ООО «ПРОМОБОТ»

ОБРАЗОВАТЕЛЬНАЯ РОБОТОТЕХНИЧЕСКАЯ ПЛАТФОРМА PROMOBOT M EDU

(M Edu)

Руководство по эксплуатации

ред. 27.05.2026

Информация для пользователя

Внешний вид изделия и пользовательского интерфейса может отличаться от изображений, представленных в данном документе. Компания постоянно улучшает продукт, в связи с чем данное руководство периодически пересматривается и изменяется.

Содержание

Определения, обозначения и сокращения

В настоящем документе применяют следующие термины с соответствующими определениями, сокращения и обозначения:

M Edu	–	образовательная робототехническая платформа Promobot M Edu;
Манипулятор	–	многофункциональный настольный четырехосевой робот-манипулятор с плоскопараллельной кинематикой и обратной связью, входящий в состав M Edu;
ОС	–	операционная система;
ПК	–	персональный компьютер;
ПО	–	программное обеспечение;
Приложение Promobot M Control/ Приложение	–	веб-интерфейс для настройки и управления манипулятором M Edu;
ШИМ	–	широтно-импульсная модуляция.

1 МЕРЫ БЕЗОПАСНОСТИ ПРИ ИСПОЛЬЗОВАНИИ

1.1 Общие меры безопасности

К работе с M Edu допускаются только лица, изучившие эксплуатационную документацию на M Edu, прошедшие инструктаж по технике безопасности с обязательной соответствующей отметкой в журнале инструктажа по технике безопасности.

Работающие с M Edu обязаны:

- выполнять требования эксплуатационной документации, правил электро- и пожарной безопасности;
- не допускать, чтобы сетевые и интерфейсные кабели были скручены или передавлены, а также располагать их там, где их могут легко повредить;
- контролировать все процессы во время работы;
- после завершения процессов немедленно выключить оборудование;
- избегать попадания рук и других частей тела в рабочую зону манипулятора M Edu во включенном состоянии;
- при появлении посторонних шумов прекратить работу и обесточить оборудование;
- содержать в чистоте рабочую поверхность манипулятора M Edu, проводить очистку от загрязнений и посторонних предметов;
- при появлении неисправностей сообщить об этом в сервисный центр предприятия-изготовителя.

Запрещается:

- производить действия, противоречащие эксплуатационной документации M Edu;
- оставлять работающий манипулятор M Edu без присмотра;
- позволять лицам младше 18 лет использовать M Edu в одиночку;
- работать во взрывоопасной среде, рядом с легковоспламеняющимися предметами;
- подключать дополнительное оборудование без выключения манипулятора;
- перемещать манипулятор во включенном состоянии;
- открывать и разбирать корпус манипулятора и сменных модулей;
- модифицировать (изменять или удалять элементы конструкции) манипулятор;
- производить ремонт M Edu самостоятельно;
- использовать M Edu не по назначению.

1.2 Меры безопасности при работе с модулем 3D-печати

При работе с модулем 3D-печати[1] не допускается расположение рабочего места в помещениях без наличия естественной или искусственной вентиляции.

Запрещается трогать нагретый экструдер и столик для печати. Запрещается располагать предметы в рабочей зоне модуля 3D-печати.

1.3 Меры безопасности при работе с модулем лазерной гравировки

При работе с модулем лазерной гравировки[1] допускается работать только в защитных очках.

Запрещается:

- смотреть на луч лазера;
- использовать модуль лазерной гравировки с материалами, выделяющими едкие вещества, а также отражающими металлами;
- направлять модуль лазерной гравировки на живых существ даже в случае, если он обесточен.

1.4 Меры безопасности при работе с модулем захвата вакуумного и механического

При работе с модулем захвата вакуумного и механического запрещается:

- поднимать груз, масса которого превышает значение грузоподъемности, указанное в технических характеристиках;
- выключать манипулятор M Edu, если груз находится в подвешенном положении;
- приступать к работе, если есть механические повреждения у присоски или механического захвата;
- поднимать мокрый или влажный груз;
- поднимать острые предметы.

[1] – При наличии модуля в комплектации M Edu.

2 ОБЩАЯ ИНФОРМАЦИЯ

Образовательная робототехническая платформа Promobot M Edu представляет собой программно-аппаратный комплекс, состоящий из многофункционального настольного четырехосевого робот-манипулятора с плоскопараллельной кинематикой и обратной связью, набора сменных рабочих инструментов и методических указаний, предназначенный для использования в образовательных целях.

M Edu обеспечивает пользователю возможность освоения навыков программирования, основ управления роботизированными системами и принципов работы производственных линий.

2.1 Эксплуатационные ограничения

Правила работы с M Edu:

1. Сборка, подготовка, включение, остановка и обслуживание устройства должны выполняться строго в соответствии с данным руководством.
2. Кто может работать с M Edu:
   - Самостоятельно работать с устройством могут только люди старше 18 лет.
   - Школьники и подростки могут использовать M Edu только под присмотром взрослых.
3. Безопасность при работе:
   - Всегда учитывайте ограничения, указанные в технических характеристиках устройства.
   - Не используйте M Edu, если температура, влажность или другие условия окружающей среды выходят за пределы, указанные в инструкции. Это может привести к поломке.
4. Пространство для работы:
   - Убедитесь, что вокруг M Edu достаточно свободного места. Минимальное расстояние до стен или других предметов должно быть не меньше 0,5 метра.
5. Защита от воды и повреждений:
   - Не работайте с устройством рядом с водой, чтобы избежать его повреждения.

- Берегите M Edu от ударов, падений и других механических воздействий. Это может привести к попаданию внутрь пыли, жидкости или посторонних предметов.

6. Чистка устройства:
   - Не используйте абразивные или химически активные средства для очистки. Это может повредить поверхность.
7. Питание и аксессуары:
   - Используйте только оригинальный блок питания и аксессуары, которые идут в комплекте с устройством.
   - Не подключайте M Edu к источникам питания с нестабильным напряжением.
8. Электромагнитные поля:
   - Не работайте с устройством рядом с источниками сильных электромагнитных полей (например, мощные магниты или генераторы). Это может нарушить работу электроники.

2.2 Упаковка

Упаковка M Edu (см. Рисунок 1) – это коробка из белого гофрокартона, с ручкой и ложементами внутри для защиты содержимого. При распаковке сохраняйте ее целостность для дальнейшего хранения и перевозки M Edu.

Рисунок 1 – Упаковка

2.3 Маркировка

Маркировка изделия (M Edu) есть как на упаковке, так и на самом устройстве.

На упаковке маркировка находится на боковых сторонах и содержит следующие данные:

- наименование изделия;
- напряжение питания;
- максимально потребляемая мощность;
- дата производства;
- информация о документах, в соответствии с которыми произведена M Edu;
- комплект поставки;
- срок службы;
- гарантийный срок;
- информация об изготовителе;
- информация о сертификации;
- параметры упаковки;
- информация о грузополучателе;
- информация о пункте назначения.

На манипуляторе маркировка расположена на дне основания манипулятора и содержит следующие данные:

- наименование изделия;
- масса;
- габаритные размеры;
- напряжение питания;
- максимально потребляемая мощность;
- дата производства;
- информация об изготовителе;
- информация о документах, в соответствии с которым произведена M Edu;
- информация о сертификации.

2.4 Гарантийные обязательства

Гарантийный срок производителя составляет 12 (двенадцать) месяцев. Гарантийный срок исчисляется по общему правилу с даты приобретения M Edu у предприятия-изготовителя. Гарантийный срок на M Edu, приобретенный розничным потребителем (гражданином, приобретающим M Edu исключительно для личных, семейных, домашних и иных нужд, не связанных с осуществлением предпринимательской деятельности) исчисляется с даты приобретения M Edu розничным потребителем.

При обнаружении в M Edu недостатков в период гарантийного срока владелец обязуется незамедлительно письменно сообщить об этом предприятию-изготовителю, описав подробно признаки выявленного недостатка, и в течение одного рабочего дня обеспечить по требованию предприятия-изготовителя дистанционный доступ к M Edu для дистанционной диагностики.

В период гарантийного срока предприятие-изготовитель обязуется при получении претензии обеспечить безвозмездное устранение недостатков M Edu в срок не более 60 рабочих дней с момента получения предприятием-изготовителем претензии и, в случае необходимости, предоставления дистанционного доступа к M Edu. Предприятие-изготовитель вправе по своему выбору осуществить замену M Edu ненадлежащего качества.

В случае существенного нарушения требований к качеству M Edu (обнаружения неустранимых недостатков, которые не могут быть устранены без несоразмерных расходов или

затрат времени, или выявляются неоднократно, либо неоднократно проявляются вновь после их устранения) владелец вправе потребовать замены M Edu ненадлежащего качества.

Гарантийный срок продлевается на время, в течение которого M Edu не мог использоваться из-за обнаруженных в нем недостатков, а именно на период времени, равный периоду с момента получения уполномоченным лицом претензии о недостатках M Edu до момента устранения недостатков.

Гарантийное обслуживание осуществляется по выбору предприятия-изготовителя по месту его нахождения или по месту нахождения M Edu. Гарантийному ремонту (замене) не подлежит M Edu:

- эксплуатируемая образом, не соответствующим требованиям руководства по эксплуатации;
- имеющая по вине пользователей механические повреждения, явившиеся причиной обращения за гарантийным ремонтом;
- эксплуатируемая или хранившаяся в условиях (среде), не соответствующей требованиям, установленным документацией на M Edu;
- прошедшая модификацию (изменения) или ремонт без участия предприятия-изготовителя.

Гарантийные обязательства распространяются на M Edu в той комплектации, в которой она находилась на момент поставки предприятием-изготовителем и не распространяются на недостатки, возникшие в результате неверной работы ПО Promobot M Control, разработанного (доработанного) пользователем самостоятельно без согласования с предприятием-изготовителем.

Порядок осуществления гарантийного обслуживания/ремонта M Edu установлен Положением о гарантийном ремонте и проведении сервисного обслуживания оборудования ООО «ПРОМОБОТ», размещенного по ссылке: https://promo-bot.ru/warranty-repair-and-service-provision/.

В гарантийное обслуживание (ремонт) не включены дополнительные услуги, в том числе, загрузка информации на M Edu, доработка ПО Promobot M Control, функционала, мониторинг состояния M Edu, не связанный с исправлением недостатков. Дополнительные услуги оказываются на основании отдельно заключенного возмездного соглашения, в частности соглашения об уровне сервиса (SLA).

2.5 Хранение

При хранении M Edu необходимо соблюдать следующие требования:

1. Упаковка:
   - Сохраняйте коробку и упаковочные материалы в сухом месте. Они могут понадобиться для перевозки или хранения устройства в будущем.
2. Место хранения:
   - Убедитесь, что в помещении нет сырости, испарений воды, горючих жидкостей или газов.
   - M Edu должен храниться в отапливаемом и проветриваемом помещении.
   - Избегайте попадания прямых солнечных лучей.
   - Температура должна быть от +5°C до +40°C (рекомендуемая +25°C). СанПиН
   - Влажность воздуха не должна превышать 65%.
3. Подготовка к хранению:
   - Перед тем как убрать устройство на хранение, протрите его корпус сухой мягкой тканью.
   - Проверьте, чтобы на рабочих поверхностях манипулятора и сменных модулей не осталось посторонних материалов.
4. Что нельзя делать:
   - Не кладите тяжелые предметы на коробку с M Edu.
   - Не допускайте посторонних людей к месту хранения устройства.

2.6 Транспортирование

При транспортировании соблюдайте следующие правила:

1. Используйте оригинальную упаковку – перевозите M Edu только в коробке, в которой он был изначально упакован.
2. Условия перевозки:
   - Перевозить устройство можно любым крытым транспортом.
   - Рекомендуемая температура: от +10°C до +35°C.
   - Влажность воздуха не должна превышать 70%.

3. Если перевозили M Edu на холоде, при минусовой температуре, то перед включением оставьте манипулятор в теплом помещении на 2–3 часа, чтобы он прогрелся до температуры не ниже +10°C.
4. Как упаковать:
   - Убедитесь, что манипулятор, сменные модули и другие детали лежат на своих местах внутри коробки.
   - Не ставьте коробку вертикально – она должна лежать ровно.
   - Перед транспортировкой проверьте, чтобы внутри коробки не было посторонних предметов.
5. Бережная перевозка:
   - Избегайте ударов и резких движений коробки во время перевозки.
   - Помните, что упаковка с M Edu – это хрупкий груз, поэтому обращайтесь с ней аккуратно.

2.7 Утилизация

Срок эксплуатации M Edu – 3 года.

Если M Edu повреждена так, что ее больше нельзя использовать –, утилизируйте.

Для предотвращения загрязнения окружающей среды все отходы, образующиеся при утилизации M Edu и ее частей, подлежат обязательному сбору с последующей утилизацией в установленном порядке и в соответствии с действующими требованиями и нормами отраслевой нормативной документации, в том числе в соответствии с СанПиН 2.1.3684-21 «Гигиенические требования к размещению и обезвреживанию отходов производства и потребления».

Если это необходимо для налогового учета, операция по утилизации должна быть отражена в бухгалтерских документах в соответствии с законодательством той страны, в которой установлено оборудование.

3 ОПИСАНИЕ УСТРОЙСТВА

3.1 Технические характеристики

Технические характеристики M Edu представлены в таблице 1.

Таблица 1 – Технические характеристики M Edu

Параметр	Единица измерения	Значение
Грузоподъемность	кг	0,5
Количество степеней свободы	–	4
Радиус рабочей зоны	мм	385
Максимальная линейная скорость центральной точки инструмента	мм/с	100
Точность позиционирования	мм	0,1
Повторяемость позиционирования	мм	0,2
Степень защиты корпуса	–	IP20
Номинальное переменное напряжение на входе блока питания	В	230
Номинальное постоянное напряжение на выходе блока питания	В	12
Потребляемая мощность, не более	Вт	180
Масса манипулятора без дополнительного оборудования, не более	кг	7
Масса комплекта в упаковке, не более	кг	12
Габаритные размеры манипулятора без дополнительного оборудования, Д×Ш×В, не более	мм	316×220×408
Габаритные размеры комплекта в упаковке, Д×Ш×В, не более	мм	528×385×275
Встроенный компьютер Raspberry Pi 5 для автономной работы	–	Наличие

Параметр	Единица измерения	Значение
Встроенный светодиодный экран	–	Наличие
Встроенный динамик	–	Наличие
Встроенный микрофон	–	Наличие
Голосовой помощник	–	Наличие
Прямое управление узлами манипулятора	–	Наличие
Режим FreeDrive	–	Наличие
Беспроводные интерфейсы для внешних коммуникаций	–	Wi-Fi, Bluetooth
Проводные интерфейсы для внешних коммуникаций	–	Ethernet, HDMI, USB, UART, RS-485, SPI, I2C, TTL, 1-Wire
Интерфейсы для подключения датчиков и исполнительных механизмов	–	цифровые входы, цифровые выходы, входы АЦП, выходы ШИМ
Материал 3D-печати (сменный экструдер для 3D-печати)[2]	–	PLA-филамент
Точность 3D-печати (сменный экструдер для 3D-печати)[2]	мм	1
Мощность лазерного гравера (сменный модуль для лазерной гравировки)[1], не более	мВт	500
Длина волны лазерного гравера (сменный модуль для лазерной гравировки)[1]	нм	405–650
Диаметр вакуумной присоски (сменный захват вакуумный)[2], не более	мм	23
Мощность насоса модуля захвата вакуумного[2], не более	Вт	6
Сила сжатия захвата механического (сменный захват механический), не более	Н	8
Максимальный раствор когтей модуля захвата механического, не более	мм	80

[1] – Параметр учитывается только в полной комплектации M Edu.

[2] – Параметр не учитывается в комплектации M Edu Старт

3.2 Состав M Edu

M Edu поставляется в трех комплектациях:

1. Полная комплектация – включает в себя манипулятор и 5 сменных насадок (вакуумный захват, механический захват, захват пишущего инструмента, модуль 3D-печати, модуль лазерной гравировки).
2. Комплектация без модуля лазерной гравировки – включает в себя манипулятор и 4 сменных насадки (вакуумный захват, механический захват, захват пишущего инструмента, модуль 3D-печати).

Комплектация M Edu Старт – включает в себя манипулятор и 2 сменных насадки (механический захват, захват пишущего инструмента).

Состав M Edu включает комплектующие, представленные в таблице 2.

Таблица 2 – Состав M Edu

№	Наименование	Краткое описание	Количество
1	Информационный лист	Ссылка и QR-код для получения пользовательской документации и методических пособий	1 шт.
2	Манипулятор	Многофункциональный настольный четырехосевой робот-манипулятор с плоскопараллельной кинематикой и обратной связью	1 шт.
3	Блок питания	Импульсный блок питания с кабелем для питания от сети переменного тока 230 В 50 Гц, вилка стандарта CEE 7/4 (тип F) или 7/7 (тип E/F) с заземлением и выходным постоянным напряжением 12 В	1 шт.
4	Пульт управления[2]	Проводной USB-геймпад для ручного управления M Edu	1 шт.

№	Наименование	Краткое описание	Количество
5	Печатающая головка[2]	Головка для печати PLA-филаментом; входит в комплект модуля 3D-печати	1 шт.
6	Экструдер[2]	Настольный блок с сервоприводом для подачи PLA-филамента к печатающей головке; входит в комплект модуля 3D-печати	1 шт.
7	Трубка тефлоновая[2]	Трубка для подачи PLA-филамента к печатающей головке; входит в комплект модуля 3D-печати	1 шт.
8	Держатель катушки PLA-филамента[2]	Две направляющие для установки катушки PLA-филамента; входит в комплект модуля 3D-печати	1 шт.
9	Защитное стекло[2]	Стекло для защиты поверхности при 3D-печати; входит в комплект модуля 3D-печати	1 шт.
10	Тестовый PLA-филамент[2]	PLA-филамент для проверки функции 3D-печати; входит в комплект модуля 3D-печати	10 м
11	Модуль лазерной гравировки[1]	Модуль лазерный красный 405–650 нм 500 мВт с фокусировкой	1 шт.
12	Очки защитные[1]	Защитные очки от фиолетового, синего и красного лазерного излучения длиной волны 405–450 нм и 635–660 нм	1 шт.
13	Внешний блок коммутации инструмента[2]	Блок с вакуумным насосом для модуля захвата вакуумного и безопасной коммутации питания модуля лазерной гравировки[1], с полиуретановой трубкой	1 шт.
14	Ключ коммутации питания модуля лазерной гравировки[1]	Ключ от ключ-выключателя, предназначенного для безопасной коммутации питания модуля лазерной гравировки	2 шт.

№	Наименование	Краткое описание	Количество
15	Модуль захвата пишущего инструмента	Захват для пишущего инструмента диаметром до 10 мм	1 шт.
16	Ручка Promobot	Шариковая ручка; используется совместно с модулем захвата пишущего инструмента	1 шт.
17	Поворотный модуль инструмента	Блок сервопривода для обеспечения вращения инструмента (не используется для модуля 3D-печати, модуля лазерной гравировки и модуля захвата пишущего инструмента)	1 шт.
18	Модуль захвата вакуумного[2]	Модуль с вакуумной присоской	1 шт.
19	Модуль захвата механического	Блок инструмента с двумя акриловыми когтями, приводимыми в движение сервоприводом	1 шт.
20	Шнур сетевого интерфейса ETHERNET	Ответный шнур разъема сетевого интерфейса ETHERNET	1 шт.
21	Подложка с разметкой[2]	Рабочее поле манипулятора с разметкой для точного размещения объектов и удобства использования при обучении	1 шт.
22	Встроенное программное обеспечение Promobot M Control	ПО, предназначенное для управления манипулятором	1 шт.
23	Краткая инструкция для пользователя M EDU	Краткая инструкция по включению / выключению M Edu, смене насадок с QR-кодами на видеоинструкции	1 шт.

№	Наименование	Краткое описание	Количество
24	Упаковочный лист	Товаросопроводительный документ, представляющий собой детальную опись содержимого каждого грузового места	1 шт.
25	Паспорт	Эксплуатационный документ, содержащий основные технические характеристики, параметры, сведения о производителе, комплектации, гарантии и правила безопасной эксплуатации товара	1 шт.
26	Пинцет для снятия разъемов	Специализированный ручной инструмент, предназначенный для безопасного и быстрого извлечения контактов из электрических разъемов	1 шт.
[1] – Включено только в полную комплектацию M Edu. [2] – Не включено в комплектацию M Edu Старт			

3.3 Манипулятор

Основным компонентом M Edu является настольный 4-осевой манипулятор (см. Рисунок 2), который состоит из:

- основания;
- башни;
- полиуретанового корпуса.

Чтобы устройством было удобно и безопасно пользоваться, в комплект входят защитные элементы, провода и расходники, которые помогают получить доступ ко всем функциям.

M Edu можно улучшать и расширять, подключая к нему дополнительные совместимые модули. Это делает устройство более универсальным и полезным для разных задач.

Рисунок 2 – Настольный 4-х осевой манипулятор

3.3.1 Основание

Основание манипулятора – это неразборный блок, включающий в себя дисплей и разъемы для внешних подключений. Дисплей, расположенный на передней панели, отображает статус работы манипулятора с помощью изображений формата GIF (см. Рисунок 3).

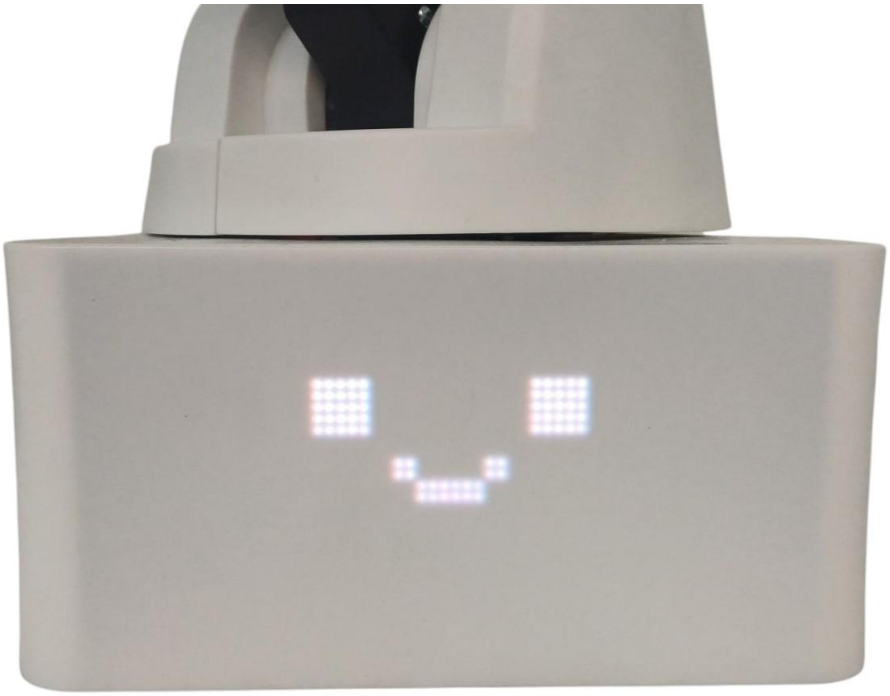


Рисунок 3 – Дисплей на основании манипулятора

Манипулятор визуально сопровождает свои действия следующими изображениями формата GIF:

 – при включении;

 – в режиме ожидания;

 – во время работы программы;

 – после остановки.

Разъемы для внешних подключений расположены на задней панели (см. Рисунок 4):

Рисунок 4 – Панель разъемов для подключения внешних устройств

1. Тумблер включения/выключения питания манипулятора:
   - «0» – питание выключено;
   - «|» – питание включено;
2. UART – для подключения внешнего оборудования по последовательному интерфейсу UART;
3. ETHERNET – для подключения манипулятора к компьютеру;
4. USB – для подключения элементов управления (компьютерная мышь, клавиатура, пульт управления);
5. GPIO – для подключения внешних датчиков и исполнительных механизмов.
6. HDMI – для подключения монитора;

7. RS-485 – для подключения внешнего оборудования по последовательному интерфейсу RS-485;
8. STEP/DIR – для прямого управления драйверами шаговых двигателей по интерфейсу STEP/DIR;
9. TTL – для подключения сервопривода с интерфейсом TTL;
10. STEPPER – для подключения шагового двигателя;
11. 12v OUT – для питания внешнего оборудования (блок коммутации инструмента, нагревательный элемент печатающей головки);
12. POWER 12v – для подключения блока питания манипулятора.

Для правильного и безопасного подключения дополнительных совместимых модулей к манипулятору пользуйтесь информацией о «распиновке разъемов».

Также в основании расположен динамик для обеспечения обратной связи пользователю при взаимодействии с M Edu.

В качестве «мозга», основного вычислительного модуля M Edu, используется одноплатный компьютер Raspberry Pi 5 (см. Рисунок 5) с операционной системой. Это делает устройство автономным (для работы с M Edu не нужен компьютер).

Рисунок 5 – Одноплатный компьютер Raspberry Pi 5

Управление дополнительными системами M Edu осуществляется через встроенный микроконтроллер STM32.

Для выполнения всех функций M Edu достаточно подключить к манипулятору монитор и элементы управления (компьютерная мышь, клавиатура).

Также можно использовать внешний компьютер, чтобы разделить задачи: Raspberry Pi 5 будет управлять манипулятором, а компьютер – отвечать за графический интерфейс.

На корпусе основания находится кнопка включения/выключения манипулятора (см. Рисунок 6). При нажатии на нее манипулятор выключается.

Эта кнопка удобна тем, что можно выключить устройство, не трогая основной переключатель питания (тумблер). Если необходимо снова включить манипулятор, просто нажмите эту кнопку еще раз.

Рисунок 6 – Кнопка включения/выключения манипулятора

3.3.2 Башня

Башня закреплена на основании манипулятора. На башне закреплены плечо и стрела манипулятора (см. Рисунок 7).

Рисунок 7 – Манипулятор, где 1 – основание; 2 – башня; 3 – плечо; 4 – стрела; 5 – блок инструмента

Движение плеча и стрелы манипулятора происходит благодаря трем поворотным узлам (J1, J2, J3) (см. Рисунок 8). Эти узлы приводятся в движение шаговыми двигателями, которые работают через ременные передачи. Чтобы контролировать положение узлов, используются энкодеры (датчики положения) и концевые выключатели (датчики, которые показывают крайние точки движения).

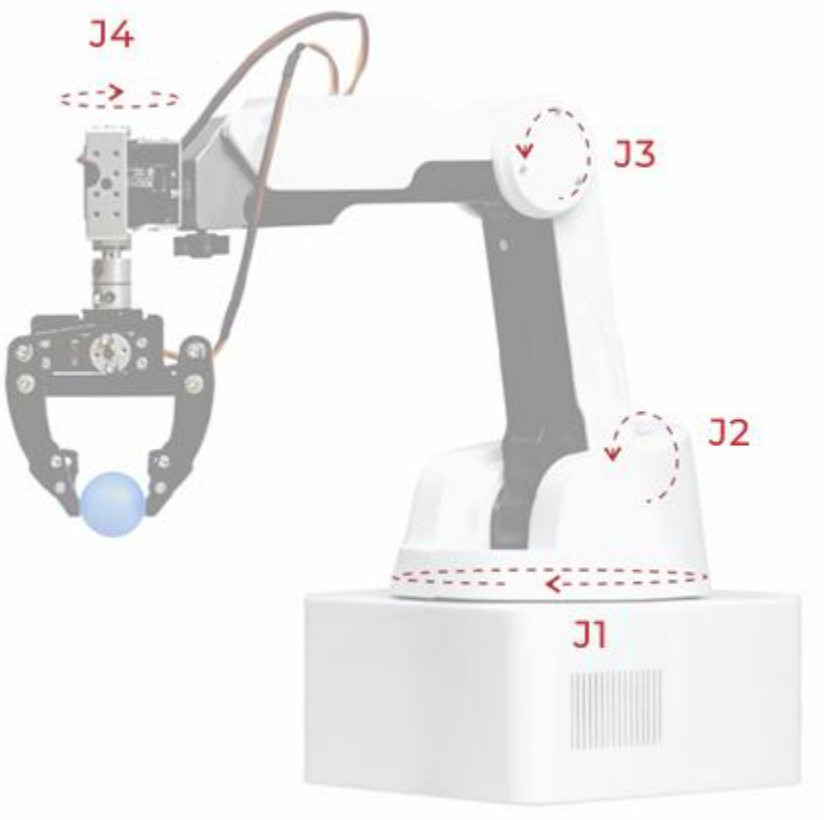


Рисунок 8 – Движение плеча и стрелы манипулятора

На стреле расположен блок инструмента и панель разъемов для подключения сменных модулей с кнопкой FreeDrive. При нажатии и удерживании кнопки FreeDrive стрела манипулятора «расслабляется» и ее можно переместить в нужное положение (см. Рисунок 9).

Рисунок 9 – Стрела манипулятора (вид сверху), где: 6 – панель разъемов для подключения сменных модулей; 7 – кнопка FreeDrive

На панели разъемов для подключения сменных модулей расположены разъемы (сверху вниз):

1. TEMP – для подключения датчика температуры печатающей головки и лазерной головки[1].
2. FAN – для подключения вентилятора печатающей головки.
3. GP3 – для подключения поворотного модуля инструмента.
4. GP3 – для подключения привода модуля захвата механического.

Для правильного и безопасного подключения дополнительных совместимых модулей к манипулятору пользуйтесь информацией о «распиновке разъемов».

[1] – При наличии в комплектации M Edu сменного модуля лазерной гравировки.

3.3.3 Порядок расположения контактов («распиновка разъемов»)

Для правильного и безопасного подключения дополнительных совместимых модулей к манипулятору пользуйтесь информацией о «распиновке разъемов» для внешних подключений, расположенных на задней панели (см. Рисунок 10) и о «распиновке разъемов» для подключения на стреле манипулятора (см. Рисунок 11).

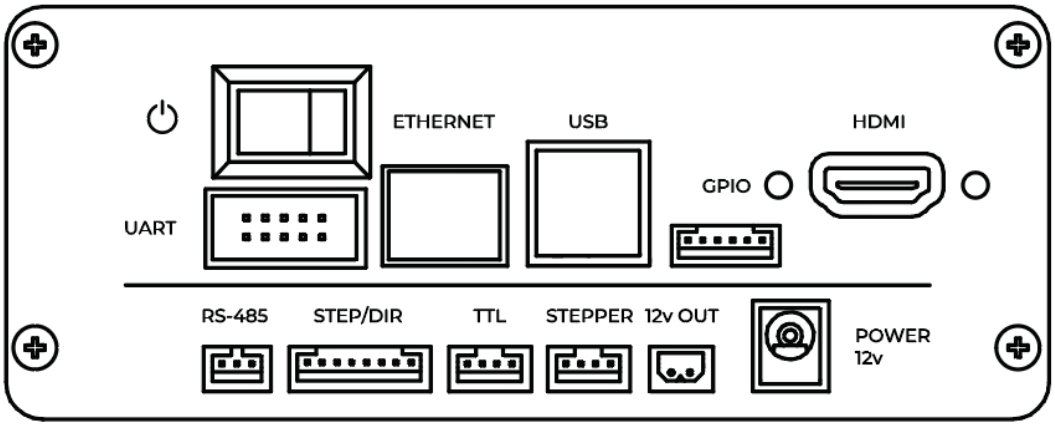


Рисунок 10 – Разъемы для внешних подключений

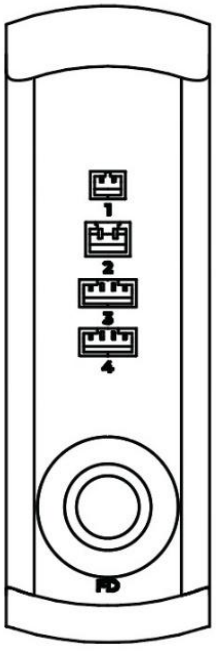


Рисунок 11 – Разъемы для подключения на стреле манипулятора

Порядок расположения контактов («распиновка разъемов») для внешних подключений описан в таблице 3.

Примечания – Если на порт GPIO разъема UART подать напряжение выше 3,3 В, то это приведет к замыканию на контакт GND (Земля) и выходу M Edu из строя.

