*Ассоциация участников технологических кружков*

УТВЕРЖДАЮ
Президент Ассоциации

_____ А.И. Федосеев

Дополнительная общеразвивающая образовательная программа

# Профиль: «Водные робототехнические системы»

направление подготовки: научно-техническое
общая трудоемкость – 54 часа или 64 часа (при наличии специального оборудования)
образовательные технологии – предметно-ориентированные

**Разработчик:**
команда разработчиков профиля
«Водные робототехнические системы» НТО
ООО «Центр робототехники»

Владивосток, 2021 г.

**Пояснительная записка**

Подводным роботам уже больше 60 лет. Но именно сейчас, с развитием систем навигации, подводная робототехника готовит самые интересные вызовы и предоставляет самые интересные возможности. Роботам приходится иметь дело с экстремальными условиями эксплуатации, при этом они становятся всё меньше — и всё умнее.
При этом важное место в данном направлении занимают автономные необитаемые подводные аппараты. Такие подводные аппараты оснащены источником энергии, бортовой системой управления, движительно-рулевой системой и средствами для ввода программы-задания и считывания результатов измерений. Индустрия АНПА охватывает немногим более 30 лет. За это время необитаемые аппараты прошли путь от ненадежных прототипов, разрабатываемых для исследовательских и военных целей, до необходимых ежедневных инструментов, без которых развитие многих направлений подводной промышленности, включая добычу нефти и газа, было бы затруднительно. Диапазон применений АНПА широк: от выполнения обзорно-поисковых операций на больших площадях вблизи морского дна до высокоточного инспектирования трубопроводов и обследования водозаполненных тоннелей. Военные ведомства проводят исследования с целью разработки АНПА для совершения длительных интеллектуальных подводных миссий, связанных с проведением разведывательных операций и обеспечением безопасности заданных объектов, выполнением противоминных операций. В настоящее время накоплен обширный опыт по разработке и использованию АНПА, сформировалось представление о том, как целесообразно с их помощью решать те или иные задачи. Всем этим вопросам, а также многим другим в области подводной робототехники посвящен данный курс.
Ученикам, в ходе работы на курсе предстоит программировать подводных роботов под различные актуальные задачи.

Цель курса: познакомить учащихся с основными типами необитаемых подводных аппаратов, с наиболее важными их системами, а также сформировать знания в областях их использования в современных морских научных исследованиях и инженерно-технических работах.

Задачи курса:

- познакомить учащихся с разнообразием автономных необитаемых подводных аппаратов;
- сформировать у учащихся навыки по программированию микроконтроллеров;
- сформировать у учащихся навыки по программированию автономных необитаемых подводных аппаратов (АНПА);
- познакомить учащихся с основами знаний по электронике и конструированию
- научить учащихся конструировать подводных роботов.

**Возраст детей:** 8-11 класс

**Количество часов**: 2 академических часа в неделю (1 занятие по 2 часа)

**Сроки реализации программы без спец. оборудования:** 54 часов (27 занятий), 6 месяцев

**Сроки программы со специальным оборудованием:** 64 часа (32 занятия) 8 месяцев

**Начальные требования к знаниям, умениям и навыкам, учащихся:**
Желательно, чтобы школьник обладал хотя бы одним из этих навыков:

- Проектирование электрических плат;
- Проектирование в конструкторских САПР;
- Навык работы с 3Д-принтером;
- Программирование микроконтроллеров;
- Знание основ теории автоматического управления;
- Знание и умения использовать OpenCV для программирования компьютерного зрения;
- Навык работы с формообразующим инструментов;
- Навык пайки.

Курс состоит из 5 основных разделов и одного дополнительного:

- Раздел 1. Программирование микроконтроллеров
- Раздел 2. Программирование АНПА
- Раздел 3. Конструирование
- Раздел 4. Электроника
- Раздел 5. Комплексные задачи
- Дополнительный. Раздел 6. Задачи с дополнительным оборудованием

**Планируемые результаты программы**

Предметные результаты обучения:

- иметь представление о водных робототехнических системах и их возможностях в современном обществе;
- иметь представление о перспективах развития автономных необитаемых подводных аппаратов;
- использовать различные средства для поиска информации о возможностях, направлениях развития автономных необитаемых подводных аппаратов;
- уметь разрабатывать электрические схемы;
- иметь навыки программирования микроконтроллеров;
- иметь навыки распознавания объектов и программирования регуляторов;
- знать основы конструирования в САПР;
- иметь навыки разработки чертежей с использованием САПР;
- иметь навыки проектирования электронных устройств;

Метапредметные результаты обучения:

- формировать универсальные учебные действия (познавательные, регулятивные, коммуникационные), обобщенные способы информационной деятельности при использовании информационных технологий, в том числе при программировании микроконтроллеров и автономных необитаемых подводных аппаратов;

- развить познавательные интересы, интеллектуальные и творческие способности путем освоения и использования методов проектирования электронных устройств;
- приобрести опыт программирования автономных необитаемых подводных аппаратов в индивидуальной, групповой и коллективной учебно-познавательной деятельности.

Личностные результаты обучения:

- личностное и предпрофессиональное самоопределение через познавательную мотивацию к получению профессий, связанных с проектированием автономных необитаемых подводных аппаратов и через познавательный интерес к достижениям в области водных робототехнических систем;
- построение дальнейшей индивидуальной образовательной траектории через получение представления о перспективах развития водных робототехнических систем;
- осознание стратегической важности для государства, общества и для своего будущего развития водных робототехнических систем.

**Содержание программы**

**Учебно-тематический план**

| № | Тема | Количество часов | Лекции | Практика | Контроль |
|---|---|---|---|---|---|
| Раздел 1. Программирование микроконтроллеров | | | | | |
| 1 | Проект электрической схемы управления коллекторным мотором и светодиодной лентой | 4 | 1 | 3 | Проект |
| 2 | Проект программирования микроконтроллера Arduino UNO, к которой подключен 7-сегментный дисплей, две тактовые кнопки и потенциометр | 2 | 0,5 | 1,5 | Проект |
| 3 | Проект программирования микроконтроллера Arduino UNO, к которому подключены клавиатура и символьный дисплей | 2 | 0,5 | 1,5 | Проект |
| | Всего по разделу | 8 | 2 | 6 | |
| Раздел 2. Программирование АНПА | | | | | |
| 5 | Программирование подводного робота. Обручи и полоски. | 4 | 1 | 3 | Проект |
| 6 | Программирование подводного робота. Запуск торпеды. | 4 | 1 | 3 | Проект |
| 7 | Программирование подводного робота. Перемещение объекта | 4 | 1 | 3 | Проект |
| | Всего по разделу | 12 | 3 | 9 | |
| Раздел 3. Конструирование | | | | | |
| 8 | Проектирование герметичного корпуса. Уплотнительное кольцо. | 4 | 1 | 3 | Проект |
| 9 | Проектирование герметичного корпуса. Резьбовое соединение. | 4 | 1 | 3 | Проект |
| 10 | Проектирование герметичного корпуса. | 4 | 1 | 3 | Проект |

| | | | | | |
|---|---|---|---|---|---|
| | Всего | 12 | 3 | 9 | |
| Раздел 4. Электроника | | | | | |
| 11 | Проект электрической схемы управления коллекторным мотором. Скорость мотора за пределами (-10000; 10000) | 2 | 0,5 | 1,5 | Проект |
| 12 | Проект электрической схемы управления коллекторным мотором. Скорость мотора в пределах (0; 10000) | 2 | 0,5 | 1,5 | Проект |
| 13 | Схема электрическая структурная к задаче 12 | 2 | 0,5 | 1,5 | Проект |
| 14 | Электронное пианино | 2 | 0,5 | 1,5 | Проект |
| 15 | Схема электрическая структурная к задаче 14 | 2 | 0,5 | 1,5 | Проект |
| | Всего | 10 | 2,5 | 7,5 | |
| Раздел 5. Комплексные задачи | | | | | |
| 16 | Хакатон. Разработка подводного LED-светильника | 8 | 2 | 6 | Проект |
| 17 | Сборка движителя | 4 | 1 | 3 | Проект |
| | Всего | 12 | 3 | 9 | |
| Раздел 6. Задачи со специальным оборудованием | | | | | |
| 18 | Конструирование АНПА с использованием набора ElementaryROV | 32 | 8 | 24 | Соревнования |
| 19 | Конструирование АНПА с использованием набора MiddleROV | 32 | 8 | 24 | Соревнования |
| | Всего | 64 | 16 | 48 | |

**Содержание программы**

| Наименование, разделов, тем | Содержание | Виды учебных занятий, учебных работ |
|---|---|---|
| Раздел 1 Программирование микроконтроллеров (8 ч) | | |
| Проект электрической схемы управления коллекторным мотором и светодиодной лентой | Создать проект электрической схемы управления коллекторным мотором и светодиодной лентой NeoPixel Ring 12. Также необходимо запрограммировать | Лекция<br>Задача 1-2 (тип 2) |

| | | |
|---|---|---|
| | Arduino UNO, чтобы выполнялось заданные требования. | |
| Проект программирования микроконтроллера Arduino UNO, к которой подключен 7-сегментный дисплей, две тактовые кнопки и потенциометр | В схеме имеется микроконтроллер Arduino UNO, к которому подключен 7-сегментный дисплей через микросхему CD4511, а также подключены две тактовые кнопки и потенциометр. Необходимо запрограммировать Arduino UNO, чтобы выполнялись заданные требования. | Лекция<br>Задача 3 (тип 2) |
| Проект программирования микроконтроллера Arduino UNO, к которому подключены клавиатура и символьный дисплей | В схеме имеется микроконтроллер Arduino UNO, к которому подключены клавиатура 4×4 и символьный дисплей 16×2. Необходимо запрограммировать Arduino UNO, чтобы выполнялись заданные требования. | Лекция<br>Задача 4 (тип 2) |
| Раздел 2. Программирование АНПА (12 часов) | | |
| Программирование подводного робота. Обручи и полоски. | В симуляторе MUR IDE выполнить задачу: Пройти сквозь обруч 1. Пройти сквозь обруч 2. Сбросить маркер в корзину. Всплыть в обруче, который расположен над корзиной. | Лекция<br>Задача 5 (тип 2) |
| Программирование подводного робота. Запуск торпеды. | • В виртуальной среде MUR IDE необходимо последовательно запустить по одной торпеде в каждую из двух мишеней, цвет которых не совпадает с цветом указателя на старте, после этого, робот должен проплыть через мишень того же цвета, что и указатель. Затем проследовать по направляющей полоске голубого цвета. Над круглой голубой платформой расположен обруч, в пределах которого нужно всплыть на поверхность. | Лекция<br>Задача 6 (тип 2) |
| Программирование подводного робота. Перемещение объекта | В виртуальной среде MUR IDE необходимо определить форму навигационной метки для определение дальнейшего маршрута. Затем нужно пройти по направлению одного из трёх указателей к платформе с кубом. Данный куб нужно подобрать и положить | Лекция<br>Задача 7 (тип 2) |

| | | |
|---|---|---|
| | в синюю корзину после чего пройти к платформе по направлению указателя, цвет которого совпадает с полоской. Над данной платформой располагается обруч, в пределах которого нужно всплыть. | |
| Раздел 3 Конструирование (12 часов) | | |
| Проектирование герметичного корпуса. Уплотнительное кольцо. | Необходимо спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. Для герметизации необходимо использовать уплотнительное кольцо сечением 2,5 мм. Толщина стенок и дна колбы должна быть от 3 до 5 мм. | Лекция Задача 8 (тип 1 или тип 2) |
| Проектирование герметичного корпуса. Резьбовое соединение. | Необходимо спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. При этом крышка и колба должны соединяться с применением резьбового соединения. | Лекция Задача 9 (тип 1 или тип 2) |
| Проектирование герметичного корпуса. Система крепления. | Необходимо спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. Он будет использоваться под водой. Дополнительные баллы можно получить за спроектированную систему крепления крышки к корпусу (система крепления: отверстия, крепёжные элементы, дополнительные детали). При проектировании система должна считаться с законами физики и механики. | Лекция Задача 10 (тип 1 или тип 2) |
| Раздел 4. Электроника (10 часов) | | |
| Проект электрической схемы управления коллекторным мотором. Скорость мотора за пределами (-10000; 10000) | Создать проект электрической схемы управления коллекторным мотором. Управление скоростью вращения мотора должно осуществляться при помощи потенциометра, а изменение направления вращения - ползунковым переключателем. Скорость мотора должна выходить за пределы (-10000; 10000) | Лекция Задача 11 (тип 2) |

| | | |
|---|---|---|
| Проект электрической схемы управления коллекторным мотором. Скорость мотора в пределах (0; 10000) | Создать проект электрической схемы управления коллекторным мотором. Управление скоростью вращения мотора должно быть реализовано при помощи потенциометра без изменения направления вращения мотора. Диапазон скорости мотора не должен выходить за пределы (0; 10000) | Лекция<br>Задача 12 (тип 2) |
| Схема электрическая структурная к задаче 12 | Необходимо разработать Схему электрическую структурную для задачи 12. | Лекция<br>Задача 13 (тип 1) |
| Электронное пианино | Необходимо, используя online-сервис Tinkercad создать проект электрического пианино на основе заданных требований. | Лекция<br>Задача 14 (тип 2) |
| Схема электрическая структурная к задаче 14 | Необходимо разработать Схему электрическую структурную для задачи 14. | Лекция<br>Задача 15 (тип 1) |
| Раздел 5. Комплексные задачи (12 часов) | | |
| Хакатон. Разработка подводного LED-светильника | В рамках хакатона участникам необходимо разработать, управляемый с ПК подводный LED-светильник. Готовое устройство должно подключаться к ПК по USB и источнику питания 12V. | Лекция<br>Задача 16 (тип 3) |
| Сборка движителя | В рамках данной задачи учащимся необходимо собрать из комплектующих движитель OpenThruster 150. | Лекция<br>Задача 17 (тип 3) |
| Раздел 6. Задачи со специальным оборудованием (32 часа) | | |
| Конструирование АНПА с использованием набора ElementaryROV | Занятие 1. Введение.<br>Техника безопасности и общие правила. Обзор набора ElementaryROV. Электромонтаж проводов посредством пайки. Задания для практики | Лекция<br>Практика |
| | Занятие 2. Электромонтаж.<br>Рассмотрение инструментов (повторение). Техника безопасности (повторение). Электромонтаж проводов посредством пайки (повторение) Рассмотрение других видов соединений Выполнение практических заданий | Лекция<br>Практика |

| | | |
|---|---|---|
| | Занятие 3. Сборка пульта управления. Рассмотрение пульта управления. Сборка пульта управления | Лекция Практика |
| | Занятие 4. Работа с пультом и движителями. Подключение пульта управления к ПК и загрузка прошивки на микроконтроллер Подключение электроники и проверка ее работоспособности. Определение направления работы движителя «методом салфетки» | Лекция Практика |
| | Занятие 5. Сборка рамы. Обзор деталей и инструментов. Сборка рамы. Установка блоков плавучести | Лекция Практика |
| | Занятие 6-7. Изготовление макетов Изготовление макетов: Маркер, «Морская звезда», Y-образный штырь, Гнездо для штыря, Крюк с сеткой, Колония губок, Пластиковый мусор | Лекция Практика |
| | Занятие 8-9. Полезная нагрузка. Знакомство с Гидравлическим и Электромеханическим захватами Поиск общих черт между созданными макетами для проектирование наиболее универсального инструмента. | Лекция Практика |
| | Занятие 10. Балластировка аппарата Балластировка необходима для достижения аппаратом в воде нейтральной плавучести (робот не всплывает и не тонет в толще воды) или небольшой положительной (робот медленно всплывает). | Лекция Практика |
| | Занятие 11.Отладка робота в бассейне. Прошивка и изменение кода. Настройка управления с пульта (стиков, кнопок). Настройка горизонтальных моторов. | Лекция Практика |
| | Занятие 12. Брифинг и работа над презентацией. | Лекция Практика |
| | Занятие 13-14. Тренировка Подготовка включает в себя тренировку демонстрации аппарата и подготовка к презентации перед судьями. | Лекция Практика |
| | Занятие 15. Соревнования Соревнования состоят из трех частей: | Практика |

| | | |
|---|---|---|
| | Изготовление командного листа;<br>Презентация перед жюри;<br>Выполнение подводных заданий в бассейне. | |
| | Занятие 16. Рефлексия<br>Обмен впечатлениями, обсуждение и анализ проделанной работы, оценивание своих достижений | Практика |
| Конструирование АНПА с использованием набора MiddleROV | Занятие 1. Введение<br>Виды подводных аппаратов и их применение. Обзор набора MiddleROV (комплектация, вариант сборки и электрическая схема). Оборудование, инструменты и расходные материалы необходимые для пайки. Техника безопасности. Виды соединения проводов | Лекция<br>Практика |
| | Занятие 2. Изготовление кабель-троса<br>Обзор набора для изготовления кабель-троса. Изготовление кабеля<br>Пайка разъёма. Герметизация пенетратора | Лекция<br>Практика |
| | Занятие 3. Проверка контроллера, шилда и связи<br>Обучение работе с мультиметром. Проверка контроллера, шилда и проверка связи. Дополнительное задание. | Лекция<br>Практика |
| | Занятие 4. Поочерёдная проверка полезной нагрузки<br>Виды моторов. Разбор переменных в прошивке и настройка пинов. Проверка движителей. Проверка захвата. Проверка камеры. | Лекция<br>Практика |
| | Занятие 5. Пайка блока электроники и прошивка пульта.<br>Изучение принципов работы моторов. Пайка движителей и захвата. Прошивка пульта и блока электроники. | Лекция<br>Практика |
| | Занятие 6. Сборка блока электроники. Понятие герметизации. Сборка блока электроники. | Лекция<br>Практика |

| | | |
|---|---|---|
| | Занятие 7-8. Проектирование конструкции.<br>Установка CAD. Создание эскиза рамы. Разработка полезной нагрузки. Разработка модели использования. | Лекция<br>Практика |
| | Занятие 9. Проектирование конструкции. Проектируем\изготавливаем полезную нагрузку. Тестируем полезную нагрузку Завершаем проектирование рамы. Создаём раскрой листового материала. | Лекция<br>Практика |
| | Занятие 10. Изготовление полезной нагрузки.<br>Изготовление макетов: Маркер, «Морская звезда», Y-образный штырь, Гнездо для штыря, Крюк с сеткой | Лекция<br>Практика |
| | Занятие 11. Сборка аппарата<br>Обзор деталей и инструментов. Сборка рамы. Установка блоков плавучести | Лекция<br>Практика |
| | Занятие 12. Отладка на воздухе.<br>Перед погружением в воду убедиться в том, что все системы правильно работают и герметичны, и исправить, если это не так. | Лекция<br>Практика |
| | Занятие 13. Балластировка аппарата и кабеля.<br>Закрепить полезную нагрузку на аппарате. Отбалластировать аппарат. Отбалластировать кабель. Испытать полезную нагрузку на аппарате | Лекция<br>Практика |
| | Занятие 14-15. Тренировки в бассейне.<br>Инструктаж по технике безопасности. Определение пилота. Распределение ролей. Заплывы. План тренировки. | Лекция<br>Практика |
| | Занятие 16. Соревнования | Практика |

## Образовательные результаты

## Контрольно-измерительные материалы

### Раздел 1. Программирование микроконтроллеров

Требования к задаче 1

Запрограммировать Arduino UNO, чтобы выполнялось следующее требование:

- Когда на фоторезистор подается свет (доступно после нажатия на кнопку "Начать моделирование"), двигатель вращается в любую сторону с любой скоростью, все светодиоды в ленте светятся с любой интенсивностью и любым цветом.
- При выключении света, двигатель перестает вращаться, а светодиоды перестают светиться.

Требование к задаче 2:

Запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

- Когда потенциометр выставлен по центру - двигатель не вращается, лента не светится.
- При вращении потенциометра по часовой стрелки - двигатель вращается по часовой стрелки, светодиоды ленты загораются и гаснут по часовой стрелки.
- При вращении потенциометра против часовой стрелки - двигатель вращается против часовой стрелки, светодиоды ленты загораются и гаснут против часовой стрелки.
- При максимальном отклонении от центра потенциометра 1 "оборот" ленты должен составлять 1 секунду, обороты двигателя ~+\- 16000rpm.
- При минимальном отклонении от центра потенциометра 1 "оборот" ленты должен составлять 10 секунд.
- Угол поворота потенциометра пропорционально влияет на скорость и направление вращения двигателя и светодиодов ленты.

Требования к задаче 3

Необходимо запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

- Устройство должно представлять собой таймер обратного отсчёта. Значение счётчика должно отображаться на дисплее.
- С помощью потенциометра пользователь должен иметь возможность установить на дисплее число от 0 до 9 включительно.
- С помощью кнопки «Сброс» (пин 3), отображаемое число сбрасывается к значению, которое установлено через потенциометр.

- С помощью кнопки «Старт/Пауза» (пин 2) должна предоставляться возможность запускать таймер и ставить его на паузу.
- После того, как значение таймера достигает нуля, должен загораться встроенный светодиод на Arduino UNO. В иных случаях светодиод должен быть выключен.

Требования к задаче 4

Необходимо запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

- Устройство должно представлять собой калькулятор, с помощью которого можно производить простые вычисления (складывать, вычитать, делить и умножать числа).
- Функции кнопок:
  - цифры - числовой ввод
  - A - сложение (+)
  - B - вычитание (-)
  - C - деление (÷)
  - D - умножение (×)
  - * - сброс
  - # - равно (=)
- На дисплее должно отображаться первое число (над которым производится математическая операция); символ, обозначающий математическую операцию; второе число, участвующее в математической операции.
- После выполнения операции (при нажатии на кнопку «равно»), результат является «первым числом», над которым пользователь будет производить математическую операцию (до нажатия кнопки сброса).
- При нажатии на кнопку «сброс», очищаются оба числа и вид математической операции.

**Раздел 2. Программирование АНПА**

Оценка задачи 5 (тип 2)

В зависимости от выполнения задач в симуляторе MUR IDE, начисляются следующие баллы:

- 10 баллов. Робот прошел сквозь обруч 1. На обруч 1 направлена полоска 1.
- 10 баллов. Робот прошел сквозь обруч 2. На обруч 2 направлена полоска 2.
- 20 баллов. Робот сбросил маркер в корзину. На корзину направлена полоска 2.

- 10 баллов. Робот всплыл в обруче, который расположен над корзиной.

Оценка задачи 6 (тип 2)

В зависимости от выполненных за 5 минут задач в симуляторе MUR IDE, начисляются следующие баллы:

- 10 баллов. Робот определил цвет указателя, над которым стартовал.
- 20 баллов. Робот последовательно запустил по одной торпеде в каждую из двух мишеней, цвет которых не совпадал с цветом указателя.
- 20 баллов. Робот проплыл через мишень того же цвета, что и указатель.
- 10 баллов. Робот проследовал по направляющей полоске голубого цвета.
- 20 баллов. Над голубой платформой расположен обруч, в пределах которого робот всплыл на поверхность.

Оценка задачи 7

В зависимости от выполненных за 5 минут задач в симуляторе MUR IDE, начисляются следующие баллы:

- 10 баллов. Робот определил форму навигационной метки (треугольник, квадрат или круг).
- 20 баллов. Робот прошел по направлению одного из трёх указателей в зависимости от формы розового объекта.
- 20 баллов. Робот правильно определил направление к платформе с кубом, подобрал его и положил в синюю корзину, путь к которой указывала полоска.
- 10 баллов. Робот прошел к платформе по направлению одного из двух указателей, цвет которого совпадает с полоской.
- 10 баллов. Робот всплыл в пределах обруча, который располагается над платформой.

**Раздел 3. Конструирование**

Оценка задания 8 (тип 1 или тип2)

Критерии оценки расчетов - от 0 до 40 баллов:

- Диаметр уплотнительного кольца – 10 баллов;
- Ширина канавки в крышке - 10 баллов;
- Глубина канавки в крышке - 10 баллов;
- Внутренний диаметр колбы в месте герметизации - 10 баллов.

Оценка чертежа колбы - от 0 до 15 баллов:

- Должен содержать разрез детали.
- Должен быть сохранен в формате pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали.

Оценка чертежа крышки - от 0 до 15 баллов

- Должен содержать разрез детали.
- Должен быть сохранен в формате pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали.

Оценка задания 9 (тип 1 или тип 2)

Критерии оценки расчетов - от 0 до 40 баллов:

- Диаметр уплотнительного кольца - 10 баллов;
- Ширина канавки в крышке - 10 баллов;
- Диаметр для посадки кольца в крышке - 10 баллов;
- Внутренний диаметр колбы в месте герметизации - 10 баллов.

Оценка чертежа колбы - от 0 до 15 баллов:

- Должен содержать разрез детали.
- Должен быть сохранен в формате .pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

Оценка чертежа крышки - от 0 до 15 баллов:

- Должен содержать разрез детали.
- Должен быть сохранен в формате .pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

Оценка задания 10 (тип 1 или тип 2)

Критерии оценки расчетов - от 0 до 15 баллов:

- Диаметр уплотнительного кольца - 5 баллов
- Ширину канавки под кольцо - 5 баллов

- Глубину канавки под кольцо - 5 баллов

Оценка чертежа колбы от 0 до 15 баллов:

- Должен содержать разрез детали;
- Должен быть сохранен в формате .pdf;
- Может быть выполнен в любой программе;
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

Оценка чертежа крышки от 0 до 15 баллов:

- Должен содержать разрез детали;
- Должен быть сохранен в формате .pdf;
- Может быть выполнен в любой программе;
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

**Раздел 4. Электроника**

Оценка задания 11 (тип 2)

Необходимо, используя online-сервис Tinkercad создать проект электрической схемы управления коллекторным мотором. Основные требования:

- Можно использовать только следующие компоненты:
  - Двигатель постоянного тока
  - Потенциометр
  - Ползунковый переключатель
  - Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
  - Элементы питания
  - Логические микросхемы находящиеся в разделе "Логика"
  - Макетные платы
  - Таймер
- Управление скоростью вращения мотора должно осуществляться при помощи потенциометра, а изменение направления вращения - ползунковым переключателем.
- Диапазон скорости мотора должен выходить за пределы (-10000;  10000).
- Скорость вращения должна меняться с помощью широтно-импульсной модуляции.
- Изменение направления вращения должно осуществляться при помощи Н-моста, собранного на транзисторах.

- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Изменение скорости вращения должно происходить плавно.

Схемы, в которых используются неразрешенные компоненты, проверяться не должны. Учащийся при этом получает за него 0 баллов.

Баллы за задачу начисляются следующим образом.

- 100 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты.
- 90 баллов - схема работает правильно в соответствие с требованиями. Используются избыточные компоненты.
- 80 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты. Но скорость меняется только в пределах (-10000; 10000).
- 50 баллов - схема работает частично. Либо только меняется направление вращения, либо - скорость.
- 30 баллов - схема работает частично. Скорость меняется в пределах (-10000; 10000). Направление вращения не меняется.
- 0 баллов - схема не работает даже частично.

### Оценка задания 12 (тип 2)

Необходимо, используя online-сервис Tinkercad создать проект электрической схемы управления коллекторным мотором. Основные требования:

- Можно использовать только следующие компоненты:
  - Двигатель постоянного тока
  - Потенциометр
  - Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
  - Элементы питания
  - Логические микросхемы находящиеся в разделе "Логика"
  - Макетные платы
  - Таймер
- Необходимо реализовать управление скоростью вращения мотора при помощи потенциометра без изменения направления вращения мотора.
- Диапазон скорости мотора не должен выходить за пределы (0;  10000).
- Скорость вращения должна меняться с помощью широтно-импульсной модуляции.

- Управление вращением должно осуществляться при помощи транзистора.
- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Изменение скорости вращения должно происходить плавно.

Схемы, в которых будут использоваться неразрешенные компоненты, не будут проверяться. Уники при этом за задачу получают 0 баллов.

Баллы за задачу начисляются следующим образом.

- 100 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты.
- 90 баллов - схема работает правильно в соответствие с требованиями. Используются избыточные компоненты.
- 80 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты. Но скорость выходит за пределы (0;  10000).
- 50 баллов - схема работает частично. Либо только меняется направление вращения, либо - скорость.
- 30 баллов - схема работает частично. Скорость выходит за пределы (0;  10000). Направление вращения не меняется.
- 0 баллов - схема не работает даже частично.

Оценка задания 13 (тип 1)

Критерии оценки:

- Наличие рамки для заглавного листа в соответствии с ГОСТ 2.104-2006
- Рамка заполнена в соответствии с ГОСТ 2.104-2006
- Схема выполнена в САПР
- Линии связи должны состоять из горизонтальных и вертикальных отрезков и не имеют изломов и взаимных пересечений
- Все линии имеют подписи и все текстовые данные, относящиеся к линиям, ориентируют параллельно горизонтальным участкам соответствующих линий
- Все надписи выполнены шрифтом ГОСТ тип Б
- Схема выполнена на одном листе А4
- Схема занимает более 30% площади листа
- Все блоки схемы имеют названия, отражающие функциональное обозначение компонентов и их количество не избыточно
- Схема не имеет избыточных соединений между блоками

Кружковое движение НТО ЦРР

Оценка задания 14 (тип 2)

Необходимо, используя online-сервис Tinkercad, создать проект электрического пианино. Основные требования:

- Можно использовать только следующие компоненты:
  - Динамик или пищалка
  - Потенциометр
  - Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
  - Кнопки
  - Элементы питания
  - Логические микросхемы находящиеся в разделе "Логика"
  - Макетные платы
  - Таймер.
- Необходимо сделать примитивное пианино способное генерировать от 5 до 12 нот, управляемых тактовыми кнопками
- Частотный диапазон нот должен соответствовать первой октаве фортепиано.
- При нажатии на соответствующей кнопки, как и в нормальном пианино должен издаваться звук правильной частоты (допускается округление до целых значений Гц).
- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Когда кнопки не нажаты посторонних звуков быть не должно.

Пианино, в котором будут использоваться неразрешенные компоненты, не будут проверяться. Уники при этом за задачу получают 0 баллов.

Баллы за задачу начисляются следующим образом.

- 100 баллов - пианино работает правильно в соответствие с требованиями. Не используются избыточные компоненты.
- 90 баллов - пианино работает правильно в соответствие с требованиями. Используются избыточные компоненты.
- 80 баллов - пианино работает правильно в соответствие с требованиями. Не используются избыточные компоненты. Но частотный диапазон нот не всегда соответствует первой октаве фортепиано.
- 50 баллов - пианино работает частично.
- 0 баллов - пианино не работает даже частично.

Оценка задания 15 (тип 1)

Критерии оценки:

- Наличие рамки для заглавного листа в соответствии с ГОСТ 2.104-2006
- Рамка заполнена в соответствии с ГОСТ 2.104-2006
- Схема выполнена в САПР
- Линии связи должны состоять из горизонтальных и вертикальных отрезков и не имеют изломов и взаимных пересечений
- Все линии имеют подписи и все текстовые данные, относящиеся к линиям, ориентируют параллельно горизонтальным участкам соответствующих линий
- Все надписи выполнены шрифтом ГОСТ тип Б
- Схема выполнена на одном листе А4
- Схема занимает более 30% площади листа
- Все блоки схемы имеют названия, отражающие функциональное обозначение компонентов и их количество не избыточно
- Схема не имеет избыточных соединений между блоками

## Раздел 5. Комплексные задачи

Оценка задачи 16 (тип 3)

Результат оценивается по 3 критериям:

- Качество пайки и изготовления электроники (0, 1 или 2 балла)
- Качество заливки и внешний вид готового устройства (0, 1, или 2 балла)
- Качество написания программного кода. Функционал устройства (0, 1, 2 балла)

Оценка задачи 17 (тип 3)

## **Раздел 6. Задачи с дополнительным оборудованием.**

Оценка задача 18 (тип 4).

Результат оценивается в формате соревнования, которые состоят из трех частей:

- Изготовление командного листа – 20 баллов;
- Презентация перед жюри – 50 баллов;
- Выполнение подводных заданий в бассейне – 200 баллов.

## Материально-технические требования для реализации программы

Раздел 1. Программирование микроконтроллеров

- ПК с установленным браузером Chrome
- бесплатный онлайн симулятор Tinkercad: https://www.tinkercad.com/

Раздел 2. Программирование АНПА

- ПК
- бесплатный симулятор MUR IDE: https://murproject.com/documents/17/murIDE.exe

Раздел 3. Конструирование

Желательно:

- ПК
- Установленный конструкторский САПР (например, Inventor: https://www.autodesk.ru/products/inventor/overview)

Раздел 4. Электроника

- ПК с установленным браузером Chrome
- Бесплатный онлайн симулятор Tinkercad: https://www.tinkercad.com/
- бесплатный онлайн редактор схем, например www.lucidchart.com

Раздел 5. Комплексные задачи

Задача 16

Для выполнения данного задания каждой команде участников необходимо предоставить:

- Arduino NANO – 1 шт.;
- LED RGB лента 12В – 1 шт. (3 элемента);
- Полевые транзисторы N-канальные (IRF540N, STP9NK60Z и их аналоги) – 3 шт.;
- Провода (желательно 3-4 цвета) – по 50 см;
- Макетная плата 30х80 мм – 1 шт.;
- USB-miniUSB кабель – 1 шт.;
- Паяльное оборудование, припой, флюс, пинцет – 1 комплект;
- Кусачки, стриппер, ножовка по металлу, напильники, изолента, термоусадочная трубка – 1 комплект;
- Шприц 50 мл (или другая подходящая емкость для заливки) – 1 шт.;
- Эпоксидная смола – 50 мл.;
- Емкость для смешивания эпоксидной смолы (стаканчик пластиковый) – 1 шт.
- Защитные очки, перчатки, респиратор – 1 комплект;
- Ноутбук с установленным Arduino IDE – 1 шт.;
- Источник питания 9-12В – 1 шт.

Задача 17:

Для сборки подводного движителя OpenThruster 150 потребуется:

- Мотор (57x27.6 мм, 9800 об/мин, 12В 0.16А);
- воск (7 грамм);
- два провода (AWG 16/18, длина подбирается по необходимости);
- изолента;
- напечатанные детали корпуса: насадка, винт, крышка и корпус (PLA, заполнение 30%). Модели приведены по ссылке (https://github.com/murproject/OpenThruster150).
- Также понадобятся нож/ножницы, паяльник, припой, цианакрилатный (супер) клей, наждачная бумага или надфиль.

