Аэрокосмические системы

Заключительный этап

Инженерный тур

Общая информация

В рамках командного тура заключительного этапа профиля «Аэрокосмические системы» 2023 года команда участников должна будет использовать как элементы «слабого искусственного интеллекта» для управления космической миссией на другой планете, так и телеуправление по узким радиоканалам.

При помощи инопланетного ровера, с которым есть только сеансовая связь, команда участников должна провести разведку местности на условной удаленной планете, а также собрать достаточное количество интересующих артефактов.

Команда участников должна запрограммировать ровера, находящегося на удаленном полигоне, для автономного выполнения части миссии, а также для выполнения других частей миссии в режиме телеуправления.

В процессе программирования и отладки работоспособности полезной нагрузки (ПН) ровера команда должна пройти ряд испытаний, проверяющих, как функционируют отдельные узлы ровера.

После предварительных испытаний, начинается этап испытаний на полигоне. На полигоне ровер, запрограммированный командой участников, должен пройти через 4 зоны полигонных испытаний и в каждой выполнить свою задачу.

Сюжет задачи

Как вы знаете, в связи с тем, что связь с далекими планетам идет с задержкой, мы не можем управлять миссией с Земли. А зачастую не можем управлять межпланетной миссией даже с орбитальной станции этой планеты. Для того чтобы межпланетная миссия была выполнена, все чаще идут разговоры о применении искусственного интеллекта в межпланетных роверах. Такие разумные роверы смогут автономно решать задачи миссии без постоянного контроля со стороны человека. Тем не менее полностью исключить возможность участия человека в управлении планетоходами пока не представляется возможным.

Решение задачи финала разбито на предварительные испытания, квалификацию и финальные заезды в четырех зонах полигона.

В первой зоне необходимо будет в полностью автоматическом режиме осуществить поиск и разведку 4-х точек на полигоне. Точки являются центрами контрастных кругов, расположенных в углах квадрата со стороной около 2 м. Ориентирование на местности надо будет осуществлять при помощи одометрии с колес ровера и компьютерного зрения. Для управления ровером в этой зоне команда должна написать соответствующий алгоритм, который в полностью автоматическом режиме без участия человека будет управлять перемещениями ровера.

Во второй зоне надо будет осуществить поиск и фотографирование неких артефактов, параметры которых станут известны командам непосредственно перед стартом. Поиск и фотографирование должно осуществляться в режиме телеуправления.

Для управления ровером в этой зоне команда должна написать соответствующий протокол связи, который в режиме узкого канала связи будет позволять управлять перемещениями ровера и передавать фотографии.

В третьей зоне надо будет осуществить перемещение объектов при помощи разработанного командой манипулятора. Для управления ровером в этой зоне команда должна написать соответствующий протокол связи, который в режиме узкого канала связи будет позволять управлять перемещениями ровера и перемещением манипулятора.

В четвертой зоне необходимо будет завезти ровер во взлетный модуль. Модуль представляет собой кубическую конструкцию размерами немного больше ровера. Для доступа в модуль (открытия аппарели) необходимо будет передать по радиоканалу, прослушиваемому модулем, специальную последовательность бит, которую команда участников узнает непосредственно перед стартом. Для управления ровером в этой зоне команда должна написать соответствующий протокол связи, который в режиме узкого канала связи будет позволять управлять перемещениями ровера и осуществлять передачу кода открытия аппарели на взлетный модуль.

Во время решения командной задачи команда не имеет контакта с ровером. Команда получает оценку за совокупность решений и вольна выбирать любой подход к программированию, настройке ровера и изготовлению полезной нагрузки.

Требования к команде и компетенциям участников

В команде участников должны быть электронщики, программисты ROS и конструкторы. Они должны обладать следующими компетенциями:

1. **Электронщики**: программирование в Arduino IDE/знание основ электроники/умение паять
2. **Программисты ROS**: администрирование Linux/ROS/программирование на Python
3. **Конструкторы**: умение работать в САПР
4. Возможно совмещать указанные компетенции в одном участнике команды, тогда команды могут быть по 2 человека, но мы рекомендуем не перегружать одного из участников команды, т. к. общая нагрузка задания финала рассчитана на то, что все 3 участника работают параллельно.

Оборудование и программное обеспечение

Организаторы обеспечивают:

1. Лабораторную зону с удаленным доступом.
2. Полигон.
3. Сервер Discord для взаимодействия с организаторами и друг с другом во время командной работы.
4. Доступ к ютуб-трансляции с внешней видеокамеры полигона для общего обзора полигона. Трансляция будет проводиться с задержкой и предназначена исключительно для общего понимания обстановки на полигоне.

5. Изготовление деталей методом лазерной резки по чертежам, присланным командой участников.
6. Работу «аватаров».

Базовый набор для каждой команды-участника

- Ровер под управлением Robot Operating System (ROS).
- Набор крепежа и расходных материалов.
- Сетевой доступ к роверу оборудованному радиостанцией и Raspberry pi 4, оборудованной радиостанцией. У каждой команды будет сетевой доступ к роверу и Raspberry pi 4. В момент соревнований доступ только к Raspberry pi 4.

Программное обеспечение

Мы рекомендуем установить следующее программное обеспечение на ноутбуки команды. Перечислено только специфическое программное обеспечение, предполагая наличие стандартного ПО:

- Windows 10 (license).
- Ubuntu 20.04 LTS (freeware).
- OpenVPN для удаленного доступа.
- ROS Noetic Ninjemys (freeware).
- Autodesk Inventor 22 (Education license or trial).
- Adobe Acrobat (freeware).
- Arduino IDE (freeware).

Полигон

Полигон представляет собой ровное пространство размерами примерно 20×40 м, ограниченное разметкой по зонам выполнения задач. В зависимости от погоды это будет или полигон под открытым небом или закрытое помещение. В случае закрытого помещения, могут быть внесены изменения в регламент заездов.

Описание ключевых зон полигона:

- **1-я зона или зона разведки** — в данной зоне расположены точки для разведки. Точки представляют собой круги контрастного цвета.
- **2-я зона или зона поиска** — в данной зоне расположены предметы для поиска и фотографирования. Тип, цвет и форму предметов, организаторы скажут участникам на старте.
- **3-я зона или зона перемещения** — в данной зоне расположены предметы для перемещения.
- **4-я зона или зона взлетного модуля** — в данной зоне расположен взлетный модуль.

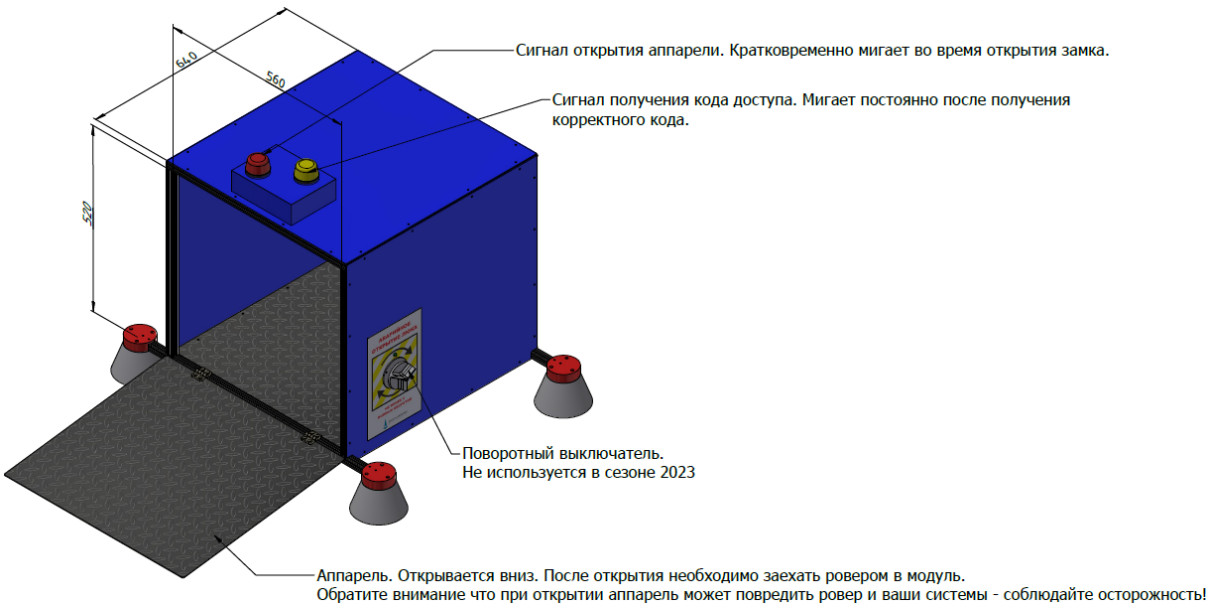


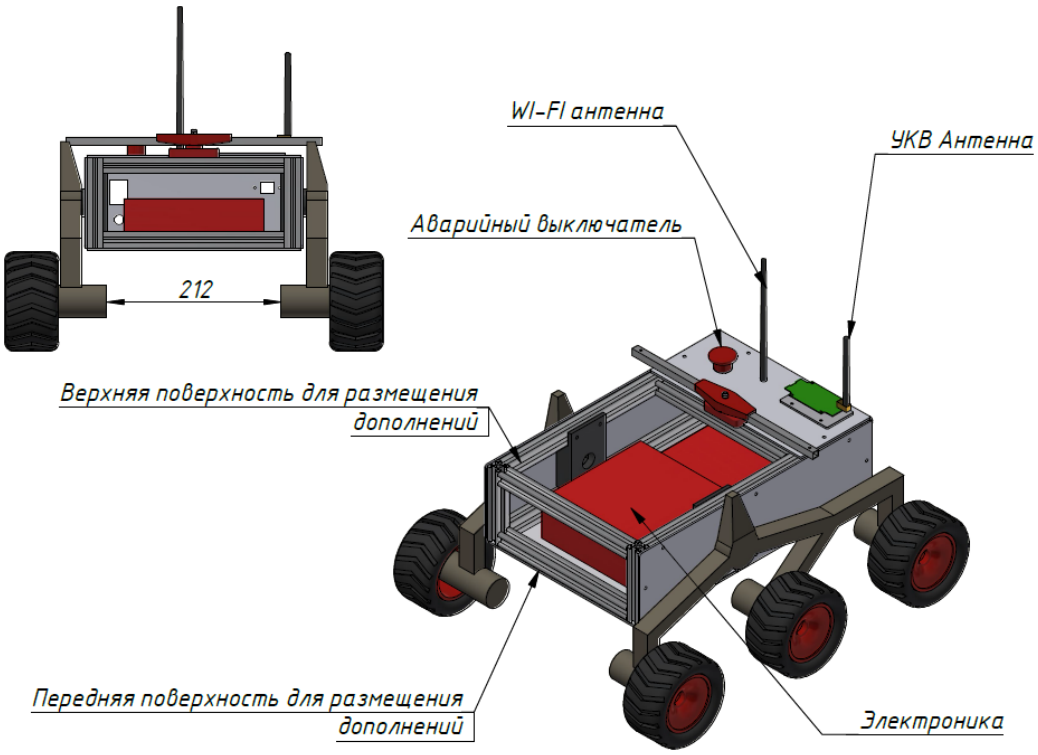
Схема модуля: https://disk.yandex.ru/i/x_EjnIjZXsgdZQ.

