WHEELTEC STM32 Moving Chassis

Руководство пользователя

Содержание

Предисловие

Полный комплект учебных материалов для роботов с ROS включает «Руководство по разработке программ движения на контроллере STM32», «Конфигурация программ для Ubuntu, «Руководство по разработке в ROS».

Руководство «Конфигурация программ для Ubuntu» посвящено разработке программ Ubuntu для виртуальных машин и Raspberry Pi (Jetson Nano).

«Руководство по разработке в ROS» описывает процесс написания программ для ROS.

Данное руководство содержит кинематический анализ движения шасси робота с ROS, описание протоколов связи, режимов управления и т.д. Робот оснащен двумя контроллерами: Raspberry Pi (Jetson Nano, Jeston TX2, промышленный контроллер или др.) и STM32. Контроллеры связаны через последовательные порты (подробное описание схем подключения см. в главе 5).

Контроллер Raspberry Pi (Jetson Nano, Jeston TX2, Industrial Control и т.д.) для работы в ROS использует систему Ubuntu. Контроллер STM32 предназначен для управления подвижной тележкой робота, сбора информации с дистанциометра, батареи, IMU-сенсора и т.д.

Названия команд, функций и др. в данном руководстве относятся к встроенным программам робота. Если вы не планируете разбираться в программах контроллера STM32, мы предлагаем пропустить функции и программы, приведенные ниже (например, функция app_show (), отправка команды data_task, функция data_transition ()), так как они не влияют на понимание процесса разработки в целом.

Данный документ распространяется на шесть роботов серии образовательных роботов ROS:

two-wheeled differential car, Ackerman car, Mecanum wheel car, omni-wheel car, crawler car, and four-wheel drive car.

1 Режимы управления

В этой главе приводится подробное описание и назначение режимов управления роботом. Всего существует 6 режимов управления: 1) через приложение; 2) проводной джойстик PS2; 3) плата управления ROS; 4) дистанционное управление полетом; 5) управление CAN; 6) управление через последовательный порт. Режим управления отображается в левом нижнем углу OLED-дисплея. После включения питания по умолчанию устанавливается режим управления ROS.

Рисунок 1-1 Режимы управления

1.1 Единицы измерения скорости

Ниже приведено описание единиц измерения робота, m/s (м/с). Формула преобразования исходных данных измерения энкодеров в м/с:

MOTOR_A.Encoder=Encoder_A_pr*CONTROL_FREQUENCY*Wheel_perimeter/Encoder_Precision,

где Encoder_A_pr — исходные данные энкокодера, CONTROL_ FREQUENCY — частота управления (Гц), Encoder_Precision — точность энкодера (связана с конструкцией двигателя и платой энкодера), Wheel_perimeter — периметр колеса (м), Motor_A.Encoder — фактическая скорость робота (м/с). Преобразование данных с энкодеров двигателей B, C, D аналогично показанному примеру для энкодера двигателя A.

На рис. 1-1-1 показана калибровка скорости относительно положительных осей XYZ.

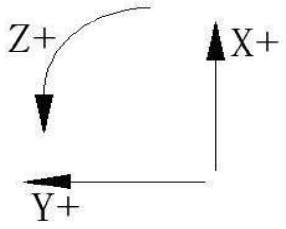


Рис. 1-1-1 Калибровка скорости относительно положительных осей XYZ

1.2 Управление через ROS (последовательный порт 3)

По умолчанию робот включается в режиме управления ROS. В этом разделе описан способ управления роботом в среде ROS. Отдельные принципы и понятия работы в среде ROS приведены в «Руководстве по разработке в ROS». Информацию о виртуальных машин и среде разработки ROS, а также об удаленной работе в системе Ubuntu см. в «Конфигурация программ для Ubuntu».

Далее описано использование клавиатуры виртуальной машины для удаленного управления движением робота.

1 Удаленный доступ к виртуальной машине Ubuntu

Обратите внимание, что использование виртуальной машины для управления движением робота требует подключения двух отдельных портов, каждый из которых имеет отдельный удаленный доступ к системе Ubuntu. Введите инструкции для виртуальной машины, показанные на рис. 1-2-1, чтобы удаленно войти в систему ubuntu.

Рисунок 1-2-1 ssh wheeltec@192.168.0.100

2 Набор команд

В первом порту включите функциональный блок робота «turn_on_wheeltec_robot». Блок «turn_on_wheeltec_robot» активирует узел управления роботом.

Рис. 1-2-2 roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch

Во втором порту включите узел управления движением робота через клавиатуру. Для этого включите узел управления клавиатурой keyboard_teleop в функциональном блоке «wheeltec_robot_rc».

Рис. 1-2-3 roslaunch wheeltec_robot_rc keyboard_teleop.launch

После включения узла управления через клавиатуру появится информация об управлении. Теперь можно использовать клавиатуру для управления движением робота, назначения клавиш показаны в табл. 1-1. Чтобы отключить узел управления, нажмите «crtl+c» или просто закройте порт.

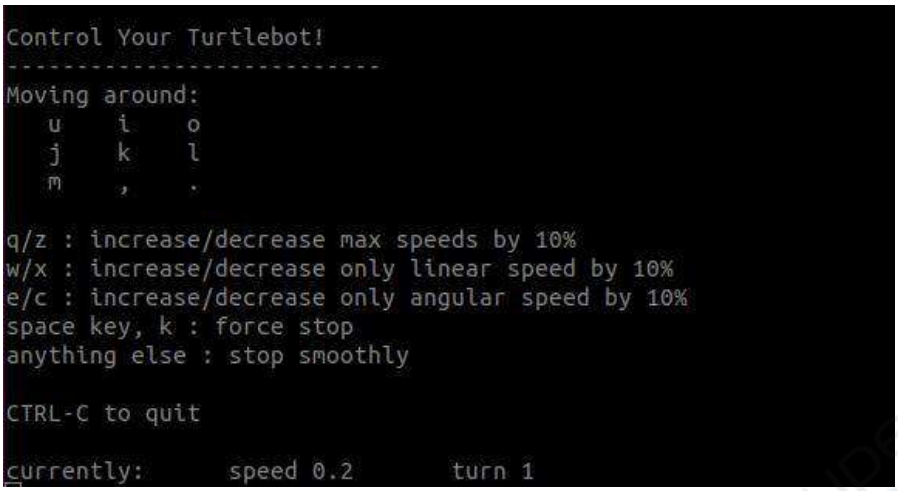


Рисунок 1-2-4 Информация о включенном узле управления через клавиатуру

Таблица 1-1 Указания к управлению движением робота через клавиатуру

Кнопка	**u**	**i**	**o**	**j**
Назначение	Движение вперед влево	Движение вперед	Движение вперед вправо	Поворот налево
Кнопка	**k**	**l**	**m**	**,**
Назначение	Экстренная остановка	Поворот направо	Движение назад влево	Движение назад
Кнопка	**.**	**Space**	**q**	**z**
Назначение	Движение назад вправо	Экстренная остановка	Скорость +10%	Скорость -10%
Кнопка	**w**	**x**	**e**	**c**
Назначение	Линейная скорость +10%	Линейная скорость -10%	Угловая скорость +10%	Угловая скорость -10%

3 Отправка данных с контроллера STM32 в ROS

Контроллеры STM32 (подвижная тележка робота) и ROS связаны между собой через последовательный порт. Контроллер STM32 использует последовательный порт 3 со скоростью передачи данных 115200. Протокол передачи данных: контроллер STM32 отправляет данные в ROS, ROS отправляет данные на контроллер STM32. Откройте исходные коды контроллера STM32. Коды связи последовательного порта находятся в файле usartx.c.

Контроллер STM32 отправляет данные в ROS с частотой 20 Гц при помощи функции date_task. Данные, которые передаются в ROS: заголовок и конец фрейма, используемый роботом флаговый разряд, скорость робота по осям XYZ, ускорение с IMU-

сенсора по осям XYZ, угловая скорость по осям XYZ, напряжение батареи, контрольный бит. Подробнее передаваемые данные описаны в табл. 1-2 .

Способ передачи данных: данные собираются в массив длиной 24 байта. Для побитовой отправки используется последовательный порт. Поскольку через последовательный порт за раз передаются только 8-бит (1 байт) данных, 2-байтные данные (тип short) разбиваются на старшие 8 битовов и младшие 8 битовов.

Функция data_transition() в файле usartx.c инициализирует данные перед их отправкой. Функция USART3_SEND() в файле usartx.c отправляет данные. Чтобы изменить данные перед отправкой, отредактируйте функцию data_transition(). Чтобы изменить длину данных, необходимо изменить длину массива SENT[], а также количество повторений цикла for в функции USART3_SEND() и размер принимаемых данных со стороны ROS.

Заголовок фрейма в отправляемых данных — фиксированное значение 0X7B, конец кадра — фиксированное значение 0X7D, flag_stop — флаг остановки двигателя (0 – включено; 1 - отключено). Контрольные биты проверяются методом BCC. Схема проверки контрольного бита показана на рис. 1-2-6.

