ZARNITZA
ROBOTICS

Производственное объединение «Зарница»

РАСШИРЕННЫЙ РОБОТОТЕХНИЧЕСКИЙ НАБОР

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»

РАСШИРЕННЫЙ РОБОТОТЕХНИЧЕСКИЙ НАБОР

Руководство по эксплуатации

2025

ОГЛАВЛЕНИЕ

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

1. НАЗНАЧЕНИЕ

Расширенный робототехнический набор предназначен для сборки и программирования автономного мобильного робота на колесах всенаправленного движения.

Мобильный робот с модулем обработки технического зрения – это современное образовательное устройство, предназначенное для обучения студентов и школьников основам робототехники, программирования и искусственного интеллекта. Этот робот сочетает в себе мобильность, простоту в управлении и расширенные возможности обработки визуальной информации благодаря встроенной камере и модулю обработки технического зрения.

Автономный мобильный робот – это платформа на базе робототехнического контроллера, оснащенная моторами, датчиками и камерой. Модуль обработки технического зрения позволяет роботу «видеть» окружающую среду: распознавать объекты, читать QR-коды, определять цвета и формы, а также отслеживать движение. Это делает его мощным инструментом для реализации различных задач в области компьютерного зрения и автоматизации.

Робототехнический набор используется для:

- изучения основ программирования: обучающиеся могут писать программы для управления роботом, реализуя алгоритмы навигации, распознавания объектов и взаимодействия с окружающей средой.
- обучения робототехнике: создание и настройка маршрутов, решение задач по обходу препятствий, выполнение различных команд по заранее созданным алгоритмам.
- изучения компьютерного зрения: использование камеры для распознавания цветов, форм, чтения QR-кодов или штрихкодов – отличная практика для понимания технологий обработки изображений.

2. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Параметр		Значение
Исполнение		Мобильное
Материал деталей для сборки платформы робота		Листовой алюминий, пластик
Внутренний диаметр стоек, мм		3
Аккумуляторная батарея 3,7 В		
	Емкость, мА/ч	1000
	Номинальное выходное напряжение, В	3,7
Аккумуляторная батарея 7,4 В		
	Емкость, мА/ч	2000
	Номинальное выходное напряжение, В	7,4
Зарядное устройство аккумуляторных батарей 12,6 В		
	Выходной ток, А	1
	Номинальное выходное напряжение, В	12,6
	Входное напряжение, В	220

Параметр		Значение
Зарядное устройство аккумуляторных батарей 8,4 В		
	Выходной ток, А	1
	Номинальное выходное напряжение, В	8,4
	Входное напряжение, В	220
Лазерный сканирующий дальномер		
	Угол сканирования, градус	360
Робототехнический контроллер		
	Поддерживаемые среды разработки	Arduino IDE, Espruino IDE, Scratch IDE
	Поддерживаемые языки программирования	C\C++, JavaScript, Python (или аналоги)
	Электропитание контроллера, В	6 – 14
Модуль обработки технического зрения		
	Интерфейсы	Wi-Fi 802.11ac (2,4 / 5 ГГц); Bluetooth 5.0; USB 2.0; USB 3.0; USB Type-C; MicroSD; UART; UART с настраиваемым напряжением 3,3 В или 5 В; I^2C; SPI с настраиваемым напряжением 3,3 В или 5 В; I^2S; USB; Ethernet
	Количество ядер процессора, шт.	4
	Частота процессора, ГГц	2,4
	Оперативная память, МБ	8000
	Порт питания +5 В, шт.	2
	Порт GND (земля), шт.	6
	Количество портов для подключения внешних цифровых и аналоговых устройств, шт.	40
Габариты робота в сборе (Д×Ш×В), мм, не более		330 × 190 × 160
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. КОМПЛЕКТНОСТЬ

Наименование		Кол-во, шт.
Расширенный робототехнический набор		
	Структурные компоненты	20
1	Детали из листового алюминия для сборки платформы робота, шт.	5
	в том числе:	
	Корпус мобильной платформы, шт.	1
	Скоба крепления аккумуляторной батареи, шт.	1
	Верхний кронштейн, шт.	1
	Передний кронштейн, шт.	1
	Задний кронштейн, шт.	1
2	Пластиковые детали для сборки платформы робота, шт.	2
	в том числе:	
	Крепление датчиков универсальное, шт.	1
	Крепление аккумулятора, шт.	1
3	Привод постоянного тока, шт.	4
	каждый из которых имеет в своем составе:	
	Привод на базе двигателя постоянного тока, шт.	1
	Понижающий редуктор, шт.	1
	Магнитный модуль для измерения числа оборотов вращения вала, шт.	1
4	Колесо всенаправленного движения (колесо Илона), шт.	4
	каждый из которых имеет в своем составе:	
	Соединительная муфта для колеса, шт.	1
5	Робототехнический контроллер, шт.	1
	имеющий в своем составе:	
	Система ориентации в пространстве, шт.	1
	Программируемая кнопка, шт.	4
	Программируемый светодиод, шт.	1
	Программируемая система звуковой индикации, шт.	1
	Порты для подключения интеллектуальных сервоприводов, шт.	2
	Разъем для подключения аккумуляторного модуля, шт.	1
	Порты для подключения двигателей постоянного тока, шт.	4
	Порты для подключения устройств по последовательному интерфейсу, шт.	1
	Порты для подключения внешних цифровых или аналоговых устройств, шт.	12
6	Контроллер сопряжения датчиков, шт.	1
	имеющий в своем составе:	
	Разъем для подключения блока питания, шт.	1
7	Латунные стойки, шт.	16
8	Комплект винтов (под внутренний шестигранник) и гаек для сборки платформы робота, шт.	1
9	Модуль технического зрения в сборе, шт.	1
	имеющий в своем составе:	
	Монокулярная камера, шт.	1
	Сервоприводы, шт.	2

Наименование			Кол-во, шт.
	10	Модуль обработки технического зрения, шт.	1
	11	Модуль охлаждения, шт.	1
	12	Модуль слежения за линией, шт.	2
		каждый из которых имеет в своем составе:	
		3х-пиновый разъем, конструктивно защищенный от неправильного подключения шлейфа, шт.	1
	13	Датчик касания, шт.	1
	14	Лазерный сканирующий дальномер, шт.	1
		имеющий в своем составе:	
		USB – UART преобразователь, шт.	1
	15	Аккумуляторный модуль, шт.	1
		имеющий в своем составе:	
		Трехпроводный разъем выхода, шт.	1
		Отсек для подключения аккумуляторов, вмещающий в себя три аккумулятора, шт.	1
		Светодиодный индикатор уровня заряда, шт.	1
		Сегменты индикатора, шт.	4
	16	Аккумуляторная батарея 3,7 В, шт.	3
	17	Аккумуляторная батарея 7,4 В, шт.	1
	18	Комплект соединительных проводов, шт.	1
		имеющий в своем составе:	
		Кабель USB-A – USB-C, шт.	1
		Кабель USB-C – USB-C, шт.	1
		Кабель 3Pin, шт.	3
		Кабель 4Pin, шт.	1
		Кабель питания, шт.	1
		Кабель подключения приводов, шт.	4
	19	Разъем для подключения внешнего питания, шт.	1
	20	Нейлоновый фиксатор, шт.	1
		SD-карта, шт.	1
		Зарядное устройство аккумуляторных батарей 12,6 В, шт.	1
		Зарядное устройство аккумуляторных батарей 8,4 В, шт.	1
		Кардридер, шт.	1
		Инструменты для сборки, шт.	4
		USB-накопитель, шт.	1
Паспорт			1
Руководство по эксплуатации			1
Инструкция по сборке			1

4. ОБЩИЙ ВИД

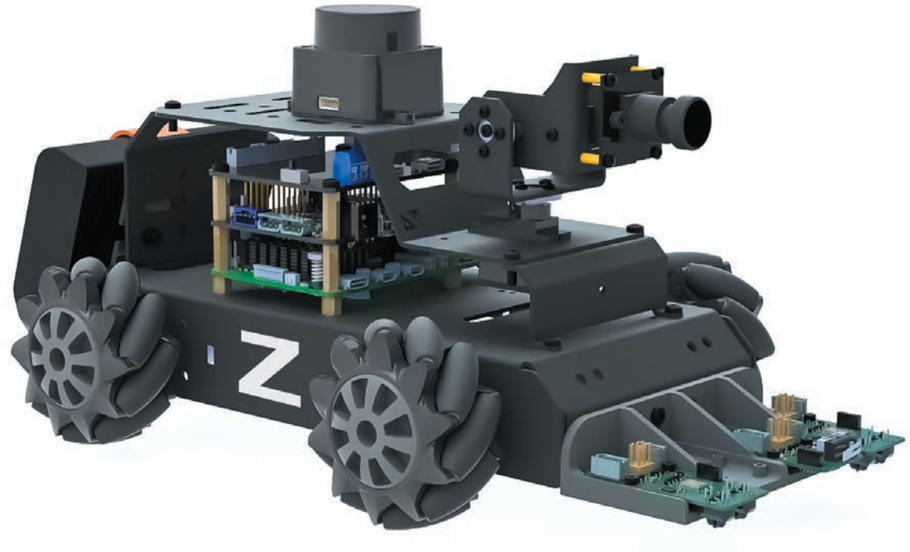


Рисунок 1. Общий вид (в сборе)

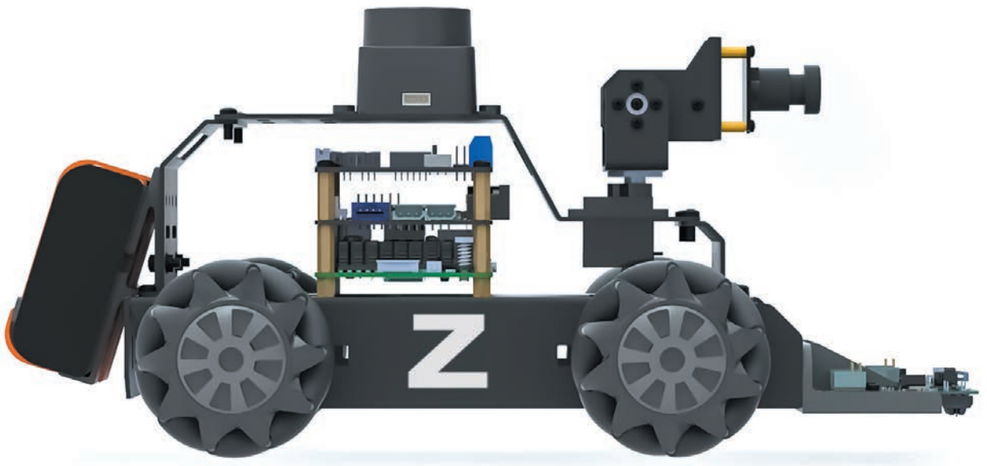


Рисунок 2. Общий вид (в сборе)

5. ОПИСАНИЕ И ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ НАБОРА

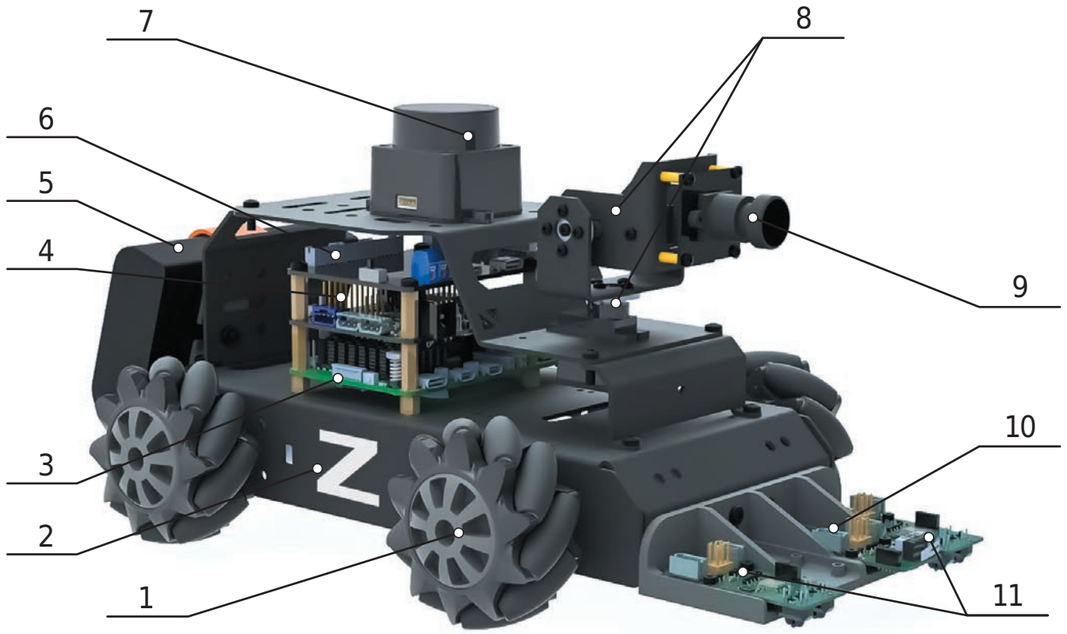


Рисунок 3. Состав набора:
1 – Колеса всенаправленного движения (колесо Илона); 2 – Корпус мобильной платформы;
3 – Модуль обработки технического зрения; 4 – Контроллер сопряжения датчиков;
5 – Аккумуляторный модуль; 6 – Робототехнический контроллер;
7 – Лазерный сканирующий дальномер; 8 – Модуль технического зрения в сборе;
9 – Монокулярная камера; 10 – Датчик касания; 11 – Модуль слежения за линией.

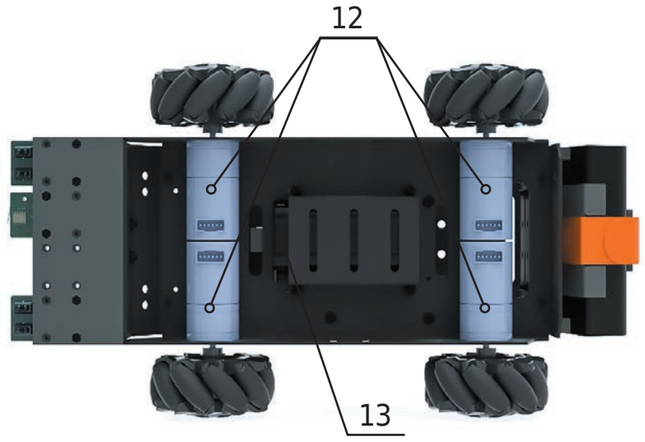


Рисунок 4. Состав набора:
12 – Привод постоянного тока; 13 – Аккумуляторная батарея 7,4 В.