**Раздел 6. Задачи со специальным оборудованием**

- Для выполнения задачи 18 необходим набор ElementaryROV https://robocenter.net/goods/kit/elementaryrov/
- Для выполнения задачи 19 необходим набор MiddleROV https://robocenter.net/goods/kit/middlerov/

## ЛИТЕРАТУРА:

# Программирование микроконтроллеров

### Задача 1 (тип 2)

Задача выполняется в симуляторе Tinkercad

Дано

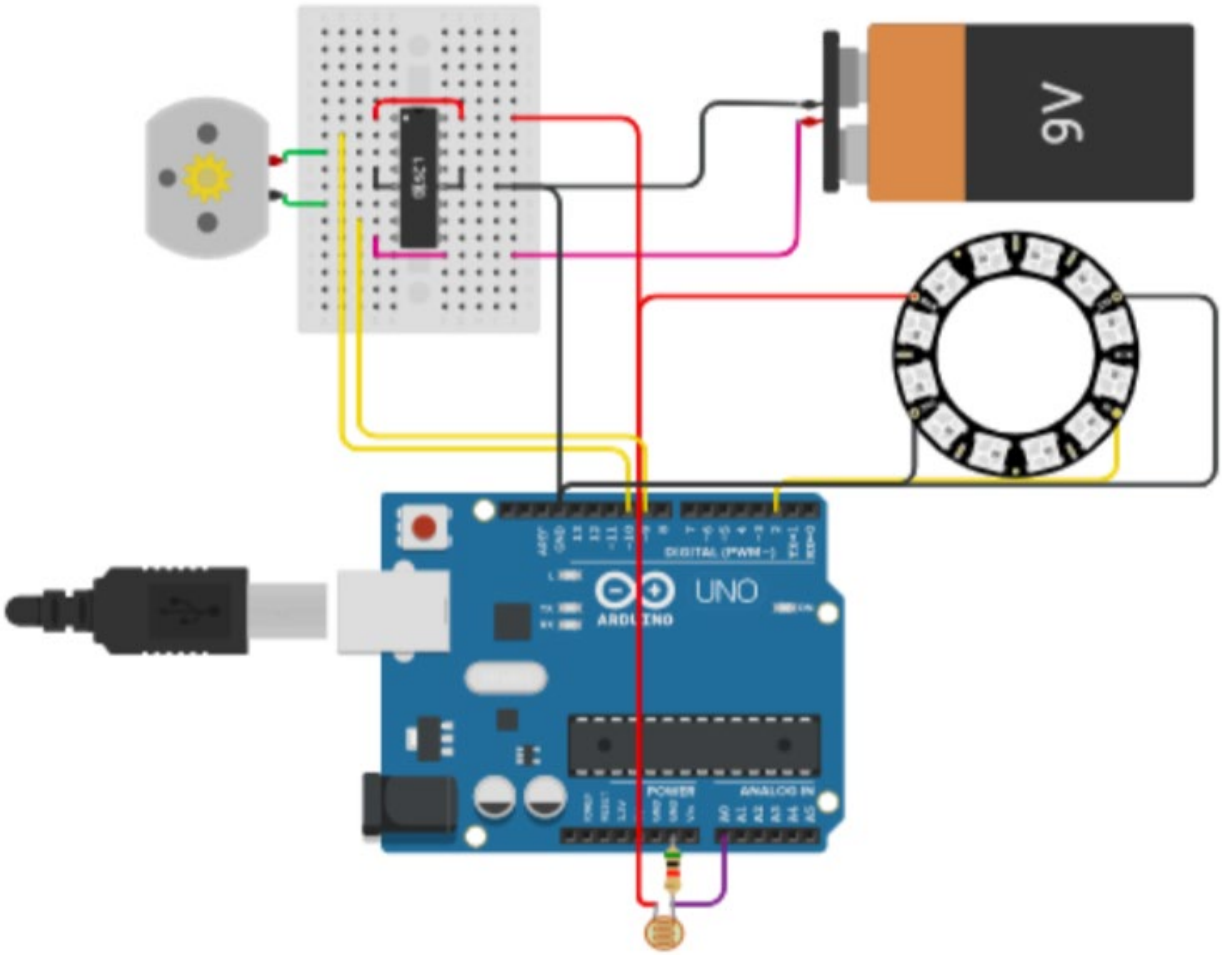
- Arduino UNO
- Батарея 9 В
- Двигатель постоянного тока
- NeoPixel Ring 12
- Макетная мини-плата
- Фоторезистор
- Резистор
- Электрический привод с Н-мостом

Необходимо, используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) создать проект электрической схемы управления коллекторным мотором и светодиодной лентой NeoPixel Ring 12. Также необходимо запрограммировать Arduino UNO, чтобы выполнялось следующее требование:

- Когда на фоторезистор подается свет (доступно после нажатия на кнопку "Начать моделирование"), двигатель вращается в любую сторону с любой скоростью, все светодиоды в ленте светятся с любой интенсивностью и любым цветом.
- При выключении света, двигатель перестает вращаться, а светодиоды перестают светиться.

## Решение задачи 1 (тип 2)

Строится схема в Thinkercad




Ссылка на проект в Thinkercad: https://www.tinkercad.com/things/3bRcg8ZG3G9-copy-of-lightdiod/editel?tenant=circuits?sharecode=kfp3agpUeXEtUMr70CHnrjp22qSbiVJhIIsySjAkjR4=

**Программный код для Arduino**

```
//Подключение библиотеки для ленты
#include <Adafruit_NeoPixel.h>

#define PIN 2   // input pin Neopixel is attached to

#define NUMPIXELS      12 // number of neopixels in strip
// Создание объекта для управления лентой
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB +
NEO_KHZ800);
```

```
void setup()
{
 //инициализация ленты, пинов для шима мотора (9, 10)
 pixels.begin();
 pinMode(10, OUTPUT);
 pinMode(9, OUTPUT);
 // АЦП для фоторезистора
 pinMode(A0, INPUT);

}

void loop()
{
 //считывание АЦП и преобразование в [0, 255]
 // цифры 28 и 974 подбираются (верхнее и нижнее значение, получаемые с АЦП)
 // в зависимости от установленного резистора
 int speed = map(analogRead(A0), 28, 974, 0, 255);

 for (int i = 0; i < 12; ++i){
  //устанавливаем все диоды в цвет пропорционально полученного значения с АЦП
       pixels.setPixelColor(i, pixels.Color(speed, speed, speed));
 }
 // рендер диодов
 pixels.show();
 // устанавливаем тягу на мотор
 analogWrite(9, speed);
}
```

## Задача 2 (тип 2)

Задача выполняется в симуляторе Tinkercad

Дано

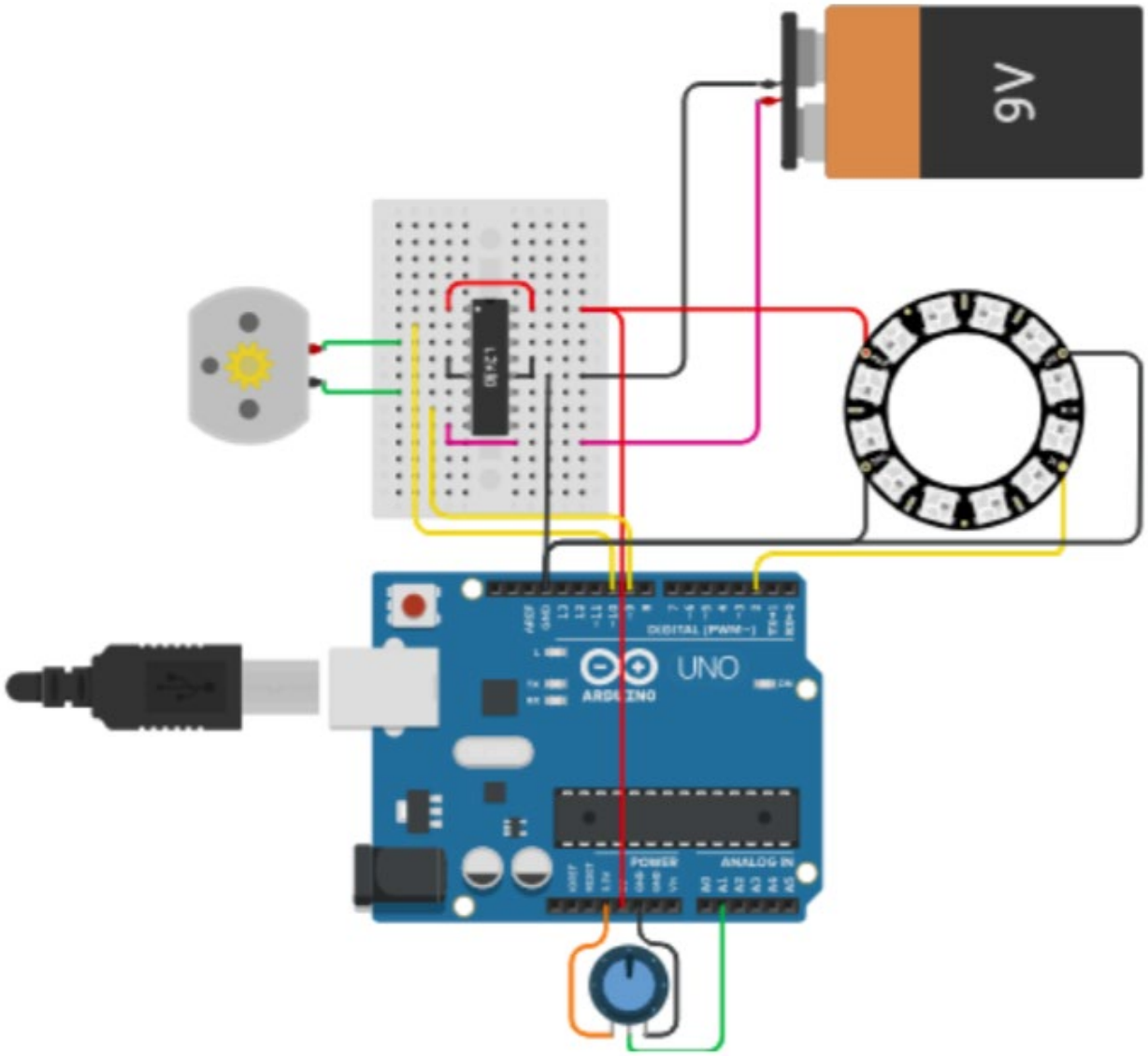
- Arduino UNO
- Батарея 9 В
- Двигатель постоянного тока
- NeoPixel Ring 12
- Макетная мини-плата
- Потенциометр
- Электрический привод с H-мостом

Необходимо, используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) создать проект электрической схемы управления коллекторным мотором и светодиодной лентой NeoPixel Ring 12. Также необходимо запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

- Когда потенциометр выставлен по центру - двигатель не вращается, лента не светится.
- При вращении потенциометра по часовой стрелки - двигатель вращается по часовой стрелки, светодиоды ленты загораются и гаснут по часовой стрелки.
- При вращении потенциометра против часовой стрелки - двигатель вращается против часовой стрелки, светодиоды ленты загораются и гаснут против часовой стрелки.
- При максимальном отклонении от центра потенциометра 1 "оборот" ленты должен составлять 1 секунду, обороты двигателя ~+\-16000rpm.
- При минимальном отклонении от центра потенциометра 1 "оборот" ленты должен составлять 10 секунд.
- Угол поворота потенциометра пропорционально влияет на скорость и направление вращения двигателя и светодиодов ленты.

## Решение задачи 2 (тип 2)

Строится схема в Thinkercad




Ссылка на проект в Thinkercad: https://www.tinkercad.com/things/ccQbVbDWyWg-copy-of-copy-of-copy-of-funky-kup/editel?tenant=circuits?sharecode=uT6nQvo1TEH2T9ns1QQnSF_Ygmqw390IrOsaUISYw6A=

**Программный код для Arduino**

```
// подключение библиотеки для ленты
#include <Adafruit_NeoPixel.h>

#define PIN 2   // input pin Neopixel is attached to

#define NUMPIXELS      12 // number of neopixels in strip
```

```
// создание объекта ленты
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB +
NEO_KHZ800);


int redColor = 200;
int greenColor = 0;
int blueColor = 0;
int speed = 0;

void setup() {
  // Initialize the NeoPixel library.
  pixels.begin();
  //инициализация моторов и АЦП
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(A1, INPUT);
}

int i = 0;

void loop() {
  // получение данных с АЦП и расчет скорости и направления мотора
  // цифра 675 подбирается для установленного резистора (верхнее значение АЦП)
  speed = (analogRead(A1) / 675.f) * 255.0f - 127;
  speed *= -2;
  speed = constrain(speed, -255, 255);

  // если потенциометр находится в середине, то диоды не двигаются
  if (speed != 0) {
     // установка тяги мотора в нужном направлении
    if (speed > 0) {
      analogWrite(10, 0);
      analogWrite(9, speed);
    } else {
      analogWrite(10, -speed);
      analogWrite(9, 0);
    }
     // выбор направления движения ленты
    if(speed > 0){
       i++;
    } else {
      if (i == 0) i = 12;
      i--;
    }
    // проверка выхода за границы количества диодов ленты
    i %= 12;
    for(int j = 0; j < 12; j++) {
      // очищаем все диоды
```

```
    pixels.setPixelColor(j, pixels.Color(0, 0, 0));
  }
    // устанавливаем текущий диод в выбранный цвет
    pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
    // рендер ленты
    pixels.show();
    // задержка обратно-пропорциональная скорости
    delay((11000.0f - 10000.0f * (abs(speed) / 255.0f)) / 12.0f);
  }
}
```

## Задача 3 (тип 2)

Дано:

- Arduino UNO
- 7-сегментный дисплей
- Микросхема CD4511
- Две тактовые кнопки
- Потенциометр

Используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) необходимо скопировать проект по ссылке: https://www.tinkercad.com/things/3TYLXIYlBnm

В данной схеме имеется микроконтроллер Arduino UNO, к которому подключен 7-сегментный дисплей через микросхему CD4511, а также подключены две тактовые кнопки и потенциометр. Редактирование электронной схемы не допускается.

Необходимо запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

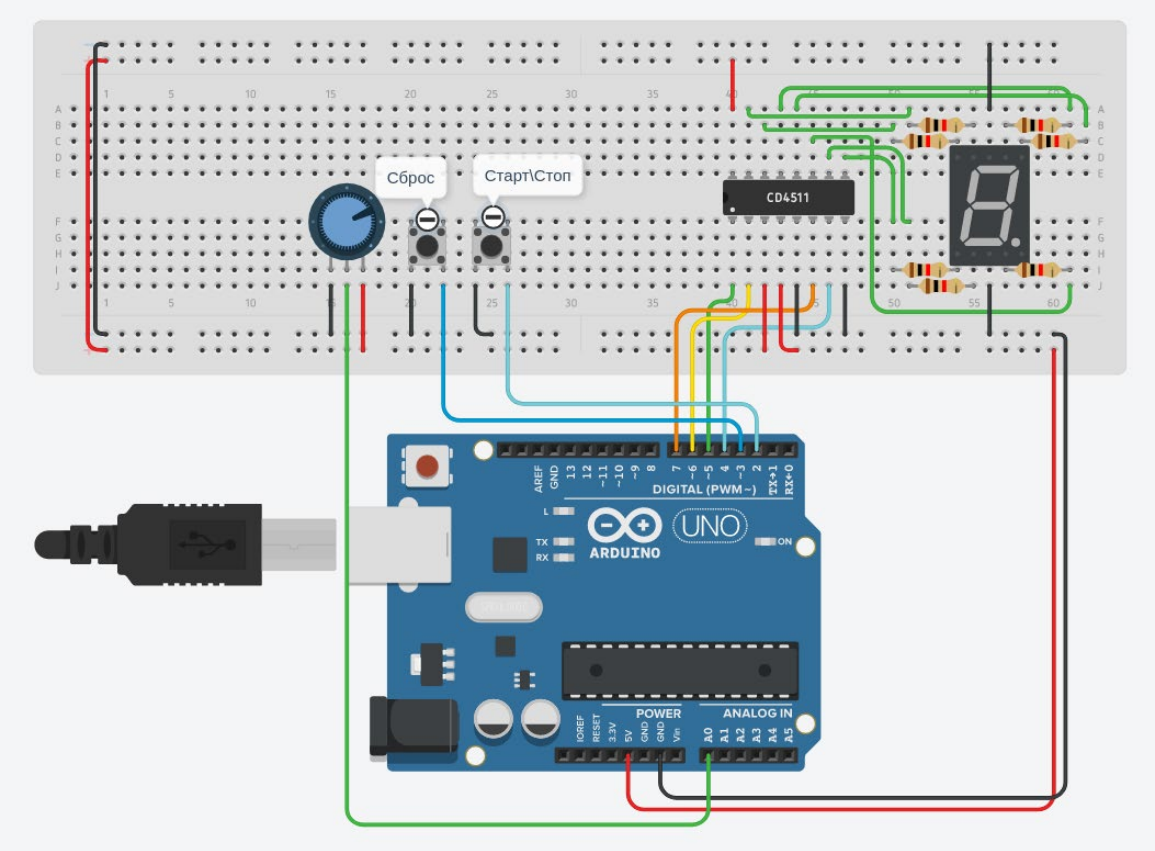
- Устройство должно представлять собой таймер обратного отсчёта. Значение счётчика должно отображаться на дисплее.
- С помощью потенциометра пользователь должен иметь возможность установить на дисплее число от 0 до 9 включительно.
- С помощью кнопки «Сброс» (пин 3), отображаемое число сбрасывается к значению, которое установлено через потенциометр.
- С помощью кнопки «Старт/Пауза» (пин 2) должна предоставляться возможность запускать таймер и ставить его на паузу.
- После того, как значение таймера достигает нуля, должен загораться встроенный светодиод на Arduino UNO. В иных случаях светодиод должен быть выключен.

### Решение задачи 3 (тип 2)

Решение доступно по ссылке https://yadi.sk/d/KrZAbhWM9nDhAw.

В данной схеме имеется микроконтроллер Arduino UNO, к которому подключен 7-сегментный дисплей через микросхему CD4511, а также подключены две тактовые кнопки и потенциометр. Редактирование электронной схемы не допускается.

Устройство представляет собой таймер обратного отсчёта.




Ссылка на схему TinkerCad: https://www.tinkercad.com/things/jvSEil0gZCt

**Программный код для Arduino:**

```
/* Таймер.
 *
 * Количество секунд для отсчёта задаётся потенциометром.
 *
 * Кнопка "старт\стоп" запускает или останавливает таймер
 * в зависимости от его текущего состояния.
 *
 * Кнопка "сброс" останавливает и сбрасывает таймер
 * и переводит его в режим настройки потенциометром.
 *
 * При достижении нуля таймер останавливает и
 * загорается светодиод.
 */

// порты для потенциометра, кнопок и светодиода
#define POTENTIOMETER A0
#define BUTTON_START 2
#define BUTTON_RESET 3
```

```
#define LED 13

// порты для микросхемы CD4511, которая
// управляет 7-сегментным индикатором
const int digit_pins[4] = {4, 5, 6, 7};

// режимы работы таймера:
// SETTING - настройка, установка секунд отсчёта
// RUNNING - отсчёт времени
// PAUSE - пауза
enum modes {SETTING = 0, RUNNING, PAUSE};

// переменная текущего режима
int mode = SETTING;

// счётчик секунд
uint8_t counter = 0;

// обработка нажатия кнопки старт\стоп
void button_start_clicked() {
  // если сейчас режим настройки или пауза…
  if (mode == SETTING || mode == PAUSE) {
    // то запускаем таймер
    mode = RUNNING;
  } else if (mode == RUNNING) {
    // если же он запущен, то ставим паузу
    mode = PAUSE;
  }
}

// обработка нажатия кнопки сброса
void button_reset_clicked() {
  // перевод таймера в режим настройки
  mode = SETTING;
  // сброс счётчика
  counter = 0;
}

// функция для чтения значений с потенциометра
int read_potentiometer() {
  // считывание необработанного значения с АЦП
  int potentiometer_value = analogRead(POTENTIOMETER);
  // перевод значения из шкалы 0…1023 в диапазон 0…9
  return map(potentiometer_value, 0, 1023, 0, 9);
}

// обработка одного тика таймера
void timer_tick() {
  // если счётчик больше нуля, то делаем отсчёт
  if (counter > 0) {
    // уменьшение счётчика на единицу
    counter--;
    // ожидание в 1 секунду
    delay(1000);
  } else if (counter == 0) { // если достигли нуля
    // то включается свтодиод
    digitalWrite(LED, HIGH);
  }
}

// функция установки времени отсчёта
void setting_time() {
  // приравнивение счётчика значению с потенциометра
  counter = read_potentiometer();
  // небольшая задержка
  delay(100);
}

// функция, которая один раз запускается на старте
void setup() {
  // инициализация портов кнопок на вход
  pinMode(BUTTON_START, INPUT_PULLUP);
  pinMode(BUTTON_RESET, INPUT_PULLUP);
  // инициализация порта светодиода на выход
  pinMode(LED, OUTPUT);
```

```
  // привязка прерываний портов кнопок к обработчикам
  // (это позволит обрабатывать нажатия кнопок асинхронно
  // и независимо от основного цикла). благодаря этому,
  // ожидаемый сигнал не будет пропущен.

  attachInterrupt(digitalPinToInterrupt(BUTTON_START),
                  button_start_clicked, FALLING);

  attachInterrupt(digitalPinToInterrupt(BUTTON_RESET),
                  button_reset_clicked, FALLING);

  // инициализация портов индикатора
  for (int i = 0; i < 4; i++) {
    pinMode(digit_pins[i], OUTPUT);
  }
}

// основной цикл программы
void loop() {
  // если режим установки времени отсчёта,
  // то вызываем соответствующую функцию-обработчик
  if (mode == SETTING) setting_time();
  // если таймер запущен, то нужно вызвать
  // функцию обработки тика таймера
  if (mode == RUNNING) timer_tick();
  // в случае паузы просто делаем небольшую задержку
  if (mode == PAUSE) delay(100);

  // вывод текущего значения счётчика на индикатор
  for (int i = 0; i < 4; i++) {
    // индикатор управляется микросхемой CD4511.
    // он преобразует двоичный код в соответствующий
    // для цифры код 7-сегментного индикатора.
    // чтобы вывести цифру, нужно подать на входы
    // микросхемы двоичный код числа.
    // для этого воспользуемся функцией bitRead,
    // которая позвоялет считать один бит числа
    // в определенной позиции.
    digitalWrite(digit_pins[i], bitRead(counter, i));
  }

  // если счётчик не равен нулю, то выключаем светодиод.
  if (counter != 0) digitalWrite(LED, LOW);
```

## Задача 4 (тип 2)

Дано:

- Arduino UNO
- Клавиатура 4×4
- Символьный дисплей 16×2

Используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) необходимо скопировать проект по ссылке: https://www.tinkercad.com/things/0DRRITK63Dt
В данной схеме имеется микроконтроллер Arduino UNO, к которму подключены клавиатура 4×4 и символьный дисплей 16×2.

Необходимо запрограммировать Arduino UNO, чтобы выполнялись следующие требования:

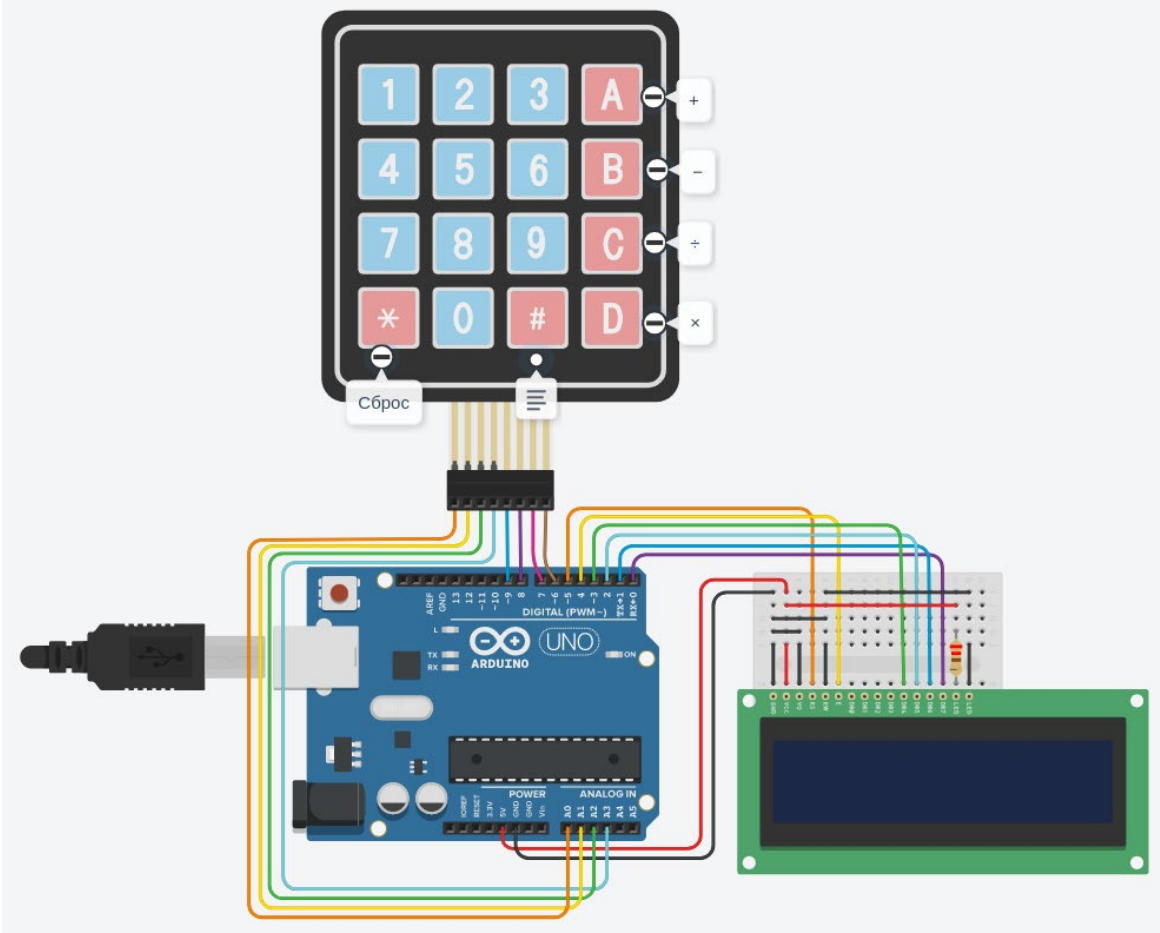
- Устройство должно представлять собой калькулятор, с помощью которого можно производить простые вычисления (складывать, вычитать, делить и умножать числа).
- Функции кнопок:
- цифры - числовой ввод
- A - сложение (+)
- B - вычитание (-)
- C - деление (÷)
- D - умножение (×)
- * - сброс
- # - равно (=)
- На дисплее должно отображаться первое число (над которым производится математическая операция); символ, обозначающий математическую операцию; второе число, участвующее в математической операции.
- После выполнения операции (при нажатии на кнопку «равно»), результат является «первым числом», над которым пользователь будет производить математическую операцию (до нажатия кнопки сброса).
- При нажатии на кнопку «сброс», очищаются оба числа и вид математической операции.

## Решение задачи 4 (тип 2)

Решение доступно по ссылке https://yadi.sk/d/KrZAbhWM9nDhAw.

В данной схеме имеется микроконтроллер Arduino UNO, к которому подключены клавиатура 4×4 и символьный дисплей 16×2.

Устройство представляет собой калькулятор.




Ссылка на схему TinkerCad: https://www.tinkercad.com/things/8fSsNwQKS1C

### Программный код для Arduino:

```
/* Калькулятор.
 *
 * Пользователь вводит первое число выражения
 * в верхней строке, после чего нажимает на
 * кнопку математической операции.
 *
 * После этого пользователь вводит второе число.
 *
 * Далее пользователь может нажать на кнопку "="
 * для вывода результата, или же снова нажать на
 * кнпоку математической операции, чтобы
 * продолжить вычисления над результатом.
 *
 * Также есть кнопка для сброса.
 */
```

```
// подключение библиотеки для дисплея
#include <LiquidCrystal.h>
// подключение библиотеки для клавиатуры
#include <Keypad.h>

// создание объекта для управления дисплеем
LiquidCrystal lcd(5, 4, 3, 2, 1, 0);

// кол-во строк и столбцов кнопок клавиатуры
const byte ROWS = 4;
const byte COLS = 4;

// установка соответствия между кнопками
// клавиатуры с символами на них
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', '+'},
  {'4', '5', '6', '-'},
  {'7', '8', '9', '/'},
  {'*', '0', '=', 'x'}
};

// задание портов для клавиатуры
byte rowPins[ROWS] = {A0, A1, A2, A3};
byte colPins[COLS] = {9, 8, 7, 6};

// создание объекта клавиатуры
Keypad customKeypad = Keypad(makeKeymap(hexaKeys),
                             rowPins, colPins, ROWS, COLS);

// доступные операции над числами
// (сложение, вычитание, деление, умножение)
enum operations {PLUS = 0, MINUS, DIVIDE, MULTIPLY};
// режимы работы (ожидание первого числа,
// ожидание второго числа, вывод результата)
enum modes {WAIT_FIRST = 0, WAIT_SECOND, RESULT};

int mode = WAIT_FIRST;  // текущий режим
int operation = 0;      // текущая операция

float num1 = 0;         // первое число
float num2 = 0;         // второе число
float result = 0;       // результат
int num_digit_count = 0;// счётчик цифр

char current_key;   // текущая нажатая кнопка

// функция очистки калькулятора
void clear_calculator() {
  // обнуление первого и второго числа
  num1 = 0;
  num2 = 0;
  // устанвока режима на ожидание первого числа
  mode = WAIT_FIRST;
  // обнуление счётчика цифр
  num_digit_count = 0;

  // очистка дисплея
  lcd.clear();
  lcd.setCursor(0, 0);
};

// перевод ASCII-символа в цифру
// (см. таблицу ASCII-кодов)
int ascii_to_int(int num) {
  // если символ попадает в диапазон цифр
  if (num >= 48 && num <= 57) {
    // то возвращаем цифру, соответствующую символу
    return num - 48;
  } else {
    // иначе возвращаем -1 (не цифра)
    return -1;
  }
}

// заполнение чисел пользователем
```

```
void fill_num(float &target_num) {

  // если клавиша не нажата, или число слишком большое
  if (!current_key || num_digit_count >= 14){
    // то выходим из функции
    return;
  }

  // получаем цифру из введенного символа
  int num = ascii_to_int(current_key);

  // если было введено число
  if (num != -1) {
    num_digit_count++;  // прибавляем счётчик цифр
    lcd.print(num);     // выводим на дисплей цифру
    target_num *= 10;   // сдвигаем десятичный разряд
    target_num += num;  // добавляем цифру
  }
}

// функция для вывода результата
void show_result() {
  // очистка дисплея
  lcd.clear();
  result = 0;

  // если было ожилание первого числа (второго ещё нет),
  // то результатом будет просто первое число
  // (т.е. операций не было произведено)
  if (mode == WAIT_FIRST) {
    result = num1;
  } else {

    // иначе будет производить вычисления

    // сложение
    if (operation == PLUS) {
      result = num1 + num2;
    }

    // вычитание
    if (operation == MINUS) {
      result = num1 - num2;
    }

    // умножение
    if (operation == MULTIPLY) {
      result = num1 * num2;
    }

    // деление
    if (operation == DIVIDE) {
      result = num1 / num2;
    }

  }

  mode = RESULT;  // устанавливаем новый режим
  num1 = result;  // первым числом становится результат
  num_digit_count = 0;  // обнуляем счётчик цифр

  // перевод курсора в начало верхней строки
  lcd.setCursor(0, 0);
  lcd.print('='); // вывод символа равенства
  // перевод курсора в начало нижней строки,
  // где будет отображён результат
  lcd.setCursor(0, 1);

  // проверки на ошибки

  if (isnan(result)) {  // если не число (not a number)
    lcd.print("error: nan");
    return;
  }
```

```
  if (isinf(result)) {  // если бесконечность (infinity)
    lcd.print("error: inf");
    return;
  }

  // проверка на переполнение
  if (result > 4294967040.0 ||
      result <-4294967040.0 ) {
    lcd.print("error: overflow");
    return;
  }

  // вывод результата
  lcd.print(result, DEC);
}

// установка математической операции
void set_operation(int new_operation) {
  operation = new_operation;

  // если первое число не было введено,
  // то отобразим на его месте 0
  if (num1 == 0) {
    lcd.setCursor(0, 0);
    lcd.print(0);
  }

  // если было введено второе число,
  // то нужно произвести вычисления
  // (это нужно, если подряд идёт
  // несколько математических операций,
  // например 12×34+56 можно будет посчитать
  // не нажимая кнопку "равно")
  if (mode == WAIT_SECOND) {
    // расчёт и вывод результата
    show_result();
    // снова устанавливаем ожидание второго числа
    // (первым числом стал результат)
    mode = WAIT_SECOND;

    // очистка дисплея
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(num1, DEC);
    // обнуление второго числа
    num2 = 0;
    // установка курсора в начало нижней строки,
    // куда будет вводиться второе число
    lcd.setCursor(0, 1);

    // выход из функции
    return;
  }

  // для вывода знака текущей операции,
  // перемещаем курсор в конец верхней строки
  lcd.setCursor(15, 0);
  // вывод знака математической операции
  lcd.print(current_key);

  // установка режима ожидания второго числа
  mode = WAIT_SECOND;
  // обнуление счётчика цифр
  num_digit_count = 0;
  // установка курсора в начало нижней строки
  lcd.setCursor(0, 1);
}

// функция, которая выполнится один раз в начале
void setup() {
  // инициализация дисплея
  lcd.begin(16, 2);
  // включение отображения курсора
  lcd.cursor();
  // обнуление калькулятора
```

```
  clear_calculator();
}

// основной цикл программы
void loop() {
  // получение текущей нажатой клавиши
  current_key = customKeypad.getKey();

  // если сейчас отображается результат
  // и при этом была нажата клавиша…
  if (mode == RESULT && current_key) {
    // очистка дисплея
    lcd.clear();

    // если была нажата клавиша математической операции,
    // то нужно произвести эту операцию над результатом
    // (первым числом является результат)
    if (current_key == '+' ||
        current_key == '-' ||
        current_key == 'x' ||
        current_key == '/' ) {

      // очистка дисплея
      lcd.clear();
      // установка курсора в начало первой строки
      lcd.setCursor(0, 0);
      // вывод первого числа (сейчас это результат
      // от предыдущего расчёта)
      lcd.print(num1, DEC);
      // обнуляем второе число
      num2 = 0;
    } else if (ascii_to_int(current_key) != -1) {
      // если при показе результата было нажато число,
      // то нужно обнулить калькулятор
      // (т.е. начался ввод новых чисел)
      clear_calculator();
    }

    // установка режима ожидания первого числа
    mode = WAIT_FIRST;
  }

  // если была нажата клавиша
  if (current_key) {

    // проверка нажатой клавиши
    // на соответствие действию (сброс, равно)
    // или математической операции

    if (current_key == '*') {
      clear_calculator();
      return;
    }

    if (current_key == '+') {
      set_operation(PLUS);
      return;
    }

    if (current_key == '-') {
      set_operation(MINUS);
      return;
    }

    if (current_key == 'x') {
      set_operation(MULTIPLY);
      return;
    }

    if (current_key == '/') {
      set_operation(DIVIDE);
      return;
    }

    if (current_key == '=') {
```

```
      show_result();
      return;
    }
  }

  // если ожидается первое число,
  // то заполняем именно его - num1
  if (mode == WAIT_FIRST) fill_num(num1);

  // если же ожидается второе число,
  // то соответственно заполняем num2
  if (mode == WAIT_SECOND) fill_num(num2);
}
```

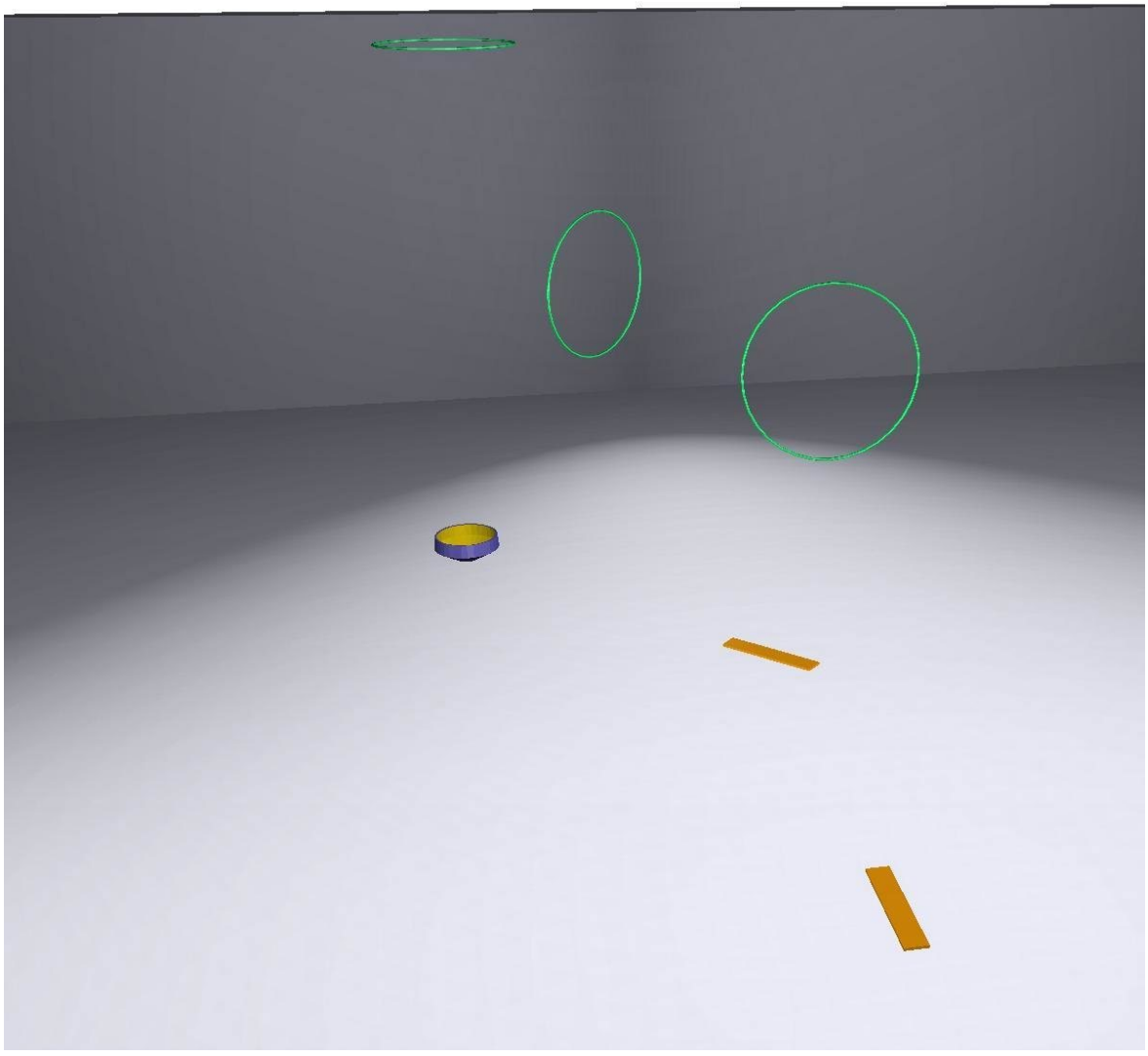
Кружковое движение НТО ЦРР

# Программирование АНПА

## Задача 5 (тип 2)

Необходимо выполнить следующие задачи в симуляторе MUR IDE:

- 10 баллов. Пройти сквозь обруч 1. На обруч 1 направлена полоска 1.
- 10 баллов. Пройти сквозь обруч 2. На обруч 2 направлена полоска 2.
- 20 баллов. Сбросить маркер в корзину. На корзину направлена полоска 2.
- 10 баллов. Всплыть в обруче, который расположен над корзиной.



