Спутниковые системы

2022/23 учебный год

Инженерный тур

Общая информация

Конечным результатом работы каждой команды участников является совместная отработка взаимодействия двух прототипов космических аппаратов (КА), в том числе интегрированной в их состав ПН, а также моделирование работы аппаратов с учетом космических факторов в ПО численного моделирования.

В качестве научного аппарата выступает прототип, собирающийся на базе конструктора Интросат. В качестве спутника-ретранслятора выступает прототип, собранный на основе конструктора Orbicraft 3D. В задачи Программиста Микроконтроллеров входит разработка алгоритмов управления разрабатываемых прототипов. В задачи Схемотехника входит разработка упрощенного прототипа бортовой полезной нагрузки научного аппарата. В задачи Радиотехника входит организация каналов связи Научный аппарат — Ретранслятор и Ретранслятор — ЦУП. В задачи Физика-баллистика/проектировщика входит планирование работы аппаратов на орбите, включая расчет необходимых орбит, целей для ориентации аппаратов, поддержание их энергобаланса на орбите Земли и, в конечном итоге, обеспечение заданных показателей работоспособности всей группировки.

Легенда задачи

Общая задача команды — разработка проекта и комплекта прототипов гетерогенной (разнородной) группировки малых космических аппаратов, из которых часть имеет научную полезную нагрузку, а часть выступает ретрансляторами для системы связи.

Предполагается, что согласно текущей практике запусков малых космических аппаратов, в том числе в рамках проекта «Space-Pi» (https://spacepi.space), одновременно на близкие орбиты выводится несколько различных спутников, разработанных разными организациями и командами участников. При этом некоторые из аппаратов, обладающие собственными двигательными установками, впоследствии совершают маневры расхождения.

Расхождение аппаратов по различным орбитам и организация каналов связи между ними позволяет максимизировать возможности передачи собранных научных данных на конкретные наземные центры управления.

В рамках этой задачи:

- Схемотехники создают проект и прототип производящей требуемые данные полезной нагрузки научного аппарата;
- Программисты микроконтроллеров/разработчики спутниковых систем работают над алгоритмами управления научного аппарата;
- Радиотехники работают над организацией каналов связи между научным аппаратом и спутниками связи, а также между спутниками связи и наземными центрами управления на основе программно-определяемого радио;

- Физики-баллистики (проектировщики спутниковых систем) решают отдельные проблемы управления, которые могут возникнуть у группировки такого типа, а также создают проект группировки с заданными требованиями в системе проектирования и численного моделирования космических миссий.

Требования к команде и компетенциям участников

Количество участников в команде: 4.

Роли в команде могут быть любыми, но в задачах профиля выделяют 4 сбалансированных направления (рекомендуется, чтобы каждым из них занимался выделенный участник):

1. **Программирование микроконтроллеров/разработка спутниковых систем**: На заключительном этапе к этому направлению относится разработка управляющих программ ко всем прототипам; решение задач потребует знакомства с STM32CubeIDE, умения работать с кодом на языке C и базовых понятий об алгоритмах управления.
2. **Радиотехника**: для выполнения заданий заключительного этапа в этом направлении понадобится представление об основах радиосвязи, умение работать с радиомодулями и организовывать каналы связи с использованием различных технических решений. Рекомендовано знакомство с ПО SDR#, GNU Radio и Houston App.
3. **Схемотехника**: для выполнения заданий заключительного этапа потребуются знания о построении электрических цепей, навык пайки, а также знакомство с ПО KiCad.
4. **Физика (баллистика)**: на заключительном этапе к этому направлению относится решение задач по проектированию миссий на орбите; для решения задач здесь в первую очередь необходимо знать основы небесной механики. Рекомендовано знакомство с синтаксисом языка JavaScript для написания алгоритмов в симуляторе.

Оборудование и программное обеспечение

Наименование	**Описание**
Конструктор спутника «Интросат» и комплект расширений, вспомогательных инструментов и материалов: https://introsat.ru	Используется в качестве научного спутника, задачами которого являются: • отработка стабилизации в магнитном поле; • обеспечение возможности интеграции разработанной участниками полезной нагрузки; • чтение данных с полезной нагрузки.
Конструктор спутника «ОрбиКрафт 3Д»: https://sputnix.ru/ru/kosmicheskoe-obrazovanie/konstruktory-sputnikov/konstruktor-sputnika-orbikraft-3d	Используется в качестве спутника-ретранслятора: принимает данные с конструктора «Интросат» на частоте 2.4 ГГц и ретранслирует их на частоте 433 МГц для дальнейшей обработки

Наименование	Описание
Онлайн-сервис «Орбита.Челлендж»: https://orbita.education	Используется для: • публикации заданий финала с разделением по дням; • загрузки решений участников; • решения задач по направлению Физика (Баллистика) на базе ПО численного моделирования.

Описание задачи

Вся задача декомпозируется на 4 направления и также декомпозируется по дням. В каждой роли можно набрать одинаковое количество командных баллов. Обычно выделяется по 100 баллов на каждую роль, которые суммарно можно заработать за все 4 дня. Этот балл потом нормируется в зачете с индивидуальной предметной частью.

Программист МК

День 1

Первый день направлен на сборку оборудования и отработку материала удаленного практикума:

- Видео по удаленной работе с конструктором IntroSat: https://youtu.be/zBBec1O_W_A.

- Презентация к видео: https://disk.yandex.ru/d/KKmoTEdyevIEFA.

В работе можно использовать проект STM32CubeIDE, который получился у вашей команды в результате отработки удаленного практикума, или можно запросить сборку стартового проекта у организаторов.

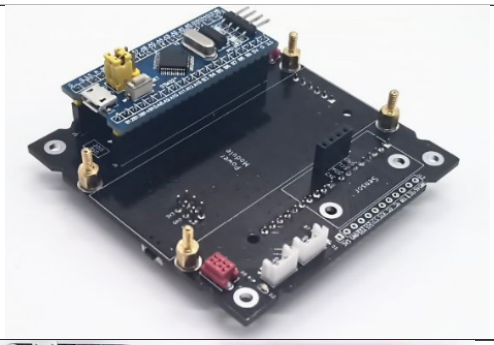
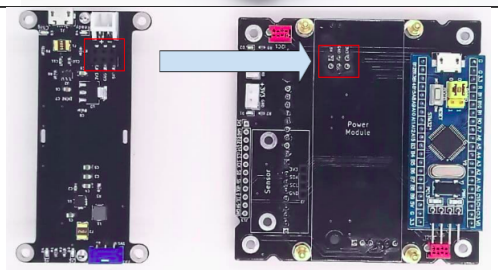
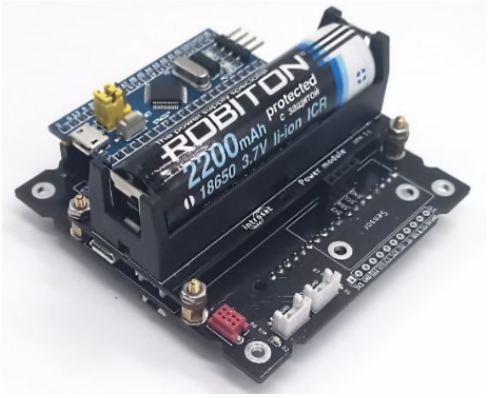
В течение дня необходимо:

- настроить беспроводную передачу информации по UART и беспроводной способ прошивки микроконтроллера;
- считать данные с датчика позиционирования по I2C;
- задать управление двигателем, продемонстрировать изменение направления и скорости вращения;
- задать управление катушками, продемонстрировать изменение полярности на катушках (можно без подключения катушек проверкой потенциала на соответствующем выходе)
- собрать стенд кольца Гельмгольца;
- осуществить расчет и намотку катушек;
- осуществить финальную сборку конструктора IntroSat при наличии времени.

Первичная сборка

Прежде чем приступить к работе с микроконтроллером, необходимо осуществить первичную сборку основных плат конструктора IntroSat.

