            publish_vel(0,0.5)
        else:
            angle_turn = False
            publish_vel(0,0)


def publish_vel(vel_x, vel_z):
    vel = Twist()
    vel.linear.x = vel_x
    vel.angular.z = vel_z
    pub.publish(vel)

rospy.sleep(0.2)

for i in range(4):
    forward(1)
    turn(math.pi/2)
```

Доработать программу, чтобы при запуске программы робот черепаха двигалась по квадрату со сторонами равными 3 метрам.

В результате работы программы черепаха должна проехать все вершины квадрата и оказаться в точке старта. Ориентация черепахи в точке финиша должна совпадать с ориентацией в момент старта.

Продвинутая разработка в ROS



Продвинутая разработка в ROS

Сервис. Пример серверной части

```
import rospy
from rospy_tutorials.srv import AddTwoInts,AddTwoIntsResponse

rospy.init_node('sum_server')

def handle_callculation(req):
    print("Returning [%s + %s = %s]"%(req.a, req.b, (req.a + req.b)))
    return AddTwoIntsResponse(req.a + req.b)

rospy.Service('sum_service', AddTwoInts, handle_callculation)
print("Ready to add two ints.")

rospy.spin()
```

Продвинутая разработка в ROS

Сервис. Пример клиентской части

```
import rospy
import random
from rospy_tutorials.srv import AddTwoInts

def add_two_ints_client(a, b):
    rospy.wait_for_service('sum_service')
    service_caller = rospy.ServiceProxy('sum_service', AddTwoInts)
    resp = service_caller(a, b)
    return resp.sum

a = random.randint(-100 , 100)
b = random.randint(-100 , 100)
print("%s + %s = %s"%(a, b, add_two_ints_client(a, b)))
```

Продвинутая разработка в ROS

Экшн-Сервер. Пример серверной части

```
import rospy
import actionlib

from actionlib_tutorials.msg import FibonacciAction, FibonacciFeedback,
FibonacciResult

class FibonacciActionCounter():
    def __init__(self):
        self.feedback = FibonacciFeedback()
        self.result = FibonacciResult()
        self.asr = actionlib.SimpleActionServer('FB_counter', FibonacciAction,
self.execute_cb)
```

Продвинутая разработка в ROS

Экшн-Сервер. Пример серверной части

```
    def execute_cb(self, goal):
        success = True
        self.feedback.sequence = []
        self.feedback.sequence.append(0)
        self.feedback.sequence.append(1)
        rospy.loginfo('Executing, creating fibonacci sequence of order%i' %
goal.order)
        for i in range(1, goal.order - 1):
            if self.asr.is_preempt_requested():
                rospy.loginfo('Action Preempted')
                self.asr.set_preempted()
                success = False
                break
```

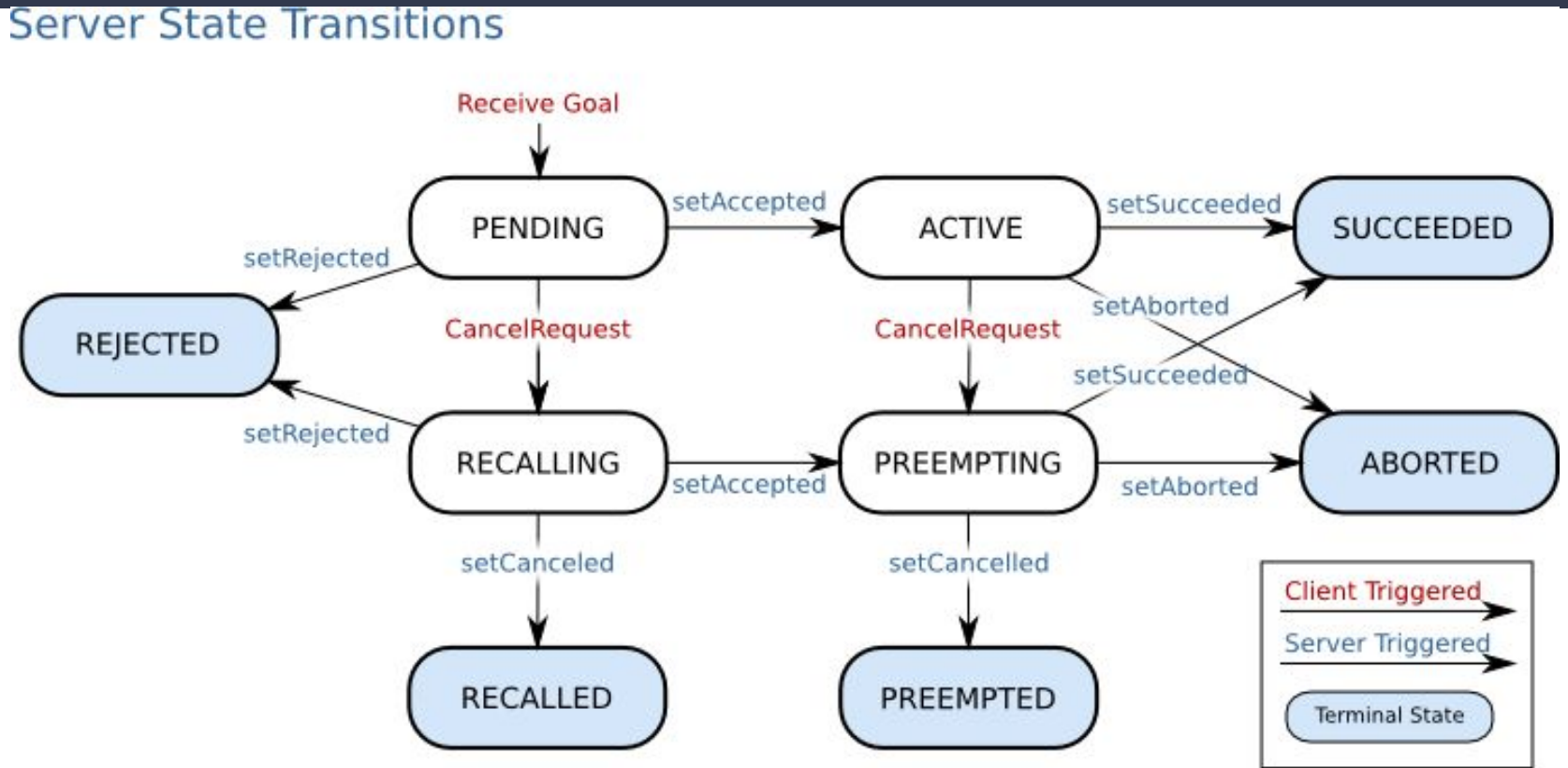
Продвинутая разработка в ROS

Экшн-Сервер. Пример серверной части

```
            self.feedback.sequence.append(self.feedback.sequence[i] +
self.feedback.sequence[i-1])
             self.asr.publish_feedback(self.feedback)
             rospy.sleep(1)
        if success:
            self.result.sequence = self.feedback.sequence
            rospy.loginfo('Succeeded')
            self.asr.set_succeeded(self.result)

rospy.init_node('fibonacci')
server = FibonacciActionCounter()
rospy.spin()
```

Продвинутая разработка в ROS

Экшн-Сервер. Пример серверной части



Продвинутая разработка в ROS

Экшн-Сервер. Пример клиентской части

```
import rospy
import actionlib
from actionlib_tutorials.msg import FibonacciAction, FibonacciGoal

class FibonacciClient():
    def __init__(self):
        self.client = actionlib.SimpleActionClient('FB_counter',
FibonacciAction)

    def fibonacci_send_goal(self):
        goal = FibonacciGoal
        goal.order = 10
        self.client.wait_for_server()
        self.client.send_goal(goal, done_cb = self.done_callback)
```

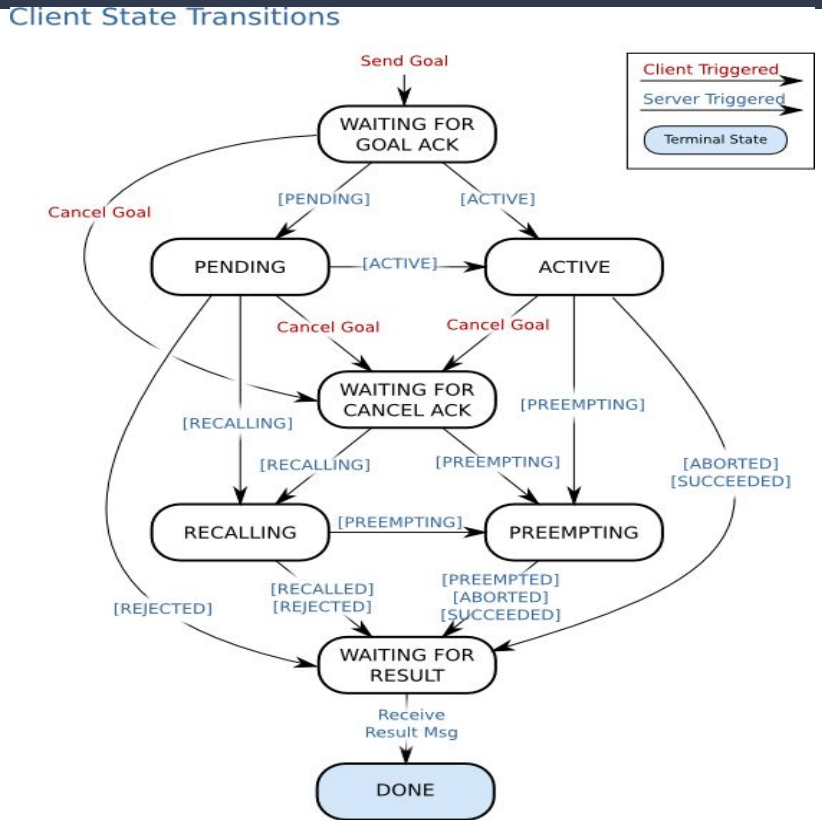
Продвинутая разработка в ROS

Экшн-Сервер. Пример клиентской части

```
    def done_callback(self, _, result):
        print(result)

rospy.init_node('fibonacci_client_py')
fib = FibonacciClient()
fib.fibonacci_send_goal()
print("Do something else")
rospy.spin()
```

Продвинутая разработка в ROS

Экшн-Сервер. Пример клиентской части



Продвинутая разработка в ROS

Параметры в ROS

Сервер параметров может хранить только простые переменные, числа, строки и тп. Он не создан для хранения сложных типов данных, например сообщений ROS.

Для работы с параметрами есть удобная консольная утилита **rosparam**

rosparam list: Получить список параметров
rosparam get parameter: Прочитать значение параметра
rosparam set parameter value: Установить значение параметра
rosparam delete parameter: Удалить значение параметра
rosparam dump file: Сохранить все переменные в файл
rosparam load file: Восстановить переменные из файла

Продвинутая разработка в ROS

Параметры в ROS, пример работы с параметрами на Python

```
import rospy
rospy.init_node('params_demo')
default_param = "some_value"

try:
    my_param = rospy.get_param("~my_param")
except KeyError as e:
    my_param = default_param
    rospy.set_param("~my_param", my_param)

print(my_param)
```

Продвинутая разработка в ROS

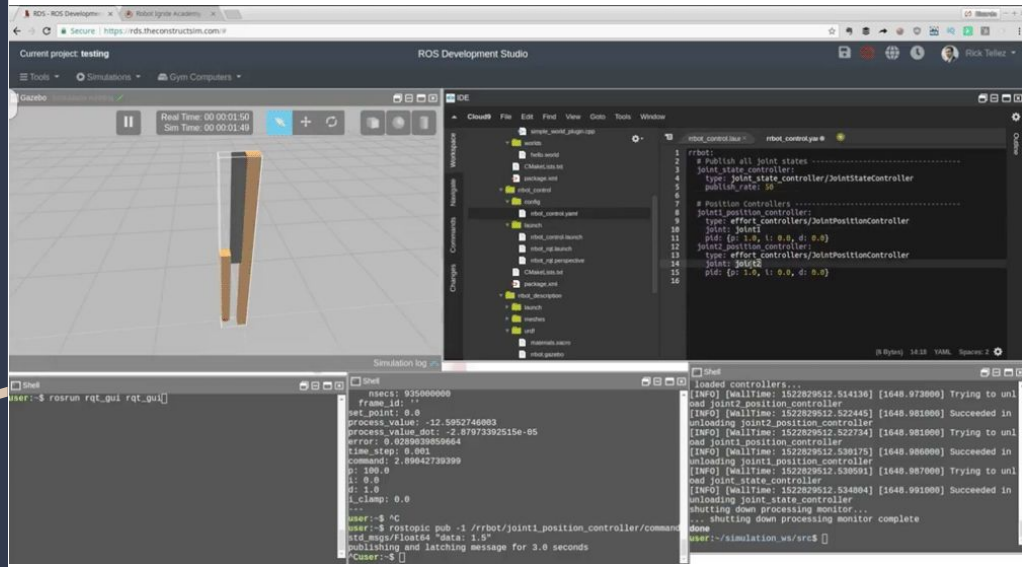
.bag файлы

Идея использования ,bag файлов довольно проста и немного похожа на машину времени. Мы можем включить "запись" и сохранять все сообщения, которые происходят в нашей системе, а результат этой записи сохраняется в .bag файлах.

Для работы с .bag служит консольная утилита rosbag, перечислим основные ее аргументы

rosbag record [TOPIC_NAME] Начать запись .bag
rosbag play [FILE_NAME] Проиграть .bag файл
rosbag compress [FILE_NAME] Архивировать файл
rosbag decompress [FILE_NAME] Разархивировать файл
rosbag record -a Записать все сообщения

Администрирование ROS



Пакеты в ROS. Утилита Catkin.

Структура пакета

Пакет ROS содержат множество различных файлов. Для того, чтобы было проще ориентироваться с файлами любого пакета, сообщество разработчиков рекомендует использовать единообразную файловую структуру пакета:

- bin/ Директория, в которой хранятся скомпилированные программы
- include/ Директория содержит файлы с заголовками (headers) библиотек
- launch/ Директория для хранения файлов конфигурации запуска .launch
- msg/ Директория для сообщений (для топиков)
- src/ Директория для хранения исходников программ (в том числе и скриптов)
- srv/ Директория для хранения сообщений для использования Сервисами (Services)
- CMakeLists.txt: Файл форматы Cmake с инструкциями для установки пакета
- package.xml Файл "манифест" для описания пакета

Пакеты в ROS. Утилита Catkin.

Установка пакета из репозитория

Поиск пакета ROS

apt seach ros-noetic-Имя

Если вы нашли пакет pkg_name на wiki, то скорее всего в системе apt будет назваться ros-noetic-pkg_name

Установка нового пакета происходит стандартным способом

sudo apt install ros-noetic-pkg_name

Пакеты в ROS. Утилита Catkin.

Сборка пакета из исходников

1. Настройка catkin_ws
2. Копирование директории пакета в catkin_ws/src/
3. catkin_make --pkg=pkg_name

Пакеты в ROS. Утилита Catkin.

создание собственного пакета

Проще всего создавать пакет при помощи утилиты **catkin_create_pkg**, она сама создаст необходимые файлы и директории и пропишет зависимости.

Перейдем в директорию, где расположены исходники пакетов

cd ~/catkin_ws/src

И создадим наш “пока пустой” пакет

catkin_create_pkg my_first_package

Пакеты в ROS. Утилита Catkin.

создание собственного типа сообщений

1. Сначала сделаем файл содержащий описание нашего сообщения

mkdir msg → **cd msg** → **nano move.msg** →
float32 s
float32 v

2. Модифицируем файл **CMakeLists.txt**

```
add_message_files(
  FILES
  move.msg)
```

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

```
catkin_package(
CATKIN_DEPENDS std_msgs
message_runtime)
```

3. Убедимся что в файле **package.xml** есть следующие строки:

```
<exec_depend>message_runtime</exec_depend>
<build_depend>std_msgs</build_depend>
<exec_depend>std_msgs</exec_depend>
```

4. Соберем пакет стандартным образом `catkin_make --pkg=my_first_package`

Пакеты в ROS. Утилита Catkin.

Launch файлы

Лаунч файлы - запускают работу программ для РОС, по пользовательским сценариям.

Создавать лаунч файлы надо в папке **launch** вашего проекта, тогда при запуске при помощи утилиты roslaunch, ROS сам найдет файл, в папке.

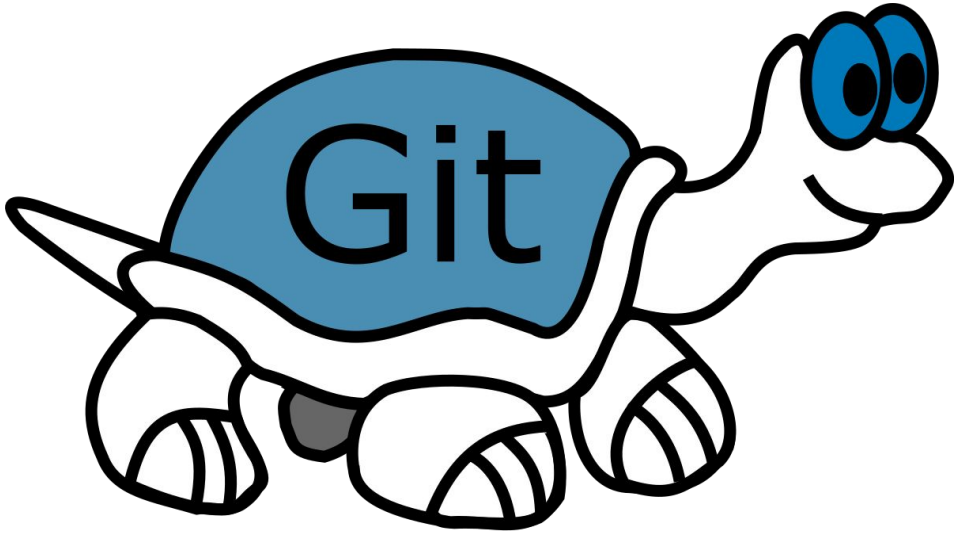
Пример самого простого лаунч файла:

```
<launch>
<node name="имя_ноды(любое)" pkg="имя_пакета" type="имя_файла.py"/>
</launch>
```

Не забудьте сделать программу исполняемой

