ZARNITZA
ROBOTICS

Производственное объединение «Зарница»

КОНСТРУКТОР ДЛЯ ОБУЧЕНИЯ И ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ РОБОТОВ

Руководство по эксплуатации

РОБОТОТЕХНИКА

ZARNITZA
ROBOTICS

Разрабатываем и производим
высокотехнологичное учебное
оборудование для любых специальностей

УП9147

Образовательный набор для изучения
многокомпонентных робототехнических систем
и манипуляционных роботов
Приказ 838 Минпросвещения РФ

УП9145

Базовый робототехнический набор
для конструирования,
изучения электроники и микропроцессоров
и информационных систем и устройств Z.Robo-2
Приказ 838 Минпросвещения РФ

УП6738

Стол для робототехники
Приказ 838 Минпросвещения РФ

Телефон 8-800-775-37-97
www.zarnitza.ru, zakaz@zrnc.ru

Производственное объединение «Зарница»

КОНСТРУКТОР ДЛЯ ОБУЧЕНИЯ И ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ РОБОТОВ

Руководство по эксплуатации

2025

СОДЕРЖАНИЕ

ВНИМАНИЕ! Перед началом эксплуатации изделия внимательно изучите настоящий паспорт и руководство по эксплуатации!

1. НАЗНАЧЕНИЕ

Конструктор для обучения и проведения соревнований роботов – это универсальный конструктор, который предназначен для обучения робототехнике и программированию. Будет интересным и полезным средством для развития технических навыков, вдохновляющим инструментом для творчества и инноваций, а также надежным помощником в подготовке к соревнованиям и проектной деятельности для детей и подростков от 8 лет и старше, для начинающих и просто любителей технологий. Он представляет собой комплект, включающий в себя разнообразные модули, детали и компоненты, позволяющие создавать множество различных роботов и устройств.

Комплект включает в себя моторы, различные датчики, что позволяет создавать многофункциональных роботов – от простых движущихся моделей, обладающих механизированными узлами, различными передачами и подвижными соединениями до сложных автоматизированных систем, оборудованных механизированной рукой-манипулятором с плоскопараллельной кинематикой. Все детали легко соединяются благодаря системе креплений и универсальным конструктивным элементам, что делает процесс сборки интуитивно понятным даже для новичков.

Программируемый управляющий контроллер, входящий в состав конструктора, совместим с платформой *Arduino* и другими популярными системами, что расширяет возможности обучения электронике и программированию.

Конструктор для обучения и проведения соревнований роботов предназначен для обучения основам робототехники, электроники и программирования, развития логического мышления, креативности и инженерных навыков. Помогает изучить и понять основы механики и конструкции роботов, логические цепочки, алгоритмы и автоматизацию процессов, дает возможность практического изучения принципов разработки и функционирования основных механизмов, применяемых в профессиональной инженерной деятельности.

2. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Параметр			Значение
Программируемый в среде *Arduino IDE*, *mBlock* контроллер			
	Поддерживаемые среды разработки		*Arduino IDE, mBlock*
	Объем памяти, килобайт		256
	Рабочее напряжение постоянного тока, В		5
	Объем памяти SRAM, килобайт		8
	Объем памяти EEPROM, килобайт		4
	Интерфейсы контроллера обеспечивают одновременное подключение:		
		двигателей постоянного тока, шт.	10
		или шаговых двигателей, шт.	4
		или двигателей с энкодером и сервоприводов, шт.	4 и 8

Параметр	Значение
Тактовая частота процессора, МГц	16
Количество контактов ввода/вывода, шт.	43
Количество последовательных портов, шт.	3
Интерфейс *I²C*, шт.	1
Интерфейс *SPI*, шт.	1
Количество аналоговых входов, шт.	15
Плата расширения	
Количество портов *RJ-25* для подключения внешних модулей, шт.	4
Сечение балок прямоугольного профиля, мм	8x24
Количество типоразмеров балок прямоугольного профиля, шт.	7
Длина наименьшей балки прямоугольного профиля, мм	16
Длина наибольшей балки прямоугольного профиля, мм	224
Сечение балок П-образного квадратного профиля, мм	8x8
Длина балок П-образного квадратного профиля, мм	24
Сечение балок плоских, мм	12x4
Количество типоразмеров балок плоских, шт.	5
Длина наименьшей плоской балки, мм	76
Длина наибольшей плоской балки, мм	220
Количество разновидностей кронштейнов, шт.	4
Количество разновидностей перфорированных пластин, шт.	4
Число зубьев зубчатых колес из алюминия и пластика	8, 56, 72, 90
Двигатели постоянного тока с встроенным энкодером	
Диаметр корпуса двигателей, мм	25
Напряжение питания двигателей постоянного тока, В	9
Максимальная скорость привода двигателя, тип 1, об/мин	86
Максимальная скорость привода двигателя, тип 2, об/мин	185
Размеры шин резиновых (наружный диаметр x ширина), мм	63x16
Наружный диаметр подшипника, мм	34
Электропитание: 6 батарей AA, В	9
Масса робота в сборе, кг, не более	2

Примечания:

1) Изготовитель допускает наличие предельных отклонений габаритных размеров изделия – ±20 мм.

2) В процессе модернизации производителем, а также в зависимости от партии общий вид, применяемые материалы и элементы конструкции изделий могут изменяться.

3) Предприятие-изготовитель оставляет за собой право вносить изменения в конструктивные особенности, а также в набор комплектующих изделия, не отраженных в эксплуатационной документации и не влияющих на уровень технических, эксплуатационных характеристик и параметров безопасности поставляемого оборудования.

3. КОМПЛЕКТ ПОСТАВКИ

Наименование	Кол-во, шт.
Конструктор для обучения и проведения соревнований роботов	
Программируемый в среде *Arduino IDE, mBlock* контроллер, шт.	1
Электронные модули, совместимые со средой программирования *Arduino-IDE,* для подключения напрямую к контроллеру:	
Bluetooth-модуль, шт.	1
Драйвер двигателя постоянного тока с энкодером, шт.	4
Плата расширения, шт.	1
Электронные модули, совместимые со средой программирования *Arduino-IDE,* оснащенные разъемами *RJ-25* для подключения к контроллеру:	
Трехосевой акселерометр и гироскоп, шт.	1
Датчик расстояния, шт.	1
Датчик линии, шт.	1
Адаптер *6P6C/RJ-25*, шт.	1
Металлические структурные элементы, выполненные из алюминия с анодированным покрытием (балки, пластины, уголки)	
Балки прямоугольного профиля, с желобом с насечкой под винт M4 по всей длине, с двумя рядами отверстий диаметром 4 мм вдоль балки	
Балки прямоугольного профиля 16 мм, шт.	4
Балки прямоугольного профиля 32 мм, шт.	5
Балки прямоугольного профиля 64 мм, шт.	3
Балки прямоугольного профиля 128 мм, шт.	2
Балки прямоугольного профиля 176 мм, шт.	1
Балки прямоугольного профиля 192 мм, шт.	2
Балки прямоугольного профиля 224 мм, шт.	1
Балки П-образного квадратного профиля, с рядом отверстий диаметром 4 мм вдоль балки, шт.	2
Балки плоские, с рядом отверстий диаметром 4 мм вдоль балки	
Балки плоские 76 мм, шт.	4
Балки плоские 92 мм, шт.	4
Балки плоские 140 мм, шт.	4
Балки плоские 188 мм, шт.	6
Балки плоские 220 мм, шт.	2
Кронштейны (угловые, фигурные)	
Кронштейн, шт.	2
Кронштейн двигателя, шт.	3
Кронштейн угловой 3x3, шт.	4
Кронштейн угловой, шт.	4
Перфорированные пластины (прямоугольные, фигурные), с отверстиями диаметром 4 мм	
Перфорированные пластины 56 мм, шт.	2

Наименование	Кол-во, шт.
Перфорированные пластины 88 мм, шт.	3
Перфорированные пластины 3x6, шт.	2
Перфорированные пластины 7x9, шт.	1
Двигатель постоянного тока с встроенным энкодером, тип 1, шт.	1
Двигатель постоянного тока с встроенным энкодером, тип 2, шт.	2
Конструктивные элементы и элементы для сборки	
Зубчатые колеса из алюминия и пластика (с числом зубьев 8, 56, 72, 90), шт.	14
Гусеницы резиновые, шт.	2
Шины резиновые, шт.	4
Акриловые крепления (прямоугольные и фигурные) с отверстиями	
Быстросъемная пластина, шт.	1
Пластина крепления двигателя, шт.	2
Пластина крепления контроллера, шт.	2
Подшипник упорный, шт.	1
Металлические валы диаметром 4 мм, длиной 50, 88, 160 мм, шт.	5
Валы с резьбой M4 на одной из сторон, длиной 39 мм, шт.	6
Вспомогательные элементы для валов (соединительные втулки, установочные кольца, кольцевые разделители), шт.	30
Комплект для сборки захватного устройства (захват манипулятора, крепежные элементы), шт.	1
Батарейный отсек (6xAA)	1
Кабели *6P6C RJ25* длиной 20 см, 35 см, шт.	3
Кабели для подключения моторов, шт.	3
Кабель *USB A – USB B*, шт.	1
Металлические крепежные элементы (винты, гайки), шт.	201
В том числе:	
Винт установочный M3x5, шт.	12
Винт установочный M3x8, шт.	8
Винт M3x8, шт.	6
Винт M3x10, шт.	4
Винт M4x8, шт.	50
Винт M4x14, шт.	46
Винт M4x16, шт.	10
Винт M4x22, шт.	4
Винт M4x30, шт.	4
Гайка M4, шт.	47
Гайка M4 с фиксатором, шт.	10
Латунные штифты M4, шт.	4
Хомуты пластиковые, шт.	10
Прокладки пластиковые круглые, шт.	14

Наименование	Кол-во, шт.
В том числе:	
Прокладка пластиковая 4x7x2, шт.	4
Прокладка пластиковая 4x7x3, шт.	8
Прокладка пластиковая 4x7x10, шт.	2
Пластиковые заклепки, шт.	60
В том числе:	
Пластиковая заклепка 6 мм, шт.	20
Пластиковая заклепка 10 мм, шт.	20
Пластиковая заклепка 12 мм, шт.	20
Комплект инструментов для сборки (отвертка универсальная, гаечный ключ, шестигранные ключи, торцевой ключ), шт.	1
USB-накопитель, шт.	1
Паспорт	1
Руководство по эксплуатации	1
Инструкция по сборке	1