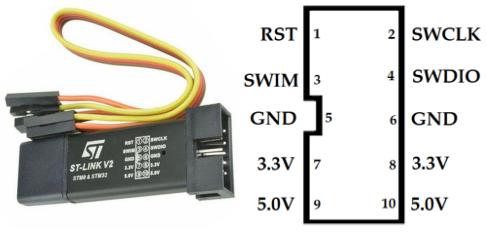
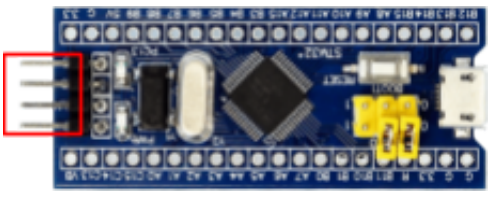
Сборка плат	
Установите плату Blue Pill на материнскую плату. ВАЖНО! Разъем microusb на плате BluePill должен находиться над надписью «USB SIDE».	
Установите модуль Bluetooth в соответствующий разъем	

Закрепите 4 стойки PCHSN5 (длиной 5 мм) на материнской плате при помощи винтов M2.5×4 с полукруглой головкой.	
Установите аккумулятор в модуль питания, при необходимости закрепите его пластиковой стяжкой через ушки в плате. Модуль питания установите на материнскую плату так, чтобы шестиконтактный разъемы сопрягались друг с другом. Закрепите гайками модуль питания	 

После этого можно проверить, что все подключено верно, сдвинув выключатель (белый ползунок) на модуле питания. Должны загореться светодиоды на плате Blue Pill и bluetooth-модуле.

Обмен данными по UART и беспроводная прошивка

К материнской плате подключается ведомый bluetooth-модуль, а ведущий соединяется с USB-UART конвертером по следующей схеме.

ST-Link V2[1]	**Blue Pill**
	
(2) SWCLK >— (4) SWDIO >— (5) или (6) GND >— (7) или (8) 3.3V >—	—> SCK —> DIO —> GND —> 3.3V

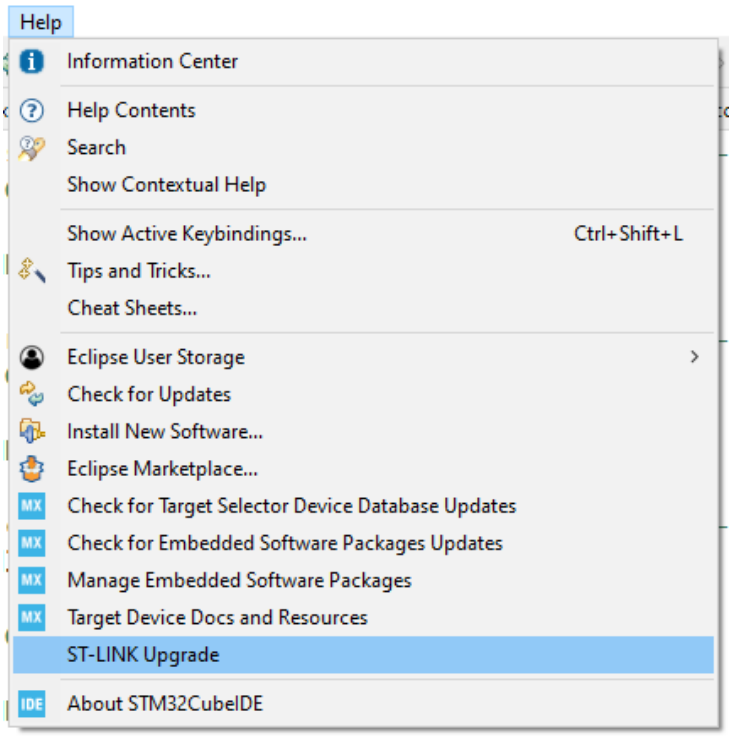
После подключения программатора к микроконтроллеру и затем к usb-порту ПК

[1]Распиновка может отличаться от приведенной, однако важно соединить правильно соответствующие надписи

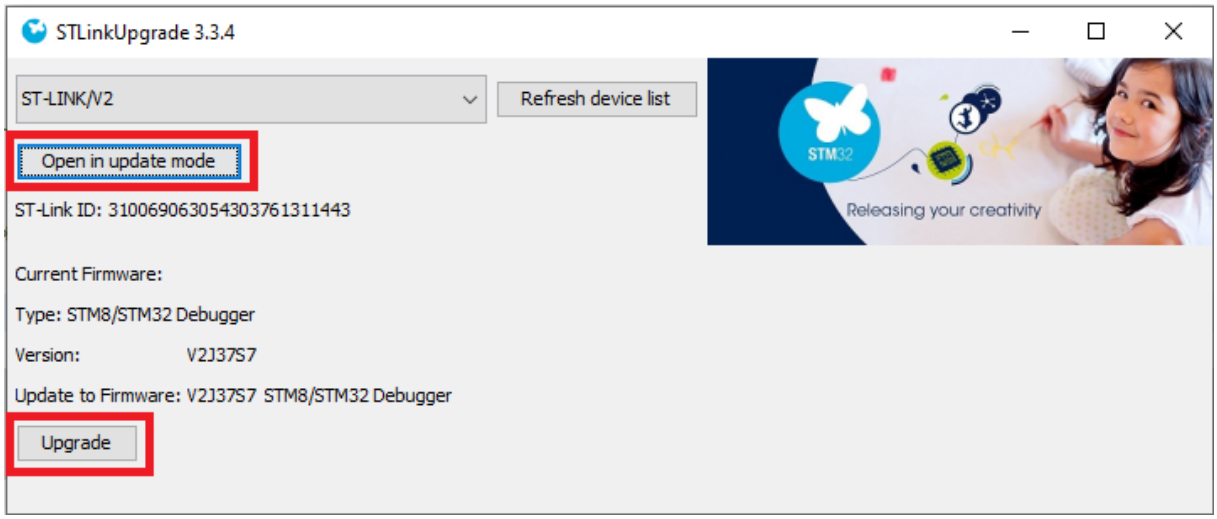
станет возможна загрузка программы в плату.

Для загрузки программы в плату нажмите на значок ( ) в верхней панели меню.

После этого, возможно, ST-Link попросит обновления. Для этого во вкладке меню Help есть соответствующая опция.



Выберите ST-LINK Upgrade. Откроется новое окно. В нем выберите среди устройств ST-LINK/V2 и затем Open in update mode. После этого нажмите Upgrade.



Иногда требуется переподключить программатор, чтобы утилита по его обновлению идентифицировала его.

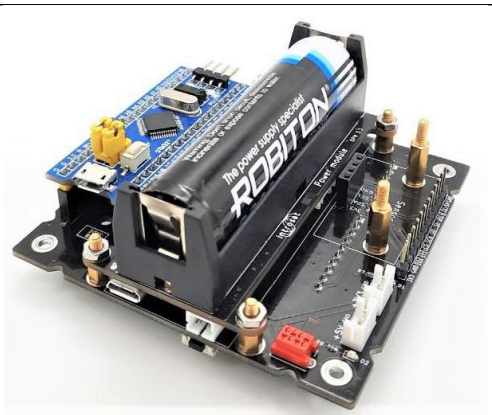
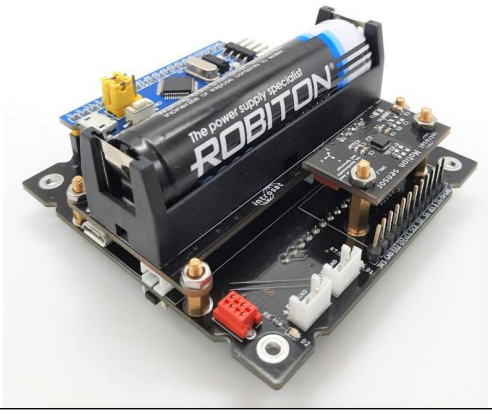
Теперь можно залить программу в память микроконтроллера, нажав значок . Если все правильно, то после окончания загрузки вы увидите, как на плате мигает синий светодиод с периодичностью в секунду.

ПРИМЕЧАНИЕ

Помимо режима загрузки существует также режим отладки. Данный режим бывает крайне полезен, если где-то в коде закралась ошибка, но сходу ее не удается найти. Для перехода в режим отладки (debug) можно нажать на значок жучка (`img2.png`) в верхней панели меню. О том, как пользоваться режимом отладки можно посмотреть, например, здесь: https://youtu.be/BVC7KaUNCS8.

На очном финале весьма полезно будет уметь пользоваться данным режимом.

Работа с датчиком позиционирования

Закрепите датчик позиционирования на материнской плате конструктора IntroSat. Закрепите 2 стойки PCHSN11 (длиной 11 мм) на материнской плате при помощи винтов M2.5×4 с полукруглой головкой.	
Установите датчик позиционирования в соответствующий разъем и зафиксируйте его гайками.	

Работа с датчиком позиционирования происходит по интерфейсу I2C. Необходимо считать значения угловых скоростей и проекций вектора магнитной индукции по трем осям и вывести их в COM-порт компьютера по UART.

Работа с двигателем

Подготовьте к работе плату маховика.