```
cd ../src
chmod +x имя_файла.py
```

Использование git



Использование git

Launch файлы

Лаунч файлы - запускают работу программ для POC, по пользовательским сценариям.

Создавать лаунч файлы надо в папке **launch** вашего проекта, тогда при запуске при помощи утилиты roslaunch, ROS сам найдет файл, в папке.

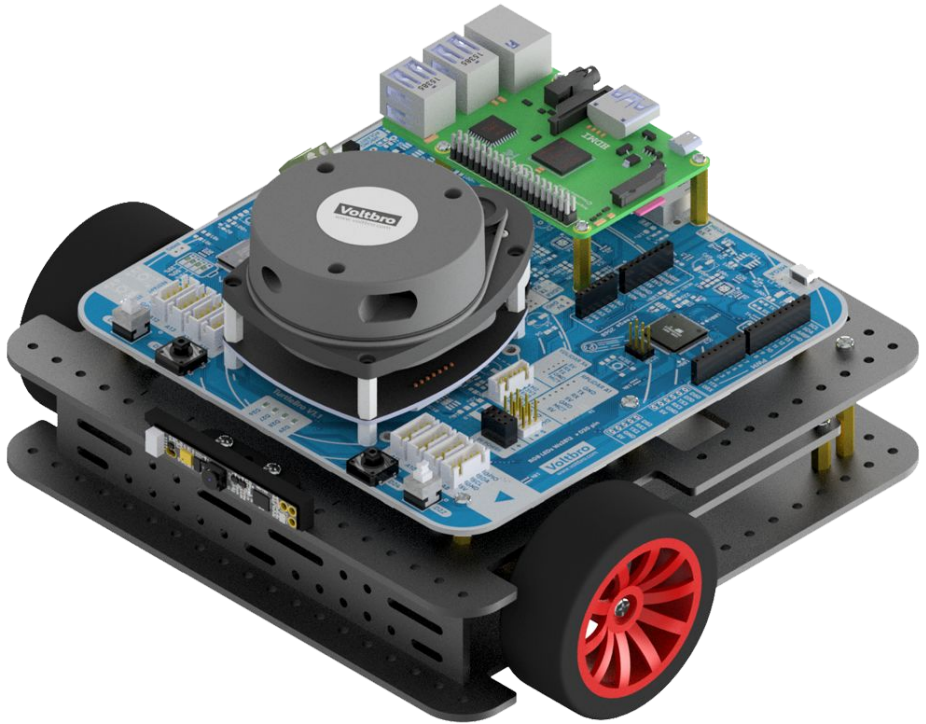
Пример самого простого лаунч файла:

```
<launch>
<node name="имя_ноды(любое)" pkg="имя_пакета" type="имя_файла.py"/>
</launch>
```

Не забудьте сделать программу исполняемой

```
cd ../src
chmod +x имя_файла.py
```

Подключение к роботу, работа с инструкцией получение информации о роботе



Робот TurtleBro

Инструкция

Образовательный робот **TurtleBro** разработан для изучения основ современной робототехники на примере мета-операционной системы Robot Operating System (ROS), работающей в среде Linux.у программ для РОС, по пользовательским сценариям.

Инструкция к роботу лежит тут: https://manual.turtlebro.ru

Робот TurtleBro

Подключение робота к сети

1. По умолчанию при старте raspberry попытается подключиться к WiFi точке доступа с параметрами

SSID: TurtleBro	SSID: TurtleBro5G
Pass: turtlew001	Pass: turtlew001

2. Подключите работающий Raspberry к Ethernet кабелю. Определите IP адрес устройства через web-интерфейс роутера. Зайдите на ssh на устройство ssh pi@IPили pi@turtlebroNNN.local Откройте в редакторе mcedit файл wpa_supplicant.conf

sudo mcedit /etc/wpa_supplicant/wpa_supplicant.conf

В конец файла добавьте настройки WiFi вашей сети в виде:

```
network={
  ssid="WIFI_NETWORK_NAME"
  psk="wifipassword"
}
```

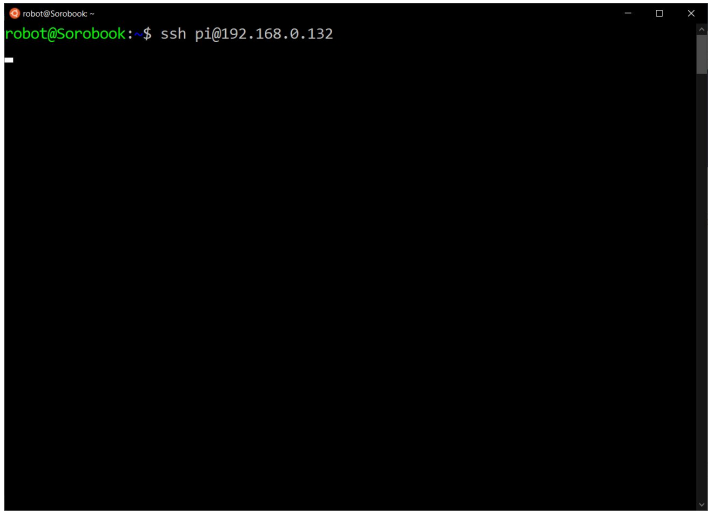
При указании пароля от wi-fi сети, обязательно наличие кавычек. Сохраните файл и перезагрузите raspberry.

Робот TurtleBro

Подключение к роботу

Условия подключения:

1. Ваш компьютер и робот в одной сети. По умолчанию это сеть **TutleBro** с паролем **turtlew001**
2. Вы знаете IP адрес робота
3. На вашем компьютере есть ssh клиент
4. Прямое подключение ssh pi@ip_robot
5. Пароль пользователя pi: **brobro** (есть в инструкции)



Робот TurtleBro

Доступные на роботе топики

Топик /bat Содержит данные о состоянии подключенной к плате батарейке. Тип сообщения sensor_msgs/BatteryState

Топик /cmd_vel Топик управления перемещения роботом. Тип сообщения geometry_msgs/Twist

Топик /imu Данные инерционного датчика (Inertial measurement unit), включающего в себя гироскоп, акселерометр и компас. Тип сообщения sensor_msgs/Imu

Топик /odom Данные одометрии (положения робота, рассчитанного на основании вращения колес). В текущей версии повороты робота рассчитываются по данным IMU датчика, а перемещение на основе данных энкодеров на моторах. Тип сообщения nav_msgs/Odometry

Топик /scan Данные, полученные с лидара (облако точек). Данные идут через Serial интерфейс лидара, USB-UART преобразователь и через USB hub. Тип сообщения sensor_msgs/LaserScan

Робот TurtleBro

Доступные на роботе сервисы

Сервис /reset Сервис вызывает сброс одометрии (данные /odom) и текущих установленных скоростей (/cmd_vel) робота.

Сервис /set_pid Сервис устанавливает значения ПИД регулятора для управления моторами тележки. Тип сообщения turtlebro/PidRegulator

Сервис /board_info Сервис выдает актуальную информацию о версии прошивки системной платы и уникальный номер процессора

Сервис /start_motor Сервис включает мотор вращения лидаром (для RPLidar). Без параметров

Сервис /stop_motor Сервис выключает мотор вращения лидаром (для RPLidar). Без параметров

Робот TurtleBro

Получение информации о роботе

1. Название дистрибутива Linux и кодовое имя сборки
2. Версия интерпретатора python и верия библиотеки rospy
3. Версия дистрибутива ROS
4. Версия пакета turtlebro
5. Версия прошивки микроконтроллера материнской платы
6. Серийный номер системной платы робота (mcu_id)
7. Размер оперативной памяти
8. Полное / Свободное пространство на SD карте (Gb)

1. uname -a
2. python3 -V pip3 show rospy
3. rosversion -d
4. rosversion turtlebro
5. rosservice call /board_info {}
6. rosservice call /board_info {}
7. cat /proc/meminfo | grep MemTotal
8. df -h

Робот TurtleBro

Проверки робота

Проверка батареи Вывод напряжения батареи в топике rostopic echo /bat -n 1

Проверка Лидара Облако точек лидара в топике rostopic echo /scan -n 1

Проверка одометрии колес В первом терминале: Команда: rostopic echo /odom

В другом терминале команда: rostopic pub /cmd_vel geometry_msgs/Twist (x = 0.05)

Команда: rostopic pub /cmd_vel geometry_msgs/Twist (x = -0.05)

Команда: rostopic pub /cmd_vel geometry_msgs/Twist (x = 0; z = 0.5)

Команда: rostopic pub /cmd_vel geometry_msgs/Twist (x = 0; z = -0.5)

Проверка IMU датчика Данные IMU датчика в консоли rostopic echo /imu -n 1

Робот TurtleBro

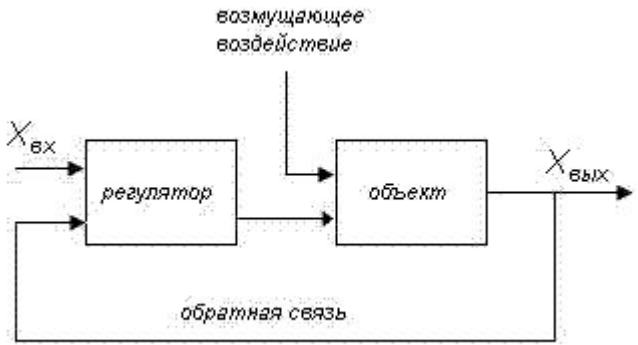
Проверка работы камеры

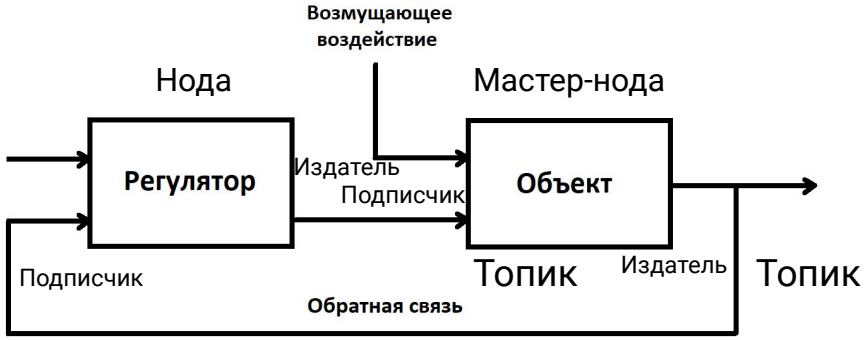
1. Проверить наличие подключенных устройств в /dev/
2. Получить данные о максимальном разрешения работы камеры
3. Проверить работу камеры через web интерфейс

1. cat /dev/ | grep video
2. v4l2-ctl --list-formats-ext
3. 10.8.0.6:8080

Управление роботом

Управление роботом





Подписчик Сообщение

Управление роботом

Робот лазерный дальномер

Автономная навигация

Работа с лидаром

За один оборот, лидар получает 360 точек с данными расстояния от центра лидара до препятствия, которое встретилось на пути лазерного луча. Каждая из 360 точек соответствует одному градусу оборота лидара.

“Сырые” данные с лидара находятся в топике scan с типом данных LaserScan


```
header:
  seq: 1998463
  stamp:
    secs: 1628508496
    nsecs: 100944372
  frame_id: "base_scan"
angle_min: -3.12413907051
angle_max: 3.14159274101
angle_increment: 0.0174532923847
time_increment: 0.000356173579348
scan_time: 0.127866312861
range_min: 0.15000000596
range_max: 12.0
ranges: [inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, 0.1379999965429306, 0.14149999618530273, inf, 0.145750001072883 6,
0.1537500023841858, inf, 0.1667499989271164, inf, 0.18250000476837158, 0.20225000381469727, inf, 0.2329999953508377, inf, inf, 0.26524999737739563, inf, inf, inf, inf, inf, 0.34424999356269836, inf, inf, 0.4137499928474426, inf, 0.4129999876022339, inf, 0.4127500057220459, 0.41225001215934753,
0.4115000069141388, inf, 0.4115000069141388, 0.41324999928474426, inf, 0.414000004529953, 0.4137499928474426, inf, 0.4154999852180481, 0.41725000739097595, 0.4189999997615814, inf, 0.42124998569488525, 0.4242500066757202, inf, 0.42649999260902405, inf, 0.4300000071525574,
0.4332500100135803, 0.43700000643730164, inf, 0.4415000081062317, 0.44624999165534973, inf, 0.45100000500679016, 0.45649999380111694, inf, 0.4622499942779541, 0.4684999883174896, inf, 0.47574999928474426, 0.4830000102519989, inf, 0.4909999966621399, 0.49924999475479126, inf,
0.5074999928474426, 0.5184999704360962, inf, 0.5287500023841858, 0.5410000085830688, inf, 0.5537499785423279, 0.5669999718666077, inf, 0.5827500224113464, inf, 0.5982499718666077, 0.6157500147819519, 0.6355000138282776, inf, 0.656000018119812, inf, 0.6779999732971191,
0.703499972820282, inf, 0.7294999957084656, 0.7609999775886536, inf, 0.7952499985694885, 0.8307499885559082, inf, 0.871749997138977, 0.9182500243186951, inf, 0.96875, 1.0497499704360962, inf, inf, 1.159000039100647, inf, 1.2607500553131104, inf, 1.3697500228881836, 1.4902499914169312,
inf, 1.6449999809265137, 1.8350000381469727, inf, 2.066999912261963, 2.4547500610351562, inf, 2.3559999465942383, inf, inf, inf, 2.3434998989105225, inf, 2.336750030517578, 2.33174991607666, 2.335249900817871, inf, 2.323499917984009, 2.335750102996826, inf, 2.336250066757202,
2.3389999866485596, inf, 2.3494999408721924, 2.3524999618530273, inf, 2.3615000247955322, 2.375, inf, 2.3935000896453857, 2.4094998836517334, 2.4327499866485596, inf, 2.4514999389648438, 2.4755001068115234, inf, 2.484999895095825, 2.513000011444092, inf, 2.5315001010894775,
2.5577499866485596, inf, 2.591249942779541, 2.6232500076293945, inf, 2.645250082015991, 2.700000047683716, 2.5959999561309814, inf, 2.5137500762939453, 2.4114999771118164, inf, 2.321000099182129, 2.243000030517578, inf, 2.177500009536743, 2.119499921798706, inf,
2.0577499866485596, 2.005500078201294, 1.9587500095367432, inf, 1.9142500162124634, 1.8769999742507935, inf, 1.8365000486373901, 1.874750018119812, inf, 1.75600004196167, 1.725000023841858, 1.687749981880188, inf, 1.659250020980835, 1.6352499723434448, inf, 1.6030000448226929,
1.5839999914169312, inf, 1.562999963760376, 1.5437500476837158, inf, 1.5237499475479126, 1.5077500343322754, 1.4915000200271606, inf, 1.4739999771118164, 1.46274995803833, inf, 1.4527499675750732, 1.437000036239624, inf, 1.4294999837875366, 1.4140000343322754,
1.4105000495910645, inf, 1.4049999713897705, 1.3969999551773071, inf, 1.3940000534057617, 1.3912500143051147, inf, 1.3880000114440918, 1.3839999437332153, inf, 1.377500057220459, 1.3819999694824219, inf, 1.378999948501587, inf, inf, inf, inf, inf, 0.5249999761581421, inf, inf, inf, inf, inf,
inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, 0.14124999940395355, 0.13899999856948853, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf,
inf]
intensities: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 13.0, 15.0, 0.0, 15.0, 15.0, 0.0,
15.0, 0.0, 15.0, 15.0, 0.0, 11.0, 0.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0, 0.0, 15.0, 0.0, 0.0, 15.0, 0.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0,
0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 0.0, 15.0, 0.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 0.0, 0.0, 0.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0,
15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0,
15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 15.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 15.0, 15.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Управление роботом

Робот лазерный дальномер

Общий алгоритм решения

1. Подписываемся на топик /scan
2. Читаем структуру LaserScan
3. “Вырезаем” из всего массива ranges - нулевой элемент
4. Выводим значение расстояния в нулевом элементе на экран

Управление роботом

Датчики робота

Робот лазерный дальномер

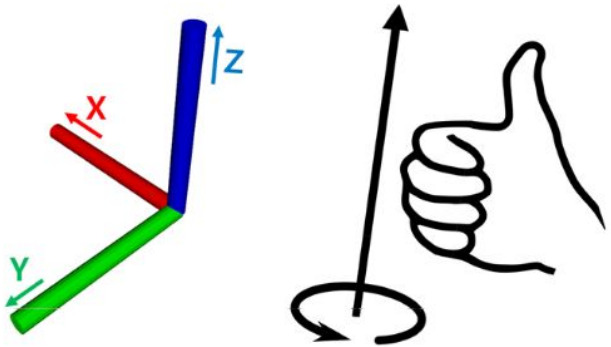
```
import rospy
import math
from sensor_msgs.msg import LaserScan
rospy.init_node('dalnomer')
def callback(scan):
    print(scan.ranges[0])
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
```

Управление роботом

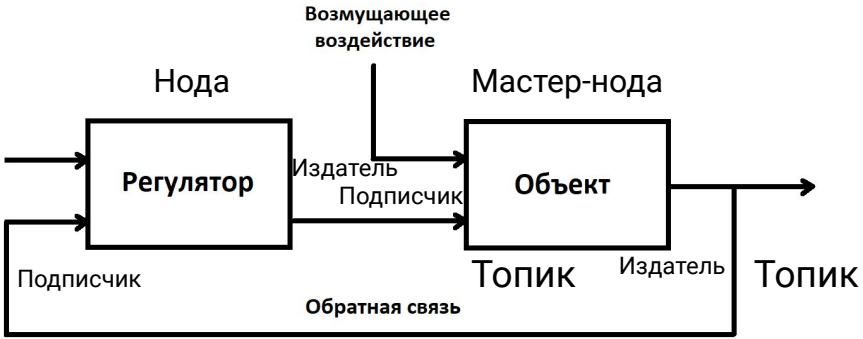
Управление движением робота

Топик /cmd_vel Топик управления перемещения роботом. Тип сообщения geometry_msgs/Twist

geometry_msgs/Vector3 linear
float64 x
float64 y
float64 z
geometry_msgs/Vector3 angular
float64 x
float64 y
float64 z