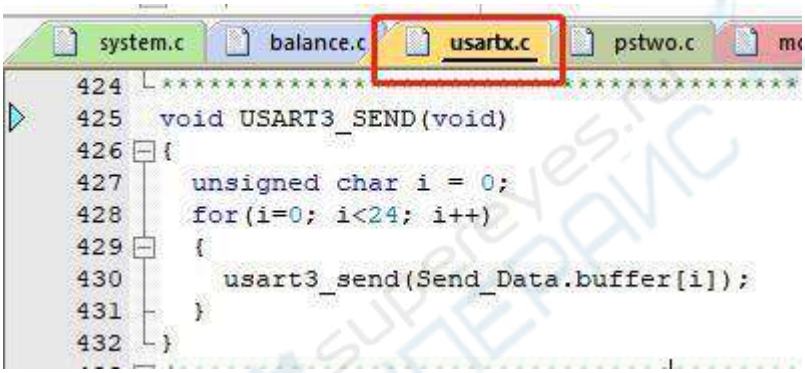


Рисунок 1-2-5 Функция USART3_SEND()

Рисунок 1-2-6 Схема проверки контрольного бита

Таблица 1-2 Данные отправляемые с контроллера STM32 в ROS

Содержимое данных	Заголовок фрейма (0X7B)		flag_stop		Скорость по оси X		Скорость оси Y		Скорость по оси Z	
Тип данных	Uint8		Uint8		short		short		short	
Используемые байты	1		1		2		2		2	
Номер массива	1		2		3	4	5	6	7	8
Содержимое данных	Ускорение по оси X		Ускорение по оси Y		Ускорение по оси Z		Угловая скорость по оси X		Угловая скорость по оси Y	
Тип данных	short		short		short		short		short	
Используемые байты	2		2		2		2		2	
Номер массива	9	10	11	12	13	14	15	16	17	18
Содержимое данных	Угловая скорость по оси Z		Напряжение батареи		Контрольный бит		Конец фрейма (0X7D)			
Тип данных	short		short		Uint8		Uint8			
Используемые байты	2		2		1		1			
Номер массива	19	20	21	22	23		24			

Следует обратить внимание, что данные о скорости робота по трем осям XYZ, данные с акселерометра, измерителя угловой скорости и напряжение батареи имеют тип с плавающей точкой (float). Так как данные с плавающей точкой неудобны для передачи через последовательный порт, четыре перечисленных вида данных перед отправкой умножают на 1000 (чтобы сохранить три знака после запятой), затем преобразуют в тип short и разделяют на 8 старших и 8 младших битов для отправки. Обратным образом после получения данных хостом, старшие и младшие 8 битовов снова объединяют в данные типа short и уменьшают в тысячу раз.

Ниже показано, как преобразовать два набора 8-битных данных в тип short, чтобы получить фактическую скорость и другие параметры: единицы измерения контрольной величин: мм/с (0.001 м/с). Знак контрольной величины (направление скорости) определяется битом наивысшего разряда старших 8 битов.

Пример 1: 21 B6 = 0010 0001 1011 011. Бит наивысшего разряда 0, положительное число, величина скорости 21 B6 = (2*16 + 1)*256 + (B*16 + 6) = (2*16 + 1)*256 + (11*16 + + 6) = 8630 (мм/с).

Пример 2: A1 2F = 1010 0001 0010 1111. Бит наивысшего разряда 1, отрицательное число. Величина скорости 2^16(FF FF + 1) - A1 2F = 5E D0 + 1 = (5*16 + + E)*256 + (D*16 + 0) + 1 = 24272 (мм/с).

На скриншоте ниже показаны фактические данные, полученные при передаче с последовательного порта 3 с помощью приложения с SerialPro.

(Обратите внимание, что последовательный порт 3 не интегрирован в чип CH340, поэтому для передачи данных между ROS и STM32 требуется специальный кабель и приложение SerialPro).

Рисунок 1-2-7 Отправка данных с последовательного порта 3 робота

Далее показано преобразование переданных и полученных 24 байтов информации.

1-й байт: 0X7B, заголовок фрейма.

2-й байт: 0X00, двигатель находится в движении.

3-й, 4-й байты: скорость по оси X, старшие 8 битов 0X01 (шестнадцатеричная система) = 0000 0001 (двоичная система), младшие 8 битов: 0X01 (шестнадцатеричная система) = 0000 0001 (двоичная система). Бит наивысшего разряда — 0, положительное

число (движение вперед). Величина скорости по оси X: 1*256 + 1 = 257 (мм/с) — фактическая скорость, измеренная энкодером робота-автомобиля.

5-й, 6-й байты: скорость по оси Y, старшие 8 битов 0X00 (шестнадцатеричная система) = 0000 0000 (двоичная система), младшие 8 битов: 0X01 (шестнадцатеричная система) = 0000 0001 (двоичная система). Бит наивысшего разряда — 0, положительное число (сдвиг влево). Величина скорости по оси Y: 0*256 + 1 = 1 (мм/с) — фактическая скорость, измеренная энкодером робота-автомобиля.

7-й, 8-й байты: скорость по оси Z, старшие 8 битов 0X00 (шестнадцатеричная система) = 0000 0000 (двоичная система), младшие 8 битов: 0X00 (шестнадцатеричная система) = 0000 0000 (двоичная система). Бит наивысшего разряда — 0, положительное число (вращение против часовой стрелки). Величина скорости по оси Z 0*256 + 0 = 0 (0.001 рад/с) — фактическая скорость, измеренная энкодером робота-автомобиля.

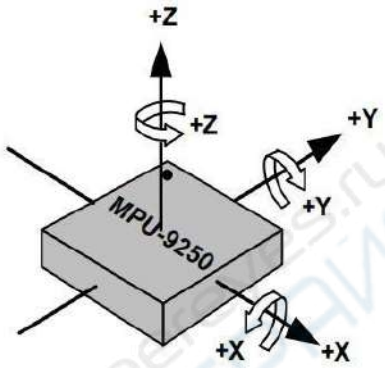


Рисунок 1-2-8 Акселерометр MPU9250, диаграмма угловой скорости по трем осям

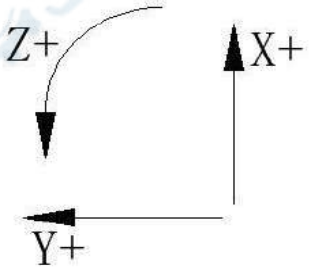


Рисунок 1-2-9 Калибровка робота по осям XYZ

Следующие 12 байтов — данные с трехосного акселерометра и измерителя угловой скорости. Обратите внимание, как изменились направления положительных осей XYZ: положительная ось X направлена вправо, положительная ось Y — вперед, а положительная ось Z — вверх. Данные акселерометра и измерителя угловой скорости — это угловые скорости по осям XYZ (см. рис. 1-2-8). Поскольку направления осей XY гироскопа отличаются от направления осей XY робота, данные об угловой скорости по осям и вокруг осей XY дополнительно корректируются при отправке.

9-й, 10-й байты: ускорение по оси X, старшие 8 битов 0XFE (шестнадцатеричная система) = 1111 1110 (двоичная система), младшие 8 битов: 0X96 (шестнадцатеричная система) = 1001 0110 (двоичная система). Бит наивысшего разряда — 1, отрицательное число, величина 2^16(FF FF+1) – FE 96 = 01 69=1*256 + 105 = 361. Преобразование в ускорение по оси X: 361/1672 = 0.2159 (м/с^2).

11-й, 12-й байты: ускорение по оси Y, старшие 8 битов 0XFD (шестнадцатеричная система) = 11111101 (двоичная система), младшие 8 битов: 0XCE (шестнадцатеричная система) = 1100 1110 (двоичная система). Бит наивысшего разряда — 1, отрицательное число, величина 2^16(FF FF+1) – FD CE = 02 3 1= 2*256 + 49 = 561. Преобразование в ускорение по оси Y: 125/1672 = 0.3355 (м/с^2).

13-й, 14-й байты: ускорение по оси Z, старшие 8 битов 0X40 (шестнадцатеричная система) = 01000000 (двоичная система), младшие 8 битов: 0X80 (шестнадцатеричная система) = 1000 0000 (двоичная система). Бит наивысшего разряда — 1, отрицательное число, величина 64*256 + 128 = 16512. Преобразование в ускорение по оси Z: 15684/1672 = 9.8756 (м/с^2).

15-й, 16-й байты: угловая скорость по оси X, старшие 8 битов 0XFF (шестнадцатеричная система) = 11111111 (двоичная система), младшие 8 битов 0XFB (шестнадцатеричная система) = 1111 1011 (двоичная система). Бит наивысшего разряда 1, отрицательное число. Величина 2^16 (FF FF + 1) – FF FB = 00 04 = 0*256 + 4 = 4. Преобразование в угловую скорость по оси X: 9/3753 = 0.0011 (рад/с).

17-й, 18-й байты: угловая скорость по оси Y, старшие 8 битов 0X00 (шестнадцатеричная система) = 00000000 (двоичная система), младшие 8 битов 0X07 (шестнадцатеричная система) = 00000111 (двоичная система). Бит наивысшего разряда 0, положительное число. Величина 0*256 + 7 = 7. Преобразование в угловую скорость по оси Y: 7/3753 = 0.0019 (рад/с).

19-й, 20-й байты: угловая скорость по оси Z, старшие 8 битов 0X00 (шестнадцатеричная система) = 0000000 (двоичная система), младшие 8 битов 0X01 (шестнадцатеричная система) = 00000001 (двоичная система). Бит наивысшего разряда 0, положительное число, величина 0*256 + 1 = 1. Преобразование в угловую скорость по оси Z: 1/3753 = 0.0003 (рад/с).

21-й, 22-й байт: напряжение батареи, старшие 8 битов 0X58 (шестнадцатеричная система) = 0101 1000 (двоичная система), младшие 8 битов 0X38 (шестнадцатеричная система) = 0011 1000 (двоичная система). Бит наивысшего разряда — 0, положительное число. Величина 88*256 + 56 = 22584, напряжение 22584 мВ (милливольты).

23-й байт: контрольный бит BBC («исключающие или» предыдущих 22 байтов): 0X83=0X7B^0X00^0X01^0X01^0X00^0X01^0X00^0X00^0XFE^0X96^0XFD^0XCE^0X40^ 0X80^0XFF^0XFB^0X00^0X07^0X00^0X01^0X58^0X38.

24-й байт: 0X7D, конец фрейма.

(Чтобы проверить результат контрольного бита, можно использовать специальные веб-платформы).