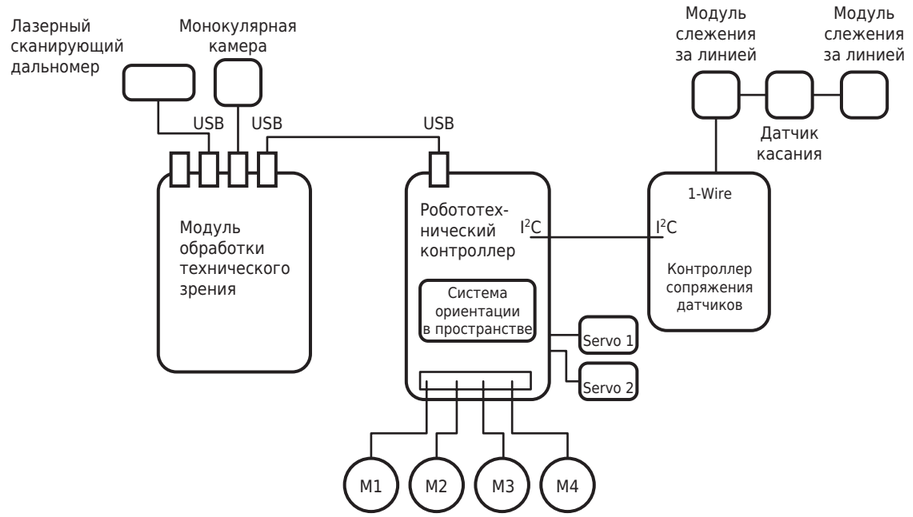


Рисунок 5. Функциональная схема собранного устройства

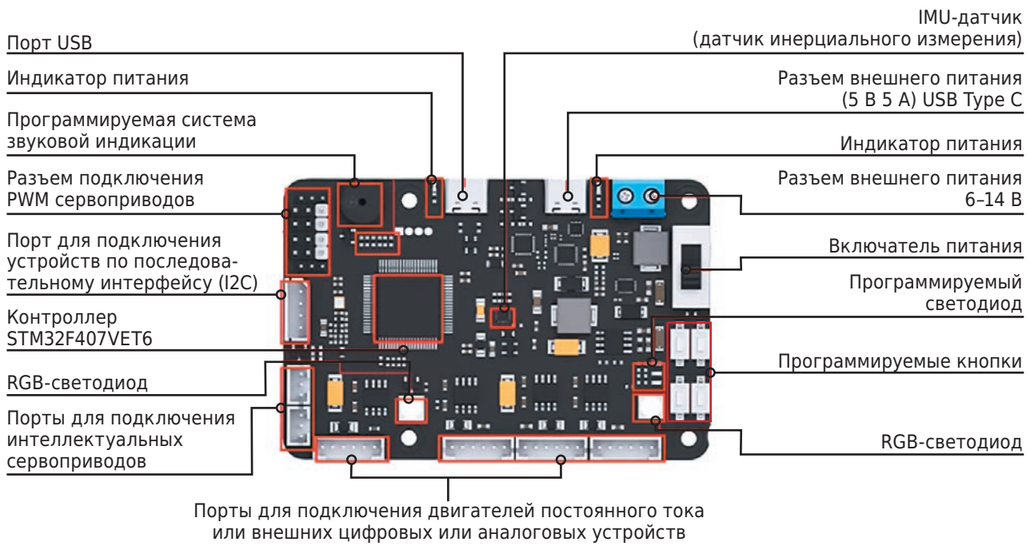
Состав набора

• **Привод постоянного тока с понижающим редуктором, магнитным модулем для измерения числа оборотов вращения вала.** Используются 4 привода. Коллекторный (щёточный) мотор-редуктор компактного размера, собран на однофазном двигателе постоянного тока. На вал двигателя установлен цельнометаллический соосный редуктор, состоящий из нескольких ступеней шестерёнчатых передач. Редуктор влияет на изменения соотношений крутящего момента и скорости вращения между валом мотора и валом редуктора, снижая скорость и обеспечивая необходимое усилие на валу редуктора.

Параметр	Значение
Номинальное напряжение	7,4 В постоянного тока
Скорость холостого хода	450 об/мин
Максимальная мощность	4,8 Вт
Передаточное отношение	1:20

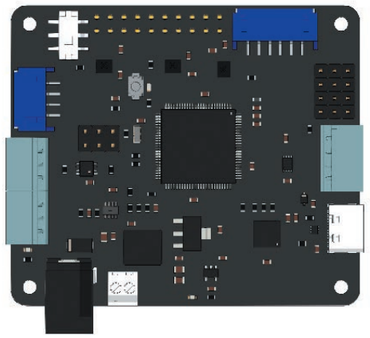
• **Колеса всенаправленного движения (колесо Илона)** – диаметром 65 мм. Эти колеса имеют уникальную конструкцию с роликами, расположенными под углом к оси вращения. Путём изменения направления и скорости вращения отдельных колёс можно заставить машину на Илоновых колёсах двигаться в любом направлении – перпендикулярно движению вперёд-назад, по диагонали под любым углов в зависимости от скорости каждого отдельного мотора. При этом трения скольжения между роликами и опорной поверхностью практически не будет. Конструкция колёс Илона позволяет вращаться на месте при минимальной силе трения и низком вращательном моменте.

• **Робототехнический контроллер** – взаимодействует с периферийными устройствами, двигателями, модулем технического зрения. Разгружает модуль обработки технического зрения от рутинных задач, позволяя модулю обработки технического зрения выделять аппаратные ресурсы на выполнение задач по управлению мобильной платформой в целом и обработкой видео. Робототехнический контроллер выполнен на базе микроконтроллера *STM32F407VET6(100PIN)*. Содержит в себе систему ориентации в пространстве на базе *IMU* сенсора – 6DOF, программируемые кнопки, светодиоды и программируемую систему звуковой индикации.



Параметр	Значение
Микроконтроллер	*STM32F407VET6(100PIN)*
Система ориентации в пространстве	*IMU* датчик *QMI8658* – это шести осевой датчик инерциального измерения *(IMU)*, который включает в себя трехосевой гироскоп и трехосевой акселерометр. Он предназначен для точного измерения углового положения, скорости и ускорения,

Параметр	Значение
	и используется в различных приложениях, таких как дроны, робототехника, игровые контроллеры и устройства виртуальной реальности. *QMI8658* обеспечивает низкое энергопотребление, высокую стабильность и чувствительность.
Порты для подключения интеллектуальных сервоприводов	2 разъема для подключения интеллектуальных сервоприводов с питанием 6-12В
Сервоприводы с функцией ШИМ	4 разъема для подключения 4-х сервоприводов
Драйверы двигателей	на базе *SA8870C*
Порты для подключения двигателей постоянного тока	4 порта
Программируемая кнопка	4 кнопки
Программируемый светодиод	1 светодиод
Программируемая система звуковой индикации	1 зуммер
Электропитание контроллера	6–14 В



• **Контроллер сопряжения датчиков** – служит для подключения модуля слежения за линией и датчика касания. Подключается к робототехническому контроллеру по интерфейсу *I²C*. Датчики подключаются к контроллеру сопряжения с помощью трехжильного кабеля по интерфейсу *1-Wire UART* (протокол связи по одному сигнальному кабелю). Контроллер поддерживает подключение датчиков *1-Wire UART* в шинную топологию.

Для связи с контроллером сопряжения датчиков можно использовать интерфейс *SPI*, с помощью которого модуль обработки технического зрения на базе микрокомпьютера *Raspberry Pi 5* физически связан с контроллером сопряжения датчиков.

• **Модуль обработки технического зрения** – обеспечивает возможность разработки и установки пользовательского программного обеспечения, использующего аппаратные вычислительные ресурсы, память, видео данные и интерфейсы модуля средствами встроенной в него операционной системы.

Модуль обработки технического зрения построен на чипе, включающем в себя четырёхъядерный 64-битный процессор *Cortex-A76 (ARMv8.2-A)* тактовой частотой 2,4 ГГц и графический процессор *VideoCore VII* тактовой частотой 800 МГц, оснащен 8 ГБ памяти *LPDDR4X-4267 SDRAM,* которая делится между *CPU* и *GPU*. Способен декодировать видео *H.265/HEVC* (до 4Kp60) и *H.264*.

Модуль беспроводной связи поддерживает двухдиапазонный *Wi-Fi 802.11ac* (2,4 и 5 ГГц) и протокол *Bluetooth 5.0 с BLE*.

Параметр	Значение
Однокристальная система	SoC Broadcom BCM2712
Центральный процессор	4× ARM Cortex-A76 (64 бита)
Тактовая частота CPU	2,4 ГГц
Графический процессор	VideoCore VII GPU с поддержкой API OpenGL ES 3.1, Vulkan 1.2
Тактовая частота GPU	800 МГц
Максимальное разрешение	2160p (60 Гц)
Оперативная память	8 ГБ LPDDR4X-4267 SDRAM
Беспроводная связь	
	• Wi-Fi 802.11ac (2,4 / 5 ГГц)
	• Bluetooth 5.0 / Bluetooth Low Energy (BLE)
Контроллер периферии	RP1 I/O
Аппаратные интерфейсы	
	• 2× micro-HDMI 2.0
	• 2× USB 2.0
	• 2× USB 3.0 (5 Гбит/с)
	• 2× MIPI (1,5 Гбит/с), 22 пина с шагом 0,5 мм
	• 1× PCIe 2.0
	• 1× Ethernet (1 Гбит/с)
	• 1× microSD (с поддержкой SDR104)
	• 40× GPIO
Разъём питания	USB Type-C
Напряжение питания	5 В
Максимальный ток потребления	5 А
Размеры	85×56×17 мм

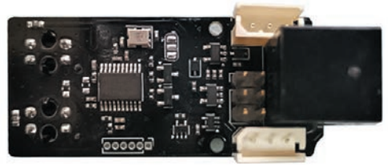
- **Модуль технического зрения в сборе.** Робот оснащен модулем технического зрения в сборе, состоящего из монокулярной камеры и двух сервоприводов. Модуль обеспечивает полный обзор на 360 градусов без слепых зон.

Параметр	Значение
Разрешение	300W(640*480)
Светочувствительный контроллер	GC0308 (gcoreinc)
Тип камеры	HS-256-650 (HS)
Аппертура	2.0
Фосус	1,7 мм
Угол фокусного поля зрения	170°
Интерфейс	USB

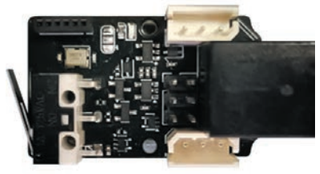
- **Сервоприводы.** В модуле технического зрения используются сервоприводы *LFD-01.*

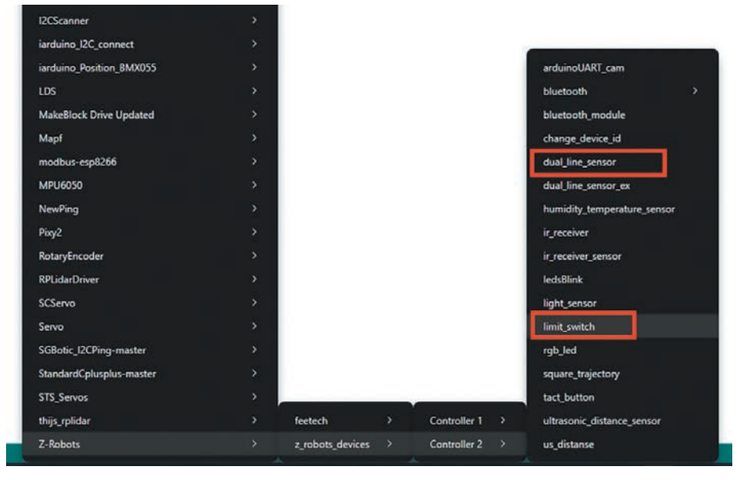
Параметр	Значение
Скорость вращения	≤ 0,11 сек/60° при 6 В
Угол вращения	0° - 180°
Интерфейс	*PWM*
Шестерни редуктора	Пластик
Рабочее напряжение	4,8 – 6 В
Максимальный крутящий момент	≥ 1,4 кг.см при 6 В
Скорость без нагрузки	≤ 0,11 сек/60° при 6 В
Размер	22,3x12,0x23,2 мм

- **Модуль слежения за линией** – датчик линии, совместимый со средой программирования *Arduino IDE.* Модуль имеет 3-пиновый разъем, конструктивно защищенный от неправильного подключения шлейфа.



- **Датчик касания**. Модули слежения за линией и датчик касания подключаются к контроллеру сопряжения датчиков по интерфейсу *1-wire UART* используя прилагаемую на электронном носителе библиотеку *Z-Robots.h.* Схема подключения датчиков имеет схему шинной топологии, каждый датчик имеет свой уникальный *ID.*





На изображении представлена среда программирования *Arduino IDE* с установленной библиотекой *Z-Robots.h.* В ней содержатся примеры программ работы с датчиками.

- **Лазерный сканирующий дальномер.** Предназначен для использования в помещениях и применяется в таких областях, как навигация и обход препятствий для домашних роботов-уборщиков и роботов для обслуживания помещений, исследования и обучение роботов *ROS*, а также статическое сканирование для измерения объема.

Лазерный сканирующий дальномер *MS200* имеет небольшие габариты (33 x 37 мм). В основу работы устройства положен принцип измерения *TOF*, который расшифровывается как *Time of Flight*: высокоэффективное и современное решение для навигации посредством дистанционной картографии и 3D-визуализации.

Максимальная дальность (радиус) сканирования может достигать 12 метров при минимально возможной дистанции 3 см. Взаимодействует с роботами по алгоритмам *ROS1, ROS2*. Частота сканирования может регулироваться в диапазоне от 7 до 15 Гц. Частота дискретизации поддерживается на высоком уровне: 4500 раз в секунду. Модуль устойчив к внешнему освещению до 30 килолюкс, что является превосходным конкурентным преимуществом устройства. Совместим с модулем обработки технического зрения.

Перемещение лазерного дальномера по кругу происходит с помощью движения встроенного в модуль бесщеточного двигателя с угловым разрешением 0,8 градуса. Устройство широко применятся для навигации автономных мобильных роботов.

Параметр	Значение
Угол сканирования	360 градусов
Частота сканирования	7–15 Гц
Частота дискретизации	4500 раз в секунду
Дальность обнаружения	до 12 м
Погрешность	2–4 %
Устойчивость к внешнему освещению	до 30 клюкс (клк)
Рабочее напряжение	5 В постоянного тока
Поддержка	*ROS1, ROS2*
Размеры	37,7 x 37,5 x 33 мм
Вес	40 г

- **Аккумуляторный модуль** имеет отсек для подключения аккумуляторов, вмещающий в себя три аккумуляторные батареи 3,7 В типоразмера 18650.

Оснащён светодиодным индикатором уровня заряда.

Позволяет производить зарядку аккумуляторов не вынимая их из модуля.

6. ПОРЯДОК РАБОТЫ С ОБОРУДОВАНИЕМ

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

Сборка

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

Подключение модулей

Подключение модулей датчиков к контроллеру сопряжения датчиков осуществляется с помощью прилагаемого набора проводов и кабелей. Модули, входящие в комплект, поддерживают *1-Wire* интерфейс. Для подключения используется кабель *3Pin* с разъемом, конструктивно защищенным от неправильного подключения.

Подготовка аккумуляторного модуля

Для начала работы с аккумуляторным модулем необходимо вставить три аккумуляторные батареи 3,7 В (поставляются в комплекте) в отсек для подключения аккумуляторов.

Убедиться, что все элементы заряжены – три сегмента светодиодного индикатора горят зеленым цветом. Свечение четвертого сегмента светодиодного индикатора сообщает о подключении зарядного устройства к сети переменного тока. Если индикаторы горят красным цветом, необходимо зарядить аккумуляторные батареи.

Зарядка батарей

Для зарядки аккумуляторных батарей в аккумуляторном модуле необходимо воспользоваться зарядным устройством на 12,6 В (поставляется в комплекте).

Для зарядки аккумуляторной батареи 7,4 Вольта необходимо воспользоваться зарядным устройством на 8,4 В (поставляется в комплекте).

7. ПРОГРАММИРОВАНИЕ РОБОТОТЕХНИЧЕСКОГО КОНТРОЛЛЕРА

Для программирования расширенного робототехнического набора необходимо подготовить компьютер, установив необходимое программное обеспечение. Например, *STM32CubeMX* и *IDE Keil (MDK-ARM v.5.36)* (файлы дистрибутивов прилагаются на электронном носителе) для робототехнического контроллера, *Arduino IDE* для контроллера сопряжения датчиков и робототехнического контроллера. Дистрибутив можно скачать отсюда:

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

Так же для работы с робототехническим контроллером в среде *Arduino IDE* необходимо установить *STM32CubeProgrammer* (файл дистрибутива прилагается на электронном носителе).