3D-модель модуля: https://disk.yandex.ru/d/eigtTwSRViu9LQ.

Ровер-планетоход

Для решения задачи финала командам выдается доступ на ровер, расположенный в удаленной лаборатории.

Общие рекомендации по управлению и программированию ровера и его полезной нагрузки (http://learn.voltbro.ru). При проверке работоспособности, настройке и программировании ровера рекомендуется пользоваться инструкцией: https://voltbro.gitbook.io/turtlebro/ (инструкция ровера TurtleBro).



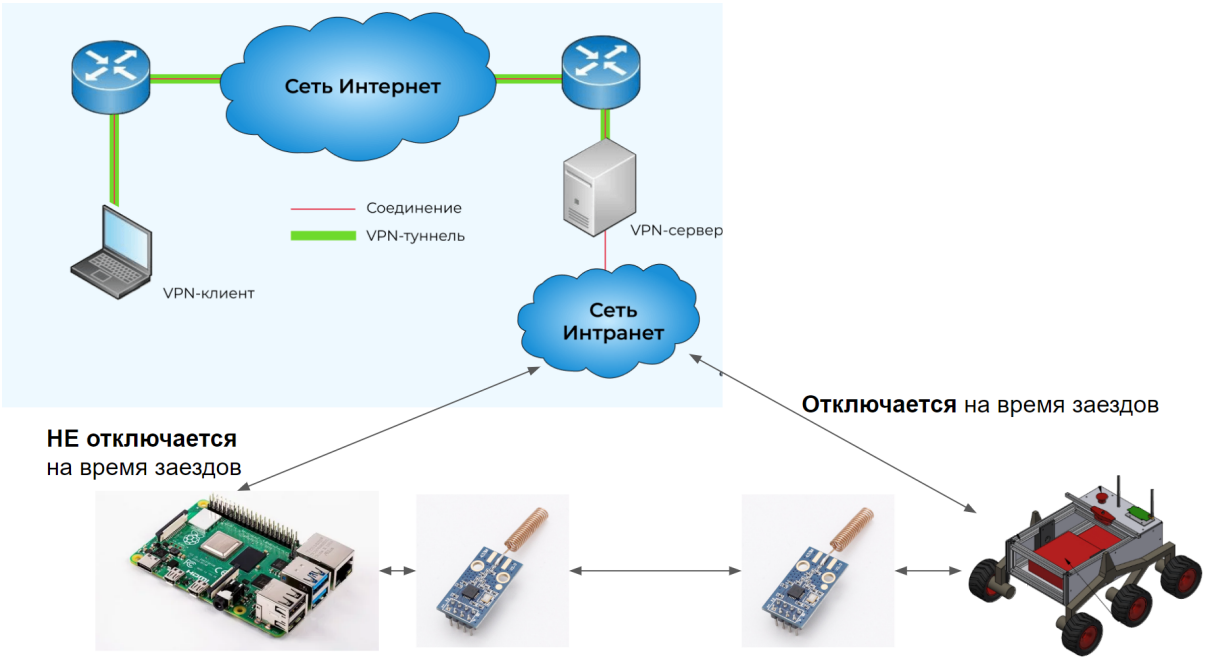


Рис. VI.2.1. Общая схема подключения на полигоне для каждой из команд

Описание задачи

Для выполнения задачи командного тура команда участников должна самостоятельно написать программу управления, которая будет управлять ровером и его дополнительным оборудованием, разработанным вашей командой.

Кроме того, команде необходимо разработать протокол управления ровером в условиях узкого радиоканала. Для реализации этих функций команда должна решить несколько предварительных заданий, проверяемых предварительными испытаниями. После прохождения предварительных испытаний команды будут допущены к выполнению заданий на полигоне.

Для решения задачи финала командам необходимо самостоятельно разработать полезную нагрузку для ровера, позволяющую выполнить задания. Какая именно полезная нагрузка должна быть изготовлена и смонтирована — описано в каждом из заданий отдельно.

Материалы и комплектующие для изготовления полезной нагрузки предоставляют организаторы. Также, в связи с дистанционным форматом проведения соревнования, организаторы предоставляют техников-аватаров, которые будут собирать полезную нагрузку, разработанную командами.

Регламент предоставления чертежей на изготовление и сборку полезной нагрузки описан далее.

Полезная нагрузка должна учитывать возможность решения всех заданий. Обязательно ознакомьтесь с задачами, которые предстоит решать.

Полезная нагрузка может крепиться только на переднюю панель ровера и верхнюю переднюю часть панели ровера. Необходимые панели проектируются вами самостоятельно. Для крепления в пазы профиля необходимы отверстия диаметром 4,5 мм расположенные ровно по центру пазов. Используйте 3D-модель для проверки.

Места для крепления ваших конструкций на этих двух панелях произвольны, однако вы не должны пересекаться с блоком электроники (отмечен красным) и не забывать о прокладке кабелей. Для крепления кабелей можно использовать стяжки — места креплений должны быть обозначены в документации.

3D-Модель ровера: https://disk.yandex.ru/d/oNjEKd2fjkUyJA.

Полезная нагрузка проектируется и описывается участниками команды, однако, поскольку соревнования предполагают удаленное участие, то собирается она техниками-аватарами. Для того чтобы это осуществлялось наиболее успешно — внимательно изучите дополнительный раздел настоящего задания «**Регламент дистанционного взаимодействия участников с организаторами по подготовке полезной нагрузки к соревнованиям**».

Только внимательно следуя правилам подготовки проектов, описанных в этом разделе, ваша команда имеет шанс решить задачи и выйти на старт.

Перечень предоставляемых деталей и расходных элементов для сборки полезной нагрузки, все 3D-модели деталей и элементов находятся в архиве: https://disk.yandex.ru/i/qPb_IKlN7vJBlQ.

Перечень доступных материалов и компонентов: https://disk.yandex.ru/d/YvDNW9ghs5RhnA.

Обратите внимание, что первый в день работы полигон функционирует в режиме тестирования оборудования. Работа в этом режиме предполагает возможное отключение доступа к полигону у всех участников, при наличии проблем с доступом у какого-то одного участника. Это делается для обеспечения равных условий всем участникам финала.

Обратите внимание, что для отчета по некоторым из подзадач надо снять видео. Для этого команда должна команде необходимо создать облачное и предоставить организаторам ссылку и права на скачивание файлов из него. Предварительные задания считаются сданными, если видео, демонстрирующее решение каждого конкретного задания, закачено в облачное хранилище команды. Временем сдачи предварительных заданий считается время окончания загрузки соответствующего видео. Планируйте ваше время!

Общие требования ко всем видео:

- Наименование файла видео должно быть сделано в формате: `НазваниеКоманды_номер_подзадачи.avi` или `mpg`.
- Видео надо снимать в разрешении достаточном для того, чтобы можно было прочитать команды в терминале и код программы.
- Необходимо сопровождать видео комментариями о происходящем с роботом и на экране.
- На команде участников лежит ответственность за демонстрацию правильности выполнения задания. К примеру, если есть требование показать работоспособность протокола радиосвязи, то необходимо продемонстрировать организаторам отключение WiFi.

Задача VI.2.5.1. В зоне разведки

Условие

Команде участников необходимо написать программу для ровера, которая в автоматическом режиме проведет ровер по четырем точкам разведки. Для ориентирования ровера должны использоваться алгоритмы компьютерного зрения на основании видеопотока с камеры ровера. В данной зоне доступ к роверу через радиоканал будет только до момента старта. После старта ровер должен выполнять все этапы задания в полностью автоматическом режиме.

Точка разведки представляет из себя шар диаметром 70 мм контрастного цвета.

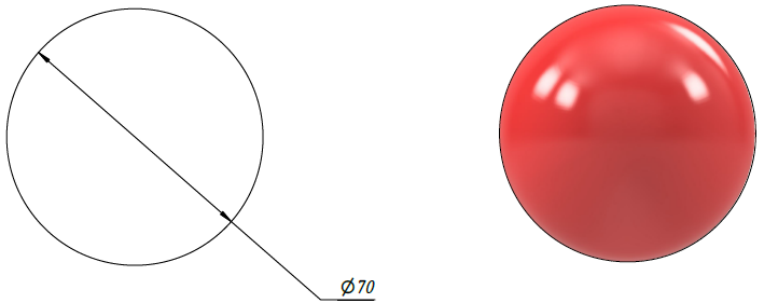


Схема шара: https://disk.yandex.ru/i/CUJLfbuWkQGJpw.

3D-модель шара: https://disk.yandex.ru/d/KCkf2dgJLD4p4g.

Для выполнения задания команда должна создать полезную нагрузку в составе:

1. Кнопка на верхней панели ровера. Данная кнопка будет использована организаторами для запуска и продолжения работы программы ровера. В рамках создания полезной нагрузки техники-аватары подключат данную кнопку туда, куда будет указано в инструкции по сборке, полученной от команды участников. Для расположения кнопки используйте пожалуйста верхнюю панель ровера.
   3D-модель кнопки: https://disk.yandex.ru/d/9wbPsazPL5ucnA.
2. Для получения данных с камеры команде участников необходимо спроектировать и при помощи техников-аватаров собрать поворотное крепление для камеры. Команде доступна одна камера.
   3D-модель камеры: https://disk.yandex.ru/d/0LSBd7AjC3uEOw.

Вы можете произвольно проектировать крепление камеры, учтите, что она вам понадобится в нескольких заданиях. Обратите внимание, что от камеры идет USB-кабель толщиной 5 мм. Для подключения к камере провода со стороны разъема необходимо оставить хотя бы 1 см зазора.

