Методические рекомендации по проведению хакатонов по аэрокосмической робототехники

Целевая аудитория: школьники 7–11 классов
Количество команд: 3–4 команды
Состав команды: 3 человека (Конструктор, Python-разработчик, C-разработчик)
Продолжительность: 3–4 часа
Период проведения хакатона: июль 2026 года.

Цель проведения: познакомить школьников с аэрокосмической робототехникой на практике и подготовить их к участию в профильных чемпионатах. Участники попробуют работать с ROS, программированием на Python и C, а также с 3D-моделированием в Компас-3D. За несколько часов команда должна создать работающий прототип системы управления захватом – от модели до кода.

Легенда хакатона:

События разворачиваются на марсианской исследовательской базе. Центр управления полетами потерял связь с манипулятором марсохода «Искатель-2». Бортовой компьютер на базе Raspberry Pi с установленной операционной системой ROS работает исправно, но драйвер сервопривода на контроллере Arduino Due вышел из строя, а сам механический захват требует замены. Участникам предстоит за три часа разработать временный мост для восстановления управления. Программист Python создает на Raspberry Pi ROS-ноду, которая принимает команды с пульта оператора и отправляет углы поворота через UART-соединение используя microros-agent. Программист C пишет ПО для Arduino Due, которое принимает эти данные и управляет сервоприводом. Конструктор проектирует в системе Компас-3D новую модель захвата.

Оборудование на одну команду:

- Raspberry Pi 4 с предустановленным образом Ubuntu 20.04 и ROS 2(1 шт.)
- SD-карта 16 ГБ с образом системы (1 шт.)
- Arduino Due (1 шт.)
- Преобразователь USB-UART (CP2102 или PL2303) (1 шт.)
- Серводвигатель MG90 или аналог (1 шт.)
- Провода DuPont (мама-мама, папа-папа) (10 шт.)
- Компьютер с установленным Компас-3D (1 шт. на конструктора)

Распределение ролей и задачи участников:

Конструктор:

- Спроектировать в Компас-3D модель захвата (тип «параллельные губки» или «клешня»), которая крепится к валу сервопривода.
- Предусмотреть в модели посадочное место под сервопривод и отверстия для крепления.
- Экспортировать модель в формат STL для последующей 3D-печати.
- Убедиться в виртуальной среде, что при вращении вала сервопривода челюсти захвата свободно двигаются без заклинивания.

Программист Python:

- Разработать ROS-ноду, которая подписывается на топик /gripper_command с типом сообщения std_msgs/Int16.
- При получении угла от 0 до 180 градусов отправить это значение в топик /due_command с типом сообщения std_msgs/Int16.
- Обеспечить автоматическое создание топика при запуске ноды.

Программист C:

- Создать прошивку для Arduino Due, которая подписывается на топик /due_command с типом сообщения std_msgs/Int16.
- Управлять сервоприводом с помощью стандартной библиотеки Servo.
- Ограничить угол диапазоном от 0 до 180 градусов.
- Загрузить прошивку в Arduino Due.

Ожидаемые результаты:

По окончании хакатона каждая команда предоставляет:

- 3D-модель захвата, созданную в Компас-3D (файл .m3d и экспортированный .step).
- Исходный код Python-ноды и C-прошивки.
- Рабочую связку Raspberry Pi и Arduino Due, управляющую сервоприводом.

Материалы для проведения:

- Пример python ноды для Raspberry Pi;
- Пример C++ ноды для Arduino Due;
- Образ для Raspberry Pi;
- Скомпилированный пакет библиотек microROSLib для ArduinoIDE.

При отсутствии необходимого оборудования хакатон может быть изменен:

Легенда хакатона:

События разворачиваются на лунной исследовательской базе. Во время выполнения научной миссии луноход «Геолог-3» потерял возможность самостоятельного передвижения из-за отказа ходовой системы. Бортовой компьютер продолжает работать, связь с аппаратом сохраняется, а собранные научные данные имеют высокую ценность для экспедиции. Для эвакуации неисправного аппарата планируется использовать луноход-спасатель, однако система буксировки для него еще не разработана. Участникам предстоит за три часа разработать концепцию робототехнической системы спасения аварийного лунохода. Конструктор проектирует механизм сцепки и буксировки в системе Компас-3D. Программист Python разрабатывает архитектуру программного обеспечения верхнего уровня на базе ROS2, определяет взаимодействие программных узлов и алгоритм выполнения спасательной операции. Программист C проектирует программное обеспечение контроллера исполнительного механизма, определяет интерфейсы обмена данными и логику управления системой сцепки.

Распределение ролей и задачи участников:

Конструктор:

- Спроектировать в Компас-3D механизм сцепки для буксировки аварийного лунохода.
- Предусмотреть в модели узел захвата буксируемого аппарата, крепление механизма к раме лунохода-спасателя, элементы фиксации буксируемого лунохода, возможность расцепки после завершения буксировки.
- Подготовить не менее двух изображений конструкции с различных ракурсов.
- С помощью встроенных инструментов Компас-3D убедиться, что механизм может выполнять сцепку и расцепку без взаимных пересечений деталей.

Программист Python:

- Разработать архитектуру программного обеспечения лунохода-спасателя на базе ROS2.
- Определить набор программных узлов, необходимых для выполнения спасательной операции.
- Разработать схему взаимодействия узлов с использованием ROS2-топиков.
- Определить состав передаваемых сообщений между узлами.
- Разработать диаграмму состояний алгоритма.

Программист C:

- Разработать архитектуру встроенного программного обеспечения контроллера системы сцепки.
- Определить набор команд, поступающих от бортового компьютера.
- Разработать блок-схему алгоритма работы контроллера.

- Предусмотреть обработку следующих ситуаций:
  - успешная сцепка;
  - неудачная сцепка;
  - потеря удержания буксируемого аппарата;
  - аварийная остановка операции.
- Подготовить описание интерфейса взаимодействия между контроллером и управляющим программным обеспечением.

Ожидаемые результаты:

По окончании хакатона каждая команда предоставляет:

- 3D-модель механизма буксировки, созданную в Компас-3D (файл .m3d и экспортированный .step);
- изображения конструкции;
- схему взаимодействия ROS2-узлов;
- диаграмму состояний алгоритма спасательной операции;
- блок-схему алгоритма работы контроллера.