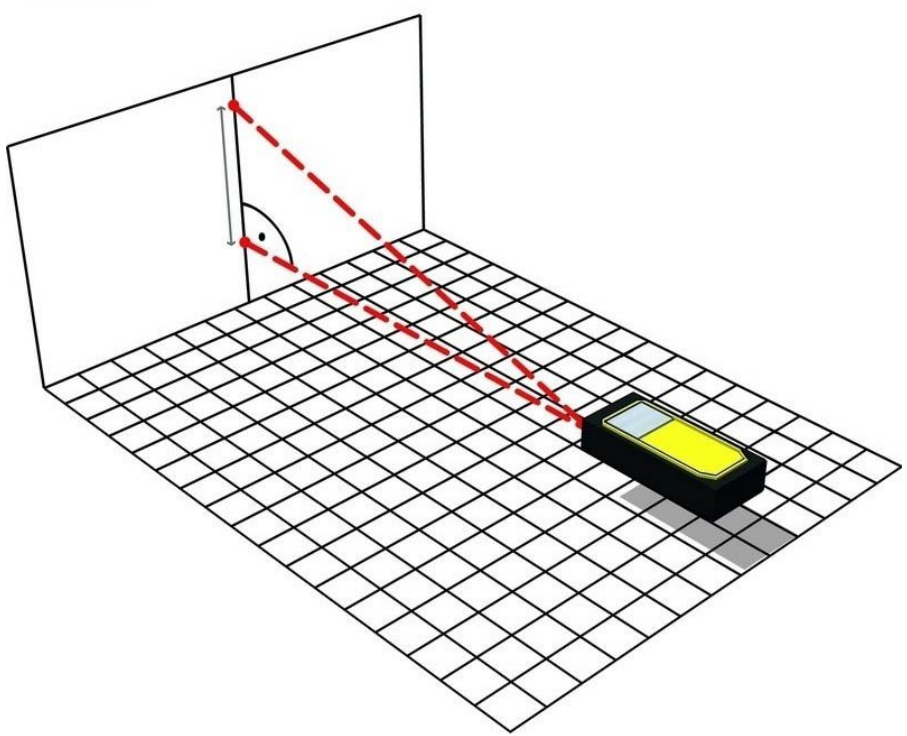
Управление роботом



Подписчик Сообщение

Управление роботом

Подъезжаем к стене на роботе



Управление роботом

Подъезжаем к стене на роботе

Общий алгоритм решения

1. Задаем управляющий сигнал на движение робота к стене (pub, cmd_vel, Twist)
2. Подписываемся на топик /scan
3. Читаем структуру LaserScan
4. “Вырезаем” из всего массива ranges - нулевой элемент
5. В цикле регулятора сравниваем значение нулевого элемента массива со значением уставки (требуемого расстояния до стены)
6. Если расстояние до стены меньше уставки, отправляем управляющий сигнал на остановку робота

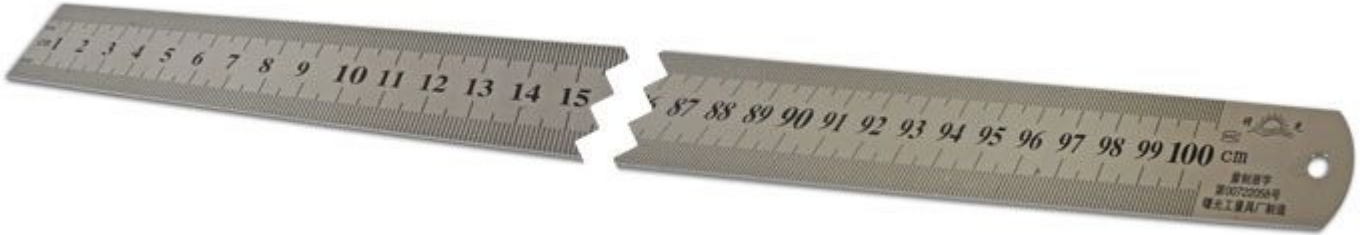
Управление роботом

Подъезжаем к стене на роботе

```
import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

rospy.init_node('dalnomer')
target = 0.3
vel = Twist()
pub = rospy.Publisher("cmd_vel", Twist, queue_size = 1)

def callback(scan):
    if scan.ranges[0] > target:
        vel.linear.x = 0.1
    else:
        vel.linear.x = 0
    pub.publish(vel)
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
```

Управление роботом

Едем один метр на роботе



Управление роботом

Едем один метр на роботе

Общий алгоритм решения

1. Задаем управляющий сигнал на движение робота к стене (pub, cmd_vel, Twist)
2. Подписываемся на топик /odom
3. Читаем структуру Odometry
4. “Вырезаем” из всей структуры Odometry значение пройденного расстояния по X
5. В цикле регулятора сравниваем значение пройденного расстояния со значением уставки (требуемого пройденного расстояния)
6. Если расстояние пройденное расстояние меньше уставки, отправляем управляющий сигнал на остановку робота

Управление роботом

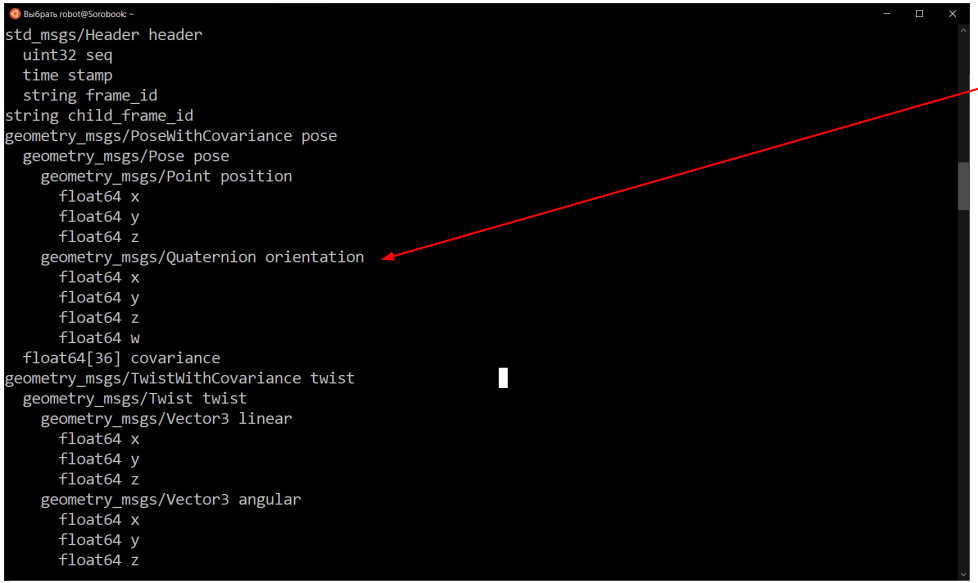
Едем один метр на роботе

```
import rospy
from nav_msgs.msg import Odometry
from geometry_msgs.msg import Twist

rospy.init_node('lineyka')
target = 0.3
vel = Twist()
pub = rospy.Publisher("cmd_vel", Twist, queue_size = 1)

def callback(odom):
    if odom.pose.pose.position.x < target:
        vel.linear.x = 0.1
    else:
        vel.linear.x = 0
    pub.publish(vel)
rospy.Subscriber("/scan", Odometry, callback)
rospy.spin()
```

Управление роботом

Данные о положении робота

Топик /odom Данные одометрии (положения робота, рассчитанного на основании вращения колес). В текущей версии повороты робота рассчитываются по данным IMU датчика, а перемещение на основе данных энкодеров на моторах. Тип сообщения nav_msgs/Odometry



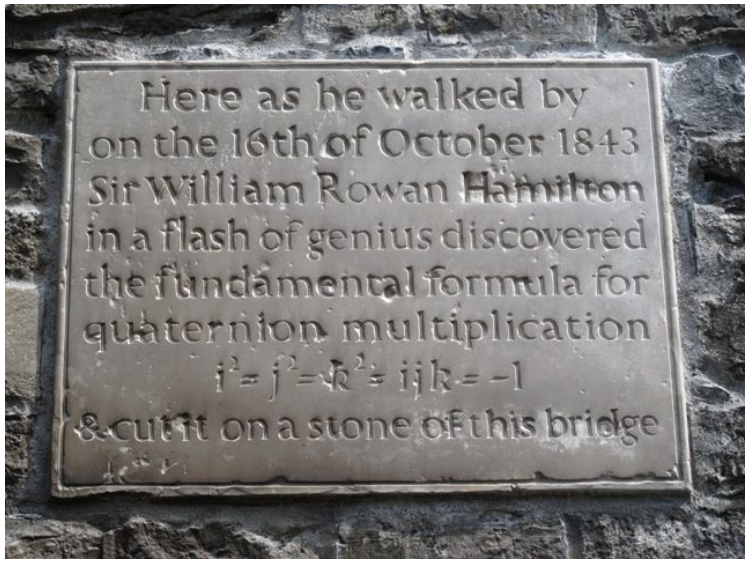
Обратите внимание, что ориентация робота задается кватернионом, а не привычными нам углами Эйлера

Управление роботом

Преобразование углов

Кватерни́оны (от лат. quaterni, по четыре) — система гиперкомплексных чисел, образующая векторное пространство размерностью четыре над полем вещественных чисел. Обычно обозначаются символом H. Предложены Уильямом Гамильтоном в 1843 году.

$$q = a + bi + cj + dk$$



Управление роботом

Преобразование углов

Кватерниóны (от лат. quaterni, по четыре) — система гиперкомплексных чисел, образующая векторное пространство размерностью четыре над полем вещественных чисел. Обычно обозначаются символом H. Предложены Уильямом Гамильтоном в 1843 году.

$$q = a + bi + cj + dk$$

Кватернион

$$\text{Axis} = (x, y, z)$$
$$\text{Angle} = \theta$$
$$\text{Quaternion} = \begin{bmatrix} x * \sin\left(\frac{\theta}{2}\right) \\ y * \sin\left(\frac{\theta}{2}\right) \\ z * \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

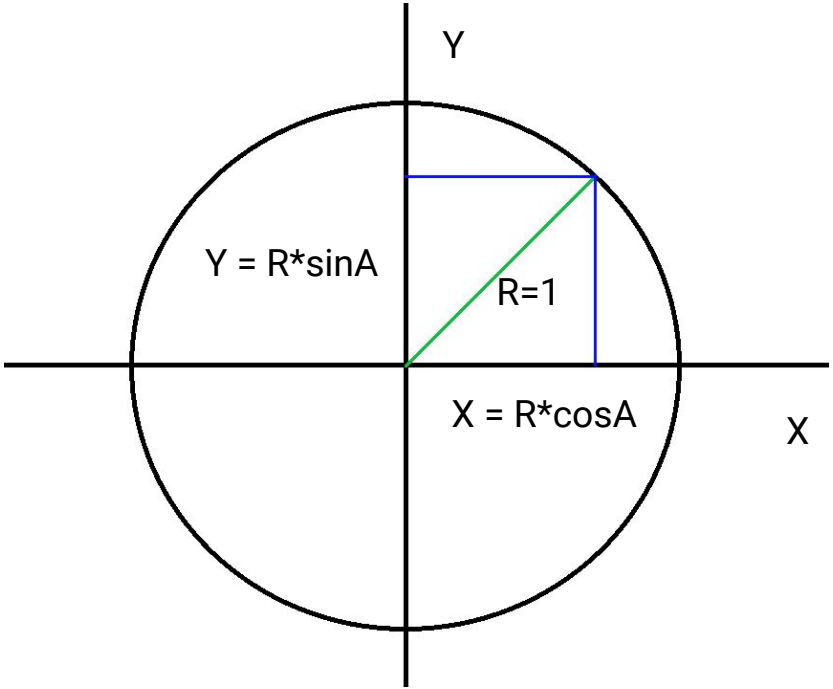
MyShared

где:

```
x = cos(A)
y = sin(A)
z = sin(B)
w = cos(w)
```

Управление роботом

Преобразование углов - Комплексные числа



a +ib == C, где i*i = -1

Сложение комплексных чисел = сложение векторов

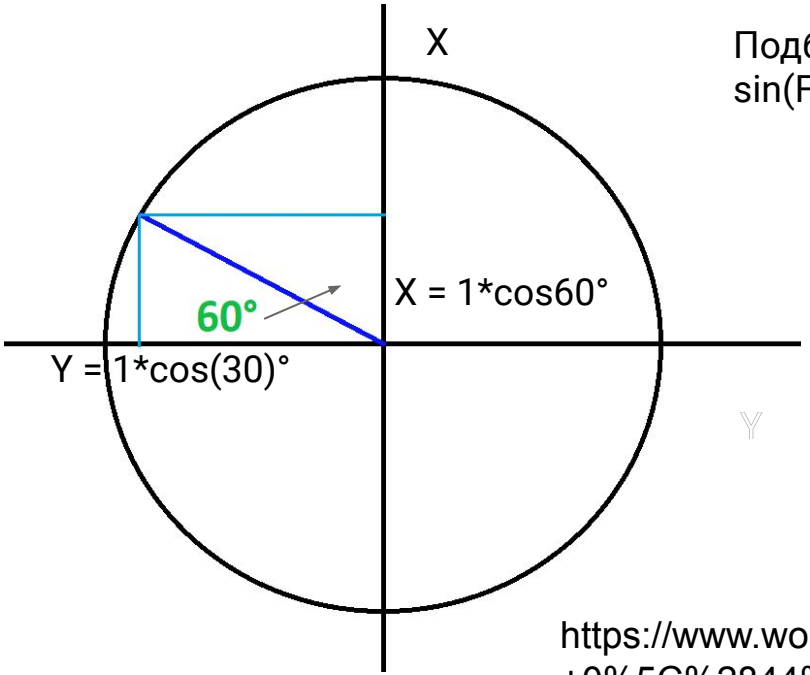
(a + ib) + (c + id) = a + c + i(b + d)

Умножение комплексных чисел = сложение углов
(a + ib) * (c + id) = ac - bd + i(bc + ad)

Управление роботом

Преобразование углов

Какому кватерниону соответствует поворот на 60 градусов влево без вращения



Подберем такой угол Fi, чтобы sin(Fi/2) = 1

W = cos(2Pi) = 0

X = cos(60) = 1 / 2 = 0.5

Y = cos (30) = 3^0,5 / 2 ~= 0.866

Z = 0, т.к. По Z нет вращения.

Кватернион будет (0 + 0,5i + 0,866j + 0k)

Кватернион

$$\text{Axis} = (x,\ y,\ z)$$
$$\text{Angle} = \theta$$
$$\text{Quaternion} = \begin{bmatrix} x * \sin\left(\frac{\theta}{2}\right) \\ y * \sin\left(\frac{\theta}{2}\right) \\ z * \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

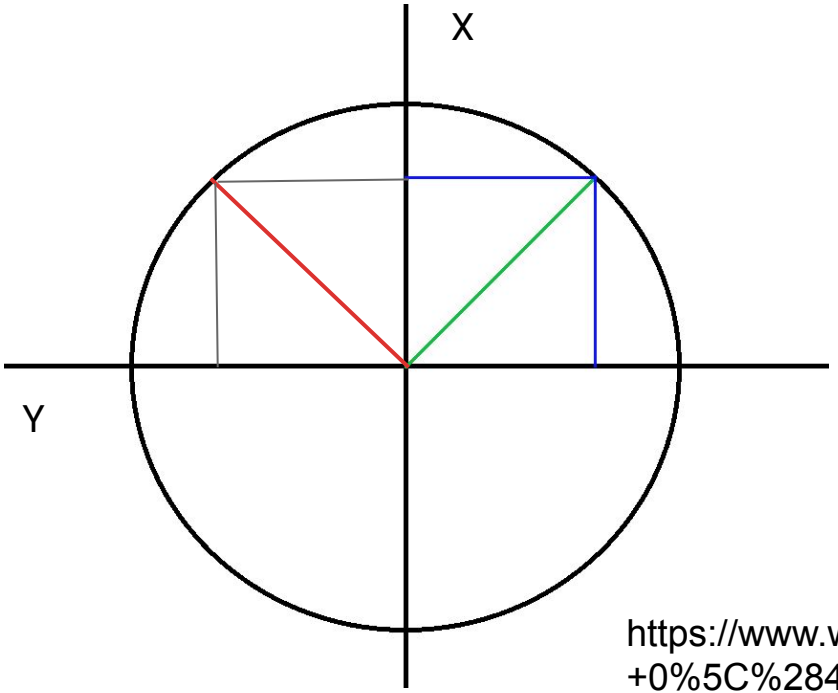
MyShared

https://www.wolframalpha.com/input/?i2d=true&i=quaternion%3A0+%2B+0%5C%2844%295i+%2B+0%5C%2844%29866j+%2B+0k

Управление роботом

Преобразование углов

Какому углу поворота соответствует кватернион 0 + 0,707i - 0,707j + 0k



X^2 + Y^2 + Z^2 = 1

W = 0 = cos(2Pi)

x = 0,707 = cos(45) или cos (-45)

y = - 0,707 = sin (- 45)

Z = 0 т.к. sin Z = 0

Т.е. Угол будет 45° вправо от оси X

Кватернион

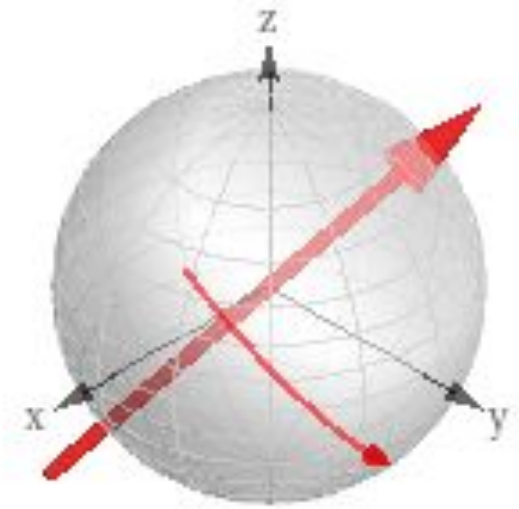
$$\text{Axis} = (x,\ y,\ z)$$
$$\text{Angle} = \theta$$
$$\text{Quaternion} = \begin{bmatrix} x * \sin\left(\frac{\theta}{2}\right) \\ y * \sin\left(\frac{\theta}{2}\right) \\ z * \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

https://www.wolframalpha.com/input/?i2d=true&i=quaternion%3A0+%2B+0%5C%2844%29707i+-+0%5C%2844%29707j+%2B+0k