Таблица 3 – Порядок расположения контактов разъемов на задней панели

Разъем	**Наименование ответного разъема на кабель**	**Номер контакта**	**Наименование контакта**	**Тип контакта**	**Параметры сигнала**
UART * Первый контакт справа сверху. Нумерация сверху вниз, справа налево	IDC-10F производитель Connfly. Маркировка первого контакта указана на корпусе в виде треугольника	1	12 В	Выход	Максимальный ток: 100 мА
		2	GND	Земля	Максимальный ток: 1 А
		3	E2	Вход/Выход (GPIO)	Максимальный ток: 20 мА Допустимое напряжение: 0...3,6 В
		4	E1	Вход/Выход (GPIO)	Максимальный ток: 20 мА Номинальное напряжение: 0...3,3 В
		5	nRS	Вход/Выход (GPIO)	Максимальный ток: 20 мА Номинальное напряжение: 0...3,3 В
		6	STOP	Вход/Выход (GPIO)	Максимальный ток: 20 мА Номинальное напряжение: 0...3,3 В

Разъем	Наименование ответного разъема на кабель	Номер контакта	Наименование контакта	Тип контакта	Параметры сигнала
		7	RX	Вход (UART), Вход/Выход (GPIO)	Максимальный ток: 20 мА Номинальное напряжение: 0...3,3 В
		8	TX	Выход (UART), Вход/Выход (GPIO)	Максимальный ток: 20 мА Номинальное напряжение: 0...3,3 В
		9	5 В	Выход	Максимальный ток: 100 мА
		10	GND	Земля	Максимальный ток: 1 А
GPIO * Первый контакт расположен слева	PHR-6 производитель JST	1	Порт 1/SDA	Вход АЦП/Выход ШИМ/Интерфейсный порт	Допустимое напряжение: 0...5,5 В Максимальный ток: 20 мА
		2	Порт 2/SCL	Вход АЦП/Выход ШИМ/Интерфейсный порт	Допустимое напряжение: 0...5,5 В Максимальный ток: 20 мА

Разъем	Наименование ответного разъема на кабель	Номер контакта	Наименование контакта	Тип контакта	Параметры сигнала
		3	Порт 3/MOSI	Вход АЦП/Выход ШИМ/Интерфейсный порт	Допустимое напряжение: 0...5,5 В Максимальный ток: 20 мА
		4	Порт 4/MISO	Вход АЦП/Выход ШИМ/Интерфейсный порт	Допустимое напряжение: 0...5,5 В Максимальный ток: 20 мА
		5	Порт 5/SCK	Вход АЦП/Выход ШИМ/Интерфейсный порт	Допустимое напряжение: 0...5,5 В Максимальный ток: 20 мА
		6	GND	Земля	Максимальный ток: 1 А
12V OUT * Первый контакт расположен слева	XHP-2 производитель JST	1	Управляемая земля	Выход	Максимальный ток: 2 А
		2	12 В	Выход	Максимальный ток: 2 А

Порядок расположения контактов («распиновка разъемов») на стреле манипулятора описан в таблице 4.

Таблица 4 – Порядок расположения контактов разъемов на стреле манипулятора

Разъем	**Название ответного разъема на кабель**	**Номер контакта**	**Название контакта**	**Тип контакта**	**Параметры сигнала**
3. GP3	PHR-4 производитель JST	1	TTL/PWM	Интерфейсный порт/Выход ШИМ	Допустимое напряжение: 0...5,5 В Максимальный ток: 2 А
		2	АЦП	Вход	Допустимое напряжение: 0...5,5 В
		3	5 В	Выход	Максимальный ток: 1 А
		4	GND	Земля	–
4. GP3	PHR-4 производитель JST	1	TTL/PWM	Интерфейсный порт/Выход ШИМ	Допустимое напряжение: 0...5,5 В Максимальный ток: 2 А
		2	АЦП	Вход	Допустимое напряжение: 0...5,5 В
		3	5 В	Выход	Максимальный ток: 1 А
		4	GND	Земля	–

3.4 Блок питания

Блок питания (см. Рисунок 12) – это устройство, которое превращает переменное напряжение 230 В (как в розетке) в постоянное напряжение 12 В. У него есть специальный контакт для заземления, который делает использование M Edu безопасным.

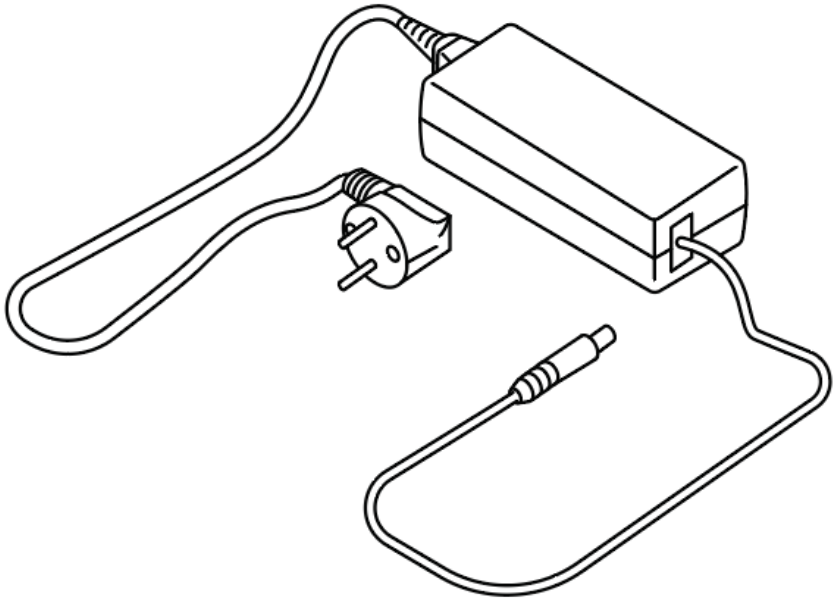


Рисунок 12 – Блок питания

Для подключения блока питания к M Edu, вставьте его в разъем POWER 12v на панели разъемов для подключения внешних устройств.

Примечание – Подключение блока питания к M Edu разрешается только при установке тумблера в положении «0». Использование M Edu запрещается при поврежденных элементах корпуса или кабелей блока питания.

3.5 Внешний блок коммутации инструмента

Внешний блок коммутации инструмента (см. Рисунок 13) предназначен для безопасной коммутации питания модуля лазерной гравировки[1] и работы модуля захвата вакуумного.

[1] – При наличии в комплектации M Edu сменного модуля лазерной гравировки.

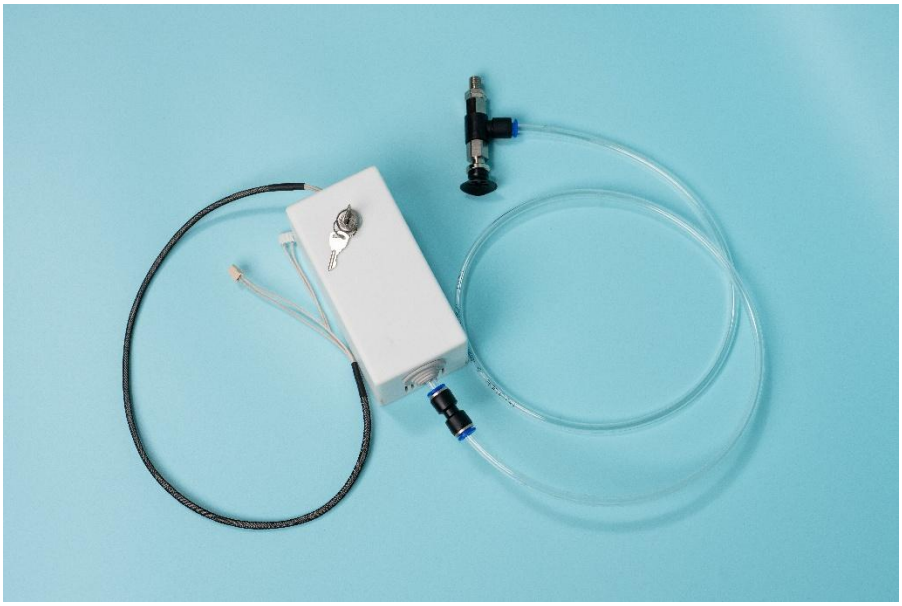


Рисунок 13 – Внешний блок коммутации инструмента

Внутри блока установлен вакуумный насос для присасывания предметов вакуумной присоской. Для этого блок коммутации инструмента требуется подключить к разъему 12v OUT на панели разъемов для подключения внешних устройств.

Снаружи блока – ключ-выключатель для включения/выключения питания лазерной головки. Для этого блок коммутации инструмента требуется подключить к разъему TTL на панели разъемов для подключения внешних устройств.

3.6 Поворотный модуль инструмента

В поворотном модуле инструмента (см. Рисунок 14) располагается четвертый поворотный узел (J4) манипулятора.

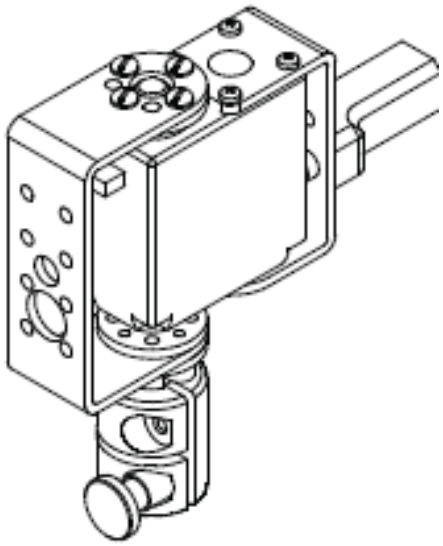


Рисунок 14 – Поворотный модуль инструмента

Поворотный модуль инструмента вставляется в специальное гнездо в блоке инструмента (см. Рисунок 15).

Рисунок 15 – Установка поворотного модуля инструмента

После этого он закрепляется винтом, который прижимает его и надежно фиксирует. Провода поворотного модуля подключаются к разъему 3. GP3 на панели разъемов для подключения сменных модулей (см. Рисунок 16).

Рисунок 16 – Подключение проводов поворотного модуля

Через поворотный модуль подключаются механический и вакуумный захваты.

Примечание – Все монтажные работы допускается производить только на обесточенном оборудовании.

3.7 Модуль захвата вакуумного

Модуль захвата вакуумного предназначен для того, чтобы перемещать предметы.

Модуль состоит из присоски и вакуумного насоса, расположенного в блоке коммутации инструмента (см. Рисунок 17).

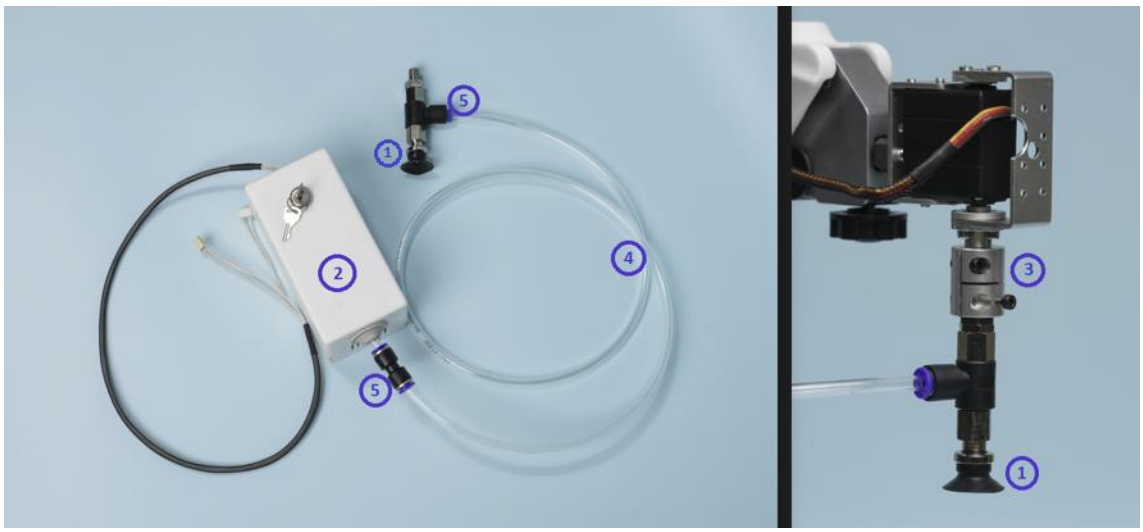


Рисунок 17 – Модуль захвата вакуумного, где: 1 – присоска; 2 – вакуумный насос; 3 – винты на муфте; 4 – полиуретановая трубка; 5 – фитинговые держатели

Принцип работы: между присоской и поверхностью предмета создается низкое давление (вакуум), благодаря чему предмет прилипает к присоске и его можно перемещать.

Установка и подключение модуля захвата вакуумного:

Примечание – Все монтажные работы допускается производить только на обесточенном оборудовании.

1. Установите присоску вакуумного захвата в поворотный модуль инструмента:
    1) Возьмите поворотный модуль, ослабьте винты на муфте.
    2) Закрутите присоску в поворотный модуль и затяните винты на муфте.
2. Присоедините полиуретановую трубку:
    1) Один конец трубки присоедините к присоске. Для закрепления используются фитинговые держатели.
    2) Второй конец трубки присоедините к блоку коммутации инструмента. Для закрепления также используются фитинговые держатели.

Примечания – Не отсоединяйте полиуретановую трубку от блока коммутации и присоски после использования модуля, их можно хранить в сборе.

3. Установите поворотный модуль инструмента в специальное гнездо в блоке инструмента и подключите провода.
4. Подключите блок коммутации инструмента к манипулятору.

5. Включите манипулятор, руководствуясь инструкцией.

Модуль вакуумного захвата готов к работе.

3.8 Модуль захвата механического

Модуль захвата механического предназначен для того, чтобы перемещать предметы. Модуль состоит из привода и двух когтей (см. Рисунок 18).

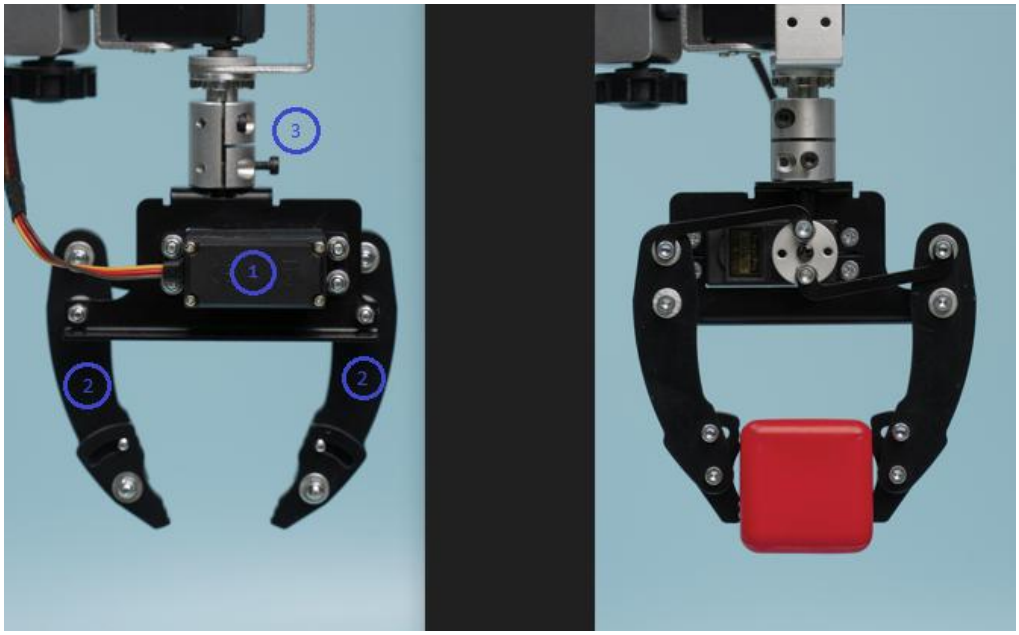


Рисунок 18 – Модуль захвата механического, где: 1 – привод механического захвата, 2 – когти, 3 – винты на муфте

Установка и подключение модуля захвата механического:

Примечание – Все монтажные работы допускается производить только на обесточенном оборудовании. Отключение проводов на стреле манипулятора допускается осуществлять только за корпус разъема с помощью пинцета – выдергивание разъемов за провода может привести к их повреждению.

1. Установите модуль механического захвата в поворотный модуль инструмента:
   1) Возьмите поворотный модуль, ослабьте винты на муфте (см. Рисунок 18).
   2) Закрутите захват в поворотный модуль и затяните винты на муфте.
2. Установите поворотный модуль инструмента в специальное гнездо в блоке инструмента и подключите провода поворотного модуля.
3. Подключите провода механического захвата к разъему 4. GP3 на панели разъемов для подключения сменных модулей (см. Рисунок 19).

Рисунок 19 – Подключение проводов механического захвата

4. Включите манипулятор, руководствуясь инструкцией.

Модуль захвата механического готов к работе.

3.9 Модуль захвата пишущего инструмента

Модуль захвата пишущего инструмента (см. Рисунок 20) предназначен для рисования. Модуль состоит из захвата пишущего инструмента и держателя захвата.

Принцип работы: в захват вставляется ручка или фломастер диаметром до 10 мм. Внутри захвата есть пружина, которая слегка нажимает на пишущий инструмент, чтобы он касался бумаги. Благодаря этому можно писать или рисовать.

Установка и подключение модуля захвата пишущего инструмента:

1. Возьмите захват и ослабьте винт.
2. Вставьте ручку (или фломастер) так, чтобы кончик выступал примерно на 4,5 см вниз и закрутите винт, чтобы зафиксировать.

3. Установите держатель захвата в специальное гнездо в блоке инструмента и закрепите винтом.
4. Расположите бумагу под стрелой манипулятора на ровную поверхность, которая не скользит, или закрепите края бумаги скотчем.
5. Включите манипулятор, руководствуясь инструкцией.

Модуль захвата пишущего инструмента готов к работе.

Рисунок 20 – Модуль захвата пишущего инструмента, где: 1 – захват пишущего инструмента; 2 – держатель захвата; 3 – винт

3.10 Модуль 3D-печати

Модуль 3D-печати (см. Рисунок 21) предназначен для печати 3D-объектов PLA-филаментом.

Рисунок 21 – Модуль 3D-печати

Модуль состоит из следующих компонентов:

1. Экструдер (блок подачи PLA-филамента).
2. Печатающая головка.
3. Тефлоновая трубка для подачи PLA-филамента.
4. Держатель для катушки с PLA-филаментом.
5. Защитное стекло.

Принцип работы: экструдер через тефлоновую трубку подает пластиковую нить (PLA-филамент) в печатающую головку. Там нить нагревается до нужной температуры, становится жидкой и наносится на защитное стекло. Манипулятор двигает печатающую головку так, чтобы получилась нужная фигура.

Установка и подключение модуля 3D-печати:

Примечание – Все монтажные работы допускается производить только на обесточенном оборудовании. Отключение проводов на стреле манипулятора допускается осуществлять только за корпус разъема с помощью пинцета – выдергивание разъемов за провода может привести к их повреждению.

1. Установите и подключите печатающую головку:
    1) Возьмите печатающую головку и приведите ее в рабочее положение (см. Рисунок 22).

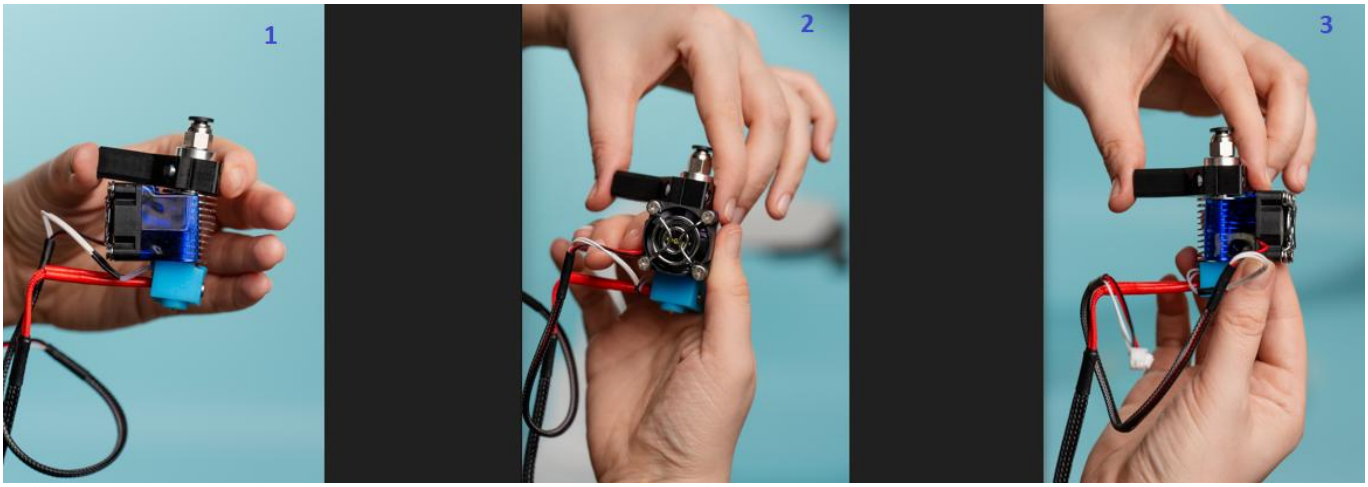


Рисунок 22 – Приведение печатающей головки в рабочее положение

2) Установите печатающую головку в блок инструмента и закрепите винтом.
3) Подключите провод нагревательного элемента к разъему 12v OUT на панели разъемов для подключения внешних устройств.
4) Подключите провода датчика температуры и вентилятора к разъемам 1. TEMP и 2. FAN на стреле манипулятора (см. Рисунок 23).

Рисунок 23 – Подключение провода датчика температуры и вентилятора

2. Установите и подключите экструдер:

Примечание – Для осуществления тестовой печати в комплект также входит PLA-филамент 10 м.

1) Возьмите тефлоновую трубку и с помощью фитинговых держателей присоедините один конец к печатающей головке, а другой – к экструдеру.
2) Установите катушку PLA-филамента с держателем на стол (при наличии).
3) Возьмите экструдер и открутите на нем прижимной винт, откройте его (см. Рисунок 24).

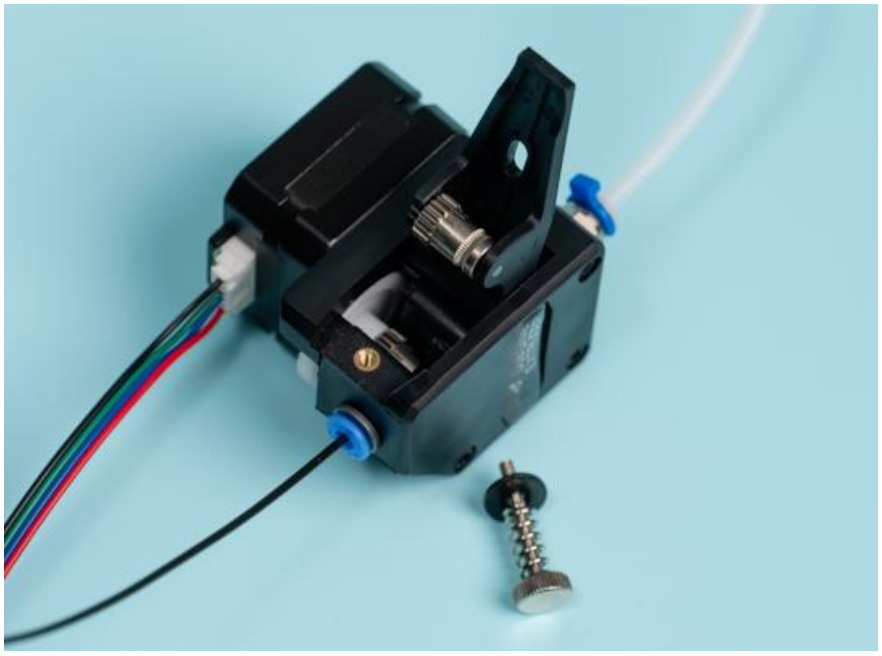


Рисунок 24 – Экструдер

4) Возьмите пластиковую нить и заправьте ее через экструдер по тефлоновой трубке до печатающей головки (до упора) (см. Рисунок 25).

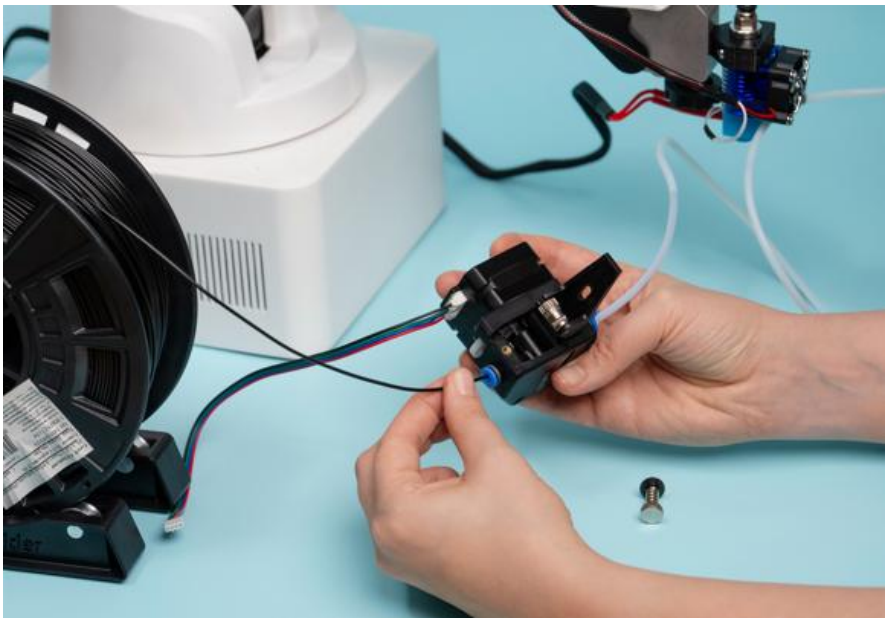


Рисунок 25 – Заправка пластиковой нити в экструдер

5) Закройте экструдер и закрутите на нем прижимной винт.
6) Подключите провода экструдера к разъему STEPPER на панели разъемов для подключения внешних устройств (см. Рисунок 26).

Рисунок 26 – Подключение проводов экструдера

3. Подготовьте поверхность для 3D-печати (см. Рисунок 27):
    1) Расположите защитное стекло на поверхности стола в 10 см от корпуса манипулятора и закрепите его бумажным скотчем по краям.
    2) Наклейте в центр стекла бумажный скотч 10×10 см для того, чтобы объект при печати не приклеился к стеклу.

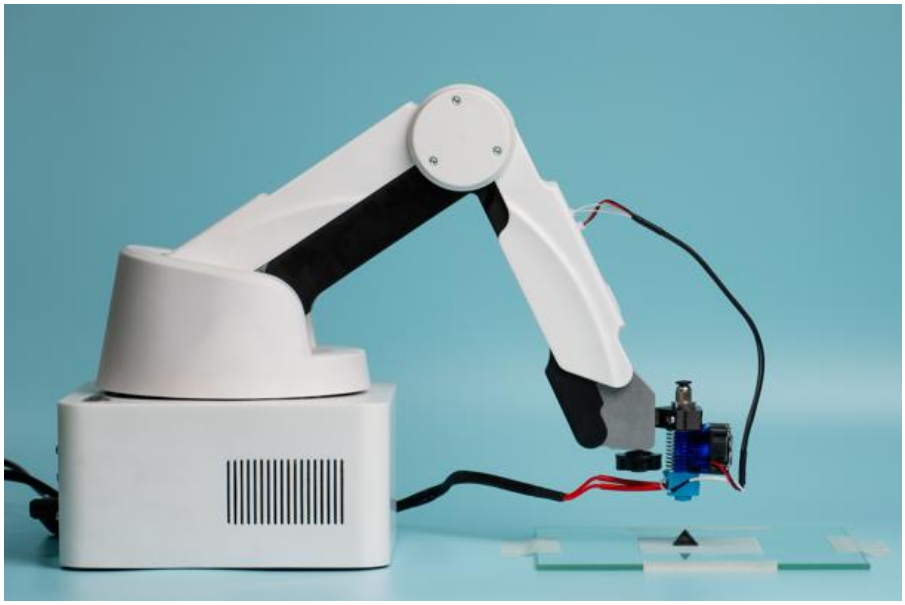


Рисунок 27 – Расположение защитного стекла

4. Включите манипулятор, руководствуясь инструкцией.

Модуль 3D-печати готов к работе.

3.11 Модуль лазерной гравировки

Модуль лазерной гравировки[1] предназначен для нанесения изображений на дерево или картон.

Примечание – Не допускается использовать другие материалы, потому что при работе они могут выделять вредные испарения.

Модуль состоит из лазерной головки с фиксатором инструмента и защитных очков (см. Рисунок 28). Для обеспечения безопасной работы питание модуля управляется через ключ-выключатель.

Рисунок 28 – Модуль лазерной гравировки

Принцип работы: лазерный луч точечно нагревает поверхность, из-за чего материал в этом месте меняет свои свойства и цвет. Благодаря этому на дереве или картоне можно получить изображение.

Установка и подключение модуля лазерной гравировки:

Примечание – Все монтажные работы допускается производить только на обесточенном оборудовании. Отключение проводов на стреле манипулятора допускается

осуществлять только за корпус разъема с помощью пинцета – выдергивание разъемов за провода может привести к их повреждению.

1. Установите лазерную головку в блок инструмента и закрепите винтом.
2. Подключите провод к разъему 4. GP3 на стреле манипулятора (см. Рисунок 29).

Рисунок 29 – Подключение проводов лазерной головки

3. Подключите блок коммутации инструмента к разъему TTL на панели разъемов для подключения внешних устройств.
4. Расположите материал для выжигания на поверхности стола в 10 см от корпуса манипулятора (под стрелой манипулятора).
5. Наденьте защитные очки.

Примечание – Прямой или отраженный луч лазера может вызвать ожоги или слепоту. Требуется использовать защитные очки при работе с модулем лазерной гравировки.

6. Включите манипулятор, руководствуясь инструкцией.

Модуль лазерной гравировки готов к работе.

[1] – При наличии в комплектации M Edu.

3.12 Пульт управления

Пульт управления выглядит как игровой джойстик с кнопками и стиком (см. Рисунок 30).

Рисунок 30 – Пульт управления

Он подключается в разъем USB на панели для подключения внешних устройств.

С помощью этого пульта можно вручную управлять движением манипулятора, используя кнопки и стик. Возможности управления через пульт зависят от версии программного обеспечения, установленного на устройстве.

4 ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ M EDU

4.1 Общие указания

Техническое обслуживание M Edu должно проводиться регулярно для обеспечения его надежной работы и продления срока службы. Рекомендуется проводить техническое обслуживание не реже одного раза в месяц, а также после каждого интенсивного использования.

Основные задачи технического обслуживания:

- проверка работоспособности всех компонентов;
- очистка от пыли и загрязнений;
- обновление программного обеспечения.

4.2 Меры безопасности

Техническое обслуживание требуется проводить в хорошо проветриваемом помещении. Перед началом обслуживания отключите M Edu от источника питания.

4.3 Порядок проведения технического обслуживания изделия

Техническое обслуживание M Edu включает в себя следующие шаги:

1. Проверьте состояние корпуса манипулятора M Edu на наличие трещин и повреждений.
2. Убедитесь, что все соединения надежны, а кабели не имеют изломов или оголенных участков.
3. Используйте мягкую ткань для протирки корпуса манипулятора M Edu и сменных модулей.
4. Удалите пыль и грязь из щелей и труднодоступных мест манипулятора и сменных модулей с помощью сжатого воздуха.
5. Для очистки стола модуля 3D-печати используйте сначала салфетку, смоченную водой, а далее спиртовую салфетку.
6. При обнаружении изношенных или поврежденных деталей обратитесь к производителю.

После завершения всех работ включите манипулятор M Edu и проведите тестирование его функций. Убедитесь, что все системы работают корректно.

5 ВКЛЮЧЕНИЕ M EDU

Для включения M Edu и запуска приложения выполните следующие действия:

1. Поставьте манипулятор на ровную поверхность. Убедитесь, что вокруг него (в радиусе 0,5 метра) нет никаких лишних предметов.
2. Переведите тумблер включения питания манипулятора в положение «0» (выключено).
3. Установите нужный сменный модуль.
4. Подключите манипулятор любым из способов:
    1) Подключение к манипулятору монитора и элементов управления:
        - подключите блок питания к манипулятору и к розетке;
        - подключите монитор к манипулятору с помощью кабеля HDMI;
        - подключите элементы управления (мышь, клавиатура) к манипулятору через разъемы USB.
    2) Подключение к манипулятору ПК через шнур сетевого интерфейса ETHERNET:
        - подключите блок питания к манипулятору и к розетке;
        - подключите ПК к манипулятору с помощью шнура сетевого интерфейса ETHERNET.
5. Переведите тумблер включения питания манипулятора в положение «|» (включено).
6. На мониторе отобразится загрузка Ubuntu.
7. Прозвучит сигнал «Система запущена. Внимание! Начинается процесс калибровки. Будьте осторожны, не касайтесь манипулятора и убедитесь, что посторонние предметы не помешают движению устройства. Калибровка начнется через 5, 4, 3, 2, 1». Запустится калибровка: манипулятор должен сначала поднять стрелу вверх, затем совершить повороты влево-вправо и остановиться в исходном положении (см. Рисунок 31). По окончании калибровки прозвучит сигнал: «Калибровка прошла успешно».