4. ОБЩИЙ ВИД

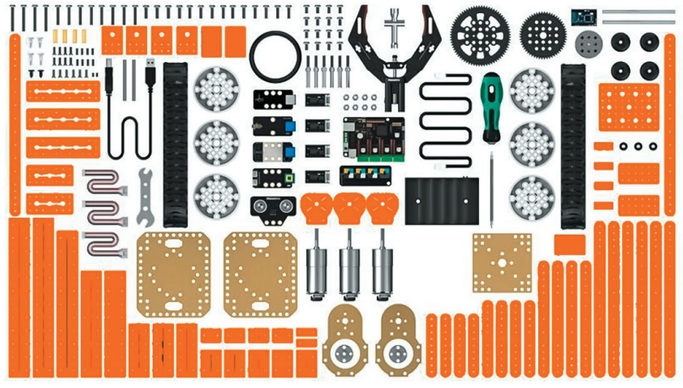


Рисунок 1. Общий вид

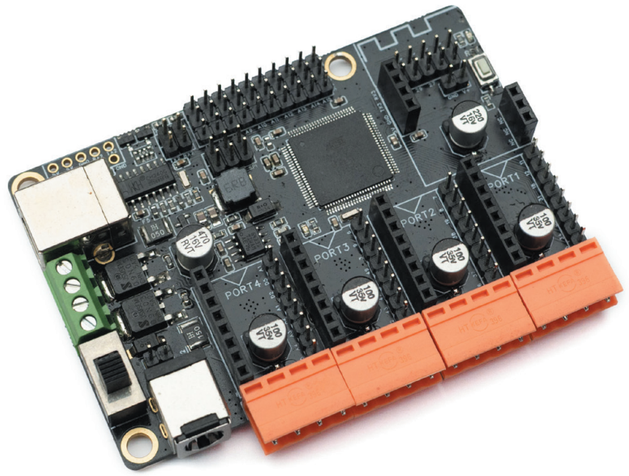
5. ОПИСАНИЕ И ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ОБОРУДОВАНИЯ

Состав набора

5.1. Программируемый в среде *Arduino IDE, mBlock* контроллер

В качестве модуля управления в конструкторе используется управляющий контроллер, созданный на базе проверенной временем *Arduino Mega 2560*, который использует тот же чип *ATmega2560.*

Контроллер поддерживает среду программирования *Arduino IDE* и графическое программное обеспечение *mBlock.*



Основные характеристики

Параметр	Значение
Основной чип управления	ATMEGA2560-16AU
Входное напряжение/ток	DC 6-12V 10A
Рабочее напряжение	DC 5V
Общие контакты ввода/вывода	43
Последовательный порт	3
Порт I2C	1
Порт SPI	1
Аналоговый входной порт	15
Номинальный постоянный ток на контакт ввода/вывода	20 mA
Флэш-память	256 КБ
Статическая оперативная память	8 КБ
EEPROM	4 КБ
Основная частота	16 МГц
Размер продукта	86 мм × 53 м

Описание платы контроллера

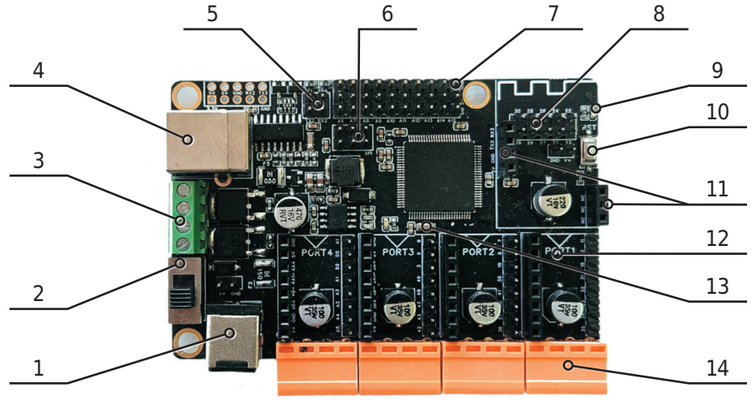


Рисунок 2. Плата контроллера:
1 – Разъем подключения источника питания (6–12 В); 2 – Выключатель питания;
3 – Разъемы подключения нагрузки с высокой мощностью потребления 0–10 А;
4 – *USB Type-B* интерфейс; 5 – *UART2, I²C*; 6 – *SPI*; 7 – Интерфейс соединения с сервоприводами x10;
8 – Цифровые пины ввода/вывода; 9 – Индикатор питания; 10 – Кнопка *Reset*;
11 – Разъем для подключения модуля беспроводной связи UART3;
12 – Разъем драйвера двигателя x4; 13 – Пользовательский индикатор ввода/вывода;
14 – Разъем подключения шагового/двигателя постоянного тока x4.

На плате контроллера расположены:

- Беспроводной интерфейс связи для добавления модуля *Bluetooth* или модуля *2.4G.*
- Десять разъемов для подключения сервоприводов, которые позволяют плате управлять до 10 сервоприводами одновременно.
- Разъем интерфейсов *UART2* и *I²C.*
- Разъем интерфейса *SPI.*
- 10 цифровых пинов ввода/вывода.
- Четыре интерфейса для подключения драйверов двигателей:

  - Порт 1 – интерфейс драйвера двигателя для управления двигателем постоянного тока, шаговым двигателем или двигателем с энкодером (требуется соответствующий модуль управления).

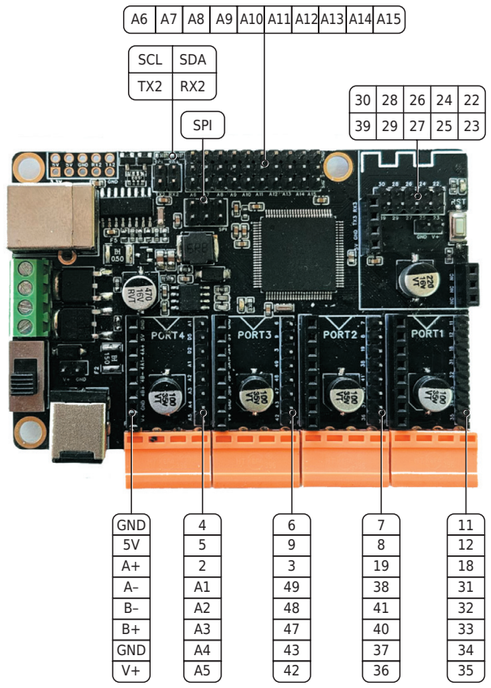
  - Порт 2 – интерфейс драйвера двигателя для управления двигателем постоянного тока, шаговым двигателем или двигателем с энкодером (требуется соответствующий модуль управления).

  - Порт 3 – интерфейс драйвера двигателя для управления двигателем постоянного тока, шаговым двигателем или двигателем с энкодером (требуется соответствующий модуль управления).

  - Порт 4 – интерфейс драйвера двигателя для управления двигателем постоянного тока, шаговым двигателем или двигателем с энкодером (требуется соответствующий модуль управления).

- Два мощных интерфейса драйвера МОП, способных управлять устройствами с максимальным током 10 А. Максимальный выходной ток обычных портов составляет 5 В постоянного тока и 3 А.
- Выключатель питания.
- Разъем *USB Type B* – интерфейс для загрузки программ и обмена данными.
- Кнопка сброса *Reset*, индикатор питания (красный) и пользовательский индикатор (подключен к *D13*) ввода/вывода (синий).

Нумерация выведенных на плату пинов контроллера



Начало работы с контроллером.
Программирование контроллера в среде программирования *Arduino IDE*

Что такое инструменты программирования роботов и зачем нужны интегрированные среды разработки (сокращенно – *IDE*)?

IDE (среда интегрированной разработки) – это программное обеспечение, которое обеспечивает все необходимые инструменты для разработки программного обеспечения в одном месте. Это своеобразное «рабочее место» для разработчиков, где они могут писать код, отлаживать программы, управлять версиями кода, создавать пользовательский интерфейс и многое другое. Программы (скетчи) в *Arduino IDE* пишутся на языке, подобном *C++*.

Представим, что мы собираем конструктор. Нужны детали, инструкция, отвертка и коробка, где все это лежит.

IDE – это как такая «коробка» для программиста. В ней есть все, чтобы писать код, проверять его на ошибки и загружать в робота:

- Текстовый редактор (как тетрадь, где пишем команды).
- Компилятор или интерпретатор (переводит наш код на язык, понятный роботу).
- Отладчик (помогает находить и исправлять ошибки).
- Инструменты для подключения к роботу (как *USB*-кабель или *Bluetooth*).

Зачем нужны *IDE* для роботов?

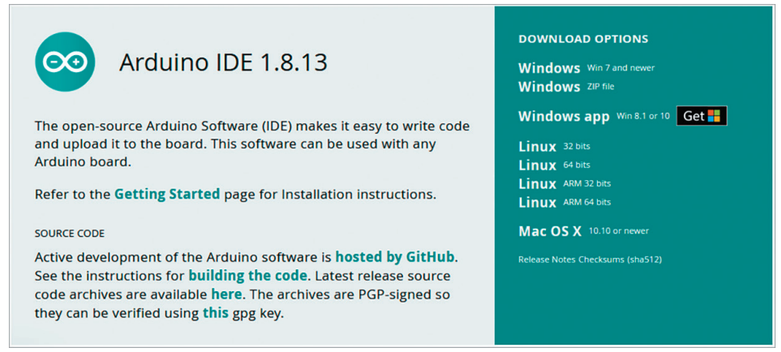
Робот – это железная «игрушка» с мозгами (микроконтроллером). Чтобы он делал то, что мы хотим, нужно написать для него программу. Но как?

- Без *IDE* это как писать письмо в блокноте, а потом вручную переводить его на китайский. Сложно и долго.
- С *IDE* все автоматизировано: пишется код на понятном языке (например, *Python*), нажимаем кнопку – и программа сразу загружается в робота.

Итак, для начала работы с контроллером необходимо установить среду программирования *Arduino IDE.*

Установка *Arduino IDE*

- Скачайте *Arduino IDE* с официального сайта *https://www.arduino.cc/en/software/*
- Установите программу, следуя инструкциям для вашей ОС *(Windows/macOS/Linux).*

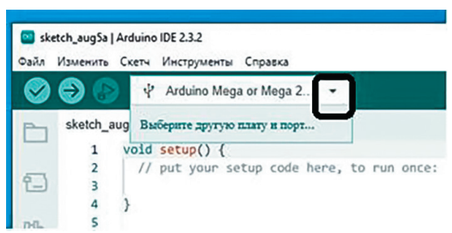


Подключение контроллера

- Соедините плату контроллера с компьютером с помощью *USB*-кабеля *(USB-A → USB-B).*
- На плате должен загореться красный светодиод *ON* (питание).

1. Настройка *Arduino IDE*

- Откройте *Arduino IDE.*

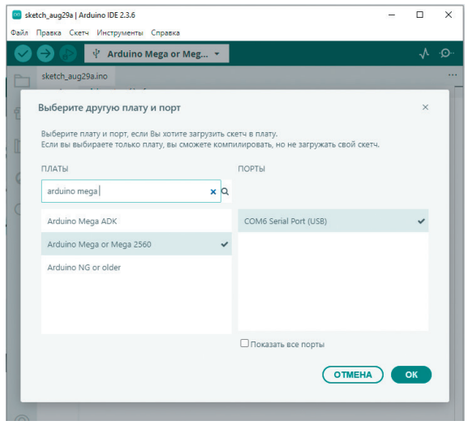


- Выберите плату:

Инструменты → Плата → *Arduino Mega or Mega 2560.*

- Выберите порт:

Инструменты → Порт → *COMx (Arduino Mega 2560).*



Загрузим тестовый скетч *Blink.*

Blink (Моргание) – это самый простой и популярный пример программы для *Arduino,* который заставляет встроенный светодиод (на пине 13) мигать: включился → подождал → выключился → подождал → и так по кругу.

Для чего он нужен?

- Проверка, что контроллер работает

Если светодиод мигает – значит, плата исправна, программа загрузилась, и все подключено правильно.