4 Прием данных из ROS контроллером STM32

Плата контроллера STM32 оснащена чипами CH340 (последовательный порт 1) и CP210 (последовательный порт 3). Процедуры обработки данных одинаковы для обоих

портов. Для передачи данных по умолчанию используются чип CP2102 (порт 3) и последовательный порт ROS. Ниже показан пример приема данных последовательным портом 3.

При приеме данных используются прерывания. Принимаемые данные включают сигнал о модели робота, флаг запуска управления, целевую скорость робота по трем осям, контрольный бит данных.

Заголовок и конец кадра по умолчанию являются фиксированными значениями. Flag_stop — бит запуска управления роботом, отправляется при запуске по умолчанию.

Целевая скорость робота по трем осям используется для управления движением робота. Подробная информация о принимаемых данных представлена в таблице 1-3. Обратите внимание, что номер массива в таблице — номер массива данных, отправленных хостов.

Таблица 1-3 Прием данных из ROS контроллером STM32

Содержимое данных	Заголовок фрейма (фикс. значение 0X7B)		Зарезервиро-ванный бит		Зарезервиро-ванный бит	Целевая скорость по оси X	
Тип данных	Uint8		Uint8		Uint8	short	
Используемые байты	1		1		1	2	
Номер массива	1		2		3	4	5
Содержимое данных	Целевая скорость по оси Y		Целевая скорость по оси Z		Контрольный бит	Конец фрейма (фикс. значение 0X7D)	
Тип данных	short		short		Uint8	Uint8	
Используемые байты	2		1		1	1	
Номер массива	6	7	8	9	10	11	

Примечание: дифференциальный робот и тележка Ackermann не поддерживают управление движением по оси Y, в отличие от робота Mecanum Wheel и всенаправленной тележки.

В режимах управления приоритет управления ROS самый высокий. Если для приема данных в качестве последовательного порта контроллера STM32 выбирается порт 3, робот принудительно включает режим управления ROS.

Чтобы исключить отправку бесполезных данных во время включения робота, в настройках выставляется «Не отправлять данные первые 10 секунд». Данные будут отправлять только спустя 10 секунд после включения. Сперва проверяется заголовок фрейма, после чего запускается процесс приема данных. По окончании приема данных повторно проверяется конец фрейма, совпадает ли контрольный бит с разрешенными

значениями. Подробнее о функции прерывания последовательного порта usart3_IRQHandler() см. в файле usartx.c.

1.3 Управление через приложение

Робот поддерживает управление через приложение с подключением к Bluetooth и Wi-Fi. В этом режиме для управления движением робота используется джойстик. В приложении можно задать скорость робота в мм/с, ускорить или замедлить движение робота, увеличить/уменьшить скорость на 100 мм/с кнопками джойстика.

При управлении с приложения робот отправляет данные на смартфон через Bluetooth (приложение поддерживает подключение к WI-FI и Bluetooth одновременно). Статус отправки данных отображается на панели Debug, а содержимое данных можно просмотреть в файле show.c, в коде функции app_show(). Принцип управления с приложения заключается в управлении через последовательный порт и Bluetooth (или Wi-Fi). Приложение использует последовательный порт 2 со скоростью передачи данных 9600. Команды управления поступают с приложения на последовательный порт 2 с помощью функции прерывания.

1 Онлайн настройки

Установите последнюю версию приложения MiniBalance на смартфон Android. Соответствующие видеоуроки к данному руководству помогут настроить параметры и управлять роботом в режиме онлайн. Во вкладке «Настройки» можно нажать на «Параметр X» и изменить его свойства (см. рис. 1-3-1). Кроме того, перед настройкой параметров PID-регулятора необходимо нажать «Получить параметры устройства» (кнопка в верхнем правом углу), чтобы обновить параметры PID-регулятора в приложении. При перемещении бегунков приложение автоматически обновляет параметры и передает их роботу.

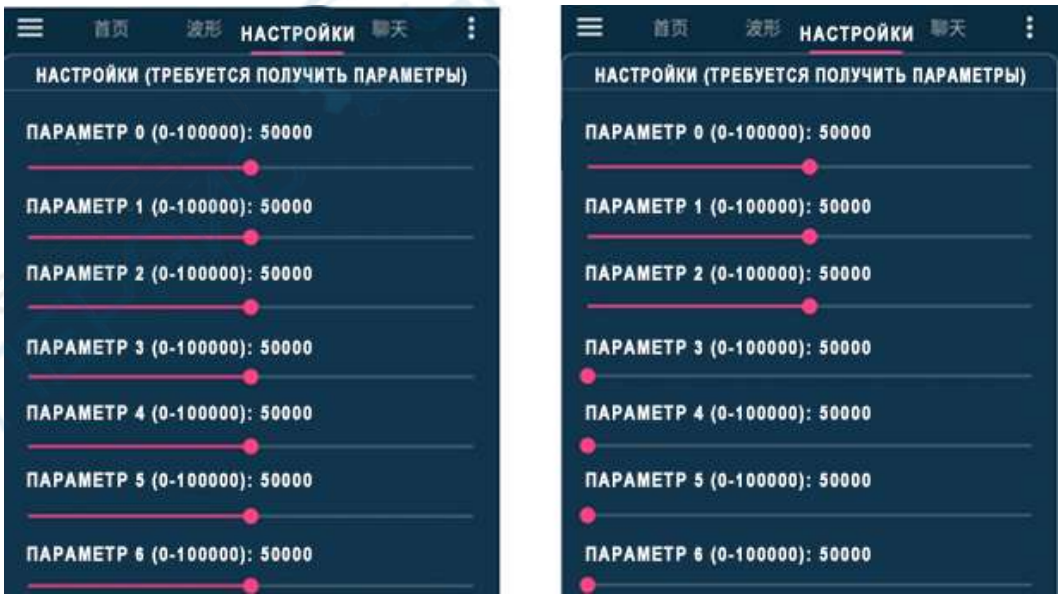


Рисунок 1-3-1 Интерфейс настроек: по умолчанию и после получения параметров

Параметр 0 — скорость робота.

Параметры 1, 2 — параметры PI-регулятора скорости робота.

Обратите внимание, если текущая скорость в параметрах управления составляет 40, для настройки скорости на 150 (за пределами текущего диапазона 0-40) сперва следует установить диапазон 0-80, затем снова нажать кнопку «Получить параметры устройства», после чего диапазон изменится на 0-160. Конечный диапазон скоростей робота будет 20-200. Подробное описание диапазонов и единиц измерения скорости робота см. глава 3, раздел «Единицы измерения скорости робота». Имена параметров задаются пользователем самостоятельно.

2. Управление роботом через приложение

Рисунки 1-3-3 Интерфейс приложения

Практически при каждом действии в приложении роботу отправляются различные команды (переключение между интерфейсами и режимами управления также отображается

в отправке команд). После приема команд робот выполняет соответствующие действия. В табл. 4-1 представлены возможные манипуляции с сенсором в интерфейсе приложения и соответствующие им команды.

Таблица 1-4 Описание команд в интерфейсе приложения

Сенсор	↑	↗	→	↘
Данные, полученные роботом	0x41	0x42	0x43	0x44
Действие робота	Движение вперед	Движение вперед вправо	Поворот на месте направо	Движение назад вправо
Сенсор	↓	↙	←	↖
Данные, полученные роботом	0x45	0x46	0x47	0x48
Действие робота	Движение назад	Движение назад влево	Поворот на месте налево	Движение вперед влево

Кнопка	Сила тяжести	Сенсор	Кнопка	Замедление	Ускорение
Данные, полученные роботом	0x49	0X4a	0x4b	0x59	0x58

Примечание: после подключения смартфона к Bluetooth, в приложении нужно надавить на сенсор в течение 0.5 с, как бы нажимая «вперед». Только после этого активируется управление роботом.

1.4 Управление с джойстика PS2

Режим PS2 использует беспроводной джойстик PS2. Алгоритм управления PS2: перед включением питания робота сперва подключите джойстик, затем включите питание, должен загореться красный светодиодный индикатор на джойстике PS2 (нормальная работа). Если индикатор не горит, нажмите кнопку «START» над индикатором, чтобы войти в режим PS2. В левом нижнем углу дисплея появится надпись «PS2».

В режиме PS2 используйте левый рычажок джойстика для движения робота вперед и назад. Для вращения робота вправо и влево используется правый рычажок (при управлении всенаправленной тележкой левый рычажок управляет движением робота в пространстве, а правый рычажок задает вращение тележки). Такая концепция управления позволяет избежать ошибок, который мог бы допустить пользователь, если бы движение вперед-назад и вправо-влево осуществлялось с помощью одного рычажка. Две кнопки в верхнем левом углу — кнопки ускорения и замедления.

Примечание: сперва подключите джойстик PS2 и только после этого включите питание. В противном случае джойстик может сгореть, что дополнительно нарушит работу двигателей (они начнут беспорядочно вращаться). После включения питания необходимо нажать кнопку «START» на джойстике PS2, чтобы войти в режим PS2.

Рисунок 1-4-1 Внешний вид джойстика PS2

1.5 Радиоуправление

Внимание: тележка Ackermann не поддерживает радиоуправление.

Алгоритм корректного радиоуправления: сперва подключите к роботу приемник дистанционного управления, включите пульт-контроллер радиоуправления, затем включите питание робота. Когда загорится индикатор приемника дистанционного управления, в нижнем левом углу дисплея появится надпись «R-C».

Режим радиоуправления: левый рычажок контроллера радиоуправления отвечает за движение робота вперед и назад, правый рычажок — за вращение вправо и влево (при управлении всенаправленной тележкой левый рычажок управляет движением робота по восьми направлениям в пространстве, а правый рычажок задает вращение на месте). Также при перемещении правого рычажка назад и вперед можно регулировать скорость робота, по аналогии с педалью газа (тележка Ackermann не поддерживает эту функцию). Тумблер «SWC» в правом верхнем углу контроллера радиоуправления переключает режимы скорости: нормальный/замедленный. По окончании работы сперва отключите питание робота и только затем отключайте контроллер радиоуправления.