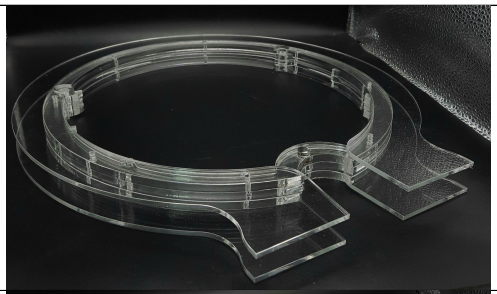
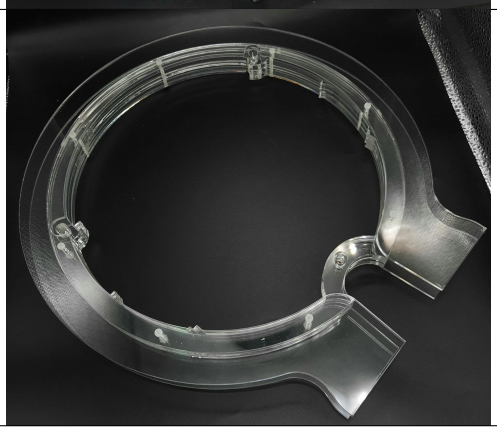
Установите маховик на адаптер на оси двигателя с помощью пары винтов M2.5×6 с выпуклой головкой.	

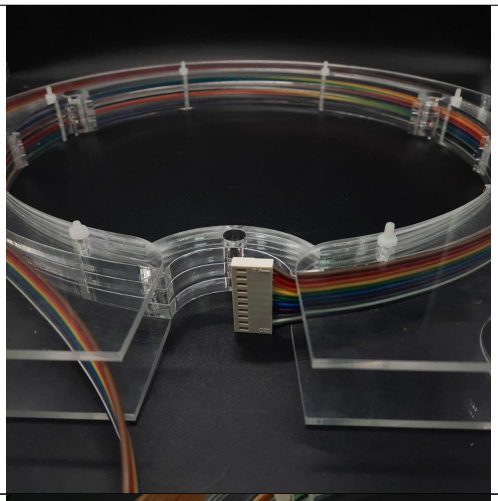
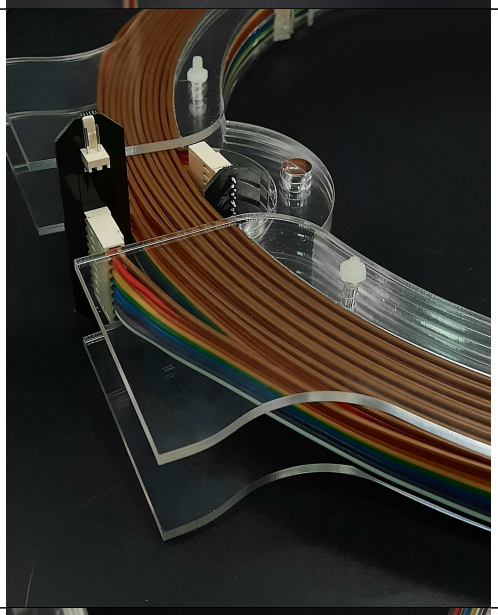
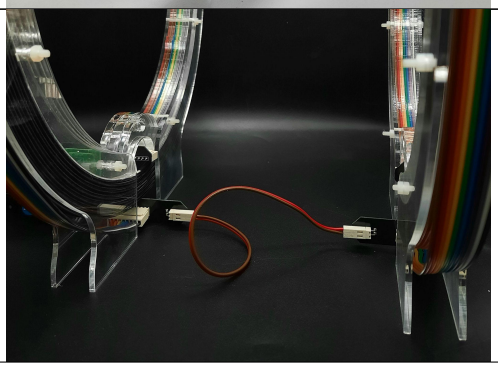
Прикрутите плату маховика к нижней грани куба винтами с потайной головкой M2.5×12 и зафиксируйте гайками.	

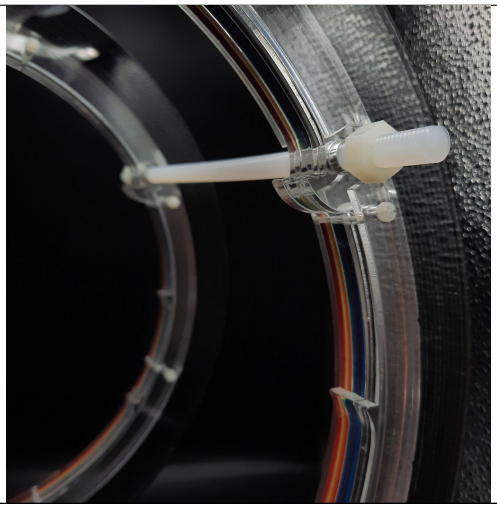
Подключите модуль маховика к материнской плате с помощью шлейфа I2C и провода питания. Для этого можно использовать любые из доступных разъемов MicroMatch на материнской плате и модуле маховика. Питание на двигатель маховика подается от 5-вольтового вывода материнской платы.

Напишите код управления двигателем, который будет сначала вращать маховик с нарастающей скоростью в одну сторону, затем останавливать его и вращать в другую сторону.

Сборка стенда кольца Гельмгольца

Сложите кольца друг на друга, как показано на фото.	
Зафиксируйте кольца пластиковыми винтами и соответствующего размера гайками относительно друг друга.	

Намотайте шлейф на кольца. Один коннектор должен быть расположен внутри, другой снаружи.	
Подключите плату коммутации колец к собранному кольцу Гельмгольца.	
Аналогичным образом соберите второе кольцо Гельмгольца и подключите к нему ответную часть платы коммутации колец. Возможно, что после подключения плат наружная часть шлейфа будет лежать свободно. Вы можете зафиксировать ее с помощью изоленты.	
Соедините платы колец между собой соответствующим проводом.	

Далее можно вставить направляющие в 3 сквозных отверстия на кольцах и гайками регулировать расстояние между ними. Измерять расстояние между кольцами можно при помощи линейки, входящей в набор.	

Расчет и создание магнитных катушек

Сперва нужно рассчитать необходимое количество витков в магнитных катушках, исходя из следующих рассуждений:

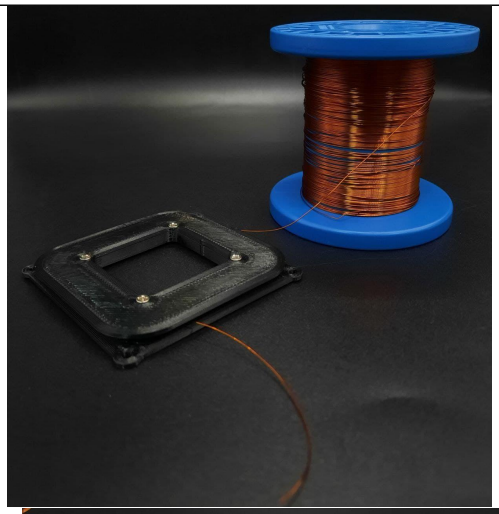
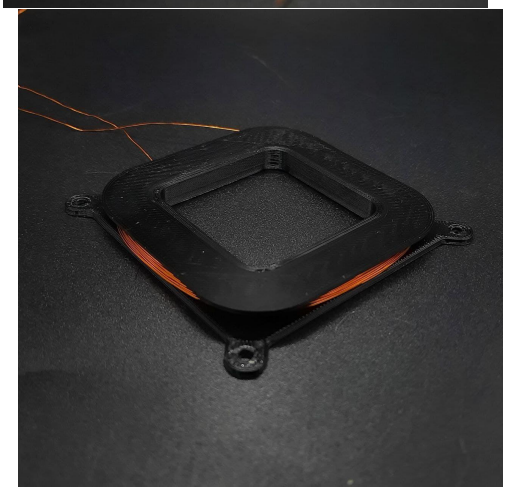
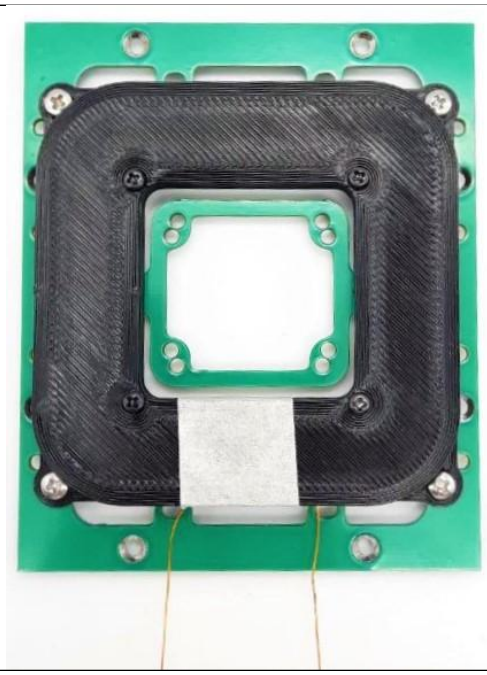
- необходимо определить магнитную индукцию, создаваемую кольцами Гельмгольца;
- необходимо определить момент сил, действующий на катушки спутника;
- с помощью своих витков катушки должны создавать такой момент, чтобы:
- придать спутнику на подвесе необходимый момент вращения;
- преодолеть момент трения подвеса.

