Содержание

Первое включение

Администрирование ROS

Пакет turtlebro

Платформа TurtleBoard

Плата TurtleBoard

Инструкция робота TurtleBro

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

Данная инструкция описывает устройство робота, основы работы с ним, а также рассказывает последовательность действий для настройки ROS на плате Raspberry Pi.

Более глубокое изучение ROS можно начать с нашей книги "Введение в ROS"

Робот разработан и поддерживается проектом "БратьяВольт"

Первое включение робота TurtleBro

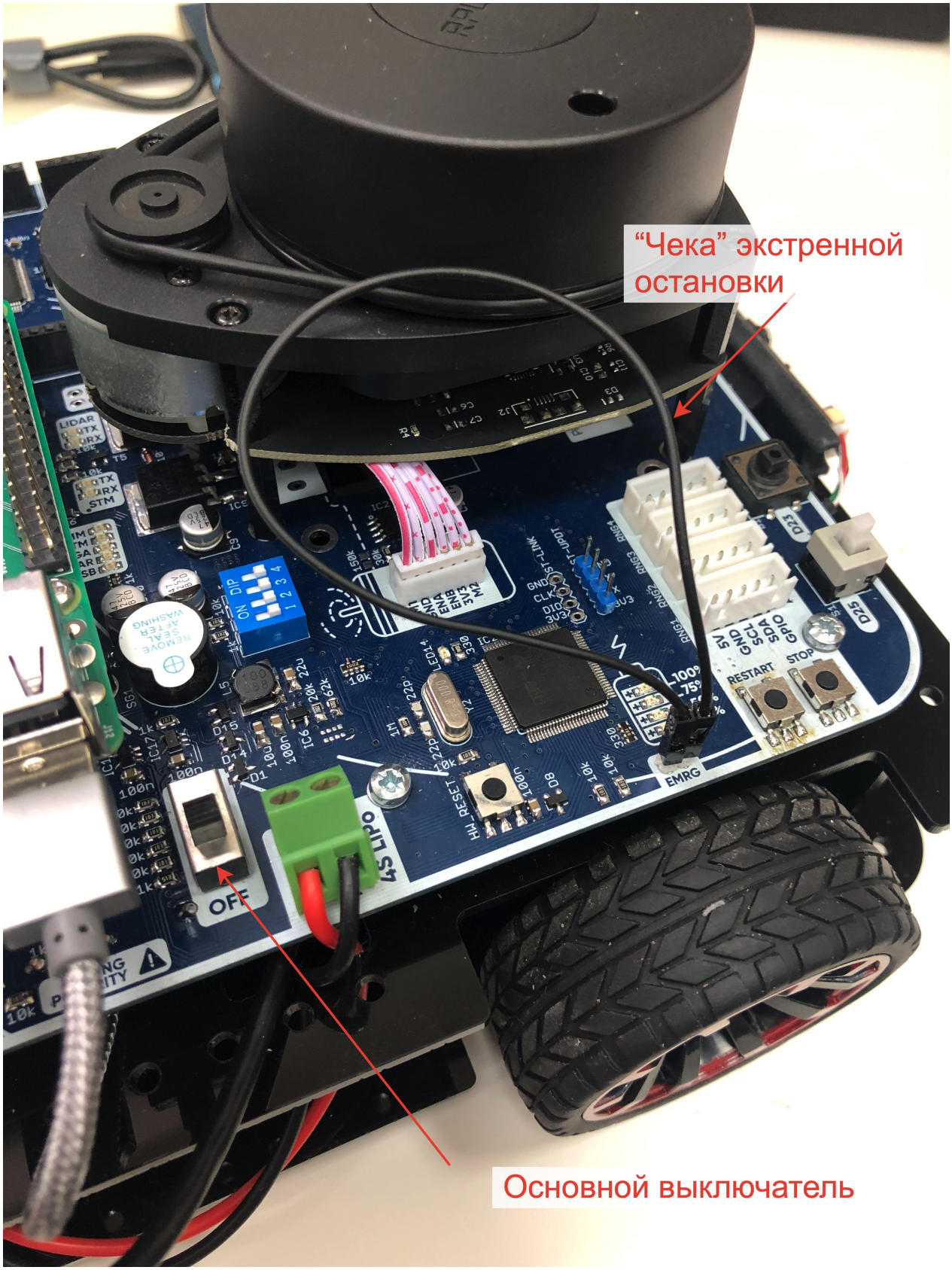
> Архив документации для ROS melodic и python2
>
> http://archive.turtlebro.ru/manual/melodic/

Безопасность

Перед использованием робота необходимо внимательно изучить данный раздел. В случае использования робота в учебном классе преподаватель должен ознакомить учеников с основными частями раздела.

Экстренное отключение моторов робота

Для экстренной остановки робота необходимо выдернуть "чеку" экстренной остановки. Чека выполнена замкнутым проводом "петлей" длиной около 10см и располагается справа от лидара.



При извлечении чеки (размыкании контакта) робот обесточивает моторы и подает постоянный звуковой сигнал. Отключение питания касается только моторов, другие системы робота продолжают работать.

Все люди, взаимодействующие с роботом (ученики, судьи, волонтеры и т.д.) должны изучить расположение чеки и произвести пробное ее выдергивание для запоминания алгоритма действий в экстренной ситуации.

Работа с Li-Po аккумулятором

В комплекте с роботом идет литий-полимерный аккумулятор. Данный тип аккумулятора может выдавать очень большие токи, что в случае короткого замыкания или механического повреждения может привести к его возгоранию.

Необходимо максимально аккуратно производить любые операции с аккумулятором, избегая его ударов, падений и других механических воздействий. Не допускается использование поврежденных (в том числе вздутых) аккумуляторов.

Аккумулятор оснащен разъемом T-Plug, который не позволит перепутать полярность при подключении к роботу.

На плате функционирует блок контроля напряжения аккумулятора, отключающий большинство систем робота при разряде батареи. В состоянии с разряженным аккумулятором робот прекращает работу и начинает издавать звуковой сигнал с интервалом 30 с. Для продолжения работы необходимо выключить тумблер питания и произвести зарядку батареи.

Подключение выносных датчиков

Запрещается подключать к роботу датчики, которые каким либо образом будут зафиксированы на теле человека:

- Медицинские датчики (ЭКГ и т.п.)
- Датчики интегрированные в одежду (перчатки с датчиками положения пальцев и т.п.)

Полезные ссылки

Книга введение в ROS http://docs.voltbro.ru/starting-ros/

Репозиторий файлов для задач WorldSkills https://github.com/voltbro/ws-sro

Образовательный портал с уроками ROS http://learn.voltbro.ru/

Пакет автономной навигации https://github.com/voltbro/turtlebro_navigation

Пакет для подключения джойстика JoyBro https://github.com/voltbro/joybro

Пример .cpp пакета для работы на RaspberryPi https://github.com/voltbro/cpp_package_demo

Youtube-канал проекта "Братья Вольт" https://www.youtube.com/channel/UCxmgjYqHVA6UUq294899gTw

Телеграм-канал технической и информационной поддержки продуктов проекта «Братья Вольт»

https://t.me/+a2Fn7chqQec1MWVi

Обновление платформы

Обновление на версию 0.9 и новее

В прошивке начиная с версии 0.9 используется ROS Noetic и Python3. Для обновления на актуальную версию необходимо:

1. Обновить прошивку системной платы на прошивку версии 2.0 или новее;
2. Подготовить SD карту для RaspberryPi с версией ПО 0.9 или новее;
3. В файле .ros_params, расположенный в директории /home/pi робота Turtlebro, проверить значение переменной окружения ROVER_WHEEL_PARAM для задания коэффициента вращения моторов. Для старых моторов параметр ROVER_WHEEL_PARAM должен равняться 22500, для новых моторов этот параметр должен быть равен 12280.

Начиная с версии 0.9 пакеты ROS устанавливаются в двух разных директориях окружений. Системные пакеты установлены в директорию `ros_catkin_ws`. В директорию `catkin_ws` установлены пользовательские пакеты.

Подключение робота к Сети

Мы рекомендуем использовать единую сеть для всех устройств входящих в учебный класс. Желательно наличие интернета в этой сети.

Все роботы по умолчанию настроены для работы в WiFi и Ethernet в режиме клиента с получением настроек по DHCP.

Настройка подключения к Wi-Fi

Настройка подключения робота TurtleBro к новым Wi-Fi сетям

Подключение к точке WiFi по умолчанию

По умолчанию при старте Rasberry Pi попытается подключиться к Wi-Fi точке доступа с параметрами

```
SSID: TurtleBro
Pass: turtlew001
```

или

```
SSID: TurtleBro5G
Pass: turtlew001
```

Настройка подключения к новой Wi-Fi через Ethernet кабель

Это самый простой способ, но требует наличия Ethernet кабеля, подключенного к роутеру.

Подключите работающий Raspberry к Ethernet кабелю.

Определите IP адрес устройства через веб-интерфейс роутера.

Зайдите на ssh на устройство `ssh pi@IP` или `pi@turtlebroNNN.local` Подключение по SSH

Откройте в текстовом редакторе файл `wpa_supplicant.conf`

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

В конец файла добавьте настройки Wi-Fi вашей сети в виде:

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

При указании пароля от Wi-Fi сети, обязательно наличие кавычек.