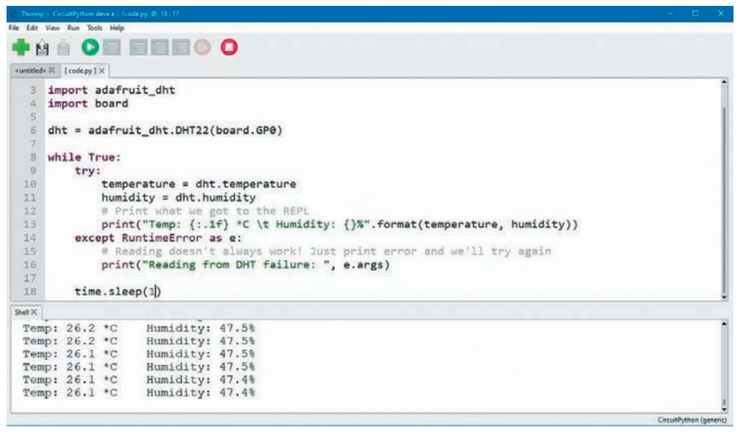
Для работы с модулем обработки технического зрения на базе микрокомпьютера *Raspberry Pi 5,* как и с любым другим компьютером, необходимо установить операционную систему. В нашем случае на *Raspberry Pi 5* установлена *Debian GNU/Linux 12 (bookworm)* – основанная на *Debian* операционная система для *Raspberry Pi.*

Если вы начинаете изучать программирование роботов с нуля, то *Python*, вероятно, самый быстрый и эффективный способ начать работу.

Во-первых, *Python* – один из самых простых языков для освоения. Другая причина заключается в том, что существует множество библиотек, написанных на *Python* для различных датчиков и компонентов. В результате, большое количество скриптов для учебников и проектов будет написано на *Python*. Если бы вы использовали другой язык, например, *JavaScript* (через *NodeJS*), вы могли бы оказаться в затруднительном положении без библиотеки для обычного датчика.

Для написания кода программ на языке *Python* будем использовать встроенную среду разработки *Thonny Python IDE* – это продвинутая *Python-IDE*, которая хорошо подходит для новичков. Хотя пользоваться ей могут вполне и профессионалы, некоторые черты этой *IDE* говорят о том, что она особенно хороша для начинающих питонистов. Она даёт в распоряжение программиста возможности по пошаговому выполнению выражений, средства визуализации стека вызовов и множество других полезных мелочей.

Если новичок возьмёт всё это на вооружение – он не только улучшит свои навыки *Python*-программирования, но и будет лучше понимать то, что происходит во время выполнения кода.



Для работы с модулем обработки технического зрения на базе микрокомпьютера *Raspberry Pi 5* необходимо подключить к нему монитор, клавиатуру и компьютерную мышь. Однако это не всегда удобно и возможно. В этом случае можно воспользоваться возможностью удаленного беспроводного подключения к рабочему столу *Raspberry Pi* посредством сети *Wi-Fi.* Для того воспользуемся приложением *VNC viewer.*

VNC – это широко распространенный метод удаленного доступа к рабочему столу компьютера по сети. Данные о нажатии клавиш и движении мыши, выполняемых пользователем на собственном компьютере передаются по сети на удаленный компьютер и воспринимаются им как действия с его собственными клавиатурой и мышью. Информация с экрана удаленного компьютера выводится на экране компьютера пользователя.

Работа по *VNC* через интернет с удаленным компьютером, находящимся в противоположной точке мира, для пользователя выглядит так, как будто этот компьютер находится непосредственно перед ним. Особенно *VNC* удобен при работе с графическим интерфейсом – с рабочим столом и программами для рабочего стола операционных систем *Windows, Linux* и других.

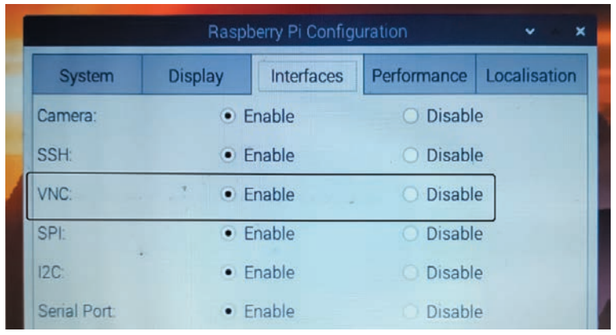
Virtual Network Computing (VNC) – система удалённого доступа к рабочему столу компьютера, использующая протокол *RFB* (англ. *Remote FrameBuffer,* удалённый кадровый буфер). Управление осуществляется путём передачи нажатий клавиш на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть.

VNC платформонезависимая и состоит из двух частей: серверная и клиентская. *VNC*-клиент, называемый *VNC viewer,* запущенный на одной операционной системе, может подключаться к *VNC*-серверу, работающему на любой другой ОС. Реализации клиентской и серверной части на сегодняшний момент существуют практически для всех операционных систем. К одному *VNC*-серверу одновременно могут подключаться множество клиентов.

При подключении *VNC*-клиента достаточно указать *DNS*-имя или *IP*-адрес удаленного компьютера, и пароль, если доступ к *VNC*-серверу защищен паролем.

Основной объем трафика по *VNC* – это передача графической информации, выводимой на экран. Характеристика пропускной возможности канала для работы от 32 Кбит/сек до 2 Мбит/сек. Комфортная работа в полноцветном режиме при разрешении экрана 1024 × 768 будет при скорости 1-2 Мбит/сек. Канал передачи максимально нагружен только при обновлении больших участков экрана, при печати текста трафик заметно меньше. При больших задержках передачи пакетов, ухудшение времени реакции на нажатие клавиш и движение мыши.

Для запуска серверной части на *Raspberry Pi* достаточно в разделе *Raspberry Pi Configuration* включить необходимый параметр:



В нашем случае *VNC-server* на модуле обработки технического зрения (на базе микрокомпьютера *Raspberry Pi 5*) уже запущен. Для установки клиента *RealVNC client* на своем рабочем компьютере скачайте дистрибутив из Интернета, например, отсюда:

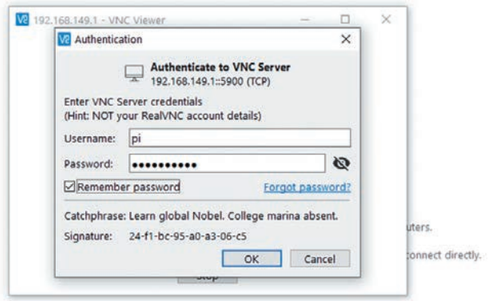
https://apps.microsoft.com/detail/xp99dvcpgktxnj?hl=ru-ru&gl=US

Установите на свой компьютер скачанный *VNC Viewer.*

Для того, чтобы подключиться к рабочему столу микрокомпьютера *Raspberry Pi 5*, сначала нужно обеспечить с ним связь по *Wi-Fi*. Для этого, на базе *Raspberry Pi 5* реализована *Wi-Fi* точка доступа. После включения питания и загрузки операционной системы будет создана точка доступа *Wi-Fi*, начинающаяся с *«HW»*. Подключите свой рабочий компьютер к этой сети *Wi-Fi* (пароль (ключ безопасности) – *«hiwonder»*).

Теперь можно запустить приложение *VNC Viewer*. Настройте соединение с сервером (модулем обработки технического зрения) – необходимо указать его *IP*-адрес (по умолчанию 192.168.149.1).

После установления соединения система предложит ввести имя пользователя и пароль. Имя пользователя *pi*, пароль – *raspberrypi.*



Установка программного обеспечения для работы с робототехническим контроллером

Рассмотрим установку программного обеспечения для работы с робототехническим контроллером, на примере *STM32CubeMX* и *IDE Keil (MDK-ARM v.5.36)* (файлы дистрибутивов прилагаются на электронном носителе).

STM32CubeMX и *MDK-ARM V5.36* – это два инструмента, используемых при разработке на микроконтроллерах *STM32*, но они выполняют разные функции и служат разным целям.

STM32CubeMX V6.15

Это графический генератор проектов и конфигуратор для микроконтроллеров *STM32*. Его основные функции:

- Настройка аппаратных ресурсов (пины, тактирование, периферия).
- Генерация исходного кода на основе выбранных настроек.
- Обеспечение интеграции с *HAL (Hardware Abstraction Layer)* библиотеками.
- Генерация проектных файлов для различных *IDE* (например, *KEIL, IAR, STM32CubeIDE*).

MDK-ARM V5.36 (Keil µVision)

Это интегрированная среда разработки *(IDE)* для программирования микроконтроллеров *ARM Cortex-M*, включая *STM32*. Ее основные функции:

- Написание, компиляция и отладка кода.
- Поддержка различных компиляторов (в том числе *Keil ARM Compiler*).
- Встроенные инструменты для отладки и симуляции.
- Управление проектами и библиотеками.

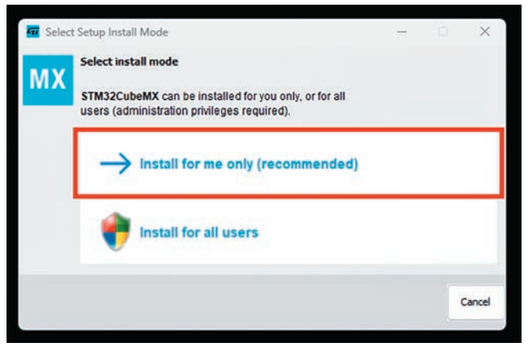
Это основной инструмент для разработки программного обеспечения — написания кода, его сборки и отладки.

Основные шаги разработки и написания кода для контроллеров STM32 выглядят следующим образом:

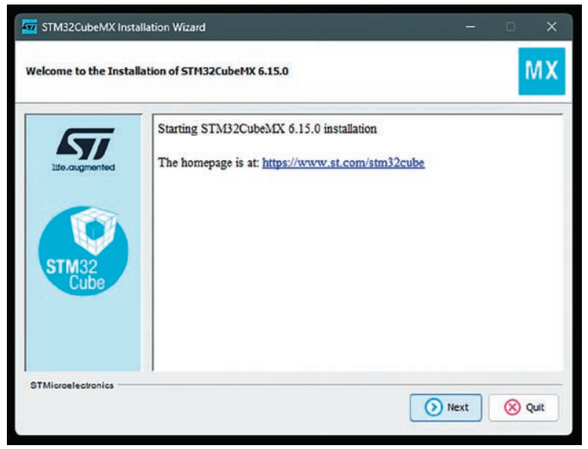
Планирование и подготовка → Конфигурация в STM32CubeMX → Генерация проекта → Разработка в MDK-ARM → Компиляция → Отладка → Тестирование → Финализация

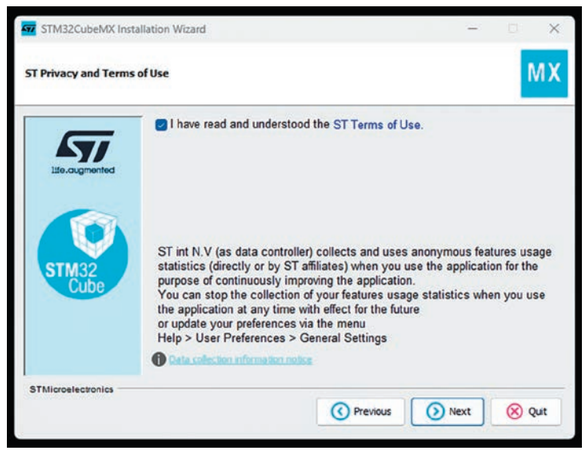
Установка *STM32CubeMX*

Найдите на входящем в комплект поставки *USB*–накопителе и запустите файл *SetupSTM32CubeMX-6.15.0-Win.exe* для установки *STM32CubeMX*, установите приложение только для себя.

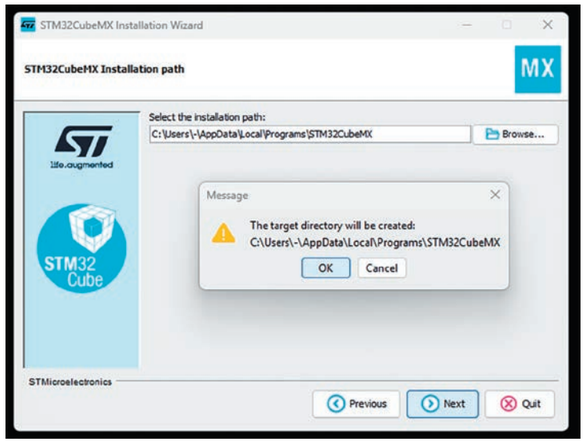


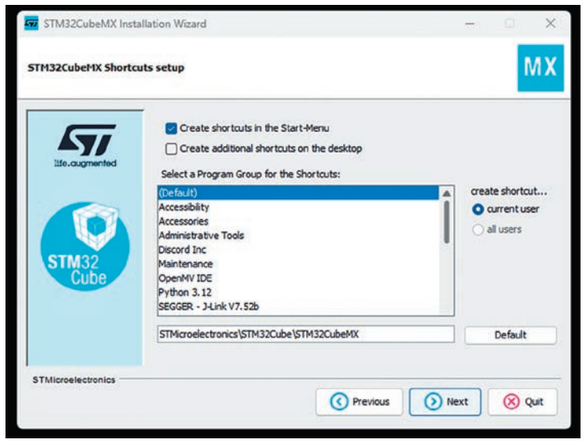
В последующих окнах согласитесь с пользовательским соглашением и условиями использования продукта.



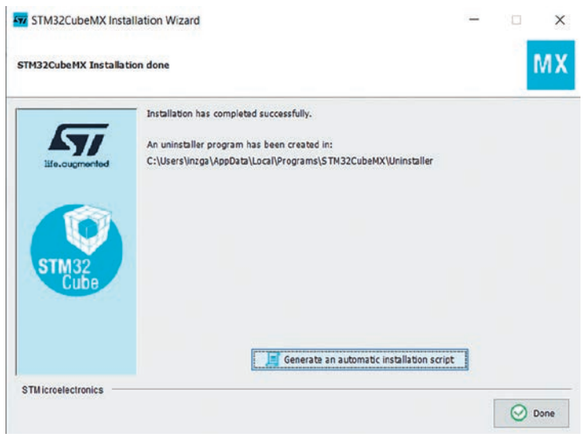


Выберите путь установки приложения и подтвердите установку.



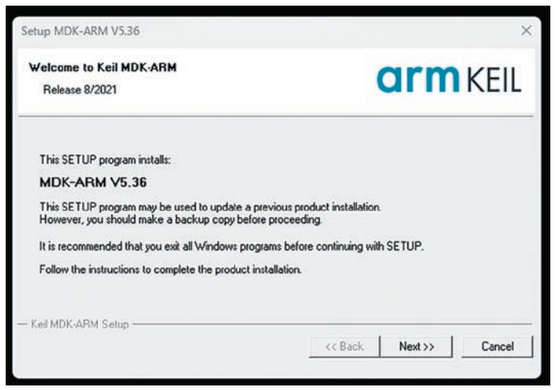


Дождитесь завершения установки и нажмите *«Done»*, чтобы закрыть установщик.