Управление роботом

Преобразование углов

Какому углу поворота соответствует кватернион 0 - 0,707i + 0,707j + 0,5k



X^2 + Y^2 + Z^2 = 1

W = 0 = cos(2Pi)

X = - 0,707 = cos (135) или cos(225)

Y = 0,707 = sin (135)

Z = 0,5 или sin(30)

Угол будет 135° от оси X и 30° вверх от горизонта

Кватернион

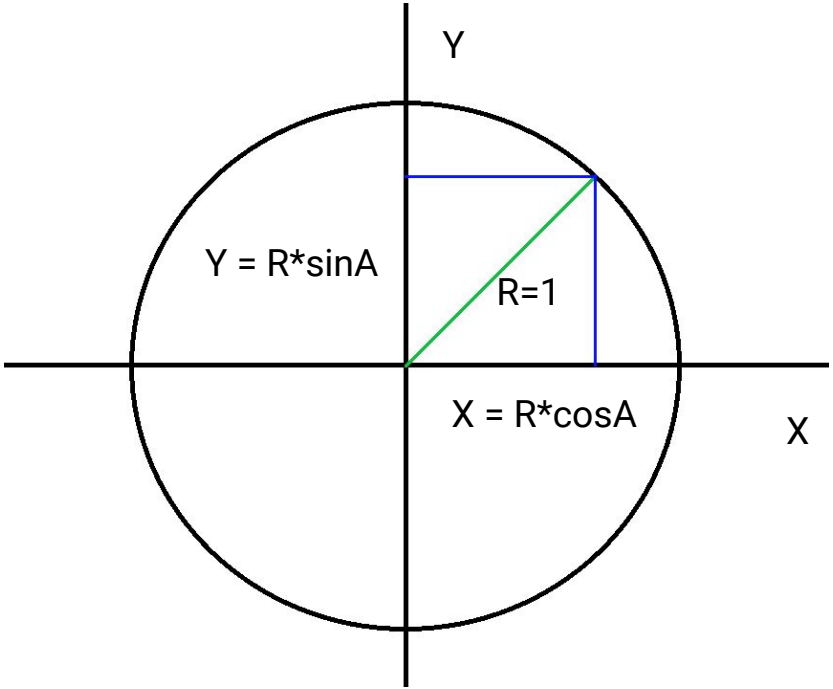
$$\text{Axis} = (x,\ y,\ z)$$
$$\text{Angle} = \theta$$
$$\text{Quaternion} = \begin{bmatrix} x * \sin\left(\frac{\theta}{2}\right) \\ y * \sin\left(\frac{\theta}{2}\right) \\ z * \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

MyShared

https://www.wolframalpha.com/input/?i2d=true&i=quaternion%3A0+-+0%5C
%2844%29707i+%2B+0%5C%2844%29707j+%2B+0%5C%2844%295k

Управление роботом

Зачем нужны кватернионы?



a +ib + jc + dk == H, где i*j*k = -1

или {w + x + y + z}

Сложение кватернионов = "смесь" вращений, т.е. мы получим вращение, которое находится между q и q'

q + q' = [x + x', y + y', z + z', w + w']

Умножение кватернионов = последовательное вращение сначала на q, потом на q'

Скалярное произведение полезно тем, что дает косинус половины угла между двумя кватернионами
q•q' = x*x' + y*y' + z*z' + w*w'

Управление роботом

Кватер-чего?

Зачем мне вообще нужны эти кватернионы? Я сейчас в углах Эйлера все посчитаю!

Пожалуйста…

$$R_Z(-\alpha)\ R_X(-\beta) = \begin{pmatrix} \cos(\alpha) & \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta) \\ -\sin(\alpha) & \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{pmatrix}$$

$$R = \begin{pmatrix} \cos(\alpha)\cos(\gamma) - \sin(\alpha)\cos(\beta)\sin(\gamma) & \cos(\alpha)\sin(\gamma) + \sin(\alpha)\cos(\beta)\cos(\gamma) & \sin(\alpha)\sin(\beta) \\ -\sin(\alpha)\cos(\gamma) - \cos(\alpha)\cos(\beta)\sin(\gamma) & -\sin(\alpha)\sin(\gamma) + \cos(\alpha)\cos(\beta)\cos(\gamma) & \cos(\alpha)\sin(\beta) \\ \sin(\beta)\sin(\gamma) & -\sin(\beta)\cos(\gamma) & \cos(\beta) \end{pmatrix}$$

Управление роботом

Зачем нужны кватернионы? Данные IMU датчика - напрямую в кватернион Фильтр Маджвика

$$q_{new} = q_0 + \frac{t}{2} * w * q_0 \qquad w = \begin{bmatrix} 0 \\ w_x \\ w_y \\ w_z \end{bmatrix}$$

Где q_new это кватернион ориентации объекта в следующий момент времени,
а q0 это начальный кватернион ориентации объекта. Следовательно, по этой формуле, мы можем в цикле суммировать угловые скорости, получаемые с инерциальных датчиков, и сразу получать кватернион ориентации, а раз для суммирования не используются сложные тригонометрические преобразования, то такие операции легко выполнять даже на не очень мощных микроконтроллерах IMU датчиков.

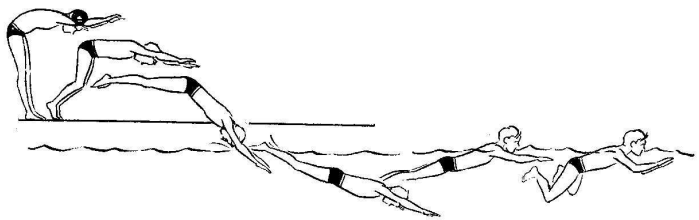
+ снижение ошибки интегрирования угловой скорости

Управление роботом

Зачем нужны кватернионы? И как с этим жить?

Общий алгоритм работы с кватернионами.

1. Если мы работаем с IMU, то по формуле из прошлого слайда мы сразу работаем с кватернионами, если мы работаем не с IMU, но предстоит большое количество вращений, а нас заставляют все это считать, то переводим углы Эйлера в кватернионы
2. Считаем повороты в кватернионах, просто перемножая их по правилам.
3. Закончили с вращениями? Переводим обратно в углы Эйлера!

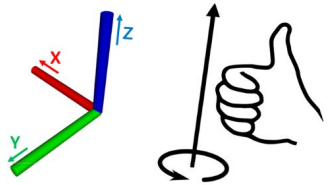


Вопрос для самопроверки:
Зачем в соревновательных бассейнах поперечная линия?

Управление роботом

Преобразование углов. Как выныривать?

Для простоты мы будем использовать только один перевод из кватерниона в угол Theta




```
def quaternion_to_theta(odom):
    return(2*math.degrees(math.asin(odom.pose.pose.orientation.z
                math.copysign(1,odom.pose.pose.orientation.w)))
```

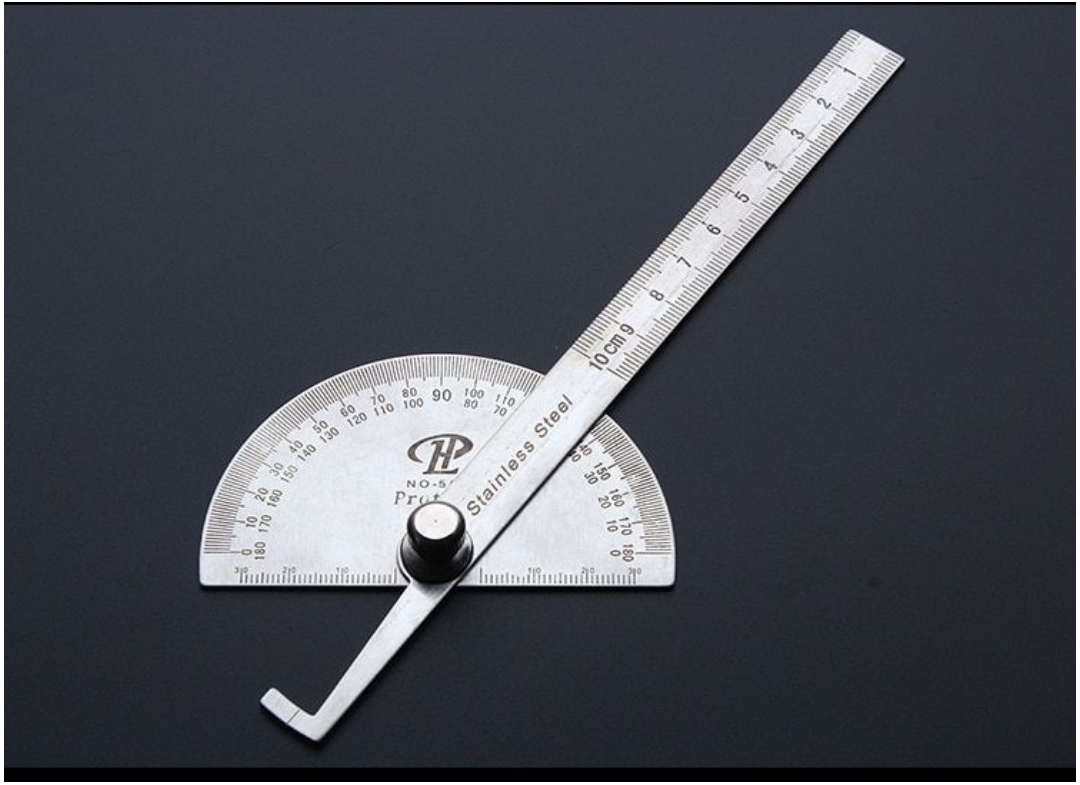
Кватернион

Axis $= (x, y, z)$

Angle $= \theta$

$$\text{Quaternion} = \begin{bmatrix} x * \sin\left(\frac{\theta}{2}\right) \\ y * \sin\left(\frac{\theta}{2}\right) \\ z * \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

Робот транспортир



Управление роботом

Робот транспортир

Общий алгоритм решения

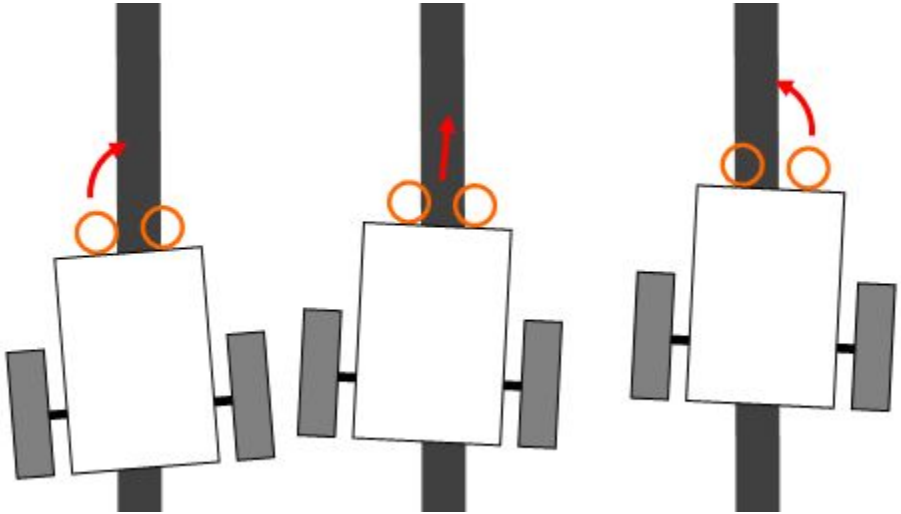
1. Подписываемся на топик /odom
2. Читаем структуру Odometry
3. Преобразуем углы из кватернионов в углы Эйлера
4. Выводим значение угла по оси Z на экран

Управление роботом

Робот транспортир

```
import rospy
import math
from nav_msgs.msg import Odometry
rospy.init_node('transportir')
def callback(odom):
     print(2*math.degrees(math.asin(odom.pose.pose.orientation.z)*
               math.copysign(1,odom.pose.pose.orientation.w)))
rospy.Subscriber("/odom", Odometry, callback)
rospy.spin()
```

“По прямой любой ценой”



Управление роботом

“По прямой любой ценой”

Общий алгоритм решения

1. Подписываемся на топик /odom
2. Читаем структуру Odometry
3. Преобразуем углы из кватернионов в углы Эйлера
4. В цикле определяем требуемое значение угловой скорости в зависимости от отклонения робота от уставки (нулевого градуса)
5. В этом же цикле публикуем рассчитанное значение угловой скорости в топик /cmd_vel

Управление роботом

По прямой любой ценой

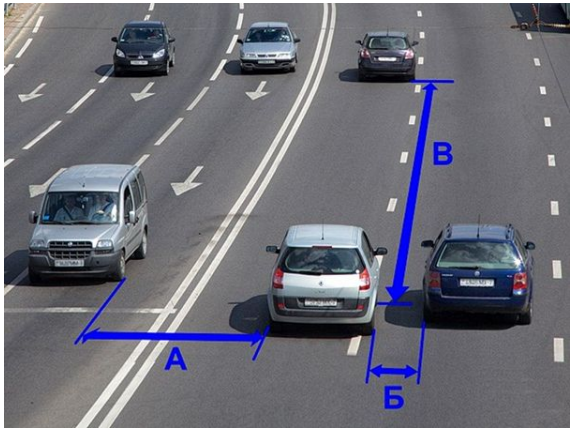
```
import rospy, math
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry

rospy.init_node('line_mover')
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=1)
pub_vel = Twist()
pub_vel.linear.x = 0.05 #будем всегда ехать вперед

def quaternion_to_theta(odom):
       return(2*(math.asin(odom.pose.pose.orientation.z)*
                     math.copysign(1,odom.pose.pose.orientation.w)))
def odomcb(odom):
    theta = quaternion_to_theta(odom)
    pub_vel.angular.z = - theta
    pub.publish(pub_vel)
rospy.Subscriber("/odom", Odometry, odomcb)
rospy.spin()
```

Практикум: Регулирование автоматизированной системы

Построение алгоритма управления

Задача

Необходимо чтобы робот постоянно ехал прямо на расстоянии 30 см от ближайшего **“справа”** препятствия. Т.е. выдерживал боковой интервал.



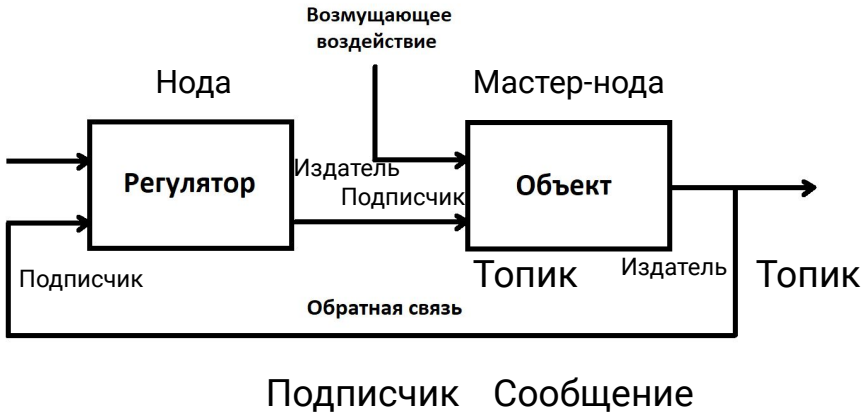
Управление роботом

Вдоль стены

```
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
rospy.init_node("turtle_mover_two")

go = Twist()
go.linear.x = 0.08
target_dist = 0.3
arr_of_err = []
Kp = 3
Kd = 10
Ki = 0.001
T_int = 16
prev_dist = 0
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=1)

def callback(msg):
    global prev_dist
    min_ranges = min(msg.ranges)
    if len(arr_of_err) < T_int:
        arr_of_err.append(target_dist -min_ranges)
    else:
        arr_of_err.pop(0)
        arr_of_err.append(target_dist -min_ranges)
    dX = min_ranges - prev_dist
    go.angular.z = Kp * (target_dist - min_ranges) - Kd * dX + Ki * sum(arr_of_err)
    pub.publish(go)
    prev_dist = min_ranges
rospy.Subscriber("scan", LaserScan, callback)
rospy.spin()
```

Регулирование автоматизированной системы

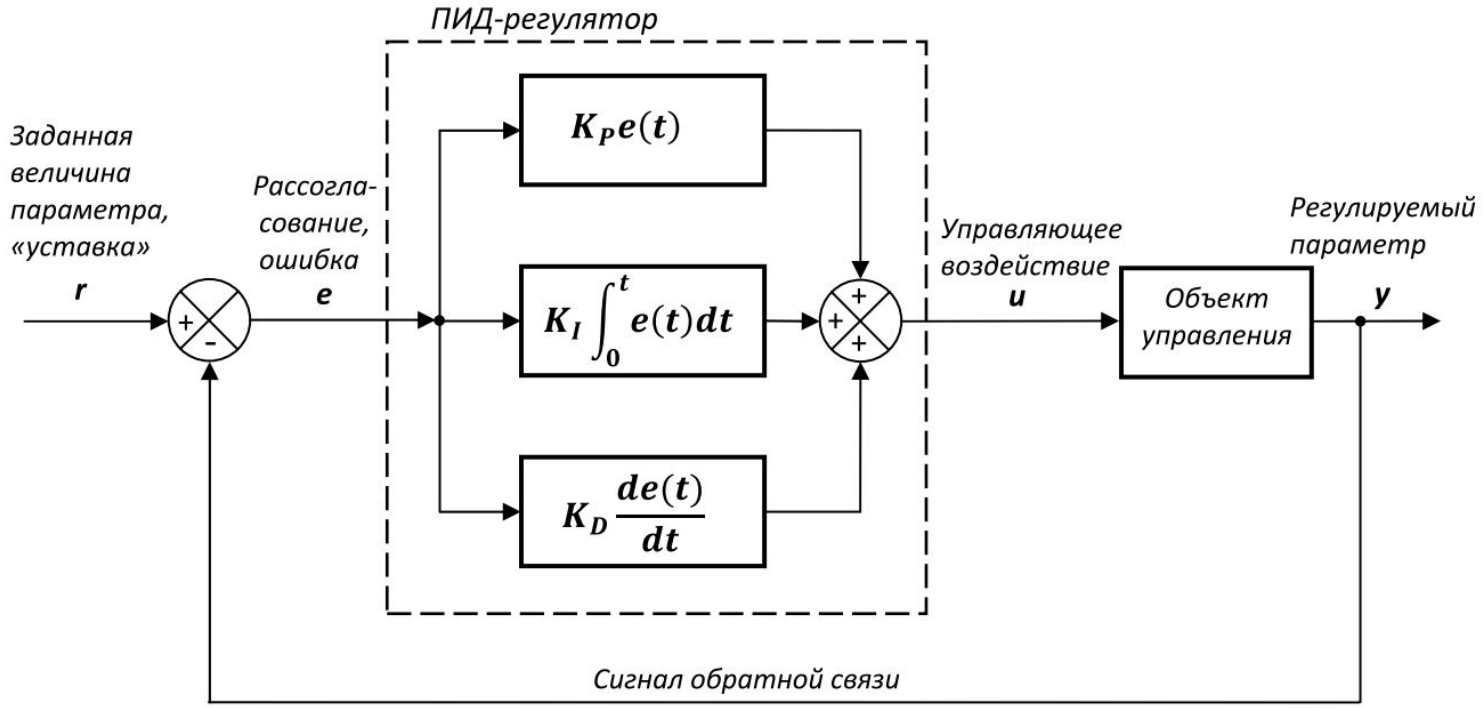
Контур обратной связи и типы регуляторов



1. R - релейный
2. P - пропорциональный

PID-регулятор

Принципиальная схема



PID-регулятор

Метод Циглера-Никольса (1942 г.)

