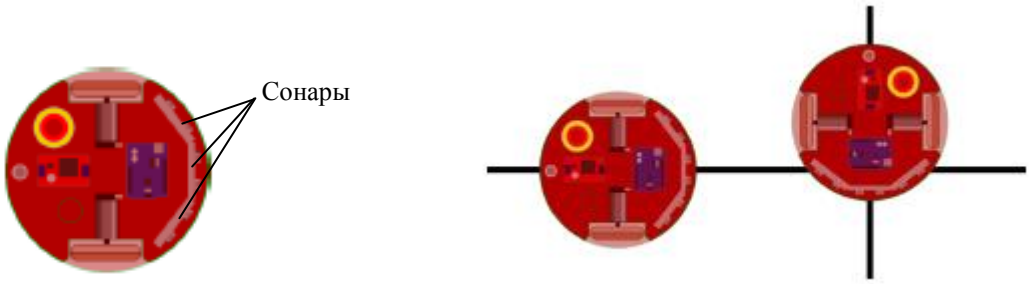
ДОМАШНЕЕ ЗАДАНИЕ N 2*

«Часть 2. Разработка и моделирование навигационной системы мобильного робота»

Цель работы: реализация алгоритма обхода препятствий мобильным роботом с системой управления на основе аппаратно-программной платформы Odroid, файловой системы ROS и рабочей среды Jupiter.

1. Теоретическая часть

В предыдущей части домашнего задания и лабораторной работы «Разработка и моделирование системы управления макетом мобильного робота» были настроены регуляторы приводов и отлажена программа прямолинейной езды. Это позволяет перейти к разработке алгоритма движения для наиболее распространенного режима езды – «*avoider*». В этом режиме робот должен объезжать неизвестные заранее препятствия, и при невозможности этого – поворачивать назад. Для этих целей в состав его системы управления дополнительно включаются три ультразвуковых дальномера – *сонара* (рис. 1):



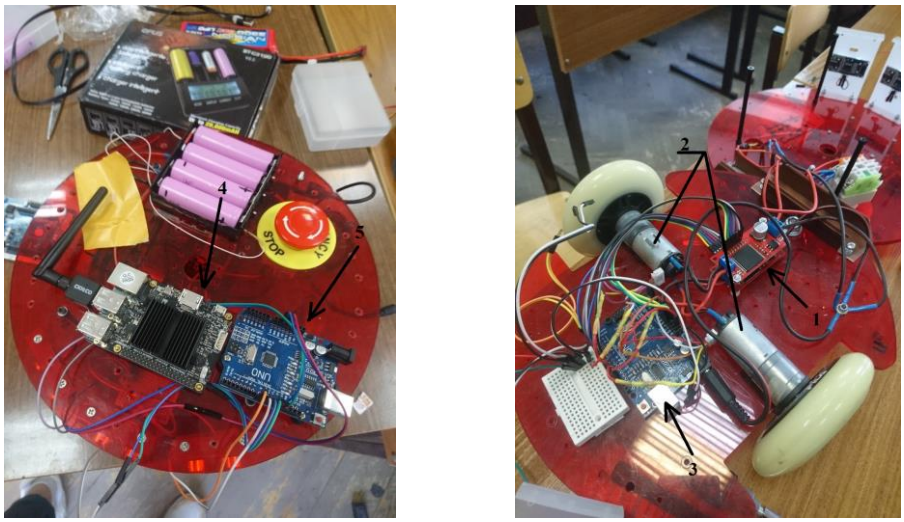
а *б*

Рис. 1. Макеты роботов для режима «avoider»: с тремя сонарами (*а*), коллизия столкновения (*б*)

Если мы установим три сонара спереди макета робота (рис. 1, *а*), то сможем в итоге определить наличие объектов в определенной зоне перед ним и делать предположение о том, как именно обходить препятствие. Такой подход применяется на многих складских роботах для предотвращения различного рода коллизий (рис. 1, *б*).

Для решения этой задачи в состав системы управления включены следующие платы:

- Odroid с установленной системой Jupiter (рис. 2, *а*);
- два микроконтроллера Arduino Uno, один из которых отвечает за обработку данных с трех ультразвуковых сонаров, другой используется как контроллер двигателей (рис. 2, *б*).



а *б*

Рис.2. Внешний вид макета робота: верхний (*а*), нижний (*б*) уровни управления. *1* – драйвер двигателей, *2* – двигатели, *3* – Arduino Uno – контроллер двигателей, *4* – плата Odroid, *5* – Arduino Uno – контроллер датчиков

Начнем с рассмотрения базовых концепций файловой системы ROS, с помощью которой работают все внутренние сервисы робота.

- Фреймворк **ROS** (*Robot Operating System*) — это файловая система для программирования роботов, предоставляющая функциональность для распределённой работы. ROS содержит набор утилит, библиотек, соглашений и работает поверх классической операционной системы, обеспечивая:
  - аппаратную абстракцию различных устройств робота,
  - низкоуровневый контроль этих устройств,
  - реализацию часто используемых функций и задач,
  - сообщений между процессами,
  - управление пакетами.
- **Пакетом** (*package*) называется наименьшая единица файловой системы. Представляет собой директорию, содержащую в себе какие-либо данные, библиотеки, исполняемые и конфигурационные файлы и т.п., логически объединенные в какой-либо полезный модуль.
- **Нода (***nodes*) — это запущенный процесс, который умеет общаться с другими процессами.
- **Топик** (*topic*) — именованный канал, соединяющий различные узлы.
- **Jupiter** — это среда, позволяющая легко работать в ROS и писать скрипты не вдаваясь во внутреннюю структуру этой системы.

В качестве основного элемента навигационной системы робота в данной работе используется система из трех **сонаров** (ультразвуковых дальномеров HC SR04) – модулей, измеряющих расстояния в радиусе четырёх метров. Принцип действия сонара основан на эхолокации: он генерирует звуковые импульсы на частоте 40 кГц и воспринимает отраженный от препятствий эхо-сигнал. Измеряя временной интервал между отправкой и получением отраженного импульса можно вычислить расстояние до препятствия (рис. 3).

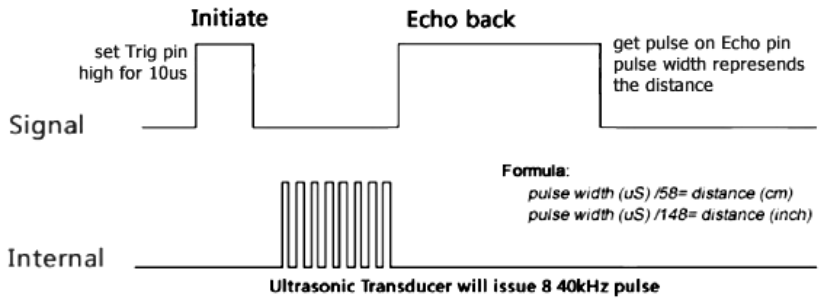


Рис. 3. Принцип действия сонара

2. Практическая часть

Практическая часть работы предназначена для получения навыков программирования систем управления мобильными роботами на основе аппаратно-программной платформы Odroid и рабочей среды Jupiter. Работа состоит из 3 частей:

1. создание и реализация алгоритма обхода препятствий;
2. калибровка дальномеров и построение их функций преобразования;
3. изучение утилиты виртуального моделирования RVIZ, позволяющей показать процесс выполнения задания визуально.

2.1. Лабораторная установка

В состав лабораторной установки входят:

1. мобильные роботы Ladybugs;
2. аппаратно-программная платформа Odroid с операционной системой ROS и рабочей средой Jupiter;
3. ноутбуки с операционной системой Windows и средой разработки Arduino IDE;
4. Wi-Fi роутер, позволяющий создавать устойчивое wi-fi соединение, к которому следует подключить ноутбуки и платформы Odroid.
5. USB-шлейфы для программирования.