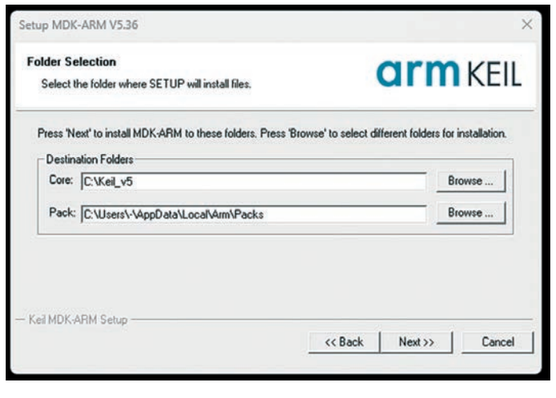


Установка *Keil µVision*

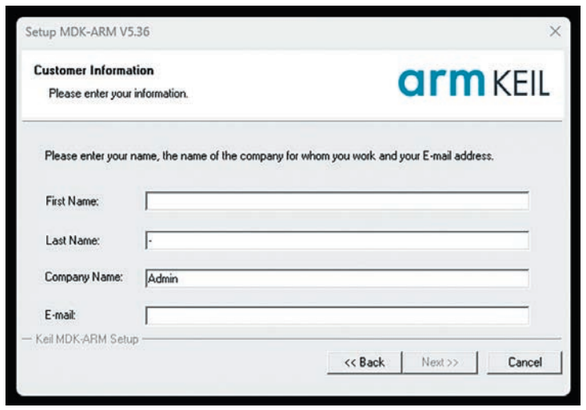
Найдите на входящем в комплект поставки *USB*-накопителе и запустите файл *MDK536.exe* для установки *Keil µVision.*



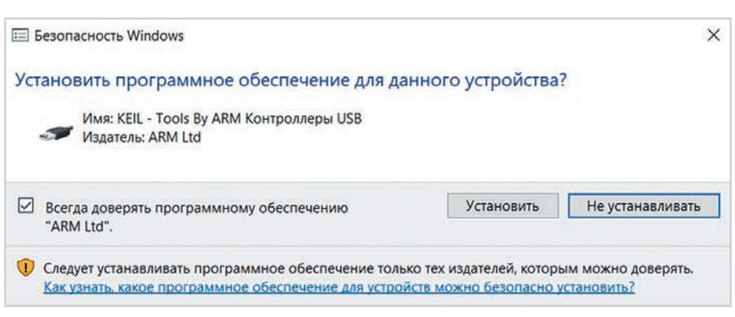
В последующих окнах согласитесь с лицензионным соглашением и выберите путь установки приложения.

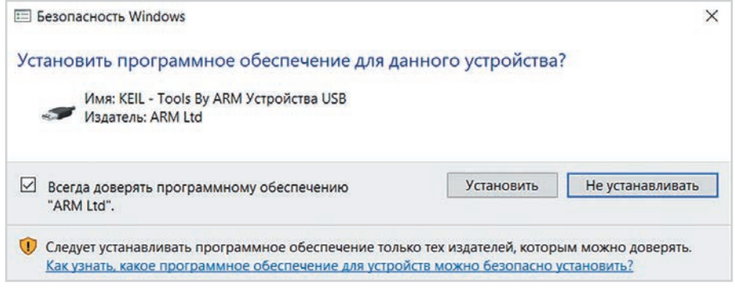


Заполните информационные поля о пользовательских данных и начните установку приложения.

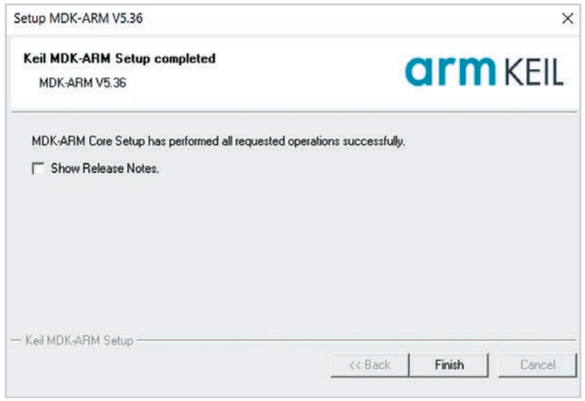


Согласитесь на установку драйверов на компьютер.





Дождитесь завершения установки и нажмите *«Finish»*, чтобы закрыть установщик.

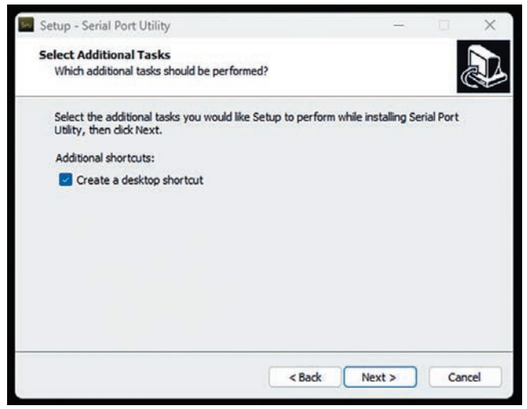


Если после окончания установки у вас появилось окно с *«Pack Installer»*, закройте его.

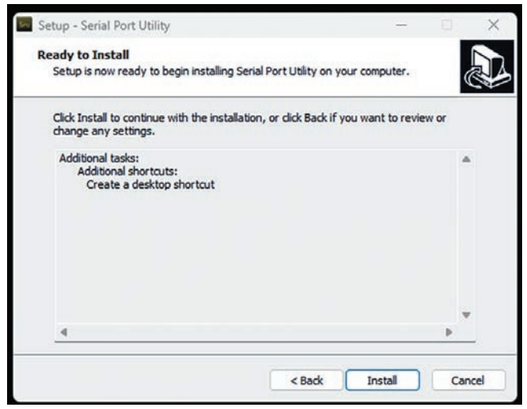
Установка *Serial Port Utility*

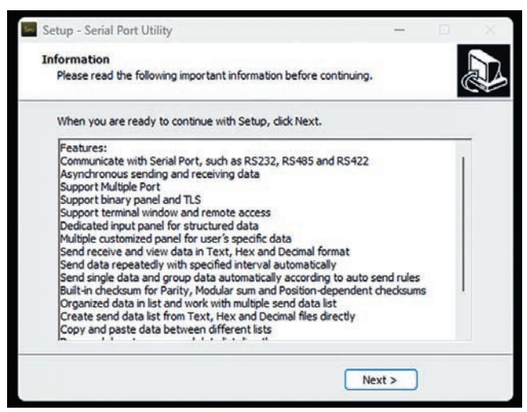
Найдите на входящем в комплект поставки *USB*-накопителе и запустите файл *serial_port_utility.exe* для установки *Serial Port Utility*, выберите английский язык.

Согласитесь с лицензионным соглашением, а в следующей вкладке подтвердите создание ярлыка на рабочем столе.

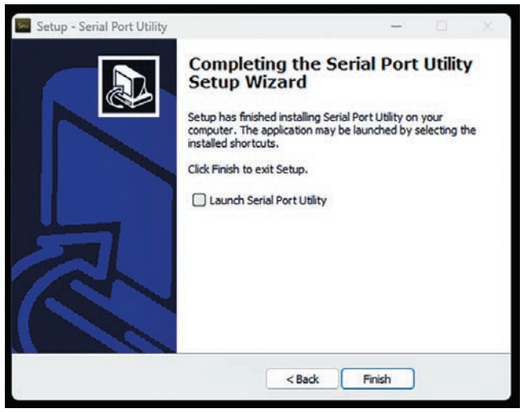


Начните установку приложения, ознакомьтесь с его возможностями.





По завершению установки, закройте окно установщика.



Работа с *STM32CubeMX* и *Keil µVision*

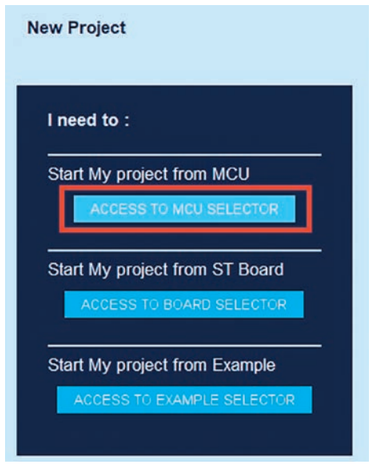
STM32CubeMX – это программа для предварительной настройки микроконтроллеров и инициализации начального кода для различных сред разработки, в том числе и *Keil µVision*.

Основная идея программного обеспечения *STM32CubeMX* заключается в предоставлении универсального инструмента для настройки и создания кода инициализации для микроконтроллеров *STM32*.

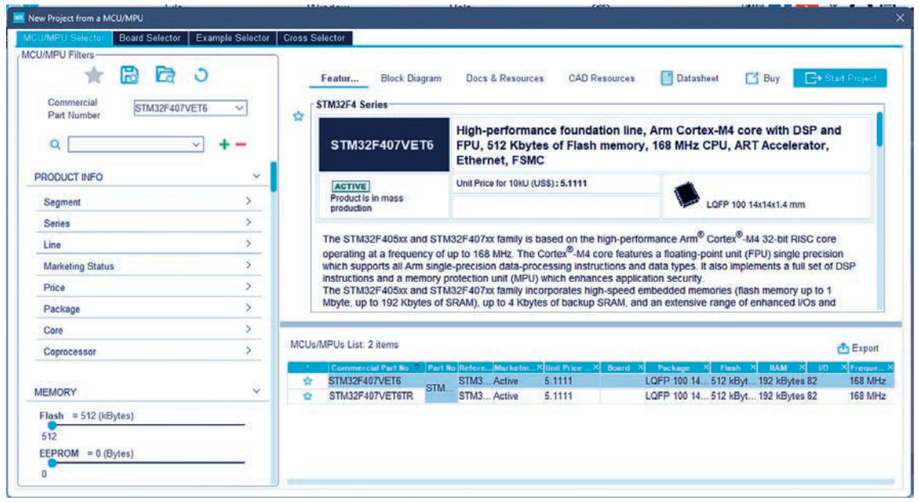
µVision – это платформа разработки программного обеспечения, которая сочетает в себе надежный и современный редактор с менеджером проекта. Он объединяет все инструменты, необходимые для разработки встроенных приложений, включая компилятор *C/C++*, макроассемблер, компоновщик и генератор файлов *HEX*.

Рассмотрим пример написания простейшей программы для робототехнического контроллера – мигание встроенным светодиодом.

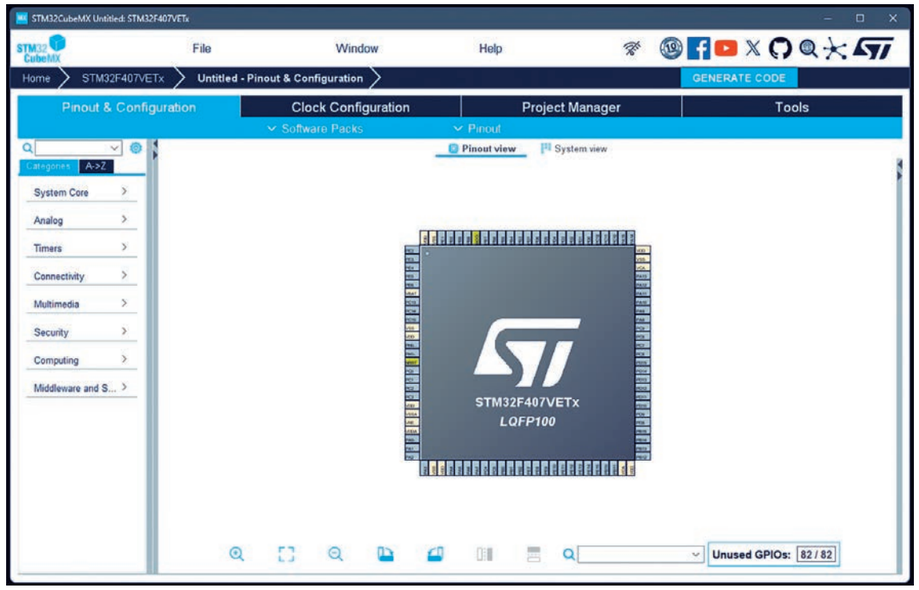
Откройте *STM32CubeMX* и на главной странице приложения, во вкладке *«NewProject»* выберите *Access to MCU selector.* На требование интернета нажмите *No*.



В строке поиска введите *«STM32F407VET6»* и в раскрывшемся в нижней части экрана списке выберите первый вариант, соответствующий вводимому в поиск имени контроллера. Нажмите *Start Project* в верхней правой части окна.



В открывшемся окне вы можете ознакомиться с контроллером и возможностями приложения.

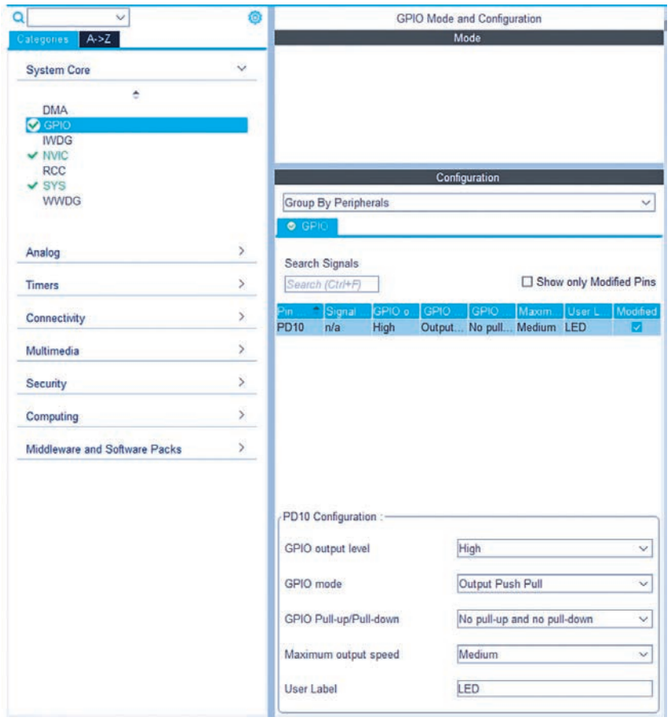


В таблице, приведенной ниже, указаны контакты контроллера для подключения периферийных устройств, расположенных на плате робототехнического контроллера.

Устройство	Контакты
PWM_Servo3	PC8
IMU датчик *QMI8658*	PB10 (SCL), PB11 (SDA), PB12 (INT2)
Motor 1 (M1)	PE13 (TIM1_CH3), PE14 (TIM1_CH4)
Encoder M1	PA0 (TIM5_CH1), PA1 (TIM5_CH2)
Motor 2 (M2)	PE9 (TIM1_CH1), PE11 (TIM1_CH2)
Encoder M2	PA15 (TIM2_CH1), PB3 (TIM2_CH2)
Motor 3 (M3)	PE5(TIM9_CH1), PE6 (TIM9_CH2)
Encoder M3	PD12 (TIM4_CH1), PD13 (TIM4_CH2)
Motor 4 (M4)	PB8 (TIM10_CH1), PB9 (TIM10_CH2)
Encoder M4	PB4 (TIM3_CH1), PB5 (TIM3_CH2)
PWM_Servo1	PA11
PWM_Servo2	PA12
PWM_Servo4	PC9
Интеллектуальные сервоприводы *BUS_Servo*	PC12 (UART5_TX)
Программируемая система звуковой индикации *Buzzer*	PA6
Программируемые светодиоды *LED1, LED2, LED3*	PD9, PD10, PD11
RGB_LED	PA7
Аналоговый вход для анализа уровня напряжения аккумулятора	PB8
I²C	PB6 (SCL), PB7 (SDA)

Рассмотрим процедуру написания программ для контроллера на примере программирования встроенного светодиода. На робототехническом контроллере находится встроенный светодиод, который соединен с ножкой *PD10* контроллера. Для того, чтобы заставить его светиться, необходимо просто подать на него питание.

Для этого найдите на микроконтроллере ножку *PD10*, нажмите на нее левой кнопкой мыши, и выберите вариант *GPIO_OUTPUT*, тем самым настраивая ее как выход – для подпитки светодиода. В левом меню *Categories* во вкладке *System Core* найдите вариант *GPIO* и нажмите на него. В открывшемся окне в самом низу экрана найдите только что заданную ножку *PD10* и настройте ее в блоке *PD10 Configuration*, как представлено на изображении.