Для отладки можете использовать эти сцены https://yadi.sk/d/xoVRegBkeki93A.

Для проверки решений мы будем использовать сцену, где может изменяться:

- Направление полоски 2. От -45 до 45 градусов.
- Расстояние от полоски 1 до полоски 2.
- Расстояние от полоски 2 до корзины.

Остальное (глубина бассейна, размеры и цвет объектов, глубина расположения обручей) меняться не будет.

При старте робот направлен на первую полоску.

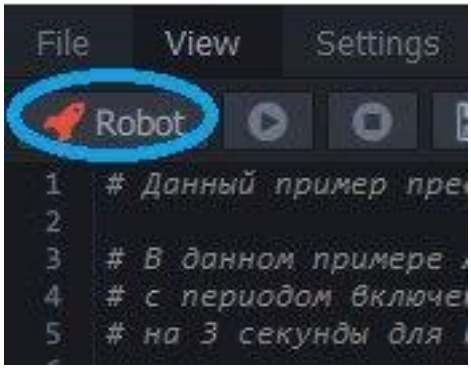
## Как выполнять задания в MUR IDE

Для выполнения отборочного задания вам необходимо:

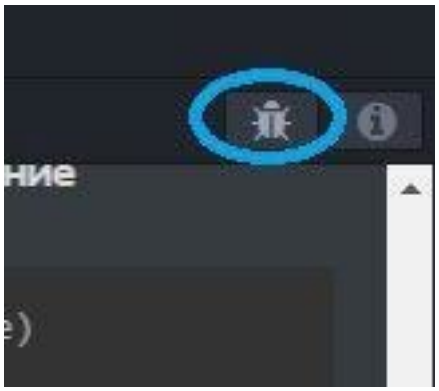
1. Скачать и установить MUR IDE: https://murproject.com/documents/17/murIDE.exe

2. После установки на рабочем столе и должен появиться ярлык MUR IDE. Запустите MUR IDE.

3. Для начала работы в симуляторе, вам необходимо перевести режим работы IDE в Local. Для этого, нажмите на кнопку с иконкой ракеты и надписью Remote в левом верхнем углу. Цвет кнопки станет синим и надпись изменится на Local.

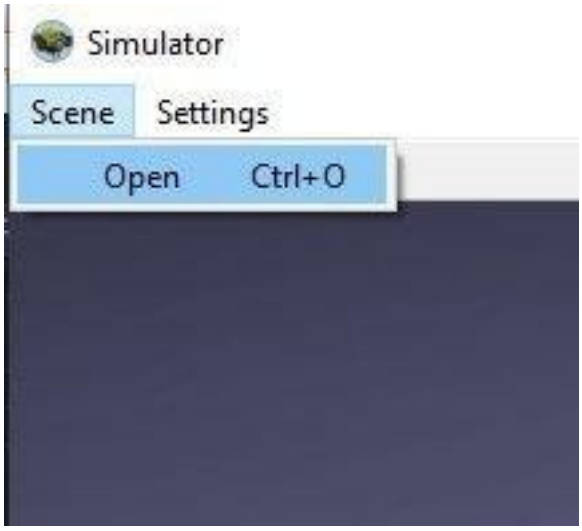



4. Далее, вам необходимо запустить симулятор нажатием кнопки с изображением жука в правом верхнем углу. Откроется окно симулятора с черным экраном.



5. Скачать сцены для отбора можно здесь https://yadi.sk/d/xoVRegBkeki93A. Можно разместить их в любом удобном каталоге.

6. В запущенном симуляторе переходим в меню Scene -> Open. В появившемся диалоговом окне выбираем одну из скачанных сцен.




7. Все готово! Вы можете начать программировать виртуальный аппарат в MiddleAUV на языке программирования Python.

8. Здесь можно посмотреть видеоуроки по MUR IDE.

## Решение задачи 5 (тип 2)

Для успешного выполнения задачи необходимо реализовать следующие функции:
- Стабилизация аппарата по курсу.
- Стабилизация аппарата по глубине.
- Поворот аппарата на по линии.
- Поиск линии.
- Стабилизация над корзиной.
- Сброс маркера
- Всплытие

Для начала реализуем ПД регулятор для нашего аппарата, который мы будем использовать в нескольких местах: стабилизация курса, глубины и выравнивание над объектами. Класс нашего регулятора будет выглядеть следующим образом:

```
class  PDRegulator(object):
    _p_gain = 0.0
    _d_gain = 0.0
    _prev_error = 0.0
    _timestamp = 0

    def __init__(self):
        pass

    def set_p_gain(self, value):
        self._p_gain = value

    def set_d_gain(self, value):
        self._d_gain = value

    def process(self, error):
        timestamp = int(round(time.time() * 1000))

        if timestamp == self._timestamp:
            return 0

        output = self._p_gain * error + self._d_gain / (timestamp -
self._timestamp) * (error - self._prev_error)
        self._timestamp = timestamp
        self._prev_error = error
        return output
```

Далее реализуем регулятор стабилизации по курсу и глубине:

```
def keep_yaw(yaw_to_set, speed):
    def clamp_angle(angle):
        if angle > 180.0:
            return angle - 360.0
        if angle < -180.0:
            return angle + 360
        return angle

    try:
```

```
        error = mur.get_yaw() - yaw_to_set
        error = clamp_angle(error)
        output = keep_yaw.yaw_regulator.process(error)
        mur.set_motor_power(0, clamp(-output + speed, -100, 100))
        mur.set_motor_power(1, clamp(output + speed, -100, 100))
    except AttributeError:
        keep_yaw.yaw_regulator = PDRegulator()
        keep_yaw.yaw_regulator.set_p_gain(0.8)
        keep_yaw.yaw_regulator.set_d_gain(0.6)


def keep_depth(depth_to_set):
    try:
        error = mur.get_depth() - depth_to_set
        output = keep_depth.depth_regulator.process(error)
        output = clamp(output, -100, 100)
        mur.set_motor_power(2, output)
        mur.set_motor_power(3, output)
    except AttributeError:
        keep_depth.depth_regulator = PDRegulator()
        keep_depth.depth_regulator.set_p_gain(45)
        keep_depth.depth_regulator.set_d_gain(5)
```

Далее создадим утилитарный класс для хранения информации о текущей глубине, скорости курсе и т.д:

```
class AUVContext(object):
    _yaw = 0.0
    _depth = 0.0
    _speed = 0.0
    _side_speed = 0.0
    _timestamp = 0
    _missions = []
    _min_area = math.inf
    _min_yaw = 0.0

    def __init__(self):
        pass

    def set_min_circle(self, yaw, area):
        if area < self._min_area:
            self._min_area = area
            self._min_yaw = yaw

    def get_min_circle_yaw(self):
        return self._min_yaw

    def get_yaw(self):
        return self._yaw

    def get_depth(self):
        return self._depth
```

```
    def get_speed(self):
        return self._speed

    def get_side_speed(self):
        return self._side_speed

    def set_yaw(self, value):
        self._yaw = value

    def set_depth(self, value):
        self._depth = value

    def set_speed(self, value):
        self._speed = value

    def set_side_speed(self, value):
        self._side_speed = value

    def push_mission(self, mission):
        self._missions.append(mission)

    def pop_mission(self):
        if len(self._missions) != 0:
            return self._missions.pop(0)
        return {}

    def get_missions_length(self):
        return len(self._missions)

    def process(self):
        timestamp = int(round(time.time() * 1000))
        if timestamp - self._timestamp > 16:
            keep_yaw(self._yaw, self._speed)
            keep_depth(self._depth)
            mur.set_motor_power(4, self._side_speed)
            self._timestamp = timestamp
        else:
            time.sleep(0.05)
```

Далее реализуем код поиска линии и установки угла.

```
def find_and_set_line_angle():
    image = mur.get_image_bottom()
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
    mask = cv.inRange(hsv_image, (15, 50, 50), (25, 255, 255))
    cv.imshow('', mask)
    contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
    cv.waitKey(1)
    if contours:
        for contour in contours:
            angle = find_rectangle_contour_angle(contour)
```

```
            context.set_yaw(context.get_yaw() + angle)
            return True

    context.set_speed(20)
    context.set_yaw(0)
    return False
```

Теперь реализуем функции для работы с корзиной.

```
def find_yellow_bin(image):
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
    mask = cv.inRange(hsv_image, (25, 50, 50), (30, 255, 255))
    contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_NONE)

    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 3000:
                continue
            print(area)
            ((_, _), (w, h), _) = cv.minAreaRect(contour)

            aspect_ratio = max(w, h) / min(w, h)
            if 0.8 <= aspect_ratio <= 1.2:
                continue

            moments = cv.moments(contour)
            try:
                x = int(moments['m10'] / moments['m00'])
                y = int(moments['m01'] / moments['m00'])
                cv.circle(image, (x, y), 2, (255, 0, 255), 2)
                cv.imshow("", image)
                cv.waitKey(1)
                return True, x, y, find_rectangle_contour_angle(contour)
            except ZeroDivisionError:
                return False, 0, 0, 0
    return False, 0, 0, 0

def find_rectangle_contour_angle(contour):
    rectangle = cv.minAreaRect(contour)
    box = cv.boxPoints(rectangle)
    box = np.int0(box)
    edge_first = np.int0((box[1][0] - box[0][0], box[1][1] - box[0][1]))
    edge_second = np.int0((box[2][0] - box[1][0], box[2][1] - box[1][1]))

    edge = edge_first
    if cv.norm(edge_second) > cv.norm(edge_first):
        edge = edge_second

    angle = -((180.0 / math.pi * math.acos(edge[0] / (cv.norm((1, 0)) *
cv.norm(edge)))) - 90)
```

```
    return angle


def move_to_yellow_bin():
    context.set_speed(20)
    found, x, y, angle = find_yellow_bin(mur.get_image_bottom())
    if found:
        context.set_speed(0)
        if stab_on_yellow_rectangle(x, y):
            context.set_yaw(angle + context.get_yaw())
            return True
    return False
```

Добавим функции всплытия и сброса.

```
def surface():
    mur.set_motor_power(2, 50)
    mur.set_motor_power(3, 50)
    time.sleep(5)
    return True

def drop():
    mur.drop()
    time.sleep(5)
    return True
```

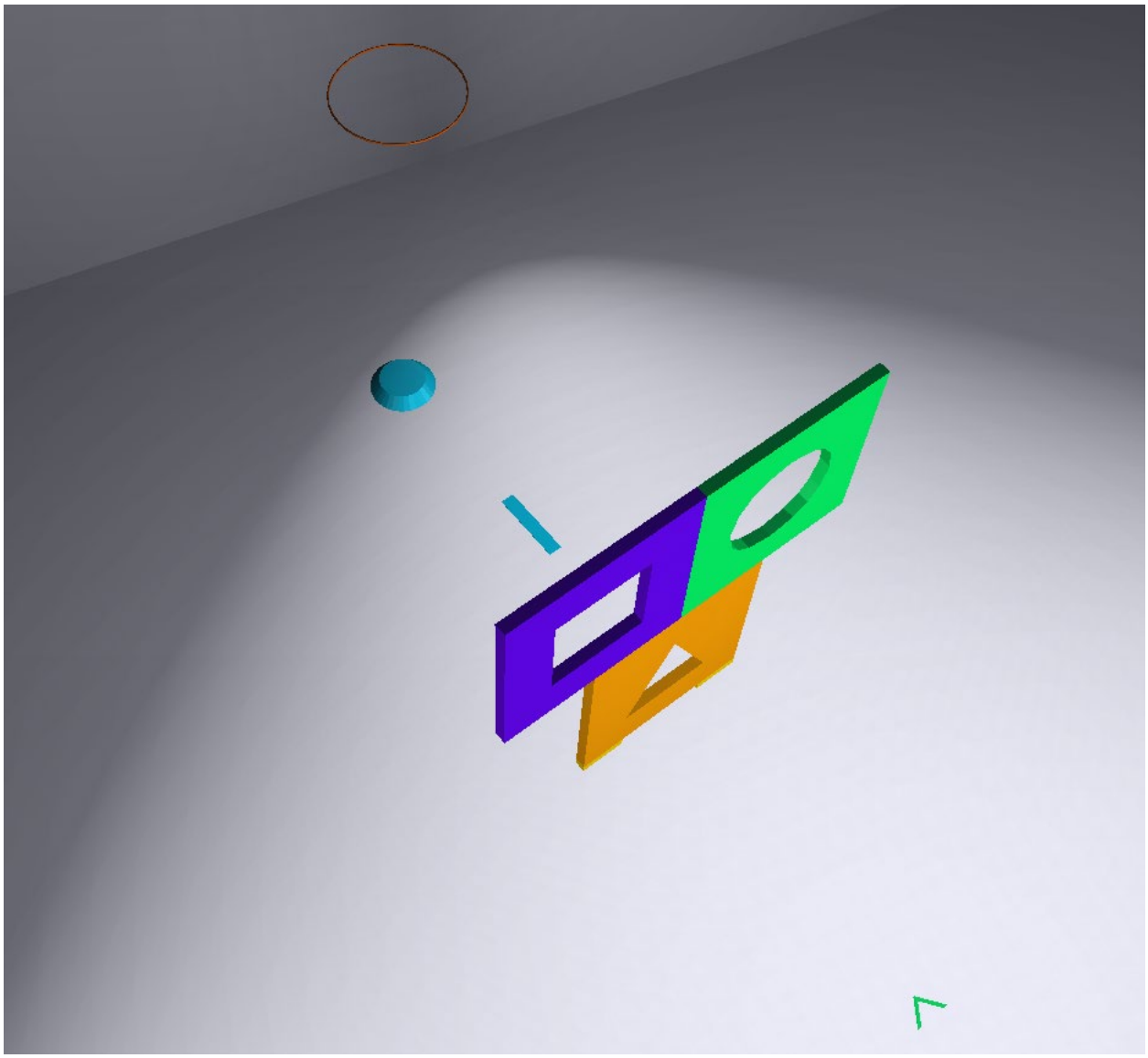
Собрав все вместе мы сможем успешно выполнить задание.

## Задача 6 (тип 2)

1. Задача выполняется в симуляторе и среде программирования MUR IDE. Сцены называются: NTI-Task-1_1, NTI-Task-1_2 и NTI-Task-1_3. И доступны по ссылке.

2. Необходимо запрограммировать подводного робота, который должен в симуляторе в автономном режиме выполнить под водой ряд задач и всплыть в заданной области. Время на выполнение задачи — 5 минут.

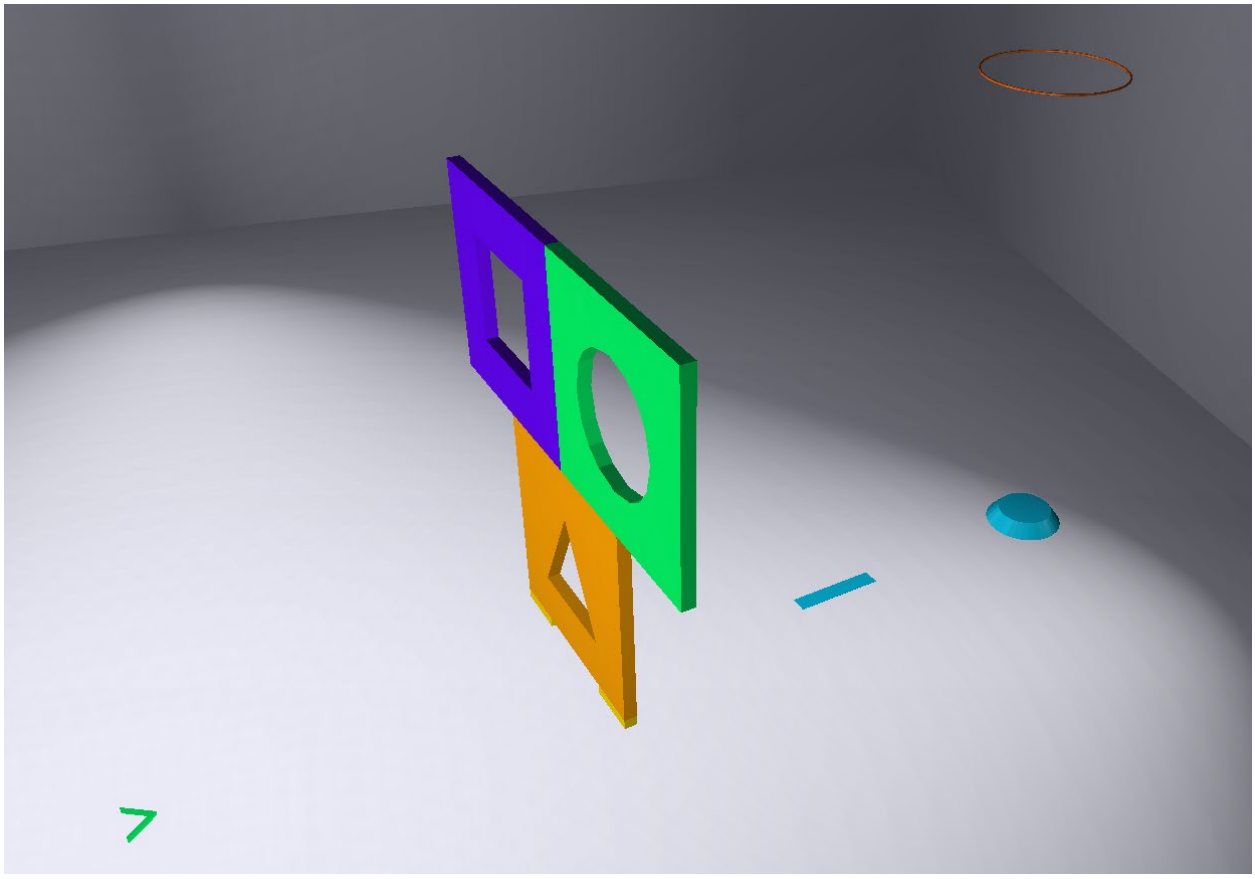
3. Как только робот всплыл или закончилось время, задача считается законченной, фиксируется количество заработанных баллов.



4. Задание:

- Необходимо определить **цвет указателя**, над которым стартует робот.
- Далее нужно последовательно запустить по одной торпеде **в каждую** из двух мишеней, цвет которых **не совпадает** с цветом указателя.
- После этого, робот должен проплыть через мишень **того же** цвета, что и указатель.

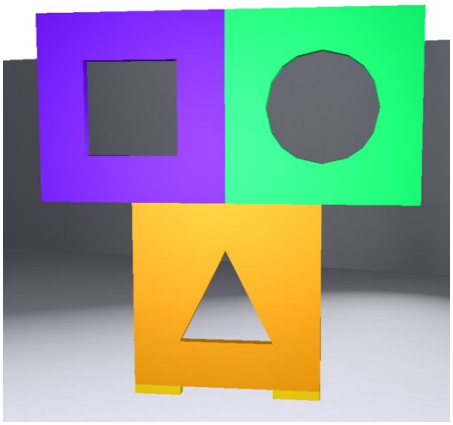
- Затем проследовать по направляющей полоске голубого цвета.
- Над круглой голубой платформой расположен обруч, в пределах которого нужно всплыть на поверхность.



5. Описание макетов:

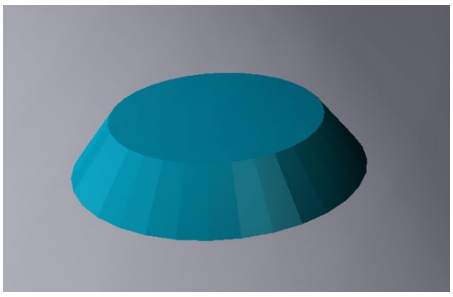
В задаче используются следующие виды объектов: указатель, стенд с мишенями, полоска, платформа, обруч.

- Указатель может быть окрашен в один из трёх цветов — синий, зелёный и оранжевый. Указатель располагается под стартовой позицией робота и указывание направление на стенд с мишенями. Цвет указателя говорит о том, через мишень какого цвета необходимо проплыть.



- Стенд с мишенями — представляет собой стенд, состоящий из трёх мишеней разных форм (квадрат, круг и треугольник) и цветов (синий, зелёный и оранжевый). Формы и цвета мишеней могут различаться в разных сценах. Необходимо будет проплыть в ту мишень, цвет которой совпадает с цветом указателя на стартовой позиции, перед этим выстрелив в две другие мишени. Мишень, через которую необходимо пройти, будет либо квадратом, либо кругом (не треугольником).

- Полоска (голубого цвета) — указывает направление к платформе. По данной полоске необходимо будет скорректировать направление к платформе.



- Платформа (голубого цвета) — служит для определения расположения обруча.

- Обруч — расположен на поверхности воды и находится над голубой платформой. Аппарат должен всплыть внутри обруча.

### Решение задачи 6 (тип 2)

Решение (NTI-Simulator-Task1.py) доступно по ссылке https://yadi.sk/d/KrZAbhWM9nDhAw.

Для успешного выполнения задания аппарату в виртуальной среде MUR IDE необходимо последовательно запустить по одной торпеде в каждую из двух мишеней, цвет которых не совпадает с цветом указателя на старте, после этого, робот должен проплыть через мишень того же цвета, что и указатель. Затем проследовать по направляющей полоске голубого цвета. Над круглой голубой платформой расположен обруч, в пределах которого нужно всплыть на поверхность.

Приступим к написанию программы. Для начала необходимо импортировать требуемые библиотеки и инициализировать API симулятора.

```
import pymurapi as api
import cv2 as cv
import time
import numpy as np
import math

mur = api.mur_init()
```

Согласно заданию, подберём диапазоны возможных цветов, которые встречаются в задании.

```
# предопределенные диапазоны цветов (HSV)
colors = {
    'green':  ((45, 50, 50), (75, 255, 255)),
    'blue':   ((130, 50, 50), (140, 255, 255)),
    'orange': ((10, 50, 50), (30, 255, 255))
}
```

Также напишем функции, которые пригодятся для дальнейших расчётов.

```
# функция для ограничения значения диапазоном
def clamp(value, min_value, max_value):
    if value < min_value:
        return min_value
    if value > max_value:
        return max_value
    return value

# функция для вычисления угла между точками
def angle_between(p1, p2):
    xDiff = p2[0] - p1[0]
    yDiff = p2[1] - p1[1]
    return math.degrees(math.atan2(yDiff, xDiff) - (np.pi / 2))

# функция для вычисления расстояния от начала координат
def length_from_center(x, y):
    return math.sqrt(x ** 2 + y ** 2)
```

Далее разработаем PD-регулятор, который будет использоваться для стабилизации курса, глубины и выравнивания над объектами.

```
# PD-регулятор
class PDRegulator(object):
    _p_gain = 0.0
    _d_gain = 0.0
    _prev_error = 0.0
    _timestamp = 0

    def __init__(self):
        pass
```

```
    def set_p_gain(self, value):
        self._p_gain = value

    def set_d_gain(self, value):
        self._d_gain = value

    def process(self, error):
        timestamp = int(round(time.time() * 1000))
        if timestamp == self._timestamp:
            return 0

        output = self._p_gain * error + self._d_gain / (timestamp - self._timestamp) * (error
- self._prev_error)
        self._timestamp = timestamp
        self._prev_error = error
        return output
```

С использованием PD-регулятора, опишем функции для регулирования курса и глубины робота.

```
# функция для установки курса робота
def keep_yaw(yaw_to_set, speed):
    def clamp_angle(angle):
        if angle > 180.0:
            return angle - 360.0
        if angle < -180.0:
            return angle + 360
        return angle

    try:
        error = mur.get_yaw() - yaw_to_set
        error = clamp_angle(error)
        output = keep_yaw.yaw_regulator.process(error)
        mur.set_motor_power(0, clamp(-output + speed, -100, 100))
        mur.set_motor_power(1, clamp(output + speed, -100, 100))
    except AttributeError:
        # создание PD-регулятора, если он отсутствует
        keep_yaw.yaw_regulator = PDRegulator()
        keep_yaw.yaw_regulator.set_p_gain(0.8)
        keep_yaw.yaw_regulator.set_d_gain(0.6)

# функция для установки глубины погружения
def keep_depth(depth_to_set):
    try:
        error = mur.get_depth() - depth_to_set
        output = keep_depth.depth_regulator.process(error)
        output = clamp(output, -100, 100)
        mur.set_motor_power(2, output)
        mur.set_motor_power(3, output)
    except AttributeError:
        keep_depth.depth_regulator = PDRegulator()
        keep_depth.depth_regulator.set_p_gain(45)
        keep_depth.depth_regulator.set_d_gain(5)
```

Напишем класс для хранения состояния робота и хода выполнения миссии.

```
# класс для хранения текущего состояния
class AUVContext(object):
    _yaw = 0.0
    _depth = 0.0
    _speed = 0.0
    _side_speed = 0.0
    _timestamp = 0
    _missions = []
    _min_area = math.inf
    _min_yaw = 0.0
    _stabilization_counter = 0
    time

    def __init__(self):
        pass

    def set_min_circle(self, yaw, area):
        if area < self._min_area:
            self._min_area = area
```

```
            self._min_yaw = yaw

    def get_min_circle_yaw(self):
        return self._min_yaw

    def get_yaw(self):
        return self._yaw

    def get_depth(self):
        return self._depth

    def get_speed(self):
        return self._speed

    def get_side_speed(self):
        return self._side_speed

    def set_yaw(self, value):
        self._yaw = value

    def set_depth(self, value):
        self._depth = value

    def set_speed(self, value):
        self._speed = value

    def set_side_speed(self, value):
        self._side_speed = value

    def get_stabilization_counter(self):
        return self._stabilization_counter

    def reset_stabilization_counter(self):
        self._stabilization_counter = 0

    def add_stabilization_counter(self):
        self._stabilization_counter += 1

    def check_stabilization(self, timeout = 3):
        if self._stabilization_counter > timeout:
            return True
        else:
            self.add_stabilization_counter()
            return False

    def push_mission(self, mission):
        self._missions.append(mission)

    def push_mission_list(self, missions):
        for mission in missions:
            self.push_mission(mission)

    def pop_mission(self):
        if len(self._missions) != 0:
            return self._missions.pop(0)
        return {}

    def get_missions_length(self):
        return len(self._missions)

    def process(self):
        timestamp = int(round(time.time() * 1000))
        if timestamp - self._timestamp > 16:
            keep_yaw(self._yaw, self._speed)
            keep_depth(self._depth)
            mur.set_motor_power(4, self._side_speed)
            self._timestamp = timestamp
        else:
            time.sleep(0.05)

# объект, где будет хранится текущее состояние
context = AUVContext()
```

Напишем ещё несколько полезных функций, которые позволяют выполнять различные действия, как развороты, выстрел и т.д.

```
# разворот на 90 градусов
def translate_to_90():
    yaw = mur.get_yaw() + 90
    if yaw < -180:
        yaw += 360
    if yaw > 180:
        yaw -= 360
    context.set_yaw(yaw)
    return True

# разворот на 180 градусов
def translate_to_180():
    yaw = mur.get_yaw() + 180
    if yaw < -180:
        yaw += 360
    if yaw > 180:
        yaw -= 360
    context.set_yaw(yaw)
    return True

# произвести выстрел
def shoot():
    if (context.get_speed() != 0):
        context.set_speed(0)
        return False
    else:
        mur.shoot()
        time.sleep(0.5)
        return True

# стабилизировать курс и глубину
def stabilize():
    yaw = mur.get_yaw()
    depth = mur.get_depth()

    if abs(yaw - context.get_yaw()) < 1 and abs(depth - context.get_depth()) < 0.3:
        if context.check_stabilization():
            return True
    else:
        context.reset_stabilization_counter()
    return False
```

В дальнейшем часто будет возникать задача по распознаванию контуров цветных объектов, для этого напишем соответствующую функцию.

```
# поиск на изображении контура по цвету
def find_contours(image, color_low, color_high, approx = cv.CHAIN_APPROX_SIMPLE):
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
    mask = cv.inRange(hsv_image, color_low, color_high)
    contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL, approx)
    return contours
```