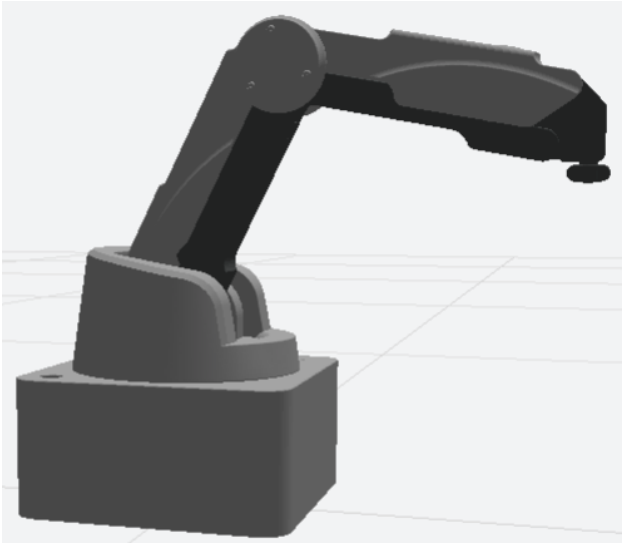


Рисунок 31 – Исходное положение манипулятора

Примечание – Исходное положение манипулятора (или нулевое положение) – это стартовая позиция для манипулятора при его включении. При некорректной работе системы, манипулятор возвращается к стартовой позиции, затем повторяет операцию. Это помогает манипулятору всегда иметь точку отсчета для работы.

8. Загрузка приложения:
   - если манипулятор подключен к монитору и к элементам управления, то на мониторе автоматически отобразится приложение Promobot M Control;
   - если манипулятор подключен к ПК, то на ПК откройте веб-браузер (Chrome, если используете ОС Windows либо Chromium, если используете ОС Linux) и в адресной строке введите ip-адрес манипулятора «10.5.0.2»; отобразится приложение Promobot M Control.

При входе в приложение по умолчанию отобразится раздел ручного управления манипулятором (см. Рисунок 32).

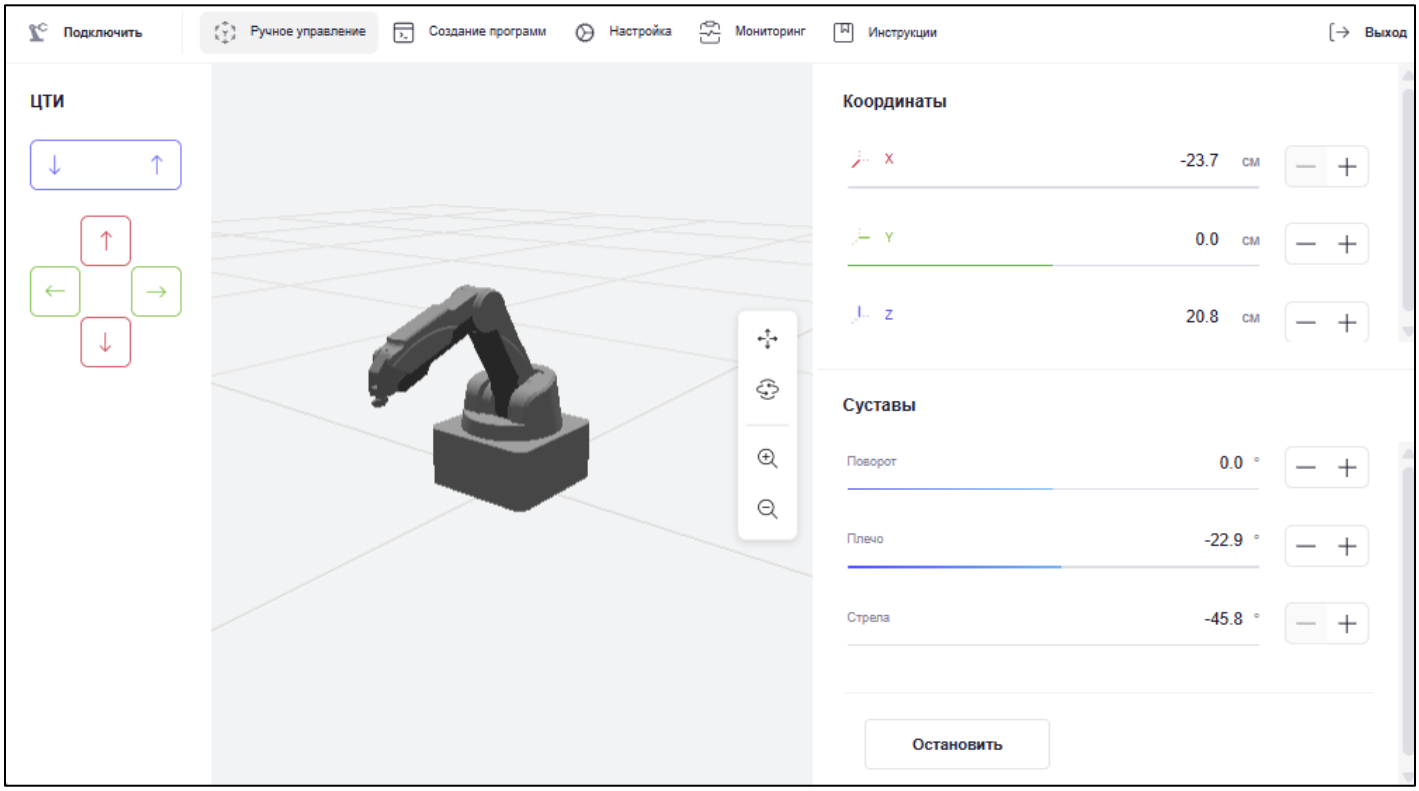


Рисунок 32 – Раздел «Ручное управление»

Примечание – При первом запуске M Edu необходимо установить последнюю версию приложения. Инструкция по обновлению описана в разделе 7.5 «Обновление приложения Promobot M Control» настоящего руководства.

9. Подключите манипулятор в приложении – нажмите кнопку «Подключить» в верхнем левом углу (см. Рисунок 33).

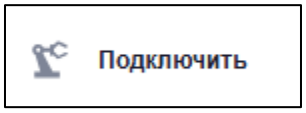


Рисунок 33 – Кнопка «Подключить»

При успешном подключении кнопка изменит цвет на зеленый и отобразится как «Отключить» (см. Рисунок 34).

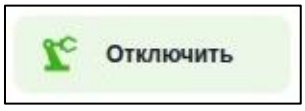


Рисунок 34 – Кнопка «Отключить»

Манипулятор готов к работе.

В случае неуспешного подключения, отобразится уведомление «Манипулятор не найден. Проверьте подключение и попробуйте снова» (см. Рисунок 35).

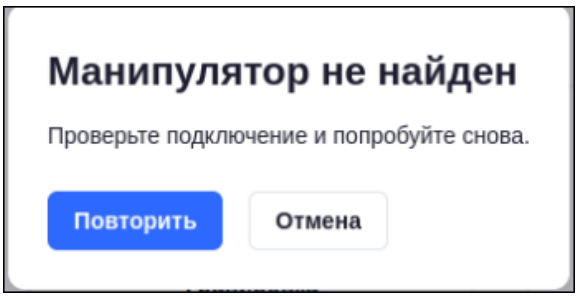


Рисунок 35 – Уведомление

Проверьте подключение и нажмите кнопку «Повторить» или «Отмена». Если манипулятор подключить не удалось, обратитесь в службу технической поддержки предприятия-изготовителя.

6 ВЫКЛЮЧЕНИЕ M EDU

Для выключения M Edu в приложении нажмите кнопку «Выход» в правом верхнем углу. Затем нажмите «Выключить манипулятор».

Отобразится уведомление о выключении (см. Рисунок 36).

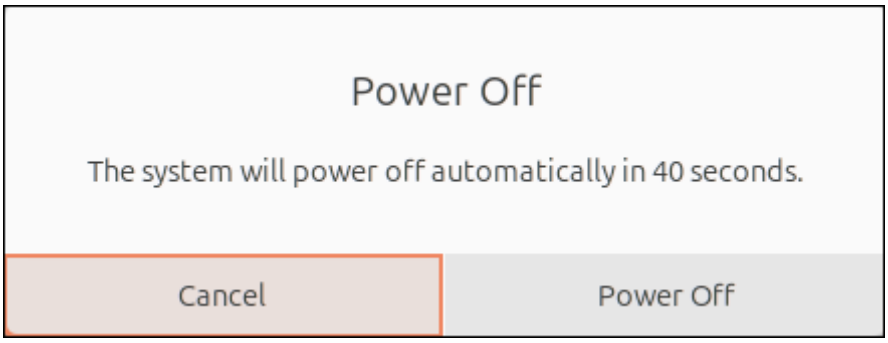


Рисунок 36 – Уведомление о выключении

Прозвучит голосовое сообщение: «Я начал подготовку к выключению. Убедитесь, что стрела манипулятора находится на столе. Сделать это можно через режим FreeDrive. Если стрела поднята, придержите ее во избежание падения».

С помощью кнопки FreeDrive опустите стрелу на стол и дождитесь отключения системы.

Переведите тумблер включения питания манипулятора в положение «0» (выключено).

Для выключения M Edu также можно использовать кнопку включения/выключения на корпусе манипулятора. Далее алгоритм выключения аналогичен предыдущему.

7 ПРИЛОЖЕНИЕ PROMOBOT M CONTROL

Приложение Promobot M Control позволяет:

- придумывать и создавать алгоритмы действий манипулятора;
- запускать воспроизведение действий на манипуляторе и на его виртуальной 3D-модели;
- выстраивать работу манипулятора с насадками;
- изучать основы программирования на языках Blockly и Python, а также запускать на манипуляторе написанный скрипт.

Приложение обладает функционалом обновления, который позволяет пользователям всегда иметь актуальную версию с улучшенной производительностью, новыми функциями и исправленными ошибками.

Примечание – Любой пользователь может зайти на сайт medu.promo-bot.ru, создавать алгоритмы действий манипулятора и запускать их на виртуальной 3D-модели манипулятора.

В верхней части приложения располагается главная панель вкладок, которая содержит: (см. Рисунок 37):

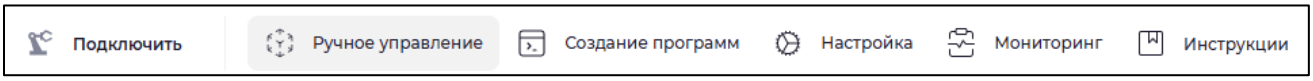


Рисунок 37 – Главная панель вкладок

- Кнопки:
  - «Подключить» – кнопка подключения/отключения манипулятора.
  - «Выход» – кнопка для выключения M Edu.
  - «Инструкции» – кнопка для открытия руководства Promobot M Edu на портале производителя.
- Вкладки:
  - «Ручное управление» – интерфейс для ручного управления манипулятором.
  - «Создание программ» – интерфейс для создания программ с помощью языков программирования Blockly, Python.
  - «Настройка» – интерфейс для настроек работы системы и инструмента.
  - «Мониторинг» – интерфейс для мониторинга работы M Edu.

В нижней части приложения располагается фиксированная нижняя панель, которая отображает наименование устройства и версию ПО (см. Рисунок 38).

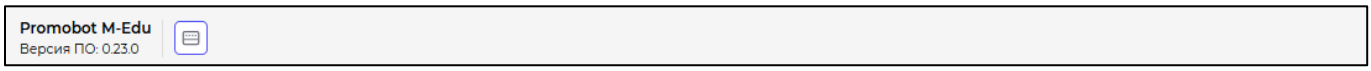


Рисунок 38 – Фиксированная нижняя панель

Также по нажатию кнопки открывается внутренняя клавиатура (см. Рисунок 39).

Рисунок 39 – Внутренняя клавиатура

7.1 Ручное управление

Ручное управление манипулятором (вкладка «Ручное управление») отображается по умолчанию при входе в приложение (см. Рисунок 40).

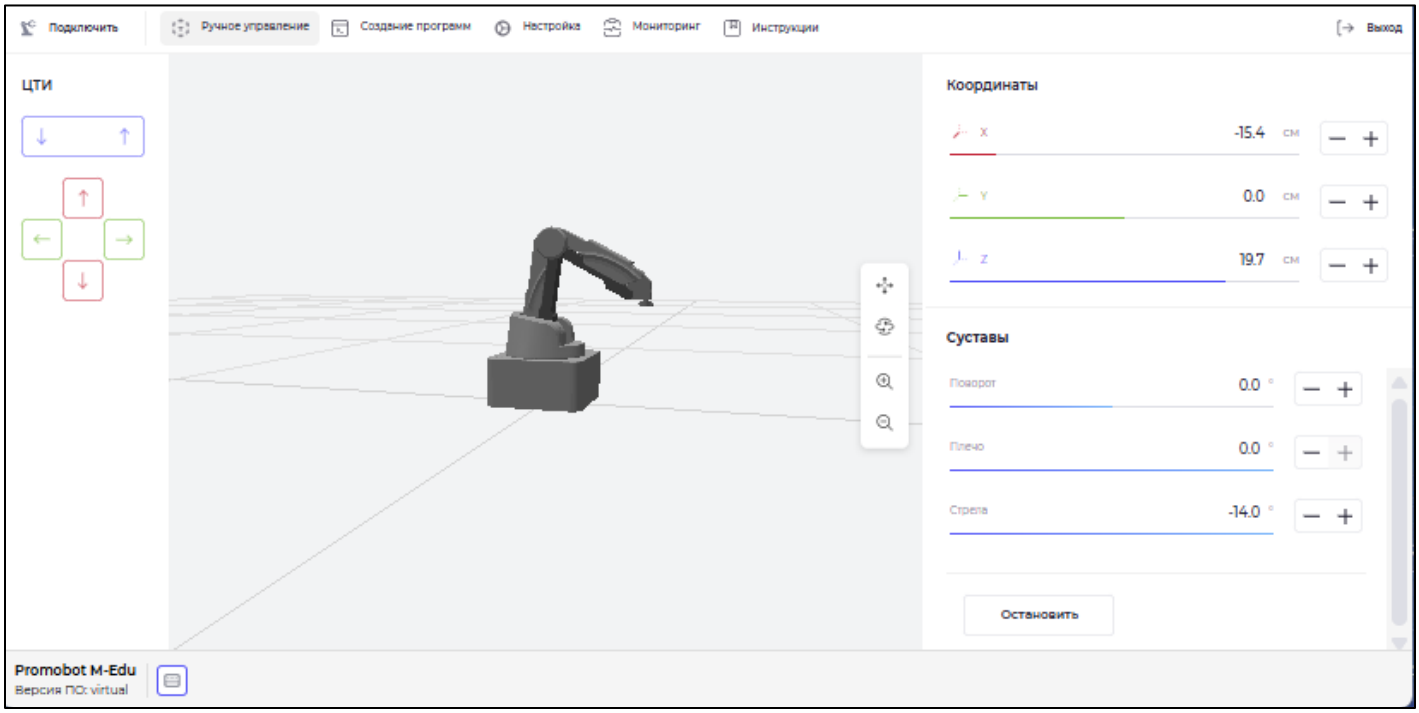


Рисунок 40 – Вкладка «Ручное управление»

Ручное управление доступно при работе с манипулятором без насадки, или с модулем захвата механического или с модулем захвата вакуумного.

7.1.1 Ручное управление «Без насадки»

Перед началом работы проверьте подключение к манипулятору.

Для управления манипулятором без насадки перейдите на вкладку «Настройка», далее «Инструмент», далее в поле «Выберите режим» установите значение «Без насадки» (см. Рисунок 72) и нажмите кнопку «Применить». Режим «Без насадки» включен по умолчанию при входе в приложение.

Перейдите на вкладку «Ручное управление», отобразится интерфейс ручного управления манипулятором в режиме «Без насадки».

Интерфейс ручного управления манипулятором без насадки состоит из следующих частей (см. Рисунок 41):

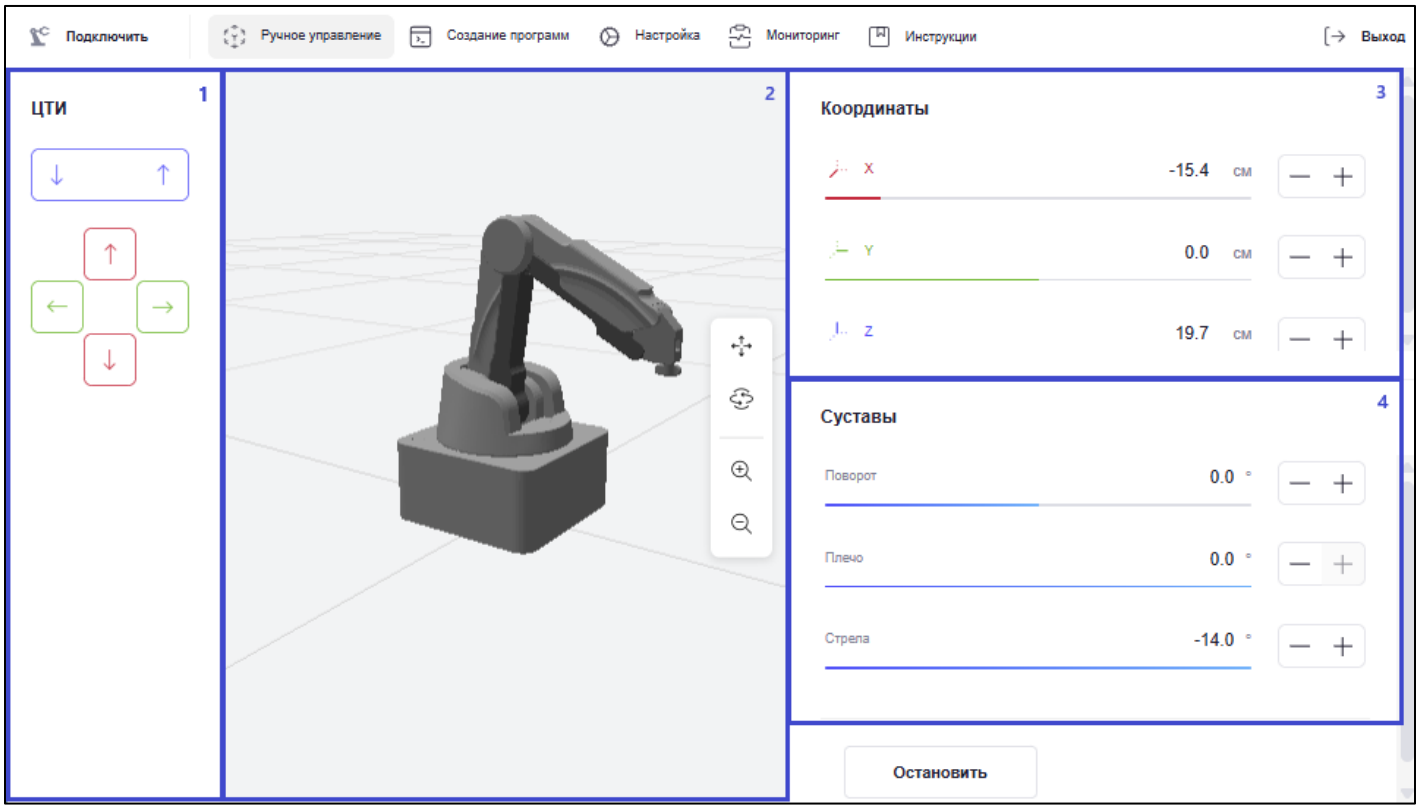


Рисунок 41 –Интерфейс ручного управления манипулятором в режиме «Без насадки»

1. «ЦТИ» – управление центральной точкой инструмента (ЦТИ) с помощью кнопок со стрелками направления.
2. Виртуальное пространство с 3D-моделью манипулятора – воспроизводит движения настоящего манипулятора в виртуальном пространстве. Обзор 3D-модели манипулятора можно регулировать с помощью нажатой левой кнопки мыши либо интерактивными кнопками (см. Рисунок 42).

Рисунок 42 – Интерактивные кнопки

3. «Координаты» – управление манипулятором в трехмерной системе координат. Система координат манипулятора задана тремя взаимно перпендикулярными линиями (см. Рисунок 43). Начало координат – исходное положение манипулятора (оси X, Y) и поверхность, на которую он установлен (ось Z):
   - Ось «X» – горизонтальная ось, простирающаяся вперед и назад.
   - Ось «Y» – горизонтальная ось, простирающаяся налево и направо.
   - Ось «Z» – вертикальная ось, простирающаяся вверх.

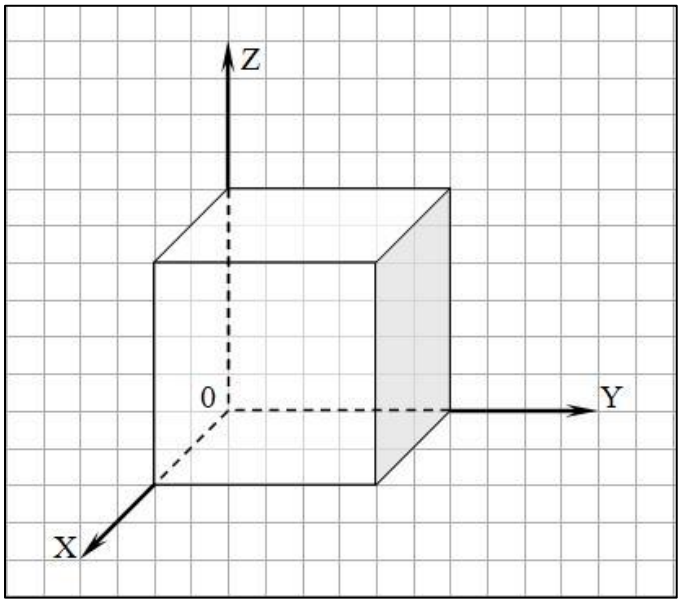


Рисунок 43 – Система координат

4. «Суставы» – управление поворотными узлами манипулятора (см. Рисунок 8):
   - «Поворот» – изменяет угол поворотного узла J1;
   - «Плечо» – изменяет угол поворотного узла J2;
   - «Стрела» – изменяет угол поворотного узла J3.

С помощью кнопки «Остановить» манипулятор можно остановить во время движения.

7.1.2 Ручное управление «Механический захват»

Перед началом работы установите модуль захвата механического (см. раздел 3.8 «Модуль захвата механического»). Проверьте подключение к манипулятору.

Для управления манипулятором с механическим захватом перейдите на вкладку «Настройка», далее «Инструмент», далее в поле «Выберите режим» установите значение «Механический захват» (см. Рисунок 72) и нажмите кнопку «Применить».

Перейдите на вкладку «Ручное управление», отобразится интерфейс ручного управления манипулятором в режиме «Механический захват» (см. Рисунок 44).

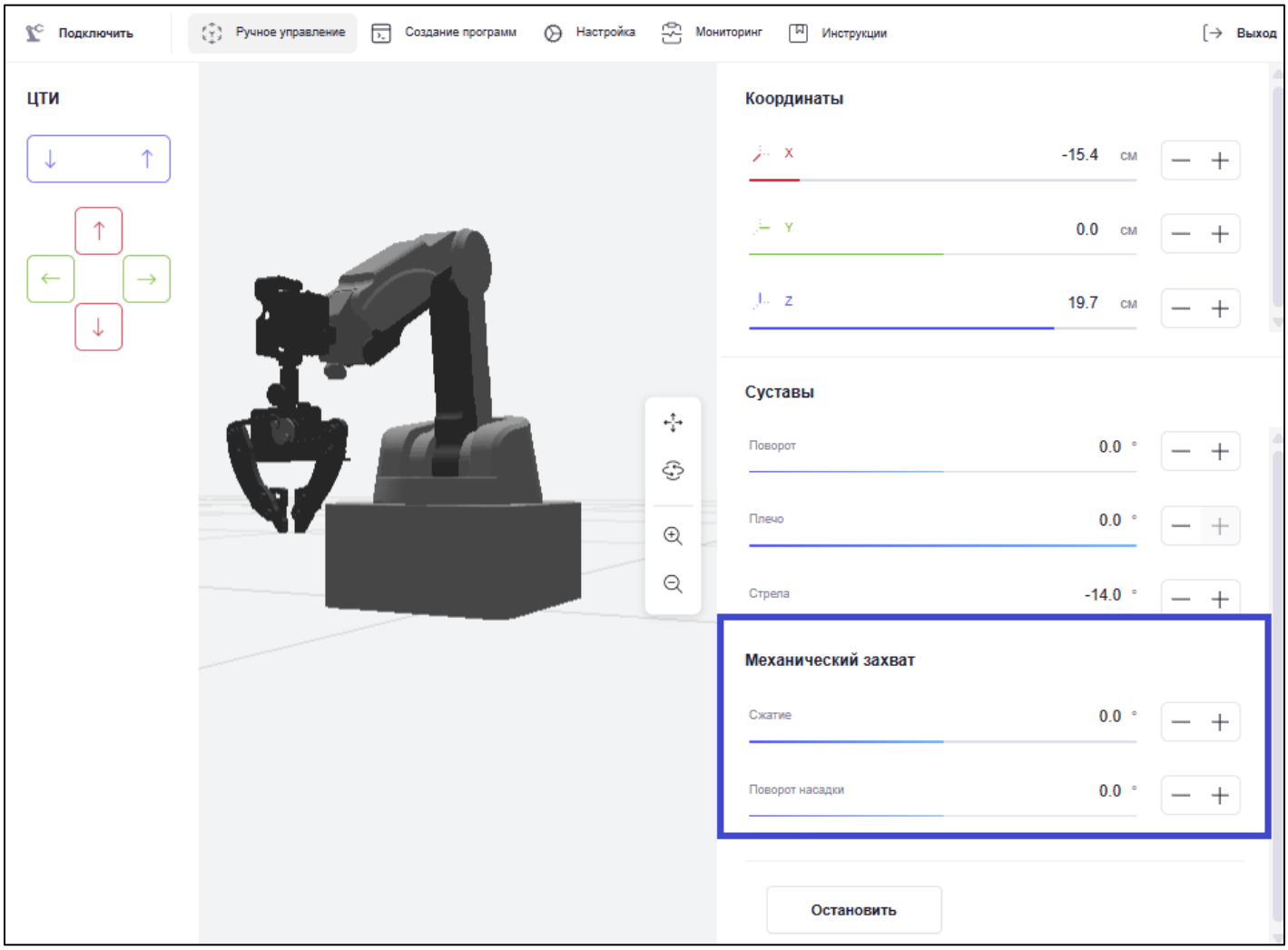


Рисунок 44 – Интерфейс ручного управления манипулятором в режиме «Механический захват»

Интерфейс ручного управления манипулятором с механическим захватом аналогичен интерфейсу ручного управления манипулятором без насадки (см. раздел 7.1.1 «Ручное управление «Без насадки»), при этом добавлены функции управления механическим захватом:

- «Сжатие» – изменяет угол раскрытия когтей;
- «Поворот насадки» – изменяет угол поворотного узла J4 (см. Рисунок 8).

7.1.3 Ручное управление «Вакуумный захват»

Перед началом работы установите модуль захвата вакуумного (см. раздел 3.7 «Модуль захвата вакуумного»). Проверьте подключение к манипулятору.

Для управления манипулятором с вакуумным захватом перейдите на вкладку «Настройка», далее «Инструмент», далее в поле «Выберите режим» установите значение «Вакуумный захват» (см. Рисунок 72) и нажмите кнопку «Применить».

Перейдите на вкладку «Ручное управление», отобразится интерфейс ручного управления манипулятором в режиме «Вакуумный захват» (см. Рисунок 45).

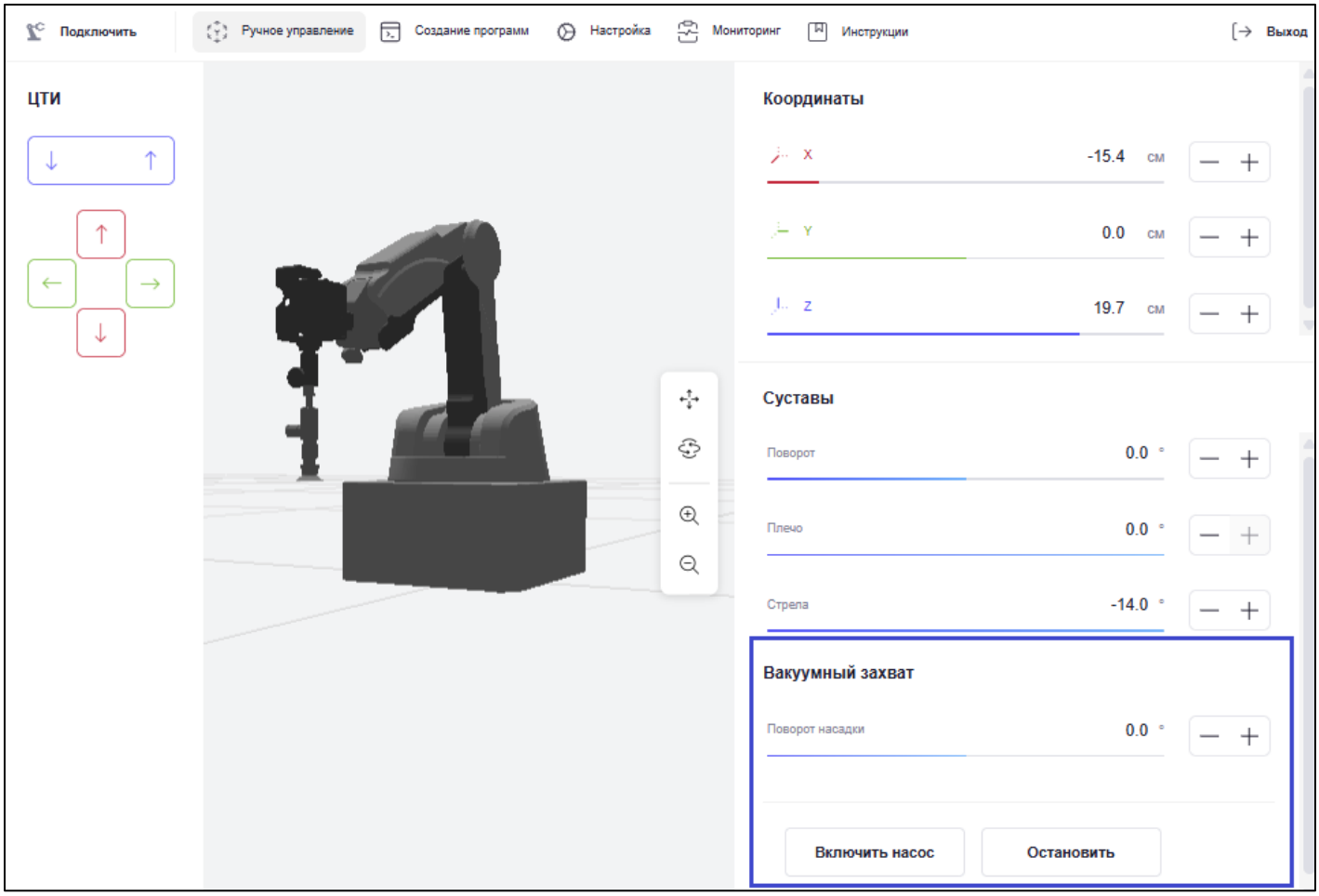


Рисунок 45 – Интерфейс ручного управления манипулятором в режиме «Вакуумный захват»

Интерфейс ручного управления манипулятором с вакуумным захватом аналогичен интерфейсу ручного управления манипулятором без насадки (см. раздел 7.1.1 «Ручное управление «Без насадки»), при этом добавлены функции управления вакуумным захватом:

- «Поворот насадки» – изменяет угол поворотного узла J4 (см. Рисунок 8).
- Кнопка «Включить/выключить насос» – для включения и выключения.

7.2 Создание программ

Приложение Promobot M Control позволяет создавать программы движения манипулятора (вкладка «Создание программ») с помощью языков программирования Blockly, Python.

Интерфейс создания программ имеет верхнее меню (см. Рисунок 46):

Рисунок 46 – Верхнее меню создания программ

«Скачать» – кнопка для скачивания созданного скрипта в текстовом формате.

«Загрузить» – кнопка загрузки готового скрипта формата *.txt.

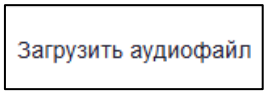
«Выбор шаблона» – кнопка для выбора готового скрипта, кнопка активна в режимах «Механический захват», «Вакуумный захват».

«Удалить все» – кнопка для очистки редактора.

«Blockly» – переход в графический редактор Blockly.

«Python» – переход в редактор кода Python.

«Запуск на манипуляторе» – кнопка запуска скрипта на манипуляторе.



«Загрузить аудиофайл» – кнопка загрузки аудиофайла с компьютера.

В интерфейсе предусмотрено поле «Результат выполнения программы» – отображает результат выполнения программы. Для увеличения поля щелкните по нему левой кнопкой мыши и поле увеличится за счет виртуального пространства (см. Рисунок 47).

Рисунок 47 – Поле «Результат выполнения программы»

7.2.1 Подготовка к созданию программы

Перед тем как начать программировать, убедитесь, что:

- вы подключены к манипулятору (если находитесь на сайте https://medu.promo-bot.ru/ функционал будет ограничен);
- на манипулятор установлена нужная насадка;
- в настройках выбран подходящий режим;

Примечание – Создание программ доступно только в режимах «Без насадки», «Механический захват», «Вакуумный захват». Неправильный выбор режима может привести к ошибкам в движениях манипулятора, а также ограничит выбор в библиотеке блоков.

- выбран язык программирования:
  - Blockly – визуальный язык программирования для начинающих;
  - Python – для тех, кто готов перейти к текстовому коду.