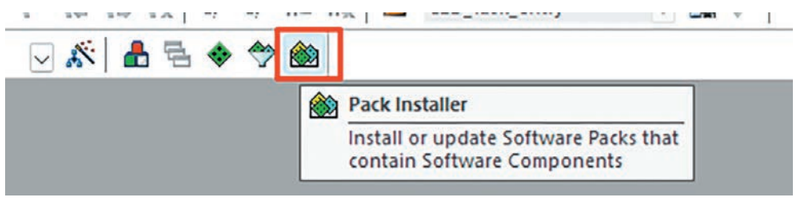


В верхней панели найдите вкладку *Project Manager*. Введите имя проекта, укажите путь для его сохранения, в строке *Application Structure* выберите *Basic*, а в *Toolchain/IDE – MDK-ARM*. Удостоверьтесь, что все параметры в блоке *Project Settings* заполнены и корректны.

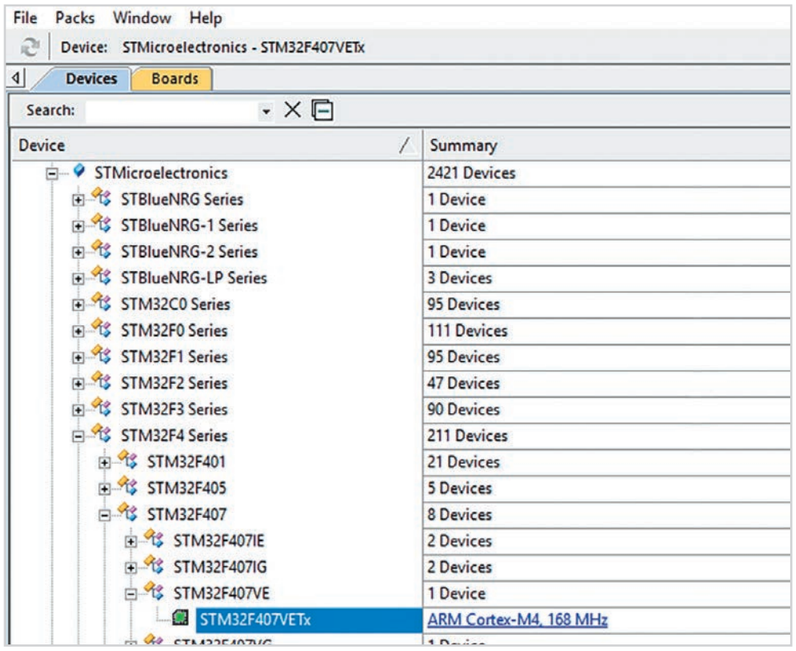
Для генерации дальнейшего кода проекта необходимо подключить необходимые файлы для работы с микроконтроллером. Для этого извлеките из архива *Firmware_STM32.rar* (находится на *USB*-накопителе) папку *STM32Cube_FW_F4_V1.28.2* и разместите ее на диске *C* (*C:\Users\{имя_пользователя}\STM32Cube\Repository* (если данная папка не было создана автоматически – создайте вручную)). А во вкладке *Project Manager* в блоке *Mcu and Firmware P*ackage снимите галочку с *Use latest available version* и выберите в *Firmware Package Name and Version* вариант *STM32Cube FW_F4 V1.28.2*.

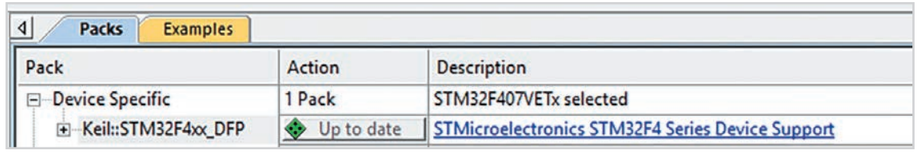
В верхней части приложения нажмите на кнопку *GENERATE CODE*. В появившемся окне *Code generation* после сообщения об успешной генерации кода нажмите *Open Project*, после чего откроется приложение *Keil µVision*.

Как только у вас открылось приложение, найдите иконку *Pack Installer* для установки необходимых пакетов для работы с контроллером, если он не открылся автоматически.



Во вкладке *Devices* найдите вкладку *STMicroelectronics*, и среди открывшегося дерева контроллеров найдите нужный – *STM32F407VETx*. Нажмите на него два раза, и если загрузка не пошла автоматически (синяя полоска в правой нижней части экрана), то во вкладке *Packs* в *Device Specifics* установите пакет у надписи «*STM32F4xx_DFP*».





Закройте *Pack Installer,* и в *Keil µVision* в левой части экрана, в проекте найдите папку *Application/User,* и в ней откройте файл *main.c.* В данном файле можно прописывать логику работы микроконтроллера, а именно – управление заданными в *STM32CubeMX* пинами, передачей данных по последовательным портам и т. д. В данном случае будет рассматриваться стандартный *Blink*, где задача светодиода – периодически моргать, то есть включаться и выключаться с задержкой в заданное количество миллисекунд. Для этого в файле *main.c* найдите функцию *int main(void)*, а в ней – бесконечный цикл *while (1).*

Так как моргание светодиода будет происходить бесконечно, то внутри данного цикла нам нужно написать следующий алгоритм:

1. Светодиод включается;
2. Пауза;
3. Светодиод выключается;
4. Пауза.


```
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Данный алгоритм можно представить следующим образом:

HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);

HAL_Delay(400);

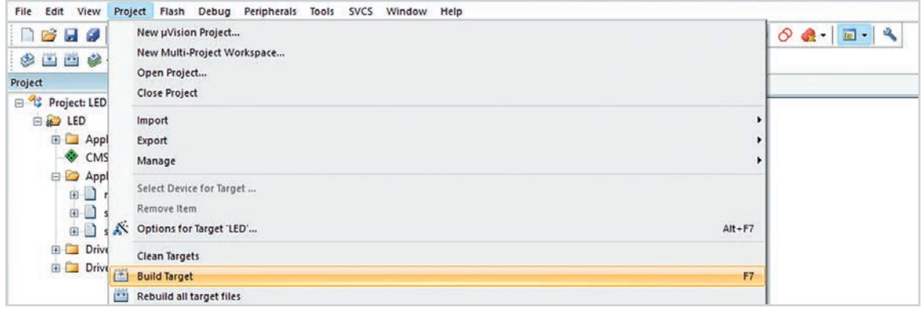
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);

HAL_Delay(400);

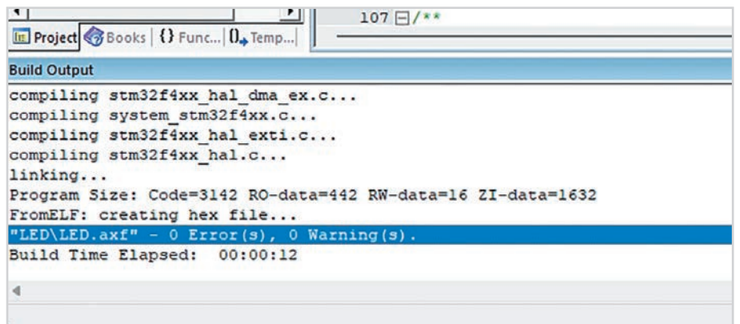
где *GPIO_PIN_RESET* – это состояние пина *LOW*, а *GPIO_PIN_SET* – *HIGH*. Данная логика, при которой в *LOW* светодиод горит, обусловлена схематикой его подключения на плате.

Необходимо учесть, что пользовательский код должен находиться в рамках комментариев */* USER CODE BEGIN*/* и */* USER CODE END*/*, иначе будет проигнорирован при сборке проекта программой.

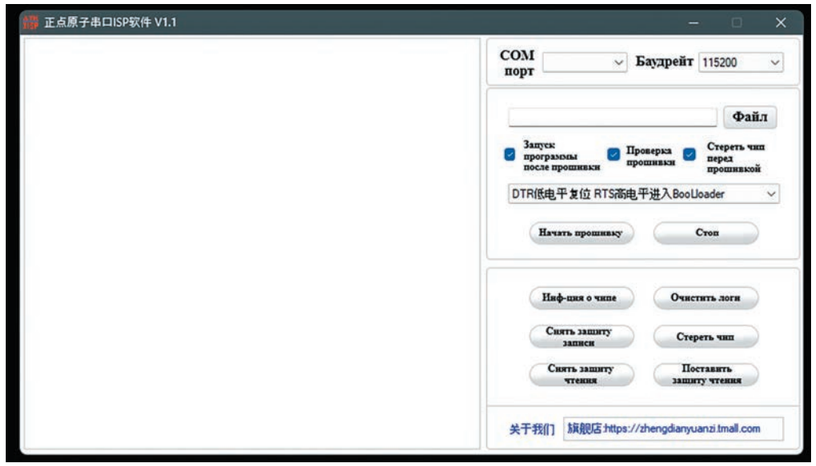
После завершения написания кода в верхней панели приложения найдите вкладку *Project* и нажмите *Build Target*, чтобы собрать весь проект в файл прошивки для контроллера.



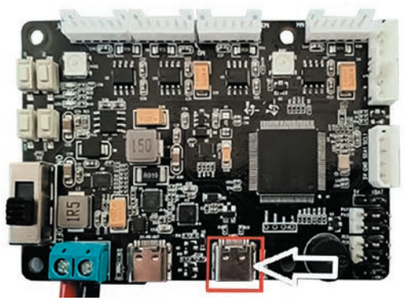
Перейдите в папку, где хранится ваш проект (указано в названии окна *Keil µVision*), и найдите файл прошивки, который вы получили, его название вы можете увидеть в консоли вывода в нижней части окна.



Для загрузки кода программ в контроллер необходимо воспользоваться программой для прошивки контроллера *ATK-XISP.exe*, которая находится на *USB*-накопителе. Запустите файл *ATK-XISP.exe*. Настройте параметры скорости передачи (баудрейта) и режима загрузки *(DTR, RTS, Bootloader)* как представлено на изображении. Выберите расположение прошивки, сформированной в *Keil µVision* в прошлом шаге, нажав на кнопку «Файл».



Подключите при помощи кабеля *USB-A–USB-C* робототехнический контроллер к вашему ПК по *UART1*.

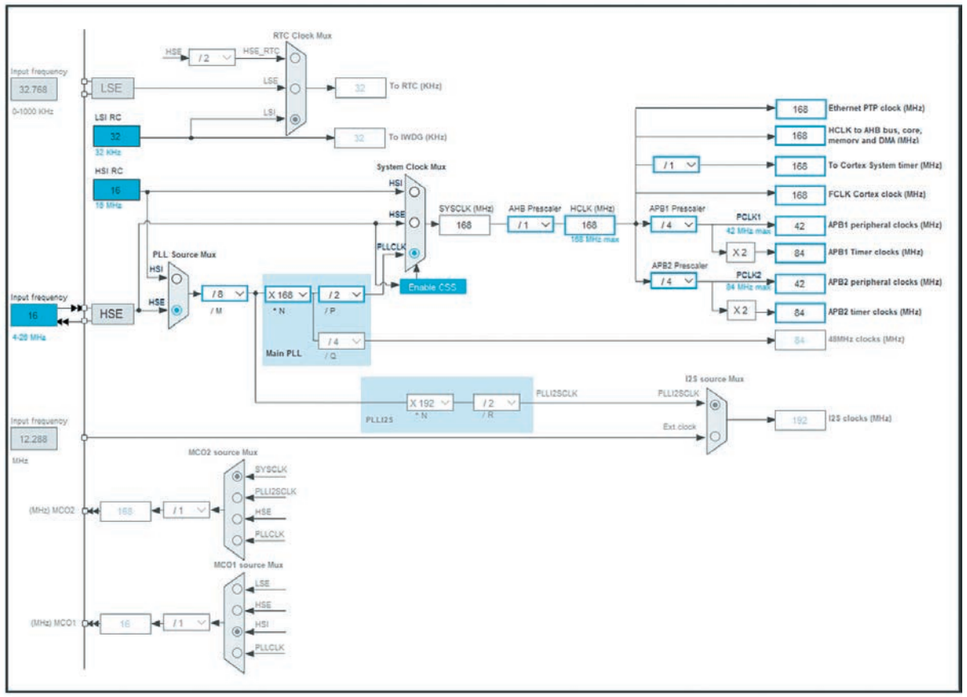


В *ATK-XISP* выберите *COM*-порт вашего ПК, к которому подключен робототехнический контроллер. Начните прошивку, нажав соответствующую кнопку в приложении, и вы услышите два звуковых сигнала от контроллера. По завершению процесса прошивки удостоверьтесь, что светодиод начал моргать.

Рассмотрим еще один пример написания программ. Для этого используем находящийся на плате робототехнического контроллера пассивный зуммер, то есть требующий управляющего ШИМ-сигнала для создания звука различного тона. Рассмотрим проект, совмещающий в себе мигание светодиода и звучание зуммера.

Для генерации ШИМ-сигнала используются таймеры, способные создавать прерывания с заданной периодичностью. Таким образом можно генерировать ШИМ-сигнал и менять скорость вращения колес, яркость свечения светодиода или тональность звучания зуммера.

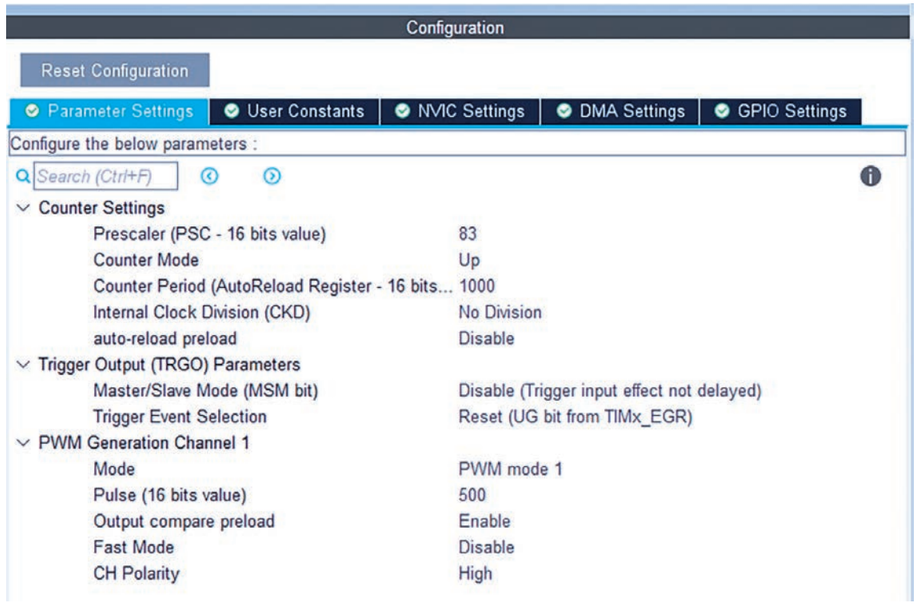
Для того, чтобы звучание зуммера было стабильным, а диапазон доступных для использования частот не был ограничен только низкими частотами, мы будем использовать встроенный на плате внешний кварцевый генератор частот. Для этого откройте ранее созданный проект в *STM32CubeMX*, и во вкладке *Pinout & Configuration*, в левой панели *Categories* найдите вкладку *System Core* и выберите *RCC*. В блоке *Mode*, в категории *High Speed Clock (HSE)* выберите *Crystal/Ceramic Resonator*. Пины *PHO_OSC_IN* и *PHO_OSC_OUT* загорятся зеленым. Далее необходимо настроить конфигурацию тактирования. Для этого перейдите во вкладку *Clock Configuration* в верхней части окна, и настройте ее как представлено на изображении.



Зуммер соединен с ножкой *PA6* робототехнического контроллера, и эта же ножка может быть назначена выводом ШИМ-сигнала таймера № 3 – *TIM3* – по первому из его каналов. Во вкладке *Pinout & Configuration*, в левой панели *Categories* найдите вкладку *Timers*.

Во вкладке *Timers* найдите *TIM3*. В параметрах таймера в разделе *Mode* задайте параметр *Clock sourse* как *Internal Clock*, а *Channel1* как *PWM Generation CH1*. Таким образом мы выбрали пин *PA6*, соединенный с зуммером, как вывод ШИМ-сигнала с третьего таймера по первому каналу.

В разделе *Parameter Settings* в *TIM3* настройте таймер как представлено на изображении, и задайте для пина *PA6* имя *BUZZER*, нажав на него правой кнопкой мыши и выбрав *Enter user label*.



Сгенерируйте код проекта по аналогии с примером со светодиодом, откройте его в *Keil µVision*, добавьте к коду светодиода следующее (в цикле while):

HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);

HAL_Delay(400);

HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_1);

HAL_Delay(400);

Соберите проект, и загрузите прошивку на контроллер. Удостоверьтесь, что светодиод моргает, а зуммер издает звук.

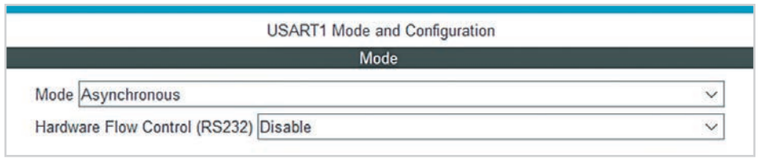
Совместная работа робототехнического контроллера с модулем обработки технического зрения

Рассмотрим пример работы робототехнического контроллера с модулем обработки технического зрения, где пользователь отправляет в терминал *Raspberry Pi* сообщение, робототехнический контроллер его принимает и подает команды при помощи ШИМ-сигнала на двигатели постоянного тока. Интерфейсом общения между контроллерами выберем *UART*. Алгоритм работы можно описать следующим образом:

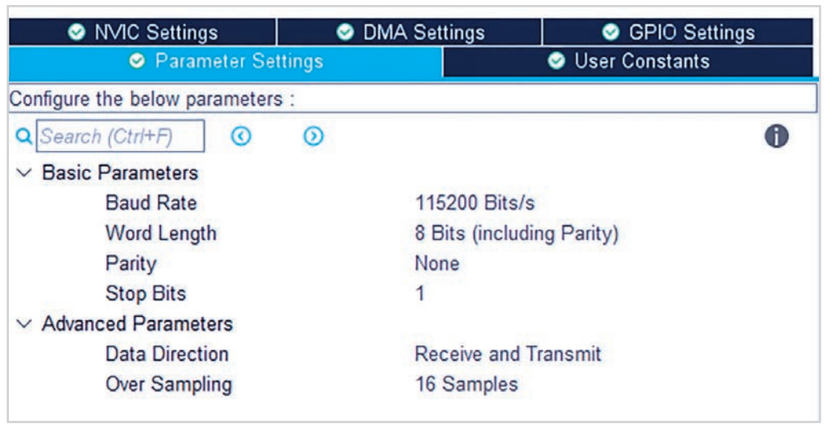
1. Модуль обработки технического зрения считывает введенную в поле терминала *Thonny IDE* команду;

2. Осуществляется проверка, является ли текст, введенный пользователем, одной из команд на движение;
3. Если команда распознана *Raspberry Pi*, он отправляет ее по *UART* на робототехнический контроллер;
4. Контроллер опознает команду и отправляет на двигатели постоянного тока управляющие сигналы направления и скорости.

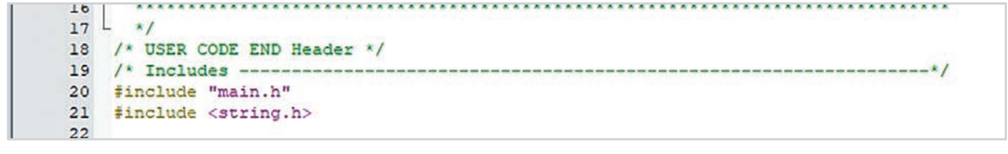
Для начала необходимо настроить *UART* на робототехническом контроллере. Для этого во вкладке *Pinout & Configuration*, в левой панели *Categories* найдите вкладку *Connectivity*. Во вкладке *Connectivity* найдите *USART1*. Выберите в блоке *Mode* в параметре *Mode* вариант *Asynchronous*.



Пины *PA9, PA10* на контроллере в *STM32CubeMX* загорятся зеленым и станут соответственно *USART1_TX* и *USART1_RX*. В блоке *Configuration* найдите *Parameter Settings* и убедитесь, что ваши параметры совпадают с представленными на изображении.



Сгенерируйте код для *Keil µVision*, откройте в нем проект, и вставьте следующий код вместо кода светодиода и зуммера в цикле *while*. Помимо этого, подключите библиотеку *<string.h>* возле строчки *#include «main.h»*.

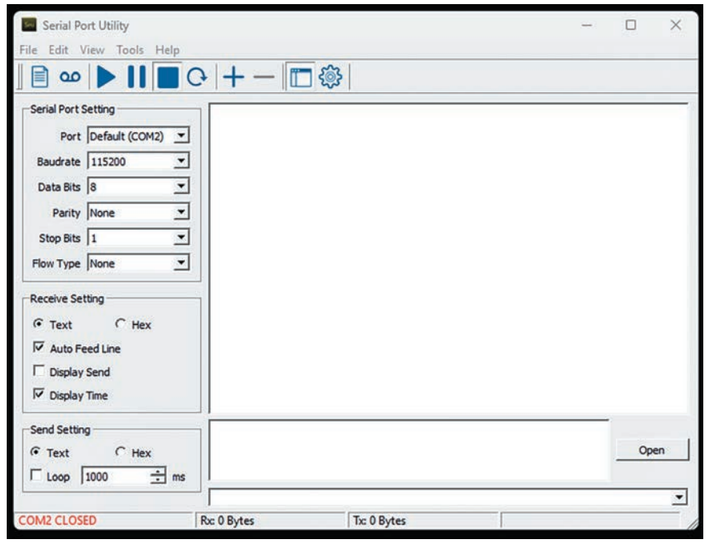