ПРИМЕЧАНИЕ
Обязательно проверьте, какое сопротивление получится на каждой из катушек после их намотки. Так как сопротивление цепи должно быть порядка 10 Ом, а катушки должны быть подключены параллельно, чтобы обеспечить большую мощность их работы, можно записать формулу для общего сопротивления цепи:

$$R_{\text{общ}} = \frac{R_{\text{кат}} \cdot R_{\text{кат}}}{R_{\text{кат}} + R_{\text{кат}}} = \frac{R_{\text{кат}}}{2}$$

Отсюда ориентировочно сопротивление катушки должно быть порядка 20 Ом.
Если у намотанных катушек сопротивление будет значительно меньше, можно добавить в цепь резисторы.

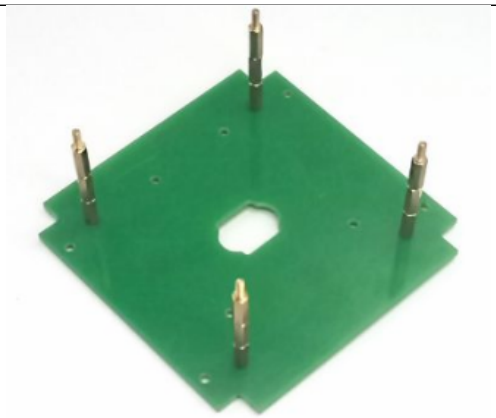
Когда будет известно количество витков для магнитных катушек, можно приступать к их изготовлению.

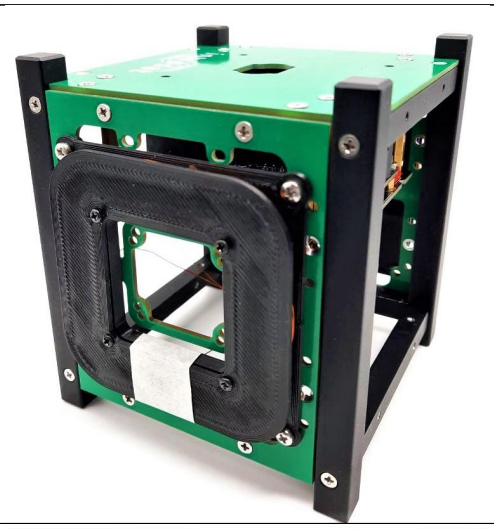
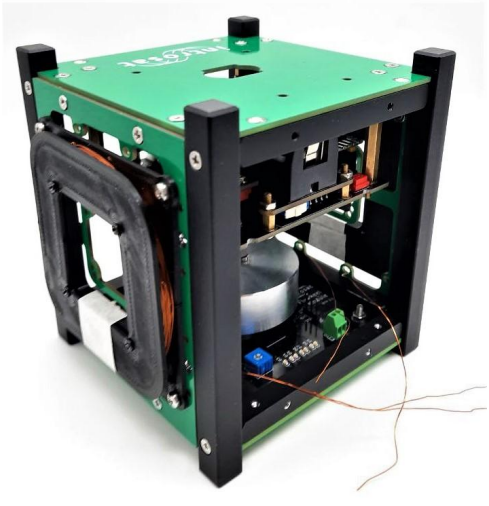
Намотайте проволоку на рамку катушки: сделайте рассчитанное количество витков. Необходимо намотать две катушки. Проверьте мультиметром сопротивление полученных катушек: оно должно быть порядка 20 Ом.	 
Закрепите катушку на грани с помощью винтов M2.5×6 с полукруглой головкой и закрепите гайками с обратной стороны.	

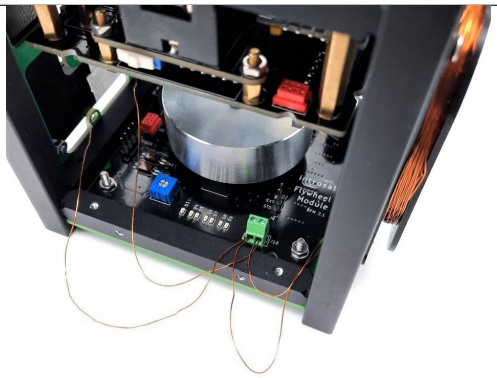
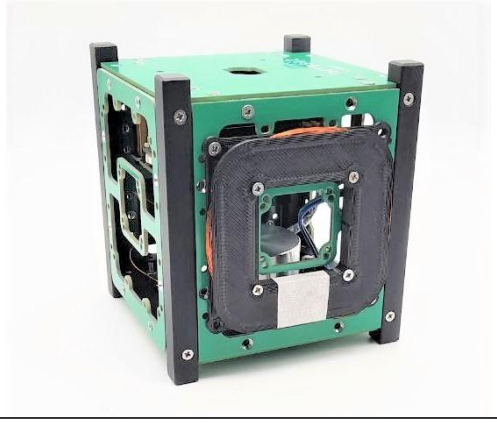
Финальная сборка конструктора IntroSat

Перед финальной сборкой рекомендуется проверить правильность всего подключения, чтобы избежать необходимости пересборки. Также удобным будет оставить ST-Link подключенным к Blue Pill, оставив его снаружи: вероятно, понадобится перепрошивать или отлаживать микроконтроллер через него.

Сборка корпуса	
Возьмите два рельса и установите в выемки перекладины с пазом.	
В получившиеся углубления на стыке рельса и перекладины установите перекладины с торцевым отверстием так, чтобы поперечные отверстия в этих перекладинах были ориентированы перпендикулярно рельсу (они используются для установки боковых панелей). Закрепите угловое соединение винтами M2.5×12 с потайной головкой.	

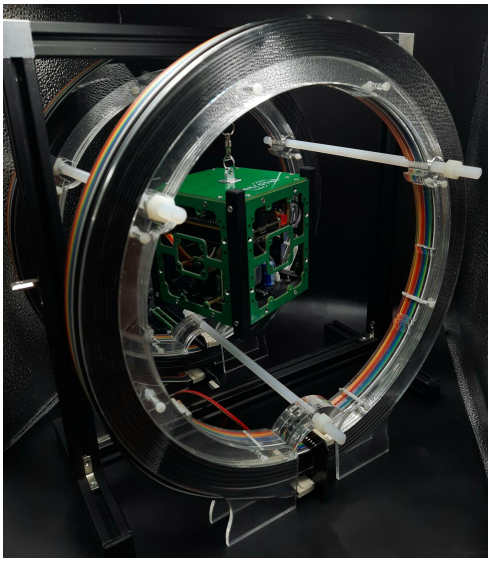
Установите две оставшиеся перекладины и два оставшихся рельса аналогичным образом и окончательно соберите конструкцию при помощи винтов M2.5×12.	
Установите собранную материнскую плату на верхнюю панель с помощью межплатных стоек PCHSN-11 (для этого нужно будет соединить стойки между собой по 3 шт.) и гаек.	 

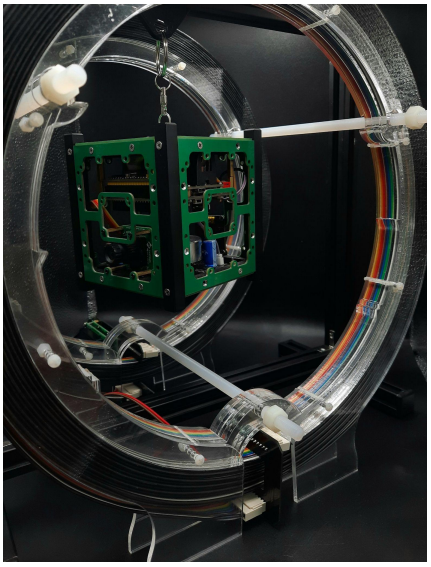
Сборка и подключени	
Закрепите верхнюю панель на раме с помощью четырех винтов M2.5×6 с потайной головкой.	
Закрепите панели с катушками на двух противоположных гранях. Лучше крепить боковые грани так, чтобы был доступ к кнопкам вкл/выкл и RESET, то есть справа и слева от модуля питания.	
Закрепите нижнюю панель с платой маховика четырьмя винтами M2.5×6.	