- Первая программа для новичков

Помогает понять, как писать код для *Arduino.* Показывает базовые команды: *pinMode(), digitalWrite(), delay().*

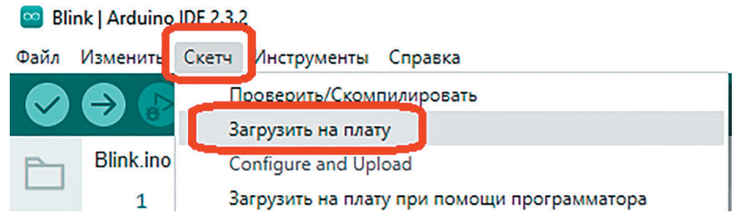
Для загрузки тестового скетча *Blink* откройте пример:

Файл → Примеры → *Basics* → *Blink.*

Проверьте код:

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```


- Нажмите → (Загрузить) или *Ctrl + U.*



- Дождитесь сообщения «Загрузка завершена».

Проверка работы

- После загрузки синий светодиод (подключенный к цифровому пину 13) начнет мигать с интервалом 1 секунду.

Работа в среде программирования *mBlock*

Сначала необходимо установить графическую среду программирования *mBlock 5.*

Для этого:

• Откройте сайт *https://mblock.cc/pages/downloads* и скачайте версию ПО для своего компьютера. Если вы используете компьютер с Linux или будете использовать WEB-версию в другой ОС, скачайте драйвер *mLink.*

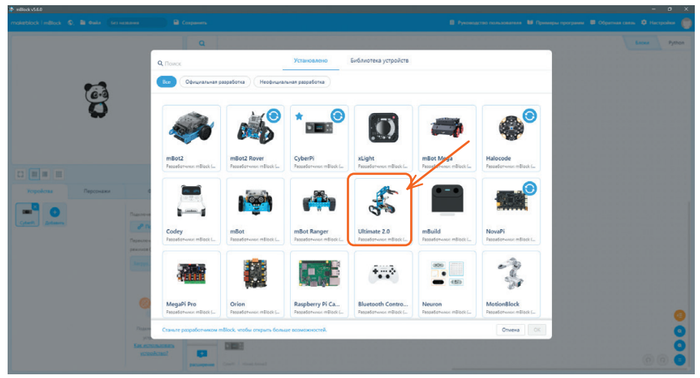
• Установите скачанное ПО на свой компьютер.

• Запустите ПО *mBlock 5.*

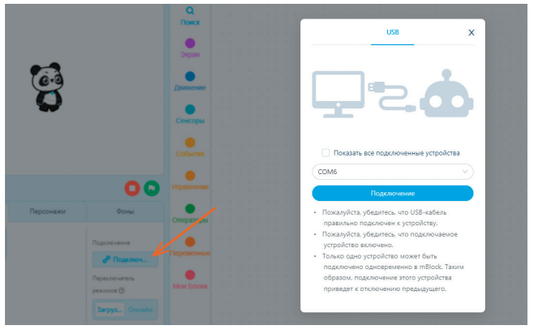
• Ознакомьтесь с интерфейсом программы.

• Для добавления устройства нажмите кнопку «+».

• В разделе оборудования выберите устройство и нажмите кнопку «OK» внизу экрана.

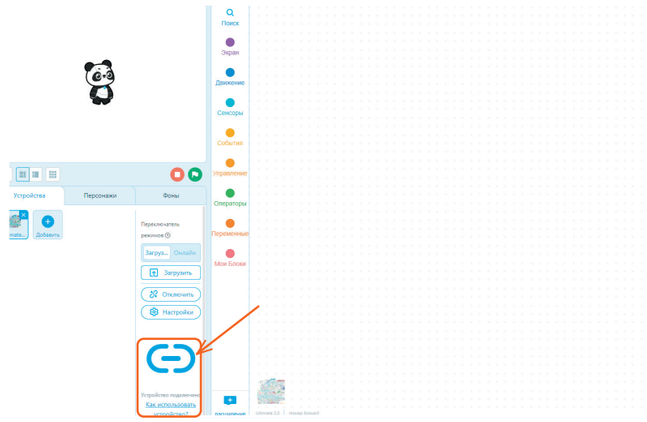


- Подключите контроллер к компьютеру с помощью кабеля *USB*, входящего в комплект. Нажмите кнопку «Подключение».



На экране появится приглашение к подключению с указанием *COM*-порта, к которому подключается контроллер. Нажмите кнопку «Подключение».

Убедитесь, что устройство подключилось.



Электронные модули, совместимые со средой программирования *Arduino IDE*, для подключения напрямую к контроллеру

5.2. *Bluetooth*-модуль

Модуль *Bluetooth* позволяет управлять роботом без проводов – с телефона, планшета или компьютера. Это удобно для дистанционного запуска программ, управления моторами и датчиками.

Bluetooth – это стандартный отраслевой протокол, который обеспечивает беспроводное подключение для множества устройств, включая компьютеры, принтеры, мобильные телефоны.

Беспроводной интерфейс с небольшим радиусом действия, получивший название *Bluetooth*, был разработан в 1994 году инженерами шведской компании *Ericsson.* Протокол получил свое название в честь Гарольда Синезубого – короля Дании, который примирил знатные семьи из Дании и Норвегии. Начиная с 1998-го развитием и продвижением данной технологии занимается организация *Bluetooth Special Interest Group (Bluetooth SIG)*, основанная компаниями *Ericsson, IBM, Intel, Nokia* и *Toshiba*. К настоящему времени список членов *Bluetooth SIG* включает более 13 тысяч компаний.

В зависимости от расстояния, *Bluetooth*-датчики делятся на три класса:

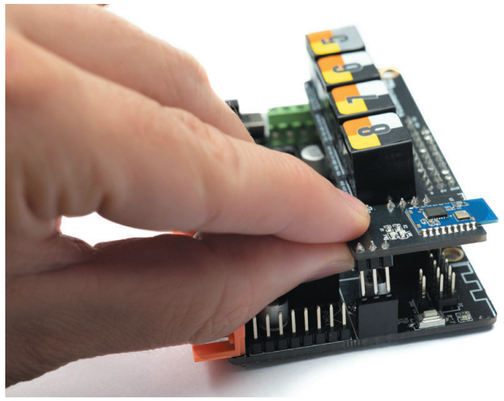
- Первый класс, способный поддерживать устойчивую связь на расстоянии 100–200 метров. В бытовых устройствах встречается редко и используется на промышленном оборудовании.
- Второй класс удерживает стабильную связь на расстоянии 10–20 метров. Такие датчики чаще всего установлены в смартфонах или планшетах.
- Третий класс наименее мощный и подходит для объединения устройств на расстоянии до пяти метров. Устанавливается на небольших гаджетах – фитнес-браслетах, умных часах и так далее.

Основные характеристики

Параметр	Значение
Тип подключения	*Bluetooth 4.0 (BLE – Low Energy* – с низким энергопотреблением)
Рабочее напряжение	3,3–5 В
Интерфейс связи	*UART* (подключение через 4-контактный разъем)
Дальность связи	до ~10 м (в зависимости от условий)

В конструкторе используется модуль *Bluetooth LE (Low Energy) v1.0.*

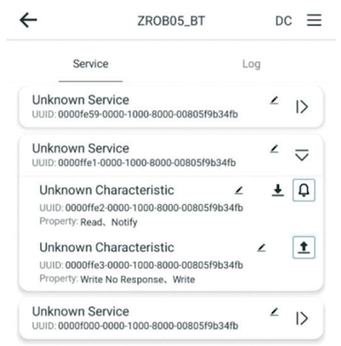
Для подключения и работы с контроллером, используя *Bluetooth*, необходимо вставить модуль в *SERIAL*-порт на плате контроллера.



Работа с датчиком:

- Подключите контроллер к компьютеру с помощью *USB*-кабеля.
- Установите на телефон приложение *BlueTooth Terminal eDebugger (Google Play).* Убедитесь, что ваш телефон поддерживает подключение по *Bluetooth LE,* и включите его в настройках телефона.
- Откройте приложение *eDebugger*, и во вкладке *BLE*, находящейся в нижнем левом углу экрана, выберите модуль *Bluetooth*, нажав на него в списке устройств по имени *ZROB05_BT.*

- Во вкладке *«Service»* выберите *«Unknown Service»* с *UUID*, начинающемся на *«0000ffe1»*. Раскройте вкладку с ним, и активируйте режим чтения сообщений, нажав значок колокольчика у *«Unknown Characteristic»* с *UUID*, начинающемся на *«0000ffe2»*. А у второго *«Unknown Characteristic»* нажмите значок со стрелкой вверх, отвечающей за запись сообщений.



- Перейдите во вкладку *«Log»*. Убедитесь, что вы получили сообщение *«Open notify 0xFFE2 successfully»* и *«Open Write 0xFFE3»*.
- Загрузите на плату контроллера приведенный скетч с помощью *Arduino IDE*.

```
void setup() {
  Serial.begin(115200);
  Serial3.begin(115200);

  Serial.println("!--- Write&Read BLE module ---!");
}

void loop() {
  if (Serial3.available()) {
    String dataRX = Serial3.readStringUntil('\n');
    dataRX.trim();
    Serial.print("\n[module->RX] str: ");
    Serial.print(dataRX);
    Serial.print(", hex: ");

    for (int i = 0; i < dataRX.length(); i++) {
      uint8_t b = (uint8_t)dataRX[i];
      Serial.print("0x");
      if (b < 16) Serial.print('0');
      Serial.print(b, HEX);
      Serial.print(" ");
    }
    Serial.println();
  }
```



```
    if (Serial.available()) {
      String dataTX = Serial.readStringUntil('\n');
      if (dataTX.length()) {
        Serial3.println(dataTX);
        Serial.print("\n[TX->module] ");
        Serial.println(dataTX);
      }
    }
  }
```

- В нижней части экрана приложения *«eDebugger»* во вкладке *«Log»* выберите режим ввода *«UTF-8»*. Введите любую цифру и нажмите *«Send»*. Проверьте, что отправленное сообщение появилось в мониторе порта *Arduino IDE*. Введите в поле текстового ввода монитора порта ответное сообщение, отправьте его, нажав *«Enter»*, и убедитесь, что вы получили его на своем телефоне.

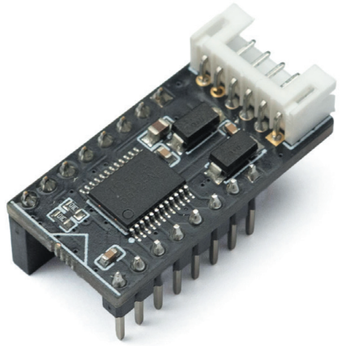
ВНИМАНИЕ! Все приведенные скетчи и библиотеки, необходимые для их работы, прилагаются на *USB*-накопителе в папке «Программное обеспечение».

5.3. Драйвер двигателя постоянного тока с энкодером

Драйвер двигателей – это электронная плата, которая управляет моторами робота. Она получает слабые сигналы от контроллера и преобразует их в мощные импульсы, чтобы моторы крутились с нужной скоростью и направлением.

Модуль драйвера двигателей на базе *TB6612FNG* может управлять двумя двигателями постоянного тока, одним двигателем с энкодером или одним шаговым двигателем.