- Для начала обнуляем все коэффициенты регулятора (пропорциональный, интегральный и дифференциальный)
- Постепенно начинаем увеличивать пропорциональный коэффициент и следим за реакцией системы. При определенном значении возникнут незатухающие колебания регулируемой величины.
- Фиксируем коэффициент K, при котором это произошло. Кроме того, замеряем период колебаний системы T
- Рассчитываем коэффициенты ПИД-регулятора:

$$K_п = 0.6 \cdot K \qquad K_д = (K_п \cdot T)/8 \qquad K_и = (2 \cdot K_п)/T$$

Метод довольно прост, но применить его можно далеко не всегда. Но тем не менее, этот метод является основным и, по большому счету, единственным широко известным. Просто подходит не всем и не всегда.

PID-регулятор

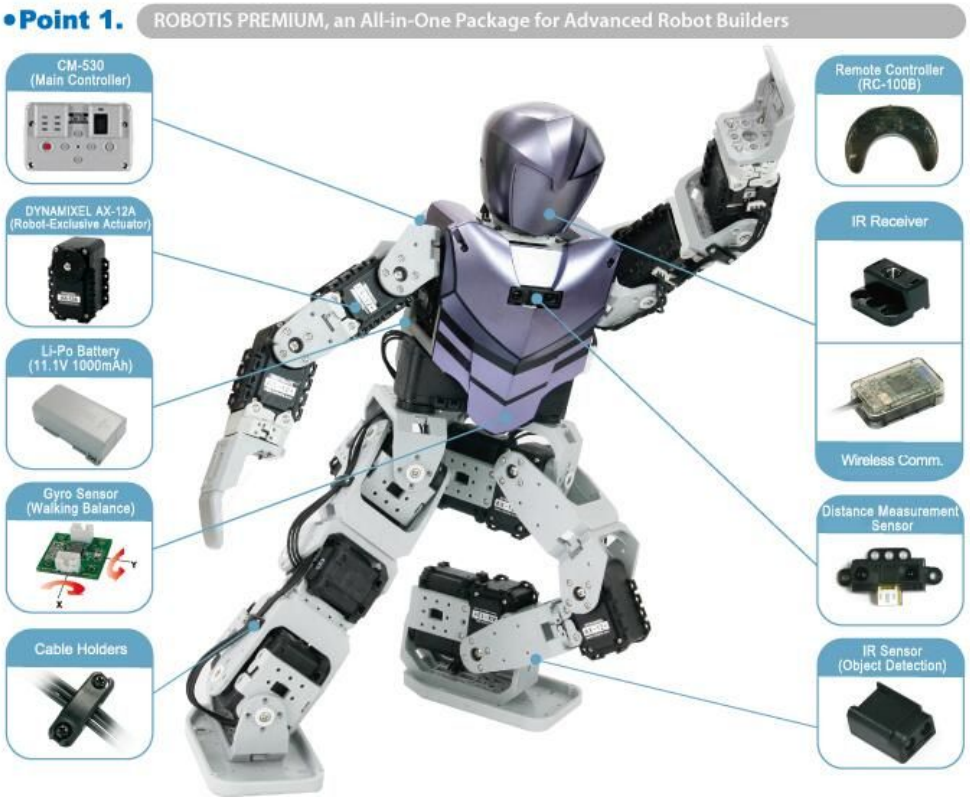
Метод Циглера-Никольса (1942 г.)

Метод Циглера – Николса

Тип управления	K_p	T_i	T_d	K_i	K_d
П	$0.5K_u$	-	-	-	-
ПИ	$0.45K_u$	$0.80T_u$	-	$0.54K_u/T_u$	-
PD	$0.8K_u$	-	$0.125T_u$	-	$0.10K_uT_u$
классический PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$
некоторый промах	$0.3\overline{3}K_u$	$0.50T_u$	$0.3\overline{3}T_u$	$0.6\overline{6}K_u/T_u$	$0.1\overline{1}K_uT_u$
нет перерегулирования	$0.20K_u$	$0.50T_u$	$0.3\overline{3}T_u$	$0.40K_u/T_u$	$0.06\overline{6}K_uT_u$

Классическая настройка PID-регулятора Циглера – Николса создает «четвертьволновое затухание». Это приемлемый результат для некоторых целей, но не для всех систем. Это правило настройки предназначено для наилучшего подавления помех в контурах ПИД-регулирования. Это приводит к агрессивному усилению и перерегулированию - некоторые системы вместо этого хотят минимизировать или устранить перерегулирование, и для них этот метод не подходит. В этом случае уравнения с пометкой «без перерегулирования» могут использоваться для вычисления соответствующих коэффициентов усиления регулятора.

Работа с периферией

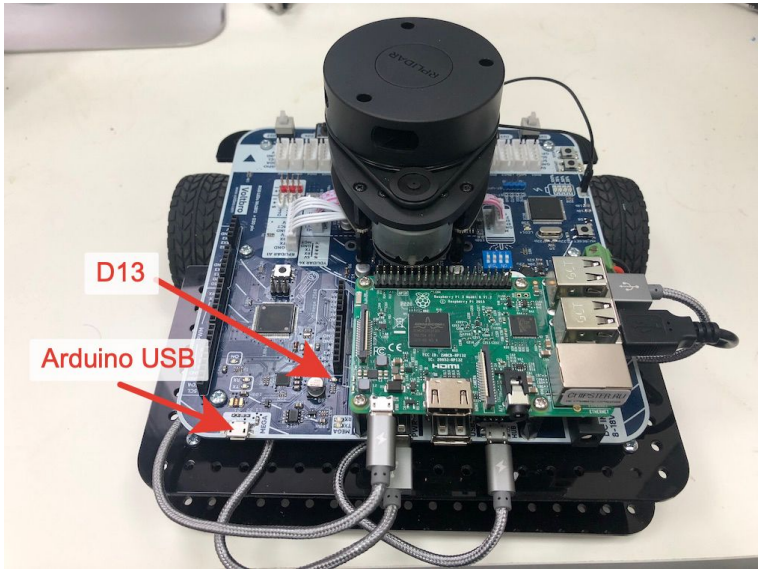


Работа с периферией

Настройки Arduino

МК Atmega 2560, функционально полностью совместимый с платами Arduino Mega. Для этого микроконтроллера вы можете самостоятельно разрабатывать скетчи, и загружать их на плату TurtleBro при помощи Arduino IDE.

Для работы с различной периферией (микроконтроллеры) в составе ROS существует специальный пакет `rosserial`. Он позволяет подключить к ядру `roscore` на Raspberry микроконтроллер через Serial порт и взаимодействовать с ним как с полноценной нодой ROS.



Для работы с Arduino через ROS необходимо установить библиотеку ros_lib <ros.h>. Для этого надо **на роботе** собрать эту библиотеку:

rosrun rosserial_arduino make_libraries.py .

Дальше надо перенести собравшуюся библиотеку с робота на компьютер и подключить ее к Arduino IDE стандартным способом в

/snap/arduino/61/libraries/

Работа с периферией

Настройки Arduino проверка - Тестовый скетч Blink.

1. Подключаем USB провод от локального компа к роботу в порт Arduino

2. В терминале локального компьютера:

```
sudo chmod 777 /dev/ttyUSB0
```

3. В Arduino IDE:

File -> Examples -> Basic -> Blink

Board: Atmega 2560
Port: USB0

Upload ->

Работа с периферией

Настройки Arduino

Для подключения к ROS со стороны Arduino необходимо инициализировать библиотеку ros_lib <ros.h> отвечающую за коммуникацию между Arduino и ROS, указав параметры Serial1 и скорость 115200.

```
#include <ros.h>
class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};
ros::NodeHandle_<NewHardware>  nh;
```

Работа с периферией

Arduino Издатель

```
#include <ros.h>
#include <std_msgs/String.h>
class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};

ros::NodeHandle <NewHardware>  nh;
std_msgs::String str_msg;
ros::Publisher pub("welcome_topic", &str_msg);
```

Работа с периферией

Arduino Издатель

```
void setup()
{
  nh.initNode();
  nh.advertise(pub);
}

void loop()
{
  str_msg.data = "Hello, robot!!!";
  pub.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
};
```

Работа с периферией

Arduino Подписчик

```
#include <ros.h>
#include <std_msgs/Empty.h>

class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};
ros::NodeHandle_<NewHardware>  nh;

void messageCb( const std_msgs::Empty& toggle_msg){
  digitalWrite(13, HIGH-digitalRead(13));   // blink the led
}

ros::Subscriber<std_msgs::Empty> sub("toggle_led", &messageCb );
```

Работа с периферией

Arduino Подписчик

```
void setup()
{
  pinMode(13, OUTPUT);
  nh.initNode();
  nh.subscribe(sub);
}

void loop()
{
  nh.spinOnce();
  delay(1);
}
```

Работа с периферией

Arduino Типовые задачи

```
#include <ros.h>
#include <std_msgs/Int16.h>
class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};
ros::NodeHandle_<NewHardware>  nh;
std_msgs::Int16 button_pressed;
ros::Publisher pub("/stop_topic", &button_pressed);
int inPin = 24;
void setup() {
   nh.initNode();
   nh.advertise(pub);
   pinMode(inPin, INPUT);
}
void loop() {
   button_pressed.data = digitalRead(inPin);
   pub.publish(&button_pressed);
   nh.spinOnce();
delay(100);}
```

Останавливать робота при нажатии на кнопку

Код для ардуино

Работа с периферией

Arduino Типовые задачи

Останавливать робота при нажатии на кнопку

```
import rospy
from std_msgs.msg import Int16
from geometry_msgs.msg import Twist

rospy.init_node("stoper_node")

pub = rospy.Publisher("/cmd_vel", Twist, queue_size = 10)

def stopcb(msg)
    pub.publish(Twist())
rospy.Subscriber("/stop_topic", Int16, stopcb)

rospy.spin()
```

Код для python

Работа с периферией

Arduino Типовые задачи

```
#include <Servo.h>
#include <ros.h>
#include <std_msgs/Int16.h>
class NewHardware : public ArduinoHardware{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};};
ros::NodeHandle_<NewHardware>  nh;
Servo servo44;
void messageCb( const std_msgs::Int16& msg){
  servo44.write(msg.data);}

ros::Subscriber<std_msgs::Int16> subServo("/servo_cmd_echo", &messageCb );
void setup(){
  nh.initNode();
  nh.subscribe(subServo);
  servo44.attach(44);servo44.write(45);}

void loop(){
  nh.spinOnce();
  delay(10);}
```

Крутить сервиком, задавая угол поворота через РОС

Код для Ардуино

Работа с периферией

Arduino Типовые задачи

Крутить сервиком, задавая угол поворота через POC

```
import rospy
from std_msgs.msg import Int16

rospy.init_node("servo_node")
pub = rospy.publisher("/servo_cmd_echo", Int16,
queue_size = 10)
rospy.sleep(1)
angle = Int16
angle.data = 90

pub.publish(angle)
```

Код для Python

Работа с периферией

Arduino Типовые задачи - Управление двумя серво через состояния

```
#include <Servo.h>
#include <ros.h>
#include <std_msgs/Int16.h>

class NewHardware : public ArduinoHardware{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};};
ros::NodeHandle_<NewHardware>  nh;

Servo Richag; //Richag
Servo Zatvor; //Zatvor

void messageCb( const std_msgs::Int16& msg)
{
  if (msg.data == 1)
```

Работа с периферией

Arduino Типовые задачи - Парктроник

```
import rospy
from sensor_msgs.msg import LaserScan
from std_msgs.msg import ByteMultiArray
class LedBlink():
    def __init__(self): #init ROS node
        self.pub = rospy.Publisher("/toggle_led", ByteMultiArray, queue_size=1)
        rospy.init_node('turtledblink')
        rospy.loginfo("Hello from PUB node")
        self.number_of_LEDs = 24
        self.minrange = 0.2 #distance in m - Full Red
        self.midrange = 0.8 #Full yellow
        self.maxrange = 1 #Full green
        self.minimum_of_each_sector_of_laser_scan_array = []
        self.filled_LED_RGB_array = [0 for i in range(self.number_of_LEDs * 3)]
        rospy.Subscriber('/scan', LaserScan, self.laser_callback)
```

Работа с периферией

Arduino Типовые задачи - Парктроник

```
def laser_callback(self, msg):
    self.minimum_of_each_sector_of_laser_scan_array= []
    for i in range(self.number_of_LEDs):
        self.minimum_of_each_sector_of_laser_scan_array.append(
                min(msg.ranges[i*(len(msg.ranges)/self.number_of_LEDs)
                 :(i + 1)*(len(msg.ranges)/self.number_of_LEDs)]))
```

Работа с периферией

Arduino Типовые задачи - Парктроник

```
def color_definition(self, array_of_minimums):
    array_of_colors_for_led = []
    for k in range(len(array_of_minimums)):
            if 0 <= array_of_minimums[k] < self.minrange:
                array_of_colors_for_led.append(127)
                array_of_colors_for_led.append(0)
                array_of_colors_for_led.append(0)
            elif self.minrange <= array_of_minimums[k] <self.midrange:
                array_of_colors_for_led.append(127)
                array_of_colors_for_led.append(int(round(
                    (array_of_minimums[k]-self.minrange)/self.midrange*127)))
                array_of_colors_for_led.append(0)
            elif self.minrange <= array_of_minimums[k] <self.maxrange:
                array_of_colors_for_led.append(int(round(
                    (self.maxrange-array_of_minimums[k])/
                    (self.maxrange-self.midrange)*127)))
                array_of_colors_for_led.append(127)
                array_of_colors_for_led.append(0)
```

Работа с периферией

Arduino Типовые задачи

```
            else:
                array_of_colors_for_led.append(0)
                array_of_colors_for_led.append(127)
                array_of_colors_for_led.append(0)
    return array_of_colors_for_led

def LED_publisher(self,array_of_leds_colors):
    leds_colors = ByteMultiArray()
    leds_colors.data = array_of_leds_colors
    self.pub.publish(leds_colors)

def controller(self):
    self.LED_publisher(
        self.color_definition(self.minimum_of_each_sector_of_laser_scan_array))
```

Работа с периферией

Arduino Типовые задачи

```
if __name__ == '__main__':

    l = LedBlink() #class invoker
    while not rospy.is_shutdown():
        l.controller()
        rospy.sleep(0.1)
```

Работа с периферией

Arduino Типовые задачи

```
#include <ros.h>
#include <FastLED.h>

#include "std_msgs/ByteMultiArray.h"

#define DATA_PIN 30
#define NUM_LEDS 24
#define BRIGHTNESS 200

CRGB leds[NUM_LEDS];
```

Парктроник

Arduino

Работа с периферией

Arduino Типовые задачи

```
class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};
void messageCb(const std_msgs::ByteMultiArray& arrscan)
{
  int pincolorred;
  int pincologreen;
  int pincolorblue;
  int i = 0;
```

Парктроник

Arduino

Работа с периферией

Arduino Типовые задачи

```
 for(i=0;i<24;i++)
 {
  pincolorred = arrscan.data[i*3];
  pincolorgreen = arrscan.data[i*3+1];
  pincolorblue = arrscan.data[i*3+2];
  leds[23-i].r = pincolorred*2;
  leds[23-i].g = pincolorgreen*2;
  leds[23-i].b = pincolorblue*2;
  FastLED.show();
 }
}
ros::NodeHandle_<NewHardware>  nh;
ros::Subscriber<std_msgs::ByteMultiArray>sub("toggle_led", &messageCb);
```

Работа с периферией

Arduino Типовые задачи

```
void setup()
{
  delay(3000); // sanity delay
  FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);
  FastLED.setBrightness( BRIGHTNESS );
  nh.initNode();
  nh.subscribe(sub);
}

void loop()
{
  nh.spinOnce();
  delay(1);
}
```

Парктроник

Arduino

Телеуправление

Телеуправление

Веб интерфейс

На роботе запущено небольшое веб приложение, которое позволяет управлять роботом прямо из браузера. Откройте в браузере адрес http://ip_робота:8080

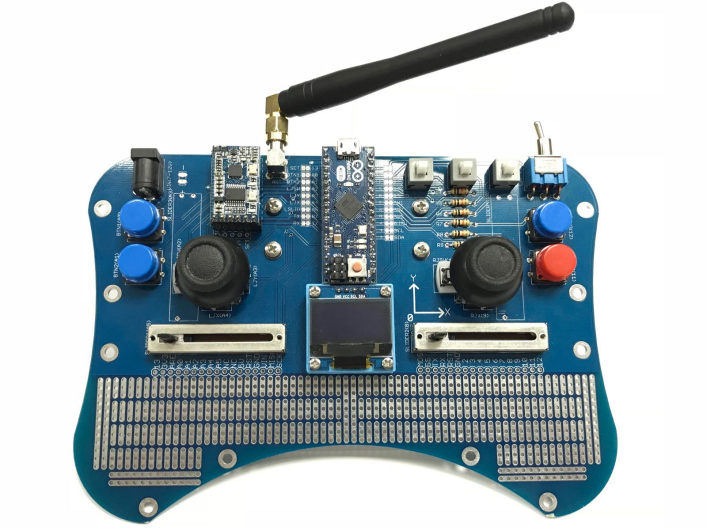
Управление роботом выполняется клавишами AWSD

В настройках веб приложения также видна основная телеметрия робота, и есть возможность менять скорости передвижения робота.