Сценарий выполнения задания

1. Организаторы совместно с командой участников выставляют ровер в стартовую позицию, не совпадающую ни с одной из точек разведки. После чего нажимают на ровере на кнопку. И одновременно включают таймер.
2. Ровер в полностью автоматическом режиме должен найти ближайшую точку разведки, подъехать к ней и остановиться таким образом, чтобы проекция

точки центра передней панели рамы ровера на плоскость точки разведки находилась на минимальном расстоянии от центра точки разведки. После полной остановки организаторы замеряют расстояние до центра точки разведки.

3. После замера расстояния организаторы снова нажимают кнопку, после чего ровер в автоматическом режиме должен найти следующую точку разведки, расположенную против часовой стрелки при виде сверху, и повторить п. 2.
4. После полной остановки у 4-й точки разведки задание считается выполненным. В случае если время выполнения задания закончилось раньше, чем ровер доехал до какой-то точки, в зачет идут баллы за посещенные точки разведки.

Для решения этой общей задачи команде необходимо решить задачи для предварительных испытаний, которые не только дадут дополнительные баллы, но и позволят контролировать прогресс продвижения. Кроме того, в целях отработки решения и калибровки камер, роверам команд будет предоставлен доступ к тестовой точке разведки. Данная точка будет находиться в прямой видимости камер роверов и геометрически, и цветом будет полностью аналогична точкам разведки на полигоне.

Обратите внимание, что в зависимости от погоды возможно решение задания на полигоне под открытым небом. В таком случае для калибровки цвета точки разведки организаторы предоставят доступ командам к тестовой точке разведки, находящейся непосредственно в условия, в которых будет решаться задание.

Задачи для предварительных испытаний:

1.1 Подключиться к роверу по ssh.

1.2 Проверить, что ровер корректно реагирует на команды управления, заданные из терминала ssh. Для проверки необходимо продемонстрировать организаторам движение ровера на расстояние не менее 0,5 м вперед и поворот на не менее 45° в любую сторону.

1.3 Загрузить на Arduino ровера скетч, реализующий движение вперед на 0,5 м при нажатии организаторами кнопки.

1.4 Написать программу, которая в автоматическом режиме найдет тестовую точку разведки и подведет к ней робота (в соответствии с требованиями основного задания №1).

1.5 Продемонстрировать организаторам общее решение из задач 1.3 и 1.4. Т. е. при нажатии на кнопку робот в автоматическом режиме ищет тестовую точку разведки, подъезжает к ней и останавливается около нее в соответствии с требованиями основного задания № 1.

Отчетность по предварительным испытаниям:

1.1 Видео, на котором видно, как участник команды подключился по SSH к роверу (задача 1.1).

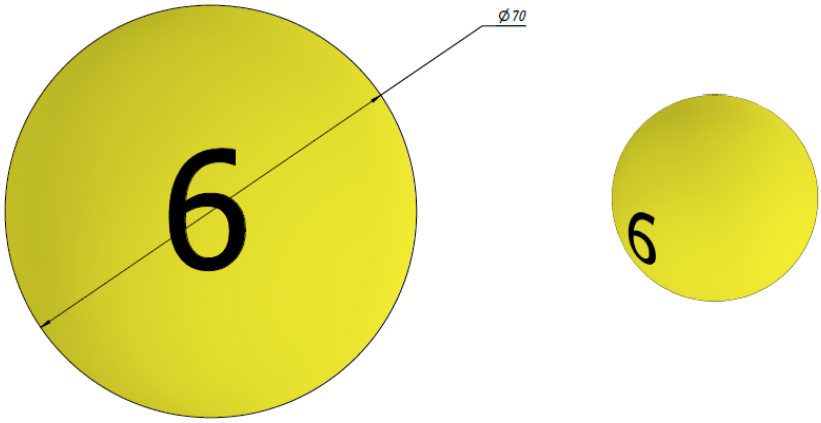
1.2 Видео, на котором видно, как участник команды подключившись по SSH к роверу, передает команды на управление ровером и как по камерам ровера видно, что команды приводят к перемещению ровера (задача 1.2).

Задача VI.2.5.2. В зоне поиска

Условие

Команде участников необходимо найти и «сфотографировать» в зоне выполнения задания некоторые объекты, которые представляют собой шары диаметром 70 мм с отметками. Цвет шаров — произвольный, Отметка — произвольная читаемая.

Пример шара с отметкой в виде цифры.



Шары могут быть разложены в произвольных местах зоны поиска.

Ответом на задание являются изображения артефактов залитые в хранилище команды на Яндекс Диске.

Для управления ровером в этой и всех других зонах используется телеуправление по радиоканалу. WiFi — не будет! Для этого команде участников необходимо разработать протокол телеуправления по радиоканалу. Для этого нужно использовать следующие вводные данные.

Для работы с радиосвязью в УКВ-диапазоне используются цифровые радиомодули модули SV610. Модули используют интерфейс UART, и через переходник UART-USB на основе микросхемы CP2102 подключены в USB разъем микрокомпьютеров Raspberry PI. Подключение и модули одинаковы и на радиостанции, и на ровере. Выход SET радиомодуля подключен к DTR выходу USB-UART преобразователя. Вход CS радиомодуля присоединен к 3,3 В.

Документация к радиомодулю: https://disk.yandex.ru/i/eQmtghiYhXmxeg.

Команда может самостоятельно настраивать любые параметры связи (скорость и другие настройки радио). Решение вопросов защиты вашего канала связи (чтобы другие команды не могли вам вносить помехи) возлагается на членов команды. Распределение команд по радиоканалам будет проведено организаторами в первый день финала по заявкам участников.

Обратите внимание: в день финальных заездов на радио модулях будет отключена возможность менять настройки с целью избежания возможности вмешаться в работу другой команды.

Сценарий выполнения задания

1. Организаторы совместно с командой участников выставляют ровер в стартовую позицию и запускают таймер выполнения задания.

2. Команда участников при помощи ровера, управляемого по радиоканалу, настроенному командой, ищет артефакты.
3. При нахождении артефакта команда должна сохранить изображение найденного артефакта на местности в папке команды на Яндекс диске и дублировать его в дискорд-канале команды.
4. Задание считается законченным если команда нашла все артефакты в зоне выполнения задания или время на выполнения задания истекло.

Обратите внимание, что при выполнении задания для управления ровером доступна лишь радиосвязь. WiFi и VPN, доступные при подготовке, будут отключены.

Обратите внимание, что изображения артефактов должны быть сделаны таким образом, чтобы наибольший линейный размер артефакта (в пк) на изображении был не менее 20% от значения ширины изображения (в пк).

Для решения этой общей задачи команде необходимо решить промежуточные задачи.

Для создания, настройки и тестирования алгоритмов управления ровером по радиоканалу командам участников будут предоставлены:

- VPN доступ на Raspberry PI 4 к которой подключен радиопередатчик.
- VPN доступ на ровера, к которому тоже подключен радиопередатчик.

Задачи для предварительных испытаний:

2.1 Написать протокол управления ровером по радиоканалу. Продемонстрировать организаторам перемещение ровера на расстояние не менее 1 м вперед и поворот на не менее 45° в любую сторону при управлении ровером по радиоканалу.

2.2 Продемонстрировать организаторам видео с камер ровера передаваемое по радиоканалу.

2.3 Продемонстрировать организаторам работу программы получения фрагментов изображения из видеопотока с камер ровера передаваемого по радиоканалу.

Отчетность по предварительным испытаниям:

2.1 Видео с камер робота (задача 2.2).

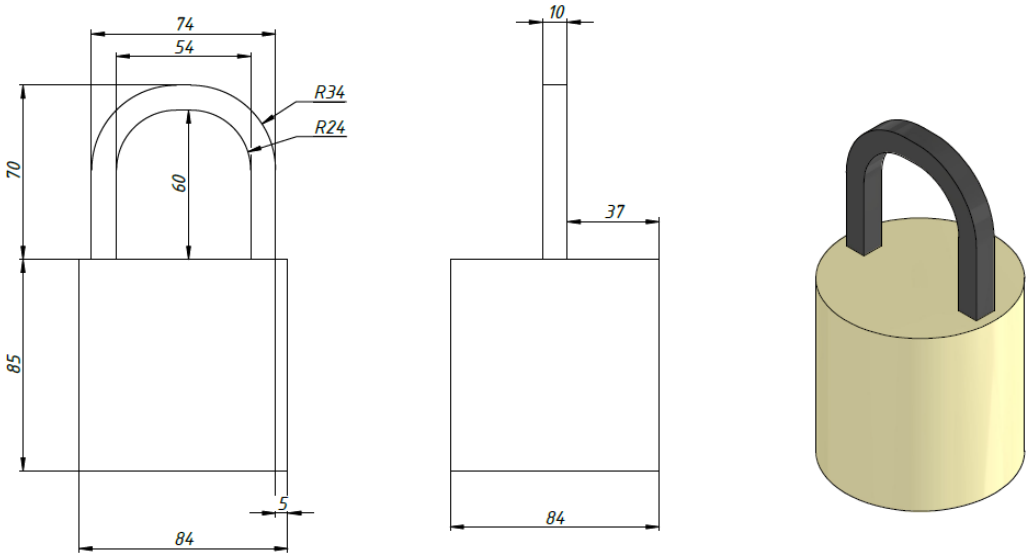
2.2 Файл изображения с камер робота (задача 2.3).

Задача VI.2.5.3. В зоне перемещения

Условие

Команде участников необходимо написать программу для ровера, которая в режиме управления по радиоканалу, при помощи разработанной командой полезной нагрузки позволит осуществить перемещение между двумя точками нескольких одинаковых предметов.

Предмет, который необходимо переместить, весит 75 г и имеет вот такую форму.



Характеристики предмета: https://disk.yandex.ru/i/l_3WQZVMlvwE8A.

3D-модель предмета: https://disk.yandex.ru/d/mGdo5BxGhfsdaA.

Предмет располагается на уровне земли. Надо обеспечить возможность захвата и перевозки предмета по пересеченной местности. После перевозки должна быть возможность поставить предмет на землю. При перевозке предмет не должен задевать землю.

Расстояние между точкой, где находятся предметы и точкой, куда их надо переместить несколько метров. Границы точки, в которой расположены предметы, и границы точки, куда их надо переместить, обозначены контрастной линией.