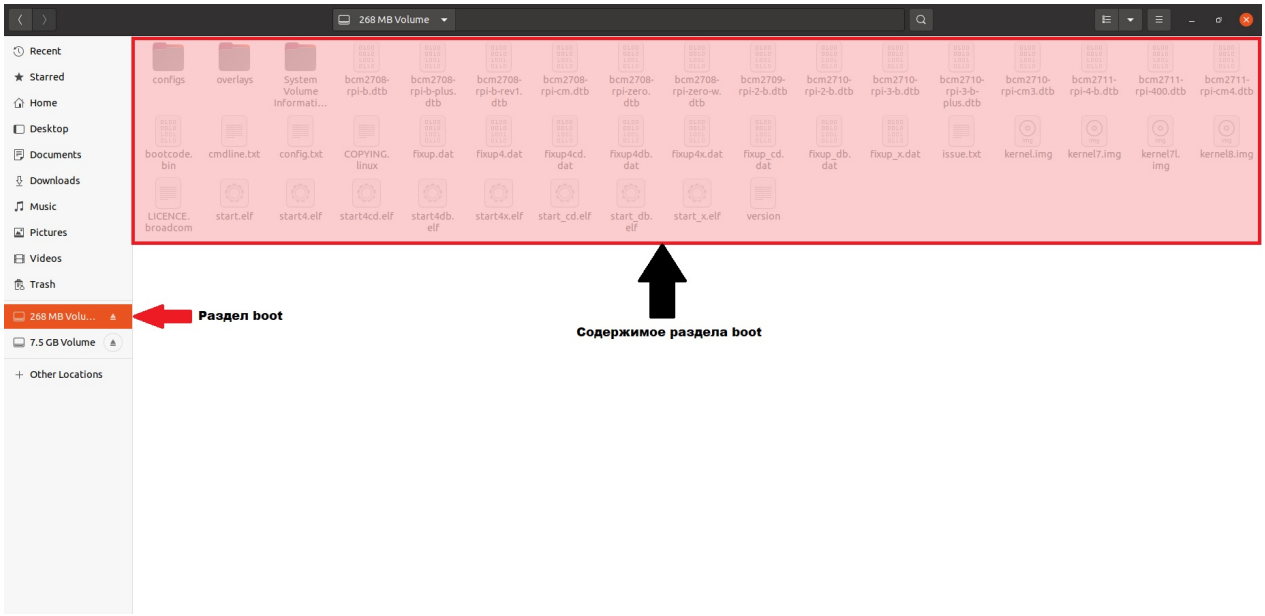
Сохраните файл и перезагрузите Raspberry Pi.

Настройка подключения к новой Wi-Fi через SD карту

На SD карте, содержащей готовый образ системы для запуска на роботе, есть два раздела разного размера. Обычно они называются `system` и `boot`, но иногда система может назвать их по-другому при подключении к компьютеру.

Раздел `system` содержит стандартный набор директорий файловой системы Linux и занимает основной объем SD карты (подробнее - https://ru.wikipedia.org/wiki/FHS). \

Раздел `boot` небольшой и содержит настройки запуска Raspberry (подробнее - https://www.raspberrypi.org/documentation/configuration/boot_folder.md)



Если на этапе загрузки Raspberry найдет файл `wpa_supplicant.conf` в разделе `boot` то этот файл будет перемещен в `/etc/wpa_supplicant/wpa_supplicant.conf` и таким образом станет конфиграционным файлом подключения к Wi-Fi сетям.

Чтобы сконфигурировать Raspberry в этом режиме подключите SD карту Raspberry к вашему компьютеру.

Создайте на SD карте файл `/boot/wpa_supplicant.conf` с содержанием:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

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

Использование кавычек `"` в файле с настройками обязательно.

Сохраните файл.

Установите SD карту в Raspberry и дождитесь завершения загрузки.

Внимание, файл перезапишет все ваши текущие настройки Wi-Fi в директории `/etc/wpa_supplicant`

Генерация passphrase

Если вы не хотите показывать ваш пароль для пользователей, вы можете сгенерировать специальный ключ `passphrase` и указать его в файле `wpa_supplicant.conf` . Пользователи не смогут "подсмотреть" в файле пароль от Wi-Fi и подключать к вашей сети свои устройства.

Для генерации `passphrase` необходимо запустить программу `wpa_passphrase` далее указать имя Wi-Fi сети и пароль. Далее вывод который покажет программа необходимо записать в файл `wpa_supplicant.conf` в параметр `passphrase = ""`

Дополнительные команды Wi-Fi

```
sudo iwlist wlan0 scan | grep ESSID #сканирование сети
sudo iwlist wlan0 freq  #доступные каналы Wi-Fi
```

Более подробно о настройке Wi-Fi через командную строку:
https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

Подключение по SSH к роботу

По умолчанию на Raspberry Pi запущен SSH сервис.

> Пароль для ssh по умолчанию: `brobro`

Для подключения вам необходимо Настроить сеть

Каждый робот имеет уникальное имя вида **turtlebroNN.local,** где NN - это номер. При правильной настройке сети и вашего роутера, вы сможете сразу подключиться к Raspberry по его имени

```
ssh pi@turtlebro20.local
```

Если подключение не происходит, вам необходимо определить IP-адрес робота. Это можно сделать подключившись к роутеру и найдя имя робота в списке подключенных устройств.

Далее подключиться к роботу можно по IP, где `192.168.0.11` это адрес робота

```
ssh pi@192.168.0.11
```

> Удобно привязать IP-адрес робота к его MAC адресу.

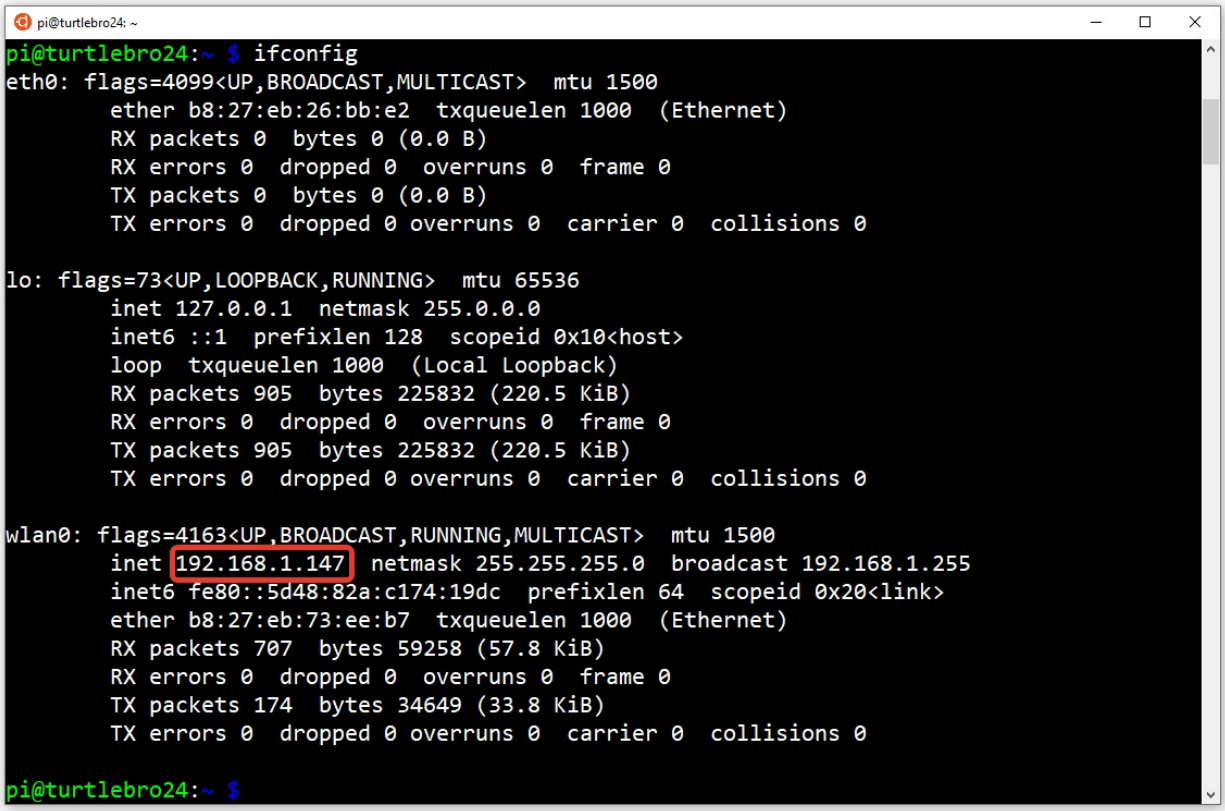
Для работы по имени вида .local для Windows необходимо установить программу https://support.apple.com/kb/DI999?locale=ru_RU

Администраторам системы необходимо настроить поддержку Multicast-DNS.

Для доступа по SSH из Windows можно использовать программу PuTTY.

Как определить IP-адрес робота

Если вы подключились к роботу, то для того, чтобы определить его IP-адрес наберите в терминале команду `ifconfig`



Если вы не смогли подключиться к роботу по его имени, вы можете посмотреть его IP-адрес на SD карте.

Выключите робота, вытащите SD карту и подключите ее к компьютеру. В папке `/boot/configs` будут расположены файлы с данными о сетевых настройках робота.

В файле `ifconfig.dump` будет находиться IP-адрес.

Если вы видите, что робот не может подключиться к вашей сети, проверьте файл `wpa_supplicant.conf` на наличие в нем вашей Wi-Fi сети.