Структура связей системы управления роботом имеет вид (рис. 4):

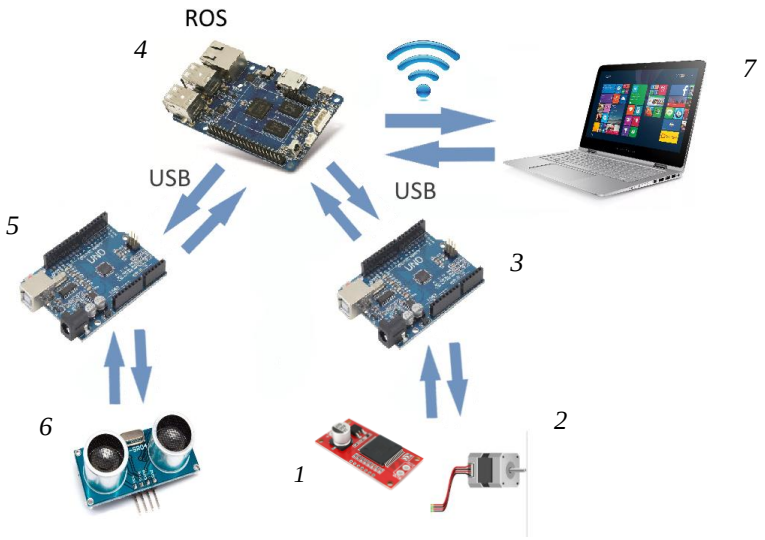


Рис. 4. Принцип соединения элементов системы управления роботом: *1* – драйвер двигателей, *2* – двигатели, *3* – Arduino Uno – контроллер двигателей, *4* – плата Odroid, *5* – Arduino Uno – контроллер датчиков, *6* – сонары, *7* – отладочный ноутбук

2.2. Порядок выполнения работы

2.2.1. Подключение платы Odroid к ноутбуку

Подключим плату Odroid к ноутбуку через порт USB и настроим ее для работы в среде Jupiter. Для этого необходимо выполнить ряд действий.

1. Сначала установим драйверы USB – для распознавания подключенной платы Odroid операционной системой. С этой целью правой кнопкой жмем на кнопку «Пуск» и выбираем пункт «Диспетчер устройств» (рис. 5):

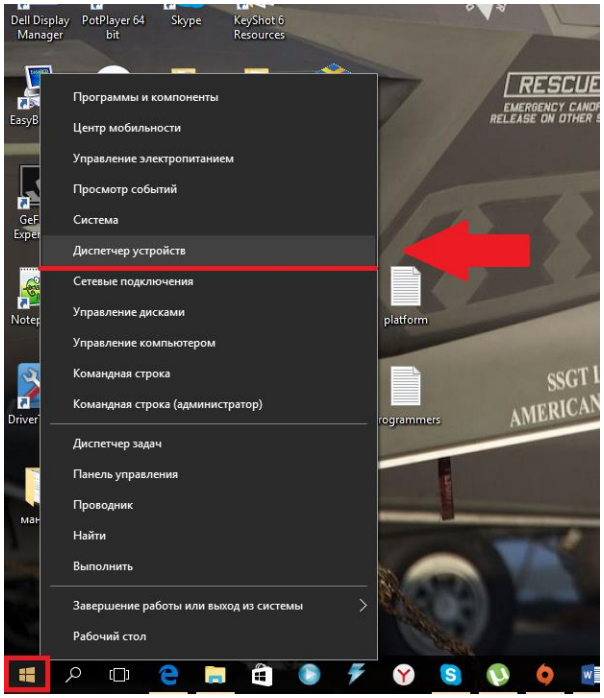


Рис. 5. Открытие диспетчера устройств

2. В появившемся окне во вкладке «Другие устройства» кликаем правой кнопкой по строчке «RNDIS» и выбираем «Обновить драйверы» (рис. 6):

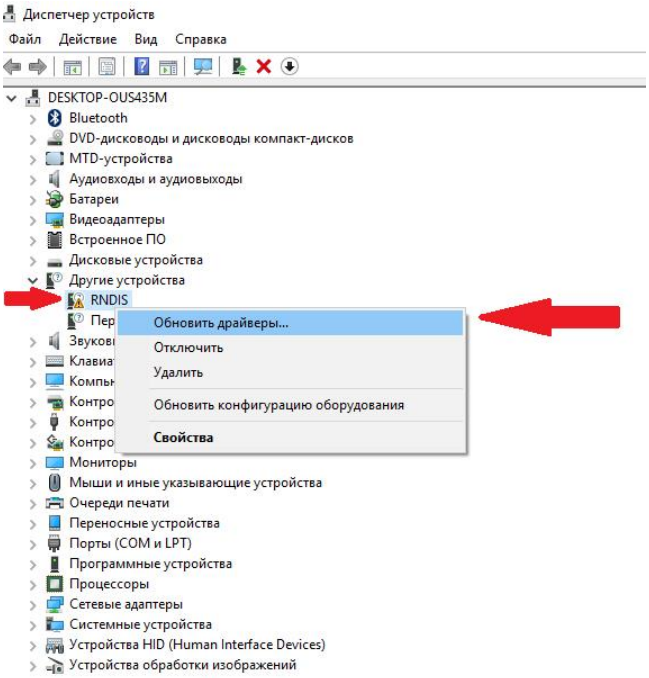
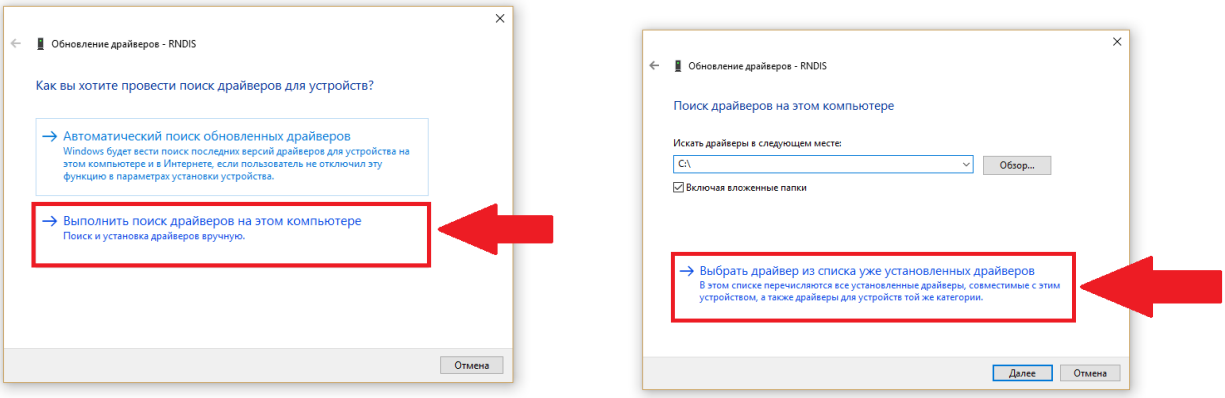


Рис. 6. Поиск и обновление драйвера платы Odroid

3. Выбираем пункт «Выполнить поиск драйверов на этом компьютере» (рис. 7, *а*):



а *б*

Рис. 7. Продолжение установки драйвера: *а* – шаг 1, *б* – шаг 2

4. Затем кликаем по «Выбрать драйвер из списка…» (рис. 8, *б*) и в появившимся списке ищем строчку «Сетевые адаптеры» (рис. 8, *а*):