7.2.2 Blockly

Blockly в приложении Promobot M Control представляет собой набор графических блоков (библиотека) определенной формы и графический редактор (см. Рисунок 48). Соединяя одни блоки с другими можно создать программу движений для манипулятора.

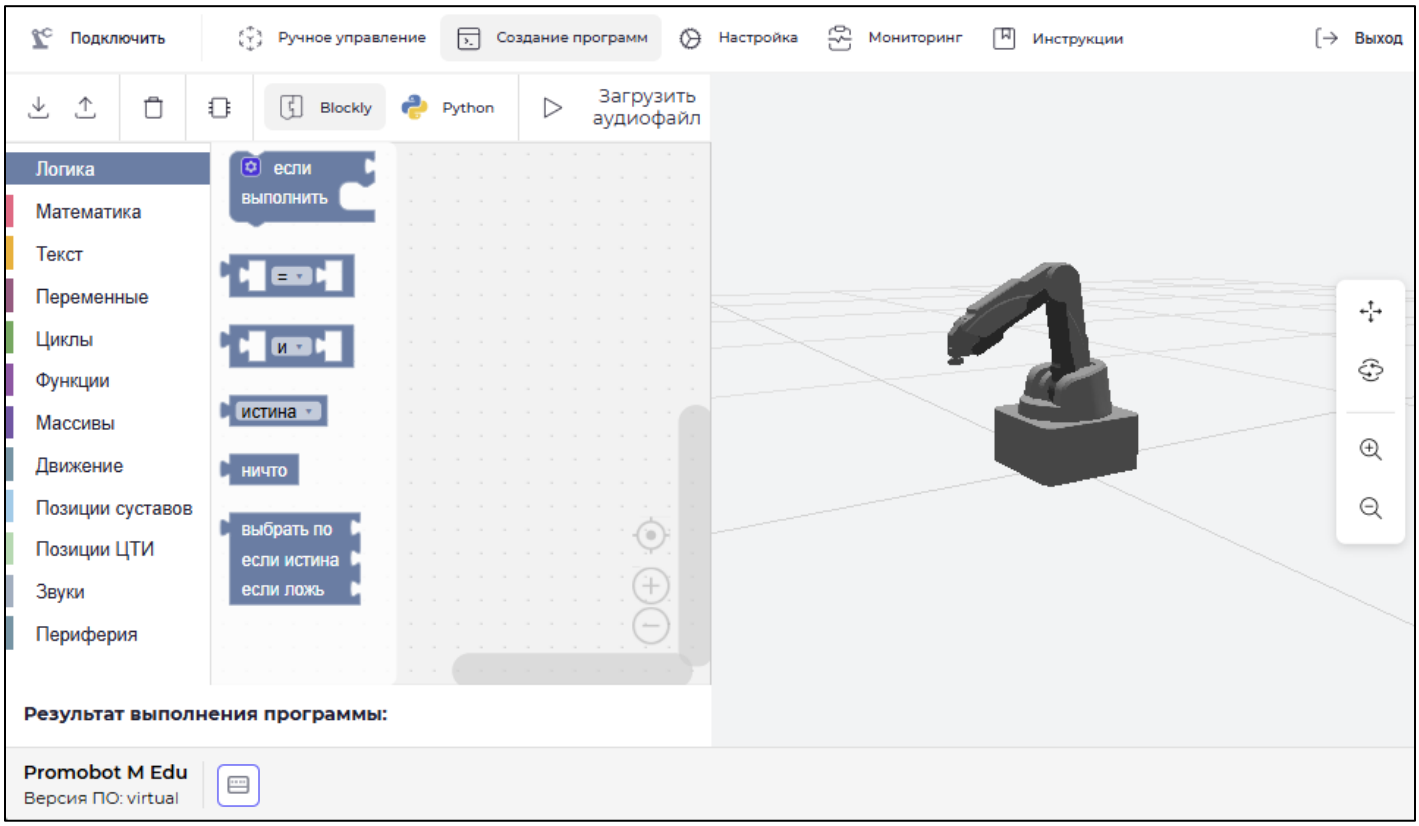


Рисунок 48 – Blockly в приложении Promobot M Control

Для получения краткой справки о блоке наведите на него курсор мыши – отобразится всплывающая подсказка (см. Рисунок 49).

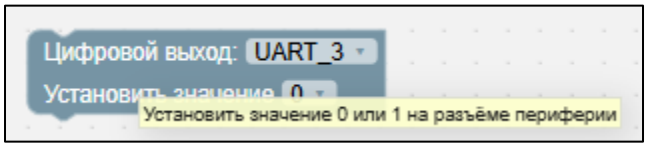


Рисунок 49 – Всплывающая подсказка

По блокам общего функционала доступна справка в контекстном меню блока (см. Рисунок 50), для этого нажмите правой кнопки мыши на блок.

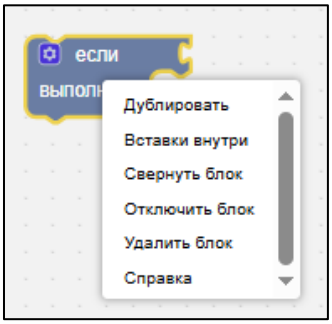


Рисунок 50 – Контекстное меню блока

7.2.2.1 Описание графических блоков (библиотека Blockly)

Библиотека Blockly в приложении Promobot M Control имеет следующие разделы:

1. «Логика» – условия, сравнения (см. Рисунок 51).

Рисунок 51 – Блоки «Логика»

2. «Математика» – вычисления, формулы (см. Рисунок 52).

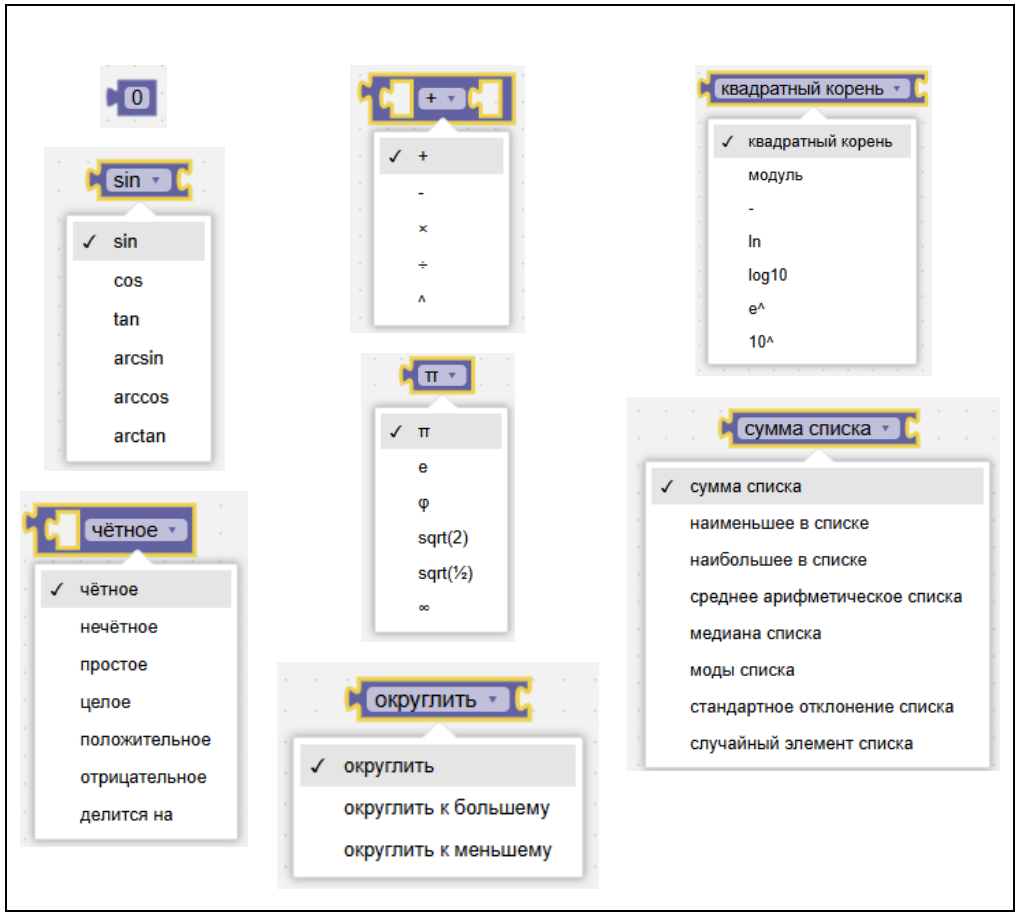


Рисунок 52 – Блоки «Математика»

3. «Текст» – работа с текстом (см. Рисунок 53).

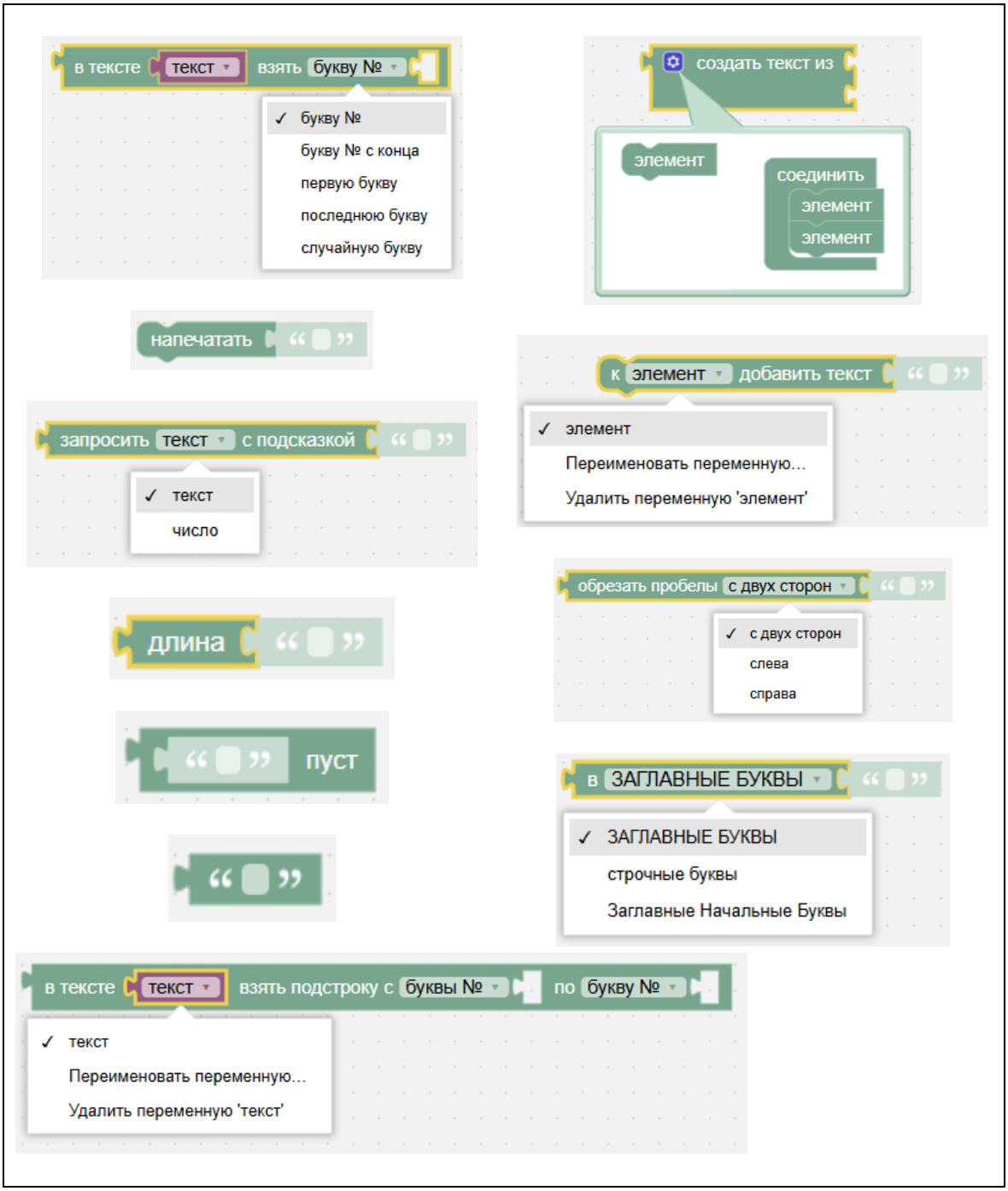


Рисунок 53 – Блоки «Текст»

4. «Переменные» – создание и редактирование переменных (см. Рисунок 54).

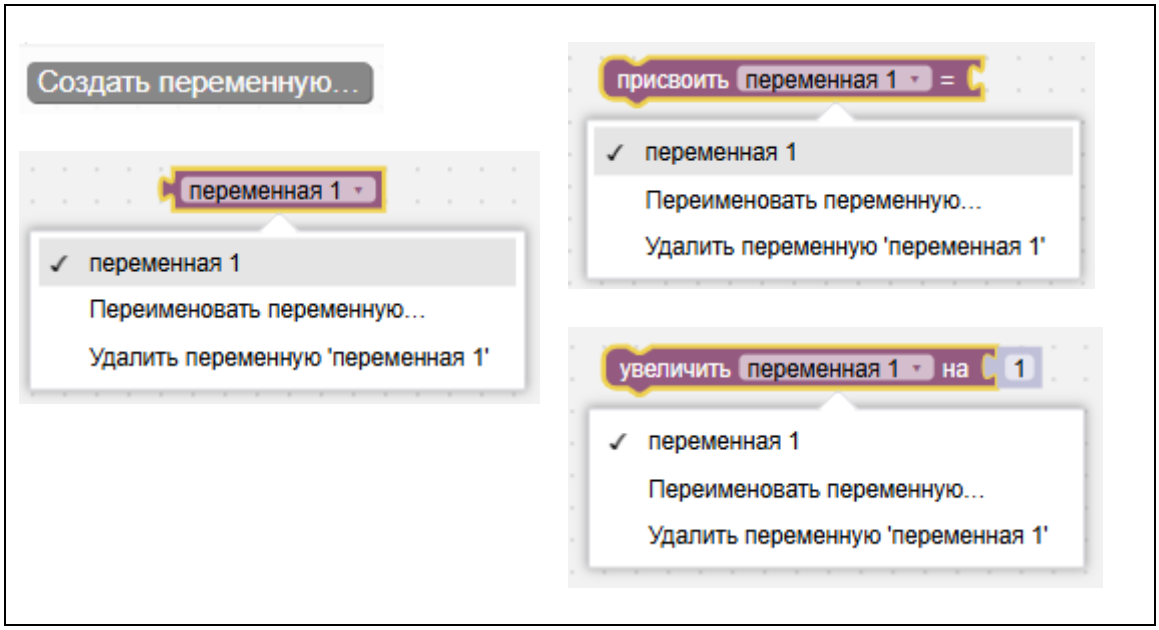


Рисунок 54 – Блоки «Переменные»

«Циклы» – многократное повторение определенных действий (команд) (см. Рисунок 55).

Рисунок 55 – Блоки «Циклы»

5. «Функции» – создание собственных процедур и именованных подпрограмм (см. Рисунок 56).

Рисунок 56 – Блоки «Функции»

6. «Массивы» – работа с элементами списков (см. Рисунок 57).

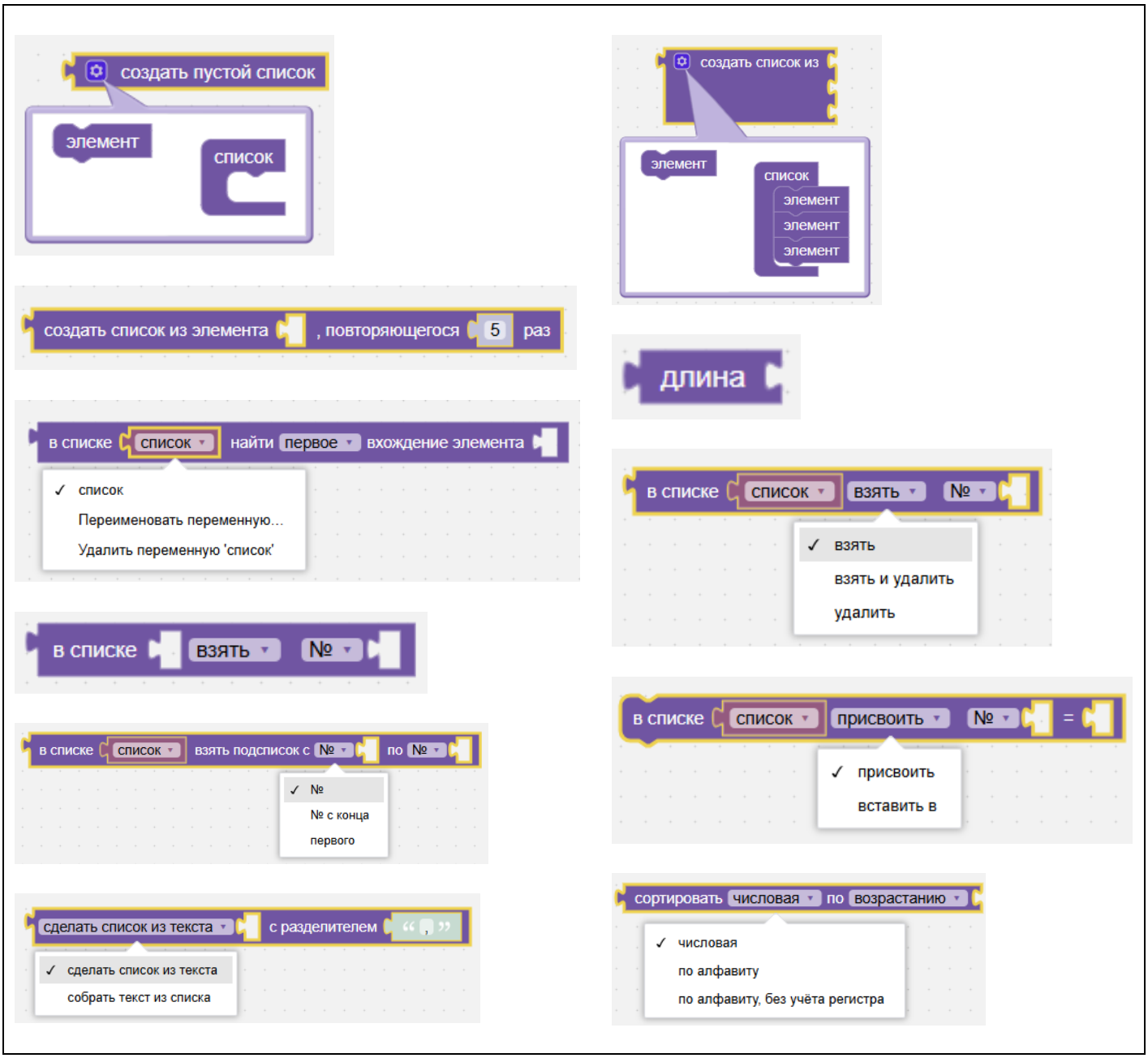
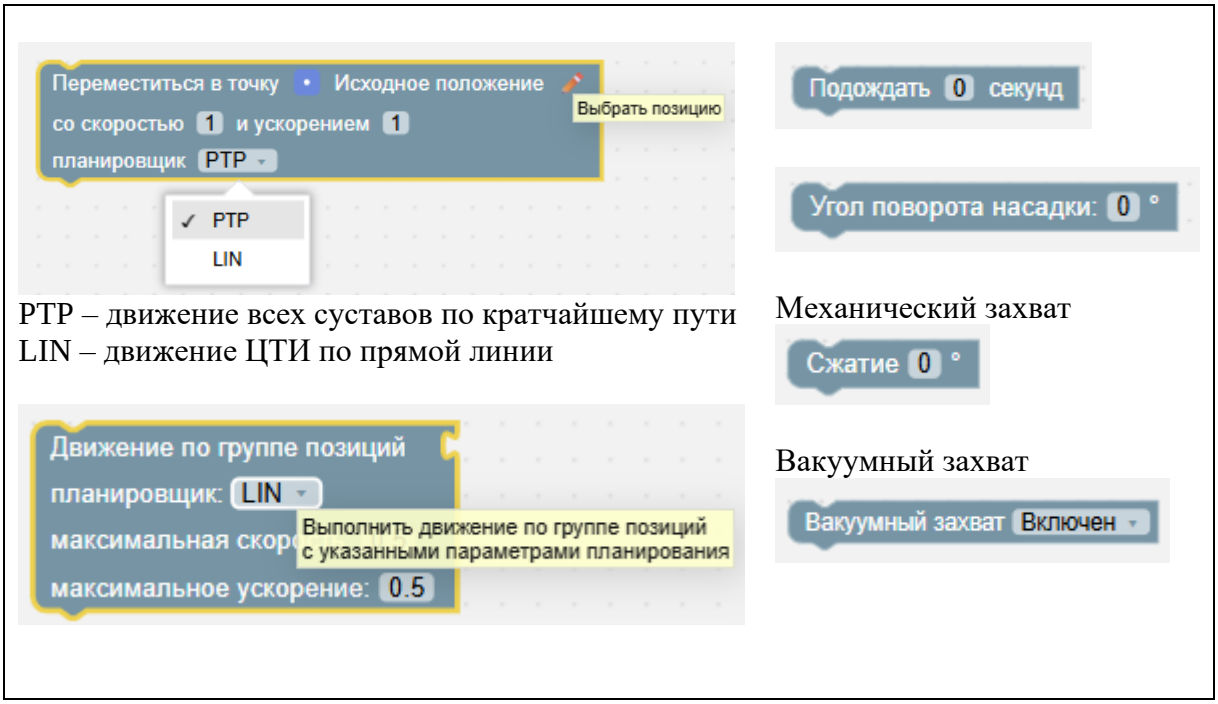


Рисунок 57 – Блоки «Массивы»

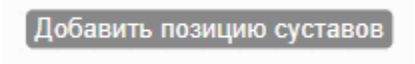
7. «Движение» – команды перемещения и работы с насадками (см. Рисунок 58).



PTP – движение всех суставов по кратчайшему пути
LIN – движение ЦТИ по прямой линии

Рисунок 58 – Блоки «Движение»

8. «Позиции суставов» – сохранение и использование точек положения поворотных узлов манипулятора (J1, J2, J3) (см. 3.3.2 «Башня»). Для добавления и сохранения позиций нажмите «Добавить позицию суставов» (см. Рисунок 59).



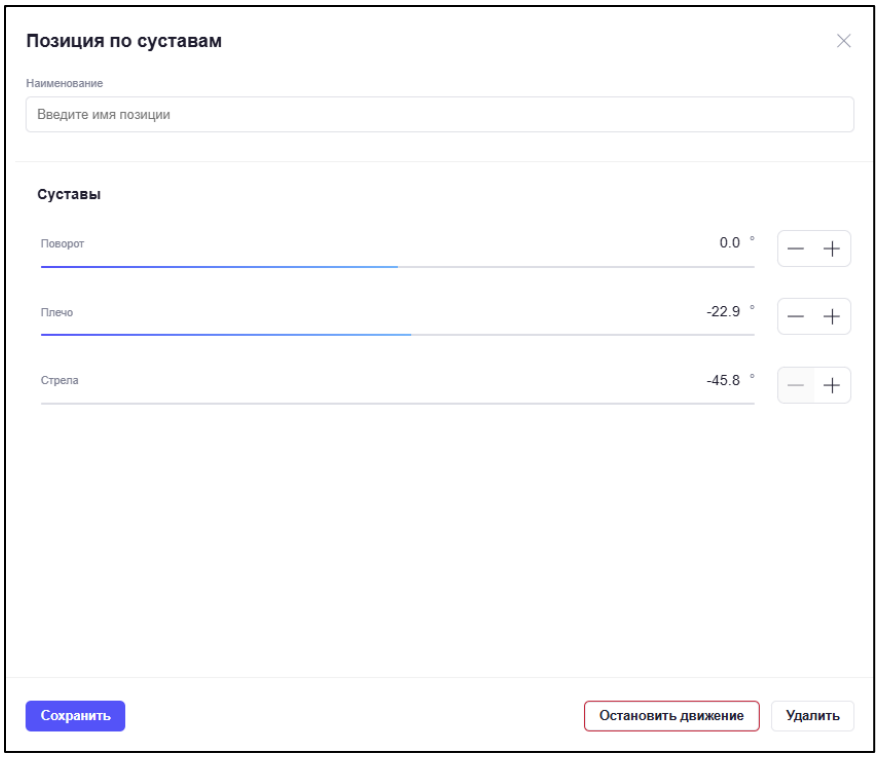


Рисунок 59 – Форма для создания позиции по суставам

9. «Позиции ЦТИ» – сохранение и использование точек положения центральной точки инструмента (ЦТИ). Для добавления и сохранения позиций нажмите «Добавить позицию ЦТИ» (см. Рисунок 60).

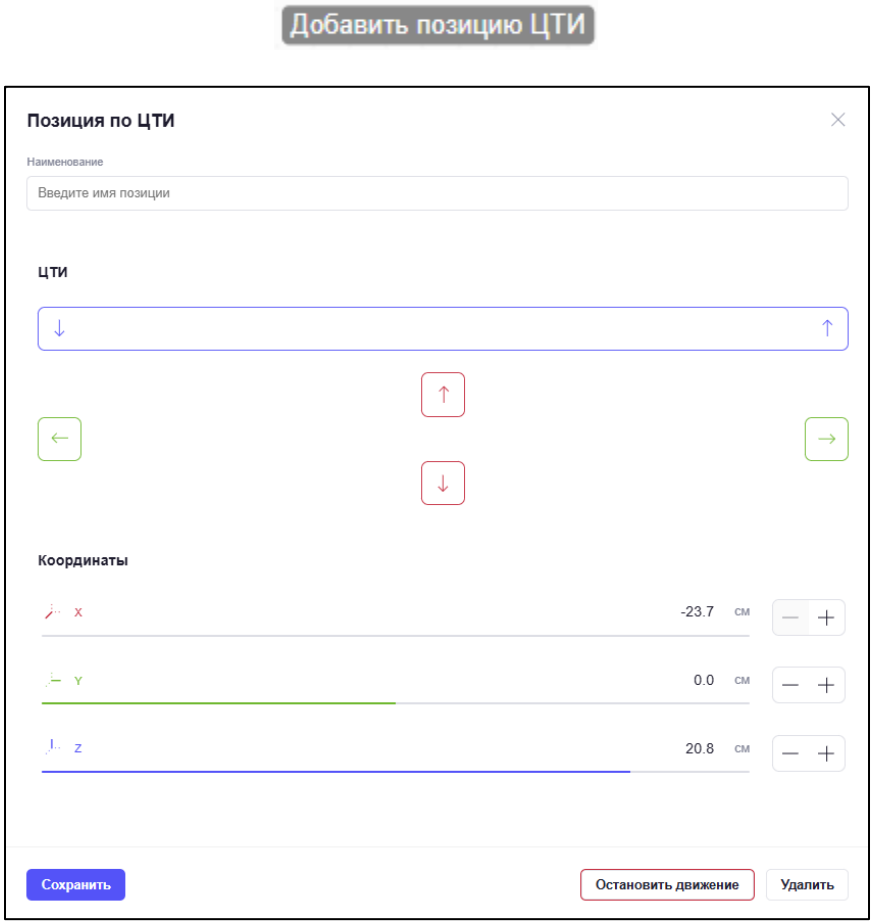


Рисунок 60 – Форма для создания позиции ЦТИ

10. «Звуки» – воспроизведение аудио с возможностью фонового режима (см. Рисунок 61). Предустановлено 4 звуковых файла. Дополнительно можно загрузить файл в формате .wav - нажмите кнопку «Загрузить аудиофайл» и он появится для выбора в блоке.

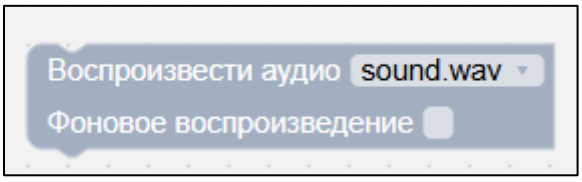


Рисунок 61 – Блок «Звуки»

11. Периферия – управление внешними подключениями (см. Рисунок 62).

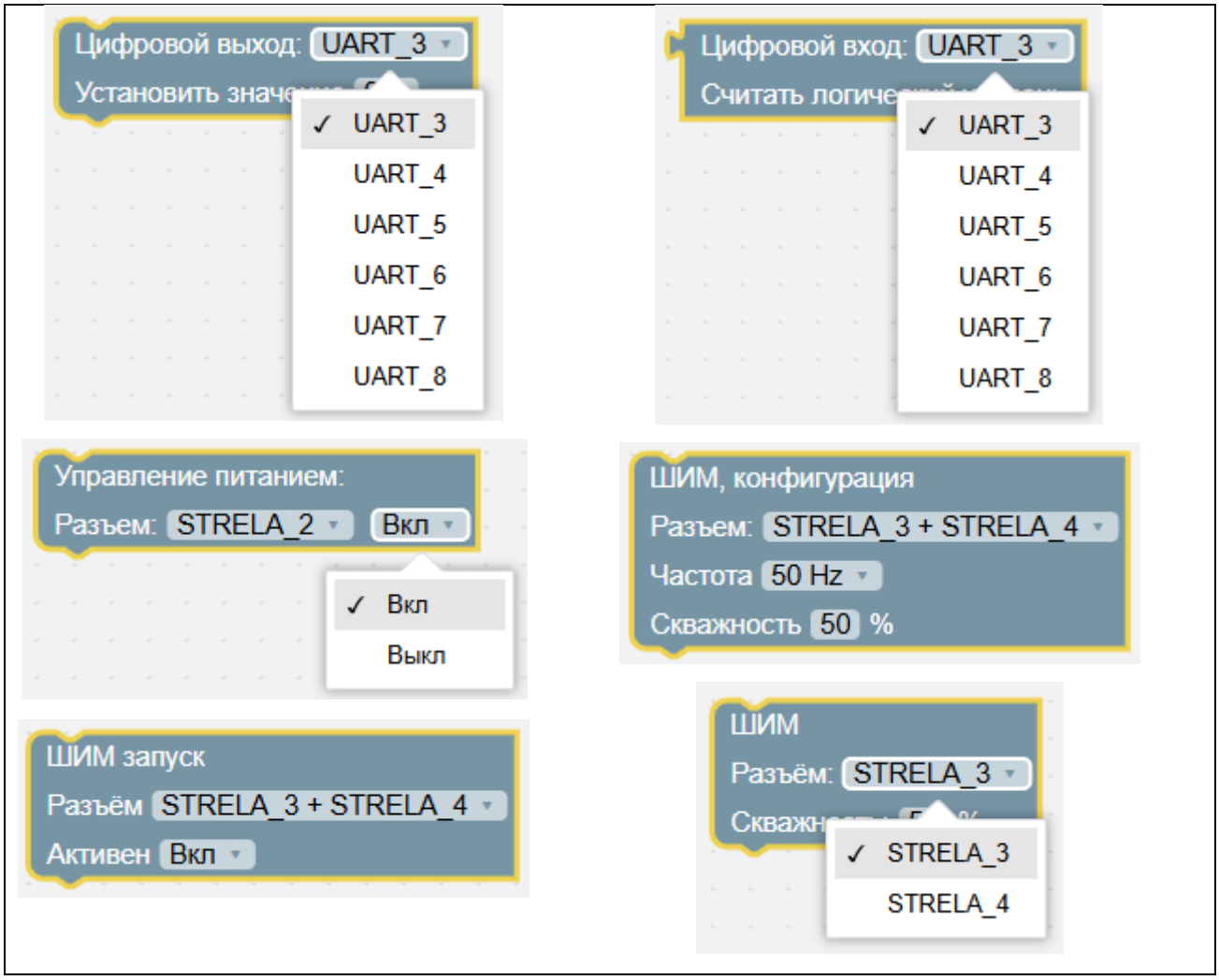


Рисунок 62 – Блоки «Периферия»

Блоки «Периферия» предназначены для взаимодействия с периферийными устройствами, подключаемыми к разъемам M Edu. Блоки позволяют организовать обмен данными по цифровым интерфейсам, управлять питанием внешних модулей, а также генерировать сигналы широтно-импульсной модуляции (ШИМ) для управления сервоприводами и другими аналоговыми устройствами.

Описание блоков:

1. Блоки цифрового ввода/вывода (UART) – данная группа блоков предназначена для настройки приемопередатчиков UART (Universal Asynchronous Receiver-Transmitter) на разъемах M Edu в режиме цифровых линий «вход» или «выход». Это позволяет подключать датчики с цифровым выходом (например, тактовые кнопки, герконы, концевые выключатели) или управлять простыми исполнительными устройствами (например, светодиодами, реле).

Примечание – Интерфейс UART по умолчанию предназначен для асинхронной последовательной передачи данных. Использование его контактов в качестве стандартных цифровых линий требует предварительной настройки в среде разработки.

1) Блок «Цифровой выход: UART_X» (см. Рисунок 63) – конфигурирует выбранный аппаратный UART для работы в режиме цифрового выхода и устанавливает на его выходной линии высокий или низкий логический уровень.