Смена пароля пользователя

Для смены пароля вам необходимо знать ваш текущий пароль или быть пользователем с возможностью запускать программы через sudo

Для смены пароля текущего пользователя, просто запустите программу `passwd` . Далее необходимо указать ваш текущий пароль и установить новый. В разных дистрибутивах могут быть свои ограничения на длину и "сложность" пароля. Поэтому поставить очень простой пароль вы не сможете.

Для смены пароля другому пользователю, выполните команду `sudo passwd user_name` Где user_name это имя пользователя, которому вы хотите сменить пароль. Так как вы запустили программу с привилегиями `sudo` то для смены пароля не нужно знать его текущий пароль. Также перестают работать ограничения на сложность пароля.

Подключение к ROS на роботе

Подключение к роботу по сети из ROS

На компьютере необходимо указать, по какому адресу находится ROS-мастер `roscore`. Для этого на компьютере в терминале необходимо установить переменную окружения `ROS_MASTER_URI`:

```
export ROS_MASTER_URI=http://192.168.0.250:11311/
```

Где `192.168.0.250` - это **IP-адрес робота**.

Для правильной работы сети, также необходимо установить на компьютере в терминале переменную ROS_HOSTNAME:

```
export ROS_HOSTNAME=192.168.0.100
```

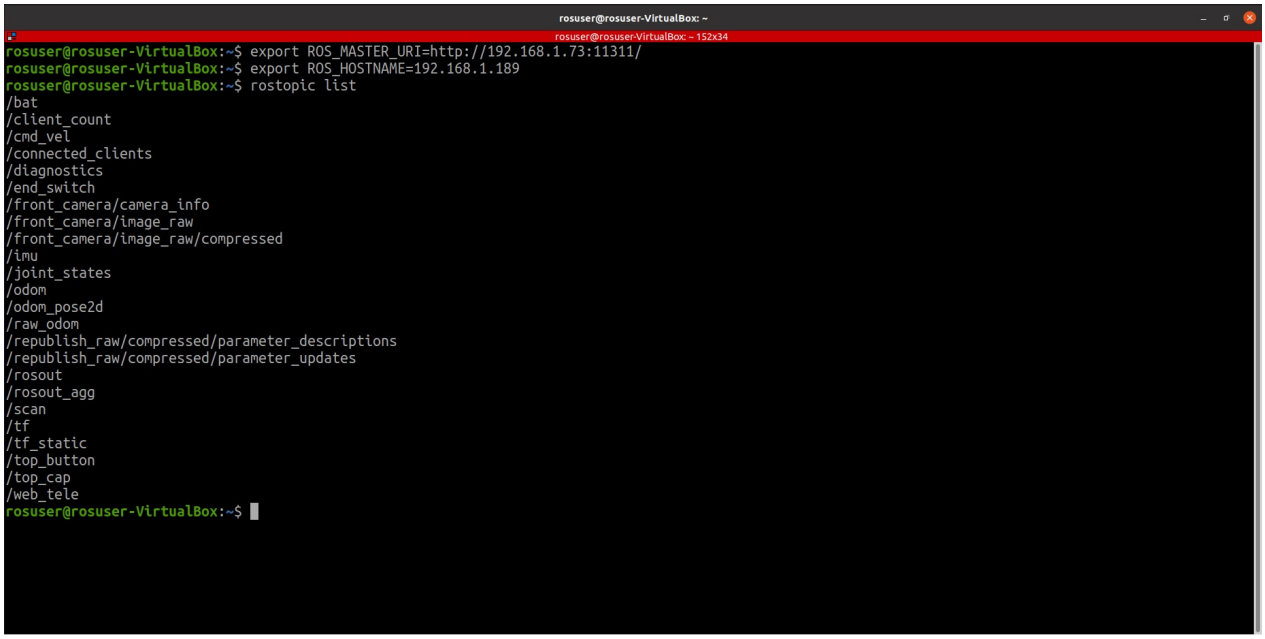
Где `192.168.0.100` - это **IP-адрес вашего компьютера**.

Удобно прописать `ROS_MASTER_URI` и `ROS_HOSTNAME` в файле `.bashrc`, для того чтобы каждый раз не делать `export`. Для этого необходимо открыть в текстовом редакторе файл `~/.bashrc` и в самый конец добавить строчки:

```
export ROS_MASTER_URI=http://<IP-адрес робота>:11311/
export ROS_HOSTNAME=<IP-адрес компьютера>
```

Аналогичные настройки `ROS_MASTER_URI` и `ROS_HOSTNAME` применены на роботе через файл `.bashrc`.

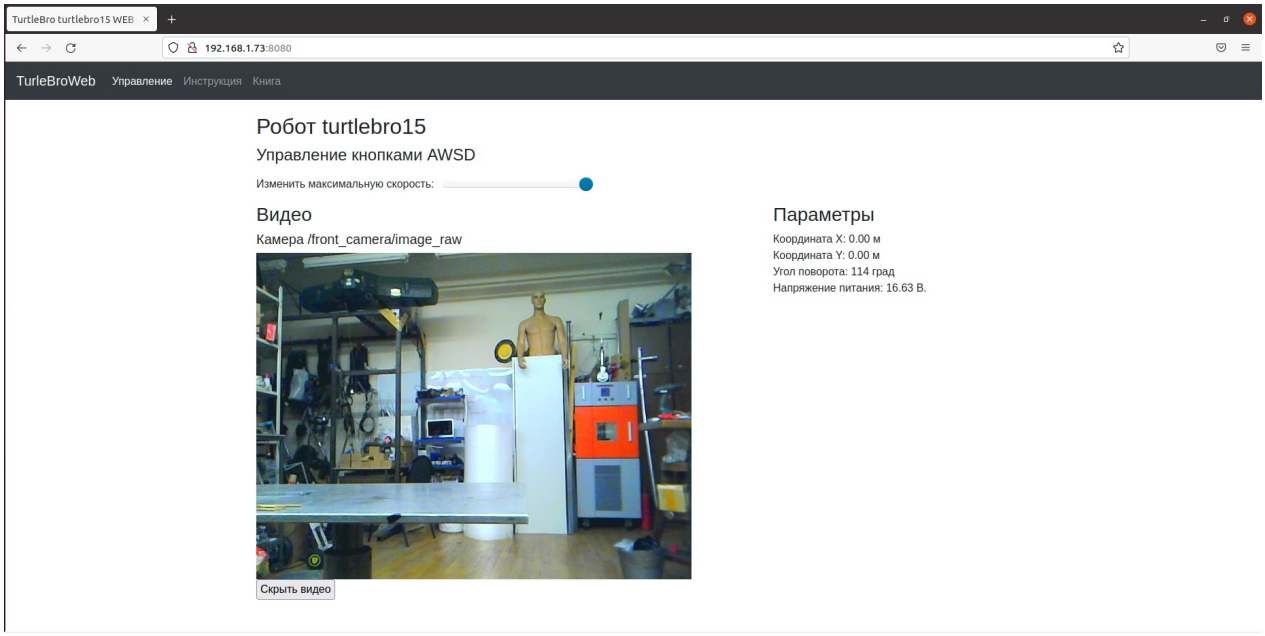
Если все настройки проведены верно, вы можете выполнить на вашем компьютере команды ROS и увидеть результат их выполнения:



Веб-интерфейс

Для начала работы с роботом вы можете зайти на веб-сервер запущенный на роботе `http://<IP-адрес робота>:8080` (указав IP адрес вашего робота)

На этой странице будут доступны основные данные робота и изображение получаемое из камеры:



Роботом можно управлять кнопками **WSAD**.

Выключение робота

Безопасное выключение (сохранение файлов на SD карту)

Если вы редактировали файлы на роботе и хотите действительно убедиться что при выключении робота они не потеряются, то прежде чем выключить роботы необходимо выполнить команду:

```
sudo sync
```

Данная команда запишет все "закешированные" файлы на SD карту.

Только после этого возможно выключить робота тумблером выключени питания или перезагрузить его.

RaspberryPi

Обновление образа операционной системы робота TurtleBro

Компьютеры Raspberry, идущие в комплекте с роботами, поставляются с предустановленными ОС `Raspberry Pi OS lite` (https://www.raspberrypi.org/downloads/raspbian/) , `ROS Noetic` и всеми необходимыми системными пакетами.