```
uint8_t rx_buff[1] = {0};
if (HAL_UART_Receive(&huart1, rx_buff, 1, 50) == HAL_OK) {
    uint8_t* tx_buff = NULL;
    int len;

    switch (rx_buff[0]) {
          case ‘F’:
                tx_buff = (uint8_t*)»MoveForward»;
                len = strlen((char*)tx_buff);
                break;
          case ‘B’:
                tx_buff = (uint8_t*)»MoveBackward»;
                len = strlen((char*)tx_buff);
                break;
          case ‘L’:
                tx_buff = (uint8_t*)»MoveLeft»;
                len = strlen((char*)tx_buff);
                break;
          case ‘R’:
                tx_buff = (uint8_t*)»MoveRight»;
                len = strlen((char*)tx_buff);
                break;
          case ‘S’:
                tx_buff = (uint8_t*)»StopMove»;
                len = strlen((char*)tx_buff);
                break;
          default:
                tx_buff = (uint8_t*)»Wrong command»;
                len = strlen((char*)tx_buff);
                break;
    }
    HAL_UART_Transmit(&huart1, tx_buff, len, 50);
    HAL_Delay(100);
}
```

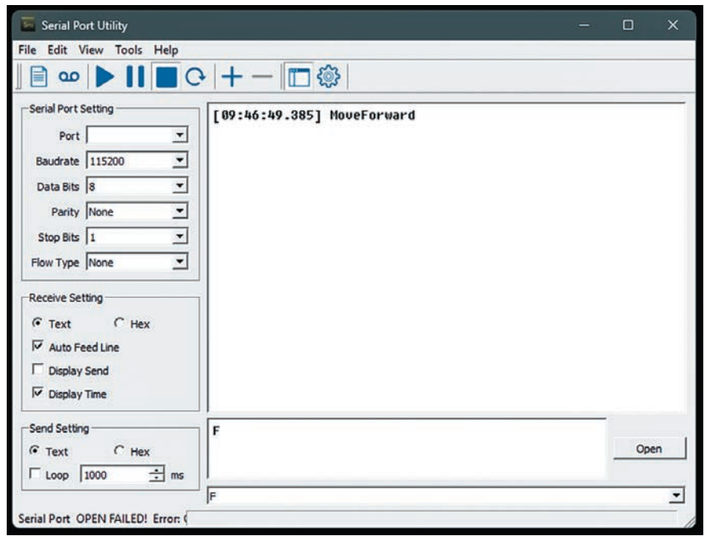
Соберите проект и прошейте контроллер аналогично прошлым примерам. Для проверки *UART* подключения откройте приложение *Serial Port Utility,* настройте параметры следующим образом (см. рисунок «Настройка параметров *Serial Port Utility*»).

Подключите при помощи кабеля *USB-A–USB-C* робототехнический контроллер к вашему ПК по *UART1*.

В *Serial Port Utility* в блоке *Serial Port Settings* в параметре *Port* выберите порт вашего ПК, к которому подключен робототехнический контроллер, и нажмите кнопку «Open» в правой нижней части экрана. Надпись в левом нижнем углу сменится с красного цвета на зеленый. В строке ввода сообщений введите *«F»*.



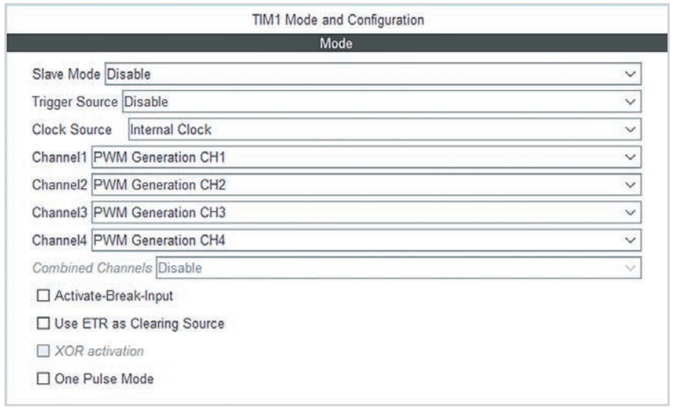
Настройка параметров *Serial Port Utility*

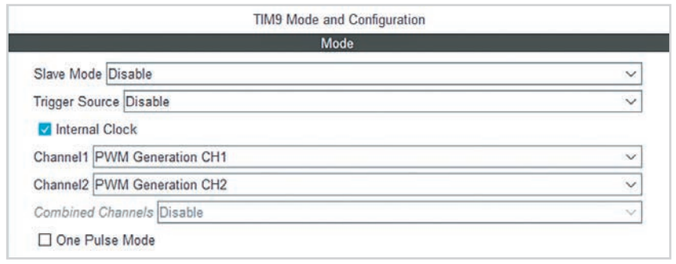


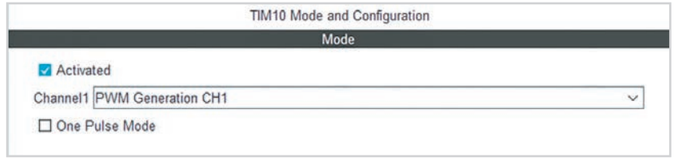
Удостоверьтесь, что в панели вывода входящих сообщений робототехнический контроллер ответил *MoveForward* как представлено.

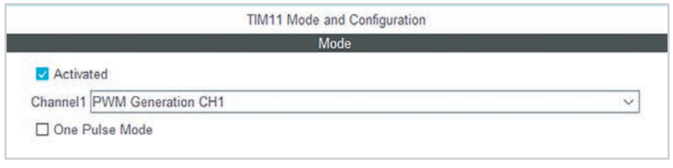
При помощи робототехнического контроллера можно реализовать управление моторами робота путем подачи на них управляющих сигналов, в том числе ШИМ-сигналов, определяющих скорость вращения колес. Так как в плату контроллера встроен драйвер моторов, облегчающий управление двигателями постоянного тока, то для работы с ними надо лишь указать в проекте контакты контроллера, соединённых с драйвером, и подавать на них нужные сигналы.

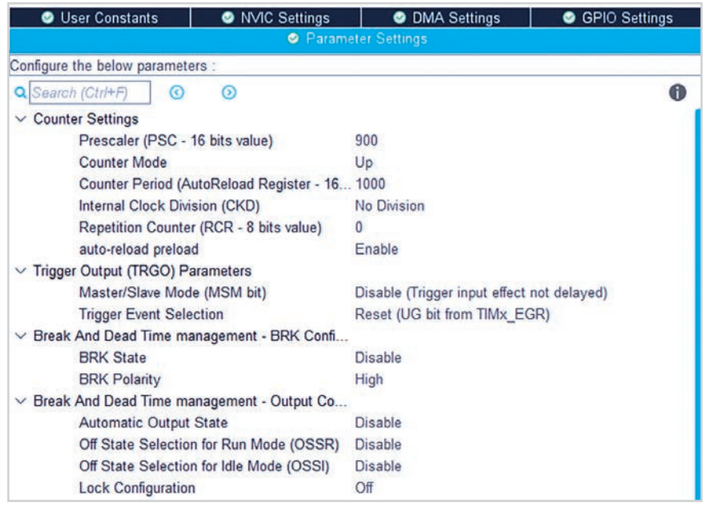
Для этого откройте ранее созданный проект в *STM32CubeMX*, и во вкладке *Pinout & Configuration*, в левой панели *Categories* найдите вкладку *Timers* и настройте таймеры *TIM1*, *TIM9, TIM10* и *TIM11* в соответсвии с приведенными изображениями.



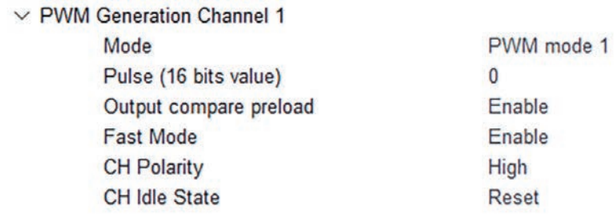








Настройте каждый ШИМ-канал (*PWM Generation Channel* в *Parameter Settings*) следующим образом как представлено на изображении.



Сгенерируйте код для *Keil µVision*, откройте в нем проект, и замените код в цикле *«while»* на следующий.