Сценарий выполнения задания

- Организаторы совместно с командой участников выставляют ровер на стартовую позицию в зоне выполнения задания. Включают таймер и сообщают об этом команде участников.
- Команда участников, используя управление по радиоканалу, подводит ровер к точке, в которой находятся предметы для перемещения.
- Команда участников, используя управление по радиоканалу управляя полезной нагрузкой ровера, «захватывает» предмет для перемещения и поднимает его на высоту не менее нижней поверхности рамы ровера.
- Ровер с захваченным предметом в режиме телеуправления переезжает к точке, где надо оставить предмет.
- Команда участников, используя управление по радиоканалу и управляя полезной нагрузкой ровера, помещает предмет внутрь границы точки для расположения предметов.
- После того как команда убеждается, что предмет надежно расположен внутри границы точки для расположения предметов, она может направить ровер за следующим предметом.
- Задача — переместить таким образом как можно большее количество предметов.
- Задание считается выполненным после того, как все предметы перемещены или закончилось время выполнения задания

Для решения этой общей задачи команде необходимо решить промежуточные задачи, которые не только дадут дополнительные баллы, но и позволят контролировать прогресс продвижения.

Для создания, настройки и тестирования алгоритмов управления ровером по радиоканалу командам участников будут предоставлены:

- VPN доступ на Raspberry PI 4 к которой подключен радиопередатчик.
- VPN доступ на ровера, к которому тоже подключен радиопередатчик.

Задачи для предварительных испытаний:

3.1 Прислать чертежи полезной нагрузки в соответствии с регламентом.

3.2 Собрать полезную нагрузку при помощи техника-аватара.

3.3 Продемонстрировать организаторам минимальную работоспособность полезной нагрузки при управлении по Wi-Fi, т. е. возможность полезной нагрузки «захватить» и «отпустить» предмет для перемещения.

3.4 Продемонстрировать организаторам возможность полезной нагрузки «захватить» и «отпустить» предмет для перемещения при управлении по радиоканалу.

Отчетность по предварительным испытаниям:

3.1 Видео скринкаста управления полезной нагрузкой (задача 3.3).

3.2 Видео скринкаста управления полезной нагрузкой (задача 3.4).

Задача VI.2.5.4. В зоне взлетного модуля (ВМ)

Условие

Команде участников необходимо в режиме телеуправления по радиоканалу завезти ровера во взлетный модуль.

Посадочный модуль, представляет собой следующую конструкцию.

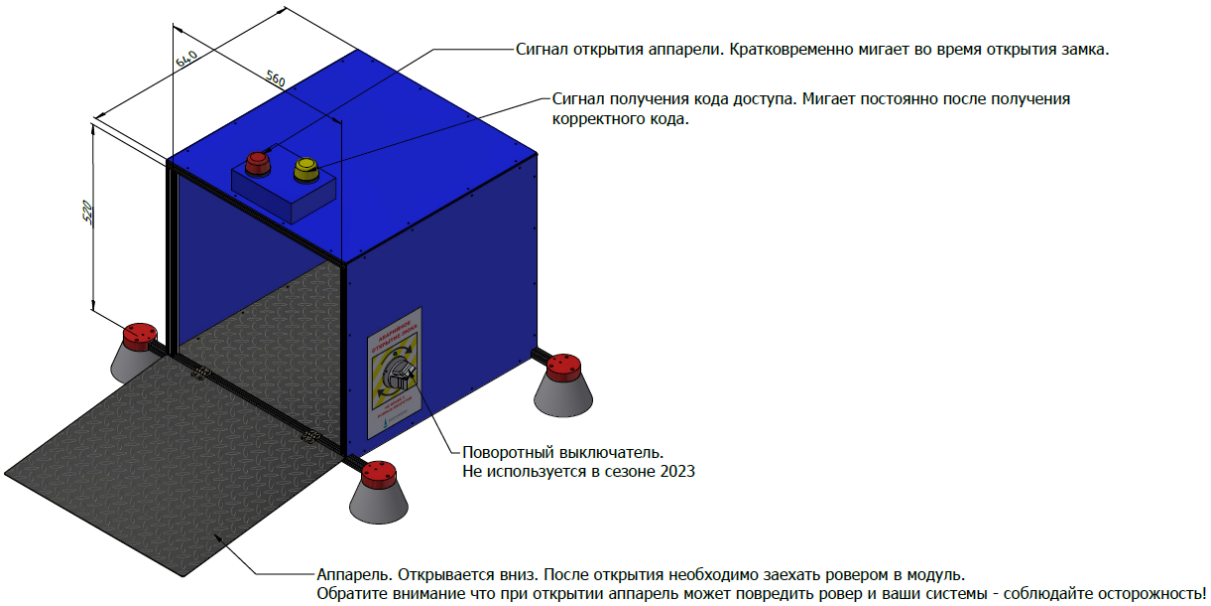


Схема модуля: https://disk.yandex.ru/i/x_EjnIjZXsgdZQ.

3D-модель модуля: https://disk.yandex.ru/d/eigtTwSRViu9LQ.

Перед тем как заехать в модуль, необходимо открыть аппарель. Открытие аппарели происходит при помощи передачи последовательности символов на взлетный модуль, используя радиомодуль HC-12, установленный в вашем ровере.

Последовательность символов и параметры радиоканала взлетного модуля будут сообщены команде непосредственно перед стартом задания.

В целях тестирования на полигоне будет расположен взлетный модуль, оборудованный тестовым передатчиком, на котором команды могут отлаживать свою программу.

Обратите внимание: все разработанное вами дополнительное оборудование не должно препятствовать заезду в модуль и закрытию аппарели.

Сценарий выполнения задания

- Организаторы совместно с командой участников выставляют ровер на стартовую позицию в зоне выполнения задания. Сообщают команде участников последовательность символов, открывающую взлетный модуль. Включают таймер и сообщают об этом команде участников.
- Команда участников, используя управление по радиоканалу, подводит ровер к взлетному модулю.
- Команда участников передает последовательность символов на взлетный модуль.
- После открытия аппарели команда участников заводит ровер внутрь взлетного модуля.
- Задание считается выполненным после того, как ровер целиком окажется внутри взлетного модуля или закончилось время выполнения задания.

Задачи для предварительных испытаний.

4.1 Написать программу, передающую заданную последовательность символов на тестовый радиопередатчик, и продемонстрировать организаторам ее работоспособность.

Предварительные испытания (ПИ)

Для участия в ПИ, описанных выше, любой участник команды пишет организаторам в Discord, и в порядке общей очереди предоставляет решение задания ПИ. Организаторы прекращают прием заявок на ПИ за 30 мин до окончания работ каждого дня. Организаторы проверяют работоспособность ровера в соответствии с заданиями и выставляют оценки каждой из команд. За успешную проверку баллы начисляются, за непрохождение проверки баллы не начисляются. **Прохождение этапа ПИ не является необходимым для дальнейшего выполнения задания**, но дает возможность получить дополнительные баллы. Команда участников, не сдавшая подзадачи ПИ в день сдачи данного набора задач (см. таблицу №2 ниже), может сдать их в следующие дни, но потеряет при этом 30% баллов от максимального за каждый день просрочки сдачи ПИ.

В конце каждого дня команды должны сдать результаты за день.

Сдача результатов за день – это перегрузка в файловое хранилище команды всех сделанных за день продуктов работы команды (чертежи, файлы с кодом, видео и т. д.). На каждый день в хранилище команды должна быть создана отдельная папка: `День1`, `День2` и т. д.

В финале команды должны сдать соответствующие материалы в папки «`Финальные заезды`» → «`Задание №...`».

День	Задачи
06.03	• Общее тестирование оборудования и связи
07.03	• ПИ Задания №1 и ПИ Задания №2 • Сдача результатов работы за день
08.03	• ПИ Задания №3 и ПИ Задания №4 • Сдача результатов работы за день
09.03	• Квалификация • Сдача результатов работы за день
10.03	• Финальные заезды • Сдача результатов работы за день

Квалификация

Квалификация — это предварительный тестовый заезд, который показывает готовность команды к финальным испытаниям на полигоне. Квалификация проходит поэтапно в соответствии с заданиями заключительного этапа, и по своей сути каждый этап квалификации является упрощенной задачей заключительного этапа. Команды, не прошедшие квалификацию соответствующий этап квалификации, не допускаются до соответствующего финального задания на полигоне.

Задачи этапов квалификации

Номер этапа	**Название этапа**	**Сценарий квалификации**	**Минимальное выполнение задания квалификации для прохождения в финал**
1	**В зоне разведки**	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Организаторы нажимают на кнопку ровера, после чего ровер в автономном режиме должен начать выполнять программу. 3. Ровер должен обнаружить точку разведки. 4. После обнаружения ровер должен приехать в точку разведки и остановиться как можно ближе от ее центра (требования описаны в задаче №1). 5. После измерения точности остановки команда в режиме управления по радиоканалу должна вернуть ровер на старт.	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Организаторы нажимают на кнопку ровера, после чего ровер в автономном режиме должен начать выполнять программу. 3. Ровер должен обнаружить точку разведки и попытаться подъехать к ней.
2	**В зоне поиска**	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до первой точки поиска. 3. Команда делает фотографию точки поиска и присылает ее в чат команды в дискорд. 4. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции.	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до первой точки поиска. 3. Команда делает фотографию точки поиска и присылает ее в чат команды в дискорд.

Номер этапа	Название этапа	Сценарий квалификации	Минимальное выполнение задания квалификации для прохождения в финал
3	**В зоне перемещения**	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда участников самостоятельно в режиме управления по радиоканалу подъезжает к предмету для перемещения и приподнимает его при помощи изготовленной полезной нагрузки. 3. Команда переставляет предмет для перемещения в произвольное место, отстоящее от начального положения предмета на расстояние не менее 20 см. 4. Команда опускает предмет на пол полигона при помощи изготовленной полезной нагрузки. 5. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции.	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда участников самостоятельно в режиме управления по радиоканалу подъезжает к предмету для перемещения и приподнимает его при помощи изготовленной полезной нагрузки.
4	**В зоне взлетного модуля**	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда передает на радиомодуль ВМ указанную организаторами последовательность символов. 3. ВМ открывает аппарель. 4. Команда в режиме управления по радиоканалу доводит ровер до въезда на аппарель. 5. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции.	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. 2. Команда передает на радиомодуль ВМ указанную организаторами последовательность символов. 3. ВМ открывает аппарель.