Рисунок 1-5-1 Пульт-контроллер радиоуправления

Обратите внимание, что каналы контроллера радиоуправления должны быть настроены так, как показано на рис. 1-5-2. Самый левый переключатель должен находиться посередине, остальные четыре переключателя — в нижнем положении. Эти каналы регулируют ориентацию управлению.

Рисунок 1-5-2 Каналы контроллера радиоуправления

Далее объясняется, как подключить приемник дистанционного управления к плате расширения.

Рисунок 1-5-3 Приемник дистанционного управления

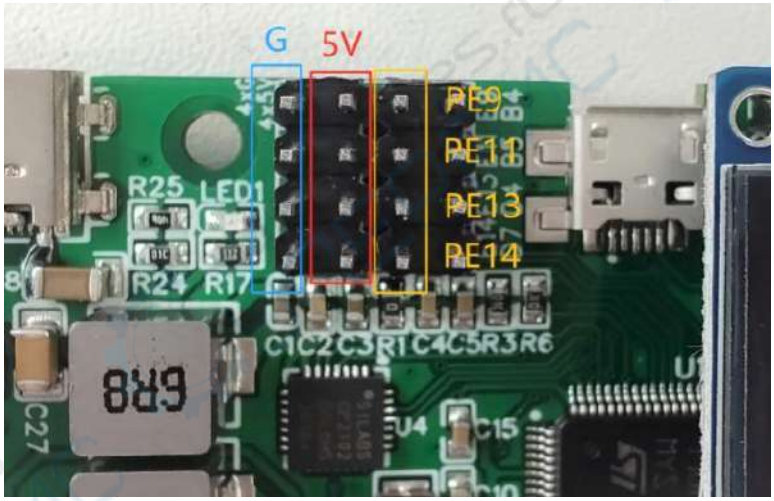


Рисунок 1-5-4 Интерфейсы радиоуправления на плате расширения

Таблица 1-5 Соответствующие пины контроллера радиоуправления и платы

Приемник радиоуправления	GND	5V	CH1	CH2	CH3	CH4
Плата расширения	G	5V	PE9	PE11	PE13	PE14

Как показано на рис. 1-5-3 и 1-5-4, а также в табл. 1-5, приемник радиоуправления имеет три ряда пинов: GND, 5V и линию сигнала. При использовании робота сперва подключают пары GND-GND и 5V-5V. Затем каналы сигнальной линии CH1, CH2, CH3, CH4 по очереди подключаются к пинам PE9, PE11, PE13, PE14 на плате расширения. Пин

PE14 используется только для всенаправленной тележки, для управления движением влево и вправо.

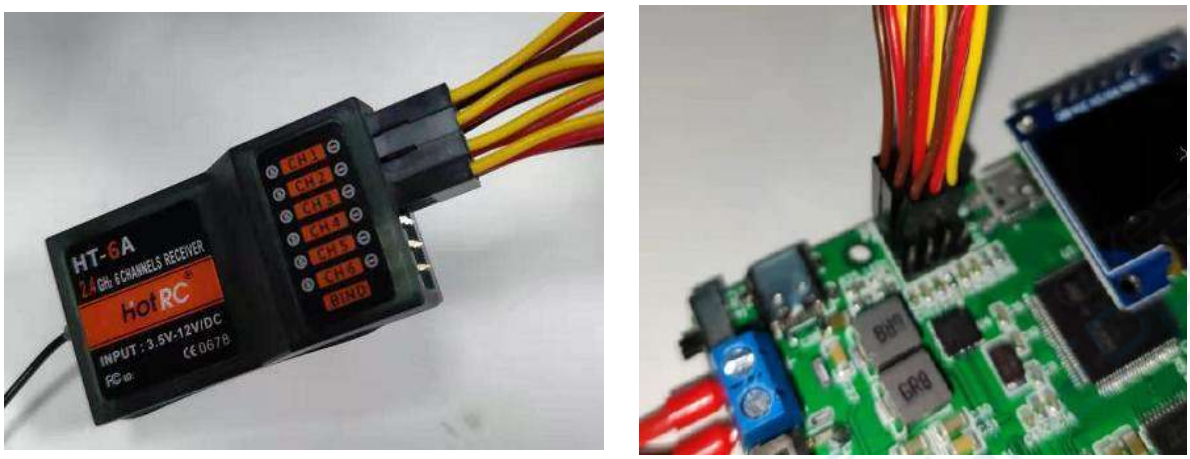


Рисунок 1-5-5 Пример подключения контроллера радиоуправления

1.6 CAN-управление

Роботы поддерживают протокол CAN. Способы связи по протоколу CAN: CANL-CANL, CANH-CANH.

Чтобы тележка получала команды управления по протоколу CAN, сначала необходимо отправить команду запуска на протокол CAN, скорость передачи 1M.

Формат команды запуска:
Идентификатор: 0X121
Тип фрейма: стандартный фрейм
Формат фрейма: DATA
DLC: 8 байтов

Таблица 1-6 Команда запуска CAN

Поле данных	tx[0]	tx[1]	tx[2]	tx[3]	tx[4]	tx[5]	tx[6]	tx[7]
Содержимое	10	12	15	19	24	30	37	Flag

Примечание: данные в таблице представлены в десятичной системе.

Когда flag=1, включается управление CAN, контроллер больше не должен получать команды запуска других режимов управления. После включения режима CAN в нижнем углу дисплея появится надпись «CAN».

Описание команд управления:
Идентификатор: 0X121
Тип фрейма: стандартный фрейм
Формат фрейма: DATA
DLC: 8 байтов

Таблица 1-7 Команды, принимаемые роботом по протоколу CAN

Поле данных	rx[0]	rx[1]	rx[2]	rx[3]
Содержимое	Старшие 8 битов контрольной величины направления X	Младшие 8 битов контрольной величины направления X	Старшие 8 битов контрольной величины направления Y	Младшие 8 битов контрольной величины направления Y
Поле данных	rx[4]	rx[5]	rx[6]	rx[7]
Содержимое	Старшие 8 битов контрольной величины направления Z	Младшие 8 битов контрольной величины направления Z	Зарезервированный бит	Зарезервированный бит

Примечание: так как дифференциальная тележка и тележка Ackermann не поддерживают управление по оси Y, величина управления по оси Y для этих роботов по умолчанию равна 0.

Rx [6], rx [7] — зарезервированные биты данных, позволяющие пользователю добавить собственные данные для передачи. Протокол CAN автоматически проверяется методом BBC, поэтому контрольный бит отсутствует.

При получении команд по протоколу CAN робот может одновременно отправлять собственные данные. Функция CAN_SEND() в файле [usartx.c] выполняет отправку данных на протокол CAN, задача отправки данных и данные для отправки настроены по умолчанию. Отредактировать содержимое данных для отправки можно только в модуле Send_Data.buffer[i] функции CAN_SEND.

Так как отправляемые данные содержат 24 байта, они отправляются тремя наборами по 8 байтов. Микроконтроллер, принимающий данные, может установить номер текущего набора данных с помощью идентификатора. Так 1-й набор данных (8 байтов) имеет идентификатор 0X101, 2-й набор — 0X102, 3-й набор — 0X103.

Содержимое данных, отправляемых с CAN на последовательный порт 1, совпадает с данными, отправляемыми с последовательного порта 3 в ROS (см. Табл. 1.2 «Управление через ROS (последовательный порт 3)»).

В программе отправки CAN предусмотрена функция прерывания для приема данных, отправленных роботом на протокол CAN. Данная программа установлена на микроконтроллеры модели с STM32F103RC или той же серии. При этом правильная передача данных между роботом и протоколом CAN возможна только после преобразования данных, для чего требуется микросхема VP230. Ниже показана схема подключения VP230: пины D, R микросхемы VP230 подключаются к пинам PD1, PD2 микроконтроллера. Затем пины CANH и CANL микросхемы подключаются к пинам CANH и CANL робота (H-H, L-L).

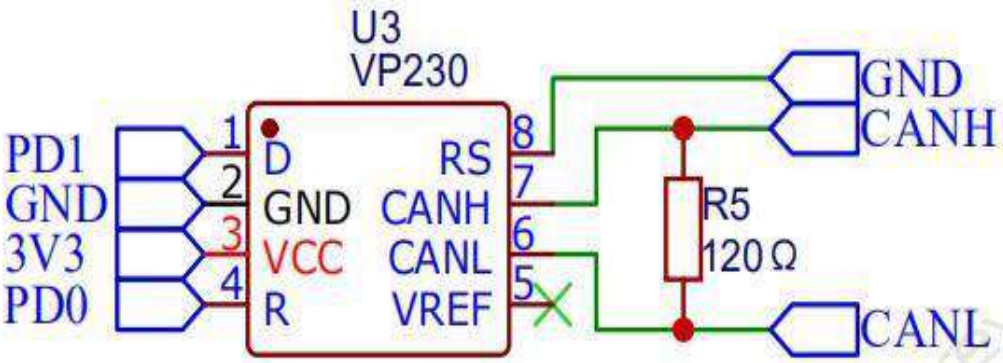


Рисунок 1-6-1 Подключение микросхемы VP230 для преобразования данных между роботом и микроконтроллером

Чтобы проверить корректность данных, полученных с протокола CAN, можно подключиться к хосту через последовательный порт 1 микроконтроллера. Контроллер получает информацию с робота через протокол CAN, которая затем отправляется на хост через последовательный порт 1, скорость передачи 9600.

Примечание: хотя представленные программы обмена данными между CAN и последовательным портом не являются исполняющими программами робота, пользователю не требуется загружать программу для последовательного порта. В роботы по умолчанию вшита нужная программа, все готово для работы. Данные программы обмена данными между CAN и последовательным портом могут быть запущены на другом микроконтроллере и использованы для связи с роботом.