```
uint8_t rx_buff[1] = {0};
if (HAL_UART_Receive(&huart1, rx_buff, 1, 50) == HAL_OK) {
    uint8_t* tx_buff = NULL;
    int len;

    switch (rx_buff[0]) {
          case ‘F’:
                tx_buff = (uint8_t*)»MoveForward»;
                len = strlen((char*)tx_buff);
                motor1(150);
                motor2(150);
                motor3(150);
```

```
        motor4(150);
        break;
case ‘B’:
        tx_buff = (uint8_t*)»MoveBackward»;
        len = strlen((char*)tx_buff);
        motor1(-150);
        motor2(-150);
        motor3(-150);
        motor4(-150);
        break;
case ‘L’:
        tx_buff = (uint8_t*)»MoveLeft»;
        len = strlen((char*)tx_buff);
        motor1(-130);
        motor2(-130);
        motor3(150);
        motor4(150);
        break;
case ‘R’:
        tx_buff = (uint8_t*)»MoveRight»;
        len = strlen((char*)tx_buff);
        motor1(150);
        motor2(150);
        motor3(-130);
        motor4(-130);
        break;
case ‘S’:
        tx_buff = (uint8_t*)»StopMove»;
        len = strlen((char*)tx_buff);
        motor1(0);
        motor2(0);
        motor3(0);
        motor4(0);
        break;
default:
        tx_buff = (uint8_t*)»Wrong command»;
        len = strlen((char*)tx_buff);
        motor1(0);
        motor2(0);
        motor3(0);
        motor4(0);
        break;
}
```



```
        HAL_UART_Transmit(&huart1, tx_buff, len, 50);
        HAL_Delay(100);
    }
```

А в комментариях */* *USER CODE BEGIN PFP* */* в верхней части кода добавьте фукнции управления колесами следующим образом:

```
55
56  /* Private function prototypes -----------------------------------------------*/
57  void SystemClock_Config(void);
58  static void MX_GPIO_Init(void);
59  static void MX_TIM3_Init(void);
60  static void MX_USART1_UART_Init(void);
61  static void MX_TIM1_Init(void);
62  static void MX_TIM9_Init(void);
63  static void MX_TIM10_Init(void);
64  static void MX_TIM11_Init(void);
65
66  /* USER CODE BEGIN PFP */
67 void motor1(int speed) {
68   if(speed > 0) {
69         __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, 0);
70         __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, speed);
71     } else if(speed < 0) {
72         HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, 0);
```

```
/* USER CODE BEGIN PFP */
void motor1(int speed) {
    if(speed > 0) {
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, 0);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, speed);
    } else if(speed < 0) {
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, 0);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,-speed);
    } else {
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, 0);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, 0);
    }
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
}

void motor2(int speed) {
    if(speed > 0) {
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, 0);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, speed);
    } else if(speed < 0) {
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, 0);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,-speed);
    } else {
```

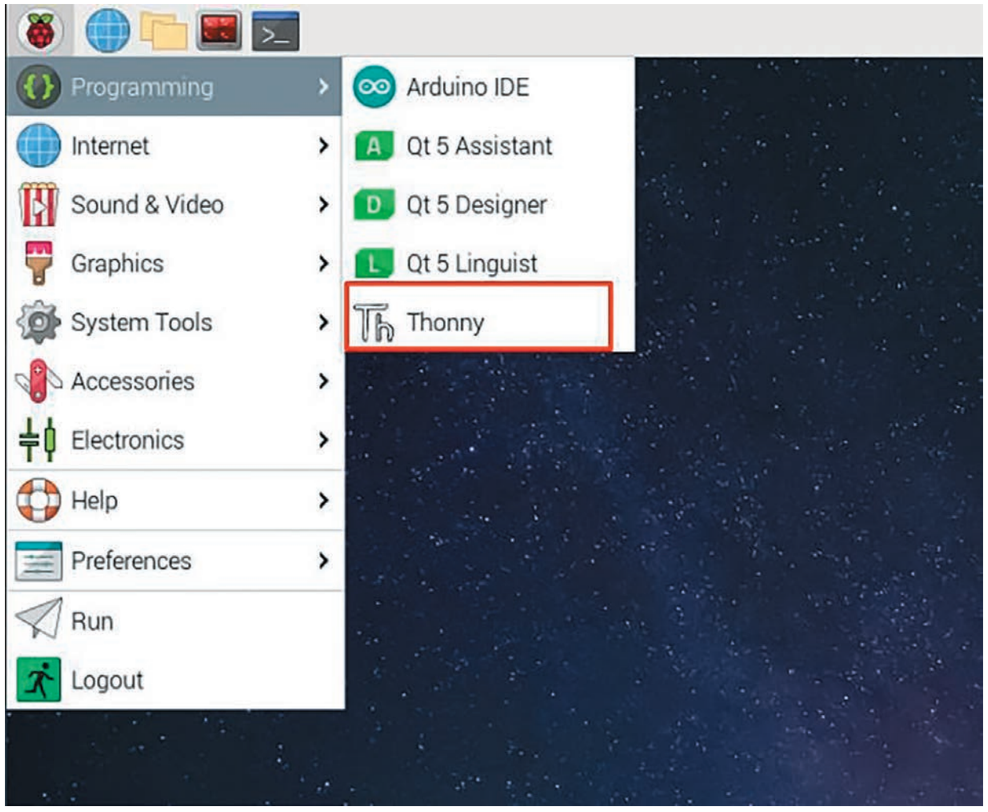
```
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,0);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,0);
}
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
}

void motor3(int speed) {
if(speed > 0) {
    __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, 0);
    __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_2, speed);
} else if(speed < 0) {
    __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_2, 0);
    __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_1,-speed);
} else {
    __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, 0);
    __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_2, 0);
}
HAL_TIM_PWM_Start(&htim9, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim9, TIM_CHANNEL_2);
}

void motor4(int speed) {
    if(speed > 0) {
        __HAL_TIM_SET_COMPARE(&htim11, TIM_CHANNEL_1, 0);
        __HAL_TIM_SET_COMPARE(&htim10,TIM_CHANNEL_1,speed);
    } else if(speed < 0) {
        __HAL_TIM_SET_COMPARE(&htim10, TIM_CHANNEL_1, 0);
        __HAL_TIM_SET_COMPARE(&htim11,TIM_CHANNEL_1,-speed);
} else {
    __HAL_TIM_SET_COMPARE(&htim10, TIM_CHANNEL_1, 0);
    __HAL_TIM_SET_COMPARE(&htim11, TIM_CHANNEL_1, 0);
}
HAL_TIM_PWM_Start(&htim10, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim11, TIM_CHANNEL_1);
}
/* USER CODE END PFP */
```

Соберите проект, и загрузите прошивку на контроллер. Удостоверьтесь, что контроллер управляет двигателями постоянного тока при подаче команд по *UART1* через *Serial Port Utility* (при включенном питании на контролере от аккумуляторной батареи), где

- *F* – движение вперед;
- *B* – движение назад;

- *L* – движение влево;
- *R* – движение вправо;
- *S* – остановка движения.

Следующим шагом подключитесь к модулю обработки технического зрения при помощи *VNC Viewer.* Через панель меню откройте *Thonny IDE.*



Вставьте следующий код на языке Python в Thonny IDE и сохраните его на рабочем столе.

```
import time
import serial

serial = serial.Serial('/dev/ttyACM1', 115200, timeout=1)
serial.reset_input_buffer()

correctInputVar = [«F», «B», «L», «R», «S»]

try:
    while True:
```

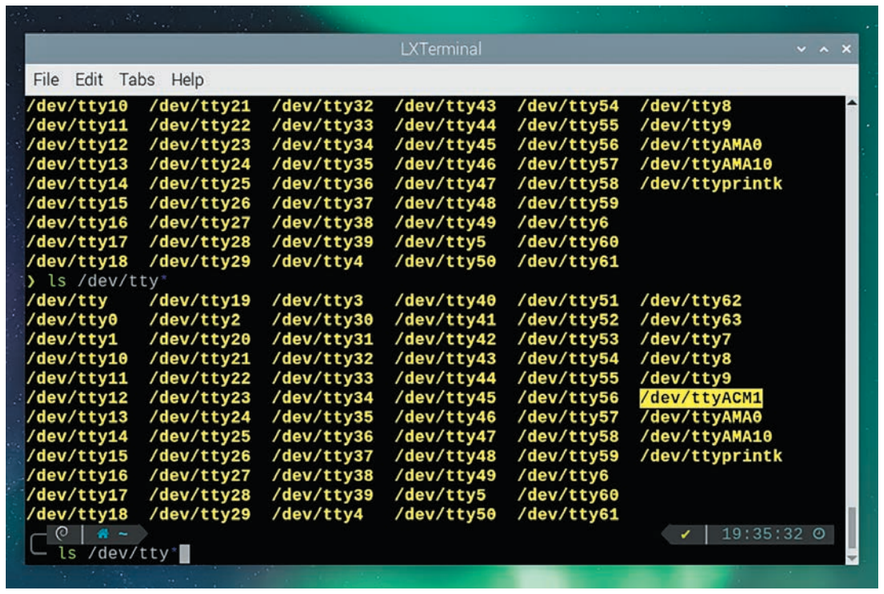
```
        inputData = input().upper()

        if inputData in correctInputVar:
            serial.write(inputData.encode('utf-8'))
            print(f»Sent: {inputData.strip()}»)

except KeyboardInterrupt:
    print(«Exiting program.»)
finally:
    serial.close()
```

Данный код будет отправлять с модуля обработки технического зрения вводимые с клавиатуры символьные команды на робототехнический контроллер, чтобы тот в свою очередь управлял движением двигателей.

Для общения модуля и контроллера по последовательному порту удостоверьтесь, что вы подключили их между собой согласно инструкции по сборке. Также для отправки команд с модуля обработки технического зрения нужно знать логический номер порта, к которому подключен робототехнический контроллер. Вы можете узнать этот номер порта, открыв терминал (значок терминала находится в верхней части экрана на панели инструментов) и прописав команду:

```
ls /dev/tty*
```

при отключенном кабеле *USB-A–USB-C*, соединяющем контроллер и модуль. На экране появится список всех портов. Подключите кабель обратно. Повторите команду в терминале, и найдите порт, который появился после подключения контроллера.



В *Thonny IDE* в код программы подставьте значение порта в строчку *serial = serial.Serial('/dev/tty*', 115200, timeout=1),* например, */dev/ttyACM1.* Запустите программу, нажав кнопку *Run* в верхней панели *IDE*, и в терминале в нижней части окна, после сообщения *%Run ваше_название_файла.py*, введите *F*, чтобы начать движение вперед.

Внимание! При работе с двигателями необходимо установить платформу робота на подставку, исключив контакт колес с поверхностью рабочего стола. Проверьте команды движения назад, влево, вправо и остановки.

Совместная работа контроллера сопряжения датчиков с модулем обработки технического зрения

Рассмотрим совместную работу контроллера сопряжения датчиков с модулем обработки технического зрения. В собранном виде контроллер уже подключен к модулю обработки технического зрения *(Raspberry Pi 5)*, к соответствующим пинам интерфейса *SPI*. Реализуем схему подключения, где *Rapberry Pi* будет выступать в роли Master, контроллер сопряжения датчиков – в роли *Slave.*

Запустите на Вашем компьютере *Arduino IDE* и подключите Ваш компьютер к контроллеру сопряжения датчиков с помощью *USB*-кабеля *Type C – Type A*. Загрузите код программы, приведенный ниже в контроллер.

```
 #include "SPI.h"
// Определяем размер буфера
#define BUFFER_SIZE 100
// Создаём буфер для данных SPI
char buff[BUFFER_SIZE];
// Создаём переменную индекса буфера
// volatile - указания для препроцессора о том,
// что переменная может измениться при прерывании
// и с ней не стоит производить никаких оптимизаций
// при компилировании
volatile uint8_t index = 0;

// Создаём флаг готовности данных
volatile bool data_ready = false;

// Создаём переменную для полученного целого числа
long i;

// Создаём индекс заголовка целого числа
volatile uint8_t int_index = 0;

void setup (void)
```

```
{
    // Инициируем работу с последовательным портом
    Serial.begin(9600);

    // Устанавливаем вывод MISO в режим выхода
    pinMode(MISO, OUTPUT);

    // Устанавливаем режим ведомого в контрольном регистре SPI (SPI Control
Register)
    SPCR |= _BV(SPE);

    // Подключаем прерывание SPI
    SPI.attachInterrupt();

}

// Вызываем функцию обработки прерываний по вектору SPI
// STC - Serial Transfer Comlete
ISR(SPI_STC_vect)
{
    // Получаем байт из регистра данных SPI
    byte c = SPDR;

    // Добавляем байт в буфер
    if (index < sizeof(buff)) {
        buff[index++] = c;

        // Как пример: байт 0xAD обозначает заголовок целого числа типа long
        if (c == 0xAD)
            // Записываем положение целого числа в байтовом массиве
            int_index = index;

        // Как пример: байт 0xAF обозначает конец пакета
        if (c == 0xAF)
            // Устанавливаем флаг готовности данных для обработки
            data_ready = true;
    }
}

void loop(void)
{
    // Если установлен флаг готовых данных
    if (data_ready == true) {
```



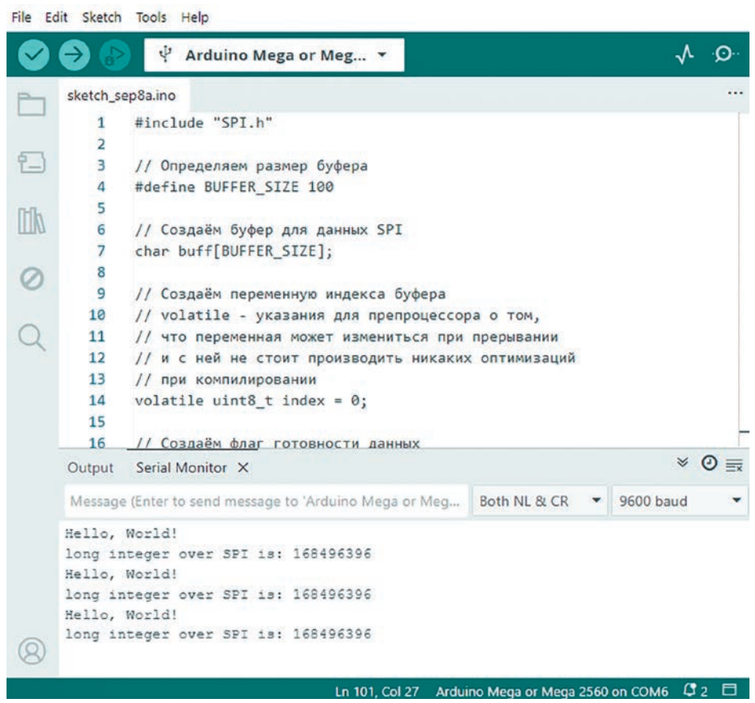
```
        // Обнуляем индекс
        index = 0;

        // Создаём строку и записываем в неё полученный буфер
        String message = String(buff);

        // Форматируем строку, убирая из неё заголовок
        // целого числа и всё после заголовка
        message = message.substring(0, int_index - 1);

        // Выводим отформатированную строку в последовательный порт
        Serial.println(message);

        // Обнуляем переменную для хранения полученного целого числа
        i = 0;

        // Записываем целое число через указатель на элемент массива
        //   Пояснение:
        //   *(выражение) - разыменовывание указателя
        //    (long *) - приводим последующее выражение к типу указателя на long
        //    buff+int_index - прибавляем к указателю на первый элемент
        //    массива buff индекс следующего после заголовка элемента массива,
        //   тем самым получая указатель на наше целое число
        i = *((long *)(buff + int_index));

        // обнуляем флаг готовности данных
        data_ready = false;

        Serial.print("long integer over SPI is: ");

        // Выводим целое число в последовательный порт
        Serial.println(i);
    }

}
```

Включите питание и после загрузки операционной системы подключитесь с помощью *VNC* к модулю обработки технического зрения. Запустите *Thonny IDE* и введите приведеннй ниже код программ на *Python:*

```
#!/usr/bin/env python3
import spidev
#import time
```

```
# Simple SPI test without any special characters
spi = spidev.SpiDev()
spi.open(0, 1)
spi.max_speed_hz = 100000

print(«SPI Test Started»)

spi.writebytes(b’Hello, World!’)
spi.writebytes([0xAD])
i = 0x0A0B0D0C
b=i.to_bytes(4, byteorder=»little»)
spi.writebytes(b)
spi.writebytes([0xAF])
```

После запуска программы в *Thonny IDE* на модуле обработки технического зрения в окне *Serial Monitor Arduino IDE* увидим сообщения, передаваемые модулем обработки технического зрения *(Raspberry Pi 5)* контроллеру сопряжения датчиков по интерфейсу *SPI.*



Для проверки возможности передачи сообщений от контроллера сопряжения датчиков модулю обработки технического зрения воспользуемся примерами программ ниже. Запустите на Вашем компьютере *Arduino IDE* и подключите Ваш компьютер к контроллеру сопряжения датчиков с помощью *USB*-кабеля *Type C – Type A*. Загрузите код программы, приведенный ниже в контроллер.