Телеуправление

Введение

Телеуправление (ТУ) — управление положением или состоянием дискретных объектов и объектов с непрерывным множеством состояний методами и средствами телемеханики. **Телеуправление** используется в различных технических устройствах. Одним из первых устройств, использующих технологию **телеуправления**, является телеграф. Сейчас самым распространенным средством **телеуправления** можно считать пульт дистанционного управления.



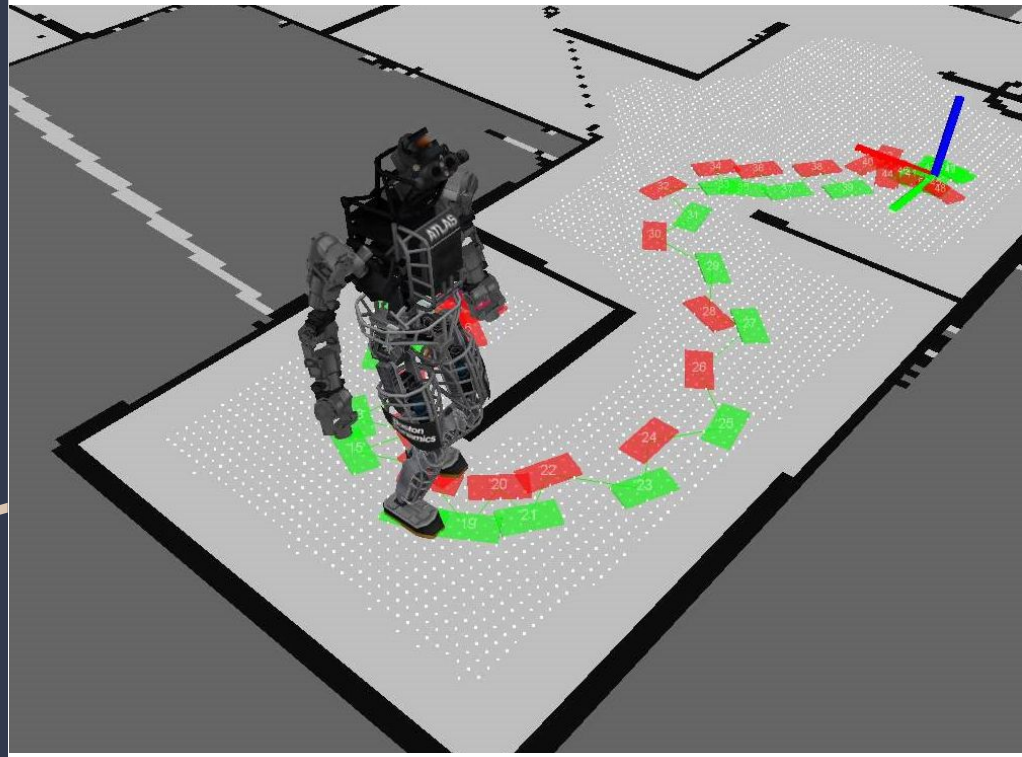
Телеуправление

Пакет JoyBro

https://github.com/voltbro/joybro

- Установить пакет на компьютер
- Установить пакет на робота
- Залить на джойстик прошивку
- Подключить джойстик к компьютеру, и выдать к устройству джойстика права доступа
- Настроить ROS_MASTER_URI и ROS_HOSTNAME на того робота которым мы хотим управлять
- Запустить ноду джойстика

Автономная навигация



Автономная навигация

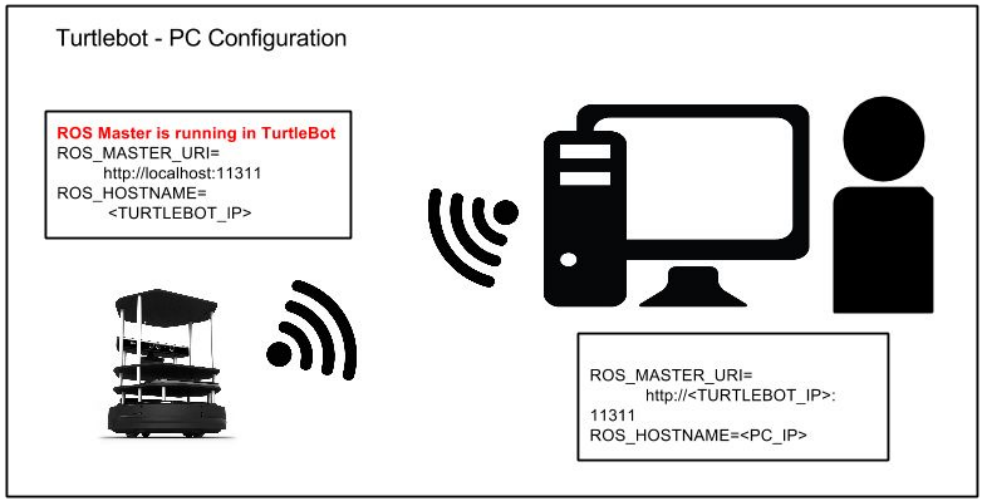
Настройки на ноутбуке

```
export ROS_MASTER_URI='http://192.168.0.145:11311'
```

РОБОТ

```
export ROS_HOSTNAME=192.168.0.111
```

НОУТБУК



Автономная навигация

Запуск SLAM

На компьютере:

1. **Сделать все экспорты с указанием на робота:**

```
export ROS_MASTER_URI='http://192.168.0.145:11311'
export ROS_HOSTNAME=192.168.0.111
```

2. **Запустить rviz и убедиться что мы видим данные с робота**
3. **Настроить rviz как на слайде ранее**

На роботе:

roslaunch turtlebro_navigation turtlebro_slam_navigation.launch

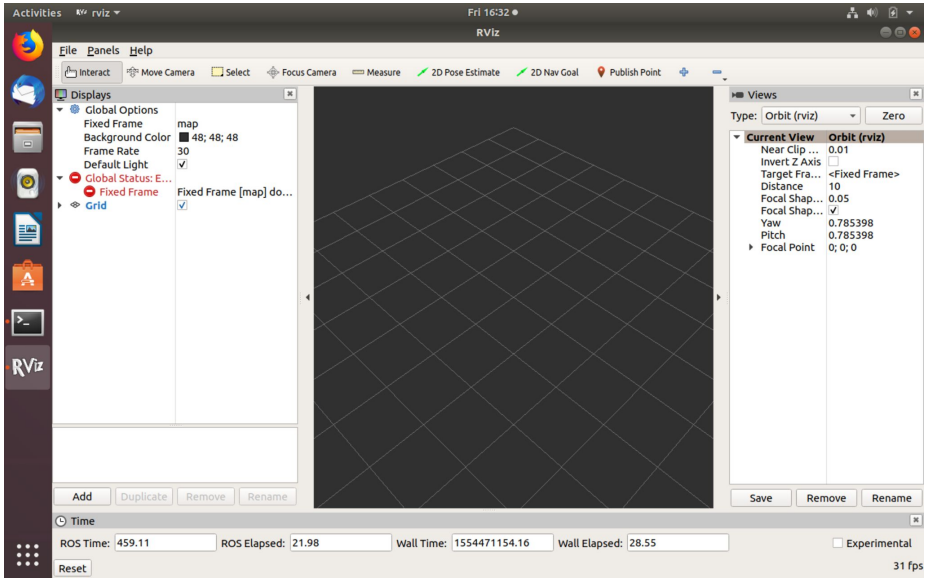
Автономная навигация

Датчики и сенсоры

В составе ROS, есть специальная графическая программа rviz, которая служит для визуализации информации из различных систем ROS. В том числе, она может показать данные, которые передает лидар.

Запустим программу из консоли:

rviz



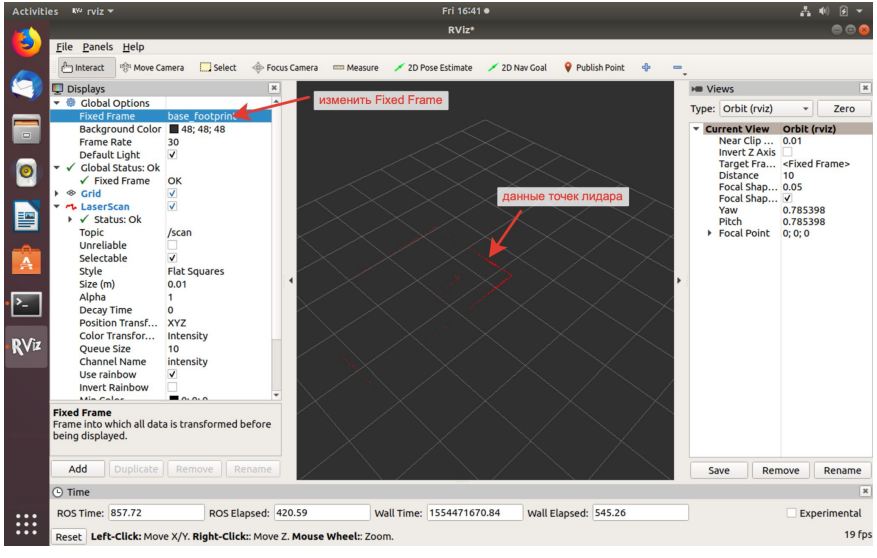
Автономная навигация

Датчики и сенсоры

Давайте посмотрим на эти данные в "человеческом" виде в программе rviz

Нажмем Add, выберем вкладку By topic, выбрать источник данных /scan/LaserScan

Изменить Fixed Frame на вкладке Global Option указав точку отсчета - базу робота base_footprint



Автономная навигация

Настройка отображения SLAM навигации

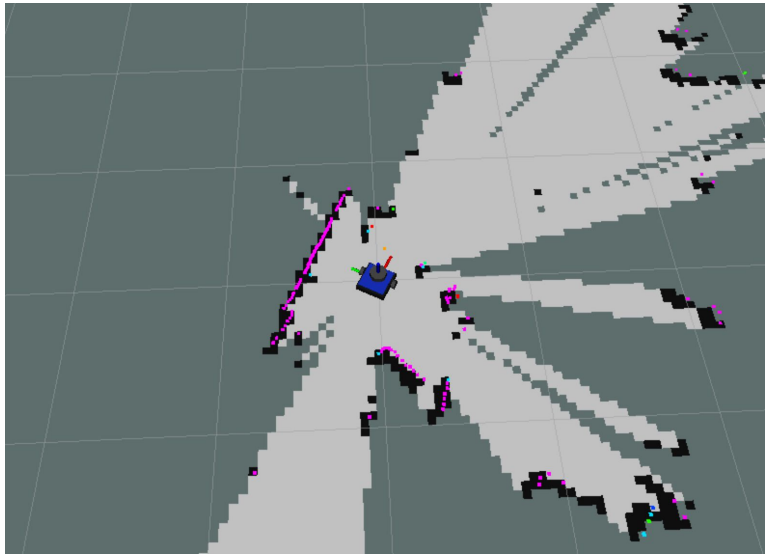
В rviz добавим отображение карты

Add->By Topic->/map->Map

А также различные другие отображения навигационных объектов:

Например локальное окно, проложенный глобальный и локальный путь и др.

Add->By Topic->



Автономная навигация

Работа с картой

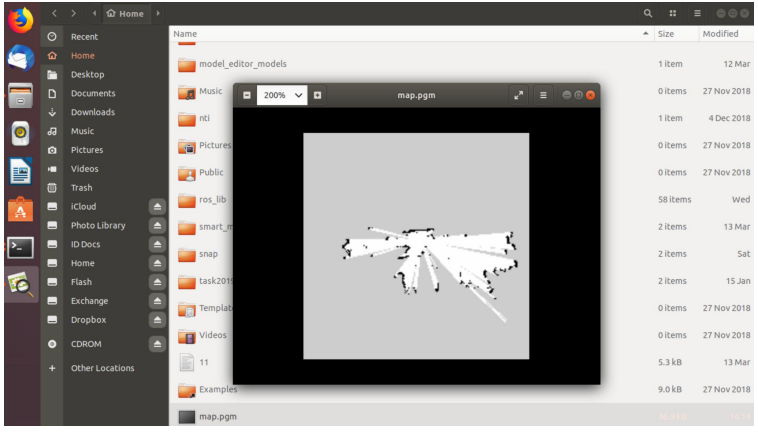
Сохранение карты

На роботе:

rosrun map_server map_saver -f map map - имя файла

После выполнения данной команды, будет создано два файла map.yaml и map.pgm

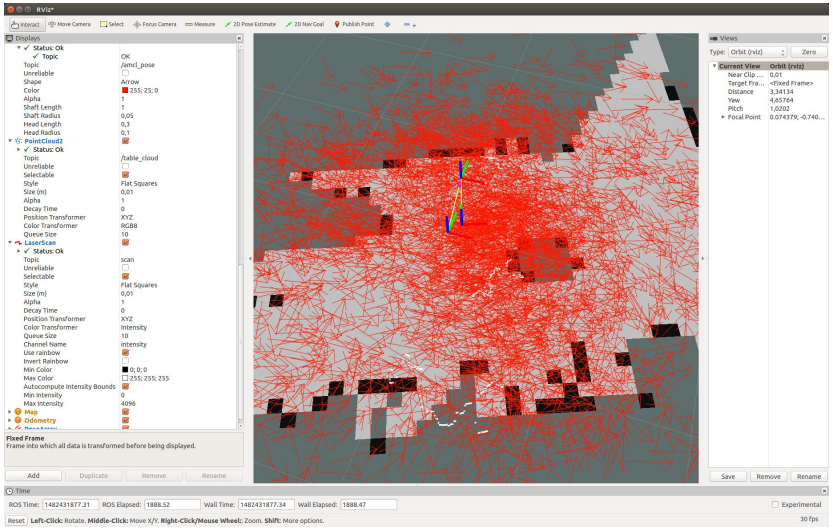
В файле map.yaml находятся параметры карты. В map.pgm - собственно изображение карты

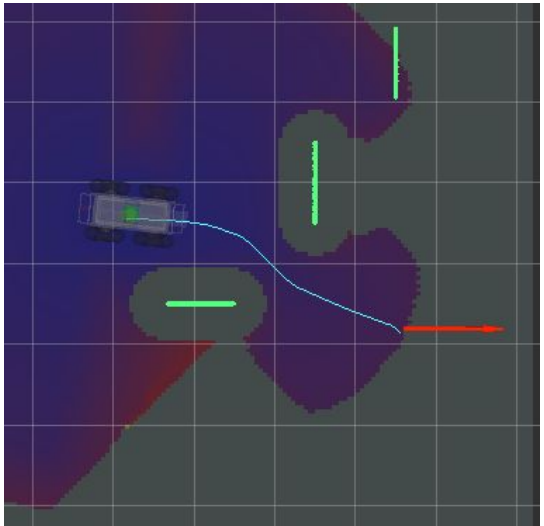


Автономная навигация

Введение

Навигация робота это комплексный подход, позволяющий роботу самостоятельно и автономно перемещаться из одного места в другое, избегая столкновения с препятствиями, на основании данных от **датчиков робота(внешних датчиков)**. При этом цель перемещения может быть выбрана как человеком, так и самим роботом. Для осуществления навигации роботам требуется следующее: **Карта, Локализация, Планирование Пути и Движение по маршруту**



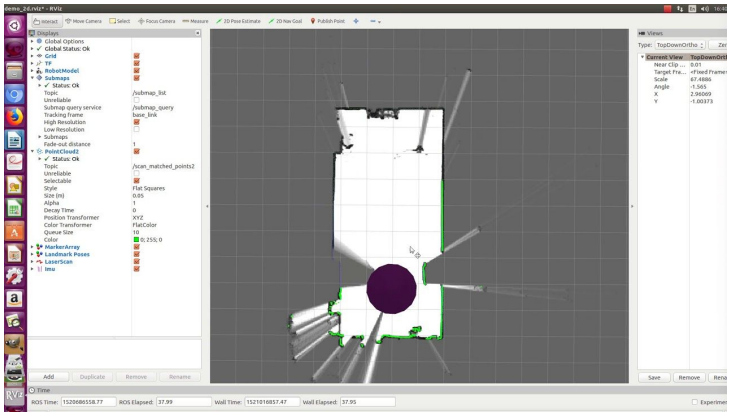


Автономная навигация

Построение карты

В рамках ROS, при работе с картой наиболее широкое распространение получили следующие подходы:

- GMapping - лидар
- Cartographer - лидар + GPS + RGBD + стерео
- Rtabmap - стерео + RGBD

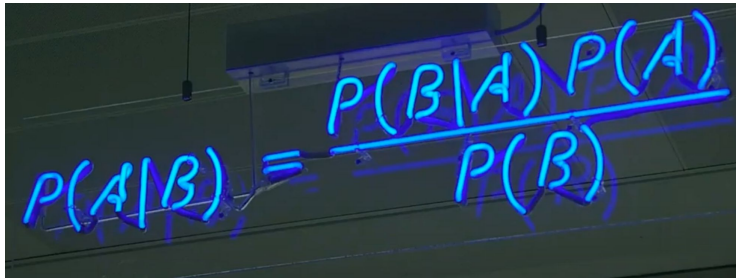


Автономная навигация

Локализация

Если у нас есть карта, то следующей необходимой информацией, будет “А где робот на этой карте находится?” - Ответ на этот вопрос дает локализация

Решения этой задачи возможно при помощи различных вероятностных подходов, в основе большинства из них лежит теорема Байеса.