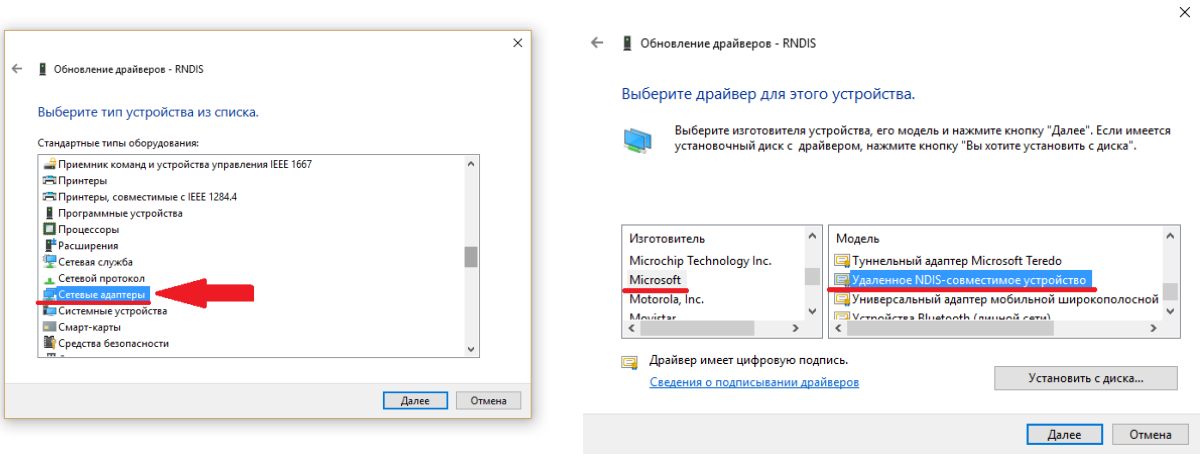


Рис. 8. Установка сетевого адаптера: *а* – выбор из списка, *б* – установка как «удаленное NDIS- совместимое устройство»

5. В списке «Изготовитель» находим строчку «Microsoft» и кликаем по ней (рис. 9, *б*). Затем, в обновленном списке «Модель» находим «Удаленное NDIS-совместимое устройство», выбираем его и жмем «Далее».
6. Жмем «Да» (рис. 9, *а*)

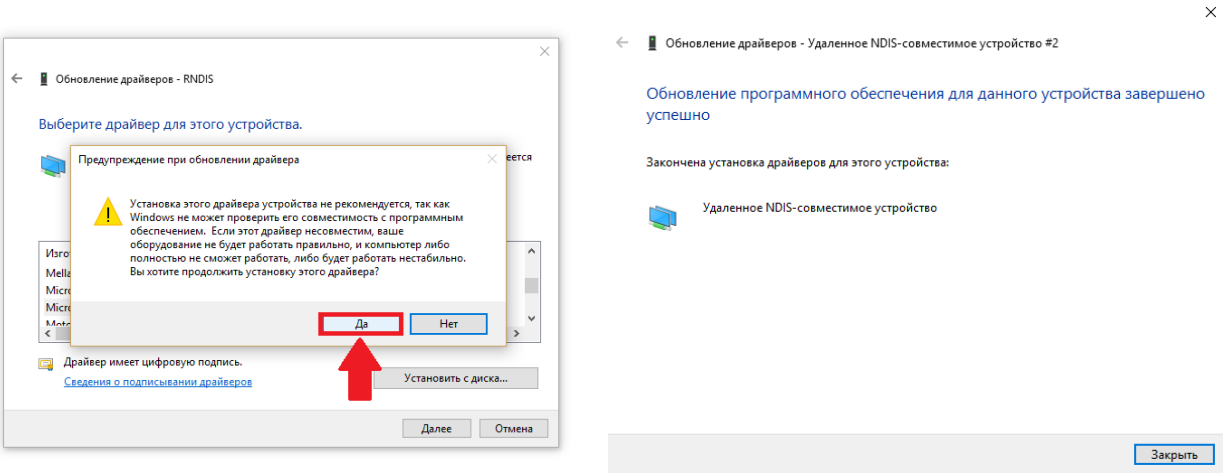


Рис. 9. Согласие с предупреждением

7. Если установка пройдет успешно, то мы получим такое сообщение (рис. 9, *б*) и вместо RNDIS во вкладке «Сетевые адаптеры» мы должны увидеть «Удаленное NDIS-совместимое устройство» (рис. 10):

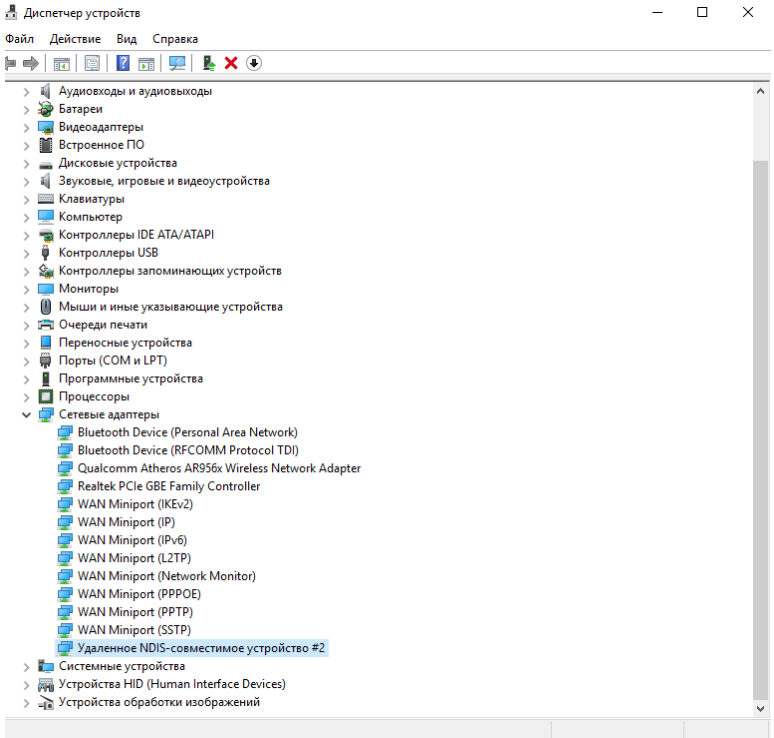


Рис. 10. Проверка диспетчера устройств после установки драйвера

8. Проверим, видит ли Arduino IDE наш сетевой порт (рис. 11):