Квалификация проходит на тестовом полигоне доступном команде участников с начала соревнований.

За каждое выполненное действие начисляются баллы.

Зачет идет по полноте и времени выполнения. Т. е. команды, которые полностью выполнили задания квалификации, но сделали это медленнее, имеют преимущество перед теми, которые не выполнили задание квалификации полностью.

Общее время на каждый этап квалификации 3 мин. Если в течение 30 с после старта ровер не начал движение, квалификация считается непройденной.

При наличии времени организаторы могут дать дополнительные попытки квалификации. При использовании каждой последующей попытки результат за прохож-

дение квалификации уменьшается на 10%.

Система оценивания

Итоговые оценки будут выставлены всем командам после завершения всех испытаний и окончания выполнения всех заданий. При наличии времени организаторы могут публиковать промежуточные результаты команд в процессе решения задачи.

После квалификации команды начинают основные испытания на полигоне в порядке, определенном квалификацией, а именно, команды получившие большее количество баллов в квалификации выступают позднее. Задания должны быть выполнены в соответствии с описанными сценариями.

Таблица VI.2.1: *Оценка испытаний на полигоне. Количество баллов, начисляемых за подзадачи, и количество баллов, снимаемых за штрафы, публикуется отдельно в оценочных протоколах.*

№	Задание	Штрафы и комментарии
0.	Квалификация в зоне разведки	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. Если команда не может этого сделать, попытка квалификации не засчитывается. 2. Организаторы нажимают на кнопку ровера, после чего ровер в автономном режиме должен начать выполнять программу. Если после истечения 30 с ровер не начал движение, попытка выполнения задания не засчитывается. 3. Потеря контроля над ровером — незачет попытки квалификации.*
	Квалификация в зоне поиска	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. Если команда не может этого сделать, попытка квалификации не засчитывается. 2. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до первой точки поиска. Если команда не может этого сделать, попытка квалификации не засчитывается. 3. Команда делает фотографию точки поиска и присылает ее в чат команды в дискорд. Если команда не может этого сделать, попытка квалификации не засчитывается.
	Квалификация в зоне перемещения	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. Если команда не может этого сделать, попытка квалификации не засчитывается. 2. Команда участников самостоятельно в режиме управления по радиоканалу подъезжает к предмету для перемещения и приподнимает его при помощи изготовленной полезной нагрузки. Если команда не может этого сделать, попытка квалификации не засчитывается.
	Квалификация в зоне ВМ	1. Команда участников самостоятельно в режиме управления по радиоканалу доводит ровер до стартовой позиции. Если команда не может этого сделать, попытка квалификации не засчитывается 2. Команда передает на радиомодуль ВМ указанную организаторами последовательность символов. Если команда не может этого сделать, попытка квалификации не засчитывается. Оценка успешности проходит по тому, открыл ли ВМ аппарель.

Таблица VI.2.1: *Оценка испытаний на полигоне. Количество баллов, начисляемых за подзадачи, и количество баллов, снимаемых за штрафы, публикуется отдельно в оценочных протоколах.*

№	Задание	Штрафы и комментарии
1.	В зоне разведки	1. После нажатия кнопки ровер должен начать движение в течение 30 с. Если после истечения 30 с ровер не начал движение, попытка выполнения задания не засчитывается. 2. Под полной остановкой понимается такое прекращение движения робота, которое позволит организаторам измерить расстояние от центра рамы ровера до центра точки разведки. 3. После замера расстояния организаторы снова нажимают кнопку, после чего ровер в также должен стартовать к следующей точке разведки в течение 30 с 4. После остановки у 4-й точки разведки задание считается выполненным. В случае если время выполнения задания закончилось раньше, чем ровер доехал до какой-то точки, в зачет идут баллы за посещенные точки разведки. 5. Потеря контроля программы над ровером — незачет попытки решения задания.*
2.	В зоне поиска	1. После запуска таймера выполнения задания ровер должен начать движение в течение 30 с. Если после истечения 30 с ровер не начал движение, попытка выполнения задания не засчитывается. 2. При нахождении артефакта команда должна сохранить изображение найденного артефакта на местности в папке команды на Яндекс диске. Максимальный линейный размер артефакта должен быть не менее 20% от ширины изображения в пк. 3. Задание считается законченным если команда нашла все артефакты в зоне выполнения задания, или время на выполнения задания истекло. В случае, если время выполнения задания закончилось раньше, чем ровер обнаружил все артефакты, в зачет идут баллы за найденные артефакты. 4. Потеря контроля над ровером — незачет попытки решения задания.*
3.	В зоне перемещения	1. Организаторы совместно с командой участников выставляют ровер на стартовую позицию в зоне выполнения задания. Включают таймер и сообщают об этом команде участников. После запуска таймера выполнения задания ровер должен начать движение в течение 30 с. Если после истечения 30 с ровер не начал движение, попытка выполнения задания не засчитывается. 2. Оценка по количеству перемещенных предметов проводится только по тем предметам, которые полностью находятся внутри границы зоны для расположения предметов. 3. Задание считается выполненным после того, как все предметы перемещены или закончилось время выполнения задания. В случае, если время выполнения задания закончилось раньше, чем ровер переместил все предметы, в зачет идут баллы за перемещенные предметы 4. Потеря контроля над ровером — незачет попытки решения задания.*

Таблица VI.2.1: *Оценка испытаний на полигоне. Количество баллов, начисляемых за подзадачи, и количество баллов, снимаемых за штрафы, публикуется отдельно в оценочных протоколах.*

№	Задание	Штрафы и комментарии
4.	В зоне взлетного модуля	1. Организаторы совместно с командой участников выставляют ровер на стартовую позицию в зоне выполнения задания. Сообщают команде участников последовательность символов, открывающую взлетный модуль. Включают таймер и сообщают об этом команде участников. После запуска таймера выполнения задания ровер должен начать движение в течение 30 с. Если после истечения 30 с ровер не начал движение, попытка выполнения задания не засчитывается. 2. После открытия аппарели команда участников заводит ровер внутрь взлетного модуля. Касание боковых стенок взлетного модуля — штрафные баллы. Падение ровера с аппарели — штрафные баллы. Под падением ровера с аппарели подразумевается такое неконтролируемое перемещение ровера, при котором одно или больше колес ровера оторвано от земли. Переворот ровера при падении с аппарели — незачет попытки выполнения задания. 3. Задание считается выполненным после того, как ровер целиком окажется внутри взлетного модуля, или закончилось время выполнения задания.

Оценочные ведомости

Сборка полезной нагрузки	**База**	**Количество** 1,00	**Итоговый балл**
Чертежи полезной нагрузки			
Присланы и корректны в соответствии с заданием	10	1,00	10
Коррекция чертежей со штрафом	−5	1,00	−5
Сборка полезной нагрузки			0
ПН собрана	10	1,00	10
Коррекция сборки	−5	1,00	−5
Взаимодействие с техником-аватаром			0
Без штрафов	10	1,00	10
Со штрафами	−5	0,00	0
Итого	30		20

Предварительные испытания	**База**	**2-й день испытаний** 1,00	**3-й день испытаний** 1	**4-й день испытаний**	**Итоговый балл**
Результаты работы второго дня					
Задание 1.2	1	0,00			
Задание 1.5	2	0,00			
Видео 1.1	1	0,00			
Видео 1.2	1	0,00			
Задание 2.2	1	0,00			
Задание 2.3	1	0,00			
Видео 2.2	1	0,00			
Видео 2.3	1	0,00			
Сдача работы за день	1	0,00			

Предварительные испытания	**База**	**2-й день испытаний**	**3-й день испытаний**	**4-й день испытаний**	**Итоговый балл**
		1,00	1		
Штрафы		0			0
Результаты работы третьего дня					0
Задание 3.3	1		0		
Задание 3.4	1		0		
Видео 3.3	1		0		
Видео 3.4	1		0		
Задание 4.1	1		0		
Сдача работы за день	1		0		
Штрафы			0		0
Результаты работы четвертого дня					0
Сдача результатов работы за день	1			0	1
Штрафы					0
Итого	17,00	0,00	0,00	0,00	0,00

Квалификация	**База**	**Номер попытки**		**Итоговый балл**
		1	2	
Квалификация				
Часть 1				
Робот доехал на старт в режиме радио	3			
Робот начал искать точку разведки после нажатия на кнопку	5			
После обнаружения ровер должен приехать в точку разведки и остановиться как можно ближе от ее центра	10			
Точность остановки над точкой				
Робот вернулся на старт в режиме радио	3			
Часть 2				
Робот доехал на старт в режиме радио	3			
Робот доехал до точки поиска	5			
Фотография точки поиска залита в дискорд-канал команды	5			
Робот вернулся на старт в режиме радио	3			
Часть 3				
Робот доехал на старт в режиме радио	3			
Робот приподнял предмет для перемещения	5			
Робот перенес предмет для перемещения	5			
Робот опустил предмет для перемещения	5			
Робот вернулся на старт в режиме радио	3			
Часть 4				
Робот доехал на старт в режиме радио	**3**			
Команда передает на радиомодуль ВМ указанную организаторами последовательность символов	**5**			
Команда в режиме управления по радиоканалу доводит ровер до въезда на аппарель	**3**			
Робот вернулся на старт в режиме радио	**3**			
Итого	69			
				0

Испытания на полигоне	**База**	**Номер попытки**		**Итоговый балл**
		1	2	
В зоне разведки				
Робот доехал на старт в режиме радио	5			
Робот начал движение	10			
Робот доехал до первой точки	20			
Длина радиус-вектора проекции на центр				
Робот доехал до второй точки	20			
Длина радиус-вектора проекции на центр				
Робот доехал до третьей точки	20			
Длина радиус-вектора проекции на центр				
Робот закончил движение	5			
В зоне поиска				
Робот доехал на старт в режиме радио	5			
Робот начал движение	5			
Фотография точки поиска загружена в облачное хранилище команды и в дискорд-канал команды	10			
Время выполнения задания				
В зоне перемещения				
Робот доехал на старт в режиме радио	5			
Робот начал движение	5			
Робот приподнял предмет для перемещения	10			
Робот опустил предмет для перемещения	10			
Предмет для перемещения находится в границах точки для расположения предметов	40			
Время выполнения задания				
В зоне ВМ				
Робот доехал на старт в режиме радио	5			
Робот начал движение	5			
ВМ открыл аппарель по команде ровера	10			
Ровер коснулся аппарели	5			
Ровер заехал внутрь ВМ	20			
Ровер коснулся ВМ	−2			
Ровер упал с аппарели	−5			
Итого	455			
				0