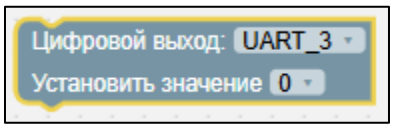


Рисунок 63 – Блок «Цифровой выход: UART_X»

Структура блока:

- Заголовок: «Цифровой выход: UART_X», где X – номер выбранного разъема/UART.
- Параметр «Установить значение» – выпадающий список, в котором пользователь выбирает целевой разъем UART для настройки.
  - Поле: «1» – соответствует логической единице, по умолчанию – сигнал высокого уровня, есть напряжение.
  - Поле: «0» – соответствует логическому нулю, по умолчанию – сигнал низкого уровня, нет напряжения.

Пример использования: для подачи управляющего сигнала на исполнительное устройство, подключенное к разъему «UART_3», необходимо выбрать в блоке значение «UART_3» и в поле параметра выбрать «1». Линия перейдет в активное состояние.

2) Блок «Цифровой вход: UART_X» (см. Рисунок 64) – конфигурирует выбранный аппаратный UART для работы в режиме цифрового входа и считывает текущее логическое состояние на его входной линии.

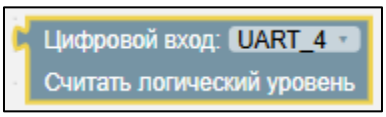


Рисунок 64 – Блок «Цифровой вход: UART_X»

Структура блока:

- Заголовок: «Цифровой вход: UART_X».

- Параметр «Считать логическое» – выпадающий список для выбора разъема UART, состояние которого необходимо проверить.

Пример использования: для ожидания нажатия кнопки, подключенной к разъему «UART_3», в цикле «ждать» или условии «если» используется данный блок.

2. Блок «Управление питанием» (см. Рисунок 65) – отвечает за подачу или отключение электропитания на внешние разъемы M Edu. Используется для энергосбережения или перезагрузки подключенных устройств. Управляет подачей напряжения питания на указанный разъем. Блок доступен только в режиме «Без насадки» (см. Рисунок 72).

Рисунок 65 – Блок «Управление питанием»

Структура блока:

- Заголовок: «Управление питанием».
- Параметр «Разъем» – выпадающий список для выбора разъема питания «STRELA_2».
  - Поле: «Вкл» – напряжение подано на разъем.
  - «Выкл» – напряжение отключено от разъема.

Пример использования: для включения внешнего осветителя, подключенного к силовому разъему «STRELA_2», перед началом видеосъемки, необходимо установить переключатель блока в положение «Вкл».

3. Блоки настройки и запуска ШИМ – широтно-импульсная модуляция (ШИМ) используется для управления сервоприводами, регуляторами оборотов двигателей, яркостью светодиодов. Работа с ШИМ в Blockly разделена на два этапа: конфигурацию и запуск. Блоки доступны только в режиме «Без насадки» (см. Рисунок 72).
   1) Блоки «ШИМ, конфигурация» и «ШИМ» (см. Рисунок 66) – задает параметры ШИМ-сигнала на указанных разъемах. Блок выполняется однократно для настройки генератора.

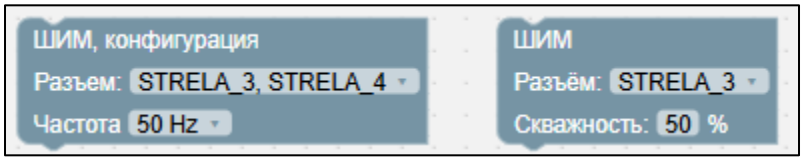


Рисунок 66 – Блоки «ШИМ, конфигурация» и «ШИМ»

Структура блока:

- Заголовок: «ШИМ, конфигурация» либо «ШИМ».
- Параметр «Разъем» – выпадающий список для выбора разъема «STRELA_3 + STRELA_4», что может означать использование двух линий на одном разъеме или выбор конкретного канала.
- Параметр «Частота» в блоке «ШИМ, конфигурация» – поле ввода или выбора частоты сигнала в Герцах (Гц). От частоты зависит период следования импульсов. Для стандартных сервоприводов обычно используется частота 50 Гц.
- Параметр «Скважность» в блоке «ШИМ» – поле ввода или выбора скважности импульса в процентах (%). Скважность определяет долю времени, в течение которого сигнал находится в состоянии «Вкл» (высокий уровень) от общего периода. Для сервоприводов изменение скважности от 5% до 10% (в зависимости от модели) изменяет угол поворота вала.

  Пример использования: для подготовки к управлению сервоприводом, подключенным к разъему «STRELA_3 + STRELA_4», устанавливаются стандартные параметры: частота «50 Гц», а начальная скважность «50 %», что соответствует центральному положению вала большинства сервомашинок.

2) Блок «ШИМ питание (5В)» (см. Рисунок 67) – активирует или деактивирует генерацию ШИМ-сигнала на предварительно сконфигурированном канале.

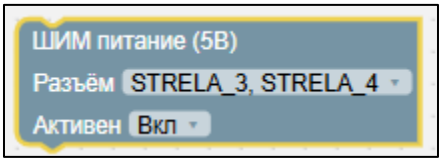


Рисунок 67 – Блок «ШИМ питание (5В)»

Структура блока:

- Заголовок: «ШИМ питание (5В)».

- Параметр «Разъем» – выпадающий список для выбора разъема (должен совпадать с разъемом, указанным в блоке конфигурации).
- Поле «Активен» – переключатель состояния:
  - «Вкл» – генерация ШИМ-сигнала с заданными параметрами начинается.
  - «Выкл» – генерация ШИМ-сигнала прекращается, напряжение на выходе устанавливается в неактивное состояние (обычно 0 В).

Пример использования: после выполнения блока конфигурации для разъема «STRELA_3 + STRELA_4», данный блок с параметром «Вкл» запускает подачу сигнала на сервопривод, заставляя его повернуться в положение, соответствующее заданной скважности. Для остановки удержания сервопривода (снятия управляющего сигнала) используется состояние «Выкл».

7.2.3 Python

Python в приложении Promobot M Control представляет собой редактор кода Python для написания, редактирования и сохранения кода, оптимизированный под синтаксис языка Python (см. Рисунок 68).

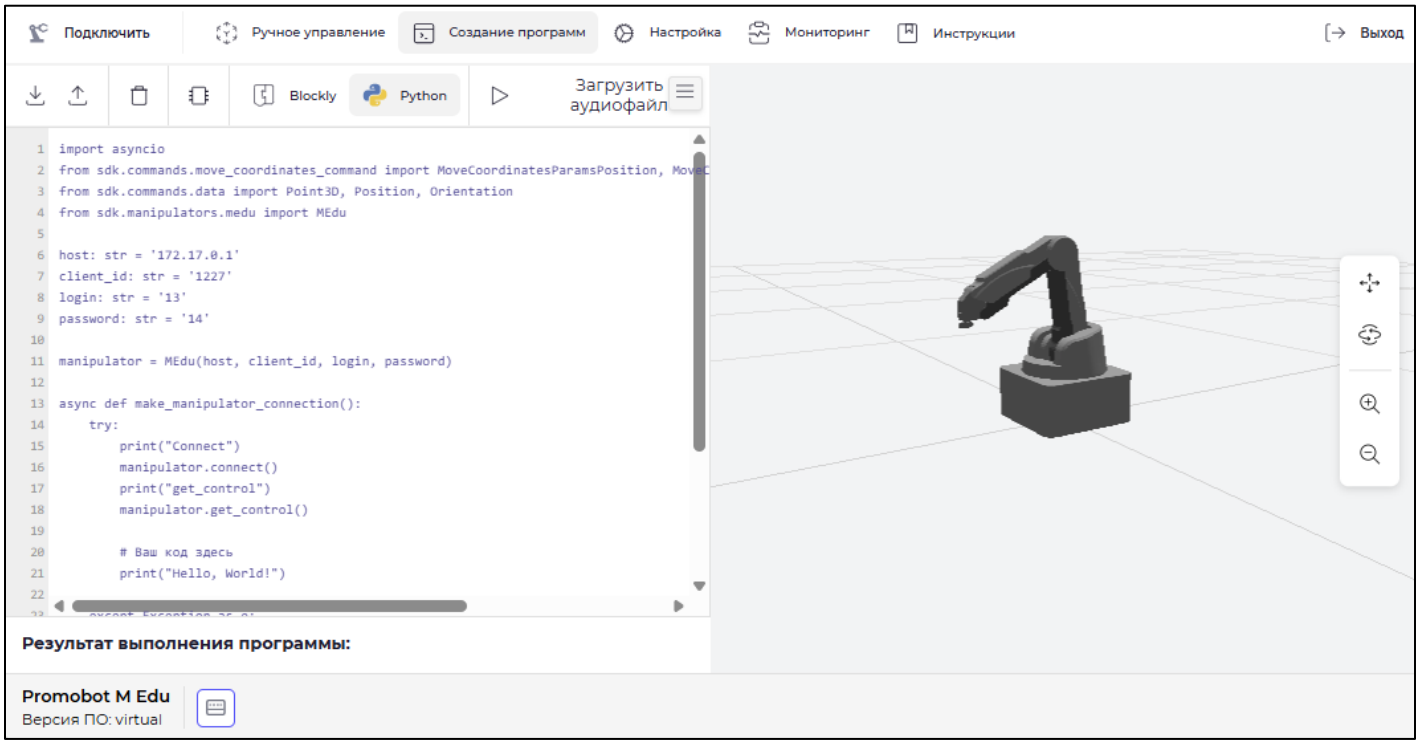


Рисунок 68 – Python в приложении Promobot M Control

Если ранее в Blockly была создана программа, то ее код может автоматически генерироваться в Python. Далее можно редактировать существующие строки.

При написании собственного кода используйте библиотеку SDK, описанную в разделе 8 «PROMOBOT M CONTROL SDK».

При загрузке готового файла Python учтите, что программный код в файле должен быть написан по правилам PEP8, сам файл должен иметь типы *.py или *json.

В соответствии со структурой *json формат сохранения программы такой – в JSON-объекте присутствует единственное поле "program": "/* текст программы */", содержащее всю программу в типе данных "строка". Каждая новая строка кода отделяется от предыдущей при помощи символа "\n" (без использования переноса строки).

Пример программного кода в формате *json:

{

"program": "import asyncio\nfrom sdk.commands.move_coordinates_command import MoveCoordinatesParamsPosition, MoveCoordinatesParamsOrientation\nfrom sdk.manipulators.medu import medu\n\nhost: str = '172.17.0.1'\nclient_id: str = '1227'\nlogin: str = '13'\npassword: str = '14'\n\nmanipulator = medu(host, client_id, login, password)\n\nasync def make_manipulator_connection():\n try:\n print(\"Connect\")\n manipulator.connect()\n print(\"get_control\")\n manipulator.get_control()\n \n # Ваш код здесь\n print(\"Hello, World!\")\n \n except Exception as e:\n print('Возникла ошибка')\n print(e)\n\nloop = asyncio.new_event_loop()\nasyncio.set_event_loop(loop)\nasyncio.get_event_loop().run_until_complete(make_manipulator_connection())\n"

}

При запуске программы есть возможность подсвечивать выполняемую строку. Функцию можно включить/отключить с помощью слайдера (см. Рисунок 69).

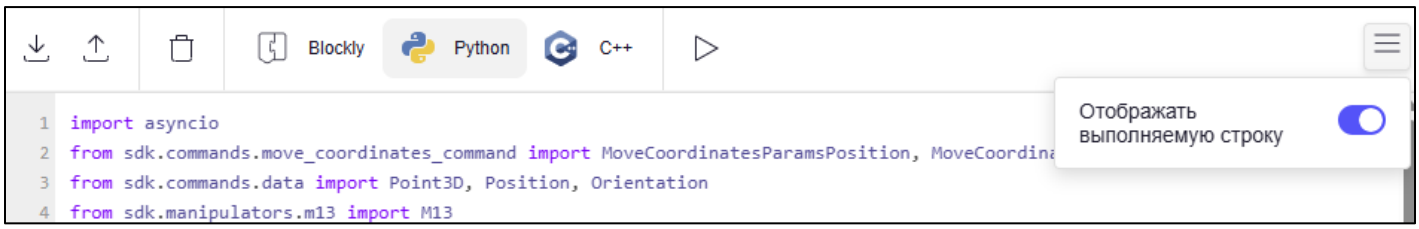


Рисунок 69 – Слайдер «Отображать выполняемую строку»

7.3 Настройка

Интерфейс для настройки (вкладка «Настройка») имеет два раздела (см. Рисунок 70):

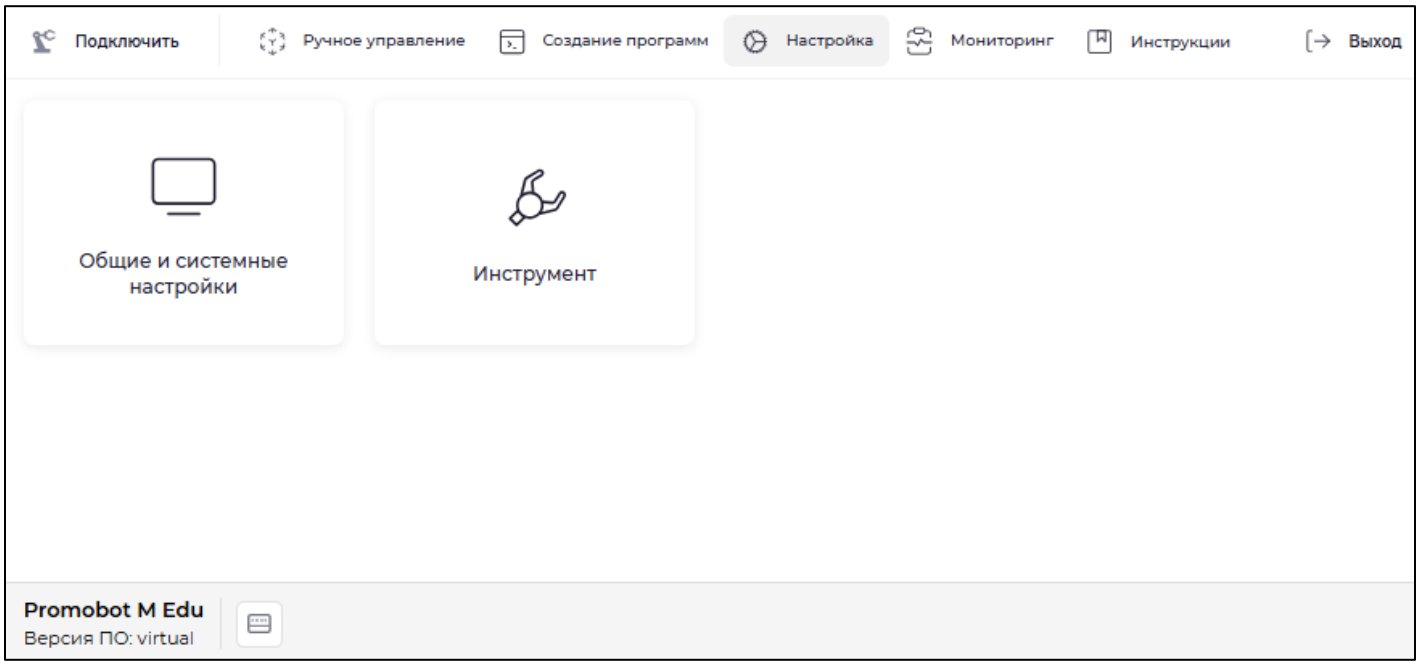


Рисунок 70 – Интерфейс для настройки

1. Раздел «Общие и системные настройки» (см. Рисунок 71).

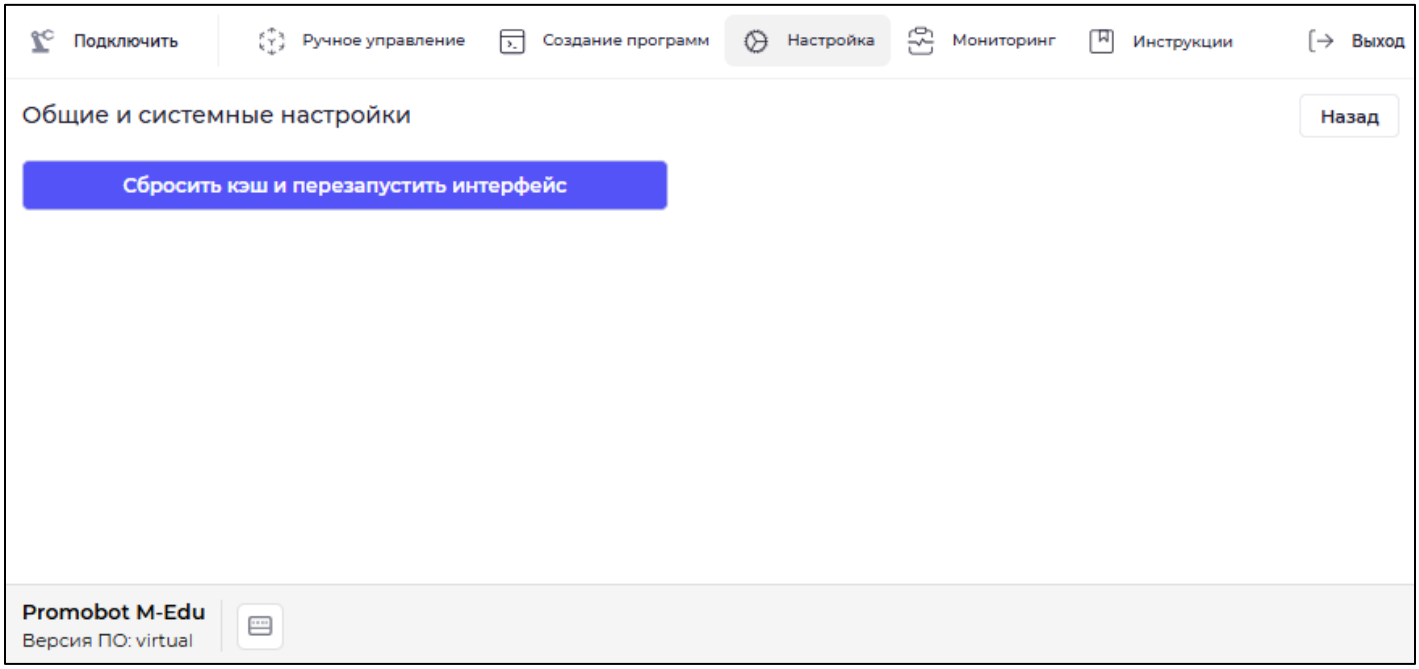


Рисунок 71 – Раздел «Общие и системные настройки»

В разделе есть кнопка «Сбросить кэш и перезапустить интерфейс», которая может быть использована для освобождения памяти и исправления ошибок, возникающих при работе с приложением.

Для возврата к другим настройкам нажмите кнопку «Назад».

2. Раздел «Инструмент» (см. Рисунок 72).

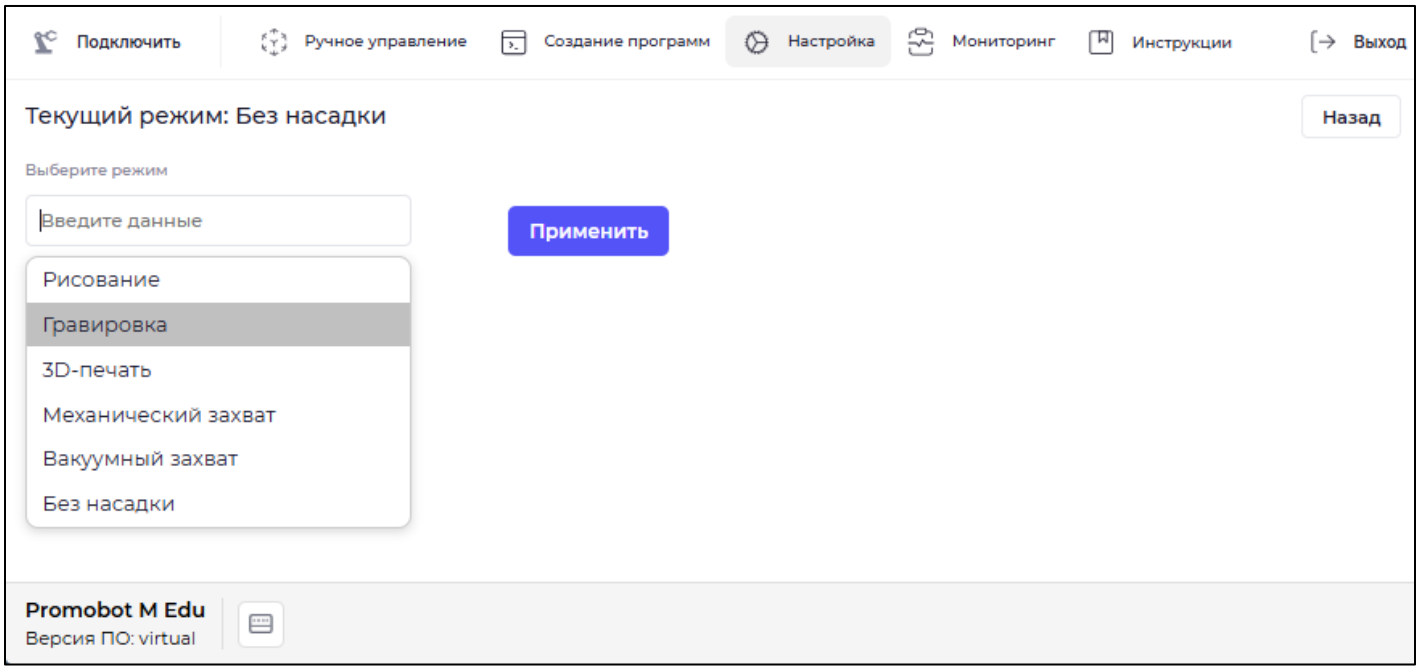


Рисунок 72 – Раздел «Инструмент»

Раздел предназначен для настройки режима работы с инструментом (насадкой). Для настройки в поле «Выберите режим» установите значение в соответствии с установленной насадкой и нажмите кнопку «Применить».

Для возврата к другим настройкам нажмите кнопку «Назад».

7.3.1 Рисование

Перед началом работы установите модуль захвата пишущего инструмента (см. раздел 3.9 «Модуль захвата пишущего инструмента»). Проверьте подключение к манипулятору.

Для управления в режиме «Рисование» нажмите кнопку «Настройка» на главной панели вкладок, в поле «Выберите режим» установите значение «Рисование» (см. Рисунок 72) и нажмите кнопку «Применить».

Нажмите кнопку «Рисование», автоматически откроется библиотека с доступными изображениями (см. Рисунок 73).

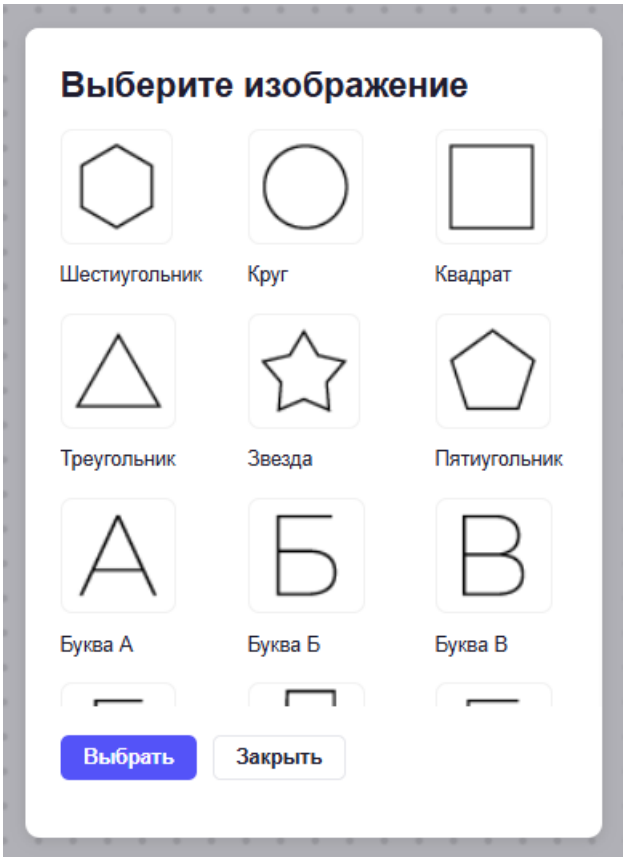


Рисунок 73 – Библиотека рисунков

Выберите изображение и нажмите кнопку «Выбрать» либо закройте форму. Изображение отобразится в области рисования (см. Рисунок 74).

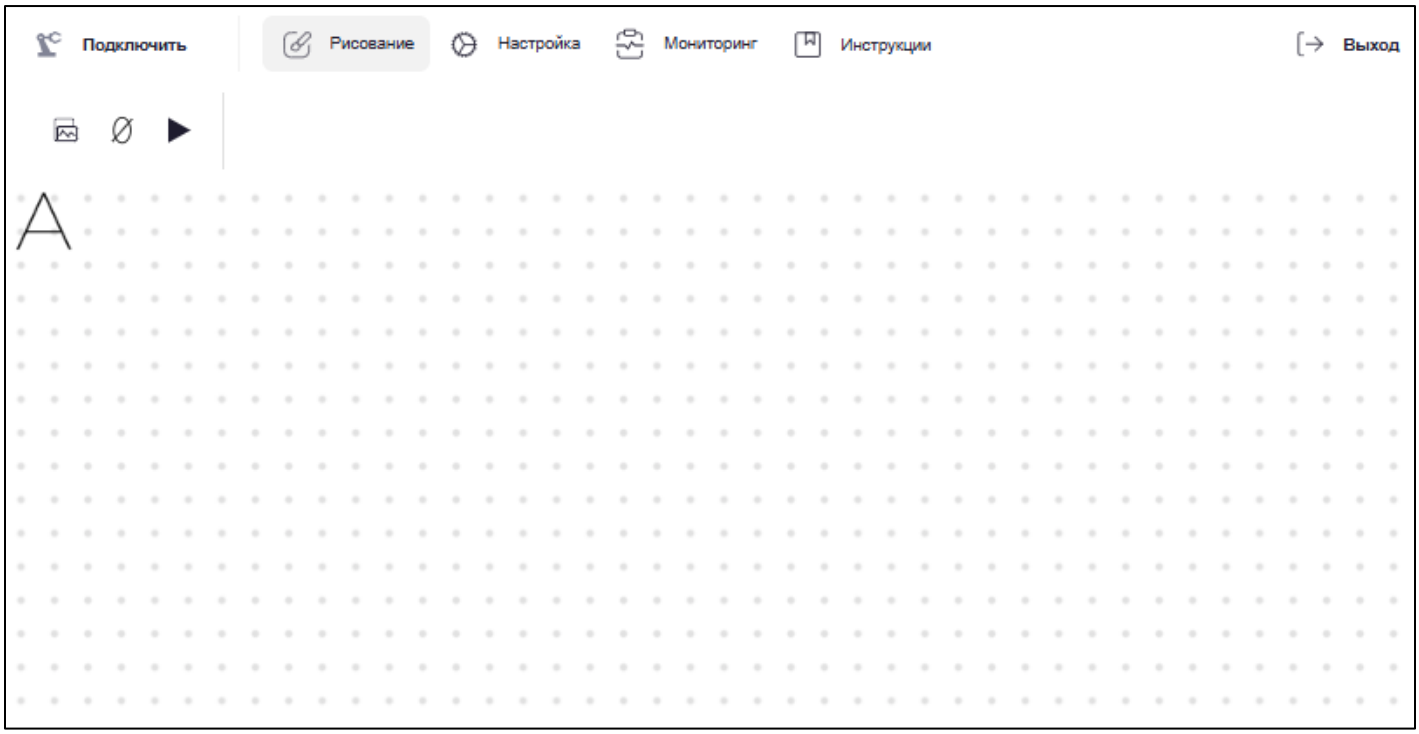


Рисунок 74 – Раздел «Рисование»

Перед запуском задайте стартовую позицию манипулятора:

1. Нажмите кнопку  . Отобразится форма «Задание нуля» (Рисунок 75).

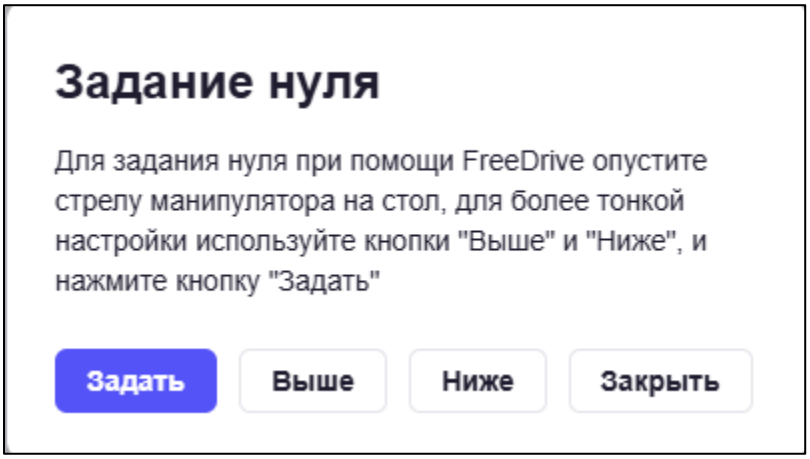


Рисунок 75 – Форма «Задание нуля»

2. Следуя инструкции, определите начальную точку рисования и нажмите «Задать».

Для запуска рисования нажмите кнопку  .

Для остановки рисования нажмите кнопку  .

Чтобы заново выбрать рисунок, нажмите кнопку  .

7.3.2 Гравировка

Перед началом работы установите модуль лазерной гравировки (см. раздел 3.11 «Модуль лазерной гравировки»). Проверьте подключение к манипулятору.

Для управления в режиме «Гравировка» нажмите кнопку «Настройка» на главной панели вкладок, в поле «Выберите режим» установите значение «Гравировка» (см. Рисунок 72) и нажмите кнопку «Применить».

Нажмите кнопку «Гравировка», автоматически откроется сообщение о технике безопасности при работе с лазером (см. Рисунок 76).

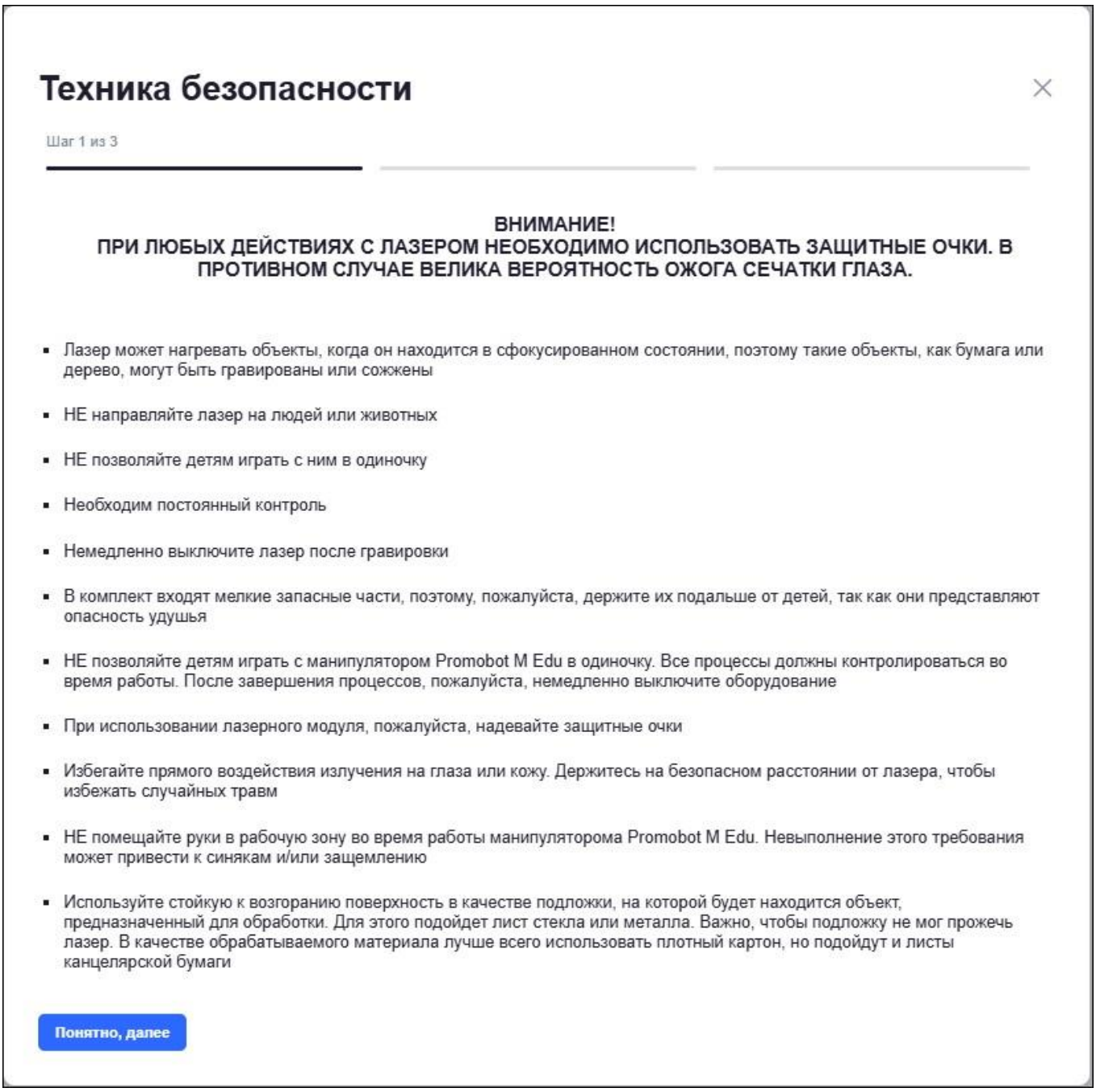


Рисунок 76 – Техника безопасности при работе с лазером

Пройдите по шагам ознакомления, отобразится раздел гравировки (см. Рисунок 77).