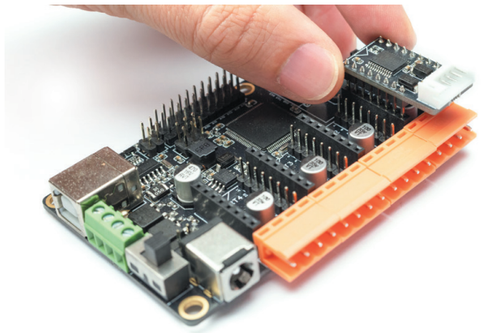
Этот компактный, но мощный драйвер моторов существенно улучшает возможности управления двигателями по сравнению с устаревшими решениями типа *L298N*.

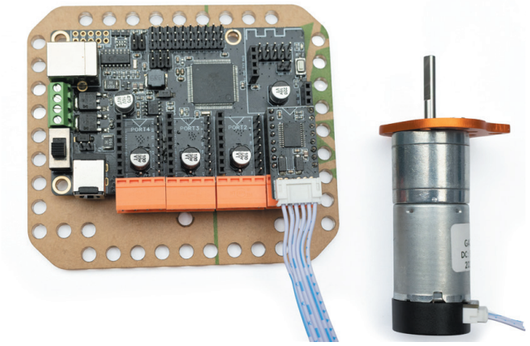


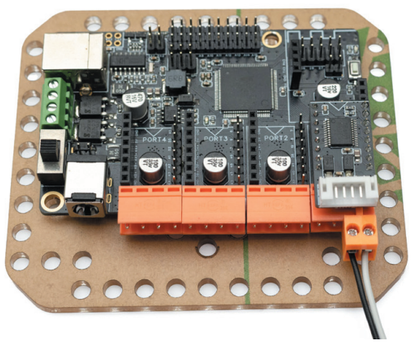
Основные характеристики

Параметр	Значение
Рабочее напряжение	2,5–13,5 В
Максимальный ток	1,2 А (3,2 А пиковый) на канал
КПД	до 95 % (меньше нагрева)
Встроенная защита от перегрева и КЗ	
Частота ШИМ	до 100 кГц
Функциональные возможности:	
Управление 2 моторами постоянного тока, одним двигателем с энкодером или одним шаговым двигателем	

Для начала работы с двигателями необходимо установить драйвер двигателя постоянного тока с энкодером в соответствующий слот на плате контроллера (*Port 1, Port 2, Port 3* или *Port 4*), подключить к нему двигатель с помощью 6-жильного кабеля для подключения моторов с энкодером и подключить к контроллеру внешний источник питания.







Двигатель постоянного тока без энкодера подключается с помощью двухжильного кабеля.

Подключите контроллер к вашему компьютеру с помощью *USB*-кабеля. Загрузите приведенный ниже пример скетча в контроллер. Переведите переключатель питания контроллера в положение «вкл».

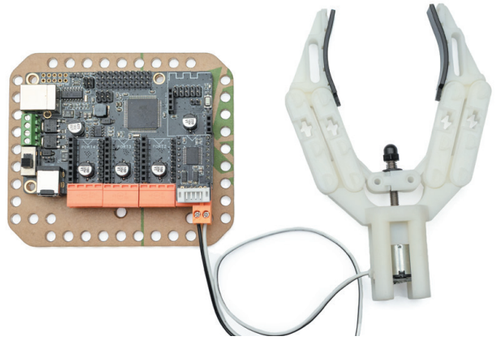
Таблица распределения пинов для управления двигателем постоянного тока без энкодера

	ШИМ A (упр. скоростью)	ШИМ B (упр. скоростью)	IN1_A	IN2_A	IN1_B	IN2_B
Port 1	11	12	32	33	34	35
Port 2	7	8	41	40	37	36
Port 3	6	9	48	47	43	42
Port 4	4	5	A2	A3	A4	A5

Пример кода управления двигателем постоянного тока без энкодера (например, захватного устройства)

С помощью ввода в монитор порта *Arduino IDE* команд можно управлять включением и направлением вращения двигателя:

- Введите символ *f* – двигатель вращается вперед.
- Введите символ *b* – двигатель будет вращаться в обратную сторону.
- Введите *s* – вращение двигателя остановится.



```
#define PWM_PIN 11    // ШИМ управление (11)
#define IN1_PIN 32    // Направление 1
#define IN2_PIN 33    // Направление 2

// Переменные управления
int motorSpeed = 0;   // Текущая скорость (-255 до 255)

void setup() {
  Serial.begin(115200);
// Настройка пинов мотора
pinMode(PWM_PIN, OUTPUT);
pinMode(IN1_PIN, OUTPUT);
pinMode(IN2_PIN, OUTPUT);
```

```
  // Остановка мотора при старте
  stopMotor();

  Serial.println("DC Motor Control Ready");
  Serial.println("Commands: f(forward), b(backward), s(stop), +(increase), -
(decrease)");
}

void loop() {
  // Обработка команд с Serial
  if (Serial.available()) {
    char command = Serial.read();

    switch(command) {
      case 'f': // Вперед
        setMotorSpeed(50);
        break;
      case 'b': // Назад
        setMotorSpeed(-50);
        break;
      case 's': // Стоп
        stopMotor();
        break;
      case '+': // Увеличить скорость
        setMotorSpeed(motorSpeed + 20);
        break;
      case '-': // Уменьшить скорость
        setMotorSpeed(motorSpeed - 20);
        break;
    }
  }
}

// Установка скорости мотора (-255..255)
void setMotorSpeed(int speed) {
  speed = constrain(speed, -255, 255); // Ограничение диапазона
  motorSpeed = speed;
  if (speed > 0) {
    // Вращение вперед
    digitalWrite(IN1_PIN, HIGH);
    digitalWrite(IN2_PIN, LOW);
    analogWrite(PWM_PIN, speed);
  }
```



```
  else if (speed < 0) {
    // Вращение назад
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, HIGH);
    analogWrite(PWM_PIN, -speed); // Берем модуль скорости
  }
  else {
    // Остановка
    stopMotor();
  }

  Serial.print("Motor speed set to: ");
  Serial.println(speed);
}

// Полная остановка мотора
void stopMotor() {
  digitalWrite(IN1_PIN, LOW);
  digitalWrite(IN2_PIN, LOW);
  analogWrite(PWM_PIN, 0);
  motorSpeed = 0;
  Serial.println("Motor stopped");
}
```

Для работы с мотором постоянного тока с энкодером используются дополнительные пины. Ниже приводится таблица распределения пинов для управления.

Таблица

	ШИМ (упр. скоростью)	IN1	IN2	Encoder_A	Encoder_B
Port 1	12	34	35	18	31
Port 2	8	37	36	19	38
Port 3	9	43	42	3	49
Port 4	5	A4	A5	2	A1

Пример кода управления DC двигателем с энкодером (Port 1)
В скетче реализовано плавное изменение скорости

```
// Конфигурация пинов
#define PWM_PIN 12     // ШИМ управление
#define IN1_PIN 34     // Направление 1
```

```
#define IN2_PIN 35     // Направление 2
#define ENCODER_A 18   // Канал A энкодера
#define ENCODER_B 31   // Канал B энкодера

// Параметры системы
volatile long encoderCount = 0;
const int encoderPPR = 16; // Импульсов на оборот вала мотора
float currentSpeedRPM = 0; // Текущая скорость в RPM
int targetSpeed = 0;      // Целевая скорость (-255 до 255)
int currentPWM = 0;       // Текущее значение ШИМ

// Для расчета скорости
long lastEncoderCount = 0;
unsigned long lastSpeedCalcTime = 0;

void setup() {
  Serial.begin(115200);

  // Инициализация пинов мотора
  pinMode(PWM_PIN, OUTPUT);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);

  // Инициализация энкодера
  pinMode(ENCODER_A, INPUT_PULLUP);
  pinMode(ENCODER_B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCODER_A), handleEncoder, CHANGE);

  // Остановка мотора при старте
  stopMotor();

  Serial.println("Система готова. Команды:");
  Serial.println("f - вперед, b - назад, s - стоп");
  Serial.println("+/- - изменить скорость");
}

void loop() {
  // 1. Расчет текущей скорости
  calculateSpeed();

  // 2. Управление мотором
  updateMotor();

  // 3. Обработка команд с Serial
  processSerialCommands();

  // 4. Вывод информации
  printDebugInfo();

  delay(10); // Небольшая задержка для стабильности
}

void handleEncoder() {
  // Обработчик прерывания энкодера
  if (digitalRead(ENCODER_A)) {
    encoderCount += (digitalRead(ENCODER_B)) ? +1 : -1;
  } else {
    encoderCount += (digitalRead(ENCODER_B)) ? -1 : +1;
  }
}

void calculateSpeed() {
  unsigned long currentTime = millis();
  unsigned long deltaTime = currentTime - lastSpeedCalcTime;

  if (deltaTime >= 100) {  // Обновляем скорость каждые 100мс
    long deltaCount = encoderCount - lastEncoderCount;
    currentSpeedRPM = (deltaCount * 60000.0) / (encoderPPR * deltaTime);
    lastEncoderCount = encoderCount;
    lastSpeedCalcTime = currentTime;
  }
}

void updateMotor() {
    if (targetSpeed != currentPWM) {
    // Плавное изменение скорости
    if (targetSpeed > currentPWM) currentPWM++;
    else if (targetSpeed < currentPWM) currentPWM--;

    // Установка направления и ШИМ
    if (currentPWM > 0) {
      digitalWrite(IN1_PIN, HIGH);
      digitalWrite(IN2_PIN, LOW);
      analogWrite(PWM_PIN, currentPWM);
    }
    else if (currentPWM < 0) {
```

```
      digitalWrite(IN1_PIN, LOW);
      digitalWrite(IN2_PIN, HIGH);
      analogWrite(PWM_PIN, -currentPWM);
    }
    else {
      stopMotor();
    }
  }
}

void processSerialCommands() {
  if (Serial.available()) {
    char command = Serial.read();

    switch(command) {
      case 'f': // Вперед
        targetSpeed = 150;
        break;
      case 'b': // Назад
        targetSpeed = -150;
        break;
      case 's': // Стоп
        targetSpeed = 0;
        break;
      case '+': // Увеличить скорость
        targetSpeed = constrain(targetSpeed + 20, -255, 255);
        break;
      case '-': // Уменьшить скорость
        targetSpeed = constrain(targetSpeed - 20, -255, 255);
        break;
    }
  }
}

void stopMotor() {
  digitalWrite(IN1_PIN, LOW);
  digitalWrite(IN2_PIN, LOW);
  analogWrite(PWM_PIN, 0);
  currentPWM = 0;
}

void printDebugInfo() {
  static unsigned long lastPrintTime = 0;
```



```
  if (millis() - lastPrintTime >= 500) {
    Serial.print("Цель: ");
    Serial.print(targetSpeed);
    Serial.print(" | Текущее: ");
    Serial.print(currentPWM);
    Serial.print(" | Энкодер: ");
    Serial.print(encoderCount);
    Serial.print(" | Скорость: ");
    Serial.print(currentSpeedRPM);
    Serial.println(" RPM");
    lastPrintTime = millis();
  }
}
```