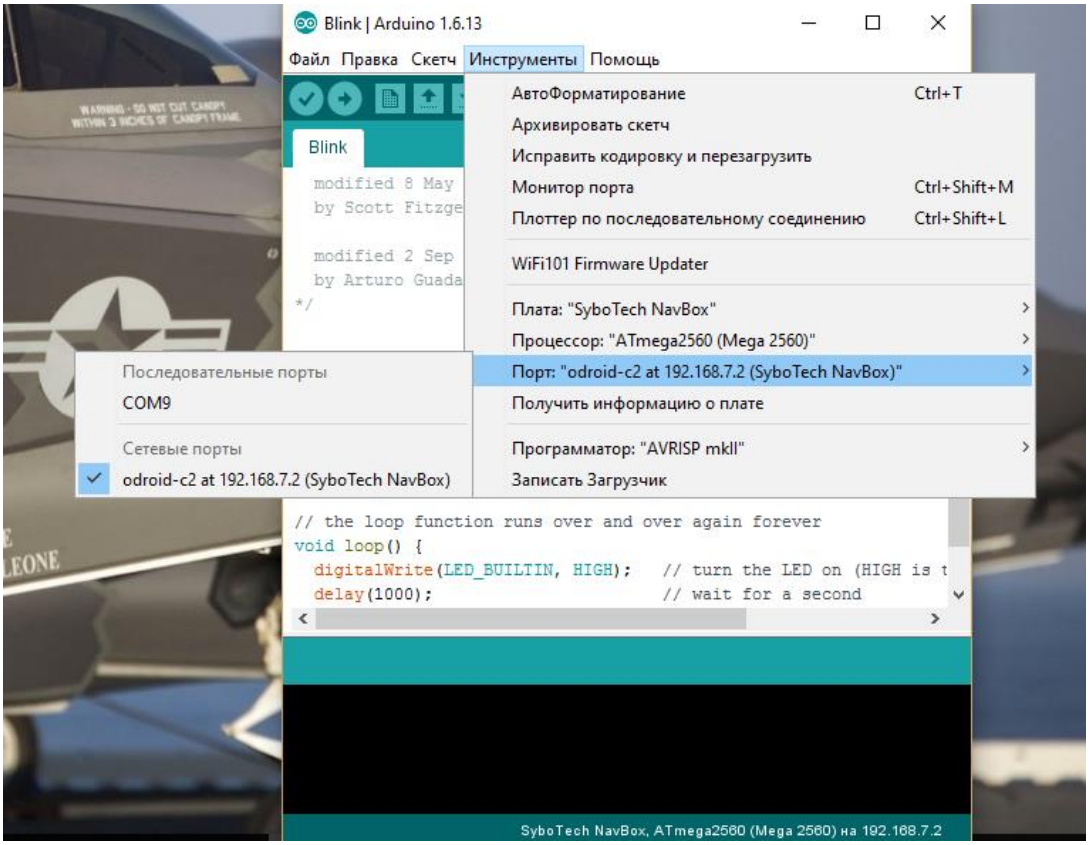


Рис. 11. Проверка соединения

2.2.2. Знакомство и работа со средой Jupiter

Для начала нужно установить соединение платы Odroid и рабочего ноутбука. С этой целью следует выполнить следующие действия.

1. Подсоединиться к роботу с помощью сети WI-FI или локально. Затем необходимо открыть браузер и набрать в адресной строке IP-адрес робота.
   Имя: *admin*
   Пароль: *aaaa*
   Открыть вкладку «system» и подключить одну из плат Arduino. Далее, обновить вкладку и только потом подключить вторую плату Arduino.
   Опять обновить.

2. В адресной строке браузера написать «*адрес_робота:8888*» и войти в рабочую среду Jupiter (пароль: *rootpass*).

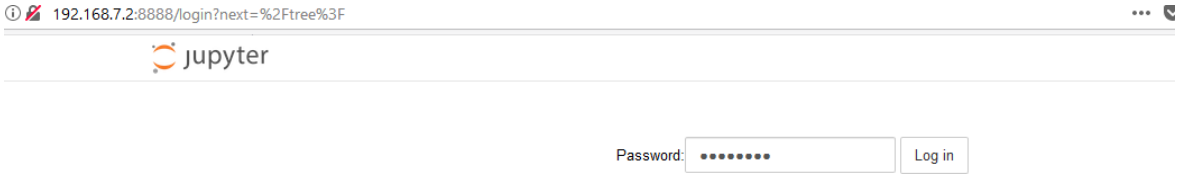


Рис. 12. Вход в систему Jupiter

Открылась файловая система робота (рис. 13).

Рис. 13. Файловый менеджер системы Jupiter

3. Необходимо открыть файл «PyPlot_example.ipynb»

4. Файл разделен на блоки, каждый блок запускается и исполняется отдельно. Управление работой блоков осуществляется с помощью клавиш, расположенных сверху (рис. 14).

Рис. 14. Элементы управления блоками скрипта

Первая часть файла отвечает за подключение библиотек. Во второй части создаем экземпляр класса «RobotTool», который позволяет инициализировать рабочую среду робота со всеми подключенными к нему устройствами.

5. Далее следует написать алгоритм объезда препятствий. Обозначим: SR04_L, SR04_M, SR04_R – топики, отвечающие за показания ультразвуковых датчиков расстояния (левый, центральный и правый, соответственно). Написание алгоритмов в среде Jupiter ведется на языке Python (рис. 15).

Initializing ROS node

```
In [2]: rospy.init_node('listener', anonymous=True)

In [3]: robot = RobotTool()

In [4]: robot.send_joint_cmd(0, 0)
        robot.send_joint_cmd(1, 0)

In [5]: distGetten = robot.get_dist()
        print(distGetten)

        [3, 3, 0]

In [ ]: robot.send_joint_cmd(0, 2)
        robot.send_joint_cmd(1, 2)
        while(1):
            distGetten = robot.get_dist()
            print(distGetten)
            if distGetten[1]>0 and distGetten[1]<=30:
                while(distGetten[1]>0 and distGetten[1]<=25):
                    distGetten = robot.get_dist()
                    robot.send_joint_cmd(0, 4)
                    robot.send_joint_cmd(1, -4)
            if distGetten[0]>0 and distGetten[0]<=30:
                while(distGetten[0]>0 and distGetten[0]<=25):
                    distGetten = robot.get_dist()
                    robot.send_joint_cmd(0, 4)
                    robot.send_joint_cmd(1, -4)
            if distGetten[2]>0 and distGetten[2]<=30:
                while(distGetten[2]>0 and distGetten[2]<=25):
                    distGetten = robot.get_dist()
                    robot.send_joint_cmd(0, -4)
                    robot.send_joint_cmd(1, 4)
            robot.send_joint_cmd(0, 4)
            robot.send_joint_cmd(1, 4)
```

Рис. 15. Скрипт работы сонаров

Переменная «DistGetten» содержит в себе расстояние от датчиков до препятствия. Далее описываются условия приближения и действия для преодоления препятствия (рис. 16).

```
        print(distGetten)
        if distGetten[1]>0 and distGetten[1]<=30:
            while(distGetten[1]>0 and distGetten[1]<=25):
                distGetten = robot.get_dist()
                robot.send_joint_cmd(0, 4)
                robot.send_joint_cmd(1, -4)
        if distGetten[0]>0 and distGetten[0]<=30:
            while(distGetten[0]>0 and distGetten[0]<=25):
                distGetten = robot.get_dist()
                robot.send_joint_cmd(0, 4)
                robot.send_joint_cmd(1, -4)
        if distGetten[2]>0 and distGetten[2]<=30:
            while(distGetten[2]>0 and distGetten[2]<=25):
                distGetten = robot.get_dist()
                robot.send_joint_cmd(0, -4)
                robot.send_joint_cmd(1, 4)
        robot.send_joint_cmd(0, 4)
        robot.send_joint_cmd(1, 4)

[12, 0, 11]
[0, 0, 1]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 2]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[7, 0, 13]
[0, 13, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
[0, 0, 14]
[0, 0, 27]
[14, 0, 94]
[0, 6, 62]
[0, 0, 76]
[0, 0, 52]
[0, 0, 37]
[0, 0, 25]
[0, 0, 13]

In [ ]:
```

Рис. 16. Данные с сонаров

После блока, отвечающего за считывание данных с сонаров, приводятся данные с топиков «SR04_R», «SR04_M», «SR04_L». Каждый из этих топиков содержит в себе переменную, показывающую расстояние от объекта до правого, среднего и левого сонаров соответственно.

2.2.3. Управление роботом с помощью пульта

Управление роботом может осуществляться с помощью кнопок, каждая из которых отвечает за определенное действие.