Аэрокосмические системы

Заключительный этап

Инженерный тур

Решение задачи

Код для управления полезной нагрузкой

```
#include <Servo.h>
#include <ros.h>
#include <std_msgs/UInt16.h>
#include <std_msgs/Empty.h>
#include <std_msgs/UInt8MultiArray.h>

class NewHardware : public ArduinoHardware
{
  public:
  NewHardware():ArduinoHardware(&Serial1, 115200){};
};

ros::NodeHandle_<NewHardware>  nh;

class ServoCnt {
public:
  ros::Subscriber<std_msgs::UInt16, ServoCnt> sub;

  ServoCnt(int pin, int min_limit, int max_limit, char* topic_name):
    sub(topic_name, &ServoCnt::callback, this)
  {
```

```
    this->pin = pin;
    this->min_limit = min_limit;
    this->max_limit = max_limit;
  }

  void setup() {
    this->servo.attach(pin);
  }

  void callback(const std_msgs::UInt16& cmd_msg) {
    int theta = constrain(cmd_msg.data, this->min_limit, this->max_limit);
    this->servo.write(theta);
  }

private:
  int pin,
    min_limit,
    max_limit;
  Servo servo;
};

class HC12 {
public:
  ros::Subscriber<std_msgs::Empty, HC12> sub;
  HardwareSerial* _serial;

  HC12(HardwareSerial* serial, const int& set_pin):
    sub("/hc12/data", &HC12::callback, this)
  {
    this->_serial = serial;
    this->_set_pin = set_pin;
  }

  void setup() {
    pinMode(this->_set_pin, OUTPUT);
    digitalWrite(this->_set_pin, HIGH);

    this->_serial->begin(9600);
  }

  void callback(const std_msgs::Empty& msg) {
    const size_t packet_length = 6;
    uint8_t packet[packet_length] = {0xAA, 0xFA, 0x03, 0x14, 0x01, 0x03};
    this->_serial->write(packet, packet_length);
  }

  String send_at_command(String cmd, bool waiting) {
    String _resp = "";

    digitalWrite(this->_set_pin, LOW);
    delay(100);
    this->_serial->println(cmd);
    if(waiting) {
```

```
75        _resp = wait_response();
76      }
77      digitalWrite(this->_set_pin, HIGH);
78      delay(100);
79
80      return _resp;
81    }
82
83    String wait_response() {
84      String _resp = "";
85      long _timeout = millis() + 2000;
86
87      while(!this->_serial->available() && millis() < _timeout)
88        ;
89
90      if(this->_serial->available()) {
91        _resp = this->_serial->readString();
92      }else {
93        _resp = "ERROR";
94      }
95
96      return _resp;
97    }
98
99  private:
100   int _set_pin;
101 };
102
103 ServoCnt camera_servo_yaw(7, 45, 135, "camera_servo_yaw"),
104         camera_servo_pitch(8, 0, 90, "camera_servo_pitch"),
105         object_servo_yaw(45, 65, 115, "object_servo_yaw"),
106         object_servo_pitch(44, 0, 90, "object_servo_pitch");
107
108 String _response = "";
109 HC12 hc12_radio(&Serial2, 40);
110
111
112 std_msgs::Empty button_msg;
113 ros::Publisher button_pub("button", &button_msg);
114
115 const int button_pin = 11;
116
117 void setup() {
118   pinMode(button_pin, INPUT_PULLUP);
119   hc12_radio.setup();
120
121   camera_servo_yaw.setup();
122   camera_servo_pitch.setup();
123   object_servo_yaw.setup();
124   object_servo_pitch.setup();
125
126   nh.initNode();
127   nh.advertise(button_pub);
128
129   // Check HC12
130   do {
131     _response = hc12_radio.send_at_command("AT", true);
132   } while(_response.indexOf("OK") == -1);
```

```
133
134   // Set up HC12
135   hc12_radio.send_at_command("AT+A000", true);
136   hc12_radio.send_at_command("AT+B9600", true);
137   hc12_radio.send_at_command("AT+C001", true);
138   hc12_radio.send_at_command("AT+FU1", true);
139   hc12_radio.send_at_command("AT+P8", true);
140
141   nh.subscribe(hc12_radio.sub);
142
143   nh.subscribe(camera_servo_yaw.sub);
144   nh.subscribe(camera_servo_pitch.sub);
145   nh.subscribe(object_servo_yaw.sub);
146   nh.subscribe(object_servo_pitch.sub);
147 }
148
149 void loop() {
150   if(digitalRead(button_pin) == LOW) {
151     button_pub.publish(&button_msg);
152   }
153
154   nh.spinOnce();
155   delay(5);
156 }
```

Код протокола передачи команд через радиоканал

```
1  import serial
2  from struct import pack, unpack
3  from collections import namedtuple
4  from enum import IntEnum
5
6  import time
7
8  MOVEMENT_RF_START_CODE = b'm'
9  MOVEMENT_PN_RF_START_CODE = b'p'
10 FRAME_START_CODE = b'i'
11 FRAME_PART_START_CODE = b'j'
12
13 SV610_config = namedtuple('SV610_config',
14     'channel freq rf_data power rf_date data_bit stop_bit no_patiry net_id
       ↪  node_id'
15 )
16
17 class Freq(IntEnum):
18     MHz_433 = 1
19     MHz_470 = 2
20     MHz_868 = 3
21     MHz_915 = 4
22
23 class Rate(IntEnum):
24     BPS_1200    = 0
25     BPS_2400    = 1
26     BPS_4800    = 2
27     BPS_9600    = 3
28     BPS_11400   = 4
29     BPS_19200   = 5
```

```
30      BPS_38400   = 6
31      BPS_57600   = 7
32      BPS_115200  = 8
33
34  class Power(IntEnum):
35      dBm_1   = 0
36      dBm_2   = 1
37      dBm_5   = 2
38      dBm_8   = 3
39      dBm_11  = 4
40      dBm_14  = 5
41      dBm_17  = 6
42      dBm_20  = 7
43
44  class SV610:
45      def __init__(self, device_name, speed):
46          self.device_name = device_name
47          self.ser = serial.Serial(self.device_name,
48              speed,
49              timeout=4)
50
51          # Go to working mode
52          self.set_mode(False)
53
54      def print_mode(self):
55          print(f'SV610 mode:{"WORKING MODE" if not self.ser.dtr else "SETTING
            ↪  MODE"}')
56
57      def set_mode(self, level):
58          self.ser.dtr = level
59          time.sleep(2)
60
61          self.print_mode()
62
63      def read(self):
64          if self.ser.in_waiting > 0:
65              return self.ser.readline()
66          return None
67
68      def read_until(self, seq):
69          if self.ser.in_waiting > 0:
70              return self.ser.read_until(seq)
71          return None
72
73      def read_all(self):
74          if self.ser.in_waiting <= 0:
75              return None
76
77          body = b''
78          while self.ser.in_waiting > 0:
79              body += self.ser.readline()
80          return body
81
82      def read_block(self):
83          while True:
84              output = self.read()
85              if output is not None:
86                  return output
87
88              time.sleep(0.1)
```

```
89
90      def current_settings(self):
91          if not self.ser.dtr:
92              print("current_settings(): first go to SETTING MODE")
93              return
94
95          self.ser.write(b'\xAA\xFA\x01')
96          config_bytes = bytes(self.read_block())[0:14]
97
98          config = SV610_config._make(unpack('!b b b b b b b i h', config_bytes))
99          return config
100
101     def update_settings(self, config):
102         if not self.ser.dtr:
103             print("update_settings(): first go to SETTING MODE")
104             return
105
106         config_raw = list(config)
107         config_bytes = pack('!b b b b b b b i h', *config_raw)
108         self.ser.write(b'\xAA\xFA\x03' + config_bytes)
109         self.read_block()
110
111     def write(self, buffer):
112         if self.ser.dtr:
113             print("update_settings(): first go to WORKING MODE")
114             return -1
115
116         return self.ser.write(buffer)
```