Далее идут функции, активно использующие машинное зрение. Это такие действия, как расчёты координат и углов объектов, определение цветов объектов, позиционирование и стабилизация над объектами. Для стабилизации также используется ранее написанный PD-регулятор.

```
# расчёт угла прямоугольника
# для определение отклонения от полоски,
# чтобы затем скорректировать курс
def find_rectangle_contour_angle(contour):
    rectangle = cv.minAreaRect(contour)
    box = cv.boxPoints(rectangle)
    box = np.int0(box)
    edge_first = np.int0((box[1][0] - box[0][0], box[1][1] - box[0][1]))
    edge_second = np.int0((box[2][0] - box[1][0], box[2][1] - box[1][1]))

    edge = edge_first
    if cv.norm(edge_second) > cv.norm(edge_first):
        edge = edge_second

    angle = -((180.0 / math.pi * math.acos(edge[0] / (cv.norm((1, 0)) * cv.norm(edge)))) - 90)
```

```
    return angle

# стабилизироваться над желтым прямоугольником
def stabilize_on_yellow_rectangle(x, y):
    y_center = y - (240 / 2)
    x_center = x - (320 / 2)
    try:
        # определеяем и проверяем отклонение
        length = length_from_center(x_center, y_center)
        if length < 4.5:
            if context.check_stabilization():
                return True
        else:
            context.reset_stabilization_counter()

        output_forward = stabilize_on_yellow_rectangle.forward_regulator.process(y_center)
        output_side = stabilize_on_yellow_rectangle.side_regulator.process(x_center)

        output_forward = clamp(int(output_forward), -50, 50)
        output_side = clamp(int(output_side), -50, 50)

        context.set_speed(-output_forward)
        context.set_side_speed(-output_side)

    except AttributeError:
        stabilize_on_yellow_rectangle.forward_regulator = PDRegulator()
        stabilize_on_yellow_rectangle.forward_regulator.set_p_gain(0.5)
        stabilize_on_yellow_rectangle.forward_regulator.set_d_gain(0.1)

        stabilize_on_yellow_rectangle.side_regulator = PDRegulator()
        stabilize_on_yellow_rectangle.side_regulator.set_p_gain(0.5)
        stabilize_on_yellow_rectangle.side_regulator.set_d_gain(0.1)
    return False

# определение цвета стрелки
def detect_arrow_color():
    image = mur.get_image_bottom()
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

    # для определения цвета, выделим контуры для
    # каждого из возможных цветов, а затем
    # подсчитаем площадь каждого контура.
    # цвет стрелки - это контура с наибольшей площадью.

    areas = {}

    for color in colors:
        mask = cv.inRange(hsv_image, colors[color][0], colors[color][1])
        contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)

        biggest_area = 0

        if contours:
            for contour in contours:
                area = cv.contourArea(contour)
                if area > biggest_area:
                    biggest_area = area

        areas[color] = biggest_area

    color = sorted(areas, key=areas.get, reverse=True)[0]

    if (areas[color] > 5):
        # устанавливаем цвет мишени, через которую нужно пройти
        context.target_color = color
        # а также создаем список мишеней, в которые нужно выстрелить
        context.shooting_colors = list(set(colors.keys()) - {color})
        return True
    else:
        return False

# определение угла, куда направлена стрелка
def detect_arrow_angle(image, contour):
    (arrow_center_x, arrow_center_y), radius = cv.minEnclosingCircle(contour)
    moments = cv.moments(contour)
```

```
    try:
        arrow_direction_x = int(moments['m10'] / moments['m00'])
        arrow_direction_y = int(moments['m01'] / moments['m00'])

        arrow_angle = (angle_between((arrow_direction_x, arrow_direction_y), (arrow_center_x,
arrow_center_y)))
        context.set_yaw(mur.get_yaw() + arrow_angle)

        target_x = arrow_center_x - (320 / 2)
        target_y = arrow_center_y - (240 / 2)
        length = math.sqrt(target_x ** 2 + target_y ** 2)

        return arrow_angle, target_x, target_y, length
    except:
        return False, False, False, False

# стабилизироваться над стрелкой (с учётом направления стрелки)
def stabilize_on_arrow():
    image = mur.get_image_bottom()

    contours = find_contours(image, colors[context.target_color][0],
colors[context.target_color][1], cv.CHAIN_APPROX_SIMPLE)

    if contours:
        for contour in contours:
            (arrow_angle, target_x, target_y, length) = detect_arrow_angle(image, contour)

            if arrow_angle != False:
                try:
                    output_forward = stabilize_on_arrow.forward_regulator.process(target_y)
                    output_side = stabilize_on_arrow.side_regulator.process(target_x)

                    output_forward = clamp(int(output_forward), -50, 50)
                    output_side = clamp(int(output_side), -50, 50)

                    context.set_speed(-output_forward)
                    context.set_side_speed(-output_side)

                    if abs(arrow_angle) < 2 and length < 5:
                        if context.check_stabilization():
                            return True
                    else:
                        context.reset_stabilization_counter()

                except AttributeError:
                    stabilize_on_arrow.forward_regulator = PDRegulator()
                    stabilize_on_arrow.forward_regulator.set_p_gain(0.5)
                    stabilize_on_arrow.forward_regulator.set_d_gain(0.1)

                    stabilize_on_arrow.side_regulator = PDRegulator()
                    stabilize_on_arrow.side_regulator.set_p_gain(0.5)
                    stabilize_on_arrow.side_regulator.set_d_gain(0.1)

    return False

# прицелиться к мишени определенного цвета.
# для выстрела и прохождения через мишень
# будет отличаться требуемая глубина,
# поэтому присутствует дополнительная
# корректировка по оси Y
def aim_target(color, y_correction = 0):
    image = mur.get_image_front()
    contours = find_contours(image, colors[color][0], colors[color][1])

    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if area < 100:
                continue

            ((x, y), (w, h), _) = cv.minAreaRect(contour)

            x_center = x - (320 / 2)
            y_center = (y - (240 / 2)) - y_correction
```

```
            # у нас нет возможности точно измерить расстояние,
            # но можно получить примерное значение из площади контура.
            # чем ближе к мишени - тем больше её контур и наоборотю

            area = w * h
            target_distance = 50000

            try:
                output_forward = aim_target.forward_regulator.process(target_distance - area)
                output_yaw = aim_target.yaw_regulator.process(x_center)
                output_side = aim_target.side_regulator.process(x_center)
                output_depth = aim_target.depth_regulator.process(y_center)

                output_forward = clamp(int(output_forward), -20, 20)
                output_side = clamp(int(output_side), -50, 50)
                output_depth = clamp(output_depth, -1, 1)

                context.set_speed(output_forward)
                context.set_yaw(context.get_yaw() + output_yaw)
                context.set_side_speed(-output_side)
                context.set_depth(mur.get_depth() + output_depth)

                length = length_from_center(x_center, y_center)

                # нужно проверить как расстояние до мишени,
                # так и центрирование (расстояние до центра изображения)

                if length < 6 and abs(target_distance - area) < 5000:
                    if context.check_stabilization(timeout=5):
                        context.set_depth(mur.get_depth())
                        context.set_speed(0)
                        return True
                else:
                    context.reset_stabilization_counter()

            except AttributeError:
                aim_target.yaw_regulator = PDRegulator()
                aim_target.yaw_regulator.set_p_gain(0.002)
                aim_target.yaw_regulator.set_d_gain(0.002)

                aim_target.forward_regulator = PDRegulator()
                aim_target.forward_regulator.set_p_gain(0.2)
                aim_target.forward_regulator.set_d_gain(0.1)

                aim_target.side_regulator = PDRegulator()
                aim_target.side_regulator.set_p_gain(0.5)
                aim_target.side_regulator.set_d_gain(0.1)

                aim_target.depth_regulator = PDRegulator()
                aim_target.depth_regulator.set_p_gain(0.01)
                aim_target.depth_regulator.set_d_gain(0.01)

    return False

# обнаружение голубой полоски
def find_cyan_line(image):
    contours = find_contours(image, (80, 50, 50), (100, 255, 255))
    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 100:
                continue

            ((_, _), (w, h), _) = cv.minAreaRect(contour)
            aspect_ratio = max(w, h) / min(w, h)

            moments = cv.moments(contour)
            try:
                x = int(moments['m10'] / moments['m00'])
                y = int(moments['m01'] / moments['m00'])
                return True, x, y, find_rectangle_contour_angle(contour)
            except ZeroDivisionError:
                return False, 0, 0, 0
    return False, 0, 0, 0
```

```
# стабилизироваться по координатам, а также по углу
# иными словами, расположиться над объектом с заданным направлением
def stabilize_x_y_angle(x, y, angle):
    y_center = y - (240 / 2)
    x_center = x - (320 / 2)

    try:
        length = math.sqrt(x_center ** 2 + y_center ** 2)
        if length < 4.5:
            if context.check_stabilization():
                return True
        else:
            context.reset_stabilization_counter()

        output_forward = stabilize_x_y_angle.forward_regulator.process(y_center)
        output_side = stabilize_x_y_angle.side_regulator.process(x_center)

        output_forward = clamp(int(output_forward), -50, 50)
        output_side = clamp(int(output_side), -50, 50)

        context.set_speed(-output_forward)
        context.set_side_speed(-output_side)

        context.set_yaw(mur.get_yaw() + angle)
    except AttributeError:
        stabilize_x_y_angle.forward_regulator = PDRegulator()
        stabilize_x_y_angle.forward_regulator.set_p_gain(0.5)
        stabilize_x_y_angle.forward_regulator.set_d_gain(0.1)

        stabilize_x_y_angle.side_regulator = PDRegulator()
        stabilize_x_y_angle.side_regulator.set_p_gain(0.5)
        stabilize_x_y_angle.side_regulator.set_d_gain(0.1)
    return False

# стабилизироваться над голубой полоской
def stabilize_over_cyan_line():
    found, x, y, angle = find_cyan_line(mur.get_image_bottom())

    if found:
        print(x, y, angle)
        if stabilize_x_y_angle(x, y, angle):
            return True
        else:
            return False

# обнаружение голубого круга
def find_cyan_circle(image):
    contours = find_contours(image, (80, 50, 50), (100, 255, 255))

    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 400:
                continue

            ((_, _), (w, h), _) = cv.minAreaRect(contour)
            (_, _), radius = cv.minEnclosingCircle(contour)
            rectangle_area = w * h
            circle_area = radius ** 2 * math.pi
            aspect_ratio = w / h

            if 0.85 <= aspect_ratio <= 1.15:
                if rectangle_area > circle_area:
                    moments = cv.moments(contour)
                    try:
                        x = int(moments['m10'] / moments['m00'])
                        y = int(moments['m01'] / moments['m00'])
                        return True, x, y, area
                    except ZeroDivisionError:
                        return False, 0, 0, 0
    return False, 0, 0, 0

# стабилизироваться над голубым кругом
def stabilize_over_cyan_circle():
```

```
    found, x, y, angle = find_cyan_circle(mur.get_image_bottom())

    if found:
        if stabilize_x_y_angle(x, y, 0):
            return True
        else:
            return False

# прицелиться для выстрела в мишень
def aim_shooting_target():
    # цвет целевой мишени возьмём из списка shooting_colors
    # для выстрела нужно расположиться чуть выше центра мишени
    if (aim_target(context.shooting_colors[-1], 25)):
        context.shooting_colors.pop() # удаляем отработанную мишень из списка
        return True
    else:
        return False

# прицелиться для прохода через мишень
def aim_through_target():
    if (aim_target(context.target_color)):
        return True
    else:
        return False
```

Опишем такие действия, как движение вперёд и назад, остановка движения, ожидание и всплытие.

```
# двигаться вперёд
def go_forward():
    if (context.get_speed() != 30):
        context.set_speed(30)
        return False
    else:
        return True

# двигаться назад
def go_back():
    context.set_depth(2.0)
    if (context.get_speed() != -30):
        context.set_speed(-30)
        return False
    else:
        return True

# ожидание 5 секунд
def wait():
    time.sleep(5)
    return True

# ожидание 10 секунд
def wait_long():
    time.sleep(10)
    return True

# остановить движение
def stop():
    if (context.get_speed() != 0):
        context.set_speed(0)
        return False
    else:
        return True

# всплыть на поверхность
def surface():
    context.set_depth(0)
    mur.set_motor_power(2, 50)
    mur.set_motor_power(3, 50)
    time.sleep(5)
    return True
```

Далее расположен тот код, который выполняется непосредственно при запуске скрипта.

Здесь будут описан алгоритм выполнения миссий, который состоит из ранее написанных функций. Миссия была разбита на подзадачи.

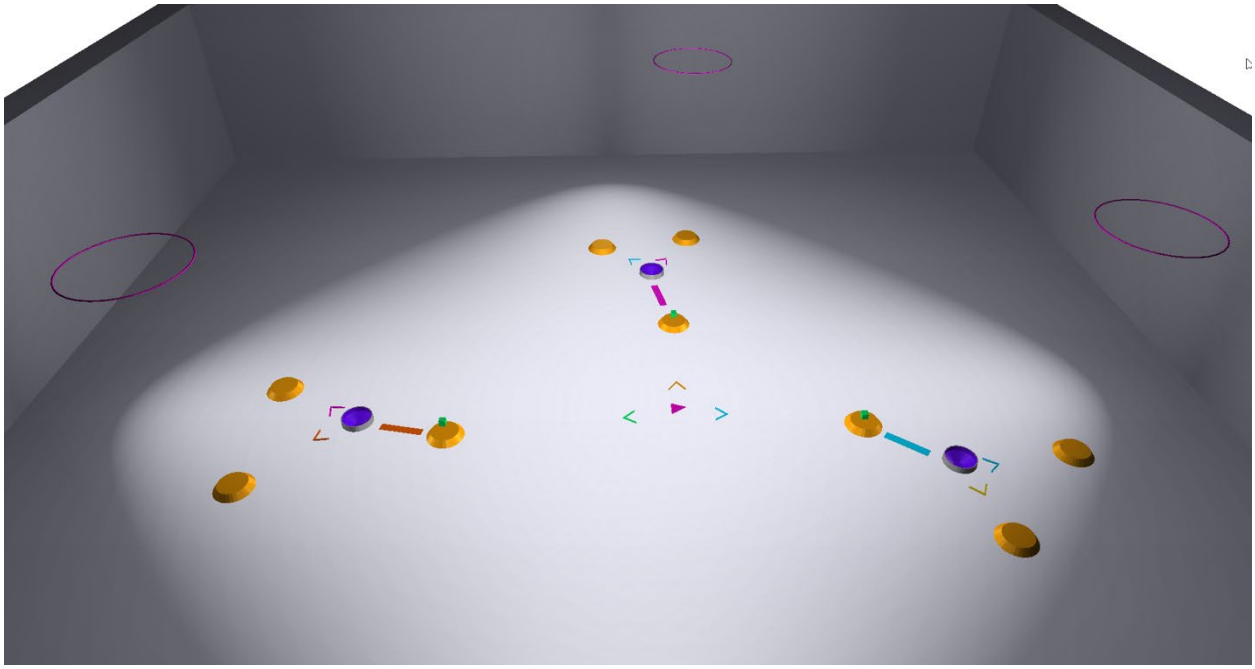
```
# основной код, выполняемый при запуске скрипта
if __name__ == "__main__":
    context.set_depth(2.0)
    context.set_yaw(0.0)

    # разделим большое задание на мелкие задачи

    # выстрел через мишень:
    shoot_target = (
        aim_shooting_target, # сначала прицелиться для выстрела
        shoot,               # затем выстрелить
        go_back,             # начать движение назад
        wait,                # ожидание 5 секунд
        stop,                # после этого остановиться
    )

    # пройти через мишень:
    go_through_target = (
        aim_through_target,  # прицелиться для прохода
        go_forward,          # начать движение вперёд
        wait_long,           # ожидание 10 секунд
        stop,                # остановить движение
        stabilize,           # стабилизироваться (подождать, пока остановимся)
    )

    # основной алгоритм миссии
    # для добавления списка действий (подзадач),
    # используется распаковка значений списка (звёздочка)
    missions = (
        detect_arrow_color,  # определяем цвет стрелки
        stabilize_on_arrow,  # стабилизируемся над стрелкой
        *shoot_target,       # выстрелить в первую мишень из списка
        *shoot_target,       # выстрел во вторую мишень из списка
        *go_through_target,  # пройти через последнюю мишень
        stabilize_over_cyan_line, # стабилизироваться над голубой полоской
        go_forward,          # движение вперед
        stabilize_over_cyan_circle,# стабилизироваться над голубым кругом
        stabilize,           # окончательно стабилизируемся
        surface,             # всплытие
    )

    # задаем список действий миссий
    context.push_mission_list(missions)

    print("start")

    # в цикле проходим каждое действие для выполнения миссии,
    # а также выводим в консоль отладочную информацию
    while (True):
        mission = context.pop_mission()
        print('starting', mission.__name__, '\t\ttime:', context.time)
        while not mission():
            context.process()
            context.time += 1

        if context.get_missions_length() == 0:
            break

    print("done!")

    context.set_speed(0)
    context.set_depth(2.5)

    time.sleep(3)
```

## Задача 7 (тип 2)

1.1 Задача выполняется в симуляторе и среде программирования MUR IDE. Сцены называются: NTI-Task-2_1, NTI-Task-2_2 и NTI-Task-2_3. И доступны по ссылке.

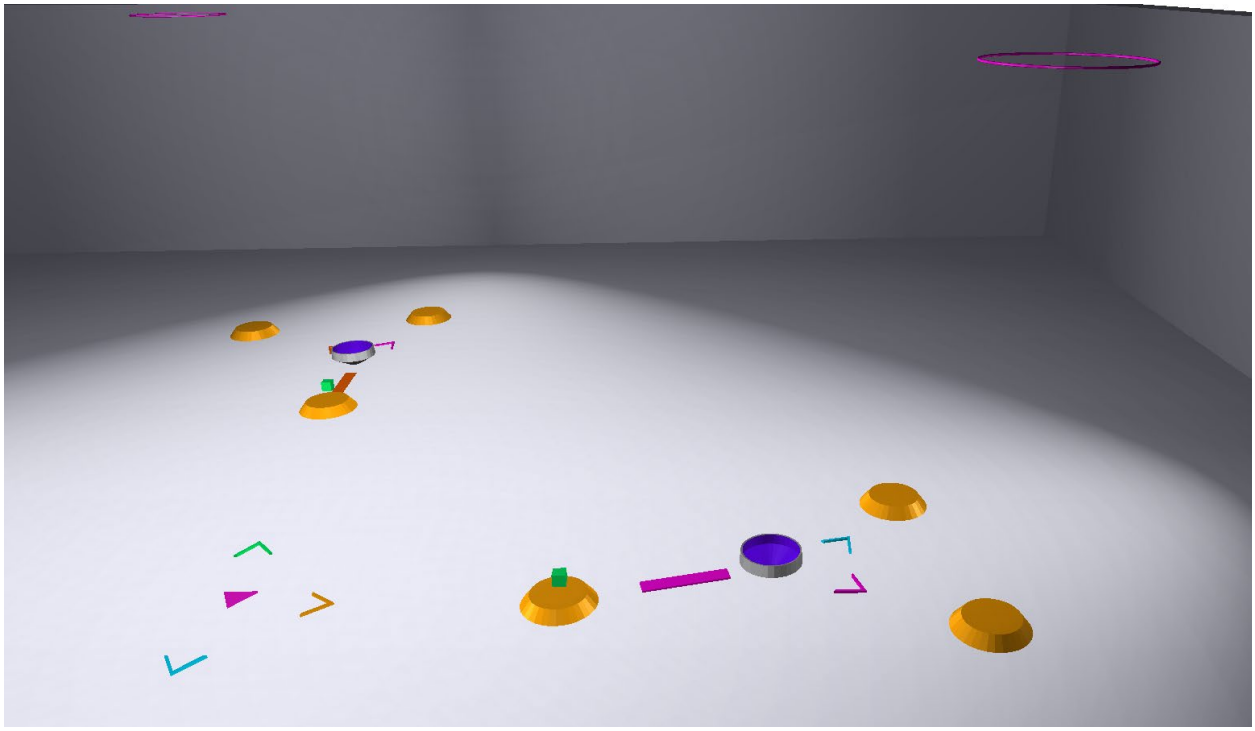
1.2 Необходимо запрограммировать подводного робота, который должен в симуляторе в автономном режиме выполнить под водой ряд задач и всплыть в заданной области.
Время на выполнение задачи — 5 минут.

1.3 Как только робот всплыл или закончилось время, задача считается законченной, фиксируется количество заработанных баллов.



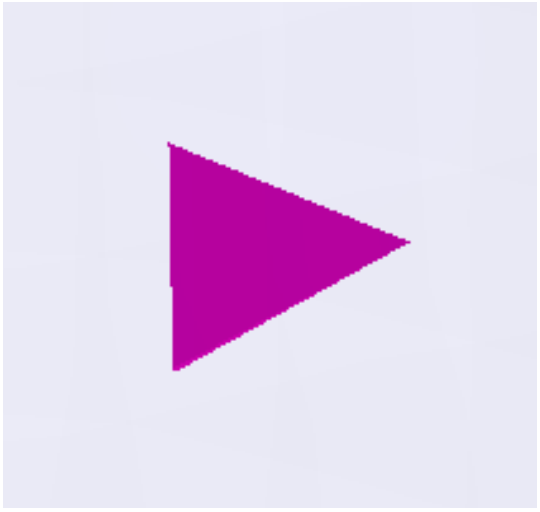
1.4 Задание:

- Робот должен определить **форму** навигационной метки (розовый объекта на старте) (треугольник, квадрат или круг).
- Затем нужно пройти по направлению одного из трёх указателей **в зависимости от формы** розового объекта
(**треугольник** — **зеленый** указатель, **квадрат** — **оранжевый** указатель, **круг** — голубой указатель).
- Каждый указатель показывает направление к платформе с кубом. Данный куб нужно подобрать и положить в синюю корзину, путь к которой указывает **полоска**.
- Далее необходимо пройти к платформе по направлению одного из двух **указателей**, **цвет** которого **совпадает с полоской**.
- Над данной платформой располагается обруч, в пределах которого нужно всплыть.



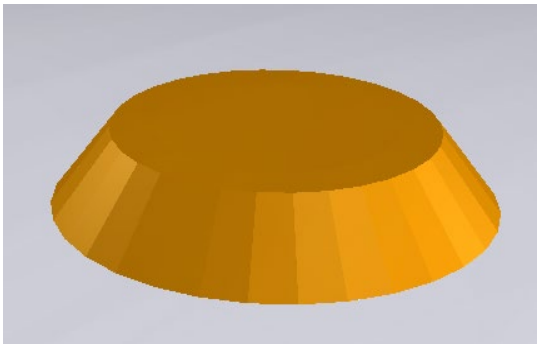
1.5 Описание макетов:

В задаче используются следующие виды объектов: навигационная метка, указатель, платформа, полоска, корзина, обруч.

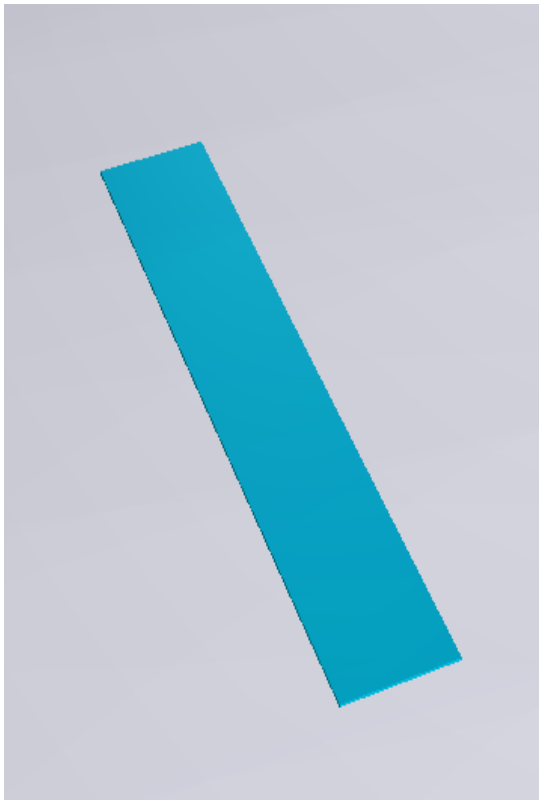


- навигационная метка — располагается на стартовой позиции, и может быть одной из трёх форм: треугольник, квадрат и круг. В зависимости от формы, необходимо выбрать один из трёх указателей (треугольник — зеленый указатель, квадрат — оранжевый указатель, круг — голубой указатель).

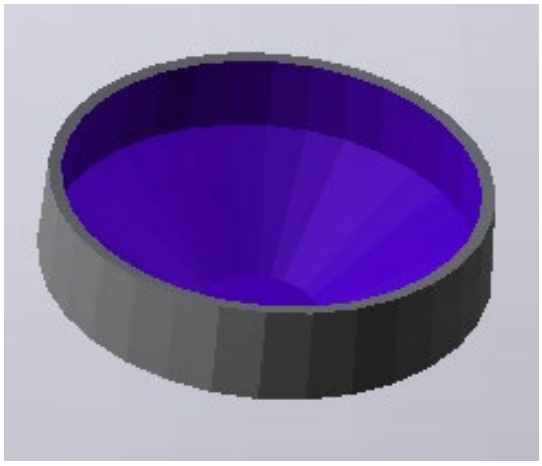
- Указатель — может быть различных цветов. Первые три указателя на стартовой позиции показывают направления к платформам с кубами. Цвет нужного указателя соответствует форме розового объекта. Также имеются два указателя после синей корзины, и они показывают направления к платформам, над одной из которых находится обруч. Необходимо пройти по направлению того указателя, цвет которого совпадает с полоской.



- Платформа — всегда оранжевого цвета. Первая встречающаяся платформа имеет куб, который необходимо подобрать.



- Полоска — указывает направление к синей корзине. В зависимости от цвета полоски необходимо будет выбрать указатель, по которому необходимо следовать. Может быть розового, голубого или красного цвета.



- Корзина — ёмкость, в которой должен оказаться куб. Всегда синего цвета.

- Обруч — расположен на поверхности воды и находится над одной из платформ.

## Решение задачи 7 (тип 2)

Решение (NTI-Simulator-Task2.py) доступно по ссылке https://yadi.sk/d/KrZAbhWM9nDhAw.

Для успешного выполнения задания аппарату в виртуальной среде MUR IDE необходимо определить форму навигационной метки для определение дальнейшего маршрута. Затем нужно пройти по направлению одного из трёх указателей к платформе с кубом. Данный куб нужно подобрать и положить в синюю корзину после чего пройти к платформе по направлению указателя, цвет которого совпадает с полоской. Над данной платформой располагается обруч, в пределах которого нужно всплыть.

Для начала импортируем библиотеки и зададим некоторые исходные данные из задания, как цвета объектов, соответствия форм и цветов объектов для определения маршрута и т.д.

```
import pymurapi as api
import cv2 as cv
import time
import numpy as np
import math

mur = api.mur_init()

# предопределеные диапазоны цветов (HSV)
colors = {
    'green': ((45, 50, 50), (75, 255, 255)),
    'blue': ((130, 50, 50), (140, 255, 255)),
    'cyan': ((80, 50, 50), (100, 255, 255)),
    'orange': ((15, 50, 50), (30, 255, 255)),
    'red': ((0, 50, 50), (15, 255, 255)),
    'magenta': ((140, 50, 50), (160, 255, 255)),
}

# соответствие формы объекта к цвету (согласно заданию)
tag_shape_to_color = {
    'triangle': 'green',
    'square': 'orange',
    'circle': 'cyan',
}

# возможные варианты цветов полоски
line_colors = ['magenta', 'cyan', 'red']
```

Аналогично предыдущему заданию напишем несколько функций для расчётов, PD-регулятор, а также функции для задания курса и глубины погружения робота.

```
# функция для ограничения значения диапазоном
def clamp(value, min_value, max_value):
    if value < min_value:
        return min_value
    if value > max_value:
        return max_value
    return value

# функция для расчёта угла между точками
def angle_between(p1, p2):
    xDiff = p2[0] - p1[0]
```

```
    yDiff = p2[1] - p1[1]
    return math.degrees(math.atan2(yDiff, xDiff) - (np.pi / 2))

# функция для рассчёта расстояния от центра до точки
def length_from_center(x, y):
    return math.sqrt(x ** 2 + y ** 2)
# PD-регулятор
class PDRegulator(object):
    _p_gain = 0.0
    _d_gain = 0.0
    _prev_error = 0.0
    _timestamp = 0

    def __init__(self):
        pass

    def set_p_gain(self, value):
        self._p_gain = value

    def set_d_gain(self, value):
        self._d_gain = value

    def process(self, error):
        timestamp = int(round(time.time() * 1000))

        if timestamp == self._timestamp:
            return 0

        output = self._p_gain * error + self._d_gain / (timestamp - self._timestamp) * (error
- self._prev_error)
        self._timestamp = timestamp
        self._prev_error = error
        return output
# функция для установки курса робота
def keep_yaw(yaw_to_set, speed):
    def clamp_angle(angle):
        if angle > 180.0:
            return angle - 360.0
        if angle < -180.0:
            return angle + 360
        return angle

    try:
        error = mur.get_yaw() - yaw_to_set
        error = clamp_angle(error)
        output = keep_yaw.yaw_regulator.process(error)
        mur.set_motor_power(0, clamp(-output + speed, -100, 100))
        mur.set_motor_power(1, clamp(output + speed, -100, 100))
    except AttributeError:
        keep_yaw.yaw_regulator = PDRegulator()
        keep_yaw.yaw_regulator.set_p_gain(0.8)
        keep_yaw.yaw_regulator.set_d_gain(0.6)

# функция для установки глубины погружения
def keep_depth(depth_to_set):
    try:
        error = mur.get_depth() - depth_to_set
        output = keep_depth.depth_regulator.process(error)
        output = clamp(output, -100, 100)
        mur.set_motor_power(2, output)
        mur.set_motor_power(3, output)
    except AttributeError:
        keep_depth.depth_regulator = PDRegulator()
        keep_depth.depth_regulator.set_p_gain(45)
        keep_depth.depth_regulator.set_d_gain(5)
```

Также как и в предыдущем задании, здесь будет использоваться класс для хранения состояния робота и хода выполнения миссии.

```
# класс для хранения текущего состояния
class AUVContext(object):
    _yaw = 0.0
    _depth = 0.0
    _speed = 0.0
    _side_speed = 0.0
```

```
    _timestamp = 0
    _missions = []
    _min_area = math.inf
    _min_yaw = 0.0
    _stabilization_counter = 0
    time = 0

    def __init__(self):
        pass

    def set_min_circle(self, yaw, area):
        if area < self._min_area:
            self._min_area = area
            self._min_yaw = yaw

    def get_min_circle_yaw(self):
        return self._min_yaw

    def get_yaw(self):
        return self._yaw

    def get_depth(self):
        return self._depth

    def get_speed(self):
        return self._speed

    def get_side_speed(self):
        return self._side_speed

    def set_yaw(self, value):
        self._yaw = value

    def set_depth(self, value):
        self._depth = value

    def set_speed(self, value):
        self._speed = value

    def set_side_speed(self, value):
        self._side_speed = value

    def get_stabilization_counter(self):
        return self._stabilization_counter

    def reset_stabilization_counter(self):
        self._stabilization_counter = 0

    def add_stabilization_counter(self):
        self._stabilization_counter += 1

    def check_stabilization(self, timeout = 3):
        if self._stabilization_counter > timeout:
            return True
        else:
            self.add_stabilization_counter()
            return False

    def push_mission(self, mission):
        self._missions.append(mission)

    def push_mission_list(self, missions):
        for mission in missions:
            self.push_mission(mission)

    def pop_mission(self):
        if len(self._missions) != 0:
            return self._missions.pop(0)
        return {}

    def get_missions_length(self):
        return len(self._missions)

    def process(self):
        timestamp = int(round(time.time() * 1000))
```

```
        if timestamp - self._timestamp > 16:
            keep_yaw(self._yaw, self._speed)
            keep_depth(self._depth)
            mur.set_motor_power(4, self._side_speed)
            self._timestamp = timestamp
        else:
            time.sleep(0.05)

context = AUVContext()
```

Также напишем функции для выполнения таких действий, как развороты, выстрел, движения и т.д.

```
# разворот на 90 градусов
def translate_to_90():
    yaw = mur.get_yaw() + 90
    if yaw < -180:
        yaw += 360
    if yaw > 180:
        yaw -= 360
    context.set_yaw(yaw)
    return True

# разворот на 180 градусов
def translate_to_180():
    yaw = mur.get_yaw() + 180
    if yaw < -180:
        yaw += 360
    if yaw > 180:
        yaw -= 360
    context.set_yaw(yaw)
    return True

# произвести выстрел
def shoot():
    if (context.get_speed() != 0):
        context.set_speed(0)
        return False
    else:
        mur.shoot()
        time.sleep(0.5)
        return True

# движение вперёд
def go_forward():
    if (context.get_speed() != 15):
        context.set_speed(15)
        return False
    else:
        return True

# движение назад
def go_back():
    if (context.get_speed() != -2.5):
        context.set_speed(-2.5)
        return False
    else:
        return True

# ожидание 5 секунд
def wait():
    time.sleep(5)
    return True

# ожидание 3 секунды
def wait_short():
    time.sleep(3)
    return True

# ожидание 12 секунд
def wait_long():
    time.sleep(12)
    return True

# прекратить движение
```

```
def stop():
    if (context.get_speed() != 0):
        context.set_speed(0)
        return False
    else:
        return True

# вспылтие на поверхность
def surface():
    mur.set_motor_power(2, 50)
    mur.set_motor_power(3, 50)
    time.sleep(5)
    return True

# стабилизировать курс и глубину
def stabilize():
    yaw = mur.get_yaw()
    depth = mur.get_depth()

    if abs(yaw - context.get_yaw()) < 1 and abs(depth - context.get_depth()) < 0.3:
        if context.check_stabilization(timeout=5):
            return True
    else:
        context.reset_stabilization_counter()
    return False
```

Для выполнения этого задания также пригодится функция для определения контуров по цвету.

```
# поиск контура по цвету
def find_contours(image, color, approx = cv.CHAIN_APPROX_SIMPLE):
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
    mask = cv.inRange(hsv_image, color[0], color[1])
    contours, _ = cv.findContours(mask, cv.RETR_CCOMP, approx)
    return contours
```