Пример кода управления мотором постоянного тока с энкодером *(Port 1)* с помощью задания требуемого количества оборотов вала двигателя

```
// Конфигурация пинов
#define PWM_PIN 12
#define IN1_PIN 34
#define IN2_PIN 35
#define ENCODER_A 18
#define ENCODER_B 31

// Параметры системы
volatile long encoderCount = 0;
const int ENCODER_PPR = 16;// Импульсов на оборот вала мотора
const float GEAR_RATIO = 46.0;//Фиксированное передаточное отношение 1:46
float targetRevs = 0;    // Целевые обороты выходного вала

void setup() {
  Serial.begin(115200);

  // Настройка пинов
  pinMode(PWM_PIN, OUTPUT);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);

  // Настройка энкодера
  pinMode(ENCODER_A, INPUT_PULLUP);
  pinMode(ENCODER_B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCODER_A), handleEncoder, CHANGE);
```

```
  Serial.println("Система управления мотором с редуктором 1:46");
  Serial.println("Введите количество оборотов выходного вала (например: 2.5):");
}

void loop() {
  // Расчет текущих оборотов выходного вала
  float currentRevs = (float)encoderCount / (ENCODER_PPR * GEAR_RATIO);

  // Обработка команд
  if (Serial.available()) {
    targetRevs = Serial.parseFloat();
    startMovement(targetRevs);
  }

  // Вывод текущего положения
  static unsigned long lastPrintTime = 0;
  if (millis() - lastPrintTime > 200) {
    Serial.print("Текущее положение: ");
    Serial.print(currentRevs, 3);
    Serial.println(" оборотов");
    lastPrintTime = millis();
  }

  delay(10);
}

void handleEncoder() {
  // Обработчик прерывания энкодера
  if (digitalRead(ENCODER_A)) {
    encoderCount += digitalRead(ENCODER_B) ? +1 : -1;
  } else {
    encoderCount += digitalRead(ENCODER_B) ? -1 : +1;
  }
}

void startMovement(float revolutions) {
  // Расчет целевого количества импульсов
  long targetCount = revolutions * ENCODER_PPR * GEAR_RATIO;
  encoderCount = 0; // Сброс счетчика

  // Установка направления
  int direction = revolutions > 0 ? HIGH : LOW;
  digitalWrite(IN1_PIN, direction);
```



```
  digitalWrite(IN2_PIN, !direction);

  // Запуск мотора
  analogWrite(PWM_PIN, 150);
  Serial.print("Выполняю ");
  Serial.print(revolutions, 3);
  Serial.println(" оборотов выходного вала...");

  // Ожидание достижения цели
  while(abs(encoderCount) < abs(targetCount)) {
    delay(10);
  }

  // Остановка
  analogWrite(PWM_PIN, 0);
  Serial.println("Задание выполнено!");
}

void stopMotor() {
  analogWrite(PWM_PIN, 0);
  digitalWrite(IN1_PIN, LOW);
  digitalWrite(IN2_PIN, LOW);
}
```

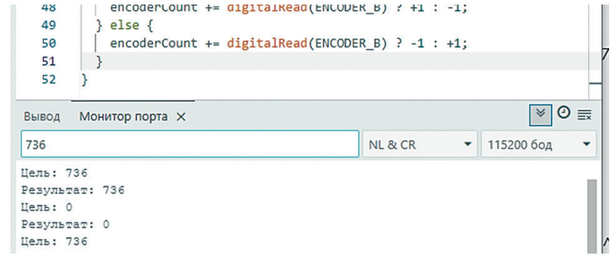
Пример кода программы управления двигателем постоянного тока с энкодером с помощью ввода требуемого значения количества импульсов энкодера

С помощью этой программы можно проворачивать вал двигателя с помощью задания количества импульсов энкодера. Пример приведен для *Port 1* контроллера.

```
#define PWM_PIN 12
#define IN1_PIN 34
#define IN2_PIN 35
#define ENCODER_A 18
#define ENCODER_B 31

volatile long encoderCount = 0;
long targetImpulses = 0;
const int SPEED = 50; // Фиксированная скорость (0-255)

void setup() {
  Serial.begin(115200);
```

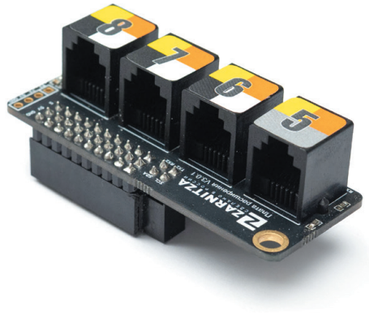
```
  pinMode(PWM_PIN, OUTPUT);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);

  pinMode(ENCODER_A, INPUT_PULLUP);
  pinMode(ENCODER_B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCODER_A), countEncoder, CHANGE);

  Serial.println("Введите количество импульсов:");
}

void loop() {
  if (Serial.available()) {
    targetImpulses = Serial.parseInt();
    Serial.print("Цель: ");
    Serial.println(targetImpulses);

    encoderCount = 0; // Сброс счетчика
    digitalWrite(IN1_PIN, targetImpulses >= 0 ? HIGH : LOW);
    digitalWrite(IN2_PIN, targetImpulses >= 0 ? LOW : HIGH);
    analogWrite(PWM_PIN, SPEED);

    while(abs(encoderCount) < abs(targetImpulses)) {
      // Ждем достижения цели
    }

    analogWrite(PWM_PIN, 0); // Мгновенная остановка
    Serial.print("Результат: ");
    Serial.println(encoderCount);
  }
}

void countEncoder() {
  if (digitalRead(ENCODER_A)) {
    encoderCount += digitalRead(ENCODER_B) ? +1 : -1;
  } else {
    encoderCount += digitalRead(ENCODER_B) ? -1 : +1;
  }
}
```


При вводе в строку монитора значения, например, 736 импульсов, получаем 1 оборот вала двигателя при значении передаточного числа редуктора 1:46.



5.4. Плата расширения

Этот модуль обеспечивает сопряжение цифровых и аналоговых выводов контроллера с унифицированными разъемами *RJ25* для подключения электронных модулей.



Основные характеристики

Параметр	Значение
Рабочее напряжение	5 В
Размеры	25 мм х 63 мм (ширина х высота)
Индикатор питания	один
Порты *6P6C RJ25:*	
Порт 5 – интерфейс коммуникационного модуля *RJ25* для добавления *Bluetooth* или *Wi-Fi*, для которого требуется *RJ25*	
Порт 6 – Порт 8 - двойные цифровые/аналоговые интерфейсы *RJ25* для добавления датчиков или модулей ввода/вывода *RJ25*	

Электронные модули, совместимые со средой программирования *Arduino-IDE*, оснащенные разъемами *RJ-25* для подключения к контроллеру:

5.5. Трехосевой акселерометр и гироскоп

Трехосевой акселерометр и гироскоп – идеальный модуль для определения движения и положения робота. Он включает в себя трехосевой акселерометр, трехосевой датчик угловой скорости и процессор движения, а также оснащен портами *I²C* для связи. Выполнен на базе *MPU6050.*

Акселерометр – датчик, измеряющий кажущееся угловое ускорение, которое является геометрической разницей между истинным угловым ускорением и ускорением силы гравитации. Показания датчика можно получать в м/с^2, или в *g* (количестве ускорений свободного падения). Предположим, что датчик неподвижен или движется равномерно, ось *Z* направлена вверх (детали на плате модуля

смотрят вверх). В таком случае проекция вектора силы гравитации не оказывает влияния на оси *XY*, их показания равны 0 м/с^2, но оказывает влияние на ось *Z* и направлена в противоположную сторону (к земле), значит

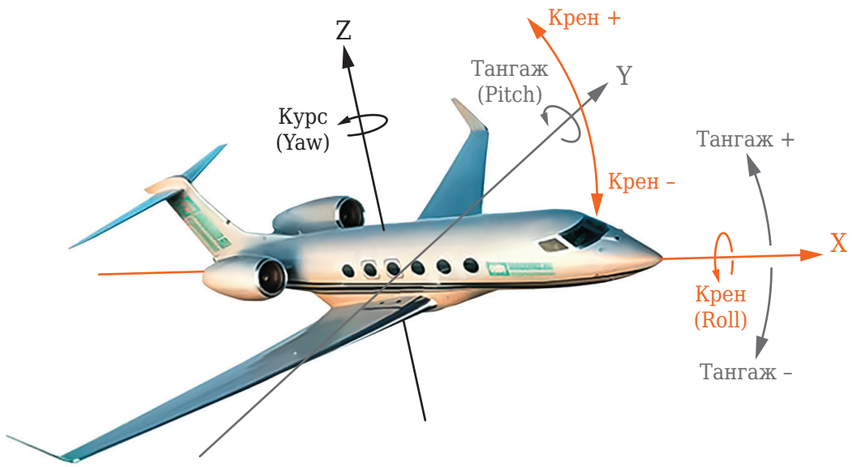
$$Z = 0 - g = 9{,}81\ м/с^2,$$

где 0 – проекция истинного ускорения модуля на ось *Z*,
g – ускорение свободного падения, взятое со знаком минус, так как его вектор противоположен направлению оси *Z*.

Если модуль наклонить, то влияние *g* на ось *Z* ослабнет, но увеличится на те оси, в направлении которых был наклонен модуль. Таким образом, зная проекцию вектора ускорения свободного падения на оси *X, Y, Z,* можно вычислить положение датчика относительно поверхности земли (углы: «крен» и «тангаж»), но только если датчик неподвижен или движется равномерно!

Гироскоп – датчик, измеряющий угловую скорость вокруг собственных осей. Показания датчика можно получать в °/с, или рад/с. Данный датчик способен определять воздействие момента внешней силы вокруг своих осей. Используя эти данные, можно компенсировать воздействие истинного ускорения на акселерометр, следовательно, используя акселерометр и гироскоп, получать положение этих датчиков относительно поверхности земли (углы: «крен» и «тангаж») во время их неравномерного движения.

Показания всех датчиков можно использовать как входные данные для фильтра Маджвика, Махони, Калмана или др., для получения кватернионов абсолютной ориентации устройства, из которых рассчитываются углы Эйлера («крен», «курс» и «тангаж»). Некоторые фильтры позволяют получать кватернионы используя данные только первых двух датчиков (без магнитометра), из которых также можно рассчитать углы Эйлера, но угол «курс» будет не истинным, а рассчитанным, он будет указывать не на север, а на изначальное направление датчика.



Углы Эйлера позволяют определить положение объекта в трехмерном (евклидовом) пространстве. Для определения положения используются 3 угла – «крен», «тангаж» и «курс».

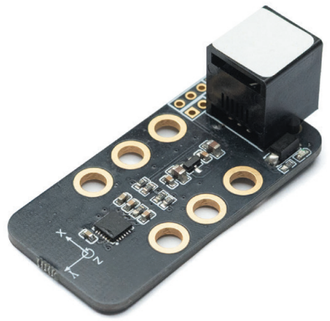
- Крен *(Roll)* – определяет наклон тела вокруг продольной оси *X*, например, крен самолета показывает, на сколько градусов вбок наклонились его крылья относительно земной поверхности.

Если самолет находится параллельно земле, то крен = 0°. Если самолет накренился (наклонился) вправо (левое крыло выше правого), то крен положительный (от 0° до 90°). Если самолет накренился (наклонился) влево (левое крыло ниже правого), то крен отрицательный (от 0° до -90°). Если самолет выполняет «бочку» (перевернулся), то крен ±180°.