1.7 Управление через последовательный порт 1

Контроллер STM32 оснащен последовательным портом 1 (CP210) и последовательным портом 3 (CP210). Процессы приема, отправки и обработки данных (содержимое данных совпадает) для обоих портов одинаковы. По умолчанию используется передача данных между последовательным портом 3 и ROS. При необходимости использовать последовательный порт 1, см. раздел 1.2 «Управление через ROS (последовательный порт 3)».

При передаче данных через последовательный порт программы необходимо согласовать скорости передачи данных обеих сторон, программно установленная скорость передачи 115200. После получения данных с последовательного порта 1 робот перейдет в режим управления через последовательный порт. В левом нижнем углу дисплея появится надпись «USART».

Таблица 1-8 Команды, получаемые роботом с последовательного порта

Поле данных	Заголовок фрейма	tx[0]	tx[1]	tx[2]	tx[3]
Содержимое	0X7B	Зарезервированный бит	Зарезервированный бит	Старшие 8 битов контрольной величины направления X	Младшие 8 битов контрольной величины направления X

Поле данных	tx[4]	tx[5]	tx[6]	tx[7]	tx[8]	Конец фрейма
Содержимое	Старшие 8 битов контрольной величины направления Y	Младшие 8 битов контрольной величины направления Y	Старшие 8 битов контрольной величины направления Z	Младшие 8 битов контрольной величины направления Z	BBC проверка контрольного бита	OX7D

Примечание: дифференциальная тележка, тележка Ackermann, гусеничный робот-автомобиль и робот с полным приводом (4WD) не поддерживают управление движением по оси Y в отличие от всенаправленной тележки и робота Mecanum Wheel.

Загрузите представленную программу отправки инструкций в другой микроконтроллер, подключите передачу данных между контроллером и роботом через последовательный порт. Робот автоматические перейдет в режим управления через последовательный порт.

Команды управления, отправляемые на робот через последовательный порт:

```
usart2_send(0X7B); // Заголовок фрейма
usart2_send(0X00); // Зарезервированный бит
usart2_send(0X00); // Зарезервированный бит
usart2_send(0X01); // Старшие 8 битов контрольной величины направления X
usart2_send(0X10); // Младшие 8 битов контрольной величины направления X
usart2_send(0X00); // Старшие 8 битов контрольной величины направления Y
usart2_send(0X00); // Младшие 8 битов контрольной величины направления Y
usart2_send(0X00); // Старшие 8 битов контрольной величины направления Z
usart2_send(0X00); // Младшие 8 битов контрольной величины направления Z
usart2_send(0X6A); // BBC проверка контрольного бита
usart2_send(0X7D); // Конец фрейма
```

Аналогичным образом можно отправлять данные на последовательный порт 1 через хост с помощью приложения Serial Pro, как показано на рис. 1-7-2.

Рисунок 1-7-2 Отправка данных на последовательный порт через Serial Pro

Последовательный порт 1 и последовательный порт 3 (ROS) принимают команды управления одинаковым образом. Подробную информацию о преобразовании данных и скорости робота см. в разделе 1.2 «Управление через ROS (последовательный порт 3)».

2 Интерфейс OLED-дисплея

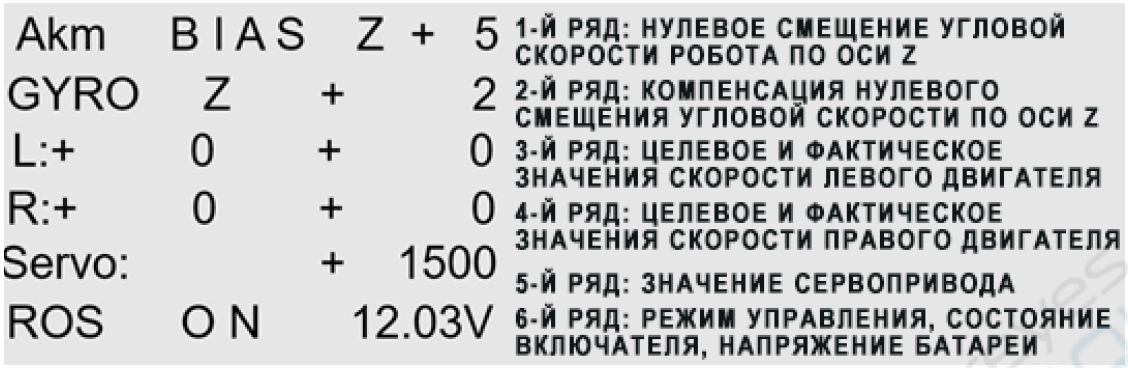
2. 1 Данные на OLED-дисплее

Роботы оснащены OLED-дисплеем. Данные, отображаемые на дисплее, для различных моделей аналогичны.

1 Гусеничный робот-автомобиль

Tank BIAS Z + 5	1-Й РЯД: НУЛЕВОЕ СМЕЩЕНИЕ УГЛОВОЙ СКОРОСТИ РОБОТА ПО ОСИ Z
GYRO Z + 2	2-Й РЯД: КОМПЕНСАЦИЯ НУЛЕВОГО СМЕЩЕНИЯ УГЛОВОЙ СКОРОСТИ ПО ОСИ Z
L:+ 0 + 0	3-Й РЯД: ЦЕЛЕВОЕ И ФАКТИЧЕСКОЕ ЗНАЧЕНИЯ СКОРОСТИ ЛЕВОГО ДВИГАТЕЛЯ
R:+ 0 + 0	4-Й РЯД: ЦЕЛЕВОЕ И ФАКТИЧЕСКОЕ ЗНАЧЕНИЯ СКОРОСТИ ПРАВОГО ДВИГАТЕЛЯ
MA + 0 MB+ 0	5-Й РЯД: ЗНАЧЕНИЕ ШИМ-СИГНАЛА ДВИГАТЕЛЯ
ROS ON 12.03V	6-Й РЯД: РЕЖИМ УПРАВЛЕНИЯ, СОСТОЯНИЕ ВКЛЮЧАТЕЛЯ, НАПРЯЖЕНИЕ БАТАРЕИ

2 Тележка Ackermann

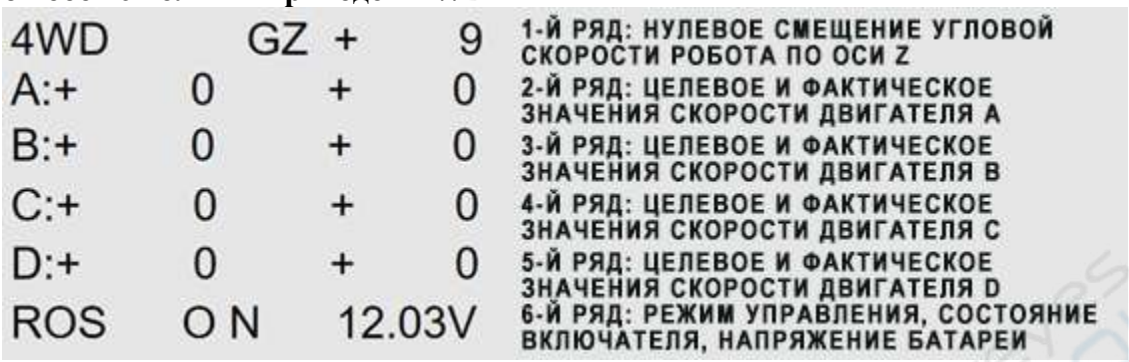


3 Двухколесная дифференциальная тележка

4 Всенаправленная тележка

5 Робот Mecanum Wheel

6 Робот с полным приводом 4WD



2.2 Универсальные данные на OLED-дисплее

1 Режим управления

Режимы управления отображаются в левом нижнем углу дисплея, см. табл. 2-1.

Таблица 2-1 Режимы управления роботом, отображение на дисплее

Режим	CAN	Приложение	Контроллер PS2	Радио-управление	Послед. порт 1	Послед. порт 3
Отображение	CAN	APP	PS2	R-C	UART1	UART3

2 Кнопка включения

Кнопка включения находится в левом верхнем углу контроллера STM32. Чтобы запустить управление двигателями робота, переведите кнопку в положение «ON».

Программа учитывает стабильность робота, поэтому после включения двигателей контроллер STM32 еще 10 секунд находится в режиме ожидания. В это время, даже если кнопка включения в положении «ON», на дисплее будет отображаться положение «OFF», и только через 10 секунд переключится на «ON». Следует помнить, что последовательные порты 1 и 3 не принимают данные в первые 10 секунд исходя из настроек программы.

2.3 Самодиагностика робота-тележки

Роботы оснащены программой самодиагностики. Подробности см. в файле «Руководство по проверке энкодеров, схем подключения, моделей».

3 Компенсация смещения нуля гироскопа

Чтобы устранить смещение нуля гироскопа, в системе навигации с ROS предусмотрен IMU-сенсор. В системе ROS данного робота IMU-сенсор интегрирован в контроллер STM32. Контроллер STM32 собирает данные с IMU-сенсора и отправляет их в ROS. На STM32 установлен IMU-сенсор MPU9250, который объединяет в себе трехосный измеритель угловой скорости, трехосный акселерометр, трехосный магнитометр. В данном случае используются только измеритель угловой скорости и акселерометр. Так как избежать смещения нуля гироскопа невозможно, в программе предусмотрен механизм его компенсации.

В первые 10 секунд после включения гироскоп считывает значение угловой скорости без компенсации смещения нуля, программа запоминает текущее значение угловой скорости в качестве смещения. Через 10 секунд после включения светодиодный индикатор начнет мигать, а измеренное значение смещения будет вычитываться из всех значений угловой скорости. Таким образом смещение нуля будет скомпенсировано.