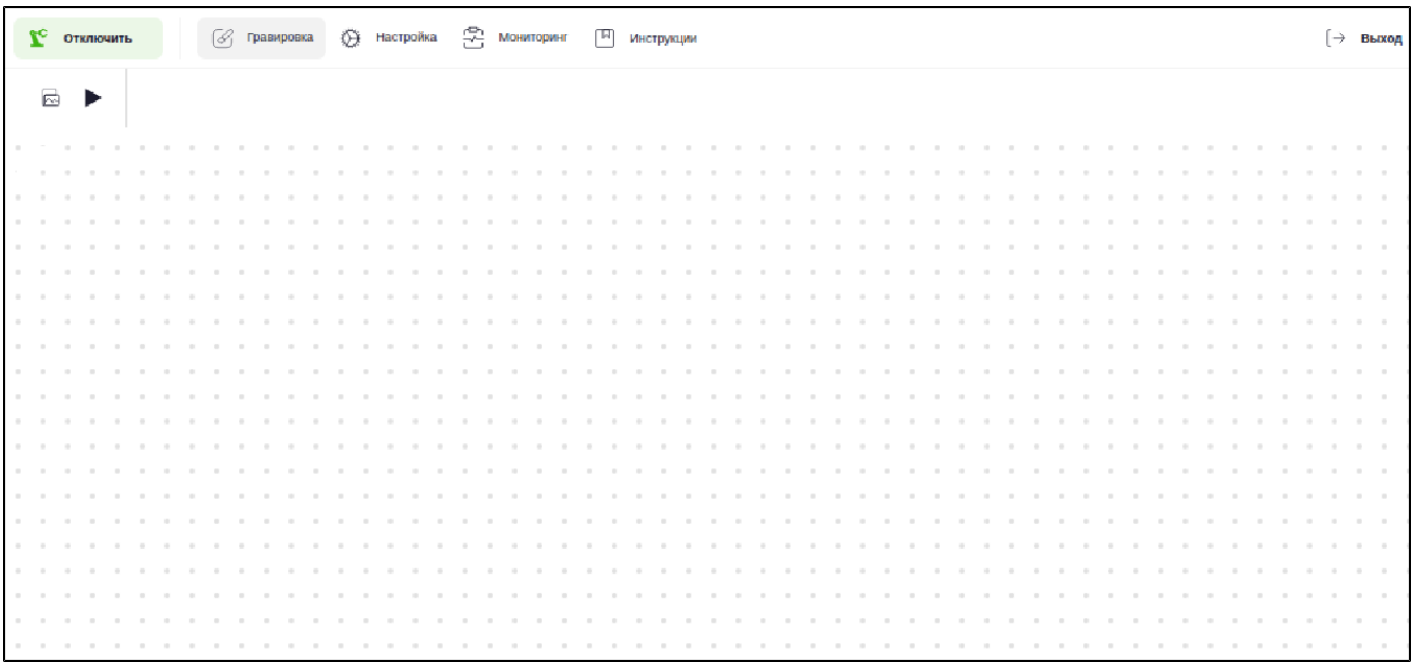


Рисунок 77 – Раздел «Гравировка»

Манипулятор выжигает изображения только из библиотеки. Чтобы выбрать изображение нажмите кнопку  . Отобразится библиотека изображений (см. Рисунок 78).

Рисунок 78 – Библиотека рисунков

Выберите изображение и нажмите кнопку «Выбрать». Изображение отобразится в области гравировки.

Для запуска гравировки нажмите кнопку  .

Для остановки гравировки нажмите кнопку  .

Чтобы заново выбрать изображение нажмите кнопку  .

7.3.3 3D-печать

Перед началом работы установите модуль 3D-печати (см. раздел 3.10 «Модуль 3D-печати»). Проверьте подключение к манипулятору.

Для управления в режиме «3D-печать» нажмите кнопку «Настройка» на главной панели вкладок, в поле «Выберите режим» установите значение «3D-печати» (см. Рисунок 72) и нажмите кнопку «Применить».

Нажмите кнопку «Печать», автоматически откроется библиотека с доступными фигурами (см. Рисунок 79).

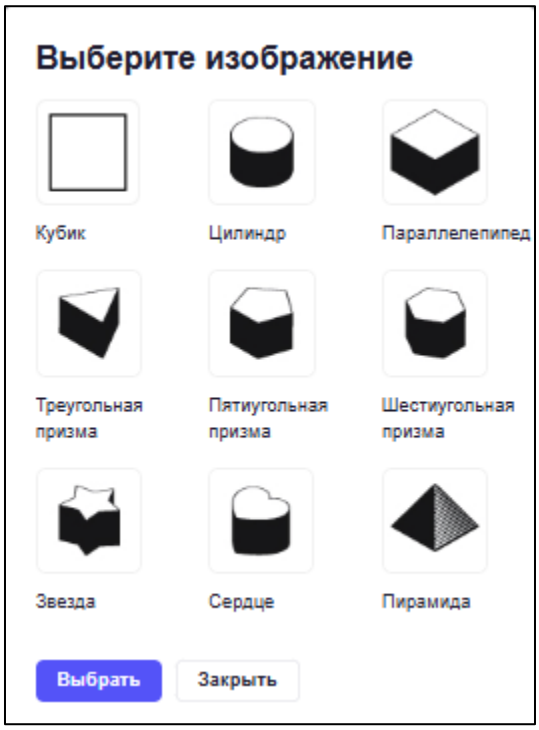


Рисунок 79 – Библиотека фигур

Выберите фигуру и нажмите кнопку «Выбрать». Фигура отобразится в области печати (см. Рисунок 80).

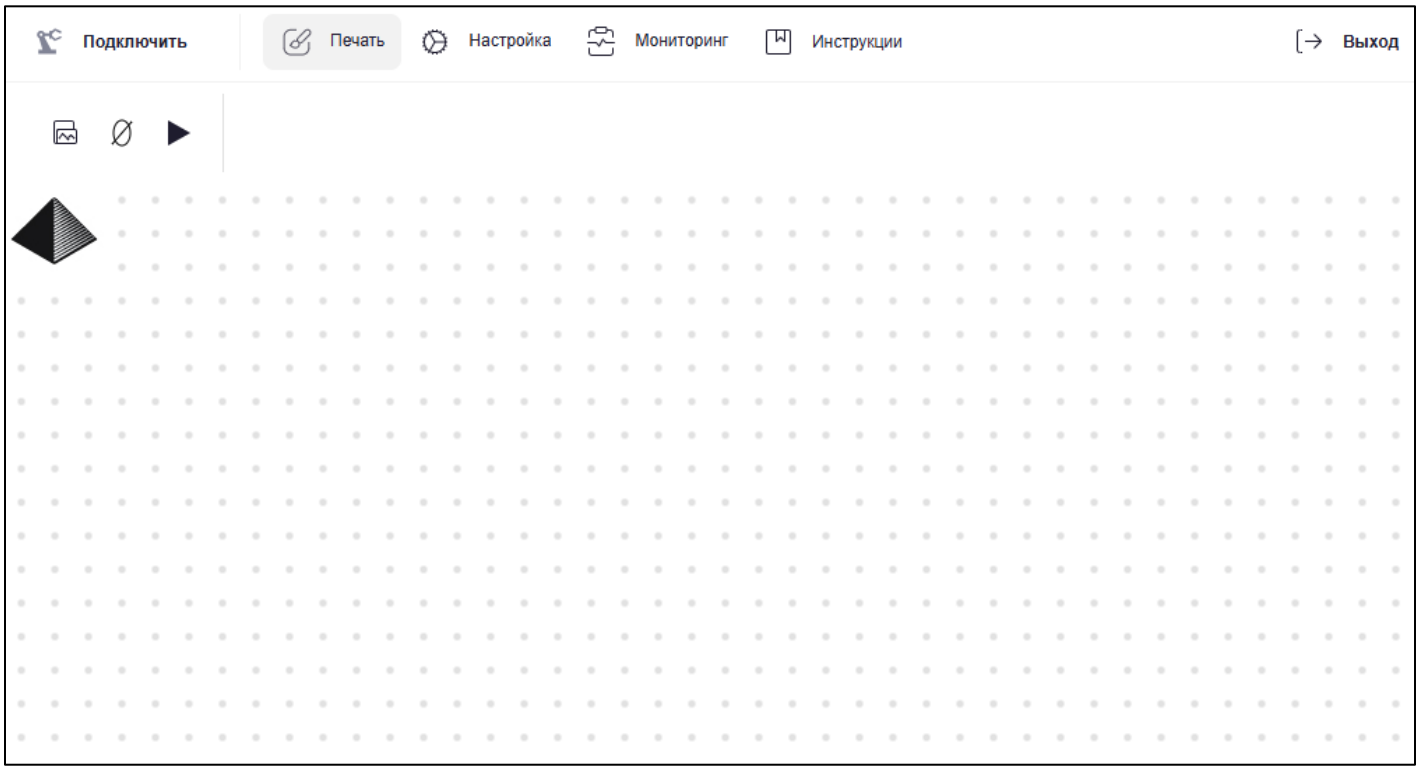


Рисунок 80 – Раздел «Печать»

Перед запуском задайте стартовую позицию стрелы манипулятора:

1. Нажмите кнопку  . Отобразится форма управления «Задание нуля» (см. Рисунок 75).
2. Следуя инструкции, определите начальную точку 3D-печати и нажмите «Задать».

Запустите процесс 3D-печати, нажав кнопку  .

Отобразится уведомление «Программа выполняется» (см. Рисунок 81).

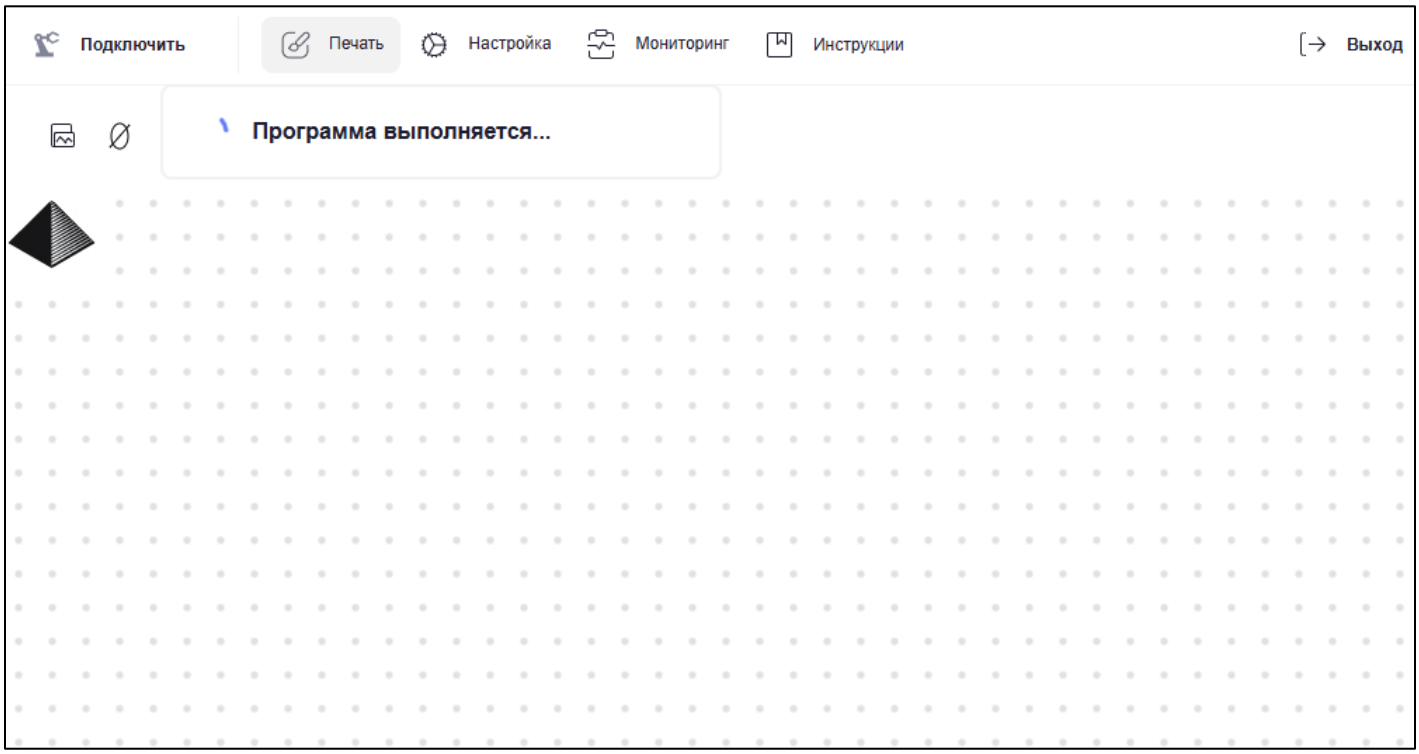


Рисунок 81 – Уведомление «Программа выполняется»

PLA-филамент в печатающей головке начнет нагреваться, при достижении нужной температуры (примерно через 5 минут) манипулятор начнет печать объекта.

Предупреждение! Печатающая головка и выходящий из нее пластик имеют высокую температуру. Прикосновение к работающему оборудованию может привести к ожогам.

Ожидайте не менее 15 минут после окончания печати для снятия 3D-объекта с защитного стекла и не менее 30 минут после окончания печати для демонтажа модуля 3D-печати.

7.3.4 Механический захват

Для ручного управления механическим захватом перейдите в раздел 7.1.2 «Ручное управление «Механический захват».

Для создания программы с механическим захватом перейдите в раздел 7.2 «Создание программ».

7.3.5 Вакуумный захват

Для ручного управления вакуумным захватом перейдите в раздел .

Для создания программы с вакуумным захватом перейдите в раздел 7.2 «Создание программ».

7.3.6 Без насадки

Для ручного управления без насадки перейдите в раздел 7.1.1 «Ручное управление «Без насадки».

Для создания программы без насадки перейдите в раздел 7.2 «Создание программ».

7.4 Мониторинг

Для того чтобы открыть систему мониторинга нажмите кнопку «Мониторинг» на главной панели вкладок.

Отобразится информация о версии интерфейса, релиза и прошивки (см. Рисунок 82). Версии отображаются только при подключенном манипуляторе.

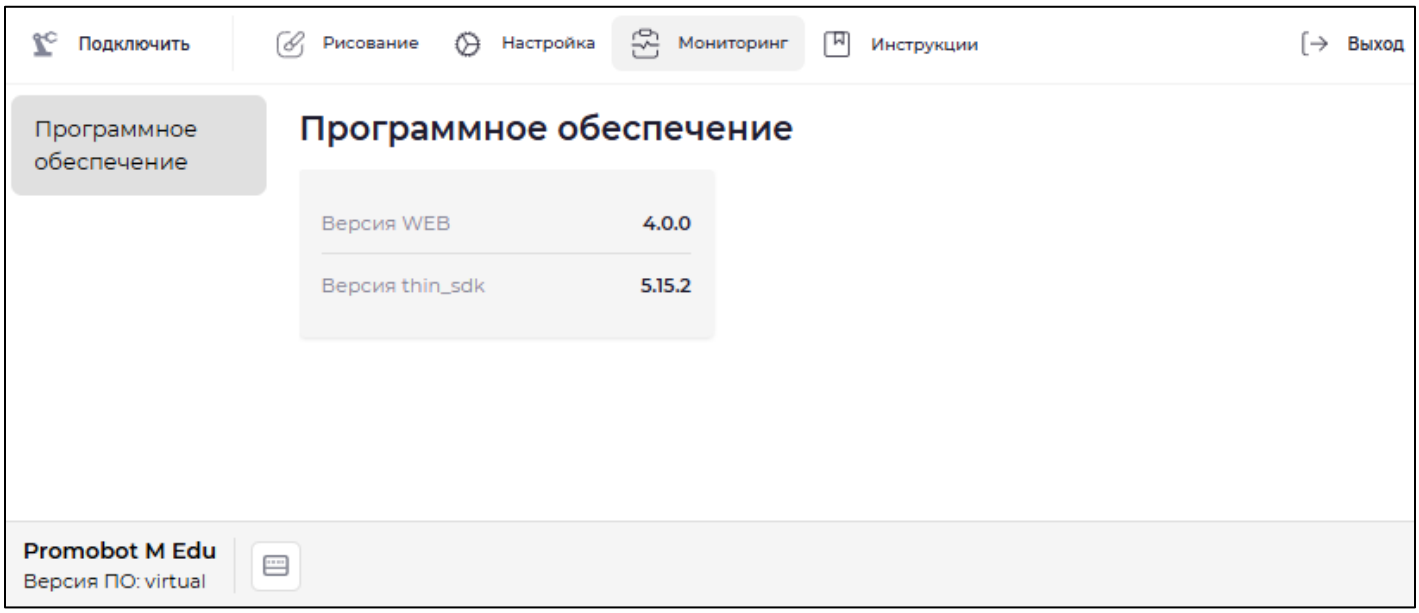


Рисунок 82 – Версии интерфейса, релиза и прошивки

7.5 Обновление приложения Promobot M Control

Перед обновлением сравните версии интерфейса, релиза и прошивки вашего M Edu с версиями в последнем релизе. Информацию о последнем релизе вы можете узнать в инструкции (в приложении нажмите кнопку «Инструкции» и перейдите в раздел 7.8 «Релизы»).

При несоответствии, подключите M Edu к интернету и выполните все обновления по очереди:

1. Обновите ПО:

1) Закройте приложение.
2) Запустите файл «Update_Mcontrol» на рабочем столе (см. Рисунок 83).

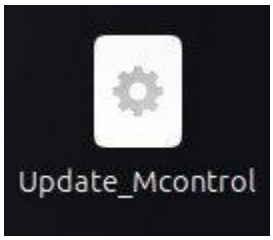


Рисунок 83 – Файл «Update_Mcontrol»

3) Откроется программный интерфейс «Terminal». Отобразится диалоговое окно для выбора типа обновления (см. Рисунок 84).

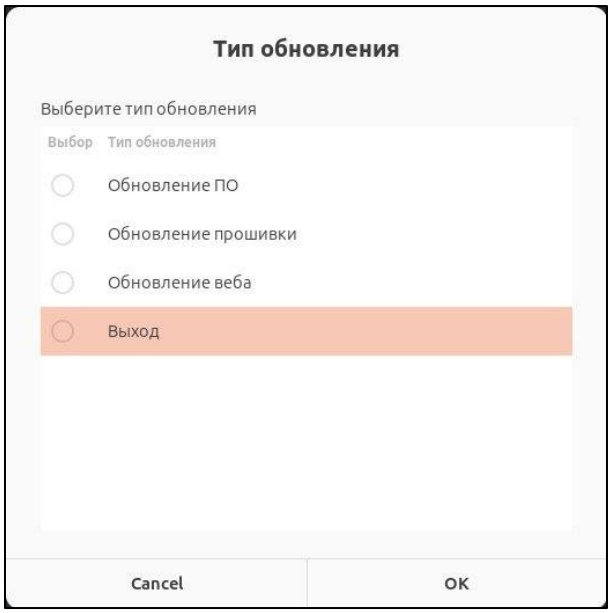


Рисунок 84 – Диалоговое окно для выбора типа обновления

4) Выберите «Обновление ПО» и нажмите «ОК». Отобразится уведомление о запуске обновления (см. Рисунок 85).

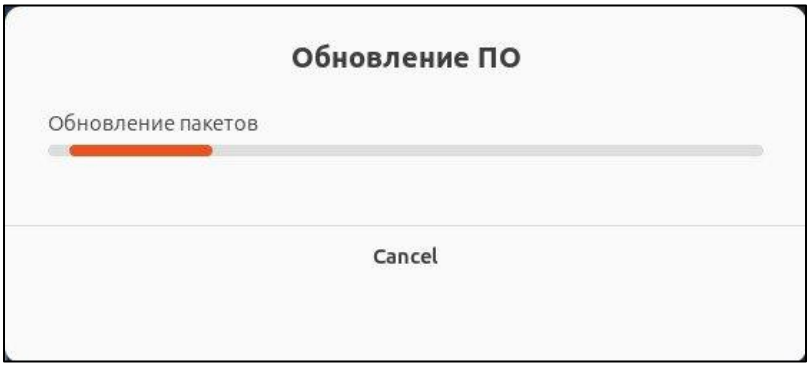


Рисунок 85 – Уведомление о запуске обновления ПО

5) По завершении обновления отобразится уведомление о завершении (см. Рисунок 86). Нажмите «ОК».

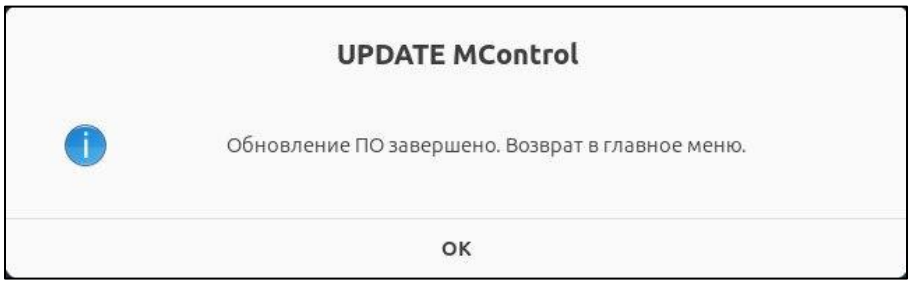


Рисунок 86 – Уведомление о завершении обновления ПО

Дождитесь появления диалогового окна для выбора типа обновления (см. Рисунок 84).

2. Обновите прошивку:

1) Выберите «Обновление прошивки» и нажмите «ОК». Отобразится уведомление «Найдена прошивка: ... Продолжить прошивку?» (см. Рисунок 87).

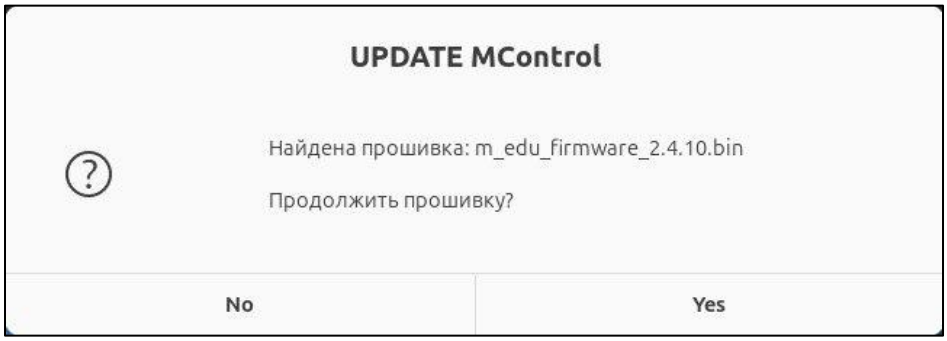


Рисунок 87 – Уведомление прошивки

2) Нажмите «Yes». Отобразится предупреждение «Убедитесь, что: 1) ПО выключено 2) Концевик в безопасном положении».

Рисунок 88 – Предупреждение

3) Убедитесь, что приложение закрыто, а стрела манипулятора лежит на столе. Нажмите «ОК».

4) По завершении обновления отобразится уведомление о завершении. Нажмите «ОК» (см. Рисунок 89).

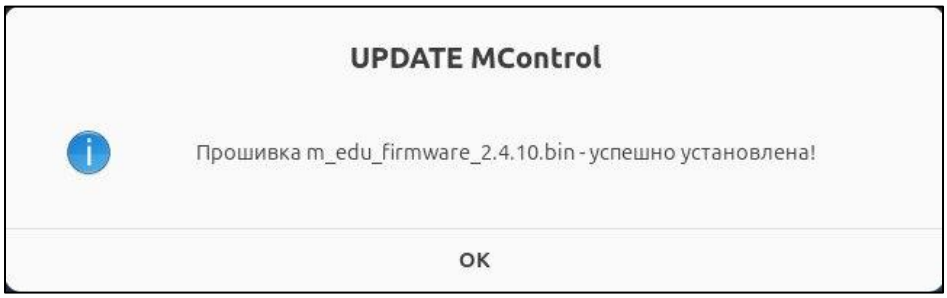


Рисунок 89 – Уведомление о завершении прошивки

Дождитесь появления диалогового окна для выбора типа обновления (см. Рисунок 84).

3. Обновите веб:

1) Выберите «Обновление веба» и нажмите «ОК».
2) По завершении обновления отобразится уведомление о завершении (Рисунок 90). Нажмите «ОК».

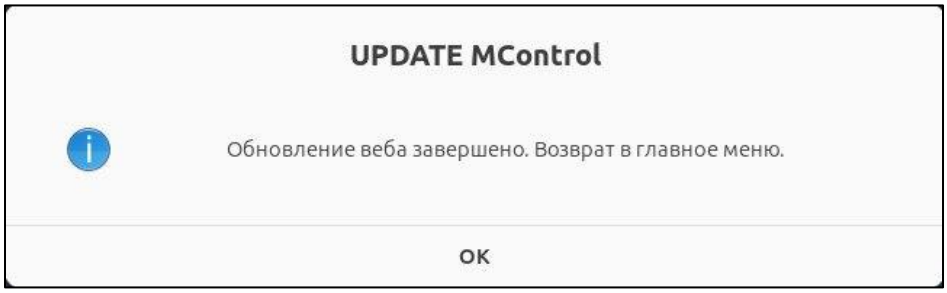


Рисунок 90 – Уведомление о завершении обновления веб-интерфейса

Дождитесь появления диалогового окна для выбора типа обновления (см. Рисунок 84).

4. Закройте диалоговое окно, нажав кнопку «Cancel».

На данный момент одновременное обновление версий ПО, прошивки и веб-интерфейса недоступно. Поочередное обновление гарантирует стабильную работу приложения.

В некоторых случаях может потребоваться перезагрузка устройства после обновления, при этом отобразится уведомление о требовании перезагрузки.

После обновления в интерфейсе мониторинга отобразятся версии, соответствующие последнему релизу.

При возникновении сложностей с обновлением приложения, обратитесь в сервисный центр предприятия-изготовителя.

8 PROMOBOT M CONTROL SDK

8.1 Назначение и область применения

Promobot M Control SDK предоставляет программный интерфейс для управления учебными роботами-манипуляторами под управлением программного обеспечения Promobot M Control, в частности образовательной робототехнической платформой Promobot M Edu, автоматизации задач и их интеграции с внешними системами. Поддерживаются как синхронные, так и асинхронные операции, что обеспечивает гибкость разработки для различных сценариев применения.

Основные возможности SDK:

- подключение манипулятора;
- перемещение и управление движением;
- задание целевых координат в декартовом пространстве или целевых углов каждого узла для точного позиционирования манипулятора;
- управление параметрами движения с помощью коэффициентов скорости и ускорения;
- реализация сложных траекторий, включая движение по дуге и последовательные перемещения по точкам;
- отслеживание положения манипулятора и его узлов в режиме реального времени;
- управление состоянием манипулятора (активный режим, режим ожидания, остановка и т.д.);
- смена режимов управления сервоприводами.

Работа с инструментом и сменными модулями:

- управление питанием инструмента манипулятора;
- управление гриппером и вакуумным насосом;
- управление ориентацией фланца инструмента;
- возможность проигрывания аудиофайлов на манипуляторе для оповещений или пользовательских сценариев.

Взаимодействие с внешними устройствами:

- управление внешними устройствами и датчиками через стандартные промышленные интерфейсы GPIO и I2C.

Автоматизация:

- выполнения программ по указанному имени;
- выполнение Python-скриптов;
- выполнение программ с передачей JSON-данных;
- тайм-ауты операций.

Диагностика:

- обработка сбоев и ошибок манипулятора.

8.2 Архитектура и принципы работы SDK

8.2.1 Компоненты

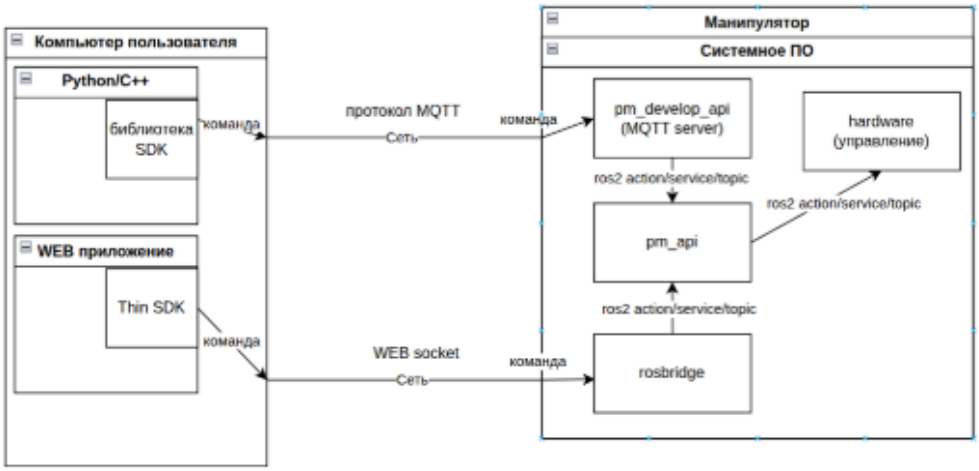


Рисунок 91 – Компоненты

Ключевые компоненты (см. Рисунок 91):

- Клиент: компьютер пользователя / приложение с SDK.
- Сервер (pm_develop_api) – работает на манипуляторе и обрабатывает команды.
- M Control – приложение с собственным SDK, которое переводит действия GUI в ROS2/rosbridge команды.
- Взаимодействие построено по модели клиент – сервер с использованием протокола MQTT.

SDK формирует унифицированные команды управления и телеметрию, которые передаются через брокера сообщений MQTT, действующий как центральный узел, принимающий сообщения от издателей и перенаправляя их соответствующим подписчикам. Серверный компонент pm_develop_api принимает их, проверяет корректность, трассирует сквозь шину обмена и преобразует во внутренние обращения к среде ROS 2. На уровне ROS 2 заявки на движение, запросы периферии и сигналы безопасности маршрутизируются к узлам планирования траекторий и низкоуровневым контроллерам сервоприводов, после чего результаты выполнения и потоковые данные о состоянии публикуются обратно в канал, позволяя приложению-клиенту получать подтверждения и актуальную диагностику.

8.2.2 Модель работы

1. Установка и настройка SDK на стороне клиента.
2. Подключение к контроллеру манипулятора Promobot M Control по IP-адресу и параметрам авторизации.
3. Захват управления манипулятором (get_control) и резервирование канала управления движением.
4. Выполнение команд движения, управления периферией или запуск программ/сценариев.
5. При необходимости – получение состояния манипулятора и диагностических данных.
6. Освобождение управления и завершение работы приложения.

8.3 SDK Python

8.3.1 Требования и подготовка среды Python

8.3.1.1 Сетевые требования

Клиентское приложение и контроллер манипулятора должны находиться в одной сети или иметь корректную маршрутизацию. Порты и протоколы, используемые SDK, не должны блокироваться межсетевыми экранами или политиками безопасности.

Конкретный перечень портов и протоколов определяется реализацией и приводится в технической документации на систему.

8.3.1.2 Требования к программному обеспечению

Требования к программному обеспечению, следующие:

- Рекомендуемая версия Python: 3.12.x.
- Установленное программное обеспечение Promobot M Control на устройстве.

8.3.1.3 Установка SDK Python

8.3.1.3.1 Проверка версии Python

Рекомендуется использование виртуального окружения Python.

Установите Python 3.12, создайте виртуальное окружение и используйте его для запуска SDK – это обеспечит воспроизводимость и отсутствие конфликтов с системными пакетами.

Если Python уже установлен, то проверьте версию:

```
python3.12 --version
python --version #,  "python"
```

Ожидаемый вывод: Python 3.12.x. Если команда не найдена или версия ниже – необходимо установить требуемую версию либо обновить.

8.3.1.3.2 Установка Python 3.12

В зависимости от операционной системы на вашем компьютере выберите необходимый способ установки:

1. Ubuntu / Debian:

```
sudo apt update
sudo apt install -y python3.12 python3.12-venv python3.12-distutils
```

2. Fedora / CentOS / RHEL:
    1) используйте пакетный менеджер дистрибутива или сборку из исходников;
    2) можно также установить через pyenv.
3. macOS (Homebrew):

```
brew update
brew install python@3.12
```

4. Windows – скачайте официальный инсталлятор Python 3.12 с python.org и выполните установку, включив опцию “Add Python to PATH”. После установки проверьте в PowerShell:

```
python --version
```

5. Переиспользование нескольких версий – pyenv (рекомендуется разработчикам):

```
#  pyenv ()
curl https://pyenv.run | bash #
pyenv install 3.12.0
pyenv global 3.12.0
```

8.3.1.3.3 Настройка. Создание виртуального окружения

Виртуальное окружение обязательно для проекта. Рекомендуется запускать SDK внутри виртуального окружения, созданного той же версией Python 3.12.