Далее идут функции, активно использующие машинное зрение. Например, это определение цветов обхектов, их углов, позиционирование и стабилизация над объектами и т.д.

```
# расчёт угла прямоугольника
# для определение отклонения от полоски,
# чтобы затем скорректировать курс
def find_rectangle_contour_angle(contour):
    rectangle = cv.minAreaRect(contour)
    box = cv.boxPoints(rectangle)
    box = np.int0(box)
    edge_first = np.int0((box[1][0] - box[0][0], box[1][1] - box[0][1]))
    edge_second = np.int0((box[2][0] - box[1][0], box[2][1] - box[1][1]))

    edge = edge_first
    if cv.norm(edge_second) > cv.norm(edge_first):
        edge = edge_second

    angle = -((180.0 / math.pi * math.acos(edge[0] / (cv.norm((1, 0)) * cv.norm(edge)))) - 90)
    return angle

# определение цвета из списка возможных.
# рассчитывается площадь контура для каждого из цветов,
# а затем выбирается цвет по наибольшему контуру
def detect_color(available_colors):
    image = mur.get_image_bottom()
    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

    areas = {}

    for color in available_colors:
        mask = cv.inRange(hsv_image, colors[color][0], colors[color][1])
        contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)

        biggest_area = 0

        if contours:
```

```
            for contour in contours:
                area = cv.contourArea(contour)
                if area > biggest_area:
                    biggest_area = area

        areas[color] = biggest_area

    color = sorted(areas, key=areas.get, reverse=True)[0]

    if (areas[color] > 5):
        return color
    else:
        return False

# определение направления стрелки
def detect_arrow_angle(image, contour):
    (arrow_center_x, arrow_center_y), radius = cv.minEnclosingCircle(contour)
    moments = cv.moments(contour)

    try:
        arrow_direction_x = int(moments['m10'] / moments['m00'])
        arrow_direction_y = int(moments['m01'] / moments['m00'])

        arrow_angle = (angle_between((arrow_direction_x, arrow_direction_y), (arrow_center_x,
arrow_center_y)))
        context.set_yaw(mur.get_yaw() + arrow_angle)

        target_x = arrow_center_x - (320 / 2)
        target_y = arrow_center_y - (240 / 2)
        length = math.sqrt(target_x ** 2 + target_y ** 2)

        return arrow_angle, target_x, target_y, length
    except:
        return False, False, False, False

# стабилизироваться над стрелкой (с учетом направления стрелки)
def stabilize_on_arrow(color):
    image = mur.get_image_bottom()

    contours = find_contours(image, color, cv.CHAIN_APPROX_SIMPLE)

    if contours:
        for contour in contours:
            (arrow_angle, target_x, target_y, length) = detect_arrow_angle(image, contour)

            if arrow_angle != False:
                try:
                    output_forward = stabilize_on_arrow.forward_regulator.process(target_y)
                    output_side = stabilize_on_arrow.side_regulator.process(target_x)

                    output_forward = clamp(int(output_forward), -50, 50)
                    output_side = clamp(int(output_side), -50, 50)

                    context.set_speed(-output_forward)
                    context.set_side_speed(-output_side)

                    if abs(arrow_angle) < 2 and length < 7:
                        if context.check_stabilization(timeout=2):
                            return True
                    else:
                        context.reset_stabilization_counter()

                except AttributeError:
                    stabilize_on_arrow.forward_regulator = PDRegulator()
                    stabilize_on_arrow.forward_regulator.set_p_gain(0.5)
                    stabilize_on_arrow.forward_regulator.set_d_gain(0.1)

                    stabilize_on_arrow.side_regulator = PDRegulator()
                    stabilize_on_arrow.side_regulator.set_p_gain(0.5)
                    stabilize_on_arrow.side_regulator.set_d_gain(0.1)

    return False

# стабилизироваться по координатам, а также по углу
def stabilize_x_y_angle(x, y, angle):
```

```
    y_center = y - (240 / 2)
    x_center = x - (320 / 2)

    try:
        length = math.sqrt(x_center ** 2 + y_center ** 2)
        if length < 2.0:
            if context.check_stabilization(timeout=10):
                return True
        else:
            context.reset_stabilization_counter()

        output_forward = stabilize_x_y_angle.forward_regulator.process(y_center)
        output_side = stabilize_x_y_angle.side_regulator.process(x_center)

        output_forward = clamp(int(output_forward), -10, 10)
        output_side = clamp(int(output_side), -10, 10)

        context.set_speed(-output_forward)
        context.set_side_speed(-output_side)

        context.set_yaw(mur.get_yaw() + angle)
    except AttributeError:
        stabilize_x_y_angle.forward_regulator = PDRegulator()
        stabilize_x_y_angle.forward_regulator.set_p_gain(0.5)
        stabilize_x_y_angle.forward_regulator.set_d_gain(0.1)

        stabilize_x_y_angle.side_regulator = PDRegulator()
        stabilize_x_y_angle.side_regulator.set_p_gain(0.5)
        stabilize_x_y_angle.side_regulator.set_d_gain(0.1)
    return False

# обнаружение полоски определенного цвета
def find_line(image, color):
    contours = find_contours(image, color)
    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 300:
                continue

            ((_, _), (w, h), _) = cv.minAreaRect(contour)
            aspect_ratio = max(w, h) / min(w, h)

            moments = cv.moments(contour)
            try:
                x = int(moments['m10'] / moments['m00'])
                y = int(moments['m01'] / moments['m00'])
                return True, x, y, find_rectangle_contour_angle(contour)
            except ZeroDivisionError:
                return False, 0, 0, 0
    return False, 0, 0, 0

# обнаружение объекта определенного цвета
def find_colored_object(image, color):
    contours = find_contours(image, color)
    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 50:
                continue

            ((_, _), (w, h), _) = cv.minAreaRect(contour)

            moments = cv.moments(contour)

            try:
                x = int(moments['m10'] / moments['m00'])
                y = int(moments['m01'] / moments['m00'])
                return True, x, y, area
            except ZeroDivisionError:
                return False, 0, 0, 0
    return False, 0, 0, 0

# стабилизироваться над полоской
def stabilize_over_line(color):
```

```
    found, x, y, angle = find_line(mur.get_image_bottom(), color)

    if found:
        if stabilize_x_y_angle(x, y, angle):
            return True
        else:
            return False

# стабилизироваться над кубом
def stabilize_over_box():
    if (context.get_depth() != 3.1):
        context.set_depth(3.1)
        return False

    found, x, y, area = find_line(mur.get_image_bottom(), colors['green'])

    if found:
        if stabilize_x_y_angle(x, y + 7, 0):
            return True
        else:
            return False

# обнаружение круга
def find_circle(image, color):
    contours = find_contours(image, color)

    if contours:
        for contour in contours:
            area = cv.contourArea(contour)
            if abs(area) < 400:
                continue

            ((_, _), (w, h), _) = cv.minAreaRect(contour)
            (_, _), radius = cv.minEnclosingCircle(contour)
            rectangle_area = w * h
            circle_area = radius ** 2 * math.pi
            aspect_ratio = w / h

            if 0.85 <= aspect_ratio <= 1.15:
                if rectangle_area > circle_area:
                    moments = cv.moments(contour)
                    try:
                        x = int(moments['m10'] / moments['m00'])
                        y = int(moments['m01'] / moments['m00'])
                        return True, x, y, area
                    except ZeroDivisionError:
                        return False, 0, 0, 0
    return False, 0, 0, 0

# стабилизироваться над кругом
def stabilize_over_circle(color):
    found, x, y, angle = find_circle(mur.get_image_bottom(), color)

    if found:
        if stabilize_x_y_angle(x, y, 0):
            return True
        else:
            return False
```

Для определения формы объекта используется подсчёт площади вписанных фигур, которые сравниваются с площадью самой фигуры. По наименьшей разнице площадей можно определить, на какую из геометрических фигур объект наиболее похож.

```
# определение формы объекта определенного цвета
def detect_shape(image, color):
    contours = find_contours(image, color)

    if contours:
        for contour in contours:

            tag_area = cv.contourArea(contour)
            if (tag_area < 100):
                continue
```

```
            # после того, как найден контур подходящего цвета,
            # нужно подсчитать площади вписанных в него
            # треугольника, квадрата и круга.
            # форма объекта определяется по наибольшему
            # совпадению площади вписанной фигуры.

            (circle_x, circle_y), circle_radius = cv.minEnclosingCircle(contour)
            circle_area = circle_radius ** 2 * math.pi

            rectangle = cv.minAreaRect(contour)
            box = cv.boxPoints(rectangle)
            box = np.int0(box)
            rectangle_area = cv.contourArea(box)

            triangle = cv.minEnclosingTriangle(contour)[1]
            triangle = np.int0(triangle)
            triangle_area = cv.contourArea(triangle)

            areas = {
                'circle': circle_radius ** 2 * math.pi,
                'square': cv.contourArea(box),
                'triangle': cv.contourArea(triangle),
            }

            tag_shapes = list(areas.keys())
            shapes_areas = np.array(list(areas.values()))
            difference = abs(shapes_areas - tag_area)
            arg = np.argmin(difference)
            tag_shape = tag_shapes[arg]

            return tag_shape

    return False

# распознать навигационную метку
def detect_tag_shape():
    image = mur.get_image_bottom()
    tag_shape = detect_shape(image, colors['magenta'])
    if tag_shape != False:
        context.tag_shape = tag_shape
        # в соответствии с условиями задания,
        # определяем цвет стрелки, по которой
        # нужно следовать
        context.first_arrow_color = colors[tag_shape_to_color[tag_shape]]
        print(tag_shape, '- follow', tag_shape_to_color[tag_shape], 'arrow')
        return True
    else:
        return False

# стабилизироваться на первой стрелкой, по которой
# нужно следовать (используя ранее определенный цвет)
def stabilize_on_first_arrow():
    return stabilize_on_arrow(context.first_arrow_color)

# обнаружение оранжевого круга
def find_orange_circle():
    image = mur.get_image_bottom()
    found, x, y, area = find_circle(image, colors['orange'])

    if found:
        return True
    else:
        return False

# стабилизироваться над оранжевым кругом
def stabilize_over_orange_circle():
    return stabilize_over_circle(colors['orange'])

# установить подходящую глубину
# для захвата куба манипулятором
def set_grabbing_depth():
    mur.open_grabber()

    if (context.get_depth() != 3.62):
        context.set_depth(3.62)
```

```
        return False
    else:
        return True

# схватить куб
def grab_box():
    mur.close_grabber()
    return True

# отпустить куб
def ungrab_box():
    mur.open_grabber()
    return True

# определить цвет полоски
def detect_line_color():
    color = detect_color(line_colors)
    if color:
        context.line_color = color
        return True
    else:
        return False

# установить обычную глубину (после захвата куба)
def set_default_depth():
    if (context.get_depth() != 3.0):
        context.set_depth(3.0)
        return False
    else:
        return True

# обнаружение синей корзины (круглой формы)
def find_blue_bin():
    image = mur.get_image_bottom()
    found, x, y, area = find_circle(image, colors['blue'])

    if found:
        return True
    else:
        return False

# стабилизироваться над синей корзиной
def stabilize_over_blue_bin():
    return stabilize_over_circle(colors['blue'])

# стабилизироваться над второй полоской (ранее определенный цвет)
def stabilize_over_target_line():
    context.set_depth(2.5)
    return stabilize_over_line(colors[context.line_color])

# стабилизироваться над второй стрелкой
def stabilize_on_second_arrow():
    return stabilize_on_arrow(colors[context.line_color])
```

Далее идёт код, который выполняется при запуске скрипта. Здесь описан алгоритм выполнения миссии, которая разбита на подзадачи.

```
# основной код, выполняемый при запуске скрипта
if __name__ == "__main__":
    context.set_depth(1.8)
    context.set_yaw(0.0)

    # определим подзадачи нашей миссии

    # для того, чтобы добраться до платформы с кубом, нужно:
    go_to_box_platform = (
        detect_tag_shape,           # определить маршрут по навигационной метке
        stabilize_on_first_arrow,   # стабилизироваться над первой стрелкой
        go_forward,                 # двигаться вперёд
    )

    # поиск и захват куба
    find_and_grab_box = (
        find_orange_circle,         # обнаружить оранжевый круг (мы все еще движемся)
        stabilize_over_orange_circle,# стабилизироваться над этим кругом
```

```
        stabilize_over_box,          # найдя круг, можно найти и стабилизироваться над кубом
        stabilize,                   # окончательно стабилизируем своё положение перед
захватом
        set_grabbing_depth,          # погружаемся ближе к кубу
        wait_short,                  # ожидание 3 секунды
        stabilize,                   # стабилизируем положение
        wait_short,                  #
        grab_box,                    # захват куба манипулятором
        wait_short,                  #
        set_default_depth,           # принимаем нормальную глубину
        go_forward,                  # двигаемся вперед
        stabilize,                   # стабилизируем курс и глубину
        wait_short,                  #
        stop,                        # останавливаем движение после 3 секунд
    )

    # найти корзину и бросить куб
    find_bin_and_throw_box = (
        detect_line_color,           # определяем цвет полоски (для определения маршрута)
        stabilize_over_target_line,  # стабилизируемся над нужной полоской (определенного
цвета)
        go_forward,                  # двигаемся вперёд
        find_blue_bin,               # находим синюю корзину
        stabilize_over_blue_bin,     # стабилизируемся над корзиной
        ungrab_box,                  # отпускаем куб, чтобы он упал в корзину
    )

    # добраться до последней платформы (над которой обруч)
    go_to_final_platform = (
        go_forward,                  # двигаемся вперёд
        wait_short,                  # ожидание 3 секунды
        stop,                        # остановка движения
        stabilize,                   # стабилизируем положение
        stabilize_on_second_arrow,   # стабилизируемся над второй стрелкой (цвет был ранее
определен)
        go_forward,                  # двигаемся вперед
        find_orange_circle,          # находим оранжевый круг
        stabilize_over_orange_circle,# стабилизируемся над этим кругом
        stabilize,                   # окончательно стабилизируем курс
    )

    missions = (
        *go_to_box_platform,         # сначала нужно добраться до платформы с кубом
        *find_and_grab_box,          # затем найти и схватить куб
        *find_bin_and_throw_box,     # найти корзину и отпустить куб
        *go_to_final_platform,       # добраться до последнй платформы
        surface,                     # и в конце всплыть
    )

    # задаем список действий миссий
    context.push_mission_list(missions)

    print("start")

    # в цикле проходим каждое действие для выполнения миссии,
    # а также выводим в консоль отладочную информацию
    while (True):
        mission = context.pop_mission()
        print('starting', mission.__name__, '\ttime:', context.time)
        while not mission():
            context.process()
            context.time += 1

        if context.get_missions_length() == 0:
            break

    print("done!")

    context.set_speed(0)
    context.set_depth(2.5)
```

time.sleep(3)

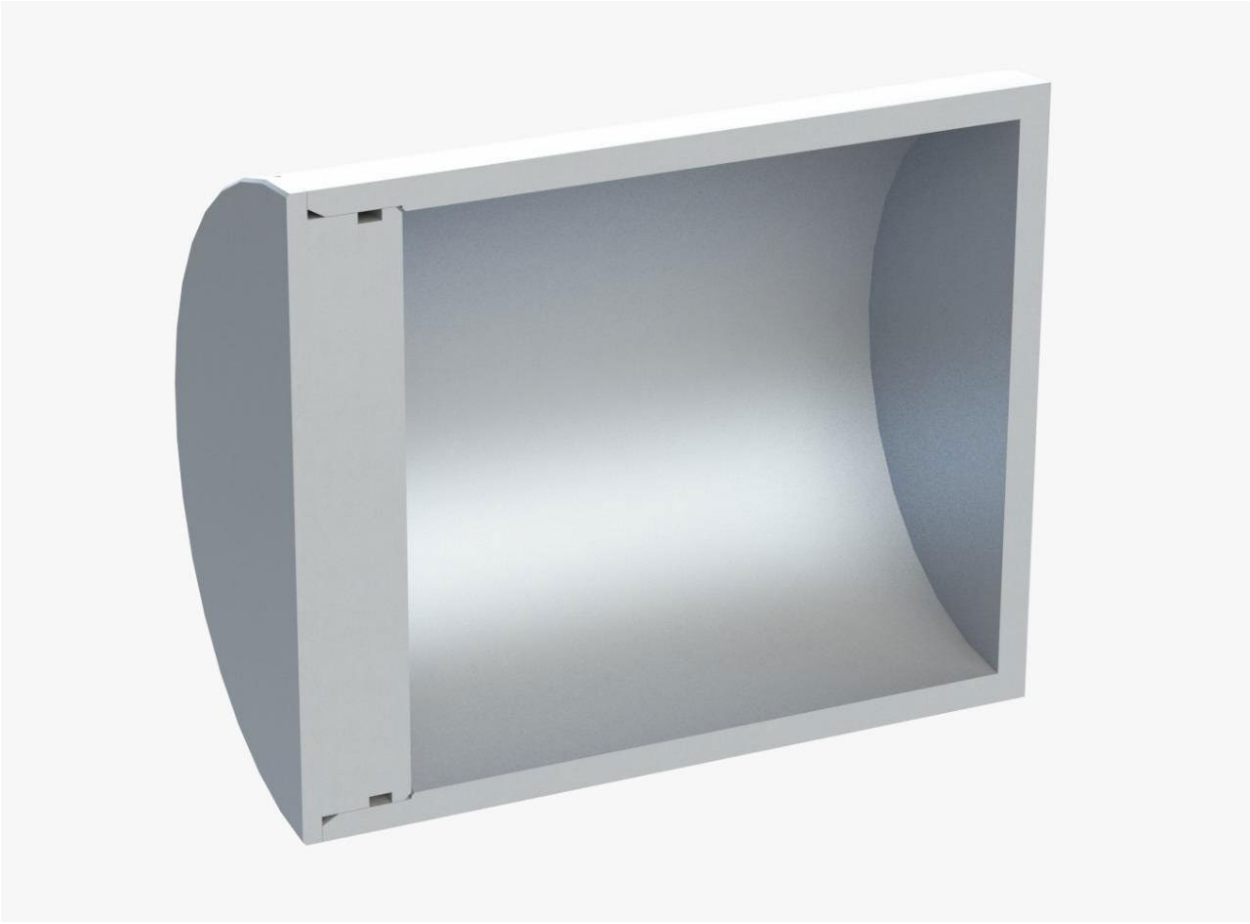
## Конструирование

### Задача 8 (тип 1 или тип 2)

Дано:

- Алюминиевый пруток длиной 200 мм, диаметром 100 мм.
- Алюминиевый пруток длиной 40 мм, диаметром 100 мм
- Резиновое уплотнительное кольцо сечением 2,5 мм и произвольного диаметра (вам необходимо самостоятельно подобрать)

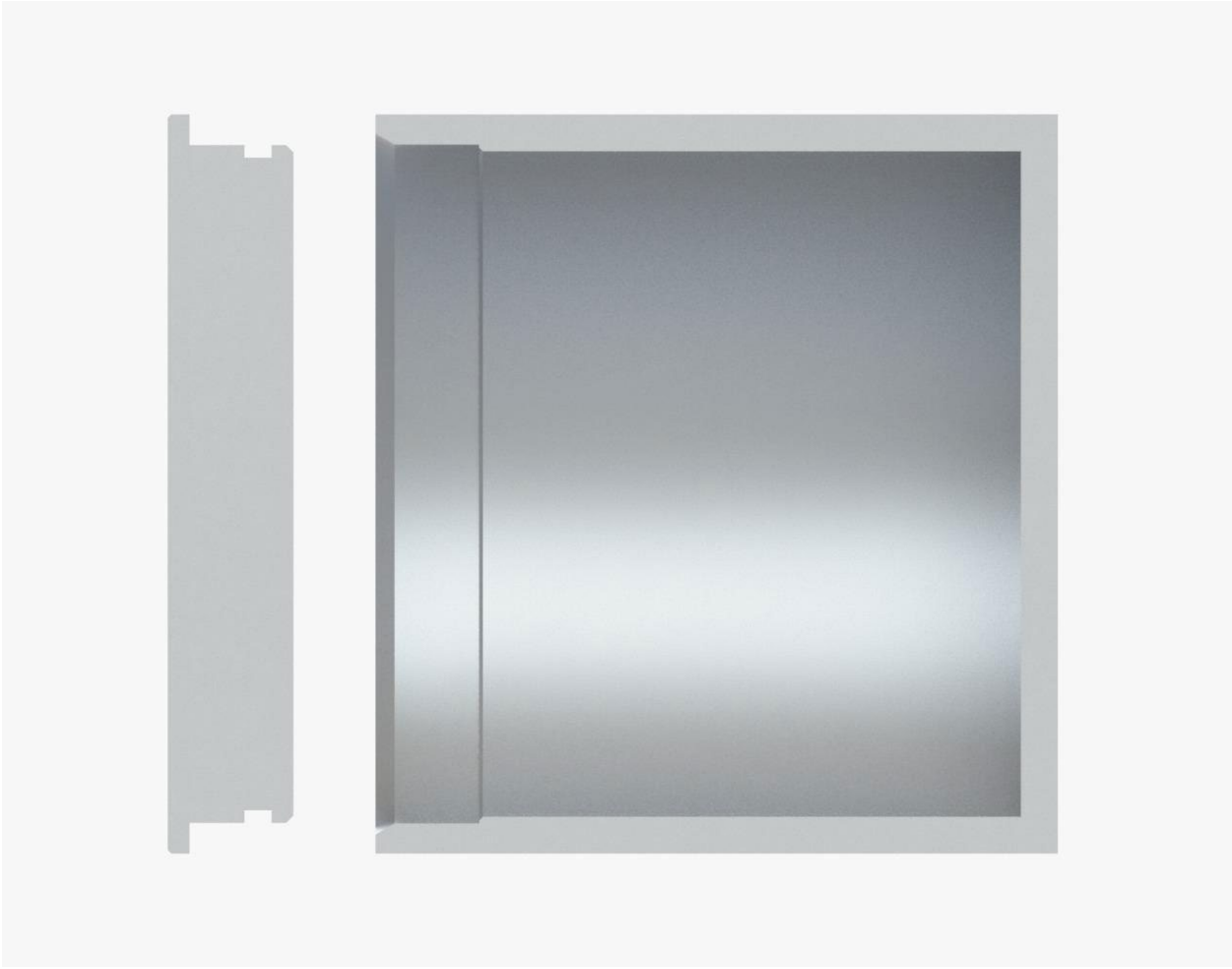
Необходимо спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. Пример приведен на картинке ниже. Для герметизации необходимо использовать уплотнительное кольцо сечением 2,5 мм. Толщина стенок и дна колбы должна быть от 3 до 5 мм.

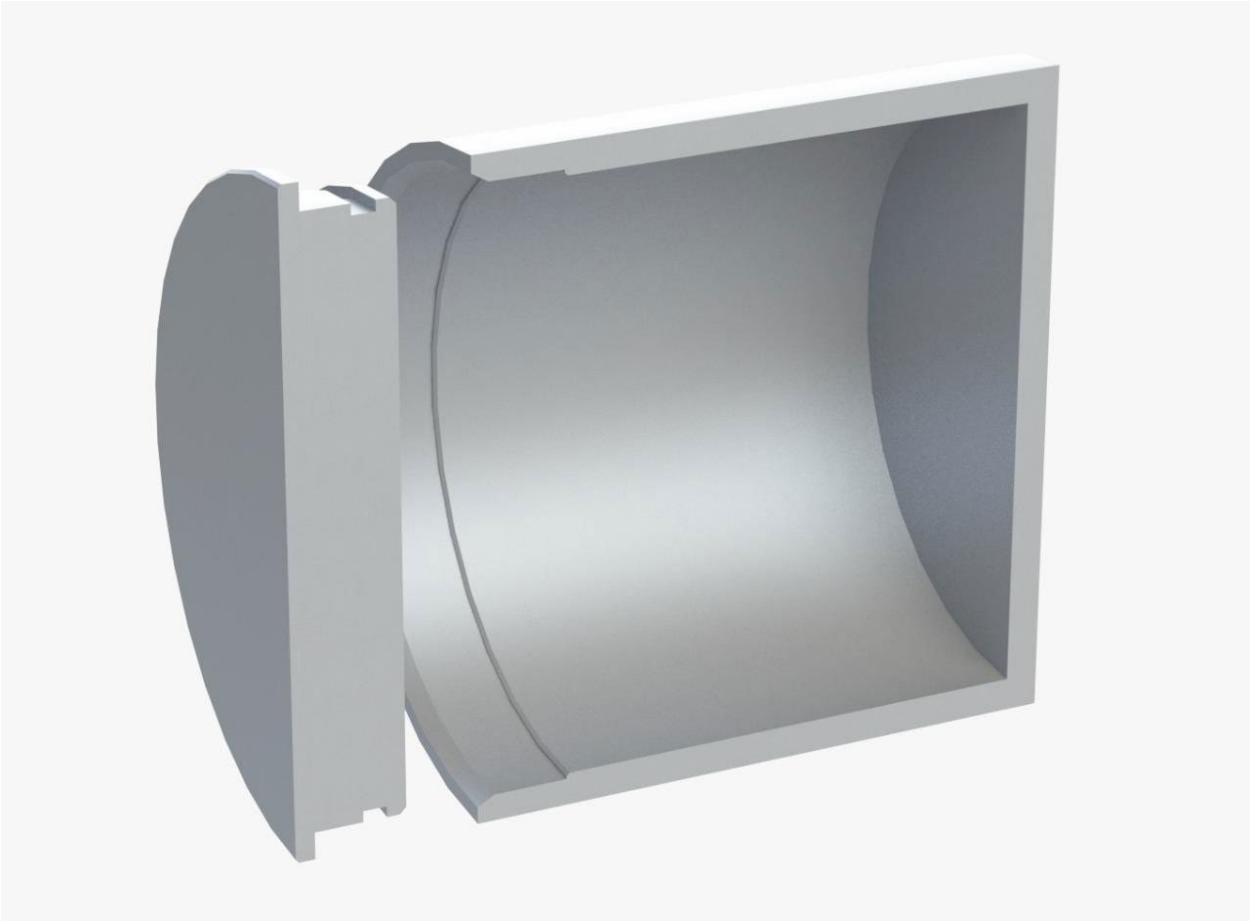


Кружковое движение

НТО

ЦРР





Участникам необходимо правильно рассчитать:

- Диаметр уплотнительного кольца - 10 баллов.
- Ширину канавки в крышке - 10 баллов.
- Глубину канавки в крышке - 10 баллов.
- Внутренний диаметр колбы в месте герметизации - 10 баллов.

Для расчетов необходимо воспользоваться ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств". Особое внимание при изучении ГОСТа уделите приложению.

Сделайте чертеж колбы - от 0 до 15 баллов

- Должен содержать разрез детали.
- Должен быть сохранен в формате pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали.

Сделайте чертеж крышки - от 0 до 15 баллов

- Должен содержать разрез детали.
- Должен быть сохранен в формате pdf.
- Может быть выполнен в любой программе.

- Должны быть проставлены все размеры, необходимые для изготовления детали.

Сделайте 3Д-модель крышки и колбы - от 0 до 30 баллов

- Должна быть выполнена в любой программе. Рекомендуем использовать Autodesk Inventor.
- Должна быть сохранена в формате step.

Решением данной задачи будет являться 3 файла:

- pdf-файл с чертежом колбы;
- pdf-файл с чертежом крышки;
- step-файл с моделью колбы и крышки.

В ответе необходимо указать результаты расчетов из пункта 1.3 Пример:
Диаметр уплотнительного кольца = 1
Ширина канавки в крышке = 2
Глубина канавки в крышке = 3
Внутренний диаметр колбы в месте герметизации = 4

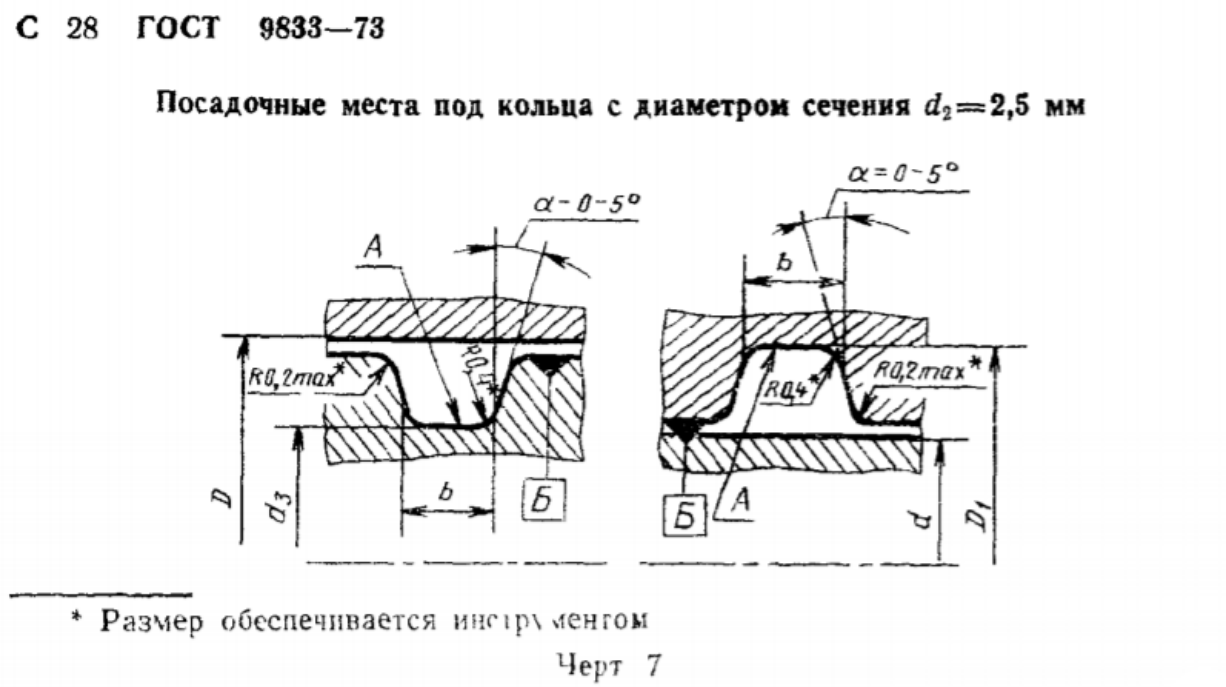
Кружковое движение НТО ЦРР

### Решение задачи 8 (тип 1 или тип 2)

1. Подбираем необходимый диаметр уплотнительного кольца:
   - Исходя из ограничений по толщине стенки и размеров заготовок из дано: 100 – 3*2 = 94мм – максимальный внутренний диаметр колбы (вторая цифра в типоразмере кольца, далее $D$, должна быть $\leq 94$), он же является внутренним диаметром колбы в месте герметизации.
   - Открываем ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" листаю до «Кольца уплотнительные сечением 2,5»
   - На стр. 7(6) определяю, что любой типоразмер кольца, меньший 090-95-25* подходит для проектирования корпуса, но исходя из логики «чем больше корпус, тем больше в него можно положить чего-либо для защиты этого чего-то от контакта с водой», выбираю для дальнейшего проектирования кольцо 088-092-25 с диаметром 86,5 мм. Диаметр уплотнительного кольца = 86,5 мм + 10 баллов.

*Это кольцо тоже можно использовать, т.к. по ГОСТу толщина стенки в месте посадки крышки может быть меньше 3 мм, но не меньше 1,5мм.

2. На стр. 29 (28) ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" нахожу таблицу с расчётами для кольца сечением 2,5 мм и вижу 2 чертежа:




Смотрим на рендеры (картинки) с образцом корпуса на странице индивидуального задания для конструкторов на Stepik'е и понимаем, что нас интересует только левая картинка, т.к. установка кольца происходит в крышке (чертёж слева), а не на распор в корпусе (чертёж справа).

3. По таблице ниже находим $D, d_3, b$ напротив 088-092-25 в столбце «Неподвижное соединение»:

$D = 92$ мм, $d_3 = 88{,}3$ мм, $b = 3{,}6$ мм. + 20 баллов.

$D$ – диаметр колбы в месте герметизации;

$b$ – ширина канавки в крышке;

$\frac{D-d_3}{2} = 1{,}85$ мм – глубина канавки в крышке.

Расчёт окончен:

Диаметр уплотнительного кольца 88,5 мм– 10 баллов;

Ширину канавки в крышке 3,6 мм - 10 баллов;

Глубину канавки в крышке 1,85 мм - 10 баллов;

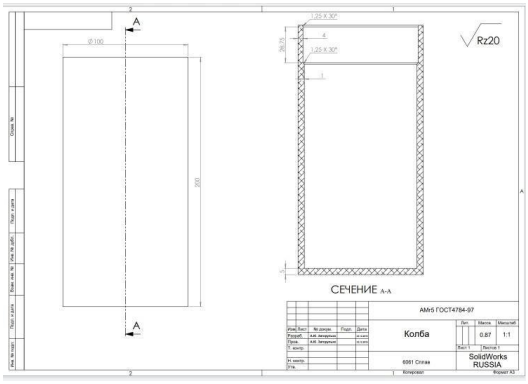
Внутренний диаметр колбы в месте герметизации 92 мм - 10 баллов.

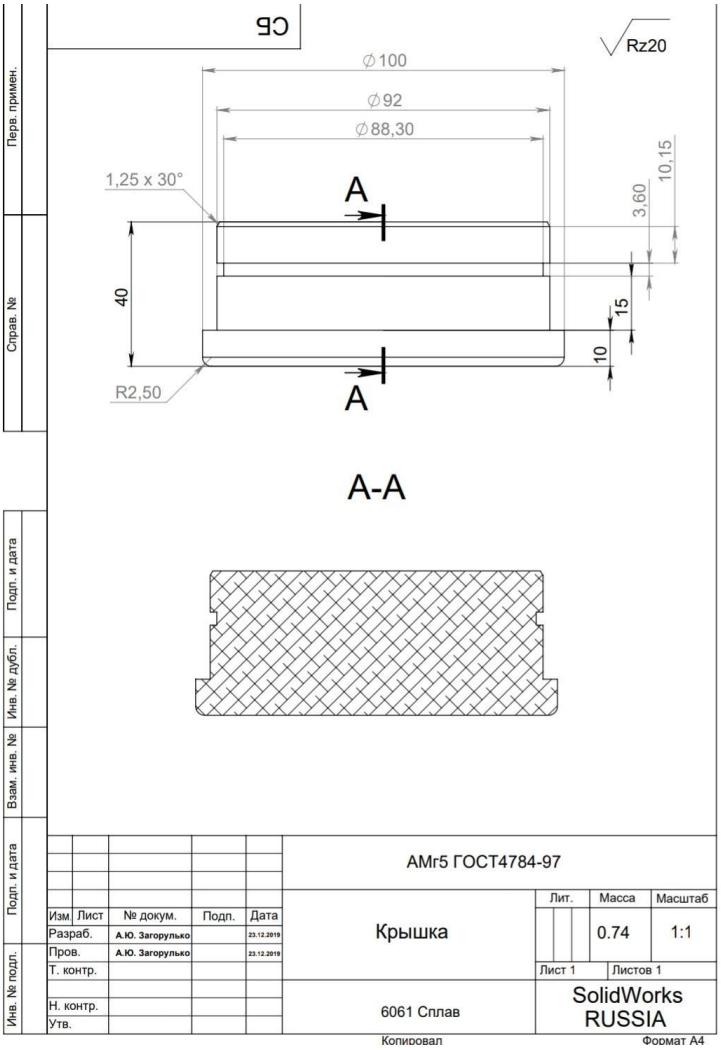
Чертёж.

Рассчитанные по ГОСТу величины являются гарантом герметичности корпуса, при построении чертежей эти размеры нужно обязательно использовать.

Построение и оформление чертежа выполняется по ГОСТ 2.304-68

Пример:





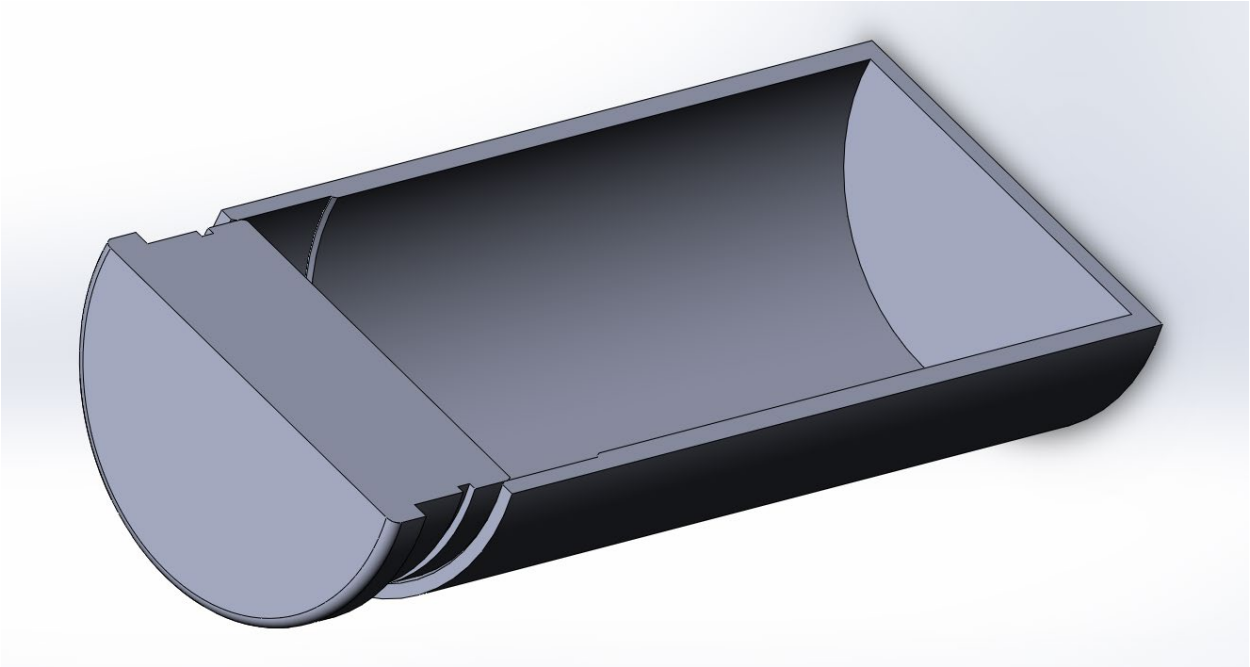

3д модель.