Обновление образа ОС возможно через скачивание и полную перезапись SD карты. Для работы необходима карта размером **16Gb.**

Образ можно использовать для следующих моделей Raspberry:

- Raspberry 3 Model B
- Raspberry 3 Model B+
- Raspberry 4

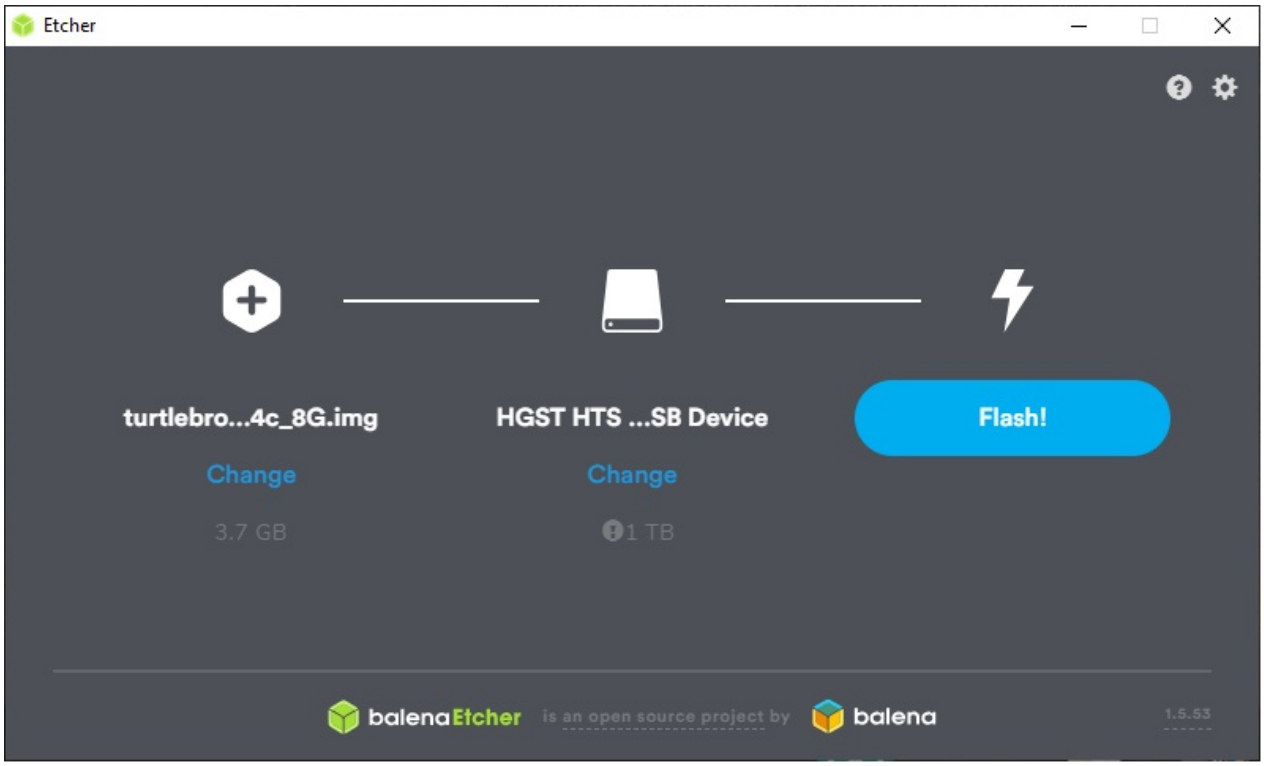
Скачать образ

Архив доступных образов: https://yadi.sk/d/U0G80JoXs9eqcA

> Рекомендуем выбирать самую последнюю версию!

Загрузка образа ОС на SD-карту

Проще всего загрузить образ на SD карту с помощью программы balenaEtcher https://www.balena.io/etcher/ Программа обладает поддержкой всех основных операционных систем.



По умолчанию, имя робота установлено `turtlebro01` . Рекомендуется сразу изменить его на имя согласно номера платы `turtlebroNN` . Для этого необходимо отредактировать файлы `/etc/hosts` и `/etc/hostname` расположенные на роботе и переименовать `turtlebro01->tutlebroNN` . Удобнее всего это сделать на компьютере с ОС Linux подключив SD карту. Или уже на включенном роботе, а потом перезагрузить его.

Версию образа можно посмотреть в файле `/boot/version` на SD карте.

Установленное ПО на SD карту

На карте установлены основные необходимые **ROS** пакеты:

```
rosinstall_generator, actionlib, actionlib_msgs, amcl, angles, base_local_planner, bond, bondcpp, camera_calibration_parsers, camera_info_manager, carrot_planner, catkin, class_loader, clear_costmap_recovery, cmake_modules, costmap_2d, cpp_common, cpp_package_demo, cv_bridge, cv_camera, diagnostic_msgs, dwa_local_planner, dynamic_reconfigure, fake_localization, gencpp, geneus, genlisp, genmsg, gennodejs, genpy, geometry_msgs, global_planner, gmapping, image_transport, joint_state_publisher, kdl_parser, laser_geometry, map_msgs, map_server, message_filters, message_generation, message_runtime, mk, move_base, move_base_msgs, move_slow_and_clear, nav_core, nav_msgs, navfn, nodelet, openslam_gmapping, orocos_kdl, pluginlib, python_orocos_kdl, python_qt_binding, robot_state_publisher, ros_environment, rosapi, rosauth, rosbag, rosbag_migration_rule, rosbag_storage, rosbash, rosboost_cfg, rosbridge_library, rosbridge_msgs, rosbridge_server, rosbuild, rosclean, rosconsole, rosconsole_bridge, roscpp, roscpp_serialization, roscpp_traits, roscpp_tutorials, roscreate, rosgraph, rosgraph_msgs, roslang, roslaunch, roslib, roslint, roslisp, roslz4, rosmake, rosmaster, rosmsg, rosnode, rosout, rospack, rosparam, rospy, rospy_tutorials, rosserial_arduino, rosserial_client, rosserial_msgs, rosserial_python, rosservice, rostest, rostime, rostopic, rosunit, roswtf, rotate_recovery, rplidar_ros, sensor_msgs, smclib, std_msgs, std_srvs, stereo_msgs, tf, tf2, tf2_geometry_msgs, tf2_kdl, tf2_msgs, tf2_py, tf2_ros, tf2_sensor_msgs, topic_tools, trajectory_msgs, urdf, urdf_parser_plugin, uvc_camera, visualization_msgs, voxel_grid, xmlrpcpp
```

Также установлены пакеты для функционирования робота:

`turtlebro` \ `turtlebro_extra` \ `turtlebro_navigation` \ `turtlebro_web`

Сервисы TurtleBro

При загрузке ОС происходит запуск сервиса `turtlebro` (файл `/lib/systemd/system/turtlebro.service`), который запускает `roscore` и запускает лаунч-файл пакета `turtlebro` из `/etc/ros/turtlebro.d/turtlebro.launch`.

Оба файла перезаписываются при сборке пакета, поэтому их редактирование имеет смысл только в экспериментальных целях. Все изменения необходимо производить в файлах пакета `turtlebro`, а потом производить его "сборку".

Команды управления сервисом turtlebro

Остановить сервис

```
sudo systemctl stop turtlebro
```

Запустить сервис

```
sudo systemctl start turtlebro
```

Убрать сервис из автозагрузки

```
sudo systemctl disable turtlebro
```

Поставить сервис в автозагрузку

```
sudo systemctl enable turtlebro
```

Сборка пакетов ROS

Рабочее окружение

На роботе создано два рабочих окружения для ROS пакетов. Директория `catkin_ws` для пользовательский пакетов и `ros_catkin_ws` для системных пакетов.

Пользовательское окружение в директории `catkin_ws`

Рекомендуется все новые и пользовательские пакеты устанавливать а директорию `catkin_ws/src`

Например установка пакета из git-репозитория:

```
cd ~/catkin_ws/src
git clone repo_name
cd ..
catkin_make --pkg repo_name
```

При работе в директории `catkin_ws` ROS использует исходные файлы сразу из этой директории. Поэтому, например, при изменении лаунч-файлов, дополнительно собирать пакеты не нужно. Такой подход упрощает тестирование и разработку.

Обновление системных пакетов

Системные пакеты ROS установлены из исходных кодов, а не загружены при помощи пакетного менеджера `apt` .Таким образом, для обновления старых системных пакетов и установки новых необходимо собирать пакеты из исходных кодов.

Сборка дистрибутива ROS

Все команды выполняются в директории основного окружения ROS `/home/pi/ros_catkin_ws`

Пересобрать все системные пакеты ROS

```
sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/noetic -DPYTHON_EXECU
TABLE=/usr/bin/python3
```

Все системные пакеты после сборки будут остановленны в директорию `/opt/ros/noetic`

Собрать один конкретный пакет можно командой:

```
sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/noetic -DPYTHON_EXECU
TABLE=/usr/bin/python3 --pkg=pkg_name
```

где `pkg_name` это имя того пакета, который надо собрать отдельно.

Установка новых системных пакетов из дистрибутива ROS

Все новые пакеты необходимо также устанавливать из дистрибутива пакетов ROS.

Установка пакета `new_pack` из дистрибутива пакетов ROS:

```
rosinstall_generator new_pack --rosdistro noetic --deps --tar > new_pack.rosinstall
vcs import src < new_pack.rosinstall
rosdep install --from-paths ./src --ignore-packages-from-source --rosdistro noetic -y
sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/noetic -DPYTHON_EXECU
TABLE=/usr/bin/python3 --pkg=new_pack
```

Если пакет не имеет зависимостей, то его можно собрать без сборки всего ROS.

```
sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/noetic -DPYTHON_EXECU
TABLE=/usr/bin/python3 --pkg=new_pack
```

Описание

Пакет должен быть установлен в ОС робота на SD карте Raspberry, который подключен непосредственно к системной плате `TurtleBoard`

В пакете собраны все необходимые зависимости и лаунч-файлы, необходимые для работы с периферией `TurtleBoard`

- Библиотека `rosserial`, настроенная для работы с пользовательским МК Arduino и системным микроконтроллером STM32
- Работа с Лидарами RPLidar/YDLidar
- Базовая модель робота в формате **URDF**, а также настройки robot_state_publisher для работы с /tf топиками

Проверить текущую версию пакета можно командой `rosversion turtlebro`

Установка и обновление пакетов

Установить последнюю версию пакета `turtlebro` можно из репозитория GitHub https://github.com/voltbro/turtlebro

Пакет необходимо установить в директорию /home/pi/catkin_ws/src/turtlebro

Скачать пакет (если пакет не был установлен):

```
cd ~/catkin_ws/src/
git clone https://github.com/voltbro/turtlebro
```

Обновить пакет (если пакет был установлен через git):

```
cd ~/catkin_ws/src/turtlebro
git pull
```

После обновления необходимо произвести сборку пакета `turtlebro` утилитой `catkin_make` :

```
cd ~/catkin_ws
catkin_make --pkg turtlebro
```

При сборке будут перезаписаны все файлы пакета в директорию `/opt/ros/noetic` и все файлы управления `systemd` сервисами.