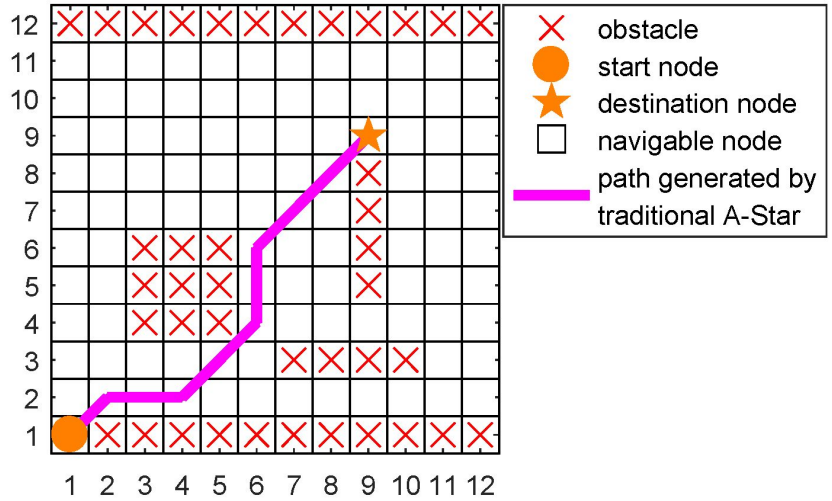


К таким подходам относятся Различные типы фильтра Калмана, фильтры частиц, Rao-Blackwellised SLAM, и другие SLAM подходы.

Автономная навигация

Планирование пути

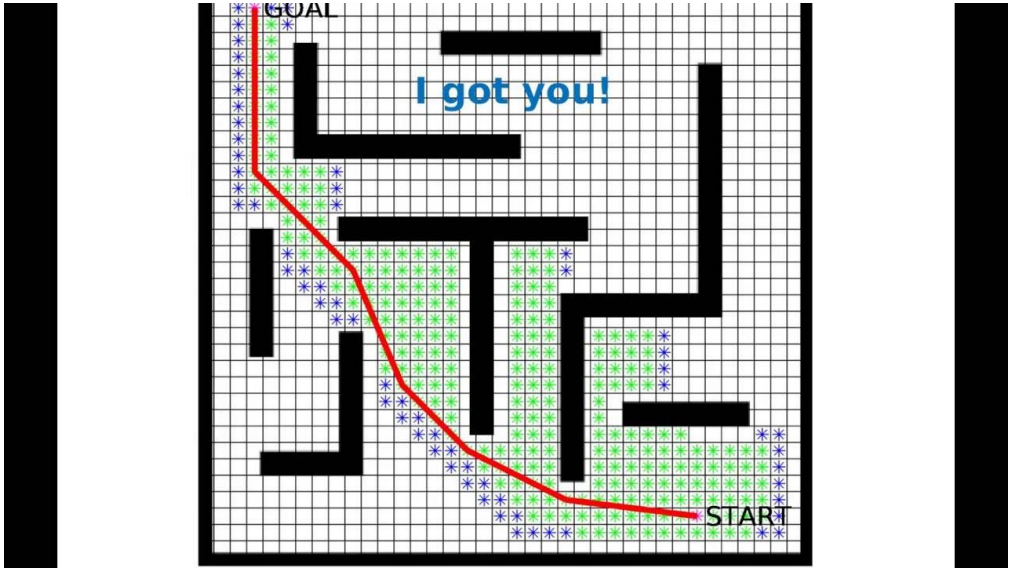
Ок, у нас есть карта и мы знаем где мы находимся, теперь нам надо спланировать как нам доехать до цели.



Автономная навигация

Планирование пути

Используемые алгоритмы: Поиск в ширину, алгоритм Дейкстры, A*, D* - тысячи их



Автономная навигация

Передвижение по маршруту

Локализация - наблюдение

Локальный планировщик - оценка ситуации

Планирование движений (motion planner) - решение

Команды на движение - действие

Автономная навигация

Задание навигационной цели для робота через python

```python
import rospy
import actionlib
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

client = actionlib.SimpleActionClient('move_base',MoveBaseAction)
client.wait_for_server()

goal = MoveBaseGoal()
goal.target_pose.header.frame_id = "map"
goal.target_pose.header.stamp = rospy.Time.now()
goal.target_pose.pose.position.x = x # Координаты в системе отсчета карты
goal.target_pose.pose.position.y = y # (0,0) - точка включения навигации
goal.target_pose.pose.orientation.w = 1.0

client.send_goal(goal)
client.wait_for_result()
```

Работа с камерой введение в компьютерное зрение

Работа с камерой

Введение в компьютерное зрение

Задача зрения: понять что находится на изображении

Задача компьютерного зрения: построение компьютерной модели того, что находится на изображении

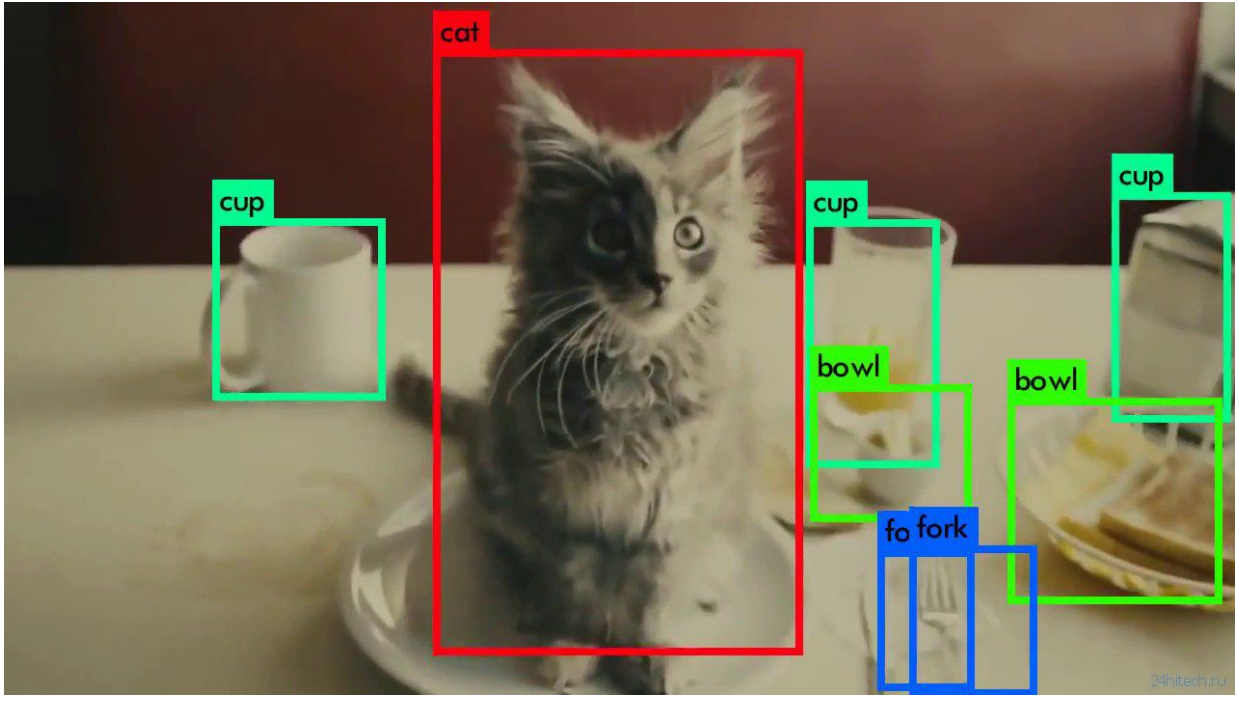
Компьютерное зрение - часть области искусственного интеллекта

Тест Тьюринга для компьютерного зрения: компьютер должен ответить на любой вопрос по изображению, на который может ответить человек

Работа с камерой

Введение в компьютерное зрение

Семантическая задача компьютерного зрения - Что находится на изображении?



Работа с камерой

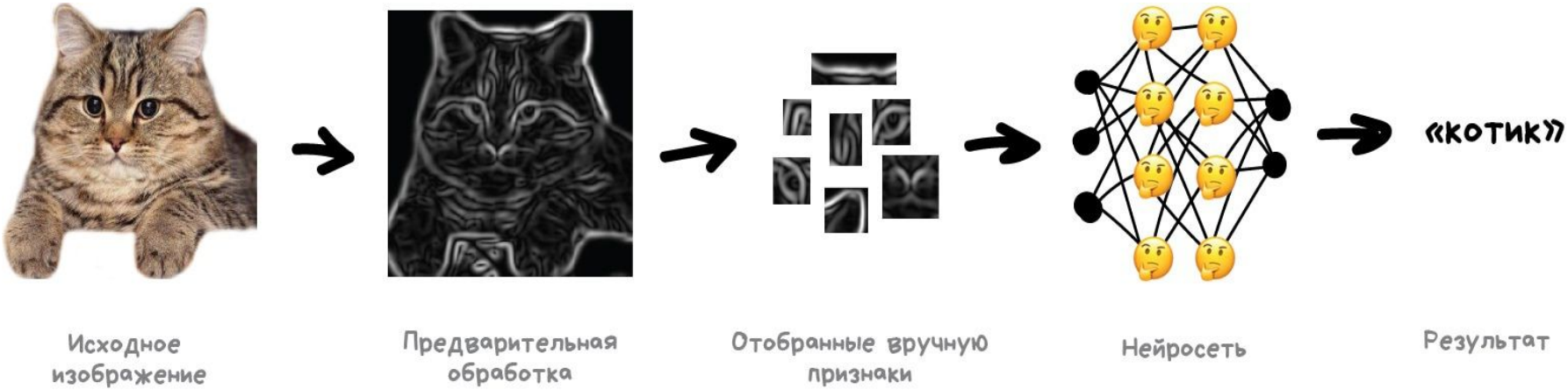
Введение в компьютерное зрение

Метрическая задача компьютерного зрения - какого размера объекты на изображении?

Работа с камерой

Введение в компьютерное зрение

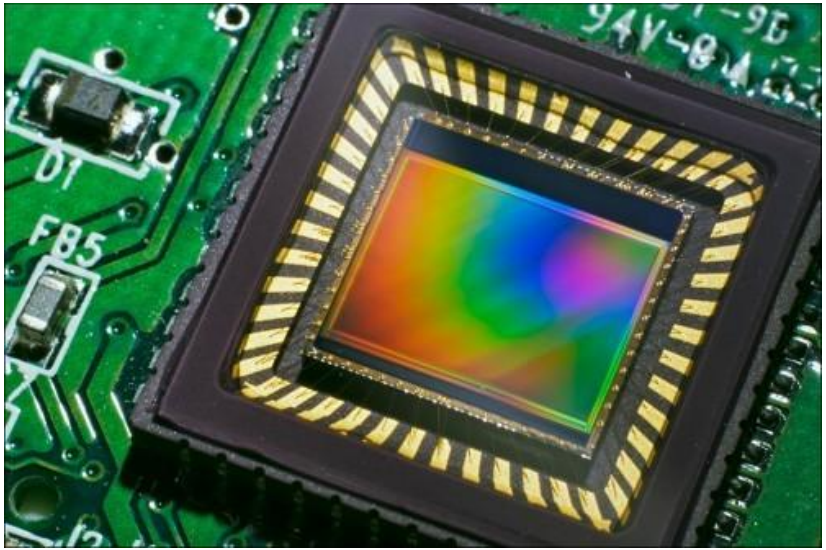
Этапы решения задач компьютерного зрения



Работа с камерой

Что необходимо сделать чтобы компьютер мог решать эти две задачи

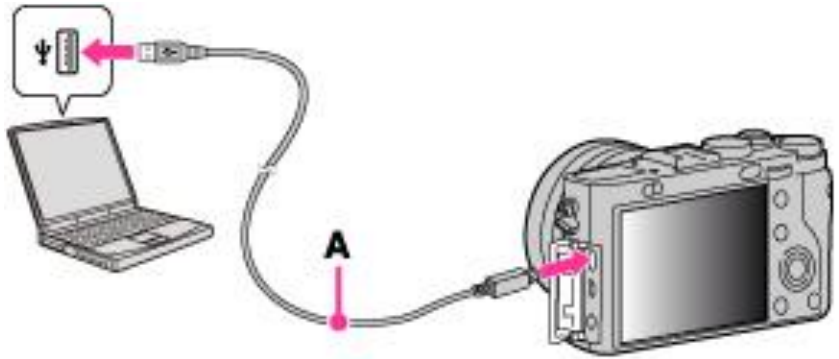
Получение изображения (линзы, диафрагма, матрица)



Работа с камерой

Что необходимо сделать чтобы компьютер мог решать эти две задачи

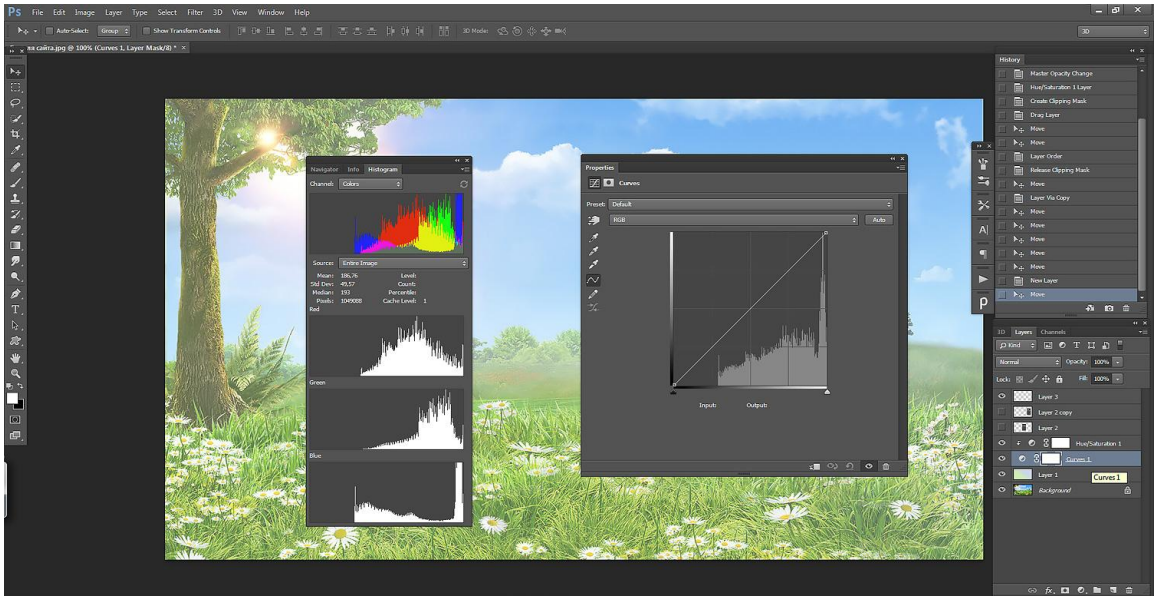
Дискретизация полученного изображения (фильтры, призмы, порты, драйвера, носители информации: диски - память)



Работа с камерой

Что необходимо сделать чтобы компьютер мог решать эти две задачи

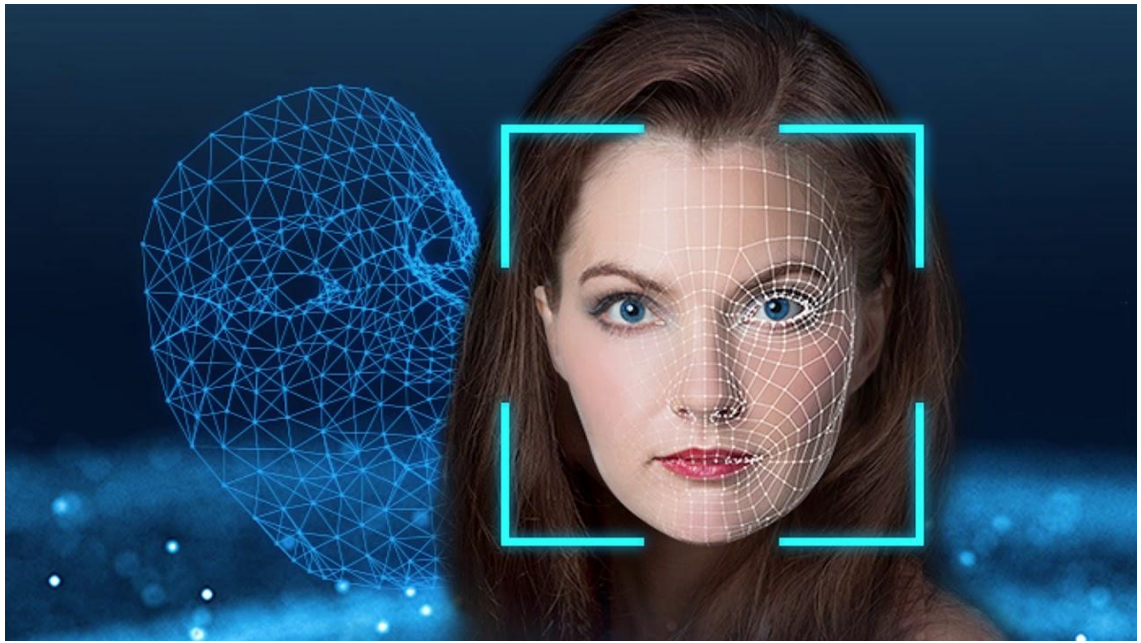
Предобработка изображения техническими алгоритмами (обрезка, цветокоррекция, коррекция искажений оптики, коррекция неравномерности освещения и т.д.)



Работа с камерой

Что необходимо сделать чтобы компьютер мог решать эти две задачи

Алгоритмическая обработка изображения



Работа с камерой

Что необходимо сделать чтобы компьютер мог решать эти две задачи

Результат применения алгоритма - решение Семантической и Метрической задач.