3д модели частей герметичного корпуса строятся отдельно в любой CAD-программе, согласно чертежам, а затем объединяются в сборку.

Кружковое движение

НТО

ЦРР



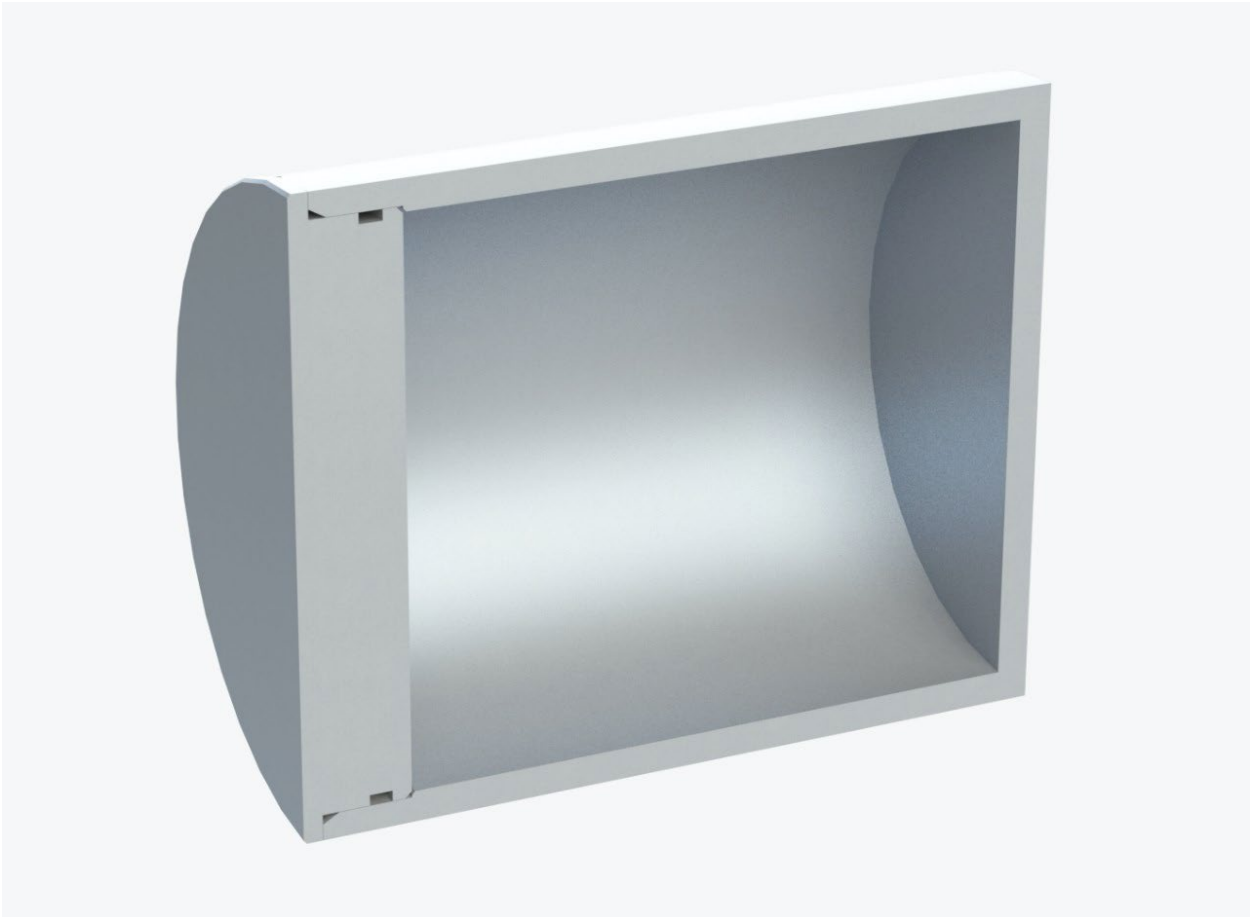
## Задача 9 (тип 1 или тип 2)

1.1. Дано:

- Алюминиевый пруток длиной 60 мм, диаметром 60 мм.
- Алюминиевый пруток длиной 40 мм, диаметром 60 мм
- Резиновое уплотнительное кольцо сечением 2,5 мм и произвольного диаметра (вам необходимо самостоятельно подобрать)

Прутки считать проточенными и подогнанными в размер.

1.2. Необходимо спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. Пример приведен на картинке ниже (модель корпуса без резьбового соединения с канавкой под радиальное уплотнение). При этом крышка и колба должны соединяться с применением резьбового соединения. При проектировании предусмотрите возможность разбора корпуса (выкручивания крышки) без применения инструмента (руками).



Для герметизации необходимо использовать уплотнительное кольцо сечением 2,5 мм. Толщина стенок и дна колбы должна быть минимум 2 мм.

В корпус должен помещаться и извлекаться куб со стороной 33 мм, обязательно оставьте зазор до внутренних стенок колбы.

1.3. Участникам необходимо правильно рассчитать:

- Диаметр уплотнительного кольца
- Ширину канавки в крышке
- Диаметр для посадки кольца в крышке
- Внутренний диаметр колбы в месте герметизации

Для расчетов необходимо воспользоваться ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств". Особое внимание при изучении ГОСТа уделите приложению, а также (с.55).

1.4. Сделайте чертеж колбы

- Должен содержать разрез детали.
- Должен быть сохранен в формате .pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

1.5. Сделайте чертеж крышки

- Должен содержать разрез детали.
- Должен быть сохранен в формате .pdf.
- Может быть выполнен в любой программе.
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

Исчерпывающие требования к чертежам указаны в ГОСТ 2.052-2006 Тщательно изучите пункт «Нормативные ссылки», при переходе по гиперссылкам вы найдёте ответы на все вопросы по оформлению чертежей.

ГОСТ 2.311-68 «ИЗОБРАЖЕНИЕ РЕЗЬБЫ»

Для основной надписи используйте форму 2а, её нужно также заполнить в соответствии с ГОСТами.

Предельные отклонения и допуски указывайте только там, где это необходимо.

Строгое соответствие всем пунктам ГОСТа 2.052-2006 и ГОСТам, на которые он ссылается, не нужно, достаточно соблюдение тех, которые от вас требуются (Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.)

ГОСТ 24705-2004 «Основные нормы взаимозаменяемости РЕЗЬБА МЕТРИЧЕСКАЯ Основные размеры»

ГОСТ 11708-82 «РЕЗЬБА Термины и определения»

ГОСТ 10549-80 «ВЫХОД РЕЗЬБЫ Сбеги, недорезы, проточки и фаски»

1.6 Сделайте 3Д-модель сборки крышки и колбы

- Должна быть выполнена в любой программе. Рекомендуем использовать Autodesk Inventor.
- Должна быть сохранена в формате .step.
- Должна быть выполнена проточка в колбе для внутренней резьбы.
- В деталях должны быть предусмотрены все необходимые фаски и скругления острых граней для безопасного использования.

1.7. Решением данной задачи будет являться 3 файла:

- pdf-файл с чертежом колбы;
- pdf-файл с чертежом крышки;
- step-файл с моделью сборки колба + крышка.

В ответе необходимо указать результаты расчетов из пункта 1.3 в мм. Пример:

Диаметр уплотнительного кольца = 1.

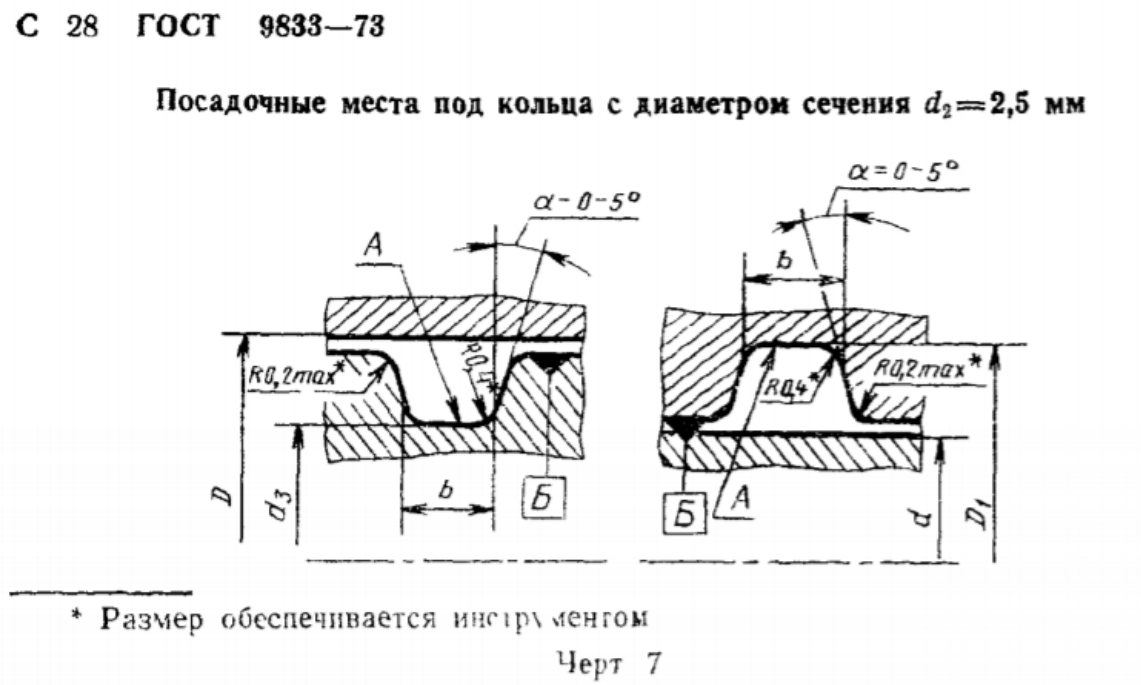
Ширину канавки в крышке = 2.

Диаметр для посадки кольца в крышке = 3.

Внутренний диаметр колбы в месте герметизации = 4.

## Решение задачи 9 (тип 1 или тип 2)

1. Подбираем необходимый диаметр уплотнительного кольца:
   - Исходя из ограничений по толщине стенки, размеров заготовок и в корпус должен помещаться и извлекаться куб со стороной 33 мм из дано: 60 – 2*2 = 56 мм – максимальный внутренний диаметр колбы, $d = a\sqrt{2}$ =33* 1,4 = 46,2 мм – минимальный внутренний диаметр колбы (формула описанной окружности около квадрата), вторая цифра в типоразмере кольца, далее $D$, должна быть 47 – 56 мм.
   - Открываем ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" листаю до «Кольца уплотнительные сечением 2,5»
   - На стр. 6 определяю, что любой типоразмер кольца, от 043-047-25 до 052-056-25 подходит для проектирования корпуса, но исходя из логики «чем больше корпус, тем больше в него можно положить чего-либо для защиты этого чего-то от контакта с водой», выбираю для дальнейшего проектирования кольцо 052-056-25 с диаметром 51 мм. Диаметр уплотнительного кольца = 51 мм + 10 баллов.
2. На стр. 29 (28) ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" нахожу таблицу с расчётами для кольца сечением 2,5 мм и вижу 2 чертежа:




   Смотрим на изображение с образцом корпуса на странице индивидуального задания для конструкторов на Stepik’е и понимаем, что нас интересует только левая картинка, т.к. установка кольца происходит в крышке (чертёж слева), а не на распор в корпусе (чертёж справа).
3. По таблице ниже находим $D, d_3, b$ напротив 052-056-25 в столбце «Неподвижное соединение»:

$D = 56$ мм, $d_3 = 52{,}3$ мм, $b = 3{,}6$ мм. + 30 баллов.
$D$ – внутренний диаметр колбы в месте герметизации;
$b$ – ширина канавки в крышке;
$d_3$ –диаметр для посадки кольца в крышке

Расчёт окончен:

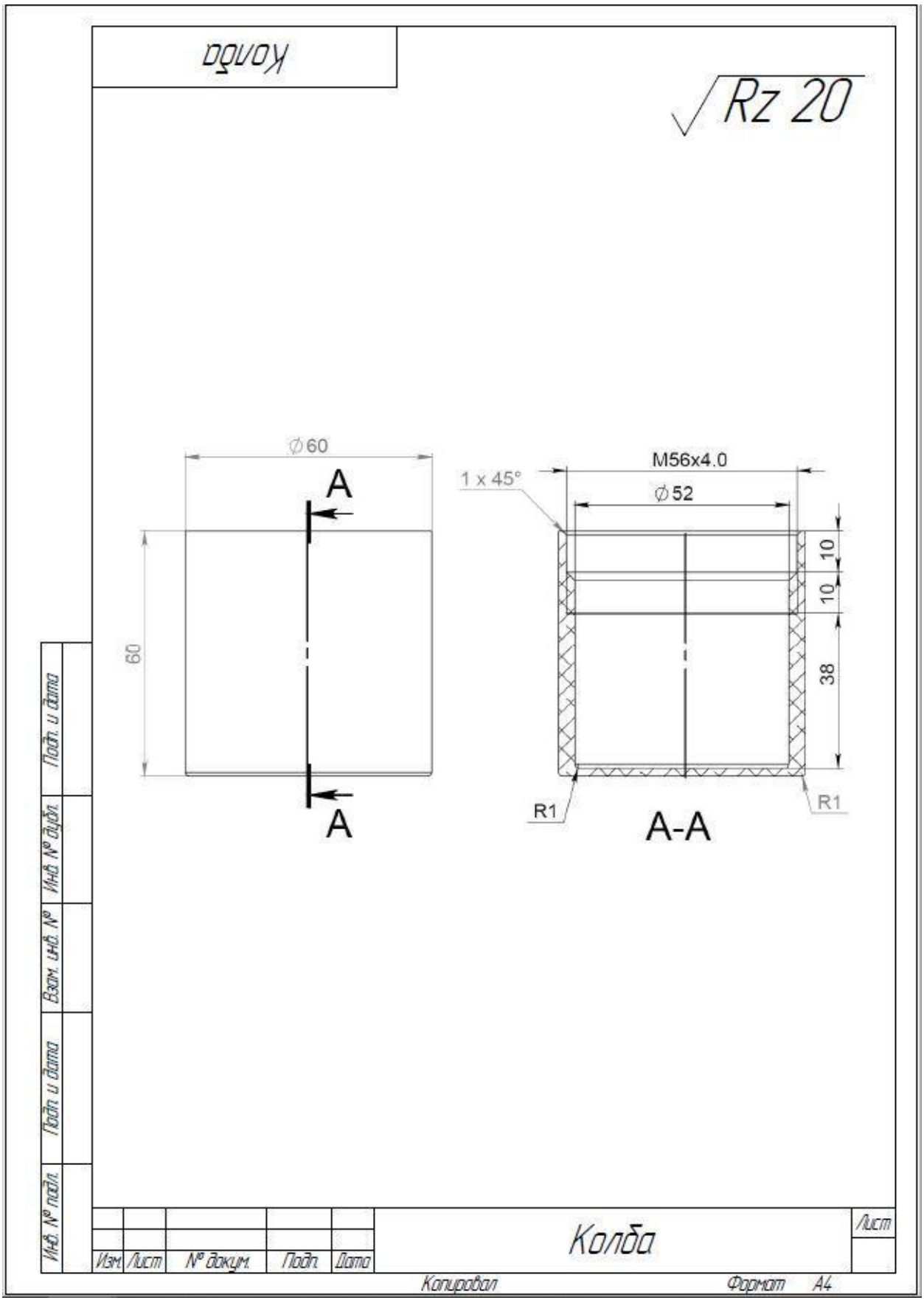
Диаметр уплотнительного кольца 51 мм - 10 баллов;
Ширину канавки в крышке 3,6 мм - 10 баллов;
Диаметр для посадки кольца в крышке 52.3 мм - 10 баллов;
Внутренний диаметр колбы в месте герметизации 56 мм - 10 баллов.

Чертёж.

Рассчитанные по ГОСТу величины являются гарантом герметичности корпуса, при построении чертежей эти размеры нужно обязательно использовать.
Построение и оформление чертежа выполняется по ГОСТ 2.052-2006

Пример:

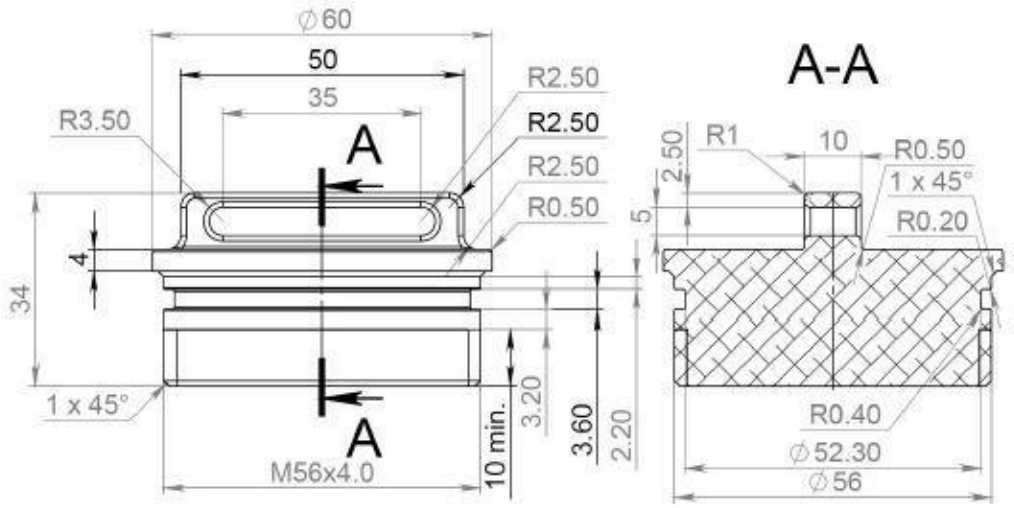
Кружковое движение НТО ЦРР



Кружковое движение НТО ЦРР

Крышка

Rz 20




| Изм. | Лист | № докум. | Подп. | Дата | Крышка | Лист |
|---|---|---|---|---|---|---|

Копировал Формат А4

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

3Д-модель сборки крышки и колбы

3д модели частей герметичного корпуса строятся отдельно в любой CAD-программе, согласно чертежам, а затем объединяются в сборку.

### Задача 10 (тип 1 или тип 2)

1.1. Дано:

- Алюминиевый пруток длиной 180 мм, диаметром 100 мм.
- Алюминиевый пруток длиной 50 мм, диаметром 180 мм.
- Резиновое уплотнительное кольцо сечением 2,5 мм произвольного диаметра (вам необходимо самостоятельно подобрать)
  Прутки считать проточенными и подогнанными в размер.

1.2. Вам предстоит спроектировать герметичный корпус, который состоит из колбы с глухим дном и крышки. Он будет использоваться под водой. Дополнительные баллы можно получить за спроектированную систему крепления крышки к корпусу (система крепления: отверстия, крепёжные элементы, дополнительные детали. Что-то может быть частью колбы или крышки (детали - не идеальные тела вращения, допускаются вырезы, пазы, ушки и т.д. Главное - не нарушить герметичность.), это необходимо для фиксации герметизации, чтобы при ситуации, когда давление внутри корпуса больше давления окружающей среды, крышку не выдавило и не нарушилась герметизация. Несмотря на то, что система крепления не ограничена в использовании материалов (прутки только для колбы и крышки) и технологий изготовления (не задумывайтесь над тем, как это изготовить), не стоит делать её громоздкой. Также при проектировании система должна считаться с законами физики и механики, не запрещается перенимать принципы работы готовых решений.

Пример герметичного корпуса с системой крепления приведен на картинке ниже, однако такая система крепления (стягивающий хомут) будет оцениваться в 0 баллов.

Для герметизации необходимо использовать уплотнительное кольцо сечением 2,5 мм. Минимально допустимая толщина стенок и дна колбы — 3 мм. Минимальное расстояние между двумя параллельными поверхностями, которые испытывают нагрузки, 1,5 мм.

В корпус должен помещаться и извлекаться один из кубов: куб со стороной 45 мм или 63 мм, минимальный зазор между кубом и стенками составляет 1 ± 0,2 мм.

1.3. Участникам необходимо правильно рассчитать в мм:

- Диаметр уплотнительного кольца
- Ширину канавки под кольцо
- Глубину канавки под кольцо

Для расчетов необходимо воспользоваться ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств". Особое внимание при изучении ГОСТа уделите приложению.

1.4. Сделайте чертеж колбы

- Должен содержать разрез детали;
- Должен быть сохранен в формате .pdf;
- Может быть выполнен в любой программе;
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

1.5. Сделайте чертеж крышки

- Должен содержать разрез детали;
- Должен быть сохранен в формате .pdf;
- Может быть выполнен в любой программе;
- Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.

Исчерпывающие требования к чертежам указаны в ГОСТ 2.052-2006 Тщательно изучите пункт «Нормативные ссылки», при переходе по гиперссылкам вы найдёте ответы на все вопросы по оформлению чертежей.

ГОСТ 2.311-68 «ИЗОБРАЖЕНИЕ РЕЗЬБЫ»

Для основной надписи используйте форму 2а, её нужно также заполнить в соответствии с ГОСТами.

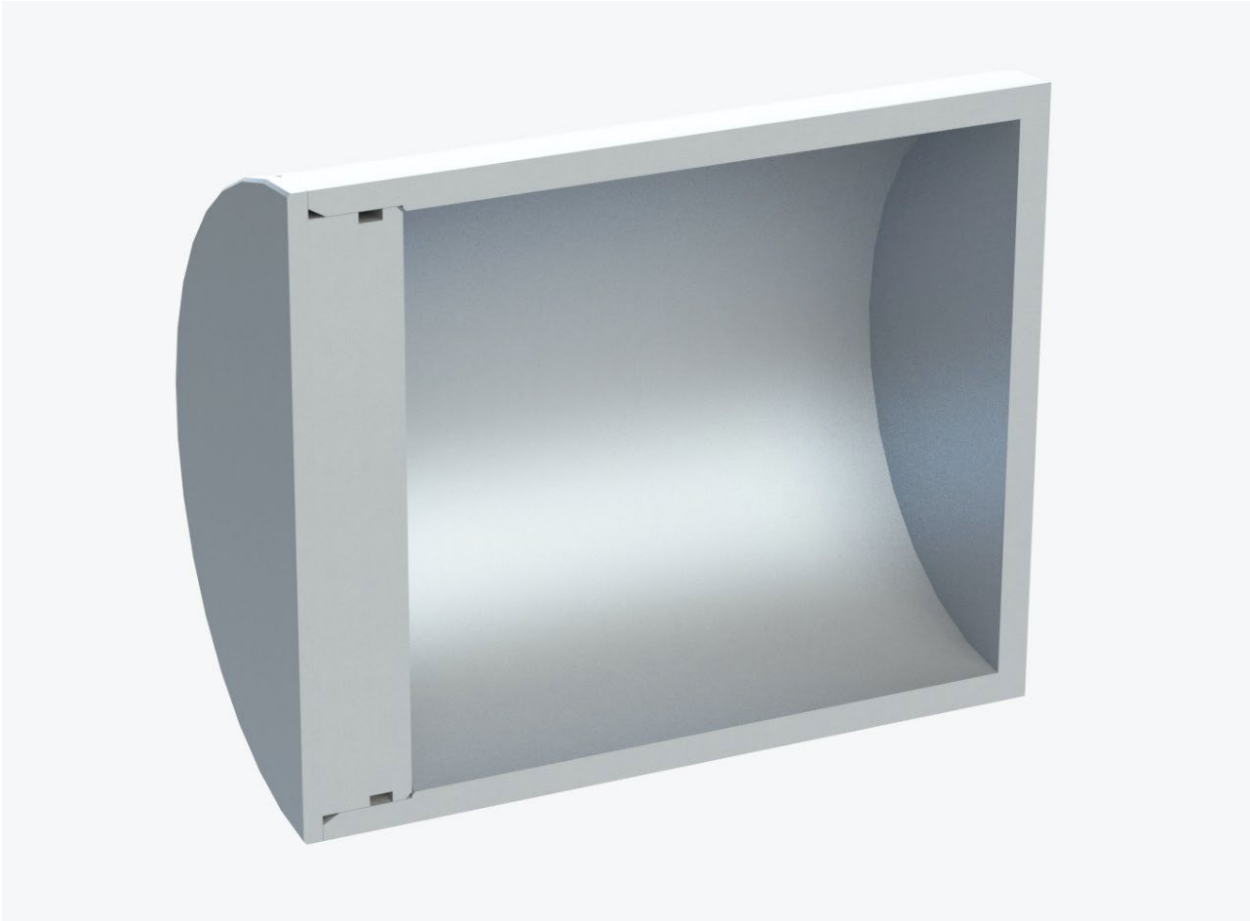
Предельные отклонения и допуски указывайте только там, где это необходимо.

Строгое соответствие всем пунктам ГОСТа 2.052-2006 и ГОСТам, на которые он ссылается, не нужно, достаточно соблюдение тех, которые от вас требуются (Должны быть проставлены все размеры, необходимые для изготовления детали, а также шероховатости поверхности и допуски.)

1.6 Сделайте 3Д-модель сборки корпуса: колба, крышка, крепление крышки к корпусу, крепёж (винты, гайки) и пр.

- Сборка должна быть выполнена в любой CAD программе. Рекомендуем использовать AutodeskInventor;
- Сборка должна быть сохранена в формате .step;
- Сборка должна содержать все элементы корпуса и системы крепления (винты гайки, шпильки обязательны, если они предусмотрены в системе);
- указана резьба, где это необходимо;
- Все детали должны быть сопряжены, а система крепления должна выполнять свои функции (не нужно разносить компоненты, для наглядности;
- В деталях должны быть предусмотрены все необходимые фаски и скругления острых граней для безопасного использования.

Пример сборки корпуса с радиальным уплотнением без системы крепления, разрез.



1.7. Решением данной задачи будет являться 3 файла:

- pdf-файл с чертежом колбы;
- pdf-файл с чертежом крышки;
- .step файл (один файл) с моделью сборки корпуса: колба, крышка, крепление крышки к корпусу, крепёж (винты, гайки) и т.д.

  В ответе необходимо указать обозначение типоразмера кольца, размер куба и результаты расчетов из пункта 1.3 в мм.

  Пример:

  Кольцо ***-***-25, куб 63мм

  Диаметр уплотнительного кольца = 1

  Ширина канавки под кольцо = 2

  Глубина канавки под кольцо = 3

## Решение задачи 10 (тип 1 или тип 2)

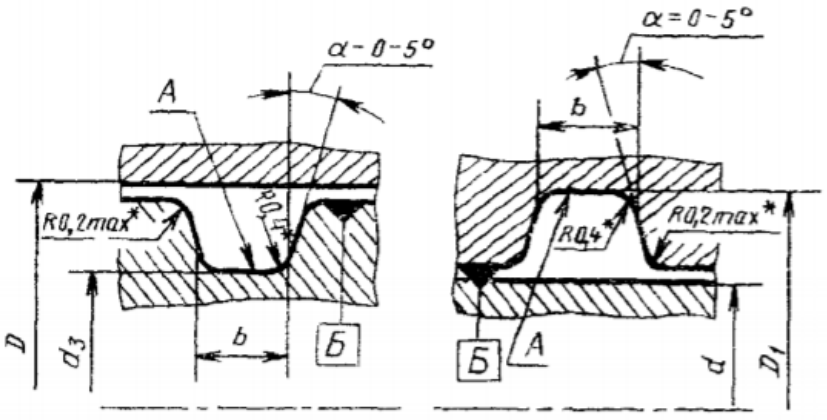
1.Подбираем необходимый диаметр уплотнительного кольца:

- Исходя из ограничений по толщине стенки, размеров заготовок и в корпус должен помещаться и извлекаться куб со стороной 45 мм или 63 мм из дано: 100 мм – максимальный внешний диаметр колбы, $d = a\sqrt{2}$ =45* 1,4 = 63 мм – минимальный внутренний диаметр колбы (формула описанной окружности около квадрата), вторая цифра в типоразмере кольца, далее $D$, должна быть 64 – 108 мм .
- Открываем ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" листаю до «Кольца уплотнительные сечением 2,5»
- На стр. 6 определяю, что любой типоразмер кольца, от 060-064-25 до 102-108-25 подходит для проектирования корпуса, но исходя из логики «чем больше корпус, тем больше в него можно положить чего-либо для защиты этого чего-то от контакта с водой», выбираю для дальнейшего проектирования кольцо 098-102-25 с диаметром 96 мм. Диаметр уплотнительного кольца = 96 мм + 5 баллов.

2. На стр. 29 (28) ГОСТ 9833-73 "Кольца резиновые уплотнительные круглого сечения для гидравлических и пневматических устройств" нахожу таблицу с расчётами для кольца сечением 2,5 мм и вижу 2 чертежа:

С 28 ГОСТ 9833—73

**Посадочные места под кольца с диаметром сечения $d_2$=2,5 мм**




* Размер обеспечивается инструментом

Черт 7

Смотрим на изображение с образцом корпуса на странице индивидуального задания для конструкторов на Stepik'е и понимаем, что нас интересует только левая картинка, т.к. установка кольца происходит в крышке (чертёж слева), а не на распор в корпусе (чертёж справа).

3. По таблице ниже находим $D, d_3, b$ напротив 098-102-25 в столбце «Неподвижное соединение»:

$b = 3{,}6$ мм + 5 баллов;

$b$ – ширина канавки в крышке;

$\frac{D-d_3}{2} = 1{,}85$ мм – глубина канавки в крышке + 5 баллов.

Расчёт окончен:

Диаметр уплотнительного кольца 96 мм– 5 баллов;

Ширину канавки в крышке 3,6 мм - 5 баллов;

Глубина канавки в крышке 1,85 мм - 5 баллов.

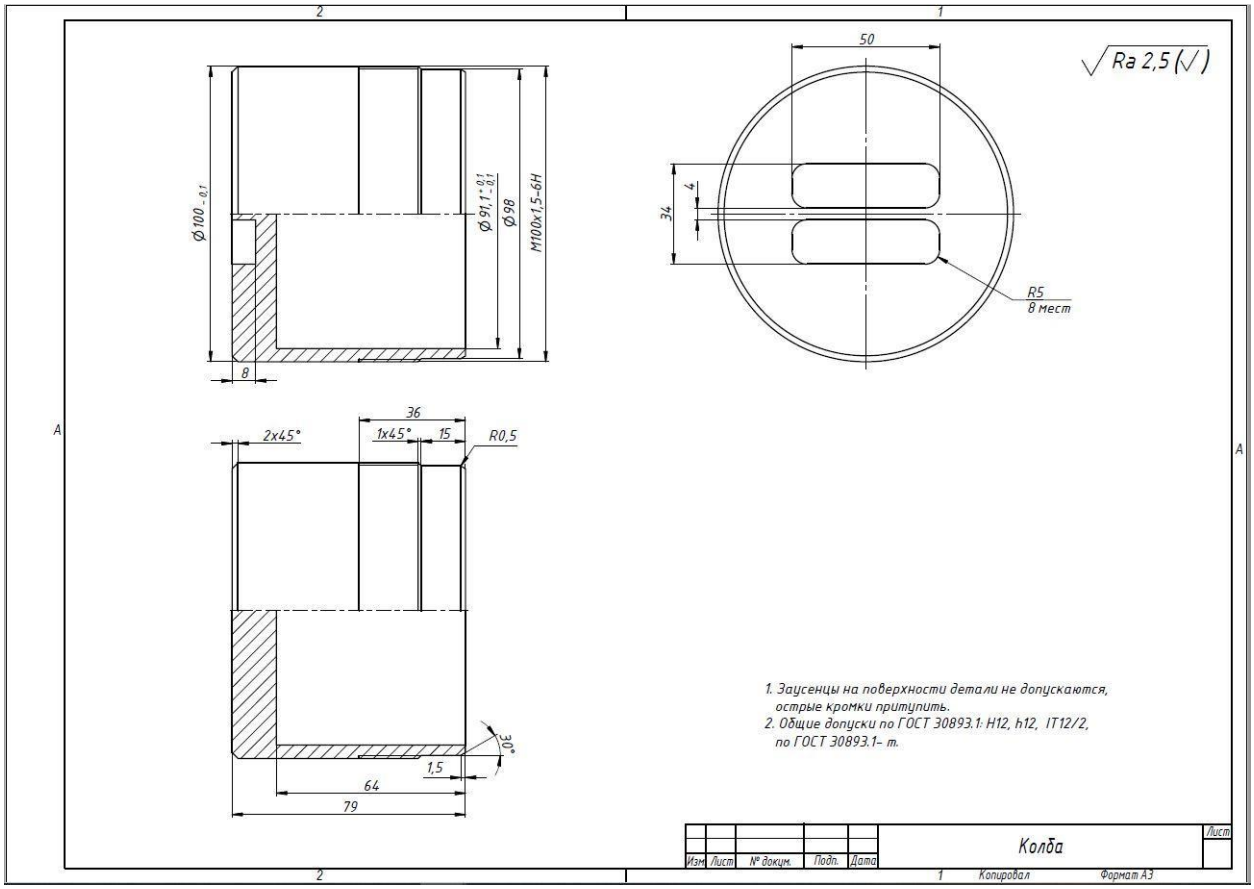
Чертёж.