Для подключения катушек на плате маховика предусмотрена клемма на два входа: 5 В и GND. Соответственно, по одному концу проволоки от каждой катушки необходимо вставить в разъем 5 В, а два оставшихся в GND. Закрутите клемму при помощи часовой отвертки, идущей в комплекте. Важно перед подключением зачистить конец проволоки, чтобы обеспечить хороший контакт. Таким образом, две катушки должны быть подключены к плате параллельно.	
Подключите к плате маховика питание 5 В и шлейф I2C, предварительно обернув его фольгированной самоклеящейся бумагой.	
Закрепите оставшиеся две боковые грани с помощью винтов M2.5×6.	

Конечная сборка

Установите подвес между кольцами Гельмгольца и подвесьте спутник.





Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

Необходимо подгрузить:

- файл `.ioc` и `main.c` с написанным кодом из проекта;
- расчет катушек в виде фото, кода или скриншота.

День 2

В течение дня необходимо:

- доделать задачи первого дня;
- реализовать автоматическое постоянное вращение с помощью катушек в магнитном поле;
- реализовать разгрузку маховика.

Примечание по работе с Bluetooth-модулями

Для более стабильной работы скорость UART на модулях снижена до 57600. Чтобы задать эту настройку в проекте минимально достаточно поменять скорость в функции настройки UART в файле `main.cpp`.

```
static void MX_USART1_UART_Init(void){
    /* USER CODE BEGIN USART1_Init 0 */

    /* USER CODE END USART1_Init 0 */

    /* USER CODE BEGIN USART1_Init 1 */

    /* USER CODE END USART1_Init 1 */
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 115200;
    huart1.Init.WordLength = UART_WORDLENGTN_9B;
```

```
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_EVEN;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN USART1_Init 2 */

    /* USER CODE END USART1_Init 2 */
}
```

Необходимо задать: `huart1.Init.BaudRate = 57600;`

Учтите, что данная настройка может скинуться при повторной генерации проекта (не следует путать сборку () и генерацию ()).

Не забудьте поменять настройки поменять настройки скорости в терминале и CubeProgrammer при прошивке.

Разгрузка маховика

У каждого двигателя-маховика есть предел угловой скорости, которую он может развить. Если на космический аппарат периодически действует сила, которая закручивает его в одну и ту же сторону, такой предел рано или поздно наступит. Когда маховик достигает предельных скоростей вращения, наступает «эффект насыщения». В таких случаях маховик стараются «разгрузить». Разгружение происходит за счет других устройств систем ориентации, например, электромагнитных систем.

Порядок выполнения разгрузки:

1. Раскрутите маховик до максимальных оборотов. Подождите пока спутник остановится за счет того, что двигатель войдет в насыщение;
2. Затем нужно постепенно сбрасывать скорость вращения маховика, но при этом за счет работы катушек спутник должен удерживаться на месте. Допускается первоначальный поворот спутника до 90 градусов для установки спутника в положение работоспособности катушек.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие. Необходимо подгрузить файл `.ioc` и `main.c` с написанным кодом из проекта.

День 3

В течение дня необходимо:

- Реализовать алгоритм стабилизации с помощью магнитных катушек (алгоритм B-dot).
- Произвести интеграцию радиомодуля и полезной нагрузки в сборку спутника.
- Реализовать финальную циклограмму спутника. Циклограмма должна состоять из следующих шагов:
  - стабилизация с помощью магнитных катушек (первоначальная закрутка производится маховиком на максимальной скорости в течение 30 секунд);

– нагрев (с заполнением ШИМ в 15%) и освещение емкости, расположенной в полезной нагрузке;
– сбор данных с датчиков полезной нагрузки для последующего формирования пакета;
– отправка пакета (сформированного в задании для Инженера связи) на Orbicraft 3D.

Алгоритм B-dot

Часто сразу после отделения космического аппарата от ракеты-носителя предлагается использовать так называемый B-dot алгоритм. Он достаточно часто используется на микроспутниках для предварительного успокоения. Катушки магнитной системы генерируют магнитный момент, пропорциональный скорости изменения вектора индукции магнитного поля Земли, взятой с обратным знаком.

Пусть B_3 — вектор магнитного поля Земли, P_{KA} — магнитный момент космического аппарата, K — коэффициент пропорциональности ($K > 0$).

Величина магнитного момента, который должны создавать катушки КА:

$$P_{KA} = -K\frac{\Delta B_3}{\Delta t}.$$

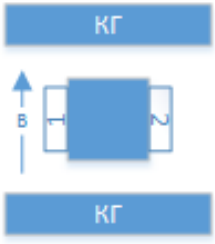
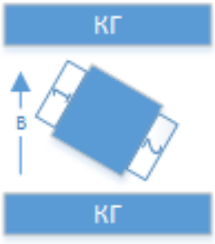
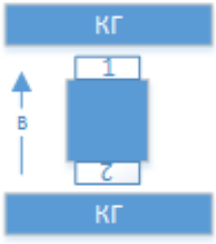
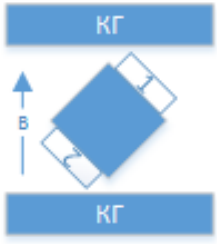
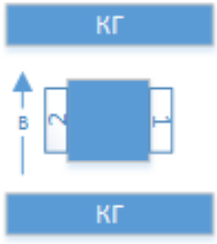
Вычисление P_{KA} следует привести к диапазону $[-32000; 32000]$, что соответствует управлению катушками по ШИМ (PWM).

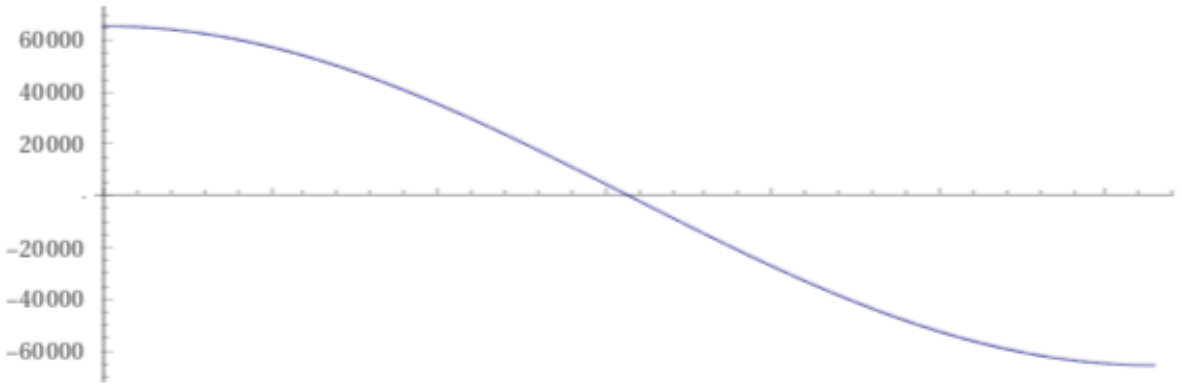
В данном случае следует использовать изменение магнитной индукции только по оси X (m_x) или Y (m_y), так как общая проекция вектора на плоскость XOY будет примерно постоянной.

$$PWM = -K\frac{m_y - m_{y_{prev}}}{t - t_{prev}}.$$

При таком управлении момент, создаваемый катушками, будет возрастать при приближении спутника к положению, когда катушки расположены к кольцам под 90°, и убывать, когда катушки будут параллельны им. Также код сам будет определять, когда необходимо поменять полярность на катушках.

Ниже для примера схематично показана такая зависимость вычисляемого значения ШИМ (PWM) и положения спутника в магнитном поле колец Гельмгольца, к которой нужно стремиться.

Макс. маг. момент	Маг. момент начинает убывать	Мин. маг. момент и переключение полярности	Маг. момент начинает возрастать	Макс. маг. момент
				



При положительном значении ШИМ нужно задавать одно направление полярности на катушках, при отрицательном — другое.

Примеры управления датчиками

Для управления датчиком температуры можно использовать библиотеку DS18B20, а для датчика цвета — TCS34725.