Если значение смещения нуля гироскопа, полученное в первые 10 секунд, не является точным и требуется его отредактировать, дважды щелкните кнопку пользовательских настроек (в нижнем левом углу контроллера STM32) .

На рис. 3-1 показан фрагмент программы сбора данных гироскопа из файла get_sensor.c с контроллера STM32.

Рисунок 3-2 Сбор данных гироскопа

4 Кинематический анализ

Чтобы робот двигался в соответствии с желаниями пользователя, просто задать целевую скорость робота недостаточно — ее нужно преобразовать в реальные целевые скорости вращения каждого двигателя и управлять ими таким образом, чтобы выполнить требования управления и обеспечить целевую скорость робота. Процесс преобразования целевой скорости робота в целевые скорости двигателей называется кинематическим анализом. Кинематический анализ предполагает решение прямой и обратной задач кинематики:

Прямая задача кинематики: отыскать скорости робота по осям X, Y и Z на основании скоростей каждого колеса.

Обратная задача кинематики: отыскать скорости каждого колеса на основании скоростей робота по осям X, Y и Z.

4.1 Исходный код PI-регулятора в программе управления

Целевые скорости двигателей, полученные с помощью кинематического анализа, передаются на PID-регулятор для контроля скорости в замкнутом контуре и получения фактических выходных скоростей двигателей близким к заданным значениям.

Исходный код PI-контроллера в программе выглядит следующим образом:

```
/**************************************************************************
*  Назначение функции: инкрементальное PI-регулирование скорости
*  Входные параметры: Encoder: фактическое значение энкодера, Target: целевое значение.
*  Возвращаемое значение: Pwm: ШИМ двигателя
*  Описание функции: pwm+=Kp[e（k）-e(k-1)]+Ki*e(k) инкрементальное PI-регулирование
**************************************************************************/
int Incremental_PI_A (float Encoder,float Target)
{ static float Bias,Pwm,Last_bias;
    Bias=Target-Encoder;                  //Расчет смещения
   PWM+=Velocity_KP*  (Bias-Last_bias)  +Velocity_KI*Bias;  //инкрементальный  PI-регулятор
   if(Pwm>7200)Pwm=7200; if(Pwm<-7200)Pwm=-7200;
   Last_Bias=Bias; //сохраняет последнее смещение
   return Pwm;           // вывод инкремента
}
```

5 Схемы подключения

В этой главе приводятся ключевые указания к подключению схем робота, а также фотографии схем для наглядности.

Контроллер STM32 имеет два канала питания 5V. Первый канал 5V обеспечивает питание контроллера STM32 и периферийных устройств (энкодер, Bluetooth-модуль, джойстик и т.д.). Второй канал 5V питает Raspberry Pi.

1 Питание Raspberry Pi

Плата питания 5V Raspberry Pi интегрирована в плату расширения контроллера STM32. Для питания можно использовать кабели TYPE-C ↔ TYPE-C для тока 3 A и выше

Рисунок 5-1-1 Схема питания Raspberry Pi

2 Подключение Raspberry Pi и STM32 через последовательный порт

Так как Raspberry Pi используется в качестве хоста для связи с контроллером STM32, для передачи данных по умолчанию выбирается последовательный порт 3 на интегрированном чипе CP2102 с преобразователем уровней.

Рисунок 5-1-2 Подключение Raspberry Pi к контроллеру STM32

3 Подключение системы навигации к Raspberry Pi

Для подключения лидара к Raspberry Pi требуется обычный кабель Micro-USB, через который Raspberry Pi одновременно питает и обменивается данными с лидаром.

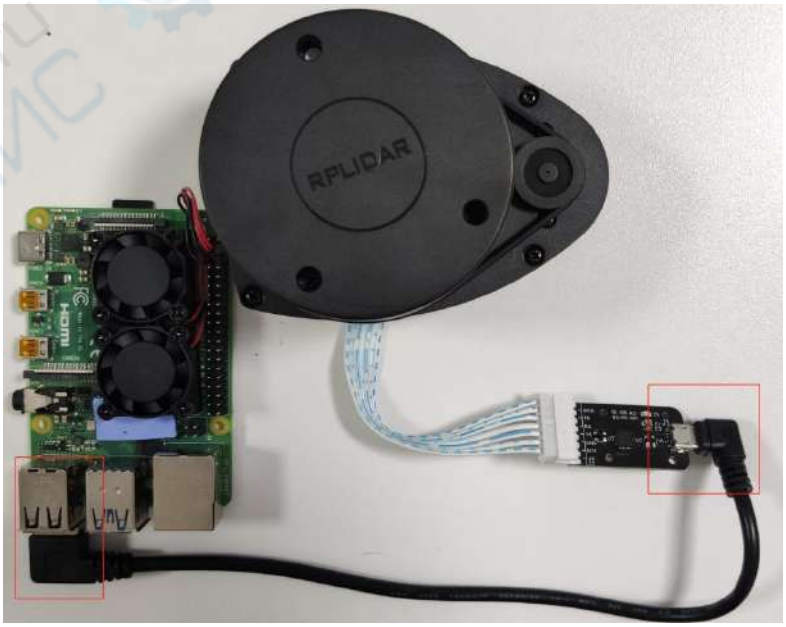


Рисунок 5-1-3 Подключение лидара к Raspberry Pi

4 Подробная схема периферийных устройств на контроллере STM32

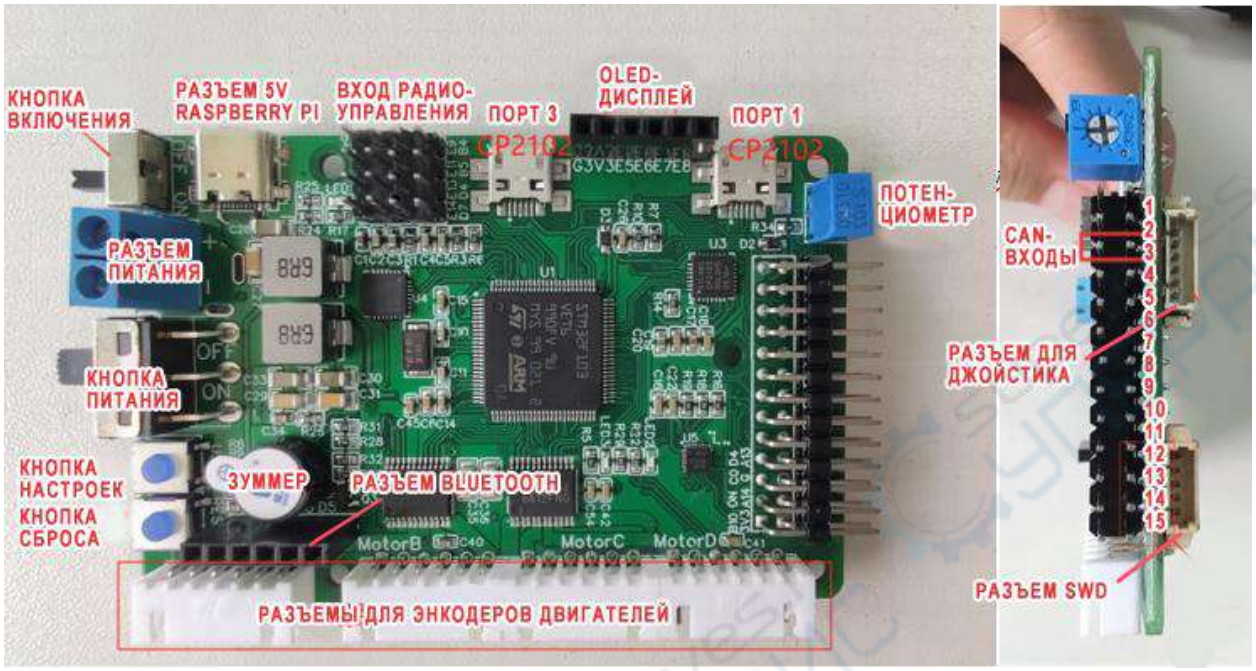


Рисунок 5-1-4 Подключение периферийных устройств к STM32

6 Блок-схема управления

6.1 Блок-схема управления двигателями робота

Робот поддерживает шесть режимов управления. Принцип управления всех режимов заключается в достижении заданного движения регулированием скорости робота. Целевые скорости двигателей рассчитываются с помощью кинематического анализа на основании целевой скорости робота, далее фактические скорости двигателя управляются в помощью PID-регуляторов (функция PID).