Далее необходимо остановить и запустить сервис turtlebro

Параметры и настройка через launch

Загрузка робота

После включения робота, происходит загрузка системы и автоматический запуск всех необходимых для работы робота нод. Управление нодами, которые будут загружены, возможно через лаунч-файл `/etc/ros/turtlebro.d/turtlebro.launch`

По умолчанию запускаются следующие ноды:

`/arduino_serial_node` \ `/republish_raw` \ `/robot_state_publisher` \ `/rosapi` \ `/rosbridge_websocket` \ `/rosout` \ `/rplidarNode` \ `/simple_odom` \ `/stm_serial_node` \ `/uvc_camera_node` \ `/web_telemetry_node` \ `/web_video_server` \ `/webserver`

Работают топики:

`/bat` \ `/client_count` \ `/cmd_vel` \ `/connected_clients` \ `/diagnostics` \ `/front_camera/camera_info` \ `/front_camera/image_raw` \ `/front_camera/image_raw/compressed` \ `/imu` \ `/joint_states` \ `/odom` \ `/odom_pose2d` \ `/raw_odom` \ `/republish_raw/compressed/parameter_descriptions /republish_raw/compressed/parameter_updates` \ `/rosout` \ `/rosout_agg` \ `/scan` \ `/tf` \ `/tf_static` \ `/web_tele`

Запущены сервисы:

`/arduino_serial_node/get_loggers /arduino_serial_node/set_logger_level /board_info /power/off /power/reset /republish_raw/compressed/set_parameters /republish_raw/get_loggers /republish_raw/set_logger_level /reset /robot_state_publisher/get_loggers /robot_state_publisher/set_logger_level /rosapi/action_servers /rosapi/delete_param /rosapi/get_loggers /rosapi/get_param /rosapi/get_param_names /rosapi/get_time /rosapi/has_param /rosapi/message_details /rosapi/node_details /rosapi/nodes /rosapi/publishers /rosapi/search_param /rosapi/service_host /rosapi/service_node /rosapi/service_providers /rosapi/service_request_details /rosapi/service_response_details /rosapi/service_type /rosapi/services /rosapi/services_for_type /rosapi/set_logger_level /rosapi/set_param /rosapi/subscribers /rosapi/topic_type /rosapi/topics /rosapi/topics_and_raw_types /rosapi/topics_for_type /rosbridge_websocket/get_loggers /rosbridge_websocket/set_logger_level /rosout/get_loggers /rosout/set_logger_level /rplidarNode/get_loggers /rplidarNode/set_logger_level /set_camera_info /set_pid /simple_odom/get_loggers /simple_odom/set_logger_level /start_motor /stm_serial_node/get_loggers /stm_serial_node/set_logger_level /stop_motor /uvc_camera_node/get_loggers /uvc_camera_node/set_logger_level /web_telemetry_node/get_loggers /web_telemetry_node/set_logger_level /web_video_server/get_loggers /web_video_server/set_logger_level /webserver/get_loggers /webserver/set_logger_level`

Настройка параметров запуска

Настройка параметров в файле .ros_params

Файл `/home/pi/.ros_params` содержит параметры окружения для старта ROS:

```
export ROS_HOSTNAME=$machine_ip
export ROS_IP=$machine_ip
export ROS_MASTER_URI=http://$machine_ip:11311
export ROVER_MODEL=turtlebro
export ROVER_WHEEL_PARAM=12280
```

Переменная окружения ROVER_WHEEL_PARAM определяет параметр **wheel_param** для определения типа моторов. Для старых моторов применяется значение: 22500; для новых моторов: 12280. Если одометрия робота из топика не совпадает с реальным перемещением робота, необходимо провести калибровку параметра.

Параметры одометрии

Для того, чтобы одометрия точно отображала реальное перемещение робота используются два параметра:

- **stm_serial_node/wheel_distance** double
  - тип данных - *числовой 64-битный тип*
  - размерность - *метры*
  - обозначает - *расстояние между колесами*
- **stm_serial_node/wheel_param** uint32_t
  - тип данных - *числовой 32-битовый без знака*
  - размерность - *безразмерный*
  - обозначает- *число тиков энкодера на метр.* Является расчётным коэффициентом.

Для расчёта параметра **wheel_param** используется следующая формула:

```
wheel_param = ticks*red_ratio/circle

ticks - количество тиков энкодера на один оборот мотора без учёта редуктора, штук
red_ratio - коэффициент передачи редуктора, безразмерный
circle - длина окружности колеса, метров

Пример рассчета для текущих моторов:
red_ratio=56
circle=pi * D(диаметр колеса)=3,14 * 0,065=0,2041
ticks=ticks_per_wheel_rotation(кол-во тиков на 1 оборот колеса)/red_ratio=2506/56=44,75

wheel_param = 44,75 * 56/0,2041 = ~12278
```

Установка параметров

Установка параметра **wheel_distance** возможна в launch-файле `rosserial.launch` в пакете turtlebro:

```
sudo nano ~/catkin_ws/src/turtlebro/launch/rosserial.launch

==============

<param name="wheel_distance" type="double" value="0.185"/>
```

Также в лаунч-файле `rosserial.launch` есть возможность задачи параметра **wheel_param**:

```
<param name="wheel_param" value="$(optenv ROVER_WHEEL_PARAM 22500)"/>
```

но как можно заметить, значение данного параметра берется из переменной окружения операционной системы, а конкретнее задается в файле `.ros_params`:

Настройка параметров в файле .ros_params

После изменения параметров необходимо выполнить на роботе команду синхронизации данных:

```
sudo sync
```

и путем выключения/включения тумблера питания перезагрузить всего робота.

В случае, когда пользователь не установил никаких параметров, происходит загрузка платы TurtleBoard с параметрами по умолчанию, которые обеспечивают работу идущих в комплекте с роботом моторов и колес. Процедура обновления параметров нужна только в том случае, когда плата TurtleBoard настраивается для работы с нестандартным шасси.

Загруженные параметры не сохраняются в постоянную память контроллера, поэтому при установке неправильных параметров необходимо произвести перезапуск устройства, чтобы загрузить значения по умолчанию.

Настройка ПИД параметров моторов

Для настройки коэффициентов ПИД регулятора можно использовать сервис ros `/set_pid`

```
rosservice call /set_pid "Ki: 0.0 Kp: 0.0 Kd: 0.0"
```

Файл turtlebro.launch

Главный файл конфигурации -- подключает другие .launch файлы и глобальные настройки.

```
<arg name="run_rosserial" default="true"/>
<arg name="run_rplidar" default="true"/>
<arg name="run_turtlebro_web" default="true"/>
<arg name="run_camera_ros" default="true"/>
<arg name="run_simple_odom" default="true"/>
```

`run_rosserial` -- запуск rosserial.launch для соединения с Arduino и STM МК\ `run_rplidar` -- запустить получение данные с RPLidar\ `run_turtlebro_web` -- запуск веб-интерфейса робота\ `run_camera_ros` -- включить камеру через пакет `uvc_camera` \ `run_simple_odom` -- подключить паблишер приведенной одометрии. Топик `/odom_pose2d`

Файл rosserial.launch

Файл запуска нод `stm_serial_node` и `arduino_serial_node` необходимых для работы с Arduino и STM

МК STM TurtleBoard

Подключается через устройство, скорость **460800** бод/с `/dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0012-if00-port0`

Arduino

Подключается через устройство, скорость **115200** бод/с `/dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0010-if00-port0`

Файл rplidar.launch

Файл для запуска ноды обработки данных с лидара. Для работы необходим пакет `rplidar` от производителя лидара. Параметры файла настроены для работы с лидаром RPLidar A1 на скорости 115200 бод/с через устройство `/dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0011-if00-port0`

Данные с лидара вычисляются относительно фрейма `base_scan`

Файл camera_ros.launch

Файл для запуска издателя с данными полученными из фронтальной камеры. Подробнее о работе с камерой

Работа с лидаром

Основной датчик робота - лазерный лидар Slamtec RpLidar A1

Изменения режимов работы лидара возможна в лаунч-файле `rplidar.launch` :

```
/home/pi/catkin_ws/src/turtlebro/launch/rplidar.launch
```

Лидары старой модели (с нижней платой под лидаром) могут работать в слудеющийх режимах:

```
Standard: max_distance: 12.0 m, Point number: 2.0K
Express: max_distance: 12.0 m, Point number: 4.0K
Boost: max_distance: 12.0 m, Point number: 8.0K
```

Новые лидары могу работать в следующих режимах:

```
Standard: max_distance: 12.0 m, Point number: 2.3K
Express: max_distance: 12.0 m, Point number: 4.0K
Boost: max_distance: 12.0 m, Point number: 7.9K
Sensitivity: max_distance: 12.0 m, Point number: 7.9K
Stability: max_distance: 12.0 m, Point number: 5.0K
```

По умолчанию, лидар включается в режиме `Boost` , в котором он выводит 720 точек в топик `/scan`

Выбор режима работы лидара зависит от условий отражений объектов и размера тестового полигона. В некоторых случаях требует изменение режима работы для более подходящих условий.