Для управления DS18B20 необходимо в соответствующие блоки добавить следующие строки:

1. `#include <DS18B20.h>`
2. `DS18B20Init(&huart2);`
   Для корректной работы библиотеки необходимо инициализировать UART2. Строку № 2 необходимо использовать в коде только при наличии датчика на шине, иначе микроконтроллер может уйти в зависание.
3. `float a = DS18B20GetTemperature();`

Для управления TCS34725 необходимо в соответствующие блоки добавить следующие строки:

1. `#include <TCS34725.h>`
2. `TCS34725Init(&hi2c1);`
3. Блок следующих переменных:

```
uint16_t r, g, b, c;
r = TCS34725Red();
g = TCS34725Green();
b = TCS34725Blue();
c = TCS34725Clear();
```

Для корректной работы библиотеки необходимо считывать все 4 спектра.

Библиотеки при необходимости можно запросить у организаторов.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

Необходимо подгрузить:

- фото спутника в сборе;
- файл `.ioc` и `main.c` с написанным кодом из проекта.

Инженер связи

День 1

В течение дня необходимо передать сообщение с передатчика nRF24l01 на Орбикрафт 3D, соблюдая необходимый формат команды. Для этого необходимо сделать следующее:

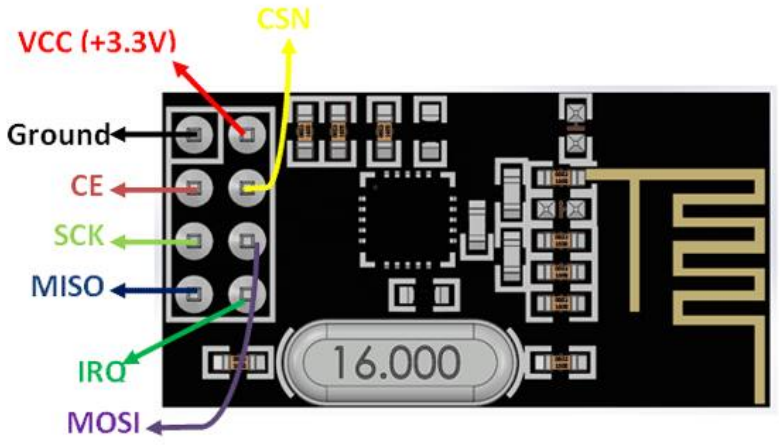
- осуществить сборку конструктора Орбикрафт 3D;
- осуществить подключение приемопередатчика к конструктору Орбикрафт 3D в модуль полезной нагрузки;
- осуществить подключение приемопередатчика к конструктору IntroSat;
- написать код управления для передатчика, соблюдая требования по формированию команды управления;
- убедиться в том, что команда управления принята конструктором Орбикрафт 3D: при успешном выполнении команды на конструкторе включится маховик.

Сборка конструктора Orbicraft 3D

Для сборки конструктора воспользуйтесь инструкцией, которая вложена в кейс. Если Вам нужна дополнительная инструкция, перейдите по ссылке: http://orbicraft3d.sputnix.ru/doku.php?id=download1.

Работа с nRF024l01

ПВ создании радиоканала используется два модуля nRF024l01: передающий модуль устанавливается на конструкторе IntroSat, а принимающий — на конструкторе Орбикрафт 3D.



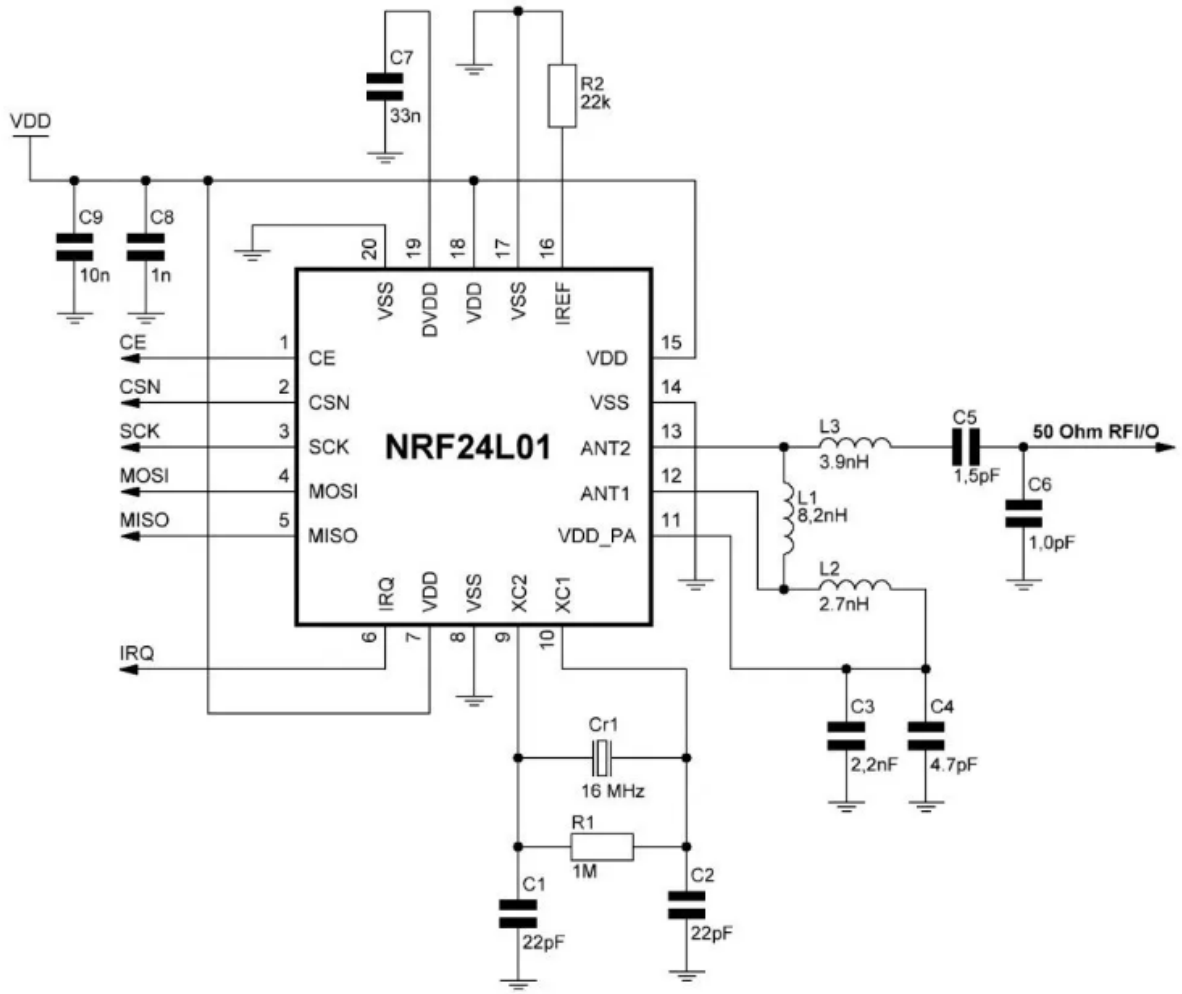


Рис. VI.2.1. Электрическая схема nRF024l01

Приемопередатчик nRF024l01 работает по интерфейсу SPI.

Подключение модуля nRF024l01 к конструктору Orbicraft

Подключение датчика осуществляется через подключение к плате Arduino Shield по интерфейсу SPI. На шилд конструктора ОрбиКрафт 3D необходимо подключать следующие пины:

```
CE ->10
CSN ->9
GND ->GND
VCC -> 3.3V
```

Пин IRQ не подключается.

Ниже указаны порты подключения по SPI к плате Arduino Shield.

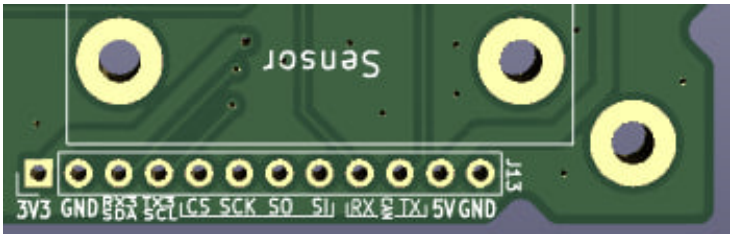
XP7

Конт.	Цепь
1	PB3 (MISO)
2	NC
3	PB1 (SCK)
4	PB2 (MOSI)
5	NC
6	GND

Подключение модуля nRF024l01 к конструктору IntroSat

Для параллельной работы с передатчиком у команды имеется двойной набор плат Blue Pill, материнских плат, модулей питания, ST-Link и UART-конвертеров. Сборка плат описана в задании для программиста микроконтроллеров в подразделе «Первичная сборка».

Передатчик подключается к материнской плате.



Оставшийся пин IRQ можно подключить к пину PA8 на обратной стороне материнской платы.