Работа с камерой

Пакет uvc_camera

Пакет публикует сжатые данные **sensor_msgs/CompressedImage** в топик **front_camera/compressed**

Конвертация из **sensor_msgs/Image** в формат **OpenCV** возможна через библиотеку http://wiki.ros.org/cv_bridge

Официальная документация пакета http://wiki.ros.org/uvc_camera
Данный пакет работает с камерой через библиотеку **OpenCV**

Работа с камерой

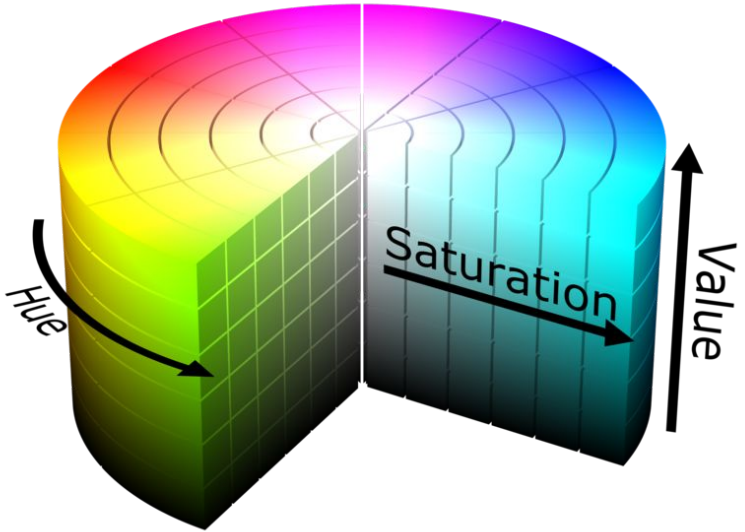
Получение изображения - Подписываемся на данные с камеры

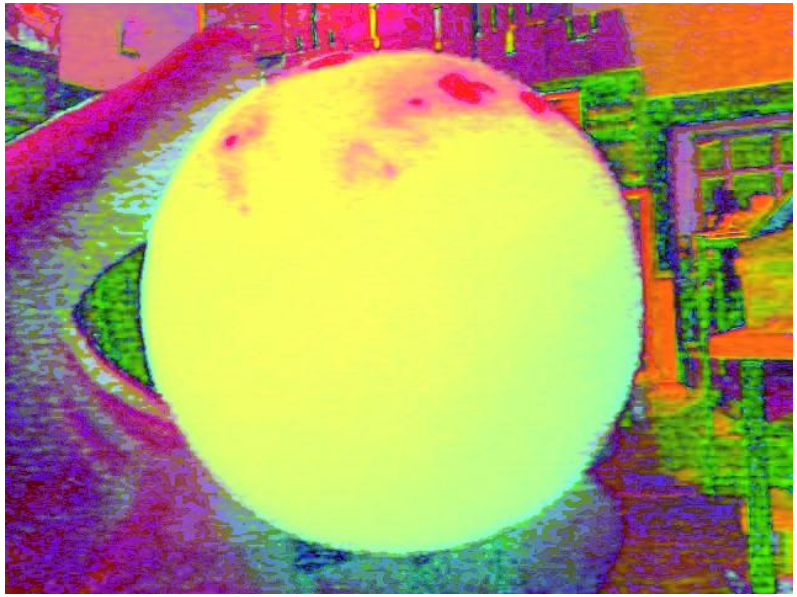
```
import rospy
import cv2
import numpy as np
import math
from sensor_msgs.msg import CompressedImage

rospy.init_node("photographer")

def cb_video_capture(image_msg):
        global image_from_ros_camera
        np_arr = np.frombuffer(image_msg.data, np.uint8)
        image_from_ros_camera = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)

rospy.Subscriber("/front_camera/image_raw/compressed", CompressedImage, cb_video_capture)
rospy.sleep(1)

cv2.imwrite("/home/pi/ball_JPG.jpg", image_from_ros_camera)
```

Работа с камерой

Предобработка изображения - Смена цветовой модели

https://ru.wikipedia.org/wiki/HSV_(цветовая_модель)





Работа с камерой

Предобработка изображения - Смена цветовой модели

Преобразование RGB → HSV

$$H \in [0,360)$$

$$S, V, R, G, B \in [0,1]$$

$$Max = \max\{R, G, B\}$$

$$Min = \min\{R, G, B\}$$

$$H = \begin{cases} 0, \text{ если } Max = Min \\ 60 \times \dfrac{G-B}{Max-Min} + 0, \text{ если } Max = R \text{ и } G \geq B \\ 60 \times \dfrac{G-B}{Max-Min} + 360, \text{ если } Max = R \text{ и } G < B \\ 60 \times \dfrac{B-R}{Max-Min} + 120, \text{ если } Max = G \\ 60 \times \dfrac{R-G}{Max-Min} + 240, \text{ если } Max = B \end{cases}$$

$$S = \begin{cases} 0, \text{ если } Max = 0 \\ 1 - \dfrac{Min}{Max}, \text{ в пр. случае} \end{cases}$$

$$V = Max$$

Работа с камерой

Предобработка изображения - Смена цветовой модели

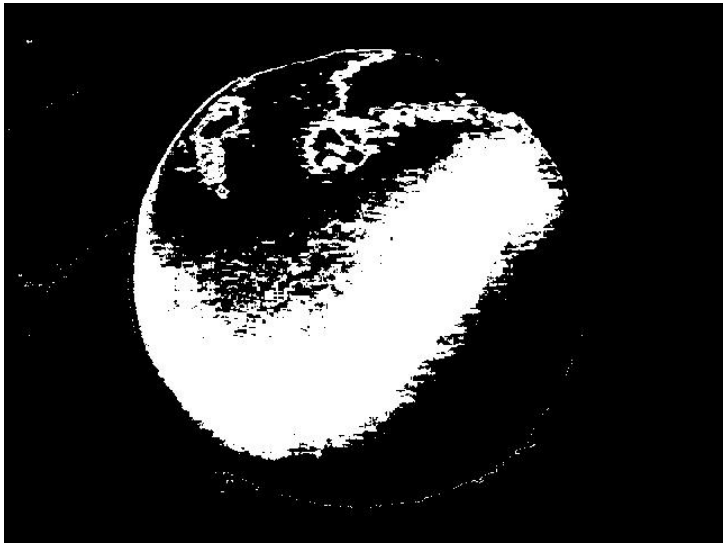
```
def RGB2HSV(im):
    for j in range(im.shape[0]):
        for i in range(im.shape[1]):
            R = float(im[j][i][0]/255)
            G = float(im[j][i][1]/255)
            B = float(im[j][i][2]/255)
            V = max(R,G,B)
            if V == 0:
                S = 0
            else:
                S = ((V - min(R,G,B)) / V)
            if R == B == G:
                H = 0
            elif V == R:
                H = (60*(G - B))/(V - min(R,G,B))
            elif V == G:
                H = (120 + 60*(B - R))/(V - min(R,G,B))
            elif V == B:
                H = (240 + 60*(R - G))/(V - min(R,G,B))
            if H < 0:
                H += 360
            im[j][i][0] = int(H/2)
            im[j][i][1] = int(S*255)
            im[j][i][2] = int(V*255)
        print(j)

    return im
```

```
rospy.sleep(1)
cv2.imwrite("/home/pi/ball_JPG.jpg",
image_from_ros_camera)
HSV = RGB2HSV(image_from_ros_camera)
cv2.imwrite("/home/pi/ball_HSV.jpg", HSV)
```

Работа с камерой

Предобработка изображения - Маска

Создадим маску изображения, т.е. закрасим черным все, что не попадает в цветовые границы



Работа с камерой

Предобработка изображения - Маска

```
def inRange(im):
    for j in range(im.shape[0]):
        print("Mask :", j)
        for i in range(im.shape[1]):
            if (im[j][i][0] < yellowLower[0] or
                im[j][i][0] > yellowUpper[0] or
                im[j][i][1] < yellowLower[1] or
                im[j][i][1] > yellowUpper[1] or
                im[j][i][2] < yellowLower[2] or
                im[j][i][2] > yellowUpper[2]):
                im[j][i][0] = 0
                im[j][i][1] = 0
                im[j][i][2] = 0
            else:
                im[j][i][0] = 255
                im[j][i][1] = 255
                im[j][i][2] = 255
    return im
```

```
rospy.sleep(1)

yellowLower = (100, 180, 80) # dark
yellowUpper = (120, 255, 255) # light


cv2.imwrite("/home/pi/ball_JPG.jpg",
image_from_ros_camera)
HSV = RGB2HSV(image_from_ros_camera)
cv2.imwrite("/home/pi/ball_HSV.jpg", HSV)
mask = inRange(HSV)
cv2.imwrite("/home/pi/ball_mask.jpg", mask)
```

Работа с камерой

Решение семантической задачи - применение алгоритмов

Моменты изображения (англ. *image moments*) в компьютерном зрении, обработке изображений и смежных областях — некоторые частные средневзвешенные (момент) интенсивностей пикселей изображения, или функция таких моментов. Как правило, выбираются моменты, имеющие полезные свойства или интерпретации.

В самом общем смысле момент функции — это некая скалярная величина, которая характеризует эту функцию и может быть использована для артикуляции ее важных свойств. С математической точки зрения набор моментов является в некотором смысле «проекцией» функции на полиномиальный базис. Он аналогичен преобразованию Фурье, которое представляет из себя проекцию функции на базис из гармонических функций[1].

Моменты изображения полезны для описания объектов после сегментации. Простые свойства изображения, которые можно найти с помощью моментов, включают в себя площадь (или суммарную интенсивность), геометрический центр и информацию об ориентации. Кроме них в математической статистике давно применяются моменты более высоких порядков, например коэффициент асимметрии и коэффициент эксцесса[1].

Работа с камерой

Решение семантической задачи - применение алгоритмов

Определим центр шарика при помощи графических моментов изображения,

```
center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
```

Графический момент это чисто математический объект. Он определяется формулой.

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Где:

i и *j* — порядки момента *M*, соответствующие координатам изображения.

I(x,y) - интенсивность пикселя

x,y - координаты пикселя

Работа с камерой

Решение семантической задачи - применение алгоритмов

Теперь по русски. Рассмотрим функции моментов изображения и посмотрим что же такое моменты, через их функции.

```
center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
```

Давайте для простоты возьмем только первый компонент центра `M["m10"] / M["m00"]`, подставим его в эту формулу и применим ее ко всем пикселям нашей маски.

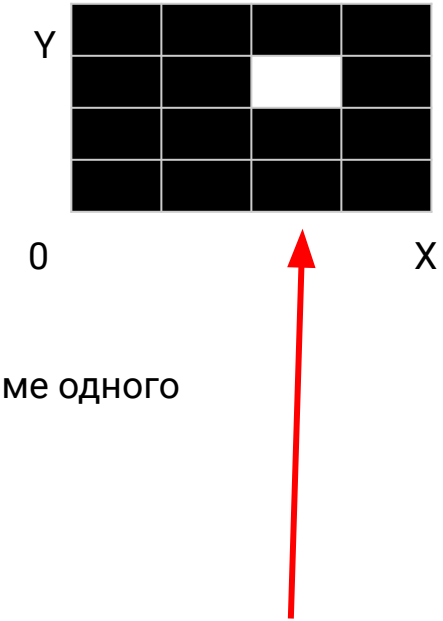
$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Тренироваться будем на тестовом изображении, разрешения 4x4 пикселя

Работа с камерой

Решение семантической задачи - применение алгоритмов

Давайте проделаем это руками на примере вот такого изображения.



Посчитаем X-компоненту `M["m10"] / M["m00"]`:
x в степени 1 - это x, y в степени 0 - это 1, I = 0 для черного и I=1 для белого пикселя.
Т.е. по формуле $$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

Получим для `M["m10"] =`
1*1*0+2*1*0+3*1*0 + 4*1*0
0 + 0 + 3*1*1+ 0
0 + 0 + 0 + 0
0 + 0 + 0 + 0 = 3

А для `M["m00"] =` везде 0, кроме одного пикселя, где 1*1*1

Таким образом `M["m10"] / M["m00"] = 3 / 1 = 3` Т.е X-компонента центра будет 3. Магия!

Работа с камерой

Решение семантической задачи - Графические моменты изображения

```
def moments(im):
    M10 = 0
    M01 = 0
    M00 = 0
    for j in range(im.shape[0]):
        print("Moment :", j)
        for i in range(im.shape[1]):
            if im[j][i][0] == 255:
                M10 += i
                M00 += 1
                M01 += j
    return (M10/M00, M01/M00, M00)
```

```
rospy.sleep(1)
cv2.imwrite("/home/pi/ball_JPG.jpg",
image_from_ros_camera)
HSV = RGB2HSV(image_from_ros_camera)
cv2.imwrite("/home/pi/ball_HSV.jpg", HSV)
mask = inRange(HSV)
cv2.imwrite("/home/pi/ball_mask.jpg", mask)
print(moments(mask))
```

Работа с камерой

Решение семантической задачи - Описание окружности вокруг шарика

```
rospy.sleep(1)
cv2.imwrite("/home/pi/ball_JPG.jpg", image_from_ros_camera)
HSV = RGB2HSV(image_from_ros_camera)
cv2.imwrite("/home/pi/ball_HSV.jpg", HSV)
mask = inRange(HSV)
cv2.imwrite("/home/pi/ball_mask.jpg", mask)
x, y, S = moments(mask)
R = math.sqrt(S/math.pi)
mask_with_circle = cv2.circle(mask, (int(x), int(y)), int(R), (255, 0, 0), 2)
cv2.imwrite("/home/pi/ball_mask_circle.jpg", mask_with_circle)
```

Работа с камерой

Давай “по-быстрому”, а? Или зачем нужны библиотеки и C++

```
import rospy
import cv2
import numpy as np
import math
from sensor_msgs.msg import CompressedImage

rospy.init_node("photographer")
yellowLower = (100, 180, 80) # dark
yellowUpper = (120, 255, 255) # light

def cb_video_capture(image_msg):
        global image_from_ros_camera
        np_arr = np.frombuffer(image_msg.data, np.uint8)
        image_from_ros_camera = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)

rospy.Subscriber("/front_camera/image_raw/compressed", CompressedImage, cb_video_capture)
```

Работа с камерой

Давай “по-быстрому”, а? Или зачем нужны библиотеки и C++

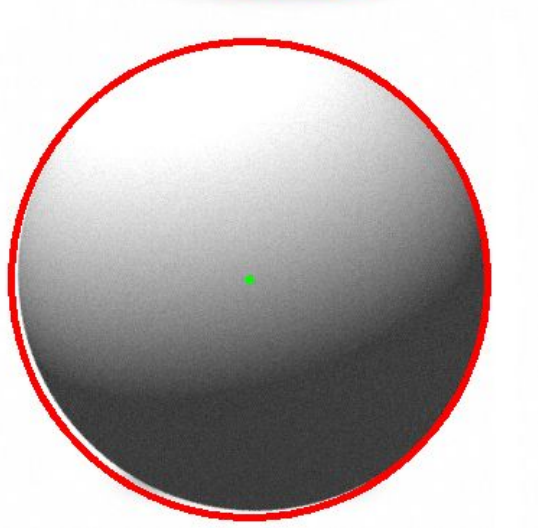
Сделаем в несколько строк чуть больше чем за всю предыдущую программу.

```
def process():
    hsv = cv2.cvtColor(image_from_ros_camera, cv2.COLOR_RGB2HSV)
    mask = cv2.inRange(hsv, yellowLower, yellowUpper)
    #mask = cv2.erode(mask, None, iterations=2)
    #mask = cv2.dilate(mask, None, iterations=2)
    Moments = cv2.moments(mask)
    x = int(Moments["m10"] / Moments["m00"])
    y = int(Moments["m01"] / Moments["m00"])
    print(x,y, Moments["m00"])
while not rospy.is_shutdown():
    rospy.sleep(0.3)
    process()
```

Работа с камерой

Семантический анализ предобработанного изображения

1. Найдем контуры на изображении `cnts = cv2.findContours(mask.copy())`
2. Воспользуемся функцией `((x, y), radius) = cv2.minEnclosingCircle(c)` и опишем окружность минимального радиуса вокруг нашего контура.



Работа с камерой

Решение семантической задачи

```
        cnts = cv2.findContours(mask.copy(),
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]

        self.current_data = [0,0,0]
        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            if radius > 10:
                self.current_data = [x,y,radius]
        return self.current_data
```

Работа с камерой

Управление роботом по данным решения семантической задачи

```
    def go_to_ball(self, data):
        self.vel.angular.z = -self.ANGULAR_SPEED*((data[0]-320)/320)
        if data[2] < self.DES_BALL_SIZE:
            self.vel.linear.x = self.LIN_SPEED * (2 - (data[2] / self.DES_BALL_SIZE))
        else:
            self.vel.linear.x = 0
            self.ball_reached = True
        self.pub.publish(self.vel)

    def search_the_ball(self):
        data = self.process()
        self.go_to_ball(data)
rospy.init_node("ball_searcher")
r = RoboTurner()
while not rospy.is_shutdown():
    r.search_the_ball()
    rospy.sleep(0.04)
```

Практикум: Работа с удаленным роботом

Практикум: Работа с удаленным роботом

Настройка VPN подключения

Для доступа к роботам удаленного полигона используется OpenVPN

sudo apt install openvpn

Администратор полигона предоставит вам ключ. При помощи этого ключа и OpenVPN, надо подключиться к полигону и не закрывать окно терминала с OpenVPN до окончания работы с удаленным полигоном

sudo openvpn --config <путь до файла с ключем>

Тест подключения

Initialization Sequence Completed
ping 10.8.0.1
ping 10.8.0.6

Практикум: Работа с удаленным роботом

Управление роботом на удаленном полигоне

Обратите внимание что вы в VPN-сети удаленного полигона, и IP поменялись!

```
export ROS_MASTER_URI=http://10.8.0.6:11311
export ROS_HOSTNAME=10.8.0.XX
```

Зайти на робота напрямую

ssh pi@10.8.0.6

Итоговая работа: Найти рядового “Шарика”

Практикум: Работа с удаленным роботом

Калибровка цвета

```
import rospy
import cv2
import numpy as np
import math
from sensor_msgs.msg import Image, CompressedImage
from cv_bridge import CvBridge
rospy.init_node("photographer")

yellowLower = (95, 220, 230) # dark
yellowUpper = (125, 255, 255) # light

br = CvBridge()

mask_pub = rospy.Publisher("/elon_mask", Image, queue_size = 1)

def process():
    hsv = cv2.cvtColor(image_from_ros_camera, cv2.COLOR_RGB2HSV)
    mask = cv2.inRange(hsv, yellowLower, yellowUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    Moments = cv2.moments(mask)
    x = int(Moments["m10"] / Moments["m00"])
    y = int(Moments["m01"] / Moments["m00"])
    mask_pub.publish(br.cv2_to_imgmsg(mask))
    rospy.sleep(0.1)

def cb_video_capture(image_msg):
        global image_from_ros_camera
        np_arr = np.frombuffer(image_msg.data, np.uint8)
        image_from_ros_camera = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)

rospy.Subscriber("/front_camera/image_raw/compressed", CompressedImage, cb_video_capture)

rospy.sleep(1)

while not rospy.is_shutdown():
    rospy.sleep(0.2)
    process()
```