Код для управления ровером при получении данных через радиоканал

Данный код содержит только методы управления и не является основным управляющим алгоритмом.

```
1  import rospy
2
3  from std_srvs.srv import Empty
4  from geometry_msgs.msg import Pose2D
5  from geometry_msgs.msg import Twist
6  from std_msgs.msg import UInt16
7  from std_msgs.msg import Empty as EmptyMSG
8
9  from angles import normalize_angle
10
11 from math import copysign, sqrt
12
13 from struct import pack, unpack
14
15 class RoverHandler:
16     MOVEMENT_CODE = b'm'
17     LINEAR_VEL = 0.2
18     ANGULAR_VEL = 0.1
19     ANGULAR_TOL = 0.05
20
21     HC12_DATA = [0xAA, 0xFA, 0x03, 0x14, 0x01, 0x03]
22
23     def __init__(self):
```

```
24          rospy.wait_for_service('/reset')
25          self.reset = rospy.ServiceProxy('/reset', Empty)
26
27          self.hc12_pub = rospy.Publisher('/hc12/data', EmptyMSG, queue_size=1)
28
29          self.cmd_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
30
31          self.cam_yaw_pub = rospy.Publisher('/camera_servo_yaw', UInt16,
            ↪  queue_size=1)
32          self.cam_pitch_pub = rospy.Publisher('/camera_servo_pitch', UInt16,
            ↪  queue_size=1)
33          self.obj_yaw_pub = rospy.Publisher('/object_servo_yaw', UInt16,
            ↪  queue_size=1)
34          self.obj_pitch_pub = rospy.Publisher('/object_servo_pitch', UInt16,
            ↪  queue_size=1)
35
36          self.msg_movement_format = '!c2e5B'
37          self.msg_movement_size = (1 + 2 * 2 + 5 * 1)
38
39          self.set_distance = 0
40          self.set_theta = 0
41          self.is_moving = False
42
43          self.prev_cam_yaw = None; self.prev_cam_pitch = None
44          self.prev_obj_yaw = None; self.prev_obj_pitch = None
45
46          self.reset()
47
48          self.odom_sub = rospy.Subscriber('/odom_pose2d', Pose2D,
            ↪  self.odom_callback)
49
50      def odom_callback(self, msg):
51          if not self.is_moving:
52              return
53
54          distance = sqrt(msg.x ** 2 + msg.y ** 2)
55          theta = msg.theta
56
57          cmd_msg = Twist()
58
59          if self.set_distance != 0:
60              if (distance - abs(self.set_distance)) >= 0:
61                  print('Destination reached...Ok')
62                  self.reset()
63                  self.is_moving = False
64              else:
65                  cmd_msg.linear.x = copysign(self.LINEAR_VEL, self.set_distance)
66
67              self.cmd_pub.publish(cmd_msg)
68
69          if self.set_theta != 0:
70              theta_d = normalize_angle(self.set_theta - theta)
71              if abs(theta_d) <= self.ANGULAR_TOL:
72                  print('Orientation set...Ok')
73                  self.reset()
74                  self.is_moving = False
75              else:
76                  cmd_msg.angular.z = copysign(self.ANGULAR_VEL, theta_d)
77
78              self.cmd_pub.publish(cmd_msg)
```

```
79
80      def send_hc12(self):
81          print(f'Send hc12 code...')
82          self.hc12_pub.publish(EmptyMSG())
83
84      """
85      Message rover movement:
86
87      start_code (char) | movement_distance (1/2 float) | movement_theta (1/2 float)
    ↪   |
88          camera_yaw (uint8) | camera_pitch (uint8) | object_yaw (uint8) |
    ↪   object_pitch (uint8) | \r\n
89      """
90
91      def decode_msg(self, msg):
92          if len(msg) != self.msg_movement_size:
93              print(f'RoverHandler: msg size should be {self.msg_movement_size},
                ↪  actual is {len(msg)}')
94              print(msg)
95              return
96
97          try:
98              decode = unpack(self.msg_movement_format, msg)
99          except Exception as e:
100             print(e)
101
102         start_code, distance, theta, \
103             cam_yaw, cam_pitch, obj_yaw, obj_pitch, need_send_hc12 = decode
104
105         if start_code != self.MOVEMENT_CODE:
106             print(f'RoverHandler: start code should be {self.MOVEMENT_CODE},
                ↪  actual is {start_code}')
107             return
108
109         if distance != 0 and theta != 0:
110             print(f'RoverHandler: distance or theta should be zero, actual is
                ↪  {distance}, {theta}')
111         elif (not self.is_moving) and (distance != 0 or theta != 0):
112             if distance != 0:
113                 print(f'Go to point with distance {distance}')
114             else:
115                 print(f'Setting orientation to {theta}')
116
117             self.set_distance = distance
118             self.set_theta = theta
119             self.is_moving = True
120         print(cam_yaw, cam_pitch, obj_yaw, obj_pitch)
121
122         if self.prev_cam_yaw != cam_yaw:
123             self.cam_yaw_pub.publish(cam_yaw)
124         self.prev_cam_yaw = cam_yaw
125
126         if self.prev_cam_pitch != cam_pitch:
127             self.cam_pitch_pub.publish(cam_pitch)
128         self.prev_cam_pitch = cam_pitch
129
130         if self.prev_obj_yaw != obj_yaw:
131             self.obj_yaw_pub.publish(obj_yaw)
132         self.prev_obj_yaw = obj_yaw
133
```

```
134         if self.prev_obj_pitch != obj_pitch:
135             self.obj_pitch_pub.publish(obj_pitch)
136         self.prev_obj_pitch = obj_pitch
137 
138         if bool(need_send_hc12):
139             self.send_hc12()
```

Сервис управления ровером через радиоканал

```
1  #!/usr/bin/env python3
2  from sv610_helper import *
3  from rover import *
4  from frame import *
5  import threading
6  import time
7 
8  event_rover = threading.Event()
9  event_image = threading.Event()
10 
11 def image_loop(sv610):
12     image_sender = ImageSender()
13     image_handler = ImageHandler(image_sender.current_frame)
14 
15     while not rospy.is_shutdown():
16         event_image.wait()
17         event_image.clear()
18         while True:
19             msg = image_sender.send()
20 
21             if msg is not None:
22                 #print(msg)
23                 sv610.write(msg)
24                 break
25         time.sleep(1)
26         event_rover.set()
27 
28 def rover_loop(sv610):
29     rover = RoverHandler()
30     while not rospy.is_shutdown():
31         event_rover.wait()
32         event_rover.clear()
33         data = sv610.ser.read_until(b'\r\n')
34 
35         if data != b'':
36             rover.decode_msg(data[:-2])
37         time.sleep(0.5)
38         event_image.set()
39 
40 if __name__ == '__main__':
41     rospy.init_node('sv610_listener', anonymous=True)
42 
43     sv610 = SV610('/dev/ttyUSB0', 19200)
44 
45     event_rover.set()
46 
47     image_thread = threading.Thread(name='image_daemon', \
48                 target=image_loop,
49                 args=(sv610,))
50     image_thread.setDaemon(True)
```

```
51      image_thread.start()
52      rover_thread = threading.Thread(name='rover_daemon', \
53                  target=rover_loop,
54                  args=(sv610,))
55      rover_thread.setDaemon(True)
56      rover_thread.start()
57      try:
58          rospy.spin()
59      except rospy.ROSInterruptException:
60          pass
```

Основной алгоритм управления ровером через WEB-интерфейс

В качестве задатчика команд используется обычная web-страница и данный сервис принимает команды со страницы и передает их на ровер, используя все модули указанные выше.

```
1   #!/usr/bin/env python3
2   from sv610_helper import *
3   import termios
4   import tty
5   import sys
6   from select import select
7   import threading
8   from collections import defaultdict
9   import base64
10  import time
11  from http.server import HTTPServer, BaseHTTPRequestHandler
12
13  event_image = threading.Event()
14  event_key = threading.Event()
15
16  jpeg_lock = threading.Lock()
17  jpeg_data = None
18
19  class S(BaseHTTPRequestHandler):
20      def do_GET(self):
21          global jpeg_data, jpeg_lock
22
23          self.send_response(200)
24          self.send_header('Content-type','multipart/x-mixed-replace;
            ↪  boundary=--jpgboundary')
25          self.end_headers()
26
27          while True:
28              with jpeg_lock:
29                  if jpeg_data is None:
30                      continue
31
32                  self.wfile.write(b"--jpgboundary\r\n")
33                  self.send_header('Content-type','image/jpeg')
34                  self.send_header('Content-length',str(len(jpeg_data)))
35                  self.end_headers()
36                  self.wfile.write(bytearray(jpeg_data))
37                  self.wfile.write(b'\r\n')
38
39              time.sleep(0.1)
40
41  def start_server(port=8000):
```

```
42      httpd = HTTPServer(('', port), S)
43      httpd.serve_forever()
44
45  def save_settings():
46      return termios.tcgetattr(sys.stdin)
47
48  def restore_settings(old_settings):
49      termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_settings)
50
51      current_frame_size = None
52      acutal_frame_size = 0
53      all_frame = b''
54      start_time = time.time()
55      frame_count = 0
56  def get_key(settings, timeout):
57      tty.setraw(sys.stdin.fileno())
58
59      rlist, _, _ = select([sys.stdin], [], [], timeout)
60      if rlist:
61          key = sys.stdin.read(1)
62      else:
63          key = ''
64      termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
65      return key
66
67  processed_frame_number = None
68  previous_part_number = None
69  total_frame = b''
70  start_time = 0
71  def decode_image(data):
72      global jpeg_lock, jpeg_data, processed_frame_number, previous_part_number,
        ↪  total_frame, start_time
73
74      if data == b'':
75          return False
76
77      IMAGE_CODE = b'i'
78      if chr(data[0]) == IMAGE_CODE.decode("ascii"):
79          if len(data) < 7:
80              print(f'Error: Image msg size should be more then 7 bytes, actual is
                ↪  {len(data)} bytes')
81              return False
82
83          _, frame_number, part_number, total_parts, packet_size = \
84                  unpack('!cH2BH', data[0:7])
85
86          if packet_size != len(data[7:]):
87              print(f'Packet size should be {7 + packet_size}, actual is
                ↪  {len(data)}')
88              return False
89          if (processed_frame_number is None):
90              if part_number == 1:
91                  processed_frame_number = frame_number
92                  previous_part_number = 0
93                  total_frame = b''
94                  start_time = time.time()
95                  print(f'Start receiving a new frame')
96              else:
97                  print(f'Error: Frame transfer begin with part {part_number}')
98                  return False
```

```
99          if (processed_frame_number is not None):
100             if processed_frame_number != frame_number:
101                 print(f'Error: Part of the image have been skipped (previous is
                    ↪ {processed_frame_number}, actual is {frame_number})')
102
103                 processed_frame_number = None
104                 return False
105             if (previous_part_number + 1) != part_number:
106                 print(f'Error: Package was missed, part with number
                    ↪ {previous_part_number + 1} was expected, actual is
                    ↪ {part_number}')
107
108                 processed_frame_number = None
109                 return False
110             total_frame += data[7:]
111             if part_number == total_parts:
112                 with jpeg_lock:
113                     jpeg_data = total_frame
114                 print(f'The frame was completely received; FPS {1 / (time.time() -
                    ↪ start_time)}')
115                 processed_frame_number = None
116             previous_part_number = part_number
117         return True
118     return False
119
120 def image_loop(sv610):
121     global jpeg_data
122
123     while True:
124         event_image.wait()
125         event_image.clear()
126         data = sv610.ser.read_until(b'\r\n')
127         if data != b'':
128             #print(data)
129             status = decode_image(bytearray(data[:-2]))
130         time.sleep(1)
131         event_key.set()
132
133 class Servo:
134     SERVO_STEP = 10
135
136     def __init__(self, lim_min, lim_max,\
137             dec_key, inc_key):
138         self.limit = (lim_min, lim_max)
139         self.keys = (dec_key, inc_key)
140         self.actual = sum(self.limit) // 2
141
142     def constrain(self, x):
143         return min(self.limit[1], max(self.limit[0], x))
144
145     def handler(self, key):
146         if key not in self.keys:
147             return self.actual
148
149         self.actual = self.constrain(self.actual + \
150                 self.SERVO_STEP * (1 if self.keys.index(key) == 1 else -1))
151         return self.actual
152
153 key_lock = threading.Lock()
154 key_global = ''
```

```
155
156 def key_loop(settings, sv610):
157     global key_global, key_lock
158
159     move_bind_dict = {
160         'w': (0.1, 0),
161         's': (-0.1, 0),
162         'a': (0, 0.18),
163         'd': (0, -0.18)
164     }
165     move_bind = defaultdict(lambda: (0, 0), move_bind_dict)
166
167     cam_yaw = Servo(45, 135, 'l', 'j')
168     cam_pitch = Servo(0, 90, 'k', 'i')
169     obj_yaw = Servo(45, 135, '6', '4')
170     obj_pitch = Servo(0, 90, '2', '8')
171
172     def encode_msg(distance, theta,\
173             cam_yaw, cam_pitch, obj_yaw, obj_pitch, send_hc12):
174         MOVEMENT_CODE = b'm'
175         msg_movement_format = '!c2e5B'
176         msg = pack(msg_movement_format, MOVEMENT_CODE,\
177             distance, theta, cam_yaw, cam_pitch,\
178             obj_yaw, obj_pitch, send_hc12)
179         return msg
180
181     while True:
182         event_key.wait()
183         event_key.clear()
184         with key_lock:
185             hc12 = 0
186             if key_global == 'f':
187                 hc12 = 1
188             msg = encode_msg(
189                 move_bind[key_global][0],
190                 move_bind[key_global][1],
191                 cam_yaw.handler(key_global),
192                 cam_pitch.handler(key_global),
193                 obj_yaw.handler(key_global),
194                 obj_pitch.handler(key_global),
195                 hc12
196             ) + b'\r\n'
197             print(f'Send movement cmd ({msg}) with size ({sv610.write(msg)})')
198             key_global = ''
199         time.sleep(0.5)
200         event_image.set()
201
202 if __name__ == '__main__':
203     sv610 = SV610('/dev/ttyUSB0', 19200)
204     settings = save_settings()
205     event_key.set()
206     key_thread = threading.Thread(name='key_daemon',\
207                     target=key_loop, \
208                     args=(settings, sv610,))
209     key_thread.setDaemon(True)
210     key_thread.start()
211     image_thread = threading.Thread(name='image_daemon',\
212                     target=image_loop, \
213                     args=(sv610,))
214     image_thread.setDaemon(True)
```

```
    image_thread.start()

    web_thread = threading.Thread(name='web_daemon', \
                    target=start_server)
    web_thread.setDaemon(True)
    web_thread.start()

    try:
        while True:
            key = get_key(settings, 0.5)
            if key != '':
                with key_lock:
                    key_global = key
                if key == 'z':
                    break

    except KeyboardInterrupt as e:
        pass
    finally:
        restore_settings(settings)
```