Создайте виртуальное окружение:

```
# venv
```

Для linux:

```
python3.12 -m venv .venv # source .venv/bin/activate #
```

Для Windows:

```
python -m venv .venv
.venv\Scripts\Activate.ps1
```

Установите библиотеки и зависимости:

```
pip install --index-url https://pypi.org/simple pydantic==2.11.7 paho-mqtt==2.1.0 --extra-index-url https://test.pypi.org/simple/ pm-python-sdk --break-system-packages
```

8.3.2 Общие концепции SDK Python

8.3.2.1 Типы команд Python

Типы команд:

- Синхронные методы – выполняются до завершения операции и возвращают результат.
- Асинхронные методы – имеют суффикс _async и используются с механизмами асинхронности языка (async/await). Возможен вариант:
  - async_await – асинхронный метод, с ожиданием, пока завершится – работает как синхронный.
  - no_wait методы – аналоги асинхронного метода, отправляют команду без ожидания завершения, использовать при необходимости минимальной задержки и с учетом рисков. Данные методы нельзя сделать await, чтобы дождаться статуса выполнился метод или нет.
- Стриминговые методы – периодическая передача команд управления (скорости, позы, суставы) при управлении малыми шагами.

Примеры вызовов:

```
# синхронный
manipulator.get_control()

# асинхронный (awaitable)
await manipulator.get_control_async_await()

#стриминговый
for i in range(5):
    manipulator.stream_cartesian_velocities(lin, ang)
    time.sleep(0.1)

# no_wait
manipulator.get_control_no_wait()
manipulator.get_control() #
```

await manipulator.get_control_async_await() # (awaitable) manipulator.set_servo_twist_mode()

8.3.2.2 Тайм-ауты и обработка ошибок

Большинство методов SDK поддерживают параметры timeout_seconds и throw_error. Рекомендуется использовать явные значения тайм-аутов и обрабатывать исключения на стороне клиентского приложения.

8.3.2.3 Единицы измерения и системы координат

Единицы измерения и системы координат:

- Декартовые координаты: метры.
- Углы суставов: радианы.
- Ориентация инструмента: кватернионы (x, y, z, w).
- Углы/параметры насадок (например, гриппер, поворотный модуль): градусы.

8.3.3 Написание программы Python

Структура программы (или программные блоки) состоит из логических блоков:

1. Подключение библиотек и зависимостей.
2. Подключение к манипулятору и захват управления.
3. Основной блок выполнения (движения, чтение данных).
4. Завершающий блок – опционально (отключение, освобождение ресурсов).

8.3.3.1 Подключение библиотек и зависимостей

Включается всегда

```
from sdk.manipulators.medu import MEdu
```

Необходимы для создания объектов манипулятора, подключения и получения управления.

Включается для управления по декартовым координатам. Используется для задания параметров.

from sdk.commands.move_coordinates_command import MoveCoordinatesParamsPosition, MoveCoordinatesParamsOrientation, PlannerType

Пример:

```
position = MoveCoordinatesParamsPosition(0.15, 0.1, 0.15)
orientation = MoveCoordinatesParamsOrientation(0, 0, 0.1, 0.1)
manipulator.move_to_coordinates_no_wait(position, orientation, 0.2, 0.2)
```

Переключение режимов стриминга

```
from sdk.utils.enums import ServoControlType
```

Пример команды:

```
manipulator.set_servo_control_type(ServoControlType.TWIST, timeout_seconds=5.0)
```

Подключаются для использования объекта насадки на манипуляторе

Например:

- использование гриппера, через объект GripperAttachment;
- использование вакуумного насоса, через объект VacuumAttachment;
- использование лазера, через объект LaserAttachment.

```
from sdk.manipulators.attachments.gripper import GripperAttachment
from sdk.manipulators.attachments.vacuum import VacuumAttachment
from sdk.manipulators.attachments.laser import LaserAttachment
```

Подключаются при использовании для паллетизации

```
from sdk.commands.data import Pose, Position, Orientation
```

Подключаются для установки пределов суставов

```
from sdk.commands.data import JointLimit
```

8.3.3.2 Подключение к манипулятору и захват управления

Подключение к манипулятору и захват управления описано в таблице 5.

Таблица 5 – Подключение к манипулятору и захват управления

Наименование	Описание	Команда	Параметры	Пример
connect	Устанавливает соединение с манипулятором	manipulator.connect()		
connect_async	Асинхронная версия установки соединения	await manipulator.connect_async()		
get_control	Получает доступ к управлению манипулятором. Необходимо вызывать перед отправкой команд	manipulator.get_control(timeout_seconds=60.0, throw_error=True)	- timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	host = "192.168.88.182" #IP манипулятора client_id = "122" #ID клиента login = "13" #Логин password = "14" #Пароль manipulator = MEdu(host, client_id, login, password) manipulator.connect() manipulator.get_control()
get_control_async	Асинхронная версия получения доступа к управлению	await manipulator.get_control_async_await(timeout_seconds=60.0, throw_error=True)		

8.3.3.3 Основной блок выполнения (движения, чтение данных)

Основной блок выполнения (движения, чтение данных) описан в таблице 6.

Таблица 6 – Основной блок выполнения (движения, чтение данных)

Наименование	Описание	Команда	Параметры	Пример
Команды перемещения				
move_to_coordinates	Перемещает манипулятор в указанные декартовые координаты	manipulator.move_to_coordinates(position, orientation, velocity_scaling_factor, acceleration_scaling_factor, planner_type=PlannerType.PTP, timeout_seconds=60.0, throw_error=True)	- position - позиция в декартовых координатах (x, y, z); - orientation – ориентация в кватернионах (x, y, z, w); - velocity_scaling_factor – масштабный коэффициент скорости (от 0 до 1); - acceleration_scaling_factor – масштабный коэффициент ускорения (от 0 до 1); - planner_type – тип	Пример: position = MoveCoordinatesParamsPosition(0.32, -0.004, 0.25) orientation = MoveCoordinatesParamsOrientation(0, 0, 0, 1.0)
move_to_coordinates_async	Асинхронная версия перемещения в указанные декартовые координаты	await manipulator.move_to_coordinates_async(position, orientation, velocity_scaling_factor, acceleration_scaling_factor, planner_type=PlannerType.PTP, timeout_seconds=60.0, throw_error=True)		

Наименование	Описание	Команда	Параметры	Пример
move_to_ coordinates_ async_await	Асинхронная версия с ожиданием пока завершиться (работает как синхронная) перемещения в указанные декартовые координаты	manipulator.move_to_ coordinates_async_aw ait(position, orientation, velocity_scaling_facto r, acceleration_scaling_f actor, planner_type=Planner Type.PTP, timeout_seconds=60.0, throw_error=True)	планировщика движения (PTP или LIN); - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	
move_to_coordi nates_no_wait	Асинхронная версия без ожидания завершения перемещения в указанные декартовые координаты	manipulator.move_to_ coordinates_no_wait(p osition, orientation, velocity_scaling_ factor, acceleration_scaling_ factor, planner_type=Planner Type.PTP, timeout_seconds=60.0, throw_error=True)		
move_to_angles	Перемещает манипулятор, устанавливая суставы в указанные углы	manipulator.move_to_ angles(povorot_osnova niya, privod_plecha, privod_strely, timeout_seconds=60.0, throw_error=True)	M Edu: - povorot_ osnovaniy – целевой угол	

Наименование	Описание	Команда	Параметры	Пример
move_to_angles_async	Асинхронная версия перемещения по углам	await manipulator.move_to_angles_async(povorot_osnovaniya, privod_plecha, privod_strely, timeout_seconds=60.0, throw_error=True) await manipulator.move_to_angles_async(sp1, sp2, sp3, sp4, sp5, sp6, timeout_secondsб throw_error)	поворота основания; - privod_plecha – целевой угол плеча; - privod_strely – целевой угол стрелы	
manipulator.arc_motion	Движение по дуге – это перемещение манипулятора по круговой траектории	manipulator.arc_motion(target, center_arc, step, count_point_arc, max_velocity_scaling_factor, max_acceleration_scaling_factor, timeout_seconds, throw_error)	target – целевая точка (объект с position и orientation); center_arc – точка центра дуги (объект с position и orientation); step – шаг между точками дуги (по умолчанию 0.05);	
manipulator.arc_motion_async	Асинхронная версия перемещение по дуге	await manipulator.arc_motion_async(target, center_arc, step, count_point_arc, max_velocity_scaling_		

Наименование	Описание	Команда	Параметры	Пример
		factor, max_acceleration_scaling_factor)	count_point_arc – количество точек для аппроксимации дуги (по умолчанию 50); max_velocity_scaling_factor – максимальный масштаб скорости (по умолчанию 0.5); max_acceleration_scaling_factor – максимальный масштаб ускорения (по умолчанию 0.5); timeout_seconds – тайм-аут ожидания завершения (секунды); throw_error – выбрасывать исключение при ошибке (по умолчанию True)	
manipulator.arc_motion_no_wait		manipulator.arc_motion_no_wait(target, center_arc, step, count_point_arc, max_velocity_scaling_factor, max_acceleration_scaling_factor)		

Наименование	Описание	Команда	Параметры	Пример
Получение информации о позиции				
get_cartesian_coordinates	Получает текущие декартовы координаты манипулятора	coordinates= manipulator.get_cartesian_coordinates() print(coordinates)	- timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд) - throw_error – флаг выбрасывания исключения при ошибке get_joint_state: Medu - 9: - povorot_osnovaniya - privod_plecha - privod_strely - extruder - privod_z - device - flange_bi - povorot_instrumenta - servo_zahvata	
get_cartesian_coordinates_async_await	Асинхронная версия получения декартовых координат	Coordinates = manipulator.get_cartesian_coordinates_async_await() print(coordinates)		
	Асинхронный метод, с ожиданием, пока завершится (работает как синхронный)	coordinates = await manipulator.get_cartesian_coordinates_async()		
get_joint_state	Получает текущее состояние суставов манипулятора	joints = manipulator.get_joint_state() print(joints)		
get_joint_state_async_await	Асинхронная версия получения состояния суставов	joints = await manipulator.get_joint_state_async_await()		

Наименование	Описание	Команда	Параметры	Пример
Управление инструментами				
nozzle_power	Управляет подачей питания на разъемы на стреле	manipulator.nozzle_power(power, timeout_seconds=60.0, throw_error=True)	- power – признак наличия питания (True – включить, False – выключить); - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	
nozzle_power_async	Асинхронная версия управления питанием на стреле	await manipulator.nozzle_power_async(power, timeout_seconds=60.0, throw_error=True)		
manipulator.nozzle_power_no_wait	Без ожидания	manipulator.nozzle_power_no_wait(True)		
Gripper Attachment	Создание гриппера (доступно только при поданном питании – nozzle_power)	gripper = GripperAttachment(manipulator)	- rotation – угол поворота насадки; - gripper – угол сжатия гриппера; - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд);	
asyncio.to_thread	Асинхронно через thread	await asyncio.to_thread(gripper.activate, rotation=20, gripper=40)		
activate	Управление гриппером	gripper.activate(rotation=20, gripper=10,		

Наименование	Описание	Команда	Параметры	Пример
		timeout_seconds=30.0)	- throw_error – флаг выбрасывания исключения при ошибке	
deactivate	Перевод грипппера в начальное положение	gripper.deactivate(timeout_seconds=30.0) начальное положение можно задать при создании объекта гриппера. gripper = GripperAttachment(manipulator, rotation_open=10, gripper_open=50)		
get_status	Получение статуса	print(f"Статус гриппера: {gripper.get_status()}")		
manage_gripper	Управляет гриппером	manipulator.manage_gripper(rotation, gripper, timeout_seconds=60.0, throw_error=True)		
manipulator.manage_gripper_async	Асинхронная версия управления гриппером	await manipulator.manage_gripper_async(rotation, gripper, timeout_seconds=60.0, throw_error=True)		

Наименование	Описание	Команда	Параметры	Пример
attachment	Создание вакуумного насоса (доступно только при поданном питании - nozzle_power)	vacuum = VacuumAttachment(manipulator)	- rotation – угол поворота насадки; - vacuum – признак включения вакуумного насоса (True – включить, False - выключить); - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	
activate	Активация вакуумного насоса	vacuum.activate(rotation=40, timeout_seconds=30.0)		
deactivate	Деактивация вакуумного насоса	vacuum.deactivate (timeout_seconds=30.0)		
manipulator. manage_vacuum	Управляет вакуумным насосом (доступно только при поданном питании - nozzle_power)	manipulator.manage_vacuum(rotation, vacuum, timeout_seconds=60.0, throw_error=True)		
manage_vacuum_async	Асинхронная версия управления вакуумным насосом	await manipulator.manage_vacuum_async(rotation, vacuum, timeout_seconds=60.0, throw_error=True)		
get_status	Получение статуса	print(f"Статус вакуума: {vacuum.get_status()}")		

Наименование	Описание	Команда	Параметры	Пример
write_gpio	Управление GPIO – это установка значения для GPIO	manipulator.write_gpio(name, value, timeout_seconds=30.0)	name – название GPIO value – значение соответствует двоичному представлению числа, которое будет записано (от 0 до 65535). timeout_seconds – Тайм-аут операции в секундах (по умолчанию 60 секунд) M edu name: '/dev/gpiochip4/stop_key_pin'	
write_gpio_async_await	Асинхронная установка значения GPIO	await manipulator.write_gpio_async_await(name, value)		
write_gpio_no_wait	Установка значения без ожидания	manipulator.write_gpio_no_wait(name, 0)		
get_gpio_value	Чтение значения GPIO	value = manipulator.get_gpio_value(name)		
get_gpio_mask	Чтение маски GPIO	mask_data = manipulator.get_gpio_mask(name)		
write_i2c	Управление i2c – это запись в i2c регистры	manipulator.write_i2c(name, value, timeout_seconds=30.0)	name – название регистра value – значение (int)	Пример: name – "commutator/pwm1" value = 444
write_i2c_async_await	Асинхронная запись в регистры i2c	await manipulator.write_i2c_async_await(name, value)		

Наименование	Описание	Команда	Параметры	Пример
Стриминг				
set_servo_ control_type	Режим TWIST (скорости)	manipulator.set_servo_ control_type(ServoCo ntrolType.TWIST) или manipulator.set_servo_ twist_mode()		
	Режим POSE (целевая поза)	manipulator.set_servo_ control_type(ServoCo ntrolType.POSE) или manipulator.set_servo_ pose_mode()		
	Режим JOINT_JOG (положение суставов)	manipulator.set_servo_ control_type(ServoCo ntrolType.JOINT_JOG) или manipulator.set_servo_ joint_jog_mode()		Пример: joint_positions = {"joint1": 0.0, "joint2": 0.1, "joint3": -0.1} - позиции куда ехать joint_velocities = {"joint1": 0.0, "joint2": 0.0, "joint3": 0.0} – с какой скоростью ехать manipulator.strea m_joint_positions (joint_positions,

Наименование	Описание	Команда	Параметры	Пример
				joint_velocities) – вызов метода
stream_cartesian_velocities	Стриминг скоростей	manipulator.stream_cartesian_velocities(position, velocities)	- position – позиция (объект с x, y, z); - velocities – скорость (объект с rx, ry, rz); - header – информация о штампе времени и системе координат (по умолчанию now и base_link)	Пример: manipulator.set_servo_twist_mode() linear_vel = {"x": 0.02, "y": 0, "z": 0} angular_vel = {"rx": 0, "ry": 0, "rz": 0.01} manipulator.stream_cartesian_velocities(linear_vel, angular_vel)
stream_ coordinates	Стриминг позы	manipulator.stream_coordinates(position, orientation)	- position – позиция (объект с x, y, z). - orientation: ориентация (объект с x, y, z, w). - Header – информация о штампе времени и системе	Пример: manipulator.set_servo_pose_mode() position = MoveCoordinatesParamsPosition(0.27, 0.0, 0.15) orientation = MoveCoordinates

Наименование	Описание	Команда	Параметры	Пример
			координат (по умолчанию now и base_link)	ParamsOrientation(0, 0, 0, 1) manipulator.stream_coordinates(position, orientation)
Программы (запуск готовых программы, JSON-сценариев и выполнение Python-код прямо на роботе)				
run_program	Запуск готовой программы синхронная команда	manipulator.run_program(name, timeout_seconds=30.0, throw_error=True)	- name – название программы (строка); - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	Пример: name = 'edum/default'
manipulator.run_program_async	Запуск готовой программы асинхронная команда	await manipulator.run_program_async(name)		
run_program_no_wait	Запуск готовой программы команда без ожидания	manipulator.run_program_no_wait(name)		
run_program_json	Запуск JSON-программы синхронная команда	manipulator.run_program_json(name, json , timeout_seconds=30.0, throw_error=True)	- name – расположение программы - timeout_seconds – тайм-аут операции в	Пример: name = "alpha/default"; program_json = {
run_program_json_async	Запуск JSON-программы	await manipulator.run_progr		

Наименование	Описание	Команда	Параметры	Пример
	асинхронная команда	am_json_async(name, json)	секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке; - json – JSON-программа – объект с корневым ключом Root и массивом команд	'Root':[{'Move':{ 'content':[{'Point': {'positions':[0.3,-0.3,-0.4],'time':0.5}}],' type':'Simple'}}] }
run_program_ json_no_wait	Запуск JSON-программы без ожидания	manipulator.run_program_json_no_wait(name, json)		
run_python_ program	Запуск Python-кода на роботе синхронная команда	manipulator.run_python_program(code, timeout_seconds=30.0, throw_error=True)	- code – строка с Python-кодом для выполнения на роботе; - timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	Пример: code = "print('Hello!')"
manipulator.run_ python_program _async	Запуск Python-кода на роботе асинхронная команда	await manipulator.run_python_program_async(code)		
.run_python_ program_no_ wait	Запуск Python-кода на роботе команда без ожидания	manipulator.run_python_program_no_wait(code)		

Наименование	Описание	Команда	Параметры	Пример
Остановка движениями (немедленная остановка текущего движения с тайм-аутом)				
stop_movement	Остановка движения синхронная команда	manipulator.stop_movement(timeout_seconds=5.0, throw_error=True)	- timeout_seconds – тайм-аут операции в секундах (по умолчанию 60 секунд); - throw_error – флаг выбрасывания исключения при ошибке	
stop_movement_async	Остановка движения асинхронная команда	await manipulator.stop_movement_async()		
stop_movement_no_wait	Остановка движения команда без ожидания	manipulator.stop_movement_no_wait()		
Состояния и данные (методы для получения состояния манипулятора и подписок)				
get_joint_state	Текущие углы суставов	manipulator.get_joint_state()		
get_home_position	Домашняя позиция	manipulator.get_home_position()		
get_cartesian_coordinates	Текущие координаты X, Y, Z.	manipulator.get_cartesian_coordinates()		
subscribe_coordinates	Подписка на координаты.	manipulator.subscribe_coordinates()		

Наименование	Описание	Команда	Параметры	Пример
subscribe_hardware_error	Подписка на ошибки оборудования.	manipulator.subscribe_hardware_error()	ошибки возвращаются в формате JSON: {"type": id, "message": "..."}	
Конвейерная лента (MGbot) (управление конвейерной лентой и связанными устройствами)				
mgbot_conveyer.set_speed_motors	Установка скорости мотора (0..100)	manipulator.mgbot_conveyer.set_speed_motors(speed)	Параметры (конвейер): - set_speed_motors(speed) – задает скорость мотора в диапазоне 0..100; - set_servo_angle(angle) – поворачивает сервопривод (градусы); - set_led_color(r, g, b) – задает цвет светодиодов (каждый канал 0..255); - display_text(text): показывает текст на	
mgbot_conveyer.set_servo_angle	Поворот сервопривода	manipulator.mgbot_conveyer.set_servo_angle(angle)		
mgbot_conveyer.set_led_color	Установка цвета светодиода (0..255)	manipulator.mgbot_conveyer.set_led_color(r, g, b)		
mgbot_conveyer.display_text	Показ текста на дисплее	manipulator.mgbot_conveyer.display_text(text)		
mgbot_conveyer.set_buzz_tone	Звуковой сигнал (1..15)	manipulator.mgbot_conveyer.set_buzz_tone(level)		
mgbot_conveyer.get_sensors_data	Включить/чтение датчиков	sensor_data = manipulator.mgbot_conveyer.get_sensors_data(enabled)		

Наименование	Описание	Команда	Параметры	Пример
	Возвращаемые поля:	"DistanceSensor", "ColorSensor", "Prox"	дисплее конвейера (utf-8); - set_buzz_tone (level) – подает звуковой сигнал в диапазоне 1..15; - get_sensors_data (enabled) – получить данные с датчиков. True – данные возвращаются в виде JSON. Возвращаемые данные (sensor_data): - DistanceSensor – значение расстояния / показания дистанционного датчика; - ColorSensor – объект с полями R, G, B, Prox (например, {"R":97,"G":114	

Наименование	Описание	Команда	Параметры	Пример
			,"B":108,"Prox": 50}); - Prox – значение приближения к ColorSensor	
Диагностика (если возникли проблемы – используйте диагностические инструменты SDK и обращайтесь в техническую поддержку ПРОМОБОТ с подробными логами)				

8.3.3.4 Завершающий блок (отключение, освобождение ресурсов)

Завершающий блок (отключение, освобождение ресурсов) описан в таблице 7.

Таблица 7 – Завершающий блок (отключение, освобождение ресурсов)

Наименование	Описание	Команда	Параметры
Команды отключения			
disconnect	Разрыв соединения с шиной сообщений и остановка внутренних потоков MQTT	manipulator.disconnect()	Метод безопасен к повторному вызову: если соединение уже разорвано, он ничего не делает

8.3.4 Камера Pixy2 в SDK Python для захвата объектов

Pixy2 – это компактная камера с процессором внутри (см. Рисунок 92). Она сама обрабатывает изображение и выделяет объекты по цветовым меткам («сигнатурам»). Манипулятор получает уже готовые данные – координаты прямоугольников объектов, их размеры и сигнатуру.

Рисунок 92 – Камера Pixy2

Основные возможности Pixy2:

- Распознавание цветовых объектов по заранее обученным сигнатурам (например, красный кубик или зеленый маркер).
- Передача координат объекта (X, Y, ширина, высота).
- Работа через разъемы USB, UART.

8.3.4.1 Начало работы с камерой Pixy2

Для работы с камерой Pixy2 выполните следующие действия:

1. Подготовьте камеру:
    1) Подключите камеру Pixy2 к манипулятору через разъем USB и установите ее по центру рабочей области.
    2) Выведите манипулятор с захватом в позицию над рабочей областью.
    3) Откалибруйте камеру и манипулятор, чтобы захват находился строго в центре камеры. Для контроля позиции манипулятора относительно центра камеры можно воспользоваться приложением PixyMon, которое можно скачать в интернете. Но во время последующей работы с Pixy2 приложение PixyMon должно быть закрыто.
    4) Откройте терминал и введите команду: ros2 topic echo /pm_api/current_cartesian_coordinates.
    5) Зафиксируйте отобразившиеся координаты манипулятора на бумаге, а также зафиксируйте высоту камеры (например, с помощью линейки).

6) Обучите Pixy2 распознавать объект:
   - наведите камеру на нужный предмет;
   - нажмите и удерживайте кнопку на Pixy2;
   - остановитесь на нужном цвете и отпустите кнопку, далее обучение в соответствии с цветом светодиода, например (красный – Signature 1, желтый – Signature 2 и т.д.**)**; когда светодиод мигнет – обучение завершено.

Теперь Pixy2 будет распознавать этот цвет и передавать координаты. Чем камера лучше распознает объект, тем ярче светится светодиод на Pixy2.

7) Откройте файл pixy2_config.yaml (/opt/promobot/lib/python3.12/site-packages/pm_develop_api_lib/applications/pixy2/config/) и введите полученные координаты и высоту, например:
   - x_center: 0.3177 – координата X;
   - y_center: -0.0038 – координата Y;
   - z_center: 0.2 – координата Z (высота инструмента над объектом);
   - height_m: 0.5 – высота камеры.

2. Настройка параметров камеры – все параметры можно указать в конфиге pixy2_config.yaml:

```
pixy:
  res_x: 316        # разрешение по X (пиксели)
  res_y: 208        # разрешение по Y (пиксели)
  x_center: 0.3177 # центр рабочей области по X (в метрах)
  y_center: -0.0038 # центр рабочей области по Y
  z_center: 0.2     # высота рабочей плоскости (м)
  height_m: 0.50    # расстояние от камеры до объекта
  hfov_deg: 60.0    # горизонтальный угол обзора (градусы)
  vfov_deg: 40.0    # вертикальный угол обзора
  x_max: 0.3785     # ограничение максимального X
  x_min: 0.0        # ограничение минимального X
  approach_distance_x: 0.030 # смещение по X при захвате
  approach_distance_y: 0.020 # смещение по Y при захвате
  timeout_sec: 5.0 # таймаут ожидания объекта
```

8.3.4.2 Пример скрипта для SDK для наведения на объект

```
import asyncio
from sdk.commands.move_coordinates_command import MoveCoordinatesParamsPosition, MoveCoordinatesParamsOrientation
from sdk.manipulators.medu import Medu
import json

# Конфигурация подключения
host: str = '127.0.0.1'
client_id: str = '122'
login: str = '13'
password: str = '14'

manipulator = Medu (host, client_id, login, password)

async def run_all_tests():
    try:
        print('\n=== Тестирование асинхронных методов ===')
        # 1. Асинхронное подключение и получение управления
        print('\n1. Асинхронное подключение и получение управления')
        await manipulator.connect_async()
        print('Асинхронное подключение к манипулятору установлено')
        await asyncio.sleep(1)

        await manipulator.get_control_async_await()
        print('Асинхронно получено управление манипулятором')
        await asyncio.sleep(2)

        await manipulator.pixy_cam_usb_control.get_blocks_async_await()
        await asyncio.sleep(1)
```

```
            coord = await manipulator.get_block_coordinates_from_pixy_async_await(1)
            #В качестве параметра передается id сигнатуры объекта (например: красный - 1)
            await asyncio.sleep(1)

            if isinstance(coord, str):
                coord = json.loads(coord)

            position = MoveCoordinatesParamsPosition(
                coord["Coordinates"]["x"],
                coord["Coordinates"]["y"],
                coord["Coordinates"]["z"],
            )

            orientation = MoveCoordinatesParamsOrientation(0.5, 0.5, -0.5, -0.5)
            await manipulator.move_to_coordinates_async_await(position, orientation, 0.1, 0.1)
            await asyncio.sleep(1)

        except Exception as e:
            print(f'Возникла ошибка: {e}')


    if __name__ == '__main__':
        print('Запуск тестирования манипулятора')
        asyncio.run(run_all_tests())
```

8.3.4.3 Возможные проблемы и их решения

Очень важную роль играет высота камеры. Так как угол обзора Pixy2 сильно ограничен, а разрешающая способность камеры низкая, то может возникнуть ситуация, когда камера находится на таком расстоянии от рабочей области, что возникает эффект «рыбьего глаза», в следствие чего манипулятор неверно рассчитывает реальное положение объекта. При высоте выше 40 см, необходимо корректировать параметры hfov_deg, vfov_deg.

При тестировании использовались следующие значения:

height_m: 0.8

hfov_deg: 75.0

vfov_deg: 51.0

Если захват не доезжает до цели, то увеличьте параметры, либо наоборот уменьшите, если переезжает.

В случае, если возникает ошибка подключения камеры Pixy2, убедитесь, что приложение PixyMon закрыто и перезапустите софт.

8.3.4.4 Управление Pixy2 в SDK

Команды управления Pixy2 в SDK описаны в таблице 8.

Таблица 8 – Команды управления Pixy2 в SDK

Наименование	**Описание**	**Команда**	**Параметры**	**Пример**
Подготовка и конфигурация				
pixy2_ config.yaml	Файл конфигурации камеры и расчета координат в рабочей зоне. Используется до запуска команд захвата по Pixy2	/opt/promobot/lib/python3.12/ site-packages/pm_develop_api_lib/ applications/pixy2/ config/ pixy2_config.yaml	- res_x, res_y – разрешение; - x_center, y_center, z_center – центр рабочей области; - height_m – высота камеры; - hfov_deg, vfov_deg – углы обзора; - x_max, x_min – ограничения по X;	pixy: res_x: 316 res_y: 208 x_center: 0.3177 y_center: -0.0038 z_center: 0.2 height_m: 0.50

Наименование	Описание	Команда	Параметры	Пример
			- approach_ distance_x, approach_ distance_y; - timeout_sec	
Калибровка и обучение сигнатуры	Подготовка к работе: установка камеры, вывод захвата в центр поля зрения, фиксация координат и обучение цветовой сигнатуре на самой Pixy2	ros2 topic echo /pm_api/current_cartesian_coordinates	Фиксируются X, Y, Z центра; высота камеры; сигнатура объекта (например, Signature 1)	Во время работы SDK приложение PixyMon должно быть закрыто
Общие методы PixyCamModuleBase (доступны для UART и USB)				
get_blocks_async	Получить список обнаруженных цветовых объектов по сигнатурам. Асинхронная форма: создает объект команды	manipulator.pixy_cam_usb_control.get_blocks_async(...)	- sigmap=0xFF – маска сигнатур; - max_blocks=100 – лимит количества объектов; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.get_blocks_async(sigmap=0xFF, max_blocks=100) await cmd.make_command_action() result = await cmd.result()
get_blocks_async_await	Получить список обнаруженных цветовых	await manipulator.pixy_cam_usb_control.	- sigmap=0xFF – маска сигнатур;	result = await manipulator.pixy_cam_usb_control.get_bl

Наименование	Описание	Команда	Параметры	Пример
	объектов по сигнатурам. Асинхронная форма с ожиданием результата	get_blocks_async_await(...)	- max_blocks=100 – лимит количества объектов; - timeout_seconds; - throw_error	ocks_async_await(sigmap=0xFF, max_blocks=100)
get_blocks	Получить список обнаруженных цветовых объектов по сигнатурам. Синхронная форма	manipulator.pixy_cam_usb_control.get_blocks(...)	- sigmap=0xFF – маска сигнатур; - max_blocks=100 – лимит количества объектов; - timeout_seconds; - throw_error	result = manipulator.pixy_cam_usb_control.get_blocks(sigmap=0xFF, max_blocks=100)
get_rgb_async	Получить RGB значения пикселя в заданной точке кадра. Асинхронная форма: создаёт объект команды.	manipulator.pixy_cam_usb_control.get_rgb_async(...)	x, y – координаты точки; saturate=False – признак насыщения значений; timeout_seconds; throw_error	cmd = manipulator.pixy_cam_usb_control.get_rgb_async(x=50, y=40, saturate=False) await cmd.make_command_action() result = await cmd.result()
get_rgb_async_await	Получить RGB значения пикселя в заданной точке кадра.	await manipulator.pixy_cam_usb_control.	- x, y – координаты точки;	result = await manipulator.pixy_cam_usb_control.get_rg

Наименование	Описание	Команда	Параметры	Пример
	Асинхронная форма с ожиданием результата	get_rgb_async_await(...)	- saturate=False – признак насыщения значений; - timeout_seconds; - throw_error	b_async_await(x=50, y=40, saturate=False)
get_rgb	Получить RGB значения пикселя в заданной точке кадра. Синхронная форма	manipulator.pixy_cam_usb_control.get_rgb(...)	- x, y – координаты точки; - saturate=False – признак насыщения значений; - timeout_seconds; - throw_error	result = manipulator.pixy_cam_usb_control.get_rgb(x=50, y=40, saturate=False)
set_lamp_async	Управление подсветкой Pixy2. Асинхронная форма: создает объект команды	manipulator.pixy_cam_usb_control.set_lamp_async(...)	- upper, lower – включить/выключить лампы (0/1); - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.set_lamp_async(1, 1) await cmd.make_command_action() result = await cmd.result()
set_lamp_async_await	Управление подсветкой Pixy2. Асинхронная форма с	await manipulator.pixy_cam_usb_control.set_lamp_async_await(...)	- upper, lower – включить/выключить лампы (0/1);	result = await manipulator.pixy_cam_usb_control.set_lamp_async_await(1, 1)

Наименование	Описание	Команда	Параметры	Пример
	ожиданием результата		- timeout_seconds; - throw_error	
set_lamp	Управление подсветкой Pixy2. Синхронная форма	manipulator.pixy_cam_usb_control.set_lamp(...)	- upper, lower – включить/выключить лампы (0/1); - timeout_seconds; - throw_error	result = manipulator.pixy_cam_usb_control.set_lamp(1, 1)
UART – специфичные методы PixyCamUartModule				
get_version_async	Получить версию Pixy2 (как ее возвращает контроллер). Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.get_version_async(...)	- timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.get_version_async() await cmd.make_command_action() result = await cmd.result()
get_version_async_await	Получить версию Pixy2 (как ее возвращает контроллер). Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.get_version_async_await(...)	timeout_seconds; throw_error	result = await manipulator.pixy_cam_uart_control.get_version_async_await()