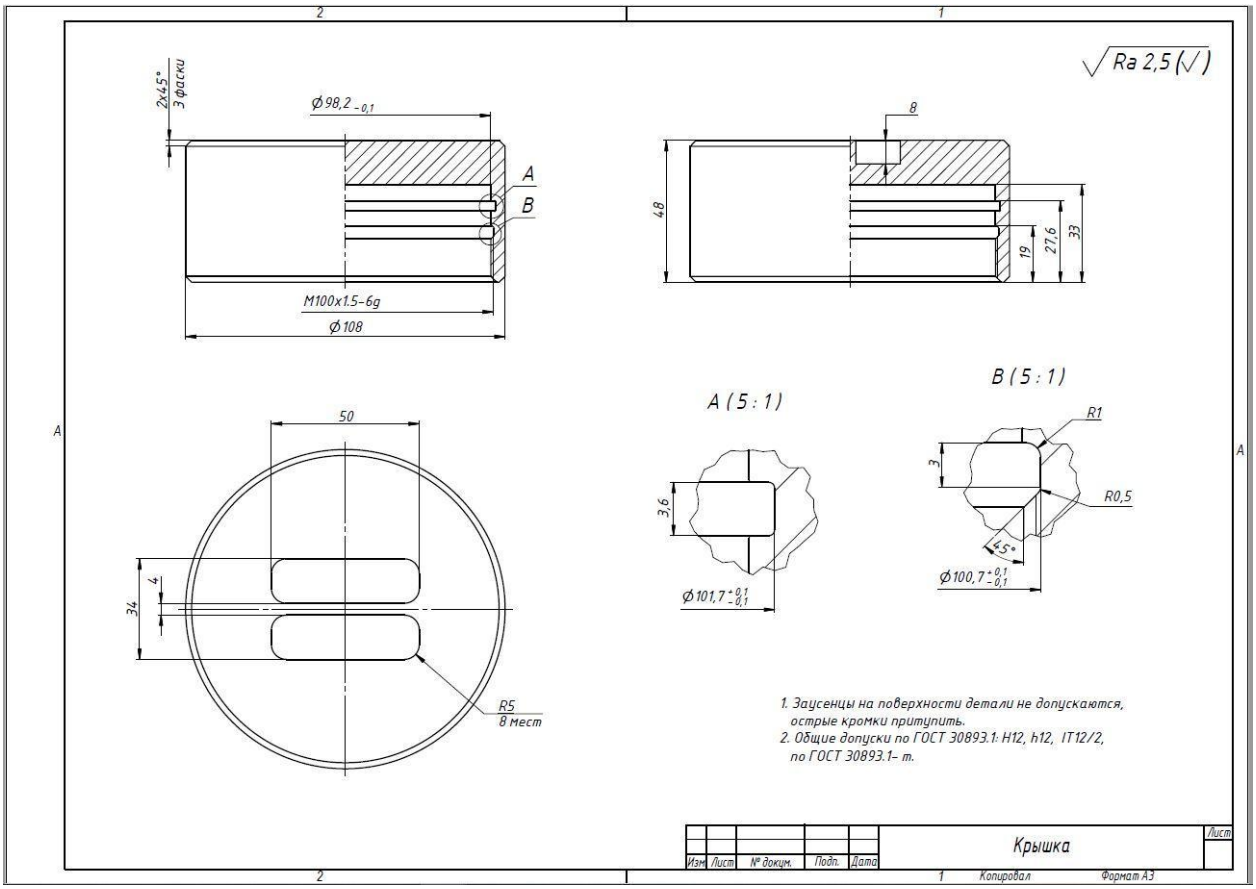
Рассчитанные по ГОСТу величины являются гарантом герметичности корпуса, при построении чертежей эти размеры нужно обязательно использовать.

Построение и оформление чертежа выполняется по ГОСТ 2.052-2006

Пример:

Кружковое движение НТО ЦРР





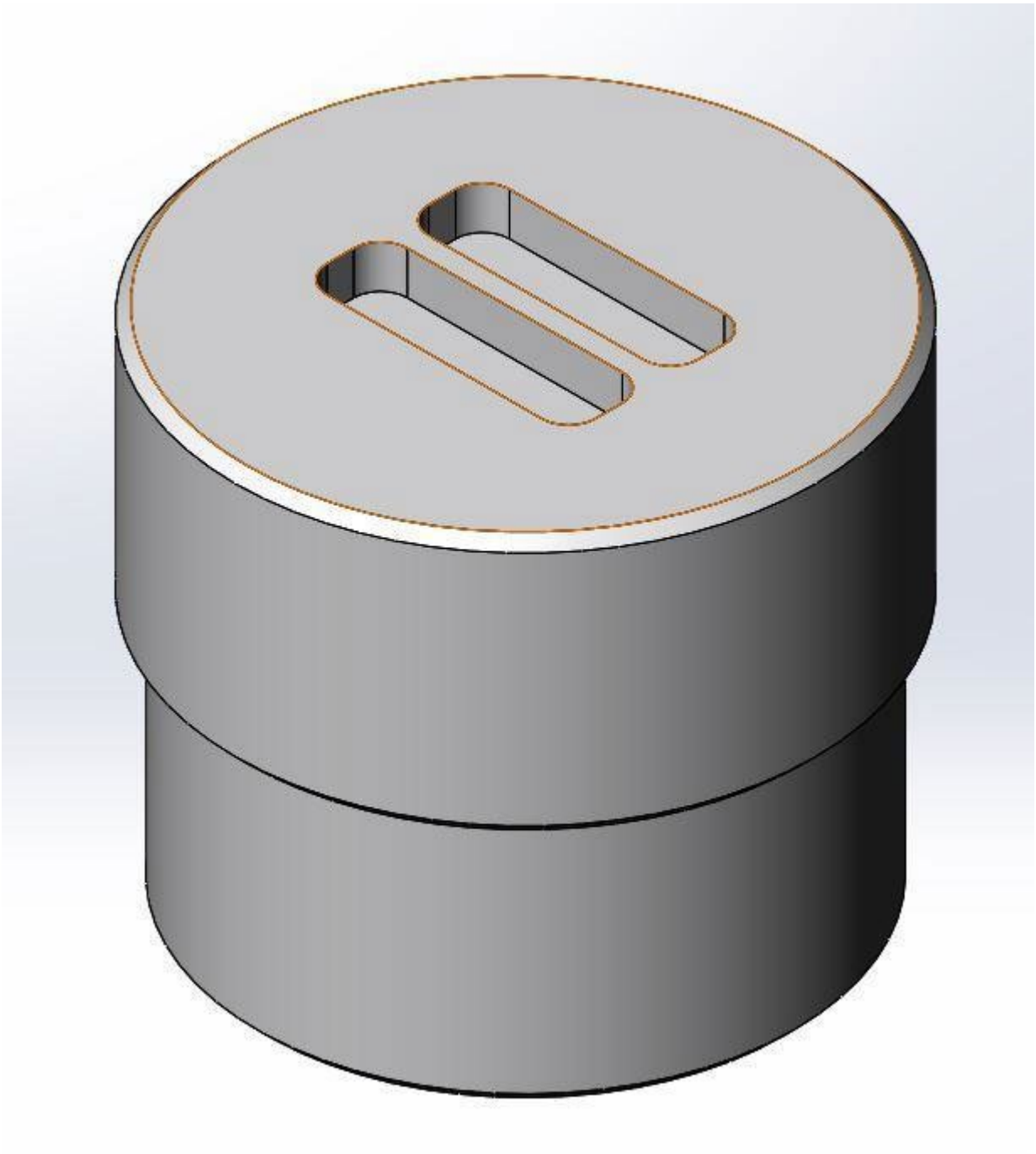

3Д-модель сборки крышки и колбы

3д модели частей герметичного корпуса строятся отдельно в любой CAD-программе, согласно чертежам, а затем объединяются в сборку.

Кружковое движение

НТО

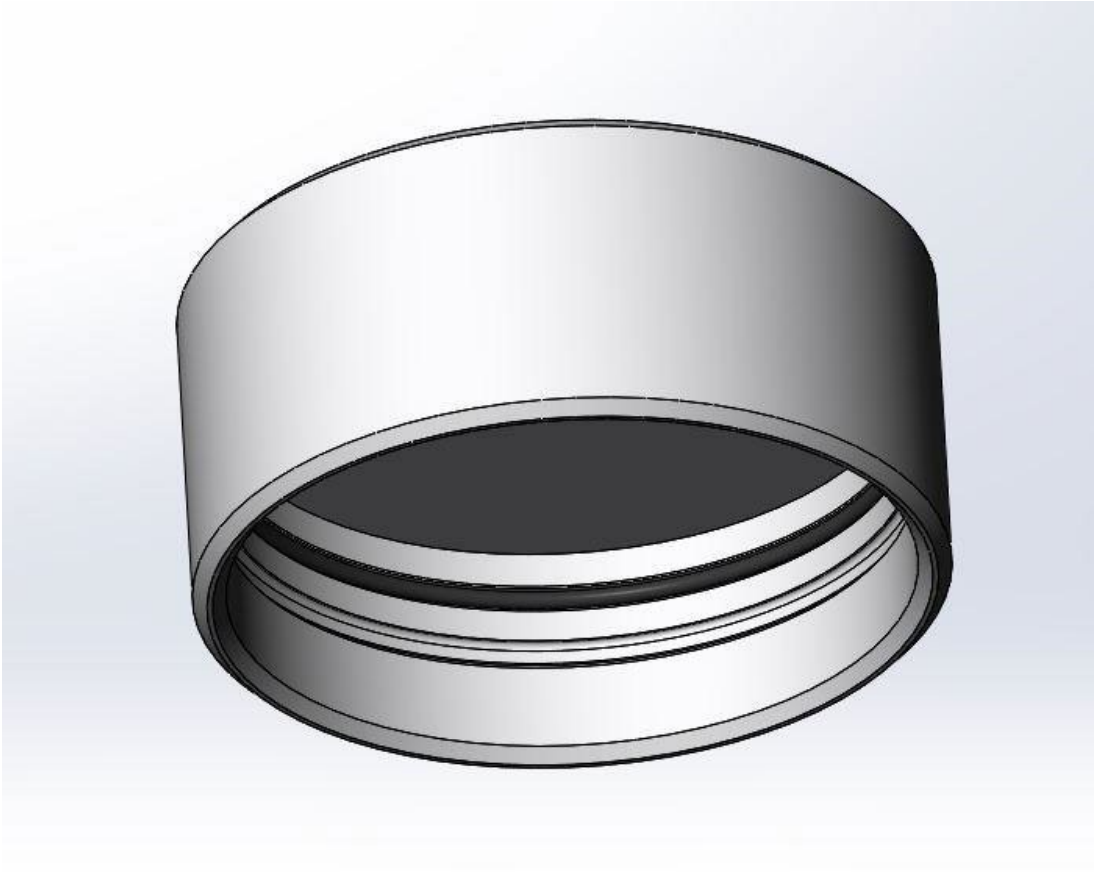
ЦРР



Кружковое движение

НТО

ЦРР



Кружковое движение НТО ЦРР

## Электроника

### Задача 11 (тип 2)

Дано (элементы в Thinkercad)

- Двигатель постоянного тока
- Потенциометр
- Ползунковый переключатель
- Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
- Элементы питания
- Логические микросхемы находящиеся в разделе "Логика"
- Макетные платы
- Таймер

Необходимо, используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) создать проект электрической схемы управления коллекторным мотором. Основные требования:

- Можно использовать только те компоненты, которые приведены в пункте 1.1.
- Управление скоростью вращения мотора должно осуществляться при помощи потенциометра, а изменение направления вращения - ползунковым переключателем.
- Диапазон скорости мотора должен выходить за пределы (-10000;  10000).
- Скорость вращения должна меняться с помощью широтно-импульсной модуляции.
- Изменение направления вращения должно осуществляться при помощи Н-моста, собранного на транзисторах.
- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Изменение скорости вращения должно происходить плавно.

Схемы, в которых будут использоваться неразрешенные компоненты, проверяться не будут. Участник получает за него 0 баллов.

Баллы за задачу начисляются следующим образом.

- 100 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты.
- 90 баллов - схема работает правильно в соответствие с требованиями. Используются избыточные компоненты.
- 80 баллов - схема работает правильно в соответствие с требованиями. Не используются избыточные компоненты. Но скорость меняется только в пределах (-10000; 10000).
- 50 баллов - схема работает частично. Либо только меняется направление вращения, либо - скорость.
- 30 баллов - схема работает частично. Скорость меняется в пределах (-10000; 10000). Направление вращения не меняется.

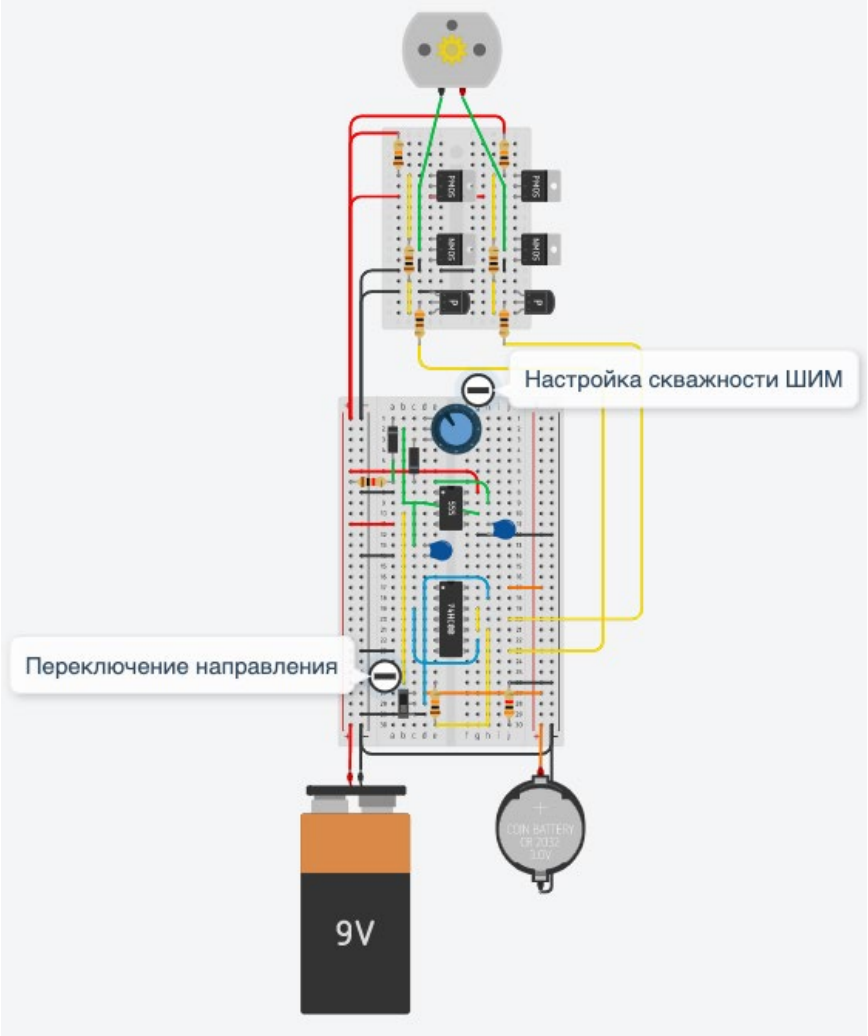
- 0 баллов - схема не работает даже частично.

## Решение задачи 11 (тип 2)

Данная схема является примером выполнения задачи на 100 баллов. Здесь представлены все элементы, которые можно использовать, без снижения баллов за их избыточное количество. Схема условно поделена на две части. Н-мост на полевых транзисторах, а также генератор ШИМ с логическим управлением направления вращения мотора.

Задание можно выполнить на максимальное количество баллов используя меньшее количество элементов. В частности, микросхему 74hc00 использовать не обязательно, так как задачу можно выполнить и без нее. Представленные номиналы обвязки микросхемы NE555 так же не являются эталонными и могут отличаться, главное удовлетворять диапазону значений, указанному в задании.

Элемент питания 3В добавлен схему для работы микросхемы 74hc00 и не является обязательным.



## Задача 12 (тип 2)

1.1. Дано (элементы в Thinkercad)

- Двигатель постоянного тока
- Потенциометр
- Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
- Элементы питания
- Логические микросхемы находящиеся в разделе "Логика"
- Макетные платы
- Таймер

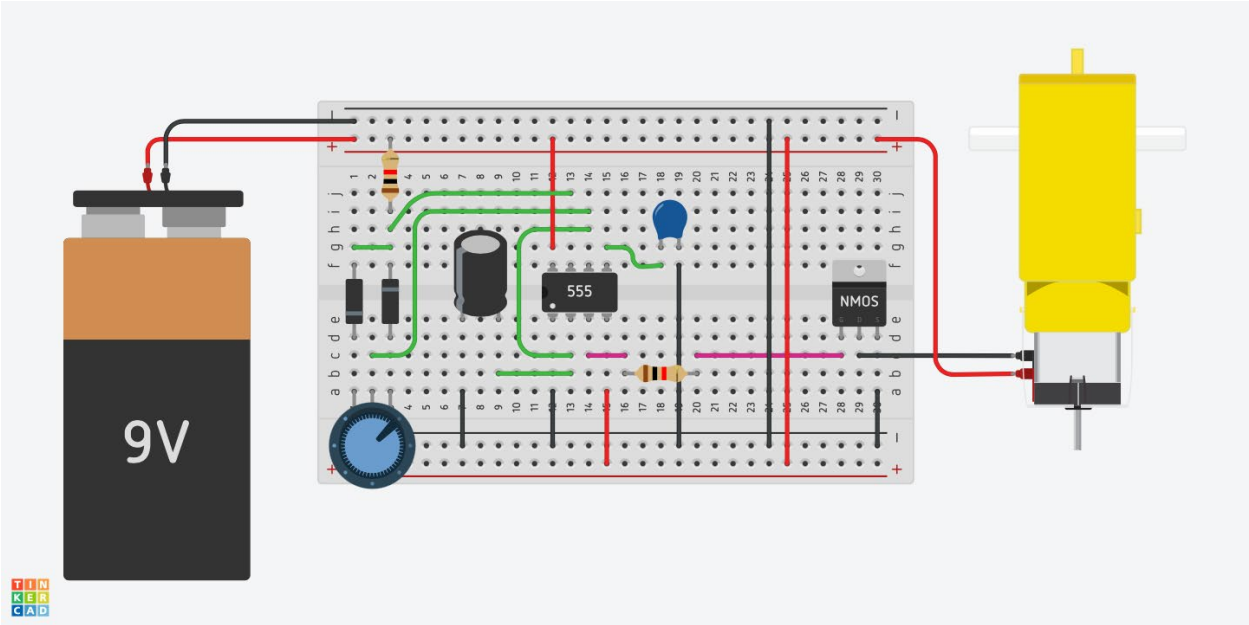
1.2. Необходимо, используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) создать проект электрической схемы управления коллекторным мотором. Основные требования:

- Можно использовать только те компоненты, которые приведены в пункте 1.1.
- Необходимо реализовать управление скоростью вращения мотора при помощи потенциометра без изменения направления вращения мотора.
- Диапазон скорости мотора не должен выходить за пределы (0;  10000).
- Скорость вращения должна меняться с помощью широтно-импульсной модуляции.
- Управление вращением должно осуществляться при помощи транзистора.
- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Изменение скорости вращения должно происходить плавно.

1.3. Схемы, в которых будут использоваться неразрешенные компоненты, проверяться не будут.

## Решение задачи 12 (тип 2)

Данная схема является примером выполнения задачи. Здесь представлены все элементы, которые можно использовать, без снижения баллов за их избыточное количество. Схема состоит из таймера (микросхема таймера NE555), а также полевого транзистора для управления нагрузкой (мотором). Для изменения скорости вращения мотора мы используем потенциометр и тем самым изменяем скважность сигнала, подаваемую на затвор полевого транзистора.




Ссылка на пример решенного задания: https://www.tinkercad.com/things/dK9AIVIrVtk-copy-of-pwmne555pinosa/editel?sharecode=6m4GdeLgPJ-ZTXaqcdv9XgtJIUgM0NBQV52OSbTF-GA

## Задача 13 (тип 1)

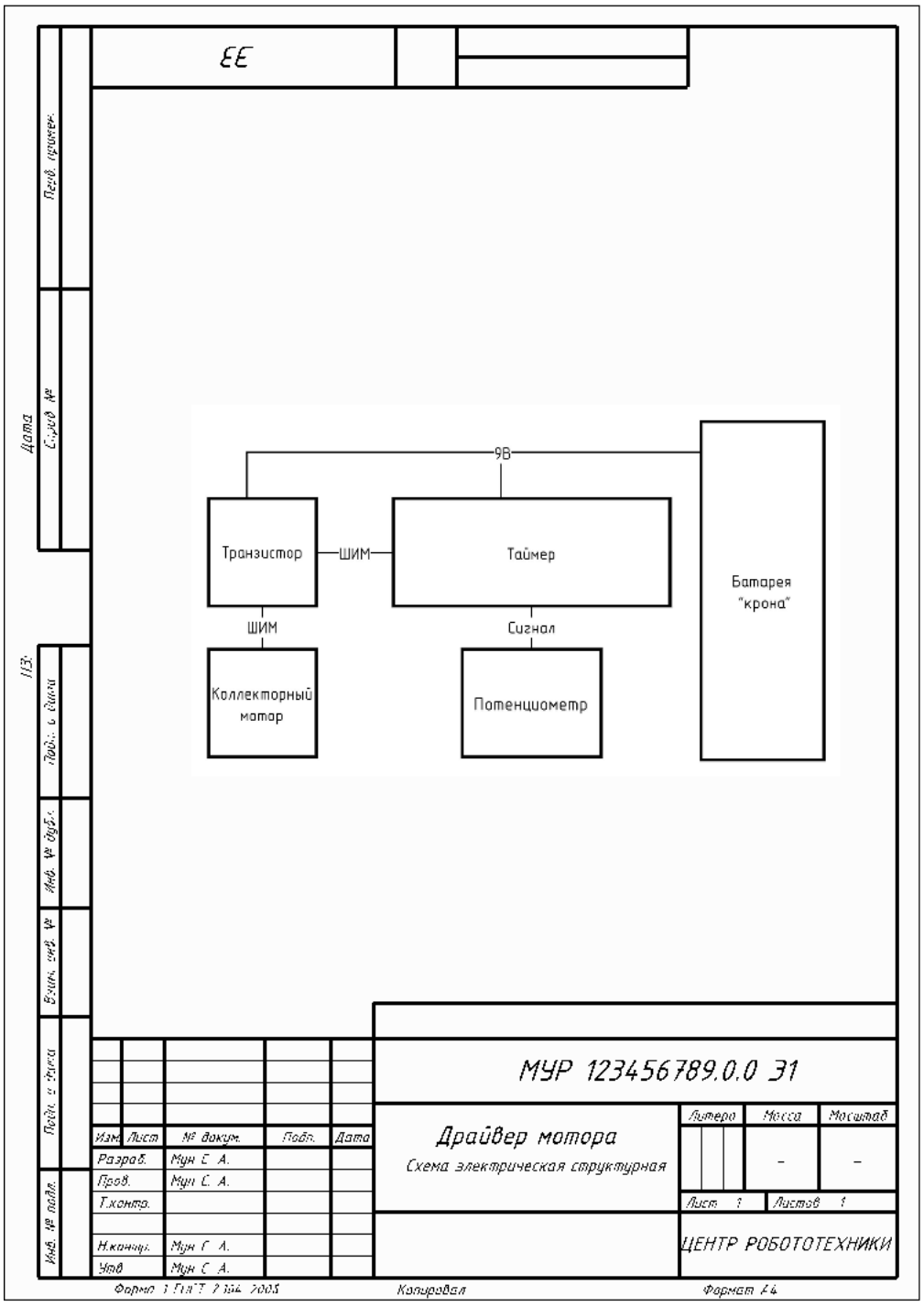
Предлагаем участникам научиться разрабатывать Схему электрическую структурную. Данная схема разрабатывается в соответствии с ГОСТ 2.702-2011. Однако в целях обучения мы немного упростили требования к ней. И свели их к следующим критериям:

Критерии оценки:

- Наличие рамки для заглавного листа в соответствии с ГОСТ 2.104-2006
- Рамка заполнена в соответствии с ГОСТ 2.104-2006
- Схема выполнена в САПР
- Линии связи должны состоять из горизонтальных и вертикальных отрезков и не имеют изломов и взаимных пересечений
- Все линии имеют подписи и все текстовые данные, относящиеся к линиям, ориентируют параллельно горизонтальным участкам соответствующих линий
- Все надписи выполнены шрифтом ГОСТ тип Б
- Схема выполнена на одном листе А4
- Схема занимает более 30% площади листа
- Все блоки схемы имеют названия, отражающие функциональное обозначение компонентов и их количество не избыточно
- Схема не имеет избыточных соединений между блоками

Необходимо разработать Схему электрическую структурную для задачи 12.

## Решение задачи 13 (тип 1)



## Задача 14 (тип 2)

1.1. Дано (элементы в Thinkercad)

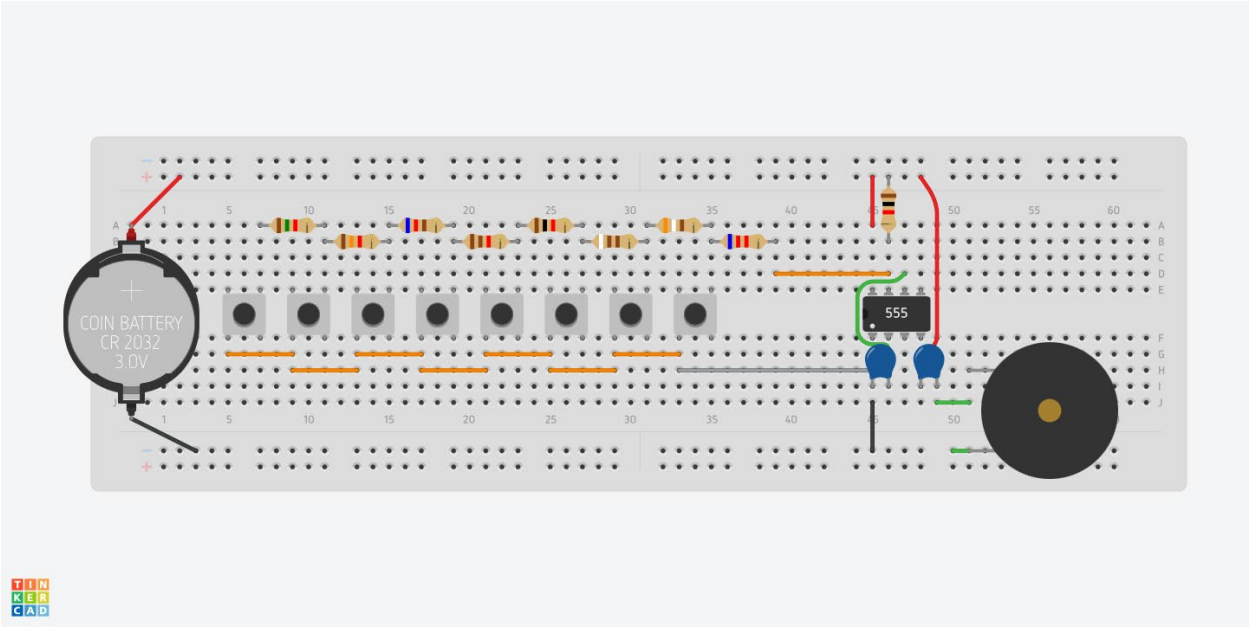
- Динамик или пищалка
- Потенциометр
- Аналоговые компоненты: резисторы, конденсаторы, индуктивности, диоды, транзисторы
- Кнопки
- Элементы питания
- Логические микросхемы находящиеся в разделе "Логика"
- Макетные платы
- Таймер

1.2. Необходимо, используя online-сервис Tinkercad (во избежание проблем с моделированием, рекомендуем пользоваться Google Chrome) создать проект электрического пианино. Основные требования:

- Можно использовать только те компоненты, которые приведены в пункте 2.1.
- Необходимо сделать примитивное пианино способное генерировать от 5 до 12 нот, управляемых тактовыми кнопками
- Частотный диапазон нот должен соответствовать первой октаве фортепиано.
- При нажатии на соответствующей кнопки, как и в нормальном пианино должен издаваться звук правильной частоты (допускается округление до целых значений Гц).
- Не разрешается использование микроконтроллеров, датчиков, контроллеров управления моторами, электрических приводов с Н-мостами, а также генераторов сигналов и регулируемых источников питания.
- Когда кнопки не нажаты посторонних звуков быть не должно.

## Решение задачи 14 (тип 2)

Данная схема является примером выполнения задачи. Здесь представлены все элементы, которые можно использовать, без снижения баллов за их избыточное количество. Схема состоит из таймера (микросхема таймера NE555), а также набора сопротивлений с кнопками для генерации определенной частоты.




Ссылка на пример решенного задания: https://www.tinkercad.com/things/4VE9SWioZjZ-pianino-nti/editel?sharecode=gsfBZBmRAjAWPFqikb3mG7nRFFhLUlCRB-HYOw1CUyI

## Задача 15 (тип 1)

Предлагаем участникам научиться разрабатывать Схему электрическую структурную. Данная схема разрабатывается в соответствии с ГОСТ 2.702-2011. Однако в целях обучения мы немного упростили требования к ней. И свели их к следующим критериям:

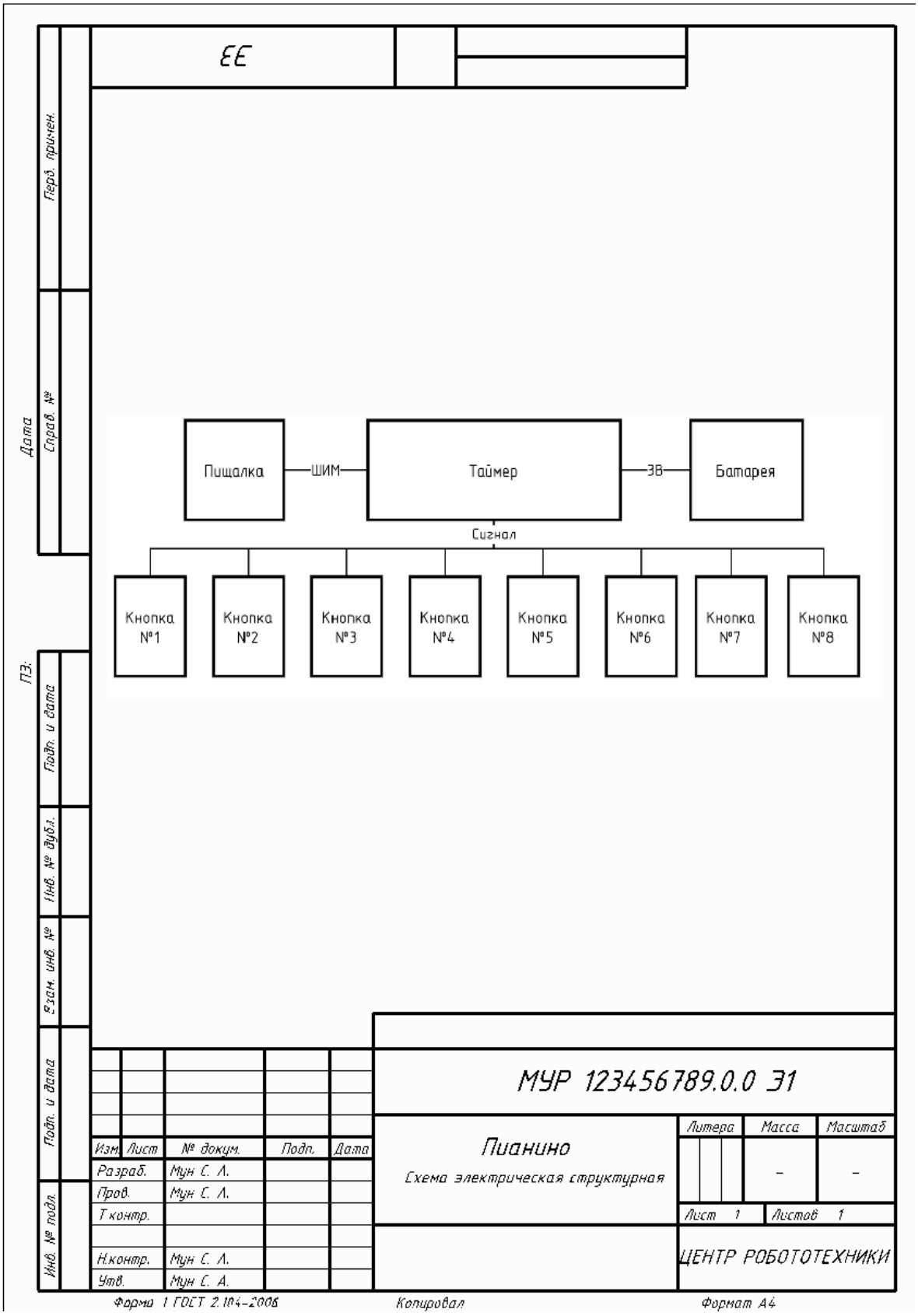
Критерии оценки:

- Наличие рамки для заглавного листа в соответствии с ГОСТ 2.104-2006
- Рамка заполнена в соответствии с ГОСТ 2.104-2006
- Схема выполнена в САПР
- Линии связи должны состоять из горизонтальных и вертикальных отрезков и не имеют изломов и взаимных пересечений
- Все линии имеют подписи и все текстовые данные, относящиеся к линиям, ориентируют параллельно горизонтальным участкам соответствующих линий
- Все надписи выполнены шрифтом ГОСТ тип Б
- Схема выполнена на одном листе А4
- Схема занимает более 30% площади листа
- Все блоки схемы имеют названия, отражающие функциональное обозначение компонентов и их количество не избыточно
- Схема не имеет избыточных соединений между блоками

Необходимо разработать Схему электрическую структурную для задачи 14.

Кружковое движение НТО ЦРР

## Решение задачи 15 (тип 1)



## Комплексные задачи

### Задача 16 (тип 3)

Требования к участникам на входе (участники должны знать, уметь):
Члены команды должны знать и уметь (не обязательно, чтобы каждый знал всё):

- Уметь программировать в Arduino IDE;
- Уметь паять;
- Знать, как работают некоторые электрические компоненты (транзистор, LED-лента).

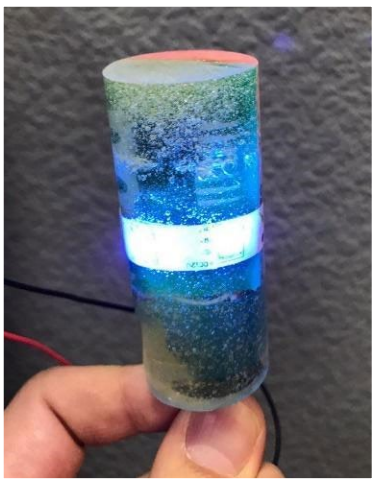
Занятие направлено на развитие следующих навыков и знаний у участников:

- Умение программировать в Arduino IDE;
- Умение проектирования электронных устройств;
- Знание и умения по работе с эпоксидной смолой.

Описание задания (Что делают участники)

В рамках хакатона участникам предстоит разработать, управляемый с ПК подводный LED-светильник.

Готовое устройство должно подключаться к ПК по USB и источнику питания 12V. По средствам стандартного монитора порта из Arduino IDE на плату передается сообщение вида: R,G,B,Off,On. Где R, G, B – компоненты цветового пространства RGB (соответственно) - целые числа в диапазоне от 0 до 255, Off – время в миллисекундах которое светодиодная лента не горит, On – время в миллисекундах которое светодиодная лента горит заданным цветом. Лента загорается и тухнет через заданные промежутки времени.