Код управления ровером в режиме автономного поиска шариков на полигоне при помощи алгоритмов компьютерного зрения

```
#!/usr/bin/env python3

import rospy

from std_srvs.srv import Empty

from std_msgs.msg import UInt16
from std_msgs.msg import Empty as EmptyMSG
from geometry_msgs.msg import Twist
from geometry_msgs.msg import Pose2D
from angles import normalize_angle

from cv_detect import *

from math import copysign

class Rover_Handler:
    ROTATION_VEL = 0.06
    FORWARD_VEL = 0.04

    HORIZ_ALIGNMENT_THR = 10
    ANGULAR_TOLERANCE = 0.08

    HORIZ_CAMERA_FIELD = 60

    HEIGHT_BALL_FRACTION = 0.28

    def __init__(self):
        rospy.wait_for_service('/reset')
        self.reset = rospy.ServiceProxy('/reset', Empty)

        self.cmd_pub = rospy.Publisher('/cmd_vel',
                Twist,
```

```
34                 queue_size=1)
35
36         self.prev_pitch = 0
37         self.cam_yaw_pub = rospy.Publisher(
38                 '/camera_servo_yaw',
39                 UInt16,
40                 queue_size=1)
41         self.cam_pitch_pub = rospy.Publisher(
42                 '/camera_servo_pitch',
43                 UInt16,
44                 queue_size=1)
45
46         self.pose = None
47         self.theta = None
48         self.odom_sub = rospy.Subscriber(
49                 "/odom_pose2d",
50                 Pose2D,
51                 self.odom_callback)
52
53     def odom_callback(self, msg):
54         self.pose = (msg.x, msg.y)
55         self.theta = msg.theta
56
57     def odom_reset(self):
58         self.reset()
59         self.pose = (0, 0)
60         self.theta = 0
61
62     def wait_button(self):
63         rospy.wait_for_message("/button", EmptyMSG)
64
65     def set_camera_yaw(self, yaw):
66         msg = UInt16()
67
68         msg.data = int(yaw)
69         print(msg.data)
70         self.cam_yaw_pub.publish(msg)
71
72     def set_camera_pitch(self, pitch):
73         msg = UInt16()
74
75         self.prev_pitch = int(pitch)
76
77         msg.data = int(pitch)
78         print(msg.data)
79         self.cam_pitch_pub.publish(msg)
80
81     def stop(self):
82         self.cmd_pub.publish(Twist())
83
84     def horiz_alignment(self, center, x_setpoint):
85         # CHANGE SIGN IF ROVER ROTATE IN WRONG DIRECTION
86         error = center - x_setpoint
87         print(error)
88         if abs(error) <= self.HORIZ_ALIGNMENT_THR:
89             return True
90
91         self.rotate(error)
92         return False
93
```

```
94      def vert_alignment(self, frame_size, y_ball, height_ball):
95          width, height = frame_size
96
97          if (height_ball / height) >= self.HEIGHT_BALL_FRACTION:
98              return True
99
100         vertical_error = (height // 2) - y_ball
101         pitch_signal = self.prev_pitch + (self.HORIZ_CAMERA_FIELD / width) *
            ↪  vertical_error
102
103         if pitch_signal >= 0 and pitch_signal <= 25:
104             self.set_camera_pitch(pitch_signal)
105         self.moving(1)
106         rospy.sleep(0.4)
107         return False
108
109     def angular_alignment(self, theta_setpoint):
110         # CHANGE SIGN IF ROVER ROTATE IN WRONG DIRECTION
111         theta_setpoint = theta_setpoint * (3.1415 / 360.0)
112         theta_d = normalize_angle(-theta_setpoint + self.theta)
113
114         if abs(theta_d) <= self.ANGULAR_TOLERANCE:
115             return True
116
117         self.rotate(theta_d)
118         return False
119
120     def rotate(self, sign):
121         cmd_msg = Twist()
122
123         cmd_msg.angular.z = copysign(self.ROTATION_VEL, sign)
124         self.cmd_pub.publish(cmd_msg)
125
126     def moving(self, sign):
127         cmd_msg = Twist()
128
129         cmd_msg.linear.x = copysign(self.FORWARD_VEL, sign)
130         self.cmd_pub.publish(cmd_msg)
131
132 BALL_COUNT = 4
133 USE_BUTTON = True
134
135 # 1 or -1
136 ROTATION_LEFT = 1
137 ROTATION_RIGHT = (-1 ^ 1 ^ ROTATION_LEFT)
138
139 def main():
140     rospy.init_node('cv_task_first')
141
142     cv_handler = CV_Handler(debug_image=True)
143     rover_handler = Rover_Handler()
144
145     try:
146         rate = rospy.Rate(10)
147
148         rover_handler.wait_button()
149
150         for i in range(5):
151             rover_handler.set_camera_yaw(90)
152             rover_handler.set_camera_pitch(8)
```

```
153
154             rate.sleep()
155
156         print(f'Setup rover was complete...Done!')
157
158         ball_index = 0
159         while ball_index < BALL_COUNT and (not rospy.is_shutdown()):
160             print(f'Produce ball with index {ball_index}')
161
162             if USE_BUTTON:
163                 print(f'Waiting button for ball...')
164                 rover_handler.wait_button()
165                 print(f'Done!')
166
167             # If ball is not first
168             if ball_index > 0:
169                 # Go backward
170                 print(f'Go backward from the previous ball...')
171                 for i in range(20):
172                     rover_handler.moving(-1)
173
174                     rate.sleep()
175                 rover_handler.stop()
176                 print(f'Done!')
177
178                 rospy.sleep(1)
179
180                 # Rotate on camera horizontal field view
181                 print(f'Rotate rover on camera field view, actual is
                    ↪ {rover_handler.HORIZ_CAMERA_FIELD}...')
182                 rover_handler.odom_reset()
183                 rospy.sleep(0.5)
184                 while not rover_handler.angular_alignment(30 * ROTATION_LEFT) and
                    ↪ (not rospy.is_shutdown()):
185                     rate.sleep()
186                 rover_handler.stop()
187                 print(f'Done!')
188
189                 rospy.sleep(1)
190
191             # Reset previous information about ball
192             cv_handler.ball_rect = None
193             cv_handler.ball_area = None
194
195             # Rotate while don`t find red ball
196             print(f'Rotating while don`t meet ball...')
197             while cv_handler.ball_rect is None and (not rospy.is_shutdown()):
198                 rover_handler.rotate(ROTATION_LEFT)
199                 rate.sleep()
200             rover_handler.stop()
201             print(f'Done!')
202
203             rospy.sleep(1)
204
205             # Centering red ball
206             print(f'Centering to ball...')
207             while not rover_handler.horiz_alignment(cv_handler.FRAME_SIZE[0] // 2,
208                     cv_handler.ball_rect[0] + cv_handler.ball_rect[2] // 2) and
                        ↪ (not rospy.is_shutdown()):
209                 rate.sleep()
```

```
            rover_handler.stop()
            print(f'Done!')

            rospy.sleep(1)

            # Moving forward to ball
            print(f'Moving forward to ball...')
            while not rover_handler.vert_alignment(cv_handler.FRAME_SIZE,
                    cv_handler.ball_rect[1],
                    cv_handler.ball_rect[3]) and (not rospy.is_shutdown()):

                rover_handler.horiz_alignment(
                        cv_handler.FRAME_SIZE[0] // 2,
                        cv_handler.ball_rect[0] + cv_handler.ball_rect[2] // 2)

                rate.sleep()
            rover_handler.stop()
            print(f'Done!')

            ball_index += 1

        rover_handler.stop()
        print(f'Task was complete)')
    except rospy.ROSInterruptException:
        rover_handler.stop()
        pass
    finally:
        rover_handler.stop()

if __name__ == '__main__':
    main()
```

Материалы для подготовки

Самый полный перечень материалов для подготовки находится в разделе «Материалы подготовки» страницы профиля на сайте НТО: https://ntcontest.ru/tracks/nto-school/kosmicheskiy-proekt/aerokosmicheskie-sistemy/.

Данный перечень постоянно обновляется и актуализируется.