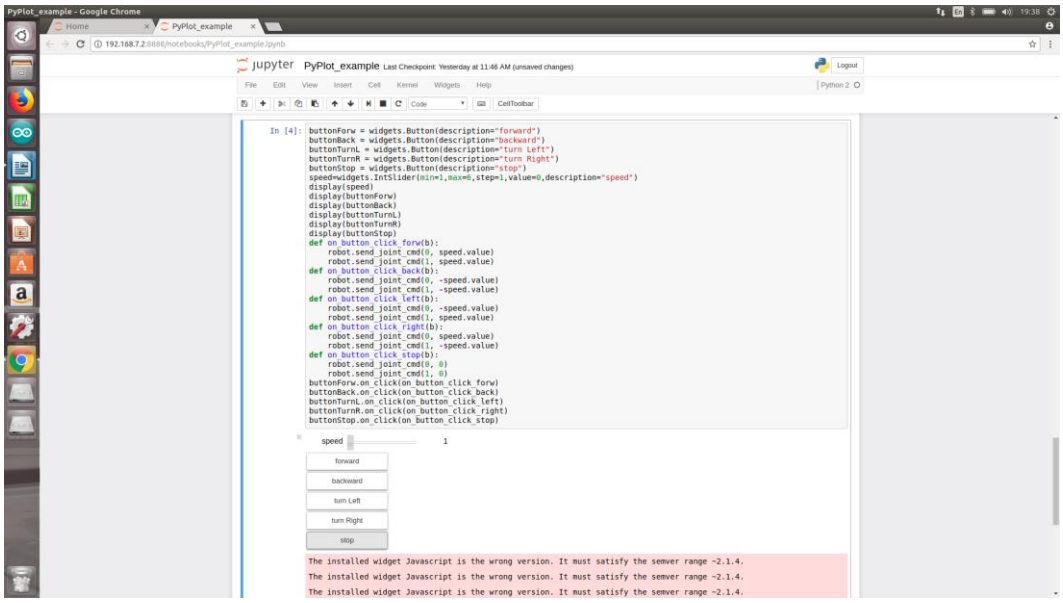


Рис. 17. Скриншот программы формирования кнопок управления двигателями робота

Для этого необходимо написать в созданном ранее файле «PyPlot_example.ipynb» код, представленный выше (рис. 17).

2.2.4. Калибровка сонаров

При калибровке сонара определяют его функцию преобразования. Построим графики показаний датчиков в виде:

$$U = L(x),$$

где U – выходные значение (код), x – дальность (мм). Для этого воспользуемся следующей программой (рис. 18):

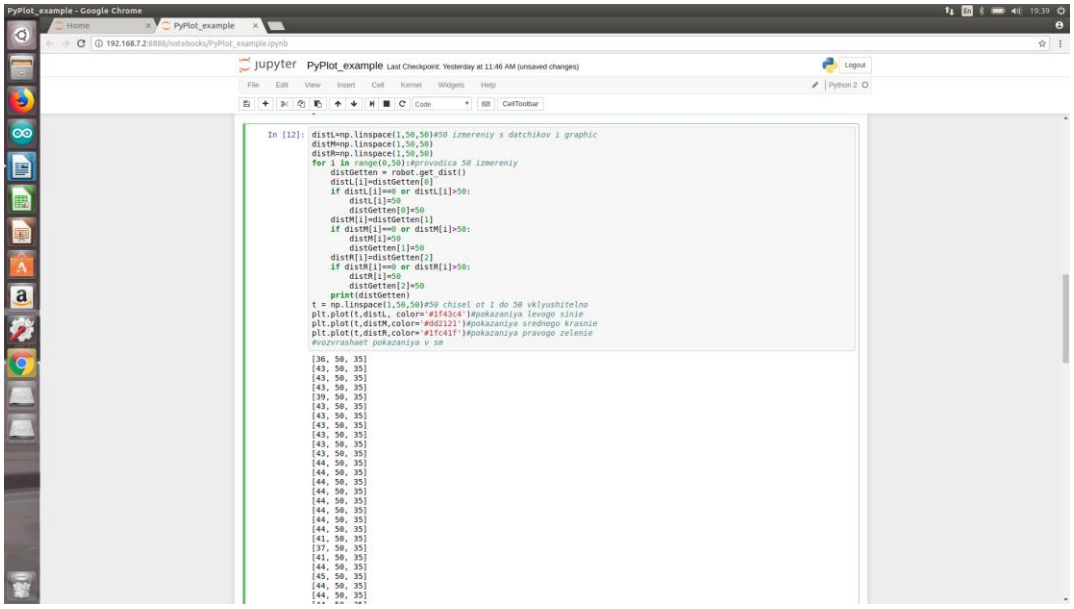


Рис. 18. Программа определения функции преобразования сонара

Мы также можем построить графики показаний сонаров (рис. 19).

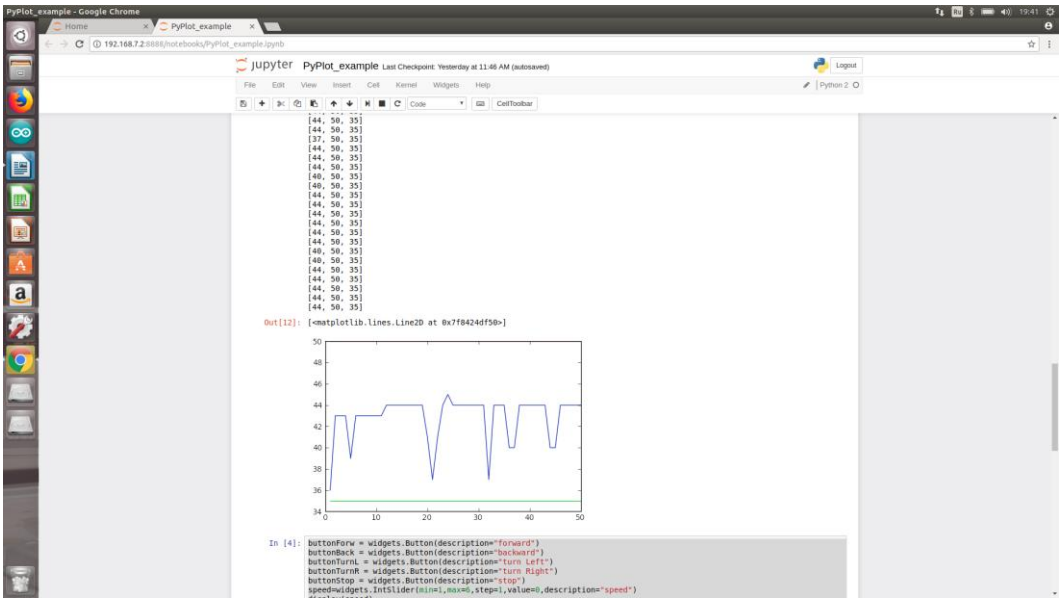


Рис. 19. График изменения показаний сонаров от положения робота

Как следует из графиков (рис. 18 и рис. 19) функцию преобразования сонара в первом приближении можно аппроксимировать линейной зависимостью:

$$U[\text{В}] = k\,x[\text{мм}],$$

где k – коэффициент преобразования сонара [В/мм].

2.2.4. Работа с утилитой RVIZ

Утилита RVIZ представляет собой средство визуализации данных, получаемых с сенсоров. Она позволяет создать виртуальную модель робота, воссоздать решение роботом задачи и визуализировать данные, получаемые роботом с различных датчиков.

Запуск RVIZ происходит через терминал. Подключение к роботу осуществляется по протоколу SSH (SSH - Secure Shell) – протоколу удаленного управления компьютером с операционной системой Linux.