Использование частот радиоканалов

Для каждой команды предписано использование своего канала для пары приемо-передатчиков. Канал приемника предустановлен и указан на комплекте оборудования в коробке «Модуль ПН» (Channel). Для передатчика необходимо указать номер канала в коде управления.

Канал УКВ также предустановлен и указан в коробке «Модуль ПН» (Частота в МГц).

Пример управляющего кода для передатчика

Для написания управляющего кода для передатчика можно воспользоваться собранным проектом, который будет предоставлен организаторами.

В проекте в файле `main.c` необходимо для начала подключить необходимые библиотеки в соответствующий блок.

```
/* USER CODE BEGIN Includes */
#include "NRF24.h"
#include "stdio.h"
```

Далее необходимо задать несколько переменных.

```
/* USER CODE BEGIN PV */
char str1[20] = {0};
uint8_t buf1[20] = {0};
```

```
struct __attribute__((packed)) data_rpm {
  float rpm;
};
```

Структура будет служить для передачи данных на Орбикрафт 3D. Затем необходимо в функции main в соответствующем блоке инициализировать передатчик.

```
/* USER CODE BEGIN 2 */

NRF24_ini();
NRF24_WriteReg(RF_CH, XX);
```

Вместо XX необходимо указать канал, который указан на оборудовании. Затем в том же блоке необходимо определить каждый элемент массива данных для передачи.

```
struct data_rpm d_data_rpm;  // зададим переменную, соответствующую заданной
                             // выше структуре данных
uint8_t data[32];  // зададим массив для данных заданного размера

// Определим значение элементов структуры
d_data_rpm.rpm = 2000.0;  // Число в качестве примера

uint16_t msgId = 0xA01;  // Зададим переменную команды, которую хотим отправить
                         // на Орбикрафт 3D
data[0] = sizeof(data);  // Определим размер передаваемых данных и поместим
                         // значение в первый элемент массива
data[1] = 0xB;  // Зададим адрес устройства Орбикрафт 3D, которому хотим
                // отправить команду Преобразуем в 2 байта значение команды
data[3] = msgId & 0xFF;
data[2] = msgId >> 8;

// Запишем в массив указатели на данные структуры
for (int i = 0; i < sizeof(d_data_rpm); i++) {
  data[i + 4] = *((uint8_t *)&d_data_rpm + i);
}
NRF24_PacketLen(32);  // Зададим в настройках передатчика длину пакета данных,
                      // который хотим отправить
```

После этого в цикле while будем отправлять данные на приемник.