Работа с камерой (v0.10)

> Актуально для версии 0.10 и более новых

При включении робота автоматически включается камера `video0` и начинают публиковаться данные в топики камеры:

```
/front_camera/camera_info # Информация о камере
/front_camera/image_raw  # Данные в формате sensor_msgs/Image (.jpeg)
/front_camera/image_raw/compressed sensor_msgs/CompressedImage
```

Данные с камеры принимаются в формате `mjpeg` и без дополнительной обработки публикуются в топик `/front_camera/image_raw/compressed`. Для заполнения топика `/front_camera/image_raw` происходит перекодирование jpeg->raw с использованием пакета `image_transport`.

Если вы не используете данные `/front_camera/image_raw` рекомендуется отключить лишнее преобразование (аргумент `republish_raw`).

> Изменение настроек параметров камеры возможно в файле: `/home/pi/catkin_ws/src/turtlebro/launch/camera_ros.launch`

Визуализация данных

Веб-интерфейс

Для просмотра видео через топики ROS, можно использовать веб-интерфейс робота. При загрузке веб-интерфейса происходит поиск всех доступных видеоданных (топики с `sensor_msgs/CompressedImage`). Если такие топики найдены, то все они будет добавлены в веб-интерфейс для просмотра. При добавлении новых топиков, веб-интерфейс робота необходимо перезагрузить.

Rviz

Вы можете использовать rviz для отображения видеопотока камеры. При удаленной работе рекомендуем выбирать для просмотра сжатые видеоданные (топик `/front_camera/image_raw/compressed`)

rqt_image_view или image_view

Для просмотра видео можно использовать специальные Linux программы. Например http://wiki.ros.org/rqt_image_view или http://wiki.ros.org/image_view

Пакет uvc_camera

Пакет публикует сжатые данные `sensor_msgs/CompressedImage` в топик `front_camera/image_raw/compressed`

Официальная документация пакета http://wiki.ros.org/uvc_camera

Работа с камерой в OpenCV

Для работы с OpenCV данные из камеры ROS необходимо конвертировать из топиков в объект OpenCV

Использование RAW топика

Для создания объекта с OpenCV используется библиотеке CvBridge и метод `imgmsg_to_cv2`, где `image_msg` это сообщение из топика.

```
from cv_bridge import CvBridge
from sensor_msgs.msg import CompressedImage, Image

cvBridge = CvBridge()
cv_image = cvBridge.imgmsg_to_cv2(image_msg, "bgr8")
```

Использование Compressed топика

```
from cv_bridge import CvBridge
from sensor_msgs.msg import CompressedImage, Image

cvBridge = CvBridge()
cv_image = cvBridge.compressed_imgmsg_to_cv2(image_msg, desired_encoding='passthrough')
```

Важно понимать, что в случае использования compressed происходит дополнительное преобразование jpeg->raw. Но при этом происходит экономия сетевых ресурсов.

Прямой захват видео

Также вы можете отключить ноду ROS по захвату изображения с камеры и сделать захват данных из своего OpenCV приложения самостоятельно.

`cap = cv2.VideoCapture(0)`

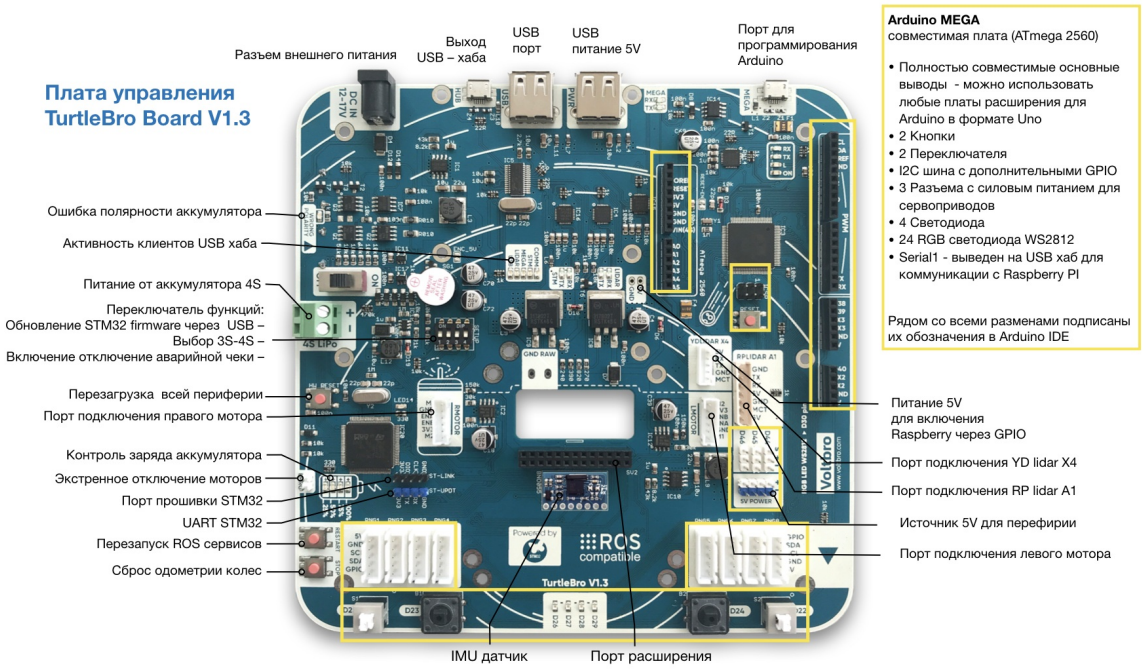
После этого удобно опубликовать данные в топик самостоятельно для просмотра через веб-интерфейс и тп.

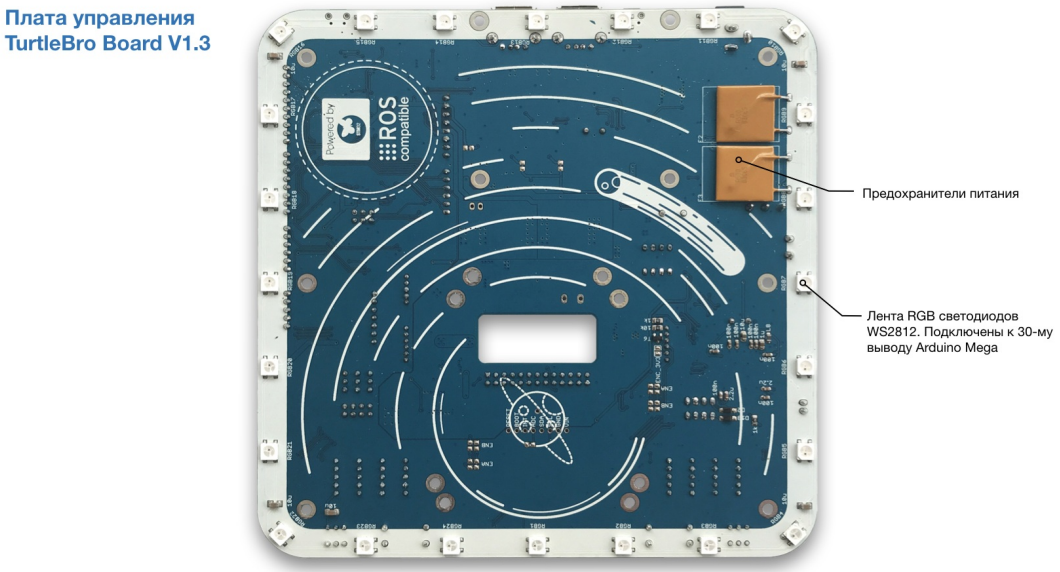
Далее производить с видео все необходимые манипуляции, и после этого, при необходимости, публиковать видео в топики.

Пример программы на pytnon, которая, используя `opencv`, следит за цветным мячиком и управляет роботом: ball_tracking.py

Системная плата

Схема платы





Назначение кнопок на плате

`restart` Нажатие кнопки перезапускает программу МК STM32. Происходит чтение rosparam, сбрасываются значения датчика IMU, одометрии, текущих скоростей колес. Кнопку необходимо использовать при изменении rosparams на стороне Raspberry. Перезапуск идет около 30 секунд.

`stop` При нажатии сбрасывается значение cmd_vel (робот останавливается), обнуляются значения одометрии и IMU датчика. Удобно пользоваться этой кнопкой для установки нулевого положения робота на полигоне.

`hw_reset` Кнопка сброса МК STM32. Осуществляет полный перезапуск робота, включая Raspberry.

`reset` Кнопка сброса Arduino. Осуществляет полный перезапуск Arduino.

`EMRG` Выходы для подключения чеки экстренной остановки моторов. Для работы в нормальном режиме контакты должны быть замкнуты.