1. Открываем терминал (CTRL+ALT+T), и прописываем команду «*ssh root@ IP-робота*» (рис. 20):

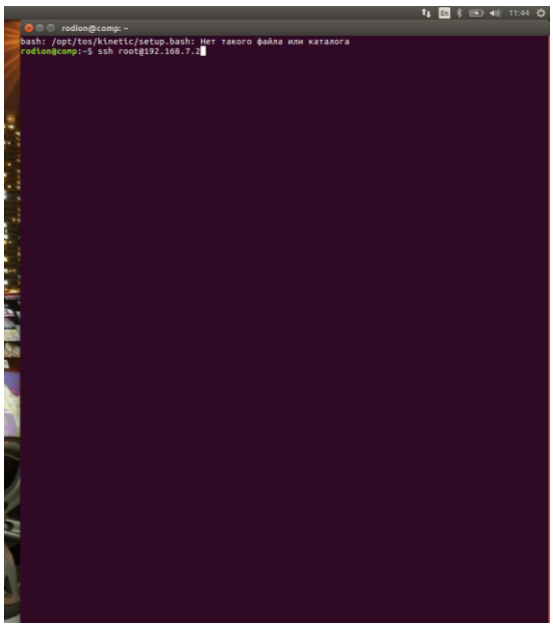
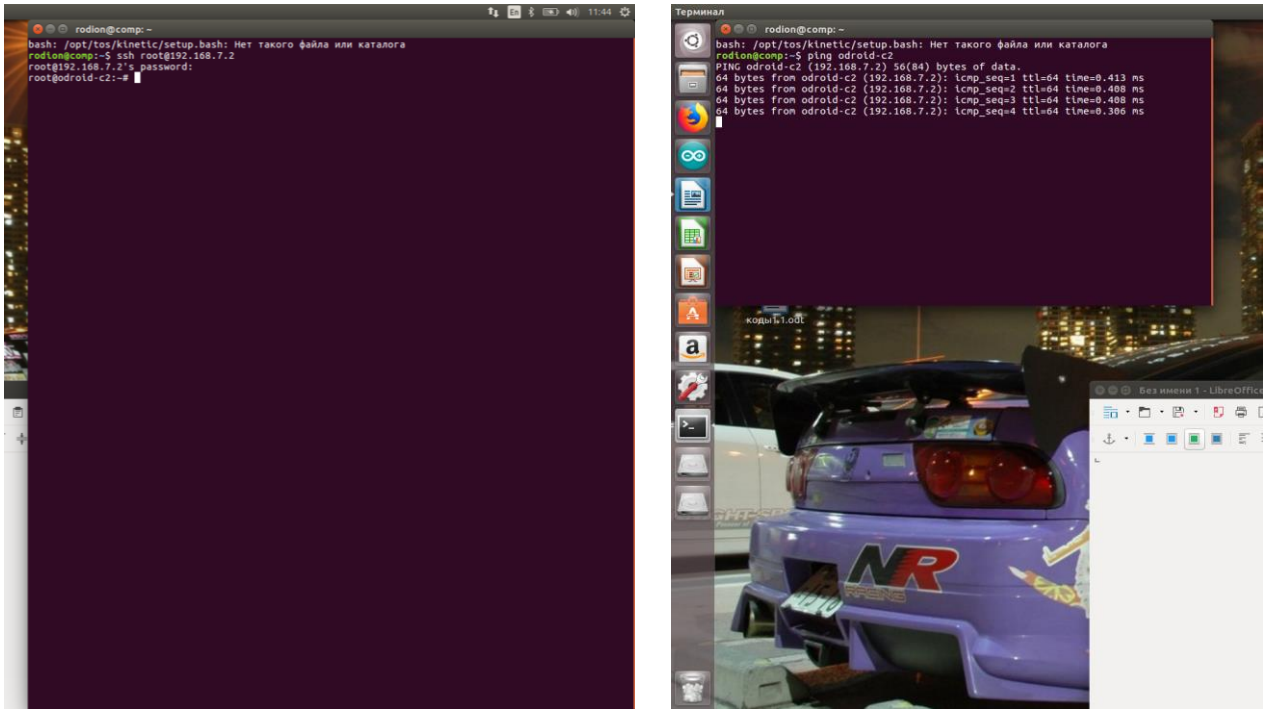


Рис. 20. Команда запуска RVIZ в терминале

2. Далее следует с помощью протокола SSH (он должен быть установлен на Odroid и ноутбуках) установить соединение между платой Odroid и ноутбуком.
   Сначала вводим пароль от робота: *rootpass* (рис. 21, *а*).



а *б*

Рис. 21. Подключение по протоколу SSH: *а* – команда подключения, *б* – проверка установления связи

3. Проверяем связь с роботом, если связь не установлена, добавляем в файл «sudo/etc/hosts» строку «*192.168.7.2 odroid-c2*» (рис. 21, *б*).

4. Далее необходимо открыть второй терминал и в обоих окнах написать команду: «*export ROS_MASTER_URI=http://odroid-c2:11311*» (рис. 22):

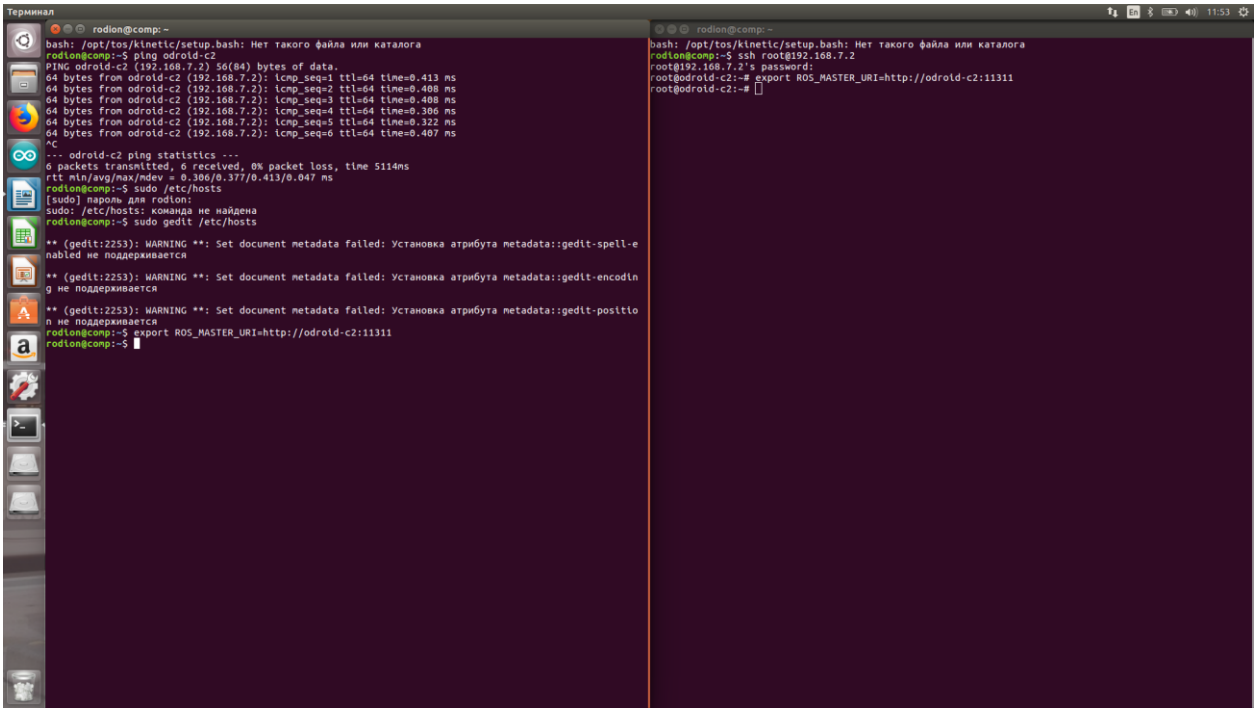


Рис. 22. Копирование адреса компьютера-мастера в файл конфигурации Odroid

5. Теперь необходимо проверить правильность установленного IP-адреса компьютера с помощью команды «*ifconfig*» (рис. 23).