```
#include <SPI.h>
const char message[] = "hello raspberry";
volatile byte current_char = 0;
void setup() {
  pinMode(MISO, OUTPUT);
  SPCR |= _BV(SPE);
  SPCR |= _BV(SPIE);
}
ISR(SPI_STC_vect) {
  if (current_char < strlen(message)) {
    SPDR = message[current_char++];
  } else {
    SPDR = 0;
    current_char = 0;
  }
}
void loop() {
  delay(100);
}
```

Включите питание и после загрузки операционной системы подключитесь с помощью *VNC* к модулю обработки технического зрения. Запустите *Thonny IDE*, и введите приведеннй ниже код программ на *Python*:

```
#!/usr/bin/env python3
# Импортируем библиотеку для работы с SPI
import spidev
import time
# Создаём объект spi
spi = spidev.SpiDev()
# Открываем устройство SPI (/dev/spidev0.1)
spi.open(0, 1)
# Ограничиваем скорость до 1 МГц
spi.max_speed_hz = 1000000
# Выводим сообщение о запуске программы
print("SPI Receiver started")
# Начало блока обработки исключений
try:
# Бесконечный цикл для непрерывного приема данных
    while True:
# Создаем пустую строку для хранения принимаемого сообщения
message = ""
```

```
        # Цикл для приема до 20 символов (защита от бесконечного цикла)
        for i in range(20):
# Отправляем байт 0x00 и получаем ответ от slave устройства
            # xfer2 отправляет и принимает данные одновременно
            # [0] - берем первый (и единственный) байт из ответа

            data = spi.xfer2([0])[0]
# Если получен нулевой байт (0x00) - конец сообщения
            if data == 0:
                break # Выходим из цикла приема
# Преобразуем числовой байт в символ и добавляем к сообщению
            message += chr(data)
# Выводим полученное сообщение (после завершения цикла приема)
                print("Message: " + message)
# Пауза 2 секунды перед следующим опросом
        time.sleep(2)
# Обработка прерывания по Ctrl+C (пользователь остановил программу)
except KeyboardInterrupt:
    print("Stopped")# Сообщение о остановке программы
# Блок finally выполняется всегда, даже если было исключение
finally:
# Закрываем SPI соединение для корректного завершения работы
    spi.close()
```

После запуска программы в мониторе *Thonny IDE* увидим сообщения, передаваемые контроллером сопряжения датчиков по интерфейсу *SPI* модулю обработки технического зрения *(Raspberry Pi 5)*.

receiver_raspberry.py

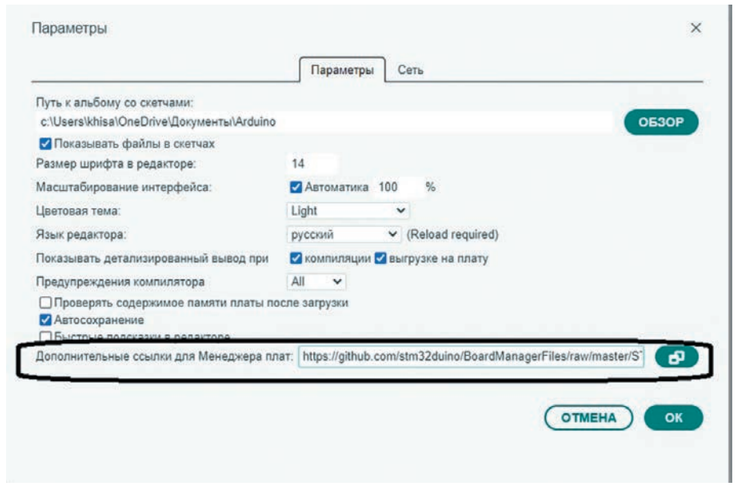
```
#!/usr/bin/env python3
import spidev
import time

spi = spidev.SpiDev()
spi.open(0, 1)
spi.max_speed_hz = 1000000

print("SPI Receiver started")

try:
    while True:
        message = ""

        for i in range(20):
            data = spi.xfer2([0])[0]
            if data == 0:
```

Оболочка

```
Message: hello raspberry
Message: hello raspberry
Message: hello raspberry
Message: hello raspberry

Python 3.11.2 (/usr/bin/python3)
>>>
```


Программирование робототехнического контроллера в среде *Arduino IDE*

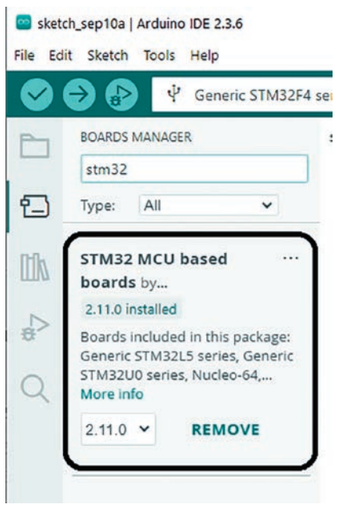
В случае, если работа с *STM32CubeMX* и *Keil* представляется сложной, начинающие пользователи могут использовать для программирования привычную среду *Arduino IDE*. Для этого необходимо добавить поддержку *STM32*.



Arduino IDE изначально не поддерживает микроконтроллеры *STM32*, но вы можете добавить их поддержку, установив ядро *STM32*. Откройте *Arduino IDE* и перейдите в «Файл» -> «Параметры...». В поле «Дополнительные ссылки для менеджера плат» добавьте следующий URL-адрес:

https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json

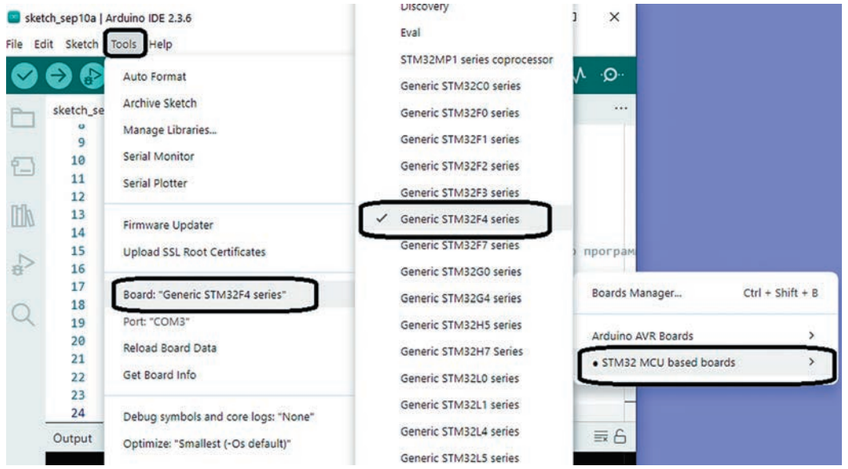
Нажмите *OK*, чтобы сохранить настройки.



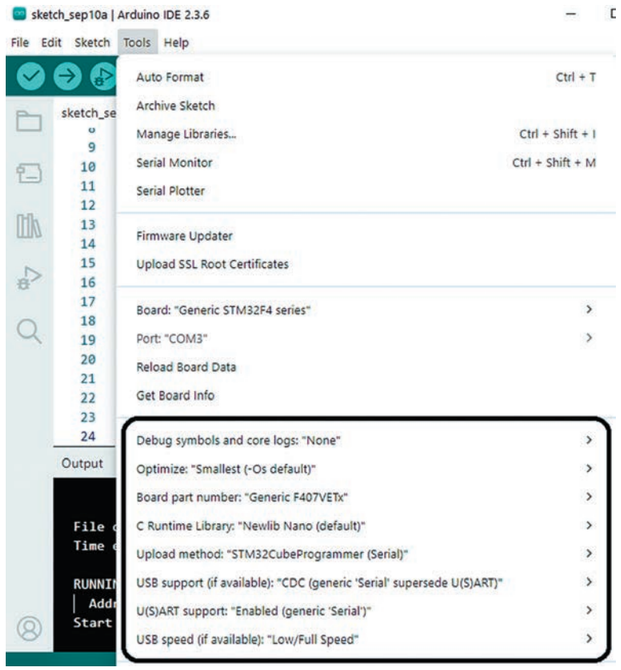
- Установите ядро STM32:

В Менеджере плат введите «*STM32*» в поле поиска. Вы должны увидеть опцию под названием *STM32 Cores by STMicroelectronics* или как показано на рисунке. Выберите ее и нажмите кнопку «Установить». Это загрузит и установит необходимые файлы для программирования микроконтроллеров *STM32* с помощью *Arduino IDE.*

- Откройте «Инструменты» -> «Плата» -> «*STM32 MCU based boards*».
- Выберите плату: *Generic STM32F4 series.*



- Выберите способ и другие параметры загрузки:



- Напишите свой код.

Теперь вы можете писать свой код *Arduino*, используя *Arduino IDE*. Вы можете использовать синтаксис и библиотеки *Arduino*, но имейте в виду, что микроконтроллеры *STM32* имеют больше функций и периферийных устройств, которые могут потребовать прямой манипуляции регистрами или использования библиотек *STM32 HAL (Hardware Abstraction Layer).*

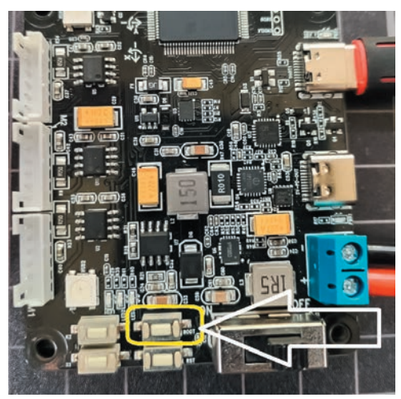
Воспользуемся кодом программы *Blink:*

```
// Blink для STM32F407VET6
// Встроенный светодиод подключен к пину PD10

void setup() {
  // Инициализация встроенного светодиода
  pinMode(PD10, OUTPUT);

  }

void loop() {
  // Включение светодиода
  digitalWrite(PD10, LOW);  // На STM32 активный низкий уровень
  delay(1000);
  // Выключение светодиода
  digitalWrite(PD10, HIGH); // На STM32 активный низкий уровень
  delay(1000);
}
```

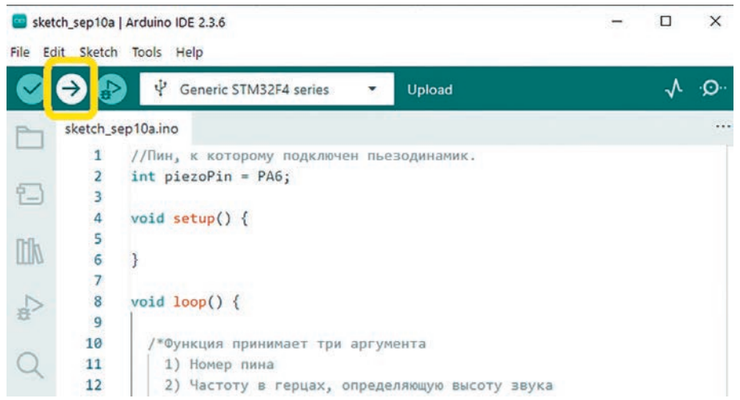
- Загрузите код.

Прежде, чем Вы подключите робототехнический контроллер к своему компьютеру с помощью *USB*-кабеля:

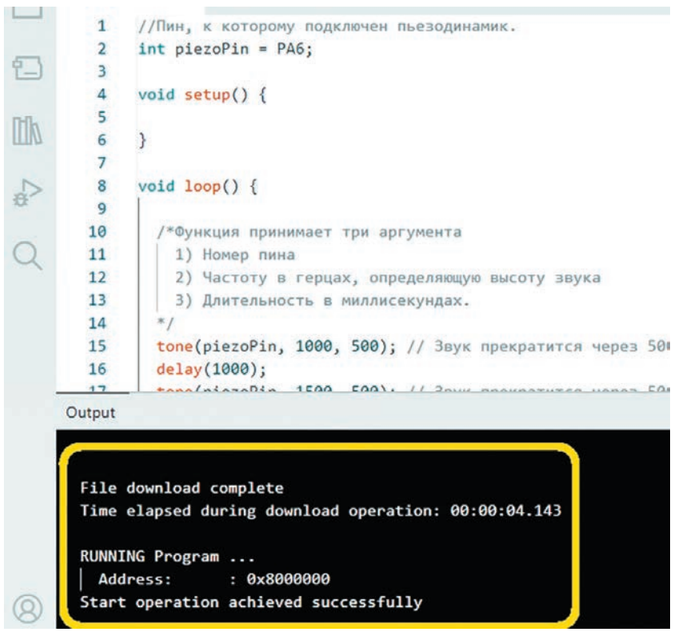
- нажмите и удерживайте кнопку *BOOT* на плате контроллера



- подключите *USB*-кабель
- нажмите кнопку компиляции и загрузки кода программы в контроллер. После начала загрузки (после компиляции) кнопку *BOOT* можно отпустить.



Код программы будет загружен в робототехнический контроллер:



Отключите *USB*-кабель и подключите снова (для выхода из режима загрузки). Синий светодиод будет моргать с заданной частотой.

Таким же образом можно загрузить код программы для работы с находящимся на плате робототехнического контроллера пассивным зуммером, требующим управляющего ШИМ-сигнала для создания звука различного тона:


```
//Пин, к которому подключен пьезодинамик.
int piezoPin = PA6;

void setup() {

}

void loop() {

  /*Функция принимает три аргумента
    1) Номер пина
    2) Частоту в герцах, определяющую высоту звука
    3) Длительность в миллисекундах.
  */
  tone(piezoPin, 1000, 500); // Звук прекратится через 500 мс, о программа останавливаться не будет!
  delay(1000);
  tone(piezoPin, 1500, 500); // Звук прекратится через 500 мс, о программа останавливаться не будет!
  /* Вариант без установленной длительности   */
 // tone(piezoPin, 2000); // Запустили звучание
  delay(1000);
  noTone(12); // Остановили звучание
}
```

Стоит отметить, что использование *Arduino IDE* с микроконтроллерами *STM32* имеет некоторые ограничения по сравнению с использованием специализированных инструментов разработки *STM32*, таких как *STM32CubeIDE* или *PlatformIO*. Эти инструменты предоставляют более продвинутые функции и лучшую интеграцию с экосистемой *STM32*. Однако, если вы знакомы с *Arduino IDE* и предпочитаете ее простоту, вышеприведенные шаги должны позволить вам начать программирование *STM32*.

8. ПРАВИЛА ТЕХНИКИ БЕЗОПАСНОСТИ ПРИ РАБОТЕ

При эксплуатации оборудования необходимо соблюдать следующие правила безопасности:

- К обслуживанию оборудования допускаются лица, изучившие паспорт и руководство по эксплуатации, а также прошедшие инструктаж по технике безопасности.
- Перед началом эксплуатации оборудования необходимо убедиться, что оборудование находится в выключенном состоянии.
- При обнаружении любых повреждений и неисправностей оборудования, а также при появлении дыма, искрения или специфического запаха перегретой изоляции, немедленно обесточьте оборудование.
- ЗАПРЕЩАЕТСЯ эксплуатировать неисправное оборудование.
- ЗАПРЕЩАЕТСЯ использовать изделие и его отдельные компоненты не по назначению.
- ЗАПРЕЩАЕТСЯ вскрывать изделие.
- ЗАПРЕЩАЕТСЯ видоизменять принципиальную схему и общие функции работы изделия.
- После хранения оборудования в холодном помещении или после перевозки в зимних условиях включать его в сеть можно не раньше, чем через 6 часов пребывания при комнатной температуре в распакованном виде.
- При эксплуатации изделия необходимо соблюдать «Правила технической эксплуатации электроустановок потребителей» и «Правила техники безопасности при эксплуатации электроустановок потребителей».
- Изделие эксплуатировать только в помещении без повышенной опасности по степени поражения электрическим током.
- Во избежание поражения электрическим током и выхода из строя элементов изделия, при работе запрещается использовать внешние источники питания.
- Не устанавливайте оборудование в непосредственной близости от легковоспламеняющихся и распространяющих огонь предметов.
- Не оставляйте оборудование включенным без присмотра.
- Не допускайте попадания жидкости внутрь оборудования.
- Не оставляйте оборудование в режиме ожидания на длительное время (более 12 часов).
- Во избежание поломок оборудования не прикладывайте чрезмерных усилий при манипуляциях с органами управления.

9. 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

РОБОТОТЕХНИКА

ZARNITZA

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

СДЕЛАНО В РОССИИ