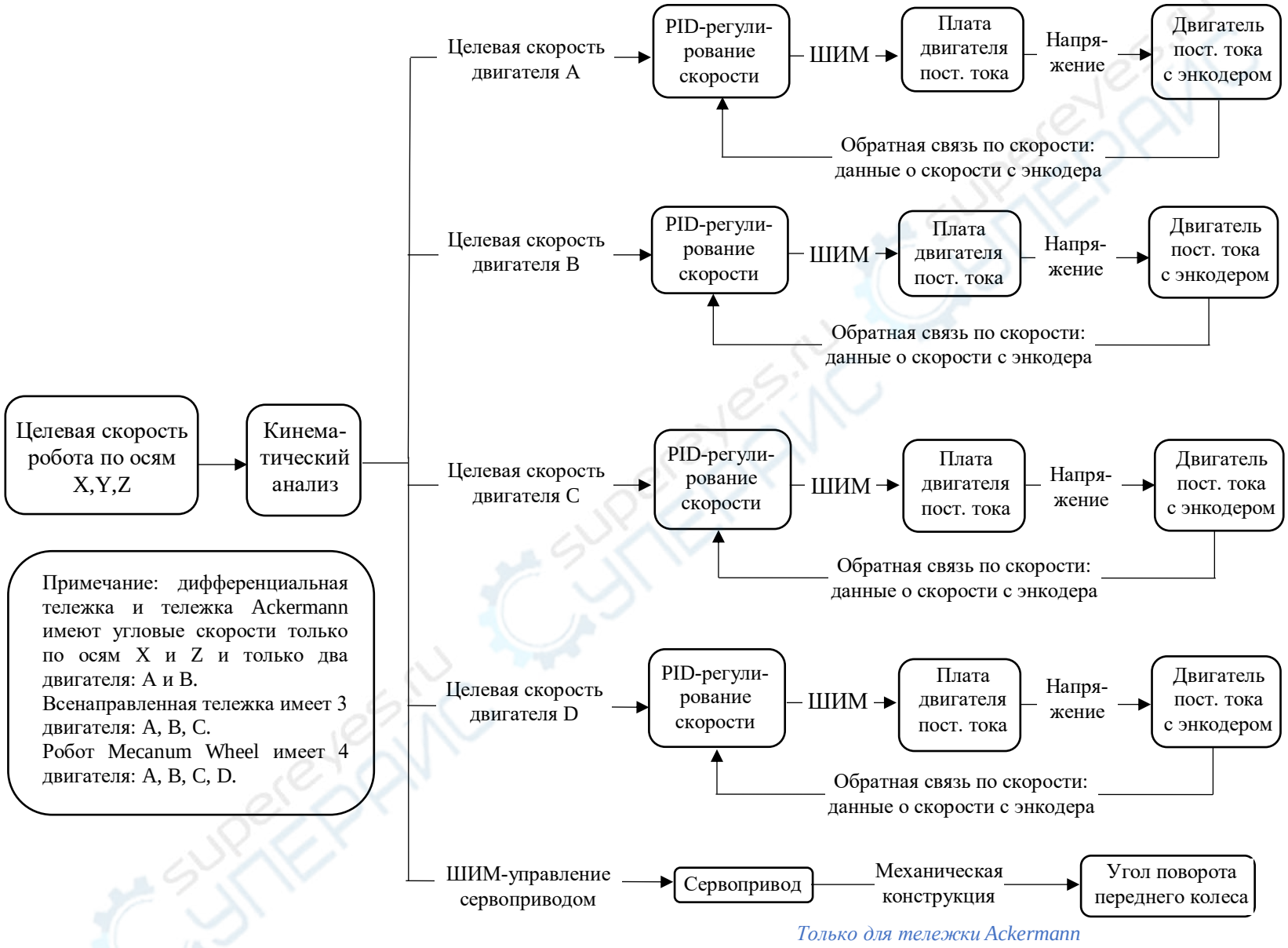


Рисунок 6-1 Блок-схема программы управления роботом

6.1 Блок-схема программы контроллера STM32

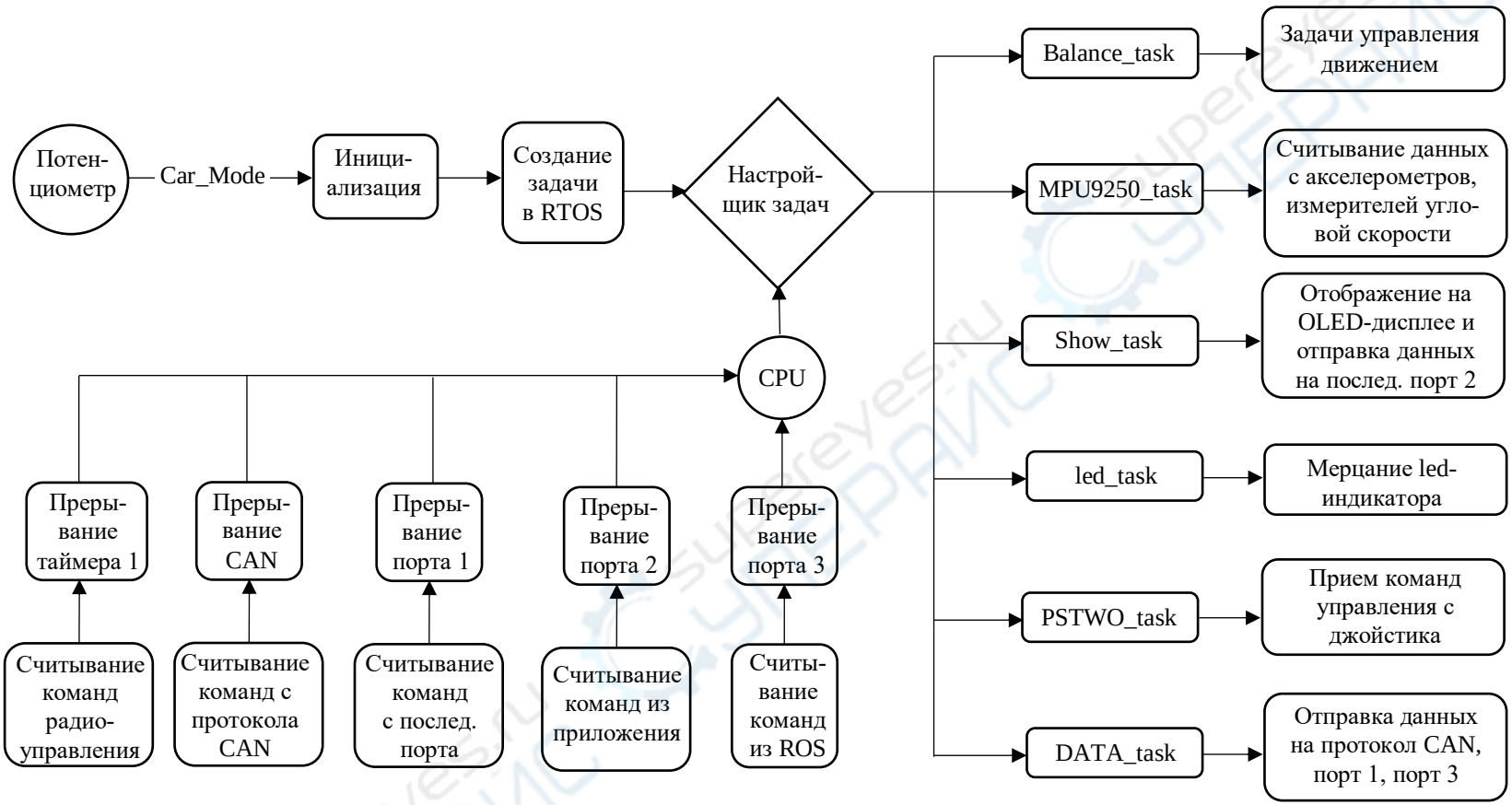


Рисунок 6-2 Блок-схема программы управления контроллера STM32

Планировщик задач RTOS определяет порядок выполнения задач исходя из их приоритета (на рис. 6-2 не указан приоритет задач, так как приоритет той или иной задачи задается в настройках программы). Время выполнения каждой задачи достаточно мало, и можно считать, что все задачи выполняются практически одновременно. Если случается прерывание, работа планировщика тоже прерывается. Прерывание последовательного порта 2 используется при управлении роботом с приложения через Bluetooth. Прерывание последовательного порта 3 используется для приема данных из ROS.

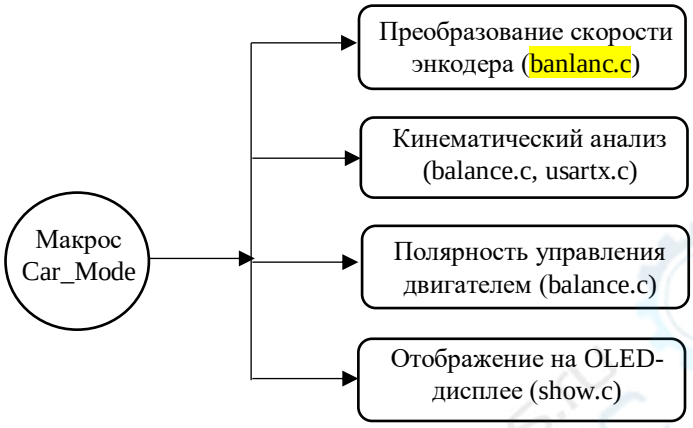


Рисунок 6-3 Влияние Car_Mode на модули программы управления

6.3 Схема подключения контроллеров робота

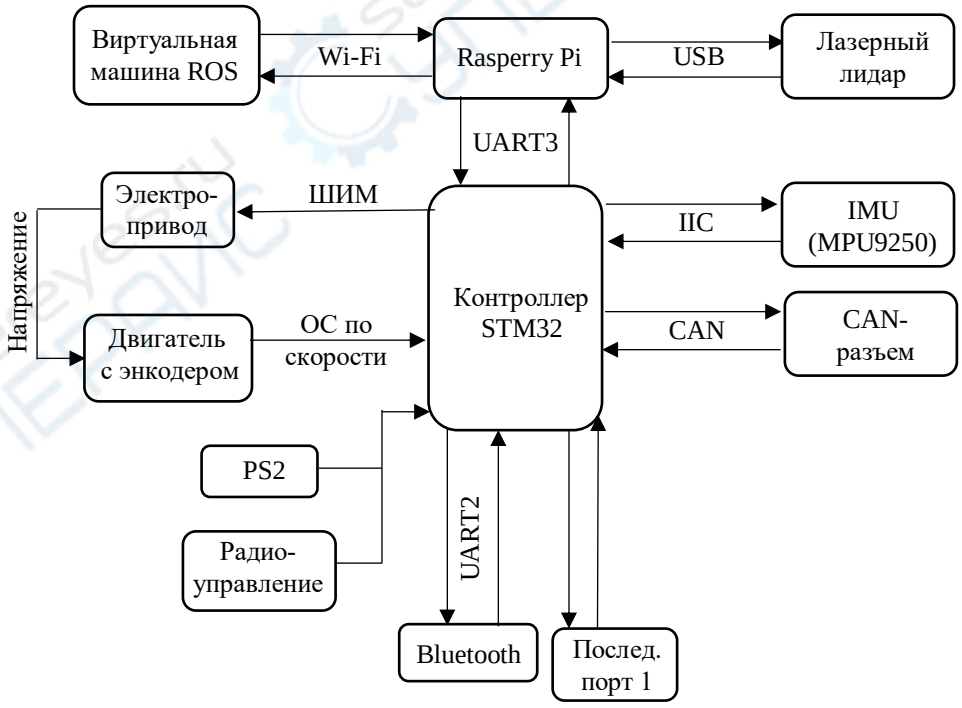


Рисунок 6-4 Схема подключения устройств к контроллеру

В роботах используются различные контроллеры и периферийные устройства: Raspberry Pi (Jetson Nano), лазерный лидар, контроллер STM32, моторы, энкодер, двойной привод, Bluetooth, джойстик PS2, пульт радиоуправления, гироскоп и др. Встроенные последовательный порт 1 и интерфейс CAN дают пользователю больше возможностей для управления. Схема подключения контроллера и периферийных устройств показана на рис. 6-4.