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  HAL_Delay(2000);  // Обязательная задержка в отправке команд для корректной
  ↪  работы
  NRF24L01_Send(data);
```

После добавления всех строчек кода можно собрать проект и прошить микроконтроллер.

Если все правильно, то на «Орбикрафт 3D» заработают маховики.

Включение маховика на конструкторе «ОрбиКрафт 3D»

У каждой команды конструктора Орбикрафт, предназначенной для управления устройствами, есть следующие значащие поля:

- адрес устройства;
- номер команды;
- передаваемое сообщение;
- размер передаваемого сообщения.

Адрес и шифр разделены дефисом, название отделено пробелом, например B-A01 rpm.

Разберем некоторые команды, предназначенные для управления маховиком.

Команда B-A01 rpm — установка скорости вращения маховика:

- B — адрес по умолчанию (в данном случае — адрес модуля маховика — `0x0A`);
- A01 — шифр команды, по которому подсистемы понимают, что это за команда;
- rpm (rate per minute) — название команды установки скорости вращения маховика.

Для корректного составления команды для передачи можно ознакомиться с полями команд в таблице: https://disk.yandex.ru/i/PE8sTLiSGFJiVw.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

Необходимо подгрузить:

- файл `.ioc` и `main.c` с написанным кодом из проекта для отработки работы маховика;
- файл `.ioc` и `main.c` с написанным кодом из проекта для работы маховика.

День 2

В течение дня необходимо:

- доделать задачи первого дня;
- передать сообщение на приемник конструктора ОрбиКрафт 3D в соответствующем формате, которое содержит указанный формат данных конструктора IntroSat на конструкторе Orbicraft в УКВ-передатчик, который будет ретранслировать указанные данные по своему бортовому УКВ-передатчику;
- в SDR# проверить наличие сигнала и частоту передачи
- начать декодирование сигнала в GNU Radio;(демодулировать и получить пакет данных без конечного декодирования передаваемых данных).

Требования к формированию сообщения для передачи между спутниками В коде управления для nrf24l01 конструктора IntroSat необходимо сформировать пакет данных, который:

- содержит длину пакета,
- адрес устройства получателя, в данном случае `0x1` — Земля,
- тип передаваемых данных; не рекомендуется выбрать подходящий из раздела Payload (Arduino) (`0x1A`) в таблице: https://disk.yandex.ru/i/PE8sTLiSGFJ

`iVw`.

Номер байта в сообщении	1	2	3	4	5	. . .	n
Описание	Длина пакета	Адрес получателя	Адрес команды/ тип данных		Данные		
Тип данных	`uint8_t`	`uint8_t`		`uint16_t`			
Комментарий	*Полная длина пакета, включая поля длины пакета и адресов*		*Старшие 8 бит*	*Младшие 8 бит*	*Последовательность байтов в структуре определяемой адресом*		

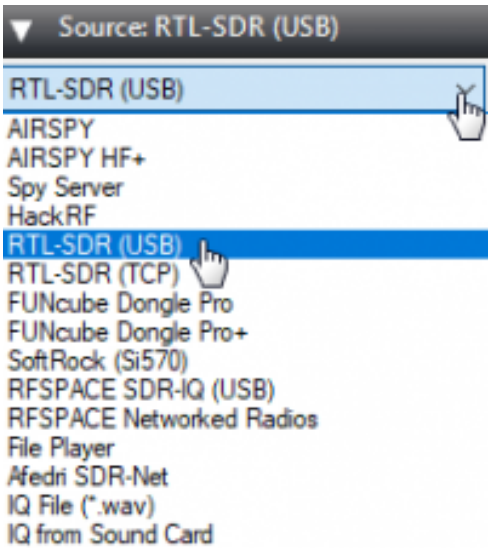
В разработке кода можно опираться на пример кода прошлого дня.

В качестве передаваемых данных необходимо передавать бортовое время с конструктора IntroSat в миллисекундах.

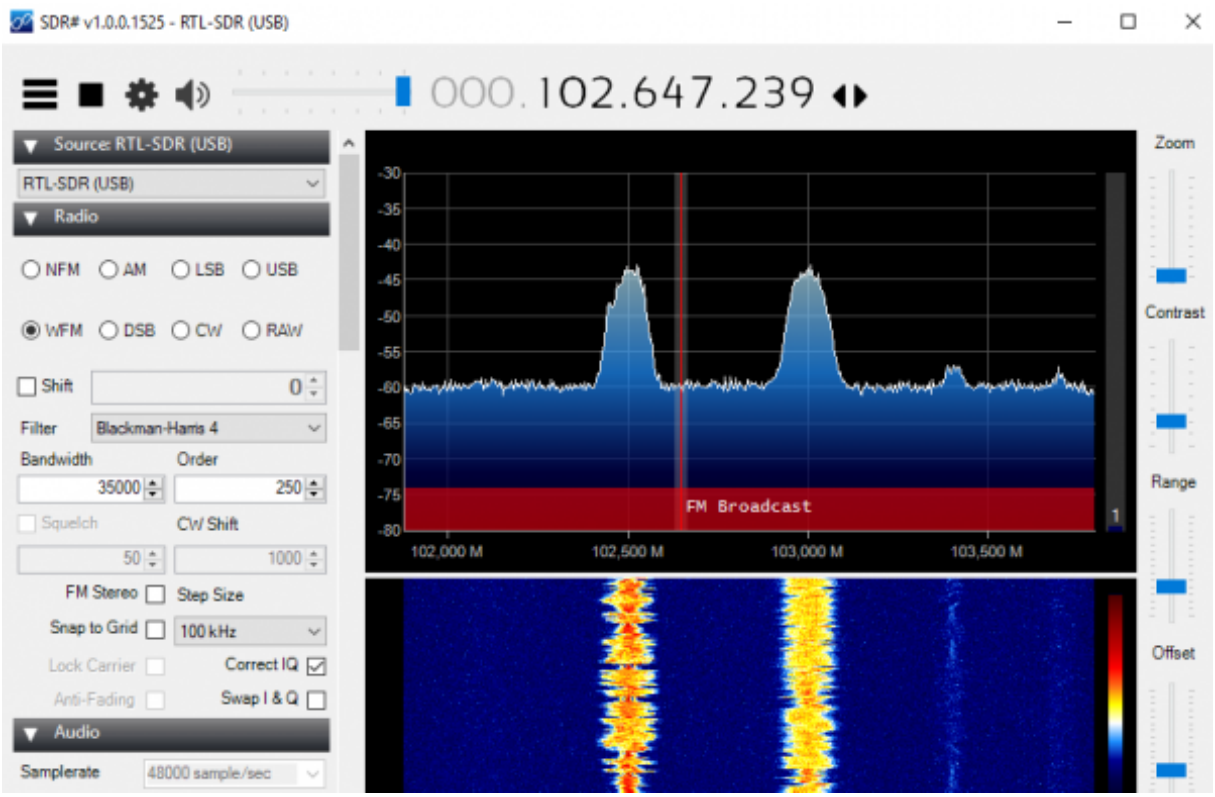
Если все правильно, то с конструктора Orbicraft 3D будет передаваться сигнал, который можно посмотреть в SDR#.

Подключение SDR-приемника к SDR Sharp

Вставьте RTL-SDR в ПК и запустите SDR#. При первом запуске программы вы должны выбрать тип радио: RTL-SDR, подключенный через USB.



Если приемник инициализируется, то можно будет запустить его работу, нажав на значок в левом верхнем углу. При успешном запуске появится уровень сигнала в установленном диапазоне.

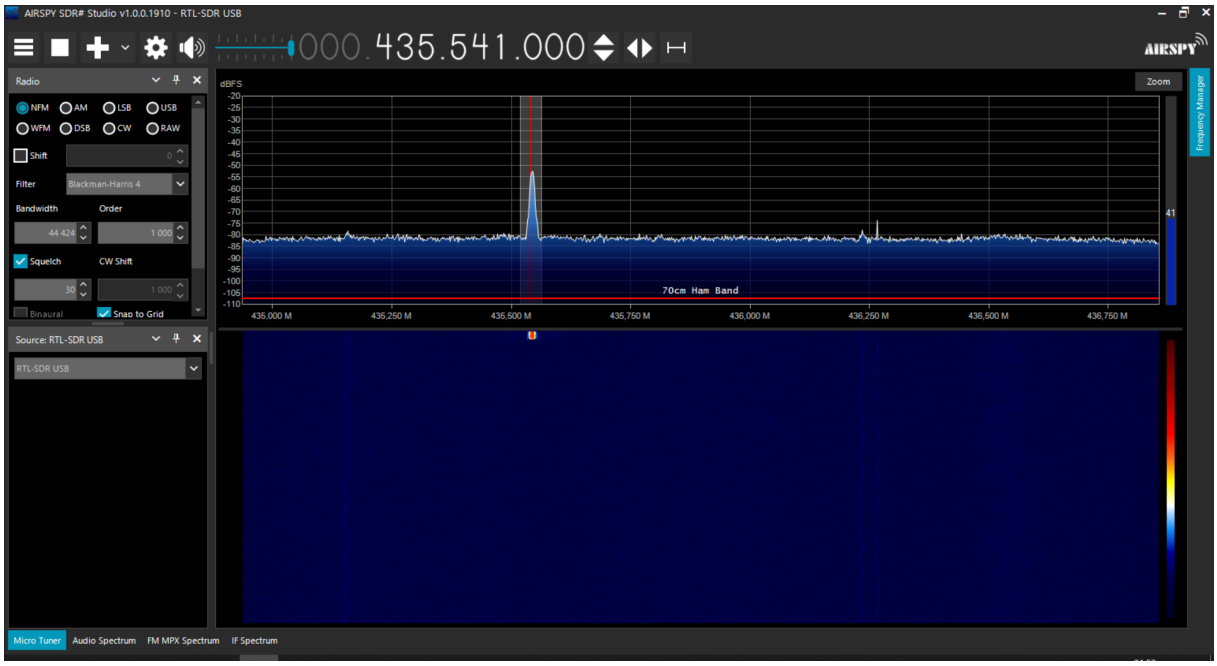


На верхнем графике показана мгновенная сила сигнала, отложенная вдоль вертикальной оси.

Внизу — мощность отображается в яркости и цвете, а время изображено на вертикальной оси. Этот график называется водопадом.

Частота принимаемых сигналов отображается на горизонтальной оси.

В SDR# сигнал с ОрбиКрафт 3D выглядит следующим образом.



Частота бортового УКВ указана на коробке модуля ПН.

Из-за особенности работы радиоприемника, частота сигнала на которой приемник

будет принимать сигнал может немного отличаться от реальной. Ориентируйтесь на показания мощности сигнала. Частота сигнала на приемнике будет совпадать в SDR# и GNU Radio. Будьте внимательны и не перепутайте свой сигнал с сигналом другой команды.

Из-за избыточной мощности передатчика возможны «отражения» сигнала на других частотах. Следует определить пик с максимальной мощностью и убедиться, что при смене прослушиваемой частоты положение сигнала не следует за окном прослушивания.

GNU Radio

После получения сигнала на SDR-приемник следует его обработать в ПО GNU Radio.

Общий ход построения графа можно посмотреть в вебинаре: https://youtu.be/wsBSd6rZTNU.

Обращаем внимание на то, что информация вебинара разбирает в качестве примера сигнал с совершенно другими характеристиками. Информация вебинара поможет понять общую логику графа и информацию о назначении блоков.

Информация, необходимая для построения графа:

- **Модуляция**: GMSK
- **Скорость передачи**: 9600 бод/с
- **Протокол**: AX.25 (Напоминаем, сами данные извлекать из пакета не требуется)

В качестве конечного результата текущего дня необходимо вывести пакет AX.25 на блок Message Debug PDU Vectors.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

День 3

В течение дня необходимо:

- завершить начатый граф в GNU Radio, декодировав телеметрию, приходящую с УКВ-передатчика конструктора ОрбиКрафт 3D до вида отправленных данных;
- осуществить передачу пакета с данными от полезной нагрузки с IntroSat на ОрбиКрафт 3D, с ОрбиКрафт 3D на SDR-приемник; часть кода по формированию пакета интегрируется затем в код программиста МК.

Требования к графу

Телеметрия должна быть декодирована в человекочитаемый вид, обработка телеметрии должна проходить строго в автоматическом порядке. Для этого необходимо создать блок с пользовательским кодом (Python block).

Рекомендуем взять следующий код для работы с данным блоком.

```
"""
Embedded Python Blocks:
```

```
Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

from gnuradio import gr
import struct
import pmt

class blk(gr.basic_block):  # other base classes are basic_block, decim_block,
↪  interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.basic_block.__init__(
            self,
            name='Example Python Block',   # will show up in GRC
            in_sig=None,
            out_sig=None
        )

        self.message_port_register_out(pmt.intern('string_data'))
        self.message_port_register_out(pmt.intern('float_data'))
        self.message_port_register_in(pmt.intern('msg_in'))
        self.set_msg_handler(pmt.intern('msg_in'), self.handle_msg)

    def handle_msg(self, msg):
         msg_decoded = bytes(pmt.to_python(msg)[1])
         ret = self.proceed(msg_decoded)
         resultFloat = pmt.PMT_NIL
         for i in ret:
            if (type(ret[i]) == str):
                self.message_port_pub(pmt.intern("string_data"),
                ↪  pmt.cons(pmt.intern(i), pmt.intern(ret[i])))
            else:
                resultFloat = pmt.list_add(resultFloat, pmt.list2(pmt.intern(i),
                ↪  pmt.make_f32vector(1, ret[i])))
         self.message_port_pub(pmt.intern('float_data'), resultFloat)

    def proceed(self, data):
        #TODO: Здесь должен быть ваш код
        pass
```

`data` — массив байтов, полученных на вход блока.

Функция должна вернуть `dict`, в котором в качестве ключа используется название поля (см. далее в описании блоков), а в качестве значения — декодированное значение.

Далее, для вывода данных необходимо использовать 2 блока:

- Для текстовых данных (`data_string`) — QT GUI Message Edit Box.
- Для числовых данных (графиков) (`data_float`) — QT GUI Time Plot.

Настройки для QT GUI Message Edit Box:

- Type: String
- Label: Отображаемое название
- Pair Mode: True

- Static Mode: True
- Key: Название поля

Настройки для QT GUI Time plot:

- Update interval [ms]: Время между обновлениями графика. Можно поставить 10
- Label y: Название поля
- Axis y: [нижняя граница по y, Верхняя граница по y]
- Range x: Размер оси x