Наименование	Описание	Команда	Параметры	Пример
get_version	Получить версию Pixy2 (как ее возвращает контроллер). Синхронная форма	manipulator.pixy_cam_uart_control.get_version(...)	- timeout_seconds; - throw_error	result = manipulator.pixy_cam_uart_control.get_version()
get_resolution_async	Получить разрешение кадра через UART-канал. Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.get_resolution_async(...)	- type=0 – режим/тип запроса; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.get_resolution_async(type=0) await cmd.make_command_action() result = await cmd.result()
get_resolution_async_await	Получить разрешение кадра через UART-канал. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.get_resolution_async_await(...)	- type=0 – режим/тип запроса; - timeout_seconds; - throw_error	result = await manipulator.pixy_cam_uart_control.get_resolution_async_await(type=0)
get_resolution_uart	Получить разрешение кадра через UART-канал. Синхронная форма	manipulator.pixy_cam_uart_control.get_resolution_uart(...)	- type=0 – режим/тип запроса; - timeout_seconds; - throw_error	result = manipulator.pixy_cam_uart_control.get_resolution_uart(type=0)

Наименование	Описание	Команда	Параметры	Пример
get_fps_async	Получить текущий FPS камеры через UART. Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.get_fps_async(...)	- timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.get_fps_async() await cmd.make_command_action() result = await cmd.result()
get_fps_async_await	Получить текущий FPS камеры через UART. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.get_fps_async_await(...)	- timeout_seconds; - throw_error	result = await manipulator.pixy_cam_uart_control.get_fps_async_await()
get_fps_uart	Получить текущий FPS камеры через UART. Синхронная форма	manipulator.pixy_cam_uart_control.get_fps_uart(...)	- timeout_seconds; - throw_error	result = manipulator.pixy_cam_uart_control.get_fps_uart()
set_camera_brightness_async	Установить яркость камеры через UART. Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.set_camera_brightness_async(...)	- brightness – целое значение яркости; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.set_camera_brightness_async(80) await cmd.make_command_action()

Наименование	Описание	Команда	Параметры	Пример
				result = await cmd.result()
set_camera_brightness_async_await	Установить яркость камеры через UART. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.set_camera_brightness_async_await(...)	- brightness – целое значение яркости; - timeout_seconds; throw_error	result = await manipulator.pixy_cam_uart_control.set_camera_brightness_async_await(80)
set_camera_brightness_uart	Установить яркость камеры через UART. Синхронная форма	manipulator.pixy_cam_uart_control.set_camera_brightness_uart(...)	- brightness – целое значение яркости; - timeout_seconds; - throw_error	result = manipulator.pixy_cam_uart_control.set_camera_brightness_uart(80)
set_servos_async	Управление двумя сервоканалами Pixy2 через UART. Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.set_servos_async(...)	- s0, s1 – целевые значения сервоканалов; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.set_servos_async(300, 400) await cmd.make_command_action() result = await cmd.result()

Наименование	Описание	Команда	Параметры	Пример
set_servos_async_await	Управление двумя сервоканалами Pixy2 через UART. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.set_servos_async_await(...)	- s0, s1 – целевые значения сервоканалов; - timeout_seconds; - throw_error	result = await manipulator.pixy_cam_uart_control.set_servos_async_await(300, 400)
set_servos_uart	Управление двумя сервоканалами Pixy2 через UART. Синхронная форма	manipulator.pixy_cam_uart_control.set_servos_uart(...)	- s0, s1 – целевые значения сервоканалов; - timeout_seconds; - throw_error	result = manipulator.pixy_cam_uart_control.set_servos_uart(300, 400)
set_led_async	Установить цвет RGB-светодиода Pixy2 через UART. Асинхронная форма: создает объект команды	manipulator.pixy_cam_uart_control.set_led_async(...)	- r, g, b – компоненты цвета; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_uart_control.set_led_async(255, 0, 0) await cmd.make_command_action() result = await cmd.result()

Наименование	Описание	Команда	Параметры	Пример
set_led_async_await	Установить цвет RGB-светодиода Pixy2 через UART. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_uart_control.set_led_async_await(...)	- r, g, b – компоненты цвета; - timeout_ seconds; - throw_error	result = await manipulator.pixy_cam_uart_control.set_led_async_await(255, 0, 0)
set_led_uart	Установить цвет RGB-светодиода Pixy2 через UART. Синхронная форма	manipulator.pixy_cam_uart_control.set_led_uart(...)	- r, g, b – компоненты цвета; - timeout_ seconds; - throw_error	result = manipulator.pixy_cam_uart_control.set_led_uart(255, 0, 0)
USB – специфичные методы PixyCamUsbModule				
line_all_ features_async	Получить полный набор line-features: векторы, пересечения, штрих-коды и др. Состав ответа зависит от backend. Асинхронная форма: создает объект команды	manipulator.pixy_cam_usb_control.line_all_features_async(...)	- max_items= 100 – ограничение количества элементов; - timeout_ seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.line_all_features_async(max_items=100) await cmd.make_command_action() result = await cmd.result()

Наименование	Описание	Команда	Параметры	Пример
line_all_ features_async_ await	Получить полный набор line-features: векторы, пересечения, штрих-коды и др. Состав ответа зависит от backend. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_usb_control. line_all_features_async_await(...)	- max_items=100 – ограничение количества элементов; - timeout_ seconds; - throw_error	result = await manipulator.pixy_cam_usb_control.line_all_features_async_await(max_items=100)
line_all_ features_usb	Получить полный набор line-features: векторы, пересечения, штрих-коды и др. Состав ответа зависит от backend. Синхронная форма	manipulator.pixy_cam_usb_control. line_all_features_usb(...)	- max_items= 100 – ограничение количества элементов; - timeout_ seconds; - throw_error	result = manipulator.pixy_cam_usb_control.line_all_features_usb(max_items=100)

Наименование	Описание	Команда	Параметры	Пример
line_main_ features_async	Получить сокращённый набор основных line-features. Асинхронная форма: создает объект команды	manipulator.pixy_cam_usb_control. line_main_features _async(...)	- max_items= 100; - timeout_ seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.line_main_features_async(max_items=100) await cmd.make_command_action() result = await cmd.result()
line_main_ features_ async_await	Получить сокращённый набор основных line-features. Асинхронная форма с ожиданием результата.	await manipulator.pixy_cam_usb_control. line_main_features _async_await(...)	- max_items= 100; - timeout_ seconds; - throw_error	result = await manipulator.pixy_cam_usb_control.line_main_features_async_await(max_items=100)
line_main_ features_usb	Получить сокращенный набор основных line-features. Синхронная форма	manipulator.pixy_cam_usb_control. line_main_features _usb(...)	- max_items= 100; - timeout_ seconds; - throw_error	result = manipulator.pixy_cam_usb_control.line_main_features_usb(max_items=100)

Наименование	Описание	Команда	Параметры	Пример
get_frame_size_async	Получить размер кадра через USB-канал. Асинхронная форма: создает объект команды	manipulator.pixy_cam_usb_control.get_frame_size_async(...)	- timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.get_frame_size_async() await cmd.make_command_action() result = await cmd.result()
get_frame_size_async_await	Получить размер кадра через USB-канал. Асинхронная форма с ожиданием результата.	await manipulator.pixy_cam_usb_control.get_frame_size_async_await(...)	- timeout_seconds; - throw_error	result = await manipulator.pixy_cam_usb_control.get_frame_size_async_await()
get_frame_size_usb	Получить размер кадра через USB-канал. Синхронная форма.	manipulator.pixy_cam_usb_control.get_frame_size_usb(...)	- timeout_seconds; - throw_error	result = manipulator.pixy_cam_usb_control.get_frame_size_usb()
set_servos_async	Управление сервоканалами Pixy2 через USB. Асинхронная форма: создаёт объект команды.	manipulator.pixy_cam_usb_control.set_servos_async(...)	- s0, s1 – целевые значения сервоканалов; - timeout_seconds; - throw_error	cmd = manipulator.pixy_cam_usb_control.set_servos_async(300, 400) await cmd.make_command_action() result = await cmd.result()

Наименование	Описание	Команда	Параметры	Пример
set_servos_async_await	Управление сервоканалами Pixy2 через USB. Асинхронная форма с ожиданием результата	await manipulator.pixy_cam_usb_control.set_servos_async_await(...)	- s0, s1 – целевые значения сервоканалов; - timeout_ seconds; - throw_error	result = await manipulator.pixy_cam_usb_control.set_servos_async_await(300, 400)
set_servos_usb	Управление сервоканалами Pixy2 через USB. Синхронная форма.	manipulator.pixy_cam_usb_control.set_servos_usb(...)	- s0, s1 – целевые значения сервоканалов; - timeout_ seconds; - throw_error	result = manipulator.pixy_cam_usb_control.set_servos_usb(300, 400)
Получение координат объекта через Manipulator				
get_block_coordinates_from_pixy_async	Получить координаты объекта по номеру сигнатуры через отдельную команду pixy_get_coordinates и топик /pixy_ coordinates. Асинхронная форма: создает объект команды	manipulator. get_block_coordinates_from_pixy_async(...)	- signature – номер сигнатуры объекта; - timeout_seconds=60.0; - throw_error	cmd = manipulator.get_block_coordinates_from_pixy_async(1) await cmd.make_command_action() result = await cmd.result()

Наименование	Описание	Команда	Параметры	Пример
get_block_coordinates_from_pixy_async_await	Получить координаты объекта по номеру сигнатуры через отдельную команду pixy_get_coordinates и топик /pixy_ coordinates. Асинхронная форма с ожиданием результата	await manipulator.get_block_coordinates_from_pixy_async_await(...)	- signature – номер сигнатуры объекта; - timeout_seconds=60.0; - throw_error	result = await manipulator.get_block_coordinates_from_pixy_async_await(1)
get_block_coordinates_from_pixy	Получить координаты объекта по номеру сигнатуры через отдельную команду pixy_get_coordinates и топик /pixy_ coordinates. Синхронная форма	manipulator.get_block_coordinates_from_pixy(...)	- signature – номер сигнатуры объекта; - timeout_seconds=60.0; - throw_error	result = manipulator.get_block_coordinates_from_pixy(1)

Наименование	Описание	Команда	Параметры	Пример
get_pixy_ coordinates_ async	Низкоуровневое ожидание сообщения из топика /pixy_coordinates	await manipulator. get_pixy_coordinates_async(...)	- timeout_seconds=60.0; - throw_error	payload = await manipulator. get_pixy_coordinates_async()
Практические замечания по эксплуатации				
Коррекция геометрии	При большой высоте камеры координаты могут рассчитываться с ошибкой из-за ограниченного угла обзора и эффекта «рыбьего глаза»	–	При высоте выше 40 см рекомендуется корректировать hfov_deg и vfov_deg	Тестовые значения: height_m: 0.8 hfov_deg: 75.0 vfov_deg: 51.0
Ошибки подключения	Если Pixy2 не подключается, нужно закрыть pixymon и перезапустить софт	–	PixyMon не должно удерживать устройство во время работы SDK	При недоезде/переезде захвата корректировать hfov_deg и vfov_deg.

8.4 SDK C++

8.4.1 Требования и использование C++ SDK

Манипулятор и компьютер должны быть доступны друг другу по сети (локально или через Интернет).

SDK требует компилятор и систему сборки CMake. Требование к компилятору: компилятор с поддержкой C++17 (GCC 9+, Clang 10+, MSVC 2019+). Библиотека распространяется в предварительно собранном виде.

Проверка компилятора

```
g++ --version
# ожидается версия 9 или выше
```

Структура установленного SDK

После получения SDK, в корневой папке находится каталог install со следующей структурой:

```
install/
├── include/
│   ├── pm_sdk/          # Заголовочные файлы SDK
│   │   ├── manipulator.hpp
│   │   ├── types.hpp
│   │   ├── debug_logger.hpp
│   │   └── nlohmann/    # JSON библиотека
│   └── mqtt/            # Заголовочные файлы Paho MQTT
├── lib/
│   ├── libpm_sdk.a      # Статическая библиотека SDK
│   ├── libpaho-mqttpp3.so    # Paho MQTT C++ библиотека
│   └── libpaho-mqtt3*.so     # Paho MQTT C библиотеки
└── bin/
    └── MQTTVersion      # Утилита для проверки MQTT
```

Подключение SDK к проекту

Пример CMakeLists.txt для вашего проекта:

```
cmake_minimum_required(VERSION 3.10)
cmake_minimum_required(VERSION 3.13)
project(sdk_demo)
```

```
set(CMAKE_CXX_STANDARD 20)

include_directories(${CMAKE_SOURCE_DIR}/../install/include)
link_directories(${CMAKE_SOURCE_DIR}/../install/lib)

add_executable(${PROJECT_NAME} main.cpp)

target_link_libraries(${PROJECT_NAME}
    pm_sdk
    paho-mqttpp3
    paho-mqtt3as
  )
)
```

8.4.2 Типы команд C++

SDK поддерживает несколько типов команд для гибкого управления:

- Синхронные – выполняются по очереди и блокируют выполнение до завершения. Принимают параметр throwException (по умолчанию true).
- Асинхронные – не блокируют выполнение; имеют суффикс Async и принимают callback-функцию.
- Стриминговые – для реального управления малыми шагами (скорости, позы, суставы).
- Подписки – для асинхронного получения данных состояния через Subscribe/Unsubscribe.

Важно! Не все методы имеют асинхронные версии. Например:

- GetControl() – только синхронный
- Connect() – только синхронный
- CancelCommand() – только синхронный
- GetCartesianCoordinates() – только синхронный (но есть асинхронная подписка SubscribeCartesianCoordinates)

Большинство команд движения (MoveToAngles, MoveToCoordinates и др.) имеют как синхронные, так и асинхронные версии. Всегда проверяйте наличие суффикса Async в заголовочных файлах.

Примеры вызовов

```
// Синхронный вызов (блокирующий)
manipulator.GetControl();
manipulator.MoveToAngles(angles);

// Асинхронный вызов (с callback)
manipulator.MoveToAnglesAsync(angles, [](CommandResult result) {
    if (result.GetState()) {
        std::cout << "Движение завершено" << std::endl;
    }
});

// асинхронная подписка на состояния
manipulator.SubscribeCartesianCoordinates([](const nlohmann::json& data) {
    std::cout << "Координаты обновлены: " << data.dump() << std::endl;
});
```

Управление логированием

Для отладки SDK предусмотрена система логирования:

```
// Подключить заголовочный файл
#include "pm_sdk/debug_logger.hpp"

// Включить все логи
DebugLogger::SetEnable(LogEntity::ALL, true);

// Включить только логи SDK
DebugLogger::SetEnable(LogEntity::SDK, true);
```

```
// Включить только логи MQTT
DebugLogger::SetEnable(LogEntity::MQTT, true);

// Отключить все логи
DebugLogger::SetEnable(LogEntity::ALL, false);
```

Далее в примерах будут использоваться синхронные методы, если не указано иное.

8.4.3 Программные блоки (структура программы)

Структуру программы удобно разбить на логические блоки:

1. Подключение заголовочных файлов SDK.
2. Создание объекта манипулятора и подключение к нему.
3. Захват управления.
4. Основной блок выполнения (движения, чтение данных).
5. Обработка исключений.

Ниже – минимальный пример подключения и захвата управления.

```
#include "pm_sdk/manipulator.hpp"
#include <iostream>

int main() {
    using namespace pb_sdk;

    std::string host = "192.168.88.182";  // IP манипулятора
    std::string client_id = "122";        // ID клиента
    std::string login = "13";             // Логин
    std::string password = "14";          // Пароль

    try {
        MEdu manipulator(host, client_id, login, password);
        manipulator.Connect();
```

```
    if (manipulator.GetControl()) {
        std::cout << "Управление получено!" << std::endl;
    }
} catch (const std::exception& e) {
    std::cerr << "Ошибка: " << e.what() << std::endl;
}

return 0;
}
```

8.4.4 Основной блок выполнения (движения, чтение данных)

Основной блок выполнения описан в таблице 9.

Таблица 9 – Основной блок выполнения (движения, чтение данных)

Наименование	Описание	Команда	Параметры	Пример
Команды перемещения				

MoveToAngles	Движение по углам M Edu: основание, плечо, стрела	manipulator.MoveToAngles(angles)	MoveAnglesMEdu Params:
			- m_positions (JointPositionsMEdu): массив с углами поворота всех суставов [рад]
			- m_velocities (JointVelocities): массив с скоростями поворота всех суставов [рад/с]
			- m_velocityFactor (float):
				#include "pm_sdk/types.hpp"
				
				MoveAnglesMEdu Params angles(
				
				JointPositionsMEdu(0.05, -0.35, -0.75), // povorot_osnovaniya, privod_plecha, privod_strely
				JointVelocities(0, 0, 0) // скорости [рад/с]
				);

Наименование	Описание	Команда	Параметры	Пример

			коэффициент скорости (0.0-1.0)
			- m_accelerationFactor (float): коэффициент ускорения (0.0-1.0)
				angles.m_velocityFactor = 0.1; // коэффициент скорости
				angles.m_accelerationFactor = 0.1; // коэффициент ускорения
				
				manipulator.MoveToAngles(angles);

MoveToCoordinates	Движение по координатам.
	В случае, если движение манипулятора не происходит и возникает ошибка расчетов траектории, указывайте более точные координаты
		manipulator.MoveToCoordinates (coords);	- position: позиция – объект Position(x, y, z);
			- orientation: ориентация – объект Orientation(x, y, z, w);
			- velocity_scaling_factor: коэффициент скорости;
			- acceleration_scaling_factor: коэффициент ускорения
				#include "pm_sdk/types.hpp"
				
				MoveCoordinatesParams coords(
				Position(0.32, -0.004, 0.25), // x, y, z
				Orientation(0, 0, 0, 1.0), // x, y, z, w (кватернион)
				0.1, // velocity_scaling_factor
				0.1 // acceleration_scaling_factor
				);
				
				manipulator.MoveToCoordinates(coords);

MoveArc	Движение по дуге	manipulator.MoveArc(arc)	- target (MoveArcParams::Target): целевая точка (содержит	#include "pm_sdk/types.hpp" // Создаем целевую точку и центр дуги

Наименование	Описание	Команда	Параметры	Пример

			Position и Orientation);
			- center_arc (MoveArcParams::CenterArc): точка центра дуги (содержит Position и Orientation);
			- step (float): шаг между точками дуги;
			- count_point_arc (int): количество точек для аппроксимации дуги;
			- max_velocity_scaling_factor (float): максимальный масштаб скорости;
			- max_acceleration_scaling_factor (float): максимальный масштаб ускорения
				MoveArcParams::Target target(
				Position(0.25, -0.1, 0.25), // x, y, z целевой точки
				Orientation(0, 0, 0, 1) // ориентация
				);
				
				MoveArcParams::CenterArc center(
				Position(0.15, 0.0, 0.20), // x, y, z центра дуги
				Orientation(0, 0, 0, 1)
				);
				
				// Создаем параметры дуги
				MoveArcParams arc(
				target, // целевая точка
				center, // центр дуги
				0.05, // step - шаг между точками
				50, // count_point_arc - количество точек
				0.5, // max_velocity_scaling_factor

Наименование	Описание	Команда	Параметры	Пример
				0.5 // max_acceleration_scaling_factor); manipulator.MoveArc(arc);
Работа с насадками				
NozzlePower	Подача питания на насадки перед их активацией	manipulator.NozzlePower(true)	true – включить питание	manipulator.NozzlePower (true);
GripperAttachment	Механический захват. Вызов через манипулятор – насадки описаны отдельными классами	GripperAttachment gripper(manipulator); gripper.Activate(rotation, gripper); gripper.Deactivate ();	Activate: - rotation – угол поворота насадки, градусы; - gripper – позиция/угол захвата, градусы	GripperAttachment gripper(manipulator); gripper.Activate(20, 10); // rotation=20, gripper=10 (закрыть) gripper.Deactivate(); // открыть
VacuumAttachment	Вакуумная насадка. Вызов через манипулятор – насадки описаны отдельными классами	VacuumAttachment vacuum(manipulator); vacuum.Activate(rotation); vacuum. Deactivate();	rotation – угол поворота насадки, градусы	VacuumAttachment vacuum(manipulator); vacuum.Activate(-40); // rotation=-40, включить вакуум vacuum.Deactivate(); // выключить
LaserAttachment	Лазерная насадка. Вызов через манипулятор –	LaserAttachment laser(manipulator, channel);	channel – номер канала	// Лазер (канал 1) LaserAttachment laser(manipulator, 1);

Наименование	Описание	Команда	Параметры	Пример

	насадки описаны отдельными классами	laser.Activate();
		laser.Deactivate();
				laser.Activate(); // включить
				laser.Deactivate(); // выключить

Стриминг				

SetServoTwistMode / SetServoPoseMode / SetServoJointJog Mode	Переключение режима стриминга: скорости (TWIST), поза (POSE), суставы (JOINT_JOG). Режим выбирается до отправки стриминговых команд	manipulator.SetServoTwistMode(); // Для скорости
		
		manipulator.SetServoPoseMode(); // Для позы
		
		manipulator.SetServoJointJogMode(); // Для суставов
			–	// Режимы
				manipulator.SetServoTwistMode(); // Для скорости
				manipulator.SetServoPoseMode(); // Для позы
				manipulator.SetServoJointJogMode(); // Для суставов

StreamCartesianVelocities	Стриминг скоростей	manipulator.StreamCartesianVelocities (linear, angular);	- linear: линейная скорость (x, y, z);
			- angular: угловая скорость (rx, ry, rz)
				// Включаем режим TWIST
				manipulator.SetServoTwistMode();
				
				LinearVelocity linear{0.02, 0, 0};
				AngularVelocity angular{0, 0, 0.01};
				manipulator.StreamCartesianVelocities(linear, angular);

StreamCoordinates	Стриминг позы	manipulator.StreamCoordinates(pos, orient);	- position: позиция (x, y, z).	// Включаем режим POSE

Наименование	Описание	Команда	Параметры	Пример

			- orientation: ориентация (x, y, z, w)	manipulator.SetServoPoseMode();
				
				Position pos(0.27, 0.0, 0.15);
				Orientation orient(0, 0, 0, 1);
				manipulator.StreamCoordinates(pos, orient);

Программы				
RunProgram	Запуск готовой программы на манипуляторе	manipulator.RunProgram(name);	name – имя программы	manipulator.RunProgram("edum/default");

RunProgramJSON	Запуск JSON-сценария	manipulator.RunProgramJSON(name, program_json)	- name – имя программы;
			- program_json – nlohmann::json
				nlohmann::json program_json = {
				{"Root", {
				{
				{"Move", {
				{"content", {
				{
				{"Point", {
				
				{"positions", {0.3, -0.3, -0.4}},
				{"time", 0.5}
				}}
				}
				}},
				{"type", "Simple"}

Наименование	Описание	Команда	Параметры	Пример

				}}
				}
				}}
				};
				manipulator.RunProgramJSON("program_1", program_json);

RunPythonProgram	Выполнение Python-кода на манипуляторе из C++ SDK	PythonProgram py(code); auto result = manipulator.RunPythonProgram(py)	- code – строка Python-кода; - env_name – имя venv (опц.); - python_version – версия Python (опц.); - requirements – зависимости (опц.)	PythonProgram py("print('Hello from C++!')"); auto result = manipulator.RunPythonProgram(py);
Остановка движений				
StopMovement	Синхронная остановка движения	manipulator.StopMovement()	–	manipulator.StopMovement();
Состояния и данные				
GetJointStates	Синхронное получение данных. Текущие углы суставов	auto joint_states = manipulator. GetJointStates();	–	auto joint_states = manipulator. GetJointStates(); std::cout << "Углы суставов: " << joint_states.dump(2) << std::endl;

Наименование	Описание	Команда	Параметры	Пример
GetHomePosition	Синхронное получение данных. Домашняя позиция	auto home_pos = manipulator. GetHomePosition();	–	auto home_pos = manipulator. GetHomePosition(); std::cout << "Домашняя позиция: " << home_pos.dump(2) << std::endl;
GetCartesian Coordinate	Синхронное получение данных. Текущие координаты X, Y, Z	auto coords = manipulator.GetCartesianCoordinates();	–	auto coords = manipulator.GetCartesian Coordinates(); std::cout << "Координаты: " << coords.dump(2) << std::endl;
GetManipulatorInfo	Синхронное получение данных. Информация о манипуляторе	auto manip_info = manipulator.GetManipulatorInfo();	–	auto manip_info = manipulator.GetManipulatorInfo(); std::cout << "Информация: " << manip_info.dump(2) << std::endl;
GetCoordinateLimits	Синхронное получение данных. Ограничения координат	auto limits = manipulator.GetCoordinateLimits();	–	auto limits = manipulator.GetCoordinateLimits(); std::cout << "Ограничения: " << limits.dump(2) << std::endl;
SubscribeCartesian Coordinates	Подписка на обновления.	manipulator. SubscribeCartesian Coordinates([]	–	manipulator. SubscribeCartesian

Наименование	Описание	Команда	Параметры	Пример
	Подписка на координаты	(const nlohmann::json& data) // Отписка manipulator.UnsubscribeCartesianCoordinates();		Coordinates([] (const nlohmann::json& data) { std::cout << "Координаты обновлены: " << data.dump() << std::endl; });
SubscribeJointStates	Подписка на обновления. Подписка на состояние суставов	manipulator. SubscribeJointStates([](const nlohmann::json& data) // Отписка manipulator.UnsubscribeJointStates();	–	manipulator. SubscribeJointStates([](const nlohmann::json& data) { std::cout << "Углы суставов обновлены: " << data.dump() << std::endl; });
SubscribeGPIOStates	Подписка на обновления. Подписка на состояние GPIO	manipulator. Subscribe GPIOStates([] (const nlohmann::json& data) // Отписка manipulator. UnsubscribeGPIOStates();	–	manipulator.SubscribeGPIOStates([](const nlohmann::json& data) { std::cout << "GPIO обновлено: " << data.dump() << std::endl; });
SubscribeCoordinateLimits	Подписка на обновления. Подписка на	manipulator. Subscribe CoordinateLimits([] (const	–	manipulator. SubscribeCoordinateLimits([](const nlohmann::json& data) {

Наименование	Описание	Команда	Параметры	Пример
	ограничения координат	nlohmann::json& data) // Отписка manipulator. Unsubscribe CoordinateLimits();		std::cout << "Ограничения: " << data.dump() << std::endl; });
SubscribeManipulatorInfo	Подписка на обновления. Подписка на информацию о манипуляторе	manipulator. Subscribe ManipulatorInfo([](const nlohmann::json& data) // Отписка manipulator. Unsubscribe ManipulatorInfo();	–	manipulator.SubscribeManipulatorInfo([](const nlohmann::json& data) { std::cout << "Информация о манипуляторе: " << data.dump() << std::endl; });
Важно! Для асинхронного получения данных программа должна продолжать работать (например, иметь цикл или ожидание), чтобы успевать обрабатывать входящие сообщения. Подписки работают в фоновом потоке и вызывают ваш callback при каждом обновлении данных на манипуляторе				
Конвейерная лента (MGbot) – управление конвейерной лентой и связанными устройствами				
MGBotSetMotorSpeed	Установить скорость мотора (0..100)	manipulator.MGBotSetMotorSpeed(speed);	speed – задает скорость мотора в диапазоне 0..100	manipulator.MGBotSetMotorSpeed(10);
MGBotSetServoAngle	Повернуть сервопривод (градусы)	manipulator.MGBotSetServoAngle(45);	angle – поворачивает сервопривод (градусы)	manipulator.MGBotSetServoAngle(45);

Наименование	Описание	Команда	Параметры	Пример
manipulator.MGBotSetLedColor	Установить цвет светодиода (0..255)	manipulator.MGBotSetLedColor(r, g, b);	r, g, b – задает цвет светодиодов (каждый канал 0..255)	manipulator.MGBotSetLedColor(255, 0, 0);
MGBotDisplayText	Показать текст на дисплее	manipulator.MGBotDisplayText (text);	text – показывает текст на дисплее конвейера (utf-8).	manipulator.MGBotDisplayText("Hello");
manipulator.MGBotSetBuzTone	Звуковой сигнал (1..15)	manipulator.MGBotSetBuzTone(level);	Level – подает звуковой сигнал в диапазоне 1..15.	manipulator.MGBotSetBuzTone(10);
MGBotGetSensorsData	Включение датчиков и получение данных	manipulator.MGBotGetSensorsData(true);	enabled – получить данные с датчиков; true – данные возвращаются в виде JSON	auto sensor_data = manipulator.MGBotGetSensorsData(true); // Поля: "DistanceSensor", "ColorSensor" (R,G,B,Prox)

Наименование	Описание	Команда	Параметры	Пример
Инструкция по подключению: 1) Подключите кабель USB к разъему на конвейере и к разъему USB на задней панели манипулятора. 2) Подключите кабель питания в разъем рядом с USB-портом на конвейере. При подаче питания моторы кратковременно инициализируются и могут сделать короткое движение. 3) Проверьте индикацию светодиодов на плате конвейера: - LED1 (синий): горит, когда есть питание 5 В через USB-порт. - LED2 (красный): горит, когда есть питание от внешнего источника 8–30 В. - LED3 (розовый): режим выхода или ШИМ. **Важные предупреждения и рекомендации:** - Разъем Type-C: контроллер чувствителен к ориентации кабеля USB-C. Если устройство не определяется – попробуйте аккуратно перевернуть разъем кабеля и повторно подключить. - Проблемы со стартом/остановкой ленты: если лента не начинает движение или не останавливается, аккуратно приподнимите стеклянную крышку и нажмите кнопку RESET слева от платы конвейера. - Подключение по Wi-Fi: при отправке команд через Wi-Fi убедитесь в стабильном соединении и достаточной пропускной способности сети – нестабильный Wi-Fi может приводить к потерям команд и задержкам. - Подсветка дисплея: при подключении USB-кабеля к манипулятору возможна потеря яркости дисплея ленты – это нормальное поведение. - Не работают сенсорные датчики: в случае отсутствия показаний сенсоров попробуйте перезапустить манипулятор. Если проблема сохраняется – проверьте подключение кабелей или обратитесь в техподдержку ПРОМОБОТ.				
Управление GPIO				
GPIO (General Purpose Input/Output) позволяет управлять цифровыми входами и выходами манипулятора. Это полезно для взаимодействия с внешними устройствами, датчиками, светодиодами и другими периферийными компонентами				
SetGPIO	Синхронная запись в GPIO	manipulator.SetGPIO(name, value);	- name: имя GPIO-пина	// Включение светодиода

Наименование	Описание	Команда	Параметры	Пример

GetGPIOStates	Чтение состояния GPIO	auto gpio_states = manipulator.GetGPIOStates();	- value: значение для записи (0.0 или 1.0)	manipulator.SetGPIO("/dev/gpiochip4/e1_pin", 1.0f);
				
				// Чтение состояния
				auto gpio = manipulator.GetGPIOStates();
				std::cout << "GPIO: " << gpio.dump(2) << std::endl;

Получение данных из I2C				
Интерфейс I2C позволяет читать данные с датчиков и устройств, подключенных к манипулятору				

GetI2CStates	Получение всех I2C данных.
	Доступные устройства (в JSON):
	- commutator/sel_serv – выбор сервопривода;
	- commutator/en_serv – включение сервопривода;
	- commutator/en_fan – включение вентилятора;
	- commutator/fd_key – статус кнопки Freedrive;
		manipulator.GetI2CStates();	–	auto i2c = manipulator.GetI2CStates();
				int fd_key = i2c["commutator/fd_key"];
				if (fd_key < 1000) {
				std::cout << "Freedrive активен" << std::endl;
				
				Примечание – Значение fd_key = 4095 означает, что freedrive выключен

Наименование	Описание	Команда	Параметры	Пример
	- commutator/adc_temp – температура с АЦП			
Воспроизведение аудио				
PlayAudio	Синхронное воспроизведение аудиофайла на манипуляторе	manipulator.PlayAudio(file_name);	file_name – имя аудиофайла на манипуляторе	manipulator.PlayAudio("start.wav");
PlayAudioAsync	Асинхронное воспроизведение аудиофайла с callback	manipulator.PlayAudioAsync(file_name, callback)		manipulator.PlayAudioAsync("notification.mp3", [](CommandResult result) { if (result.GetState()) { ... } }); Для сборки демо: cd demo mkdir build cd build cmake .. make ./sdk_demo

8.4.5 Заключение

Если возникли проблемы – используйте отладочный вывод:

```
#include "pm_sdk/debug_logger.hpp"
// Включение всех логов
pb_sdk::DebugLogger::SetEnable(pb_sdk::LogEntity::ALL, true);
```

И обращайтесь в техническую поддержку ПРОМОБОТ с подробными логами.

Рекомендация: для повышения надежности исполнения все вызовы команд следует оборачивать в конструкцию try/catch, чтобы корректно обрабатывать возможные ошибки.