7 Особые указания

7.1 О кодах

Программа контроллера STM32, написанная для планировщика RTOS, отличается от программ для контроля прерываний. Планировщик RTOS выполняет задачи по очереди в зависимости от их приоритета (приоритет прерываний выше, чем приоритет задачи). Важно отметить, что если в задаче допущена логическая ошибка, программа может зависнуть и перестать работать.

Пример: в программе стоит задача отправки данных на последовательный порт 3, но последовательный порт 3 не инициализирован или в коде инициализации допущена ошибка, тогда при выполнении задачи программа просто зависнет. В таких случаях, если программа зависла и не работает после перезапуска, следует выяснить, нет ли ошибок в выполняемых задачах.

7.2 О разъемах питания на плате расширения

Контакты 5V и 3.3V на плате расширения могут использоваться для питания внешних устройств, но не следует подключать к ним слишком большие нагрузки. Контакт 5V предназначен для токов не более 1,5 А, контакт 3.3V предназначен для токов не более 200 мА.

Как видно на схеме, плата расширения оснащена двумя цепями питания 5V, одна из которых предназначена для питания периферийных устройств, а другая — для питания Raspberry Pi (Jetson Nano) (USB-разъем).

Подробнее см. в разделе 5 «Схемы подключения»

7.3 О двигателях

Избегайте заклинивания двигателей во время работы робота, чтобы не повредить системную плату. Перед тестированием робота поднимите его, дайте двигателям поработать на весу, проверьте правильность их работы, ничего ли не мешает вращению.

7.4 О батарее

Напряжение батареи отображается на дисплее. Когда заряд батареи ниже 10.8 В, необходимо ее зарядить. Батарея оснащена защитой от избыточной зарядки. Пожалуйста, не используйте батарею, когда она заряжается. Схема зарядки показана на рис. 7-4-1.

Рисунок 7-4-1 Схема подключения батареи к зарядному устройству

После зарядки закройте разъем зарядного устройства крышкой, чтобы избежать короткого замыкания батареи (см. рис. 7-4-2).

Рисунок 7-4-2 Разъем зарядного устройства робота

8 Как загрузить программы на контроллер STM32

Загружать программы на контроллер STM32 можно через последовательный порт или SWD-интерфейс. Загрузка программ на последовательный порт осуществляется через USB-кабель (поставляется в комплекте). Для загрузки через SWD-интерфейс рекомендуется использовать StLink-программатор в металлическом корпусе.

8.1 Загрузка через последовательный порт

Для удобства пользователя загрузка программ на системную плату осуществляется одной кнопкой. Потребуется только обычный кабель Micro-USB для смартфона.

Требуемое аппаратное обеспечение:

1. Контроллер STM32.
2. Кабель Micro-USB для передачи данных.

Программное обеспечение:

Приложение для записи mcuisp (доступно в бесплатных материалах), соответствующий драйвер USB-TTL для CP2102. Все драйверы доступны в бесплатных материалах. Если возникли проблемы с установкой драйверов, скачайте приложение «DriverGenius».

По окончании установки откройте диспетчер устройств и убедитесь, что драйвер успешно установлен. В противном случае появится красный восклицательный знак.

Рисунок 8-1-1 Проверка драйвера CP210x в Диспетчере устройств

Программы загрузки через последовательный порт очень проста, достаточно подключить плату расширения к компьютеру с помощью MicroUSB. Откройте приложение mcuisp в бесплатных материалах, установите его по алгоритму, указанному на рис. 8-1-2.

После установки и настройки нажмите «Начать программирование», загружайте программу. Так как стоит галочка «Выполнить после программирования», программа запустится автоматически после загрузки.

Примечание: уберите галочку с пункта «Запись байта опции при программировании во FLASH». Если вы используете плату F4, скорость передачи данных должна быть 76800.

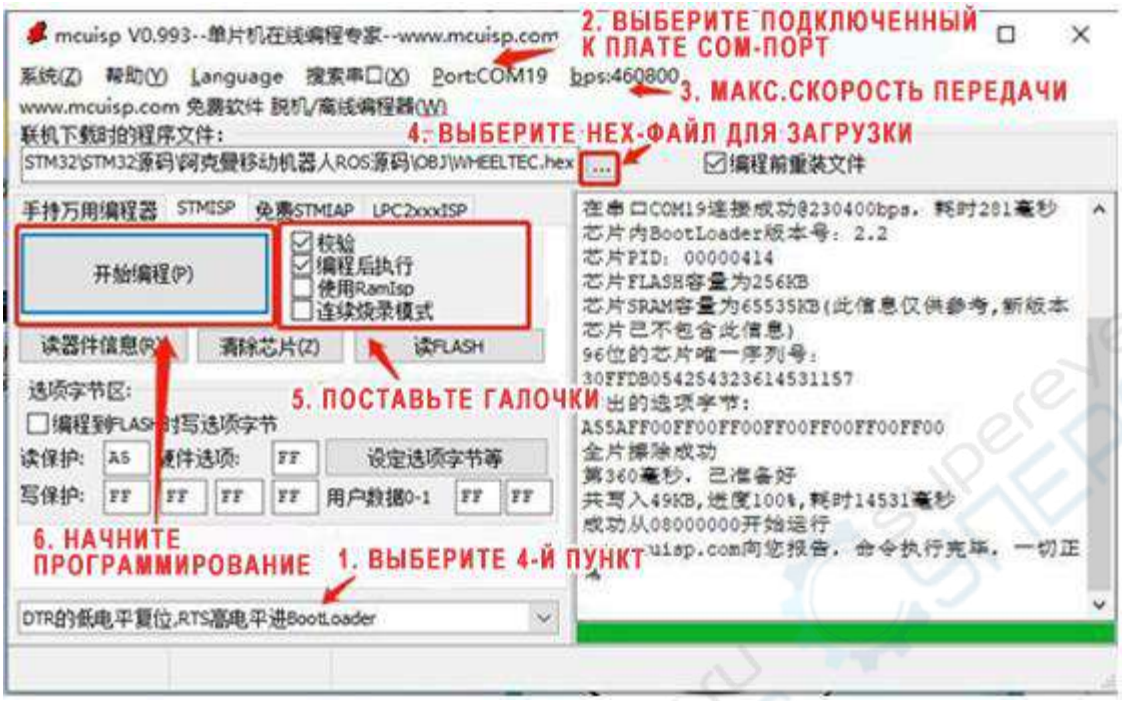


Рисунок 8-1-2 Инструкции по настройке загрузчика FlyMCU

8.2 Загрузка через SWD

Также можно загружать программы на контроллер STM32 с помощью SWD-интерфейса, на системной плате есть соответствующие маркировки, PA13 и PA14.

Требуемое аппаратное обеспечение:

1. Контроллер STM32.

2.Программатор Stlink.

2. Программное обеспечение:

Установите соответствующие драйверы STLink или Jlink.

После успешной установки откройте диспетчер устройств и проверьте, отображается ли устройство STLink.

Рисунок 8-2-1 Проверка драйвера STlinkx в Диспетчере устройств

Как видно, драйвер успешно установлен!

3. Подключение

STlink ---------- Контроллер STM32
SWDIO----------PA13
SWCLK----------PA14
GND----------GND

4. Загрузка программ

В загрузчике нажмите на кнопку, указанную на рис. 8-2-2 красной со стрелкой, загрузите программу. Так как стоит галочка «Выполнить после программирования», программа запустится автоматически после загрузки. Конфигурация программы по умолчанию настроена для STLink. Чтобы настроить загрузку в Jlink, необходимо отредактировать параметр MDK.

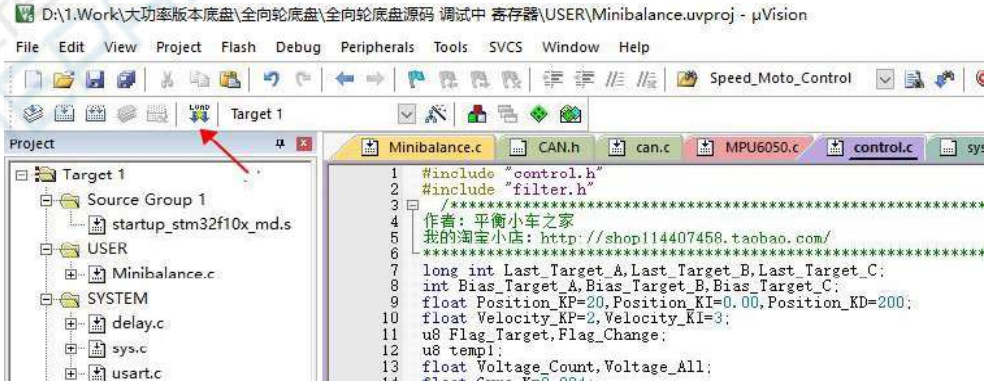


Рисунок 8-2-2 Интерфейс загрузчика STLink