- Тангаж *(Pitch)* – определяет наклон тела вокруг поперечной оси *Y*, например, тангаж самолета показывает, на сколько градусов поднят (или опущен) его нос относительно земной поверхности. Если самолет находится параллельно земле, то тангаж = 0°. Если самолет поднял нос вверх (кабрирует, взлетает), то тангаж положительный (от 0° до 90°). Если самолет опускает нос (пикирует, приземляется), то тангаж отрицательный (от 0° до -90°). Если самолет выполняет «мертвую петлю», то тангаж доходит до ±180° (полет назад вверх ногами).

- Курс *(Yaw)* – это самый простой для понимания угол Эйлера (еще его называют рыскание), он определяет направление вдоль земной поверхности. Например, для самолета курс определяет, куда самолет летит. Если самолет летит на север, то курс = 0°. Если самолет отклоняется от севера влево (на запад, или против часовой стрелки, если смотреть сверху), то курс отрицательный. Если самолет отклоняется от севера вправо (на восток, или по часовой стрелке, если смотреть сверху), то курс положительный.

Белая метка на интерфейсе этого модуля указывает на то, что это интерфейс связи *I²C*, и его следует подключать к порту, на котором есть белая метка.



Основные характеристики

Параметр	Значение
Рабочее напряжение	5 В постоянного тока
Рабочая температура	0–70 °C
Режим сигнала	связь *I²C*
2C device address 0x68	
Белая зона на модуле предназначена для соединения с металлическими балками	
Выводит цифровые данные расчетов слияния по 6 или 9 осям в таких форматах, как матрица вращения, кватернион и угол Эйлера	

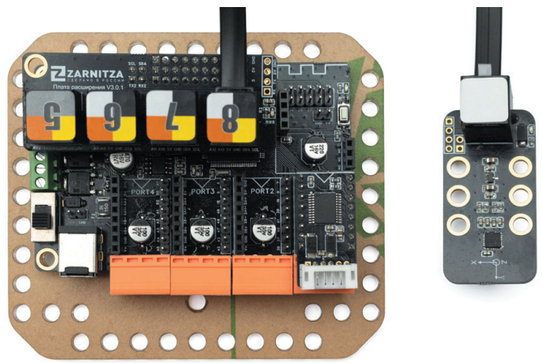
Параметр	Значение
Управляемый диапазон измерения 3-осевого датчика угловой скорости	±250 град/с, ±500 град/с, ±1000 град/с и ±2000 град/с
Диапазон контролируемых измерений 3-осевого акселерометра	±2 g, ±4 g, ±8 g и ±16 g
Механизм цифровой обработки движения (DMP) предназначен для снижения нагрузки на сложную интеграцию движения, синхронизацию датчиков и определение положения	
В модуль встроены алгоритмы калибровки датчиков разницы времени работы и магнитных датчиков	

Пример программы, демонстрирующей возможности модуля трехосевого акселерометра и гироскопа

Чтобы начать работать с трехосевым акселерометром и гироскопом, подключите его к контроллеру с помощью прилагаемого кабеля *6P6C RJ25.*

Подключите контроллер к вашему компьютеру с помощью *USB*-кабеля.

Загрузите приведенный ниже скетч в контроллер.



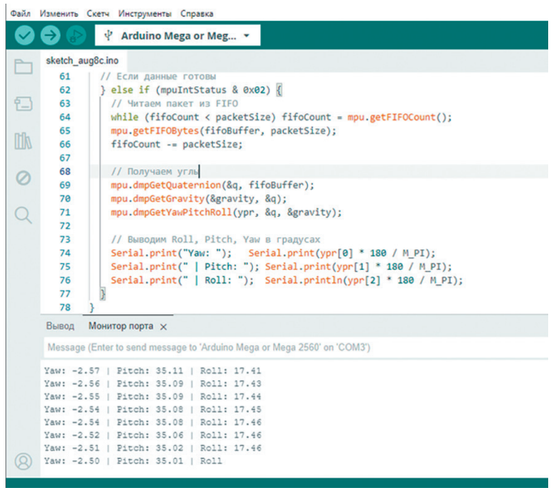

```
#include <Wire.h>
#include <MPU6050_6Axis_MotionApps20.h>

MPU6050 mpu;

// Переменные для DMP
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64]; // Буфер FIFO

// Структуры для данных
Quaternion q;           // Кватернион
VectorFloat gravity;    // Вектор гравитации
float ypr[3];           // [Yaw, Pitch, Roll] в градусах

//
================================================================================
=== НАСТРОЙКА =======================
//
```



```
void setup() {
  Serial.begin(115200);
  Wire.begin();

  mpu.initialize();
  devStatus = mpu.dmpInitialize();

  // Калибровочные смещения (замените своими значениями!)
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);

  if (devStatus == 0) {
    mpu.setDMPEnabled(true);
    dmpReady = true;
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    Serial.print("Ошибка инициализации DMP (код ");
    Serial.print(devStatus);
    Serial.println(")");
  }
}

// =====================================================
// ====================== ОСНОВНОЙ ЦИКЛ ================
// =============================================
void loop() {
  if (!dmpReady) return;

  // Проверяем готовность данных
  mpuIntStatus = mpu.getIntStatus();
  fifoCount = mpu.getFIFOCount();

  // При переполнении FIFO
  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
    mpu.resetFIFO();
    Serial.println("FIFO overflow!");

  // Если данные готовы
  } else if (mpuIntStatus & 0x02) {
    // Читаем пакет из FIFO
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
```

```
        mpu.getFIFOBytes(fifoBuffer, packetSize);
        fifoCount -= packetSize;

        // Получаем углы
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

        // Выводим Roll, Pitch, Yaw в градусах
        Serial.print("Yaw: ");   Serial.print(ypr[0] * 180 / M_PI);
        Serial.print(" | Pitch: "); Serial.print(ypr[1] * 180 / M_PI);
        Serial.print(" | Roll: ");  Serial.println(ypr[2] * 180 / M_PI);
    }
}
```

В последовательном мониторе *Arduino IDE* можно наблюдать значения углов по осям *X, Y* и *Z.*



Результат измерений в последовательном мониторе.

Калибровка

Модуль трехосевой акселерометр и гироскоп на базе *MPU-6050* требует калибровки для повышения точности измерений:

- Положите модуль на ровную поверхность.
- Запустите калибровочный скетч (например, из примеров библиотеки *MPU6050*).
- Запишите смещения *(offsets)* гироскопа и акселерометра, затем используйте их в коде.

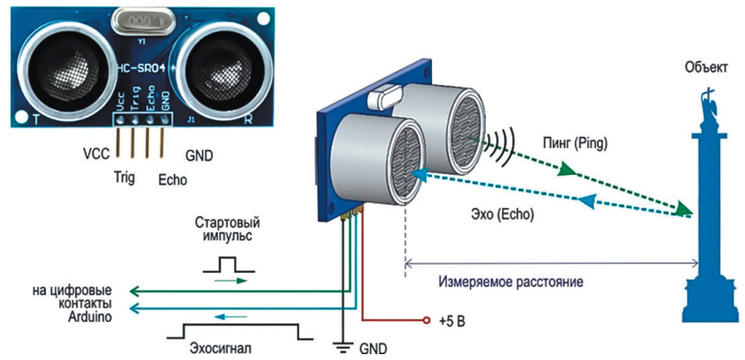
5.6. Датчик расстояния

Ультразвуковой датчик расстояния – определяет расстояние до объекта (например, до стены) от 3 до 400 см.

Ультразвуковой датчик посылает звуковую волну и измеряет время ее возврата (как летучая мышь).

Определяет расстояние до объектов точно так же, как это делают дельфины или летучие мыши. Он генерирует звуковые импульсы на частоте 40 кГц и слушает эхо. По времени распространения звуковой волны туда и обратно можно однозначно определить расстояние до объекта.

В отличие от инфракрасных дальномеров, на показания ультразвукового дальномера не влияют засветки от солнца или цвет объекта. Даже прозрачная поверхность будет для него препятствием. Но могут возникнуть трудности с определением расстояния до пушистых или очень тонких предметов. Поэтому определить расстояние, например, до кошки, выполнить на нем будет затруднительно.

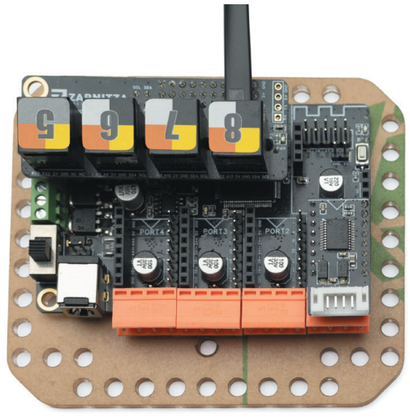


Ультразвуковой датчик конструктора оснащен двумя «глазами», один из которых является ультразвуковым передатчиком, а другой – ультразвуковым приемником. Ультразвуковой передатчик излучает ультразвуковую волну в определенном направлении и запускает отсчет времени. Ультразвуковая волна распространяется в воздухе, мгновенно возвращается обратно при встрече с препятствием и мгновенно останавливает отсчет времени, когда ультразвуковой приемник принимает отраженную волну. Скорость распространения звуковой волны в воздухе составляет 340 м/с. Используйте время, зафиксированное таймером, для расчета расстояния от точки передачи до препятствия, то есть

$$s = 340 \times \frac{t}{2}.$$

Для начала работы с ультразвуковым датчиком расстояния подключите его к контроллеру с помощью прилагаемого кабеля *6P6C RJ25*.

Подключите контроллер к вашему компьютеру с помощью *USB*-кабеля.



Загрузите приведенный ниже скетч в контроллер.

```
#define PIN_SONAR A12  // Один пин для TRIG и ECHO
void setup() {
  Serial.begin(9600);
  pinMode(PIN_SONAR, OUTPUT);
  Serial.println("Single-pin Ultrasonic Ready");
}

float getDistance() {
  // Переключаем пин в OUTPUT для TRIG
  pinMode(PIN_SONAR, OUTPUT);
  digitalWrite(PIN_SONAR, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_SONAR, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_SONAR, LOW);

  // Переключаем пин в INPUT для ECHO
  pinMode(PIN_SONAR, INPUT);
  long duration = pulseIn(PIN_SONAR, HIGH, 30000); // 30ms timeout (~5m)

  return duration * 0.017; // cm
}

void loop() {
  float dist = getDistance();

  if(dist <= 0 || dist > 400) {
```



```
    Serial.println("No object");
  } else {
    Serial.print(dist);
    Serial.println(" cm");
  }

  delay(200);
}
```

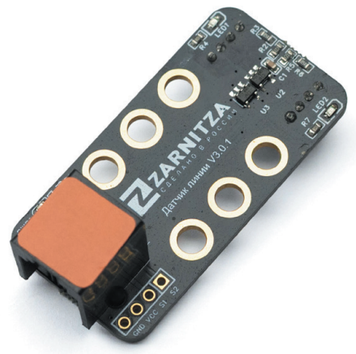
Таблица соответствия портов

	Trigger (T)& Echo (R)
Port 6	A9
Port 7	A11
Port 8	A12

Основные характеристики

Параметр	**Значение**
Рабочее напряжение	5 В постоянного тока
Угол измерения	30 градусов
Диапазон измерений	3 см – 400 см (с погрешностью менее 1 см)
Частота ультразвука	42 кГц
Режим управления	управление одним цифровым портом
Размер модуля	56 мм × 36 мм × 31 мм (Д × Ш × В)

5.7. Датчик линии



Распознает контрастные линии (например, черную полосу на белом полу). В качестве источника излучения использует инфракрасный свет – когда светодиод излучает инфракрасный свет, световой поток отражается от поверхности и попадает на фототранзистор, где преобразуется в электрический сигнал. Темный цвет отражает меньше света, светлый – больше. Используя несколько таких датчиков, робот определяет темную линию трассы и следует по ней.