Обновление микропрограммы с помощью USB-UART переходника

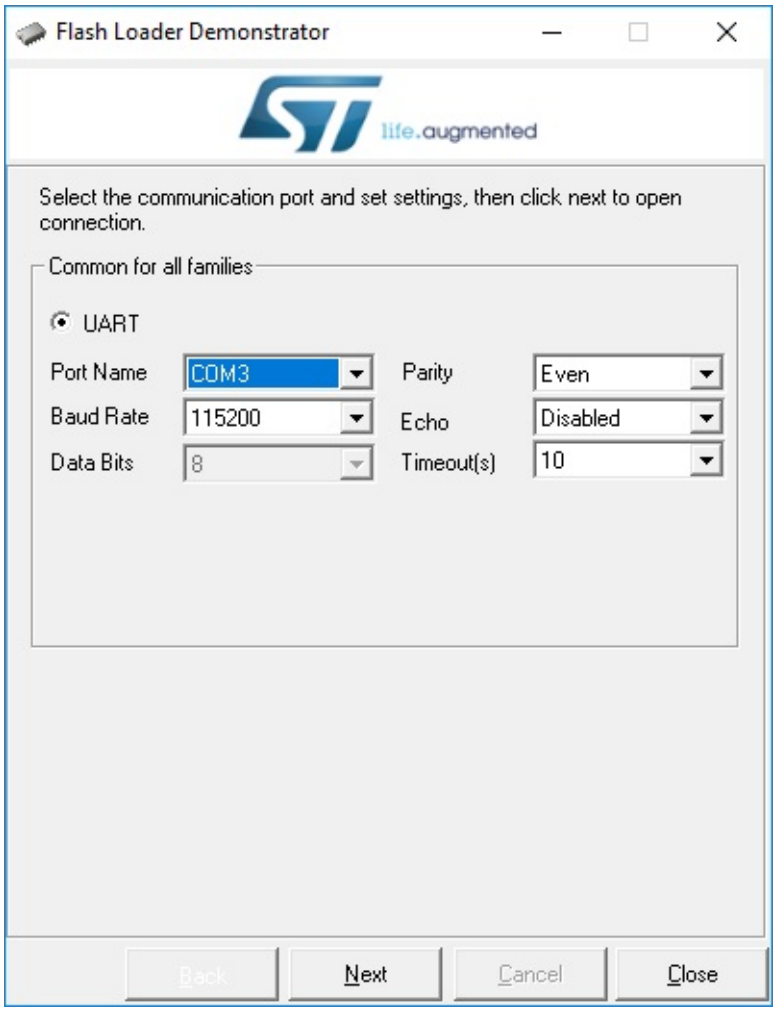
Для обновления микропрограммы системной платы TurtleBoard необходимо использовать программу STM32 Flash loader demonstrator.

Придерживайтесь следующей инструкции:

- Выключите питание робота и извлеките из него батарею.
- Подсоедините USB-UART переходник к разъему ST-UPDT голубого цвета на системной плате. Обратите внимание на то, что ножка TX переходника должна быть соединена с ножкой RX разъема ST-UPDT, а ножка RX - к TX.

Ножка 3V3 разъема ST-UPDT может быть подключена только к источнику напряжения 3.3В! Подключение ее к 5В может привести к повреждению платы!

- Переведите DIP-переключатель №4 в положение "ON"
- Подключите USB-UART к компьютеру, он должен определиться системой как COM порт.
- Нажмите кнопку HW_RESET на плате, это должно привести к тому, что все светодиоды погаснут (если ранее были включены) и плата перейдет в режим ожидания прошивки.
- Запустите программу STM32 Flash loader demonstrator, выберите COM порт своего UART преобразователя. Нажмите кнопку Next.

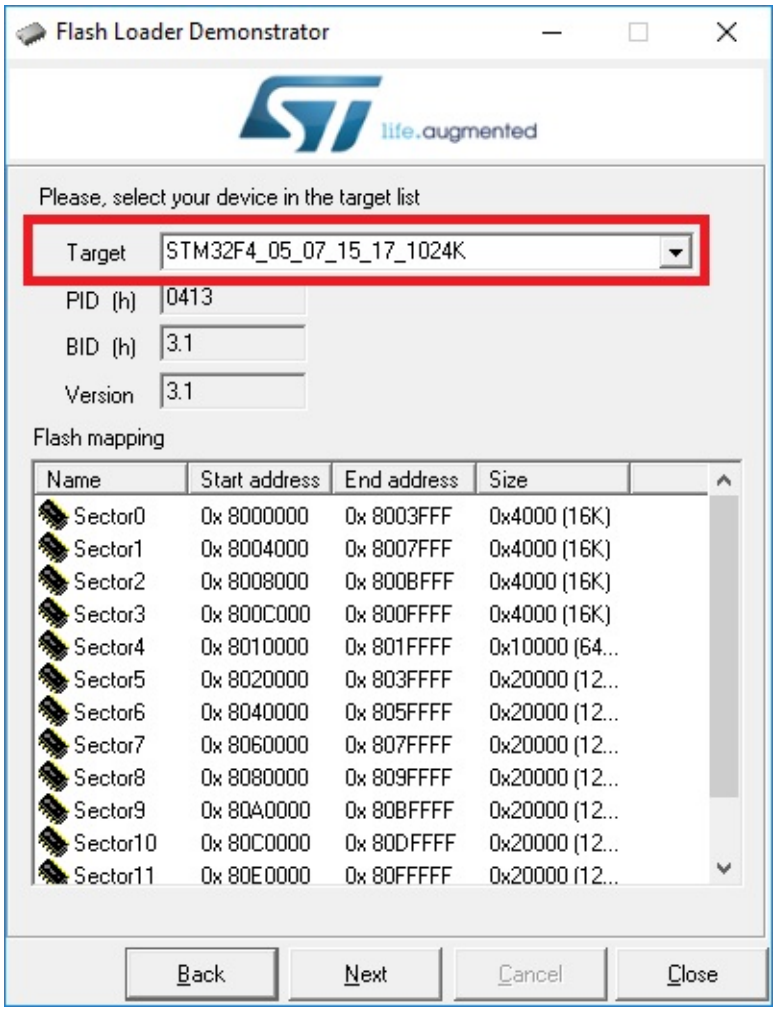


Вы должны увидеть сообщение:

```
Target is readable. Please click "Next" to proceed
```

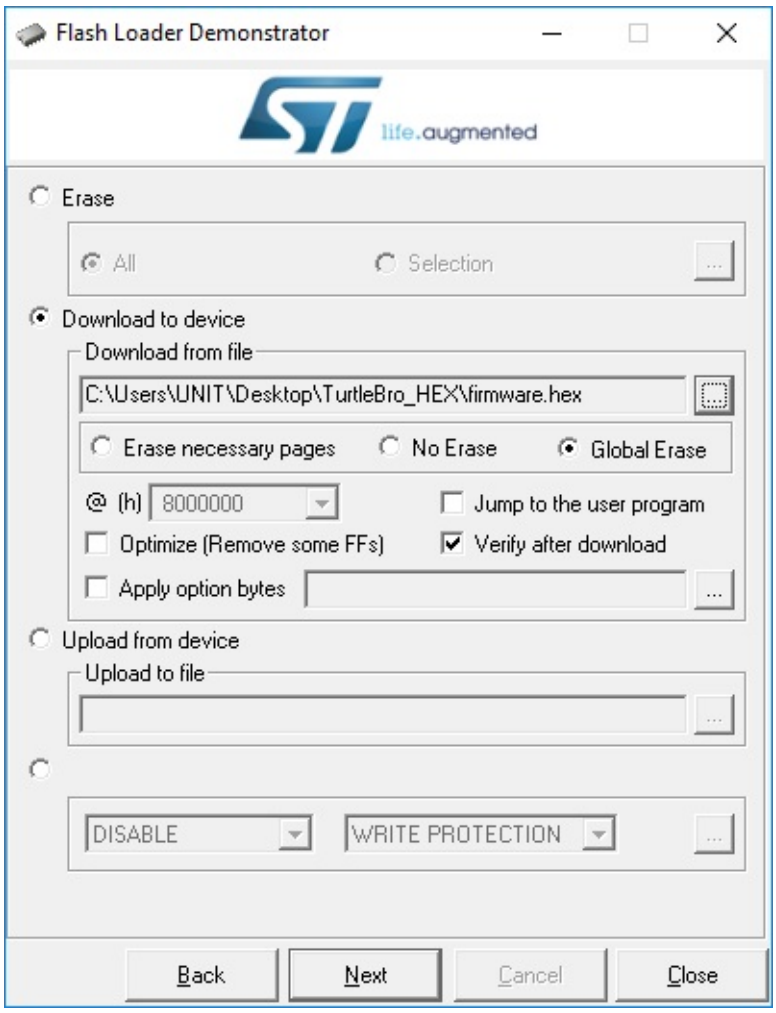
Если этого не произошло, проверьте правильность выполнения предыдущих шагов и попробуйте еще раз

- В следующем окне вы должны увидеть приблизительно следующее:



Обратите внимание на поле Target: его содержимое должно точно совпадает с картинкой выше.

- В следующем окне необходимо выставить настройки точно так же, как на картинке внизу:



В графе Download from file необходимо указать на файл прошивки, скачанный с нашего сайта. Обратите внимание на то, что по умолчанию всплывающее окно не отображает необходимые нам файлы с расширением .hex, поэтому в правом нижнем углу окошка необходимо выбрать тип файла hex Files (*.hex)

- После нажатия кнопки Next должен начаться процесс обновления микропрограммы. Он может занять несколько минут, пожалуйста, дождитесь его окончания. После окончания загрузки вы должны увидеть сообщение:

```
Download operation finished succesfully
```

- Закройте программу
- Отключите USB-UART переходник от компьютера
- Отключите USB-UART переходник от платы TurtleBoard
- Переведите DIP-переключатель №4 в положение Off

При следующей запуске робот будет использовать обновленную микропрограмму

Доступные топики

На плате запущены **Издатели** и **Подписчики**, взаимодействующие с ROS через библиотеку `rosserial`. STM32 и Arduino не имеют собственного интерфейса USB, поэтому, для того, чтобы они могли общаться с Raspberry, на плате TurtleBoard установлены UART-USB преобразователи и USB-hub, которые позволяют обеспечить коммуникацию с микрокомпьютером с помощью всего лишь одного провода USB.