Пример готового устройства – управляемый с ПК подводный LED-светильник

Список необходимого оборудования

Для выполнения данного задания каждой команде участников необходимо предоставить:

- Arduino NANO – 1 шт.;
- LED RGB лента 12В – 1 шт. (3 элемента);

- Полевые транзисторы N-канальные (IRF540N, STP9NK60Z и их аналоги) – 3 шт.;
- Провода (желательно 3-4 цвета) – по 50 см;
- Макетная плата 30х80 мм – 1 шт.;
- USB-miniUSB кабель – 1 шт.;
- Паяльное оборудование, припой, флюс, пинцет – 1 комплект;
- Кусачки, стриппер, ножовка по металлу, напильники, изолента, термоусадочная трубка – 1 комплект;
- Шприц 50 мл (или другая подходящая емкость для заливки) – 1 шт.;
- Эпоксидная смола – 50 мл.;
- Емкость для смешивания эпоксидной смолы (стаканчик пластиковый) – 1 шт.
- Защитные очки, перчатки, респиратор – 1 комплект;
- Ноутбук с установленным Arduino IDE – 1 шт.;
- Источник питания 9-12В – 1 шт.

Список используемых расходных материалов на 1 команду (3-4 ученика):
- Arduino NANO – 1 шт.;
- LED RGB лента 12В – 1 шт. (3 элемента);
- Полевые транзисторы N-канальные (IRF540N, STP9NK60Z и их аналоги) – 3 шт.;
- Провода (желательно 3-4 цвета) – по 50 см;
- Макетная плата 30х80 мм – 1 шт.;
- USB-miniUSB кабель – 1 шт.;
- Шприц 50 мл (или другая подходящая емкость для заливки) – 1 шт.;
- Эпоксидная смола – 50 мл.;
- Емкость для смешивания эпоксидной смолы (стаканчик пластиковый) – 1 шт.
- Источник питания 9-12В – 1 шт.

Знания, необходимые педагогу для проведения мероприятия:
- Умение программировать в Arduino IDE;
- Умение проектирования электронных устройств;
- Знание и умения по работе с эпоксидной смолой.

Результат оценивается по 3 критериям:
- Качество пайки и изготовления электроники (0, 1 или 2 балла)
- Качество заливки и внешний вид готового устройства (0, 1, или 2 балла)
- Качество написания программного кода. Функционал устройства (0, 1, 2 балла)

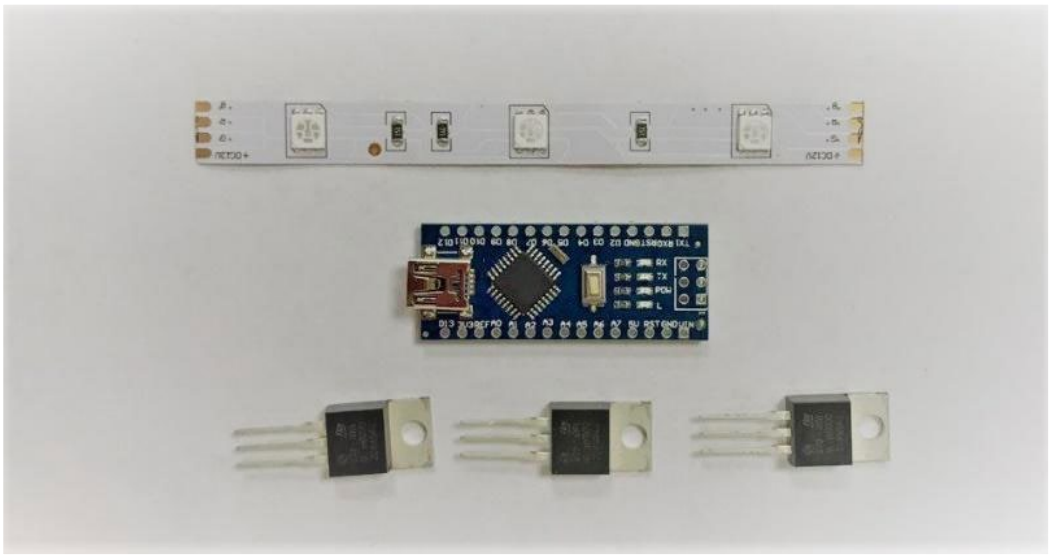


Рис.1. Электронные компоненты, необходимые для выполнения задачи.


Рис. 2. Инструменты и средства защиты

Кружковое движение НТО ЦРР

## Методические материалы к задаче 16

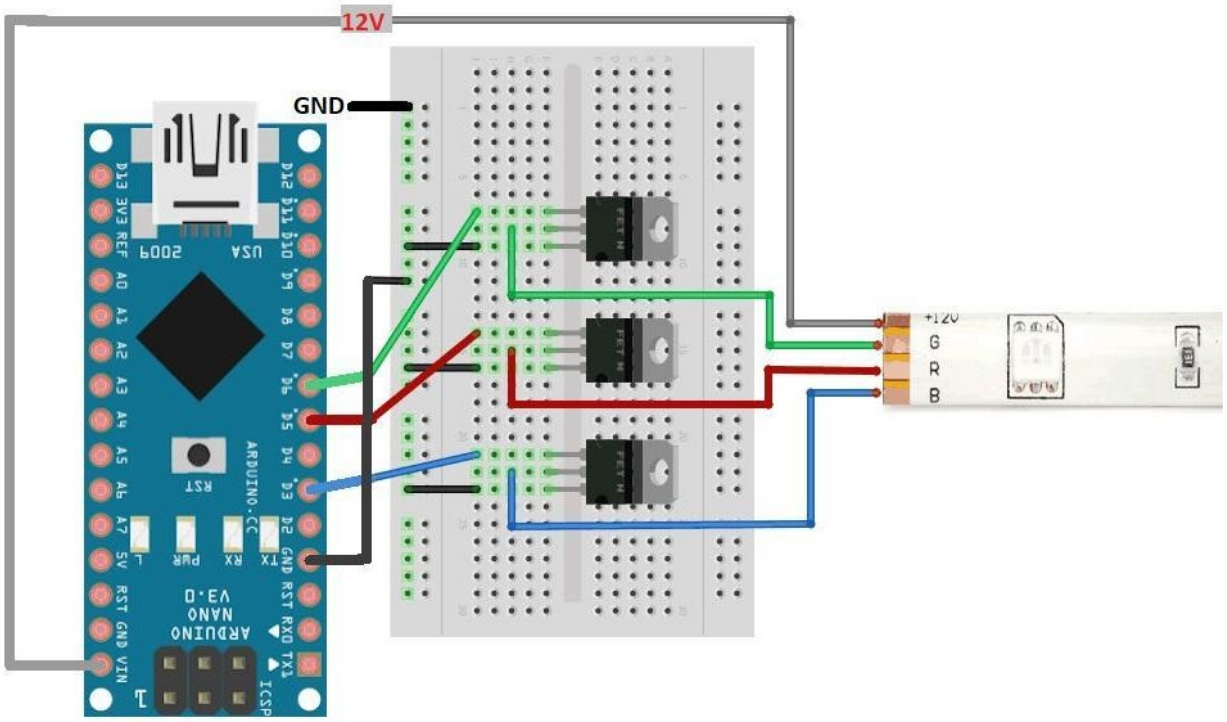
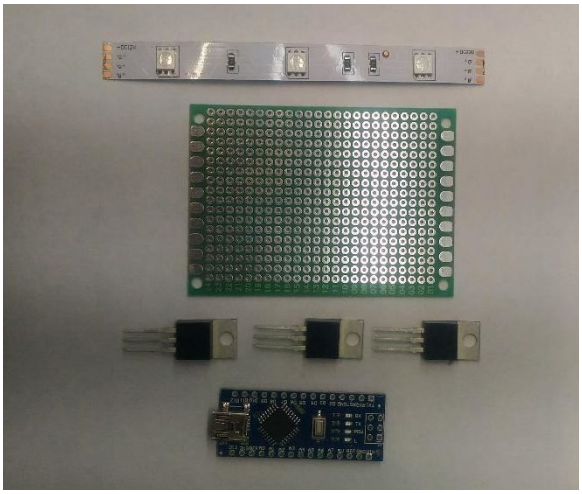
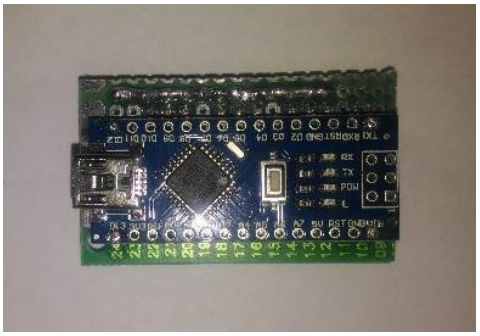
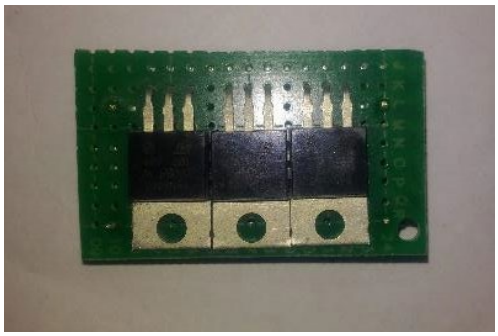



Рис. 3. Электрическая схема подводного LED-светильника

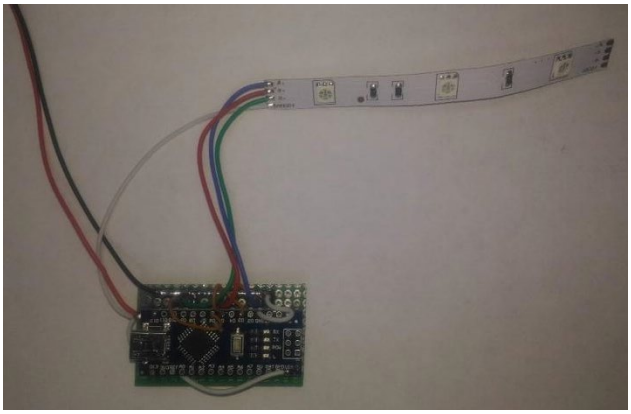
**Ход работы**



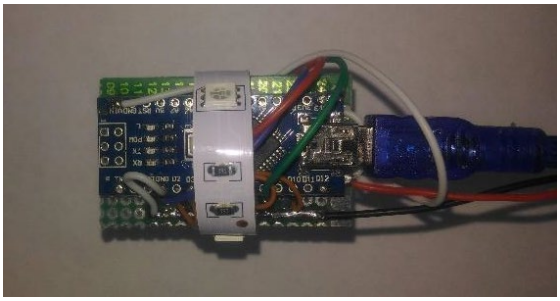
Шаг 1. Подготовка основных электронных компонентов

Шаг 2. Пайка основных элементов на макетную плату (с двух сторон)

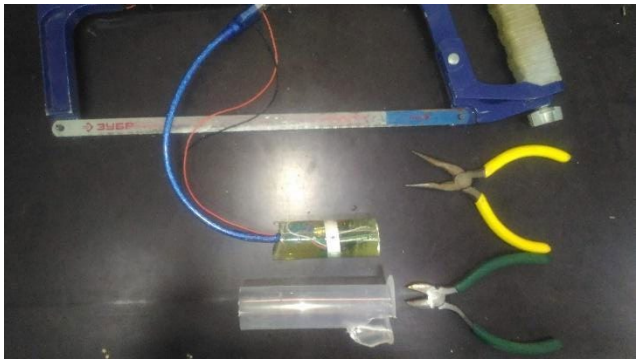


Шаг 3. Пайка проводов и LED-ленты




Шаг 4. Подготовка компонентов к заливке эпоксидной смолой


Шаг 5. Заливка элементов эпоксидной смолой



Шаг 6. Извлечение из формы и обработка

**Примечание: всю работу необходимо вести, используя средства индивидуальной защиты (респираторы, защитные очки, перчатки). Пайку, заливку и механическую**

**обработку заготовки производить строго в хорошо вентилируемом помещении.**

**Описание конечного результата**

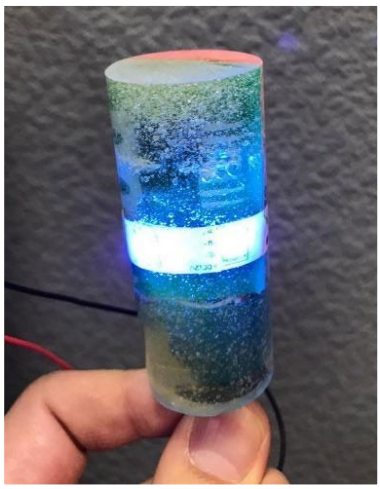


Рис. 4. Пример готового устройства – управляемый с ПК подводный LED-светильник.

Готовое устройство должно подключаться к ПК по USB и источнику питания 12V. По средствам стандартного монитора порта из Arduino IDE на плату передается сообщение вида: R,G,B,Off,On. Где R, G, B – компоненты цветового пространства RGB (соответственно) - целые числа в диапазоне от 0 до 255, Off – время в миллисекундах которое светодиодная лента не горит, On – время в миллисекундах которое светодиодная лента горит заданным цветом. Лента загорается и тухнет через заданные промежутки времени.

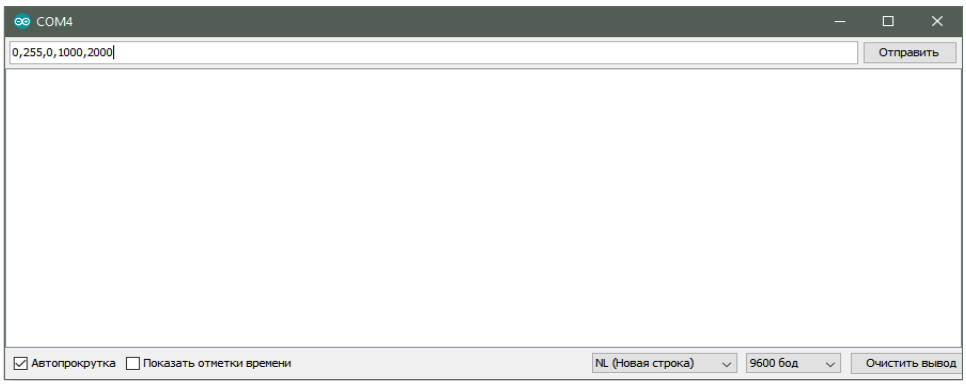



Рис. 5. Пример команды, передаваемый на Arduino NANO.

**Пример решения**

В качестве примера программного обеспечения для Arduino NANO может быть использован следующий код.

```
constexpr int redPin = 3;
constexpr int greenPin = 5;
constexpr int bluePin = 6;

int red = 0;
int green = 0;
int blue = 255;
int sleepTime = 0;
int lightTime = 0;

void doLED() {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
  delay(lightTime);
  analogWrite(redPin, 0x00);
  analogWrite(greenPin, 0x00);
  analogWrite(bluePin, 0x00);
  delay(sleepTime);
}

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  while (Serial.available() > 0) {

    red = Serial.parseInt();
    green = Serial.parseInt();
    blue = Serial.parseInt();
    sleepTime = Serial.parseInt();
    lightTime = Serial.parseInt();

    if (Serial.read() == '\n') {
      red = constrain(red, 0, 255);
      green = constrain(green, 0, 255);
      blue = constrain(blue, 0, 255);
    }
  }
  doLED();
}
```

Какие комплектующие и где их можно купить:

- RGB LED-лента: https://www.chipdip.ru/product0/8600178613
- Транзистор IRF540N: https://www.chipdip.ru/product/irf540n
- Транзистор STP9NK60Z: https://www.chipdip.ru/product0/8565447021?from=suggest_product
- Макетная плата: https://www.chipdip.ru/product0/8747169030
- Arduino Nano (на базеATmega328) https://www.chipdip.ru/product/arduino-nano
- USB-miniUSB кабель : https://www.chipdip.ru/product/pl1308

## Задача 17 (тип 3)

В рамках данной задачи учащимся необходимо собрать из комплектующих движитель OpenThruster 150. Для сборки движителя потребуются детали и комплектующие, приведённые ниже.

Для сборки подводного движителя OpenThruster 150 потребуется:

- Мотор (57x27.6 мм, 9800 об/мин, 12В 0.16А);
- воск (7 грамм);
- два провода (AWG 16/18, длина подбирается по необходимости);
- изолента;
- напечатанные детали корпуса: насадка, винт, крышка и корпус (PLA, заполнение 30%). Модели приведены по ссылке ссылка для скачивания.

Также понадобятся нож/ножницы, паяльник, припой, цианакрилатный (супер) клей, наждачная бумага или надфиль.

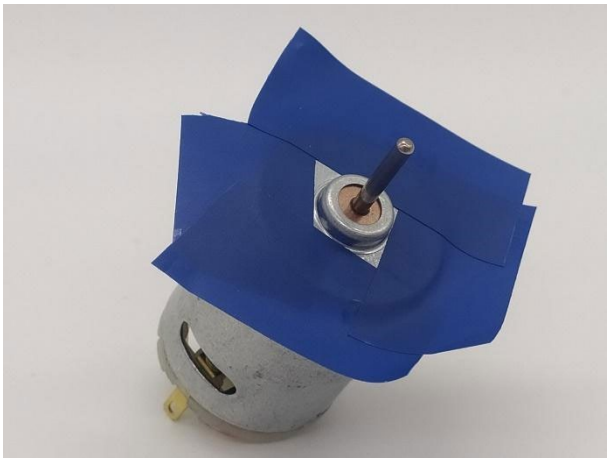

Комплектующие и материалы для сборки движителя

## Методические материалы к задаче 17

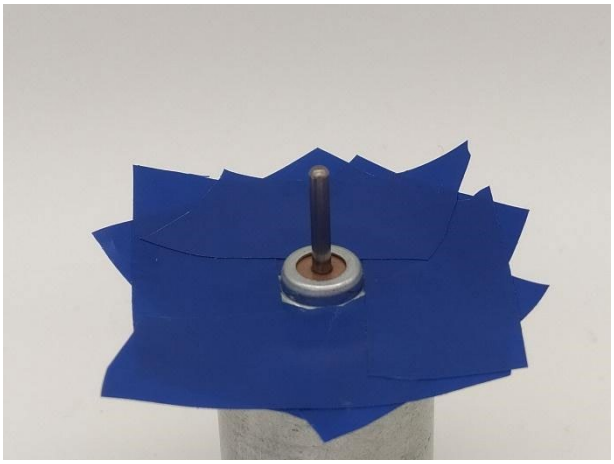
### Изоляция мотора

Перед началом работы по изоляции, необходимо обезжирить поверхность мотора, например, спиртом или обезжиривателем.

Маленькими отрезками изоленты нужно закрыть отверстия верхнего торца мотора. Каждый из отрезков должен полностью (не частично) закрывать одно\два отверстия. Кусочки следует клеить внахлёст (Шаг 1). Вторым слоем нужно перекрыть стыки изоленты первого слоя, а также покрыть как можно больше корпуса (Шаг 2).



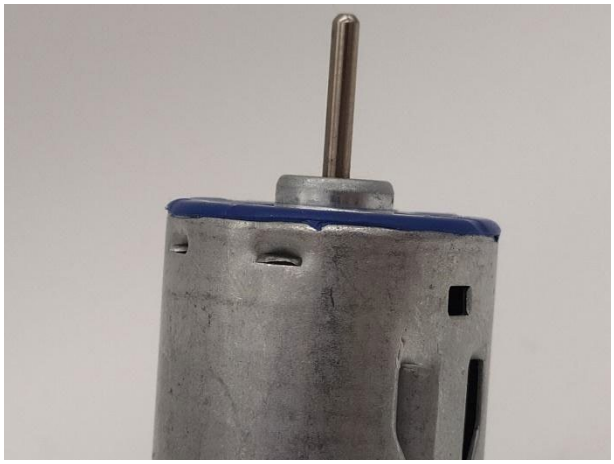
*Шаг 1*



*Шаг 2*

Затем нужно натянуть торчащую изоленту. Делать это нужно аккуратно, чтобы не сдвинуть слои. Этот шаг нужен, чтобы убрать складки в месте скругления корпуса мотора (Шаг 3). Затем обрезаем лишнее так, как показано в шаге 4.

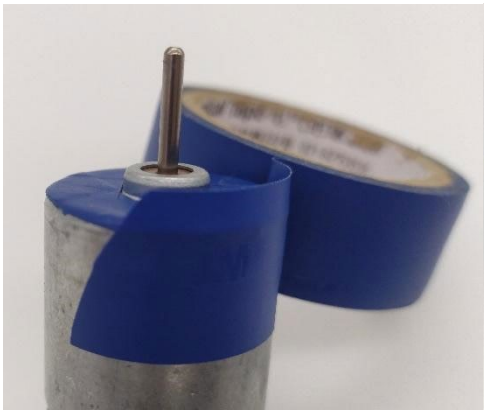
*Шаг 3*



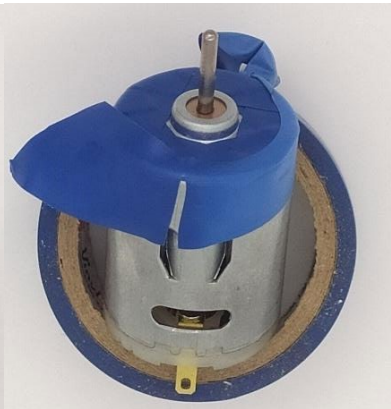
*Шаг 4*

Далее наносим изоленту на боковую поверхность мотора (Шаг 5), после четверти оборота начинаем натягивать изоленту так, чтобы она покрывала торец мотора. Кусок, который был не натянут нужно отрезать (Шаг 6). Продолжаем натягивать изоленту ещё 2

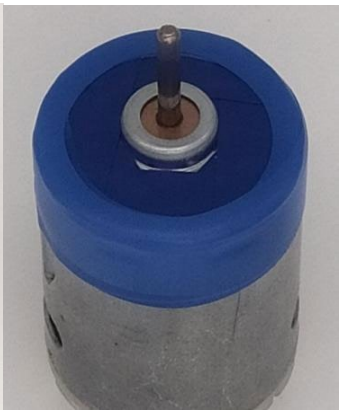
оборота (Шаг 7).



*Шаг 5*

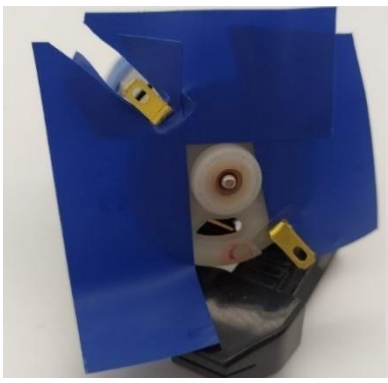


*Шаг 6*

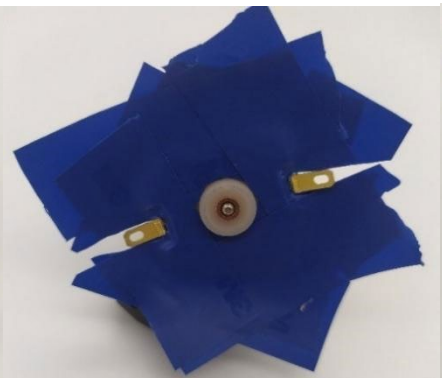


*Шаг 7*

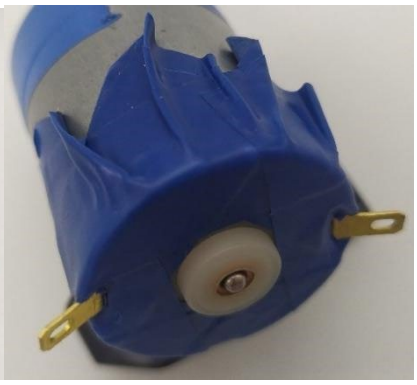
Для удобства при оклеивании нижнего торца моторчика контакты мотора можно отогнуть. Правила такие же, как для верхнего торца, однако в месте контактов изоленту нужно подрезать (Шаги 8-13).



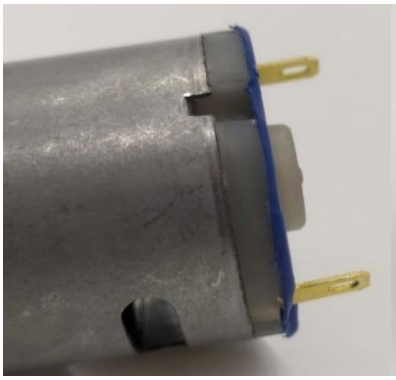
*Шаг 8*



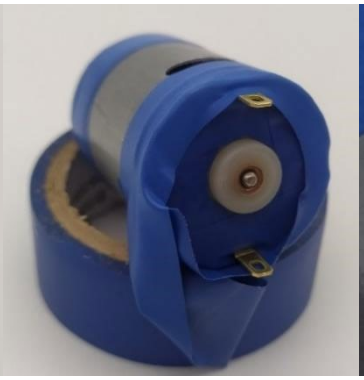
*Шаг 9*



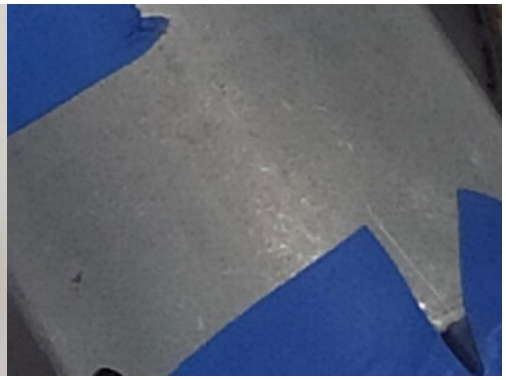
*Шаг 10*



*Шаг 11*

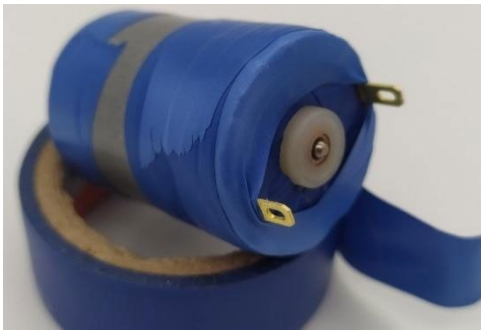


*Шаг 12*

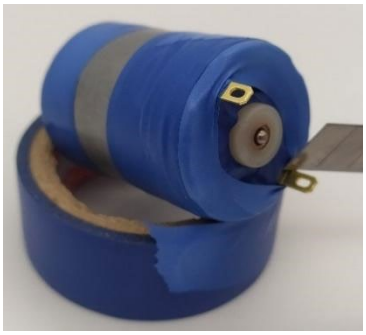


*Шаг 13*

В завершении нужно подрезать изоленту возле контактов, чтобы она не задиралась, и приклеить её (Шаг 15). Последними несколькими оборотами изоленты закрепляем предыдущие намотки, а также не оклеенную боковую поверхность мотора.



*Шаг 14*



*Шаг 15*

*Шаг 16*

### Покрытие воском

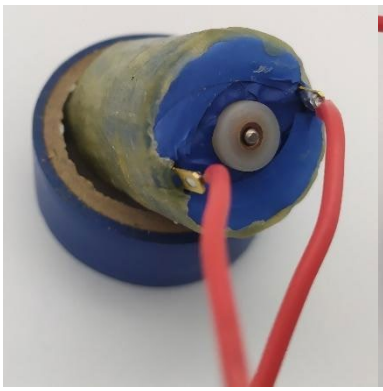
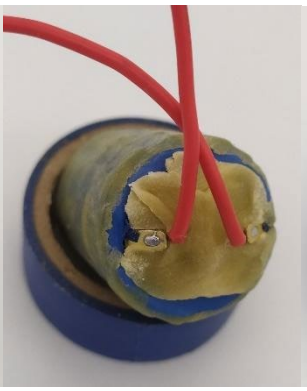
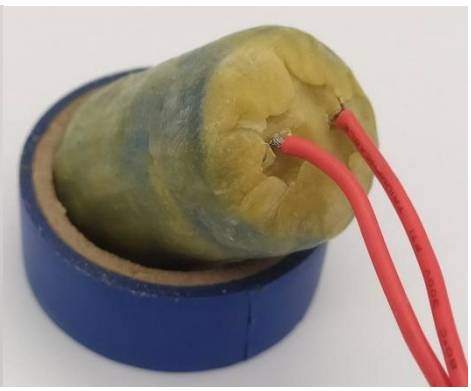
Разминаем воск (нагреваем его руками), чтобы он стал мягким (Шаг 17). Формируем толстый «блинчик» и нанизываем его на мотор (Шаг 18). Максимально сильно вдавливаем его в верхний торец и начинаем растягивать по стенкам моторчика (Шаг 19). Слой должен быть тонким, но без пробоев (если они, вдруг появятся, тщательно замажьте их кусочком воска).

*Шаг 17*

*Шаг 18*

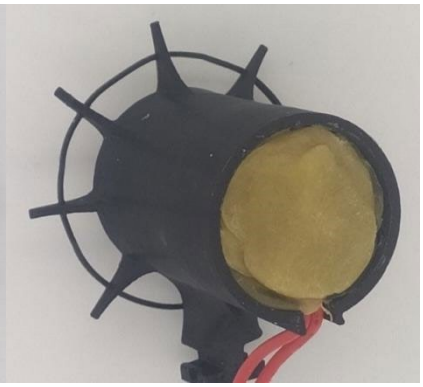
*Шаг 19*

Затем припаиваем провода и залепляем воском, предварительно размяв его, нижний торец мотора (Шаги 20-22).

*Шаг 20* *Шаг 21* *Шаг 22*

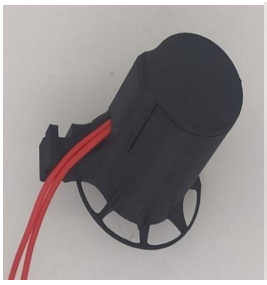
Вставляем мотор в корпус. Он должен заходить плотно (Шаг 23). Если мотор застревает, то нужно его вытащить и размять воск в месте, где он мешает входу мотора в корпус, сделав толщину воска меньше. Оставшееся пространство сверху нужно заполнить воском (Шаг 24).

*Шаг 23* *Шаг 24*

## Сборка движителя

Далее укладываем в канал провода и надеваем крышку (Шаги 25-26). Приклеиваем винт на цианакрилатный (супер) клей (Шаг 27). Рекомендуем перед приклеиванием отшлифовать винт наждачной бумагой или надфилем, чтобы лопасти стали гладкими.



*Шаг 25* *Шаг 26* *Шаг 27*

Запрессовываем гайки в насадку (Шаги 28-29).

*Шаг 28*

*Шаг 29*

Вставляем корпус в насадку до щелчка (Шаг 30). Двигатель готов.

*Шаг 30*

## Задачи со специальным оборудованием

### Задача 18 (тип 4)

Для выполнения задачи 18 необходим набор ElementaryROV
https://robocenter.net/goods/kit/elementaryrov/

Ссылка на методические материалы 16 занятий с планами занятий и презентациями.

# ElementaryROV

## КРАТКОЕ ОПИСАНИЕ

Набор состоит из комплектующих и материалов и предназначен для разработки и сборки телеуправляемого необитаемого подводного аппарата (ТНПА). Рассчитан на школьников 7-10 лет. Способствует изучению основ конструирования, электроники и программирования.

Набор позволяет изготовить ТНПА для участия в соревнованиях MATE Russia Far East ROV Competition в категории Scout.

## КОМПЛЕКТАЦИЯ

- соединительная коробка
- платы и компоненты для пульта управления
- 3 движителя
- коробка для пульта управления
- кабель
- батарейный отсек
- ПО для мироконтроллера
- элементы рамы
- элементы плавучести
- крепеж
- расходные материалы (герметик, провода и др.)

## ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

**ОБЩИЕ**

| | |
|---|---|
| Вес на воздухе (прибл.) | 2 кг |
| Глубина погружения | до 5 м |
| Пульт управления | на базе Arduino |
| Язык программирования | C++ |
| Кол-во движителей | 3 |
| Длина кабеля | 5 м |

**ПРОИЗВОДИТЕЛЬНОСТЬ**

| | |
|---|---|
| Тяга одного движителя | до 0,14кгс<br>до 0,13 кгс на севере |
| Батарейный отсек | 3 аккумулятора 18650 |
| Электропитание ТНПА | 12 В<br>до 2,5 А |
| Автономность (от 3 аккум.) | до 3 часов |

ООО «Центр робототехники»
г. Владивосток, ул. Комсомольская, 1
+79242580541, info@murorobot.com

## Задача 19 (тип 4)

Для выполнения задачи 19 необходим набор MiddleROV
https://robocenter.net/goods/kit/middlerov/

Ссылка на методические материалы 16 занятий с планами занятий и презентациями.

robocenter.net

# MiddleROV

## КРАТКОЕ ОПИСАНИЕ

Набор состоит из комплектующих и материалов и предназначен для разработки и сборки телеуправляемого необитаемого подводного аппарата (ТНПА). Рассчитан на школьников 11-17 лет. Способствует изучению программирования микроконтроллеров, систем автоматического управления, основ конструирования подводных аппаратов, схемотехники.

Набор позволяет изготовить ТНПА для участия в соревнованиях, таких как MATE ROV Competition категории Scout и Navigator. Также с помощью ТНПА можно проводить исследования в реальных условиях.

Данный набор не является конструктором, учащимся самим необходимо спроектировать раму, произвести электромонтаж блока электроники и собрать аппарат целиком.

## КОМПЛЕКТАЦИЯ

- Герметичные корпуса
- платы и компоненты для блока электроники
- 3 движителя
- 1 поворотная камеры
- 1 захват
- пульт управления с джойстиком и ТВ
- ПО для мироконтроллера
- кабель
- элементы плавучести
- материал для изготовления рамы*
- крепеж
- расходные материалы (герметик, провода и др.)

*опционально можем изготовить на ЧПУ вашу раму

## ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

| | |
|---|---|
| Вес на воздухе (прибл.) | 10 кг |
| Глубина погружения | до 10 м |
| Бортовая электроника | на базе Ардуино Искра |
| Связь с роботом | UAR |
| Язык программирования | C++ |
| Длина кабеля | 20 м |
| Электропитание | 12 В<br>до 15 А |
| Пульт управления | вход-ое напр. 220 В |

| | |
|---|---|
| Кол-во движителей | 3 |
| Тяга одного движителя (при 12 В) | до 7 кгс<br>до 0,85 кгс на реверсе |
| Кол-во камер | 1 |
| Тип сигнала | Аналоговый |
| Угол обзора камеры | 72 градуса |
| Вращение камеры | 180 градусов |
| Разрешение камеры | 976x494 |
| Кол-во подводных захватов | 1 |

ООО «Центр робототехники»
г. Владивосток, ул. Комсомольская, 1
79242582941, info@rcrobproject.com