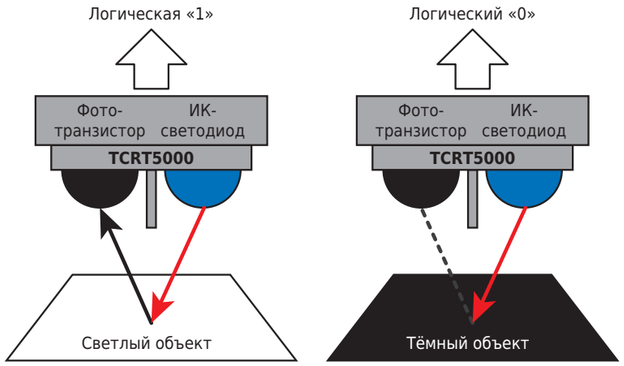


Таблица соответствия портов

	Сенсор 1	**Сенсор 2**
Порт 8	A13	A12
Порт 7	A10	A11
Порт 6	A8	A9

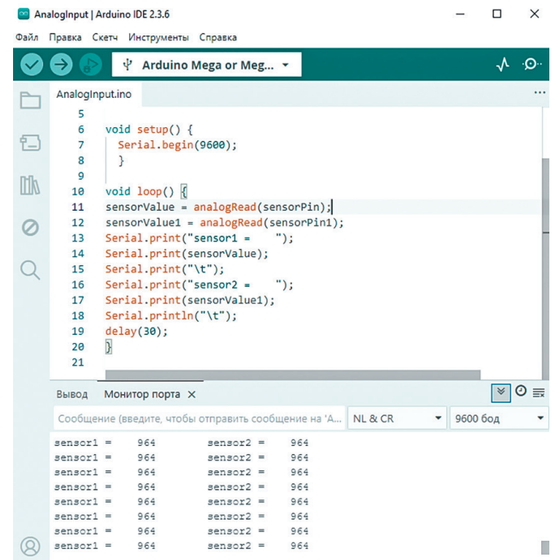
Подключите датчик линии к контроллеру с помощью прилагаемого кабеля *6P6C RJ25*.

Подключите контроллер к вашему компьютеру с помощью *USB*-кабеля.

Загрузите приведенный ниже скетч в контроллер (пример скетча для Порт 6):

```
int sensorPin = A8;   // sensor 1
int sensorPin1 = A9 ; // sensor 2
int sensorValue = 0;
int sensorValue1 = 0;

void setup() {
  Serial.begin(9600);
  }
void loop() {
sensorValue = analogRead(sensorPin);
sensorValue1 = analogRead(sensorPin1);
Serial.print("sensor1 =    ");
Serial.print(sensorValue);
Serial.print("\t");
Serial.print("sensor2 =    ");
Serial.print(sensorValue1);
Serial.println("\t");
delay(30);
}
```


В результате выполнения скетча в мониторе *Arduino IDE* можно наблюдать, как в случае нахождения сенсоров (и *Sensor1*, и *Sensor2*) над белой поверхностью контроллер получает максимальное значение сигнала с каждого датчика.



В случае, когда *Sensor1* находится над черной полосой, контроллер получает с этого сенсора значение 0.

Таким образом, необходимо написать управляющий скетч для контроллера, который должен отслеживать значения сигналов с *Sensor1* и *Sensor2*, и в случае схода мобильного робота с полосы (значение сенсора > 0) управлять двигателями робота для возврата на полосу, поворачивая в сторону сенсора, значение которого = 0.

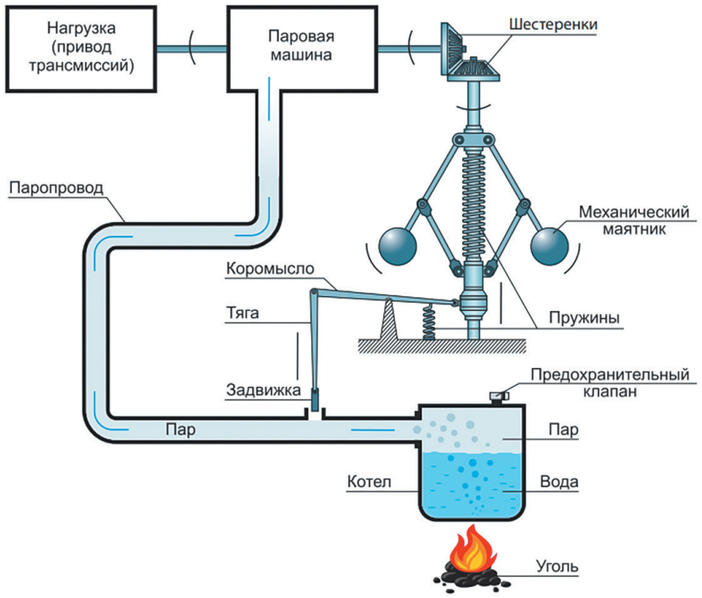
В данном случае принято говорить, что для управления мобильной платформой, для обеспечения движения платформы по черной линии, оборудованной датчиками линии, необходимо использовать принцип обратной связи.

Что такое обратная связь применительно к роботам?

Поскольку концепция обратной связи играет ключевую роль для любой автономной или самоуправляемой системы, например, робота, необходимо четко понимать ее суть. Основная идея здесь заключается в том, что функция управляющего органа (мозга животного или компьютера робота) должна состоять не только в формировании и передаче инструкций (в виде нервных импульсов, поступающих на мышцы, или электрических сигналов, идущих к электродвигателю), но и в восприятии и интерпретации тех данных, по которым можно судить о выполнении этих инструкций и о характере их воздействия на объект управления.

Таким образом, обратная связь в робототехнике – это отрицательная обратная связь, соединяющая датчики робота с виртуальными (алгоритм софта) или реальными датчиками источника сигнала управления роботом через каналы связи, приводы, пропорциональные датчики. Устройства обратной

связи представляют собой класс устройств, необходимых для работы в замкнутом контуре. Они передают сигнал обратно приводу или контроллеру движения для мониторинга операции или процесса и проверки правильности работы. На рисунке приведен пример использования одного из первых механизмов обратной связи для управления подачей пара и скорости вращения. Этот механизм называется регулятор Уатта.



Для датчика линии – это получение информации о наличии под ним черной полосы для движения, и выдача сигнала контроллеру в случае схода движущейся платформы с полосы. Контроллер же дает сигнал на двигатели для возврата на черную полосу.

Таким образом, система с отрицательной обратной связью сравнивает свое текущее состояние (например, скорость робота) с желаемым (заданной скоростью). Если есть отклонение – система корректирует работу (увеличивает/уменьшает мощность моторов). Или, в случае с движением по линии – в случае схода робота с линии дает сигнал двигателям для возврата.

Однако при этом могут возникать проблемы – если регулятор только «грубо» усиливает ошибку, возможны:

- Перерегулирование (робот резко «рыскает» вокруг нужной скорости или черной линии – и может съехать с нее).
- Медленное устранение ошибки (например, при изменении нагрузки, или в случае с черной линией – не успеть вернуться и потерять ее).

Решением в данном случае является использование ПИД-регулятора – он комбинирует три способа реакции на ошибку для точного и быстрого управления:

- П (Пропорциональный) – «Чем больше ошибка, тем сильнее реакция».
- И (Интегральный) – «Учитывает накопленную ошибку со временем» (устраняет постоянные отклонения).
- Д (Дифференциальный) – «Предсказывает будущую ошибку» (сглаживает резкие изменения).

То есть цель ПИД-регулятора – поддерживать заданный параметр (температуру, скорость, угол и т. д.) с максимальной точностью и минимальными колебаниями.

Например, задача мобильного робота двигаться с постоянной скоростью 50 см/с.

Датчик замеряет текущую скорость (40 см/с → ошибка = 10).

ПИД-регулятор:

П: увеличивает мощность моторов пропорционально ошибке (например, +20 %).

И: если скорость долго ниже заданной, плавно добавляет мощность (компенсирует трение).

Д: если скорость начала резко расти, снижает мощность заранее (чтобы не было «рывка»).

В итоге робот быстро выходит на 50 см/с и поддерживает скорость, даже если на пути подъем или изменился вес груза.

Или простыми словами, представьте, что вы рукой регулируете температуру воды в душе:

П: крутите кран сильнее, если вода слишком холодная.

И: если после первой попытки все еще холодно, продолжаете медленно добавлять горячую.

Д: если вода вдруг стала обжигающей, быстро уменьшаете напор, чтобы не получить ожог.

ПИД делает то же самое, но для технических систем – автоматически.

Адаптер 6P6C/RJ-25

Адаптер *6P6C/RJ25* предназначен для подключения модулей с интерфейсом *I²C*, в том числе сторонних производителей к модулям конструктора. Он преобразует разъем *6P6C RJ25* в 2 общих отдельных разъема *I²C* (которые содержат интерфейсы питания и интерфейс сигнала).

5.8. Двигатель постоянного тока с встроенным энкодером, тип 1

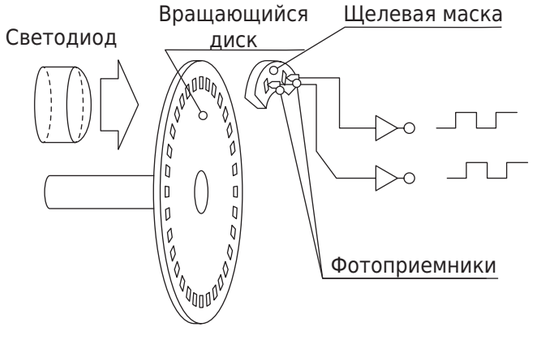
Этот мотор входит в комплект конструктора и используется для точного управления движением робота в робототехнических проектах.

Он включает в себя:

- *DC*-двигатель.
- Понижающий редуктор (для увеличения крутящего момента).
- Встроенный оптический энкодер.
- Стандартный разъем *RJ25* для подключения к плате контроллера.

Основные характеристики

Параметр	Значение
Номинальное напряжение	9,0 В (DC)
Передаточное число	1:75
Скорость вращения	86 RPM ±10 %



Оптический энкодер двигателя – это датчик, который используется для измерения вращательного движения вала двигателя и преобразования его в электрические сигналы. Он состоит из диска с чередующимися прозрачными и непрозрачными участками, установленного на валу, и оптического датчика, который считывает эти изменения. Таким образом, при вращении диска оптический датчик фиксирует прохождение или блокирование света, что позволяет определить скорость, направление и положение вращения вала.

5.9. Двигатель постоянного тока с встроенным энкодером, тип 2

В состав конструктора входит еще один тип двигателя. Он отличается только передаточным числом редуктора, а соответственно и скоростью вращения вала.

Основные характеристики

Параметр	Значение
Номинальное напряжение	9,0 В (DC)
Передаточное число	1:46
Скорость вращения	185 RPM ±10 %

5.10. Колеса всенаправленного движения

Колеса всенаправленного движения с роликами под углом 90° – это специализированные колеса, обеспечивающие перемещение робота в любом направлении, включая боковое скольжение, без необходимости разворота.