Контроллер платы turtlebro STM32 публикует в ROS Raspberry следующие топики, поддерживающие стандартные для ROS типы сообщений:

Топик /bat

Содержит данные о состоянии подключенной батарейки к плате. Тип сообщения `sensor_msgs/BatteryState`

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 voltage
float32 current
float32 charge
float32 capacity
float32 design_capacity
float32 percentage
uint8 power_supply_status
uint8 power_supply_health
uint8 power_supply_technology
bool present
float32[] cell_voltage
string location
string serial_number
```

Топик /cmd_vel

Топик управления перемещения роботом. Тип сообщения `geometry_msgs/Twist` При публикации данных в топик робот начинает движение. Робот выполняет последнюю полученную команду до тех пор, пока не получит новые данные. Поэтому, например, для остановки робота необходимо передать "нулевые" значения скорости.

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

Топик /imu

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

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w
float64[9] orientation_covariance
geometry_msgs/Vector3 angular_velocity
  float64 x
  float64 y
  float64 z
float64[9] angular_velocity_covariance
geometry_msgs/Vector3 linear_acceleration
  float64 x
  float64 y
  float64 z
float64[9] linear_acceleration_covariance
```

Данные о ковариации не заполняются. Данные публикуются с частотой 20 Герц.

В связи с тем, что rviz по умолчанию некорректно отображает данные IMU, необходимо поставить соответствующий плагин для корректного отображения данных с IMU. Установка плагина осуществляется командой:

```
sudo apt install ros-noetic-rviz-imu-plugin
```

При добавлении визуализации данных в rviz необходимо выбрать тип сообщения rviz_imu_plugin->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`

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

Топик /odom_pose2d

Упрощенные данные одометрии в 2d пространстве

```
float64 x
float64 y
float64 theta
```

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` .

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

Топик /raw_odom

Данные полученные энкодеров колес. Время, счетчик "тиков" и угол датчика IMU

```
turtlebro/RawOdom

uint64 timestamp
int32 left_ticks
int32 right_ticks
float32 theta
```

Доступные сервисы

Через библиотеку rosserial доступны следующие сервисы для управления платой Turtlebro.

Сервис /reset

Сервис вызывает сброс одометрии (данные / `odom`) и текущих установленных скоростей (/ `cmd_vel`) робота. Робот останавливается и сбрасывает свое положение. Сервис вызывается без параметров.

> Если после сброса одометрии перенести робота с одного места на другое, то параметры одометрии не будут равны нулю, хотя колеса не вращались. Это происходит потому, что данные о положении робота рассчитываются в том числе по инерционному датчику IMU, который способен регистрировать ускорение робота.

Сервис /set_pid

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

```
float32 Ki
float32 Kp
float32 Kd
---
bool status
```

Сервис /board_info

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

```
mcu_id:
  data: "0025001D3148501020333044"
firmware_version:
  data: "0.1_c32cfe6"
```

Сервис /start_motor

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

Сервис /stop_motor

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

Работа с Arduino

На плате размещен микроконтроллер ATMega 2560 с обвязкой, идентичной плате Arduino Mega. Контроллер поставляется с прошитым бутлоадером Arduino. Таким образом, микроконтроллер полностью готов к запуску скетчей Arduino IDE и работе со стандартными шилдами для Ардуино.

Для работы с МК необходимо скачать и запустить Arduino IDE с сайта arduino.cc. В настройках IDE выбрать плату Arduino Mega 2560.

Библиотека Arduino ros_lib

Для работы с Arduino через ROS необходимо установить библиотеку ros_lib: `<ros.h>`.

Скачать библиотеку необходимо с вашего робота. Для этого можно воспользоваться утилитой копирования файлов между компьютерами SCP:

```
scp pi@turtlebroXX.local:/home/pi/ros_lib_noetic.tar.gz /home/<USERNAME>/
```

Вам необходимо распаковать архив, зайти в распакованную директорию и внутри нее должна находиться папка с названием `ros_lib`. Она должна содержать большое количество файлов, необходимых для компиляции программ содержащих вызовы `<ros.h>`.\ После этого необходимо скопировать папку `ros_lib` в папку `libraries`, которая находится внутри той директории, куда Arduino IDE сохраняет новые скетчи. Там же должны находиться и те библиотеки, которые вы загружали стандартным способом - через менеджер библиотек Arduino IDE https://www.arduino.cc/en/guide/libraries

Но если вы используете собственные сообщения или у вас появляются ошибки при сборке скетчей, вам необходимо "пересобрать" библиотеку `ros_lib` самостоятельно с помощью команды (выполнив ее на роботе)

```
rosrun rosserial_arduino make_libraries.py .
```

Вызванная утилита `rosserial_arduino` соберет новую библиотеку на основе настроек ROS вашего робота и положит ее в корневую директорию пользователя `/home/pi/`. Дальше вам надо переписать ros_lib с робота на ваш компьютер и поместить его в директорию библиотек Arduino в соответствии с инструкцией по установке библиотек для Arduino IDE.

Взаимодействие с ROS

Arduino Mega подключена к Raspberry через порт Serial1. Со стороны ROS запущен сервис `rosserial` который организует взаимодействие МК и ROS.

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, как показано ниже.

```
#include <ros.h>

class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};

ros::NodeHandle_<NewHardware>  nh;
```

Примеры можно посмотреть в официальной документации rosserial http://wiki.ros.org/rosserial_arduino/Tutorials

Дополнительные возможности Arduino

В передней части системной платы робота расположены две кнопки, подключенные к ножкам `D24` и `D23` МК Arduino и два переключателя, подключенные соответственно к `D22` и `D25`. При нажатии кнопки или переключателя на пине Arduino будет сигнал `HIGH` Для чтения значения необходимо использовать Arduino функцию `digitalRead(pin)`

С левой стороны системной платы находятся пины `D44 D45 D46`, которые можно использовать для подключения сервомашинок.

ВНИМАНИЕ: на сервомашинки подается напряжение 5В от отдельного источника питания! Ни в коем случае нельзя соединять пины питания сервомашинок с пинами питания, выведенными на колодку Ардуино.

Перед лидаром расположены 4 светодиода, подключенные к пинам `D26, D27, D28, D29.`

Общение с Arduino через Serial

Внимание! Для прямого общения с Arduino необходимо соединить usb-порт Raspberry и microusb порт контроллера кабелем.

\ В некоторых случаях необходимо получать данные от встроенного микроконтроллера без применения ROS. Для этого можно применять консольные утилиты типа minicom (https://linux.die.net/man/1/minicom).\ Для чтения данных от Ардуино можно применить следующую команду:

```
minicom -b 9600 -o -D /dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0
```

для завершения чтения следует нажать `Ctrl-A` и затем `X`. Скорость и параметры подключения следует указать такими же, как при инициализации Serial в скетче Arduino.\

Еще одна возможность для получения данных от Arduino это применении библиотеки Serial языка Python https://pyserial.readthedocs.io/en/latest/shortintro.html

Пример простого скрипта для чтения данных от Arduino:

```
#!/usr/bin/env python3

import serial
if __name__ == '__main__':
    ser = serial.Serial('/dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0', 9600, timeo
ut=1)
    ser.flush()
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            print(line)
```

Работа со светодиодной лентой

Под платой расположено 24 RGB светодиода модели `WS2812`. \ https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

`WS2812` - это три RGB-светодиода и микросхема-драйвер для управления этими светодиодами, собранные в одном SMD корпусе. Корпус каждого светодиода имеет 4 вывода: два вывода данных и два вывода питания. Выводы данных предыдущих светодиодов соединены со входами следующих, создавая цепочку светодиодов, управляемых через один пин микроконтроллера.

Лента подключена к пину `D30` встроенного контроллера Arduino. Число светодиодов - 24. Для управления светодиодами мы рекомендуем использовать библиотеку FastLed.\ https://github.com/FastLED/FastLED

Пример управления светодиодной лентой из Arduino-скетча:\ https://randomnerdtutorials.com/guide-for-ws2812b-addressable-rgb-led-strip-with-arduino/\ https://github.com/FastLED/FastLED/tree/master/examples

Удаленная загрузка скетча Arduino

Если есть необходимость удаленно (без доступа к роботу) обновить прошивку Arduino, то это возможно сделать имея только удаленный доступ.

Подготовить скетч к загрузке

1. В Arduino IDE выбрать МК Arduino/Genuino Mega or Mega 2560
2. Скомпилировать программу (кнопка Проверить)
3. В меню выбрать Скетч→Экспорт Бинарного файла
4. В директории где находиться файл скетча, будут созданы два файла с бинарными данными вида (sketch_mar24a.ino.mega.hex sketch_mar24a.ino.with_bootloader.mega.hex)
5. Мы должны использовать файл `sketch_mar24a.ino.mega.hex`

Загрузить бинарный файл на Arduino:

1. Скопировать файлы `.hex` на робота, например командой linux scp
2. На роботе выполнить команду "прошивки" платы Arduino

```
avrdude -v -v -p atmega2560 -c wiring -P /dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0 -b 115200 -D -U flash:w:sketch_mar24a.ino.mega.hex:i
```

Где `sketch_mar24a.ino.mega.hex` имя файла с прошивкой.\ \ Для возможности удаленной прошивки платы, необходимо чтобы Arduino разъем на плате Turtlebro был подключен к RaspberryPi через Micro-USB.