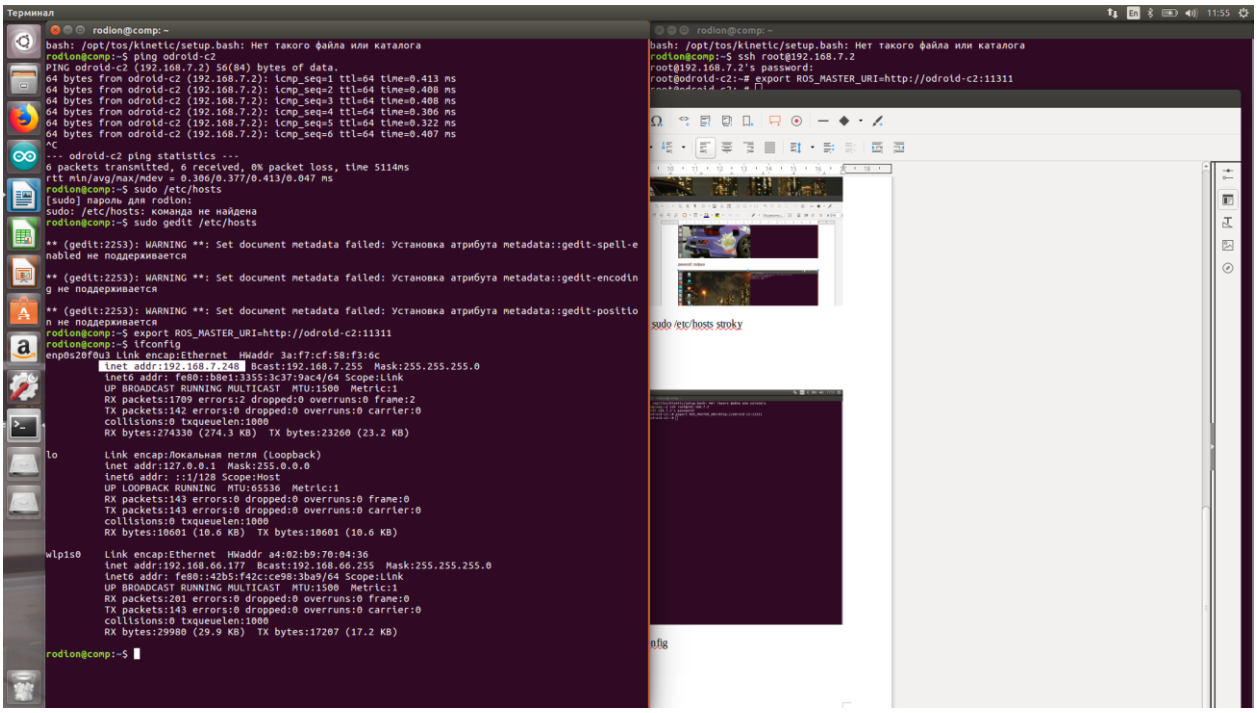
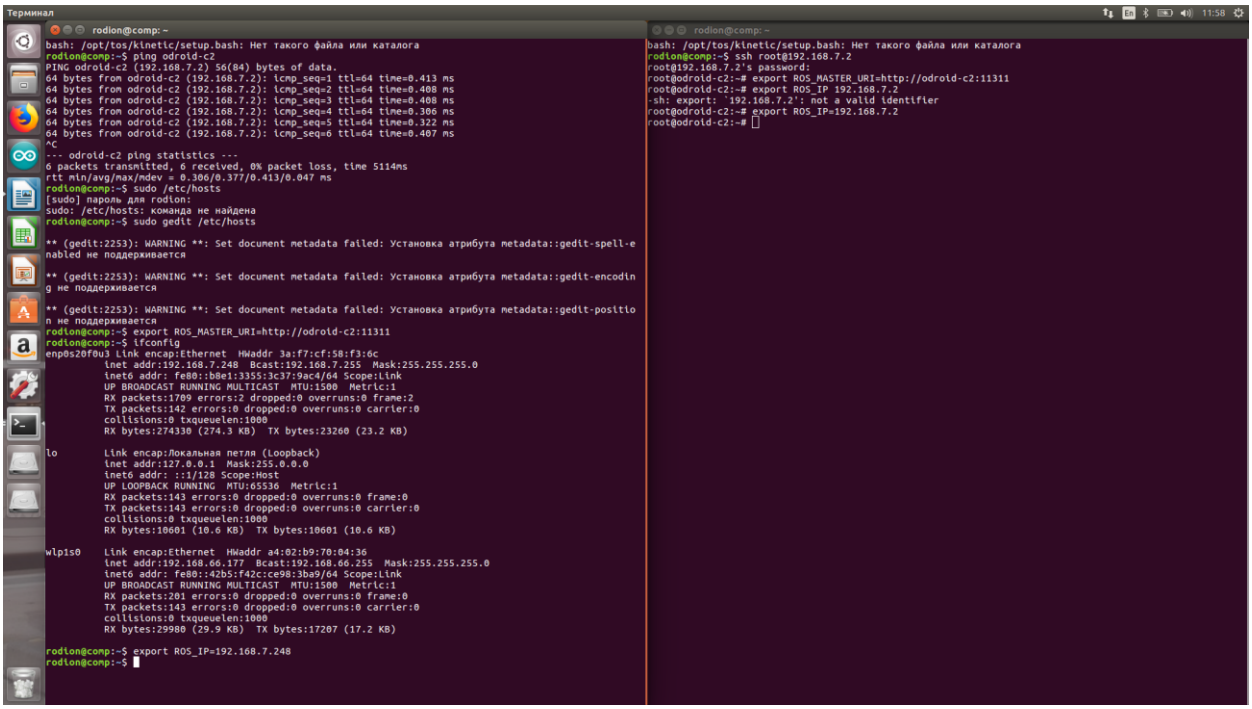


Рис. 23. Проверка правильности IP-адреса мастера

6. Далее набираем команды:
   - на терминале компьютера: «*export ROS_IP=192.168.7.248*» (рис. 24, *а*);
   - на терминале робота: «*export ROS_IP=192.168.7.2*» (рис. 24, *б*).