Их конструкция включает:

- Основное колесо – стандартный обод для передачи крутящего момента.
- Свободно вращающиеся ролики – расположены под прямым углом (90°) к плоскости основного колеса, обеспечивая боковое движение.

Такие колеса изготавливаются из ударопрочного пластика (например, *ABS*) с резиновыми или полиуретановыми роликами для улучшенного сцепления.

Колеса всенаправленного движения применяются в проектах, требующих сверхманевренности и точного контроля перемещения:

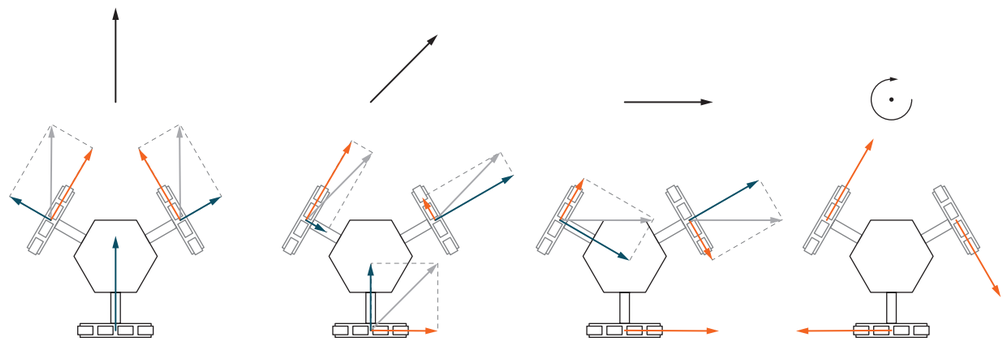
- Роботы для соревнований *(FIRST Robotics, RoboMaster)* – для быстрого обхода препятствий и тактического маневрирования.
- Промышленные тележки – перемещение грузов в узких пространствах без разворотов.
- Интерактивные и сервисные роботы – например, роботы-официанты или музейные гиды, где важно плавное движение в любом направлении.

Преимуществами использования таких колес в своих проектах являются:

- Полная свобода перемещения – движение вперед/назад, боком (влево/вправо), по диагонали и вращение на месте без изменения ориентации корпуса.
- Возможность параллельной парковки за счет комбинации вращения колес.
- Плавность и точность – мягкое скольжение роликов исключает рывки.
- Эффективность в ограниченном пространстве
- Минимальный радиус разворота (вплоть до нулевого).
- Не требуют места для маневров – например, движение вдоль стены без отворота.

На рисунке ниже приведены основные принципы управления и определения векторов скоростей колес для трехколесного шасси -для обеспечения необходимого направления движения робота.

Конструктивные элементы и элементы для сборки:



5.11. Гусеницы резиновые

Резиновые гусеницы в конструкторе предназначены для расширения функциональности робота и позволяют ему эффективно передвигаться по разным поверхностям.

Их основные преимущества:

- Улучшенное сцепление и проходимость

Гусеницы обеспечивают лучшее сцепление, чем колеса, особенно на скользких поверхностях (линолеум, плитка), неровных покрытиях (ковер, грунт, песок), наклонных плоскостях.

Подходят для соревновательных заданий, где требуется устойчивость (например, езда по трассе с препятствиями).

- Повышенная грузоподъемность

Распределяют вес робота более равномерно, чем колеса, что позволяет перевозить дополнительные грузы (датчики, манипуляторы), использовать тяжелые конструкции (например, подъемные механизмы).

- Универсальность в проектах

С гусеницами можно собрать танк или вездеход – для преодоления препятствий, транспортную платформу – для перемещения объектов.

5.12. Шины резиновые

Резиновые шины в конструкторе служат для улучшения подвижности и управляемости робота. Их функции и преимущества отличаются от гусениц, и выбор зависит от задач, которые должен выполнять робот.

5.13. Комплект для сборки захватного устройства (захват манипулятора, крепежные элементы)

Захватное устройство – это устройство, позволяющее роботу удерживать, переносить и манипулировать объектами.

В конструкторе он реализован на базе двигателя постоянного тока и конструкционных элементов.

Для движения клешней используется тип передачи винт-гайка – вращение винта (резьбового вала двигателя) преобразуется в линейное движение гайки. Гайка соединяется с подвижными частями клешни, заставляя их сжиматься/разжиматься. Программа управления для него пишется как для обычного двигателя постоянного тока без энкодера.

5.14. Батарейный отсек (6xAA)

Батарейный отсек (6xAA) – это источник питания для двигателей (сервомоторы, двигатели постоянного тока, шаговые двигатели), контроллера, датчиков (ультразвуковых, гироскопов) и т. д. Для его использования необходимо вставить в батарейный отсек 6 элементов AA (пальчиковые батарейки типоразмера AA).

При работе в стационарном режиме – написании и отладке программ – для питания контроллера можно использовать любой блок питания от сети переменного тока 220 В, обеспечивающий выходное напряжение 12 В (1-2 А).

5.15. Металлические структурные элементы, выполненные из алюминия с анодированным покрытием (балки, пластины, уголки)

Обеспечивают основу для создания каркаса/рамы робота, являются несущими элементами для крепления двигателей и датчиков. Используются для создания удлиненных конструкций (рычаги, манипуляторы).

Плоские пластины могут быть использованы в качестве площадки для монтажа электронных компонентов (контроллеров, батарей), создания плоских поверхностей (платформы, крепежные панели). Уголки (L-образные элементы) используют для соединения элементов под углом 90°, угловое крепление двигателей, создания жестких стыков между балками.

6. ПОРЯДОК РАБОТЫ С ОБОРУДОВАНИЕМ

Перед началом эксплуатации изделия необходимо ознакомиться с настоящим руководством и правилами техники безопасности.

Для работы с набором (программирования контроллера) необходимо подготовить компьютер, установив среду программирования *Arduino IDE*. Дистрибутив можно скачать отсюда:

https://www.arduino.cc/en/software/

Для работы с графической средой программирования *(mBlock)* необходимо скачать версию ПО для своего компьютера с сайта

https://mblock.cc/pages/downloads

Сборка

Сборка мобильной платформы производится с помощью прилагаемого набора инструментов и крепежных элементов в соответствии с прилагаемой Инструкцией по сборке.

Подключение модулей

Подключение модулей к контроллеру осуществляется с помощью прилагаемого набора проводов и кабелей. Для подключения модулей сенсоров используется 6-жильный кабель с разъемами *RJ-25,* для подключения двигателей – 6-жильный плоский кабель.

Перед включением питания необходимо загрузить в плату контроллера программное обеспечение из среды разработки *Arduino IDE* или *mBlock.*

Загрузка программного обеспечения в контроллер осуществляется с помощью кабеля USB, входящего в комплект.

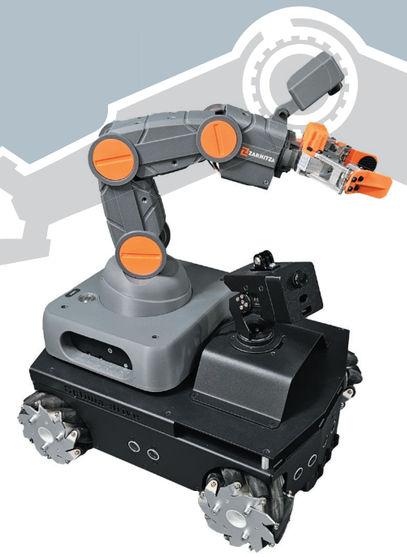
7. ПРАВИЛА ТЕХНИКИ БЕЗОПАСНОСТИ ПРИ РАБОТЕ

1. В целях предупреждения несчастных случаев напряжение выше 36 В следует считать опасным для жизни.
2. Составление, разборку или изменение схемы производят только с разрешения преподавателя.
3. Запрещается включать вновь составленную или измененную схему без предварительной проверки ее преподавателем.
4. Запрещается прикасаться к токоведущим частям и металлическим частям незаземленных электрических аппаратов, если на щите имеется напряжение.
5. Перед включением напряжения следует убедиться в том, что все регулирующие аппараты находятся в исходном положении. После отключения напряжения необходимо немедленно восстановить на всех регулировочных аппаратах исходное положение.
6. Перед включением напряжения следует предупредить об этом всех участников работы. Необходимо убедиться, что никому из них не угрожает опасность попасть под напряжение.
7. Если при прикосновении к какой-либо части оборудования ощущается напряжение, то необходимо прекратить работу, выключить ток и вызвать преподавателя.
8. Если до начала работы или в ходе работы обнаружена неисправность оборудования, следует прекратить работу, отключить напряжение и сообщить преподавателю или инженеру о неполадках в работе. Устранять неполадки собственными силами запрещается.
9. При работе с цепями переменного тока, содержащими конденсаторы, следует соблюдать особую осторожность, имея в виду возможность значительного возрастания напряжения на отдельных участках по сравнению с напряжением источника тока вследствие возможного явления резонанса напряжений.
10. Следует остерегаться вращающихся частей машины. В связи с этим запрещается находиться в лаборатории в свободной одежде, с шарфами или шалями, с распущенными волосами, незакрепленным галстуком. Запрещается приближаться к вращающейся муфте сцепления.
11. Запрещается приступать к выполнению работы до тех пор, пока преподавателем не будет установлено, что обучающемуся известны цель работы, метод ее выполнения, способ обращения с оборудованием, диапазон переменных величин и предполагаемые результаты.

8. ВОЗМОЖНЫЕ НЕИСПРАВНОСТИ И МЕТОДЫ ИХ УСТРАНЕНИЯ

Неисправность	Возможная причина неисправности	Способы устранения неисправности
Нет связи контроллера и ПК	Плохой контакт *USB*-кабеля	Отключить и снова включить кабель *USB*
Нет связи контроллера и ПК	Ошибка в работе драйверов	Проверить состояние драйверов, переустановить
Нет связи контроллера и датчика	Плохой контакт кабеля	Отключить и снова включить кабель

ZARNITZA ROBOTICS
Разрабатываем и производим высокотехнологичное учебное оборудование для любых специальностей



УП6153

Робот-манипулятор с колёсами всенаправленного движения Optima Pro + Optima Drive
Приказ 838 Минпросвещения РФ

УП6161

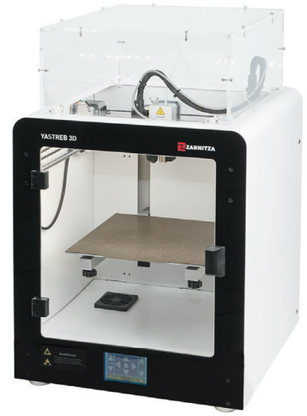
Комплект для изучения операционных систем реального времени и систем управления автономных мобильных роботов Optima Drive
Приказ 838 Минпросвещения РФ

УП6340

3D-сканер ручной профессиональный Yastreb 3D
Приказ 838 Минпросвещения РФ

УП6338

3D-принтер профессионального качества Zarnitsa Yastreb 3D
Приказ 838 Минпросвещения РФ



Телефон 8-800-775-37-97
www.zarnitza.ru, zakaz@zrnc.ru

РОБОТОТЕХНИКА

ZARNITZA

8 (800) 775-37-97
zakaz@zrnc.ru

RU СДЕЛАНО В РОССИИ