а *б*

Рис. 24. Назначение IP-адресов

7. Затем, в терминале робота, пишем: «*source /opt/ros/indigo/setup.bash*» (рис. 25):

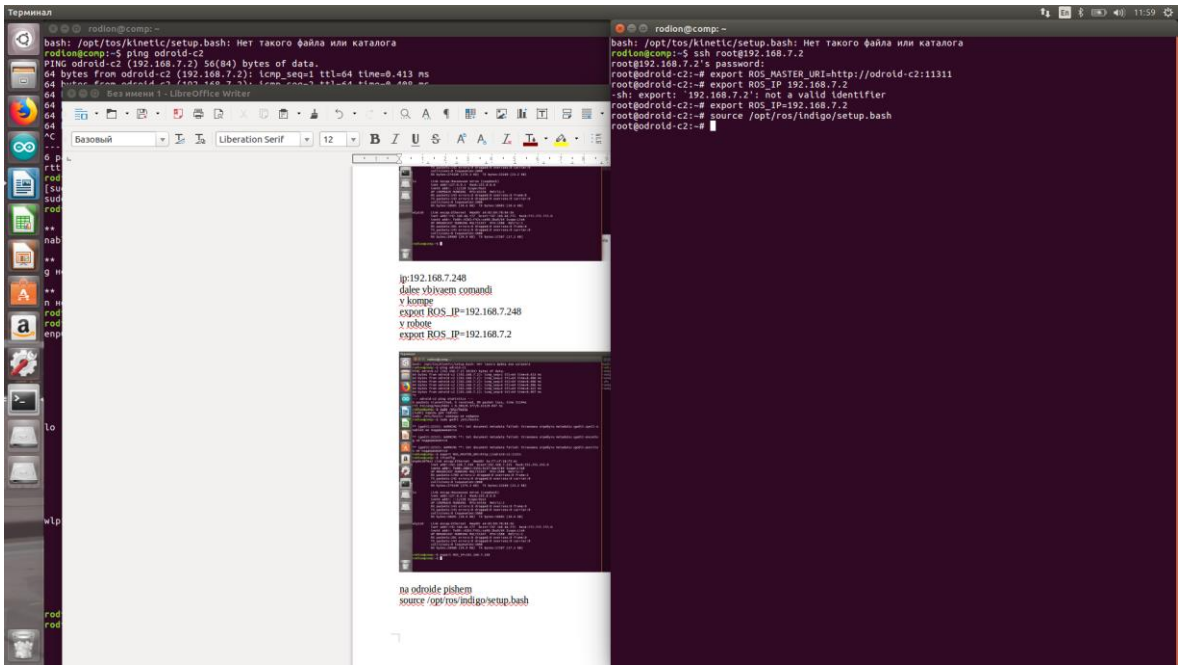


Рис. 25. Обновляем данные путем ссылки на файл setup.bash

8. Запускаем утилиту RVIZ командой «*rosrun rviz rviz*» (рис. 26):

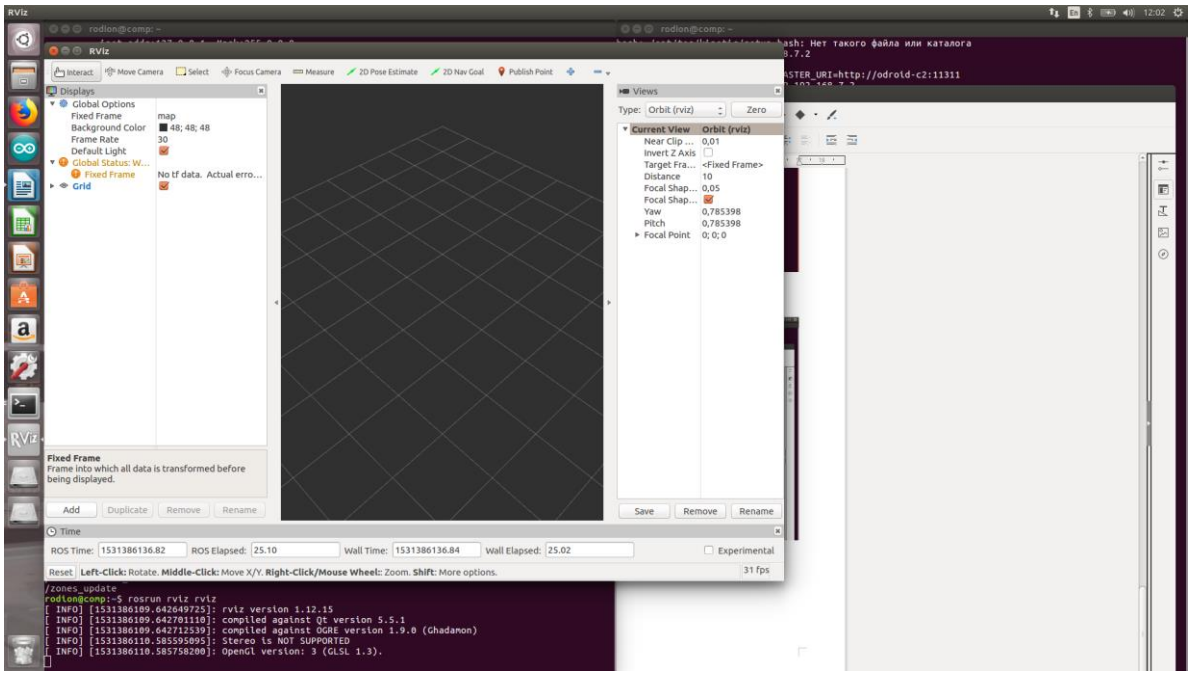


Рис. 26. Окно утилиты rviz

9. Добавляем функцию «odometry» (в ней выбираем «axes», «axes» и топик ODOM), топик TF; в поле «fixed frame» выбираем также топик ODOM (рис. 27). Это позволяет роботу сохранять свою одометрию и ориентироваться в пространстве с помощью фрейма из топика TF. (Здесь, под фреймом понимается базовая точка отсчета, относительно которой определяется положение робота при его движении).

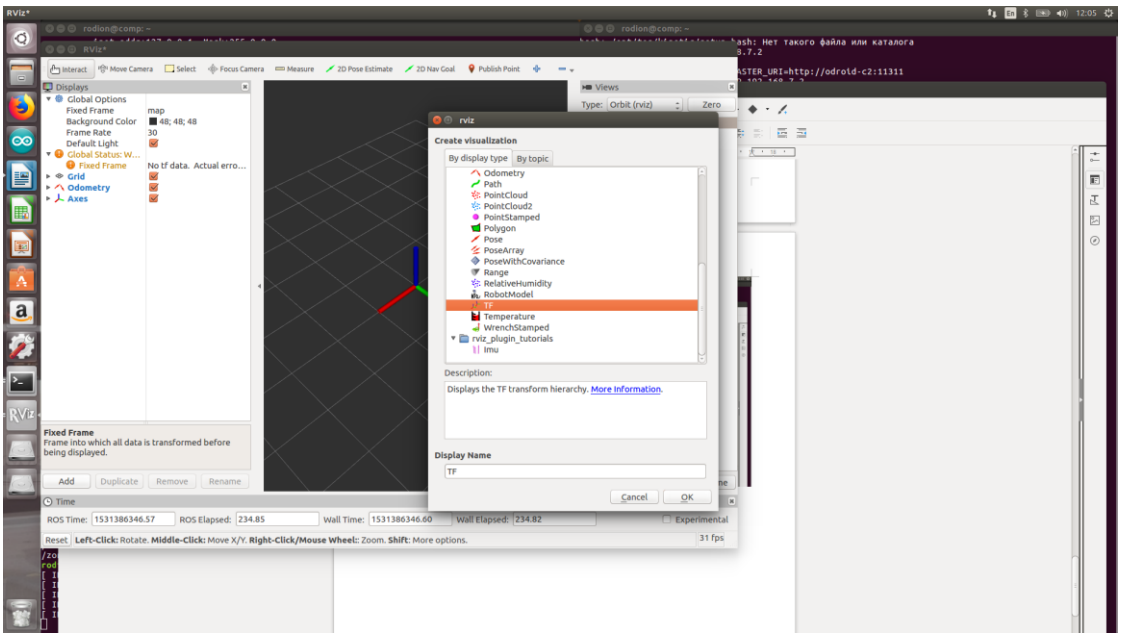


Рис. 27. Добавление топика TF

10. Теперь мы можем наблюдать движение робота и координаты, по которым происходит его ориентирование.

2.2.5. Задание для самостоятельной работы

1. Определить коэффициент преобразования каждого сонара и параметры его диаграммы направленности.
2. Построить маршрут движения робота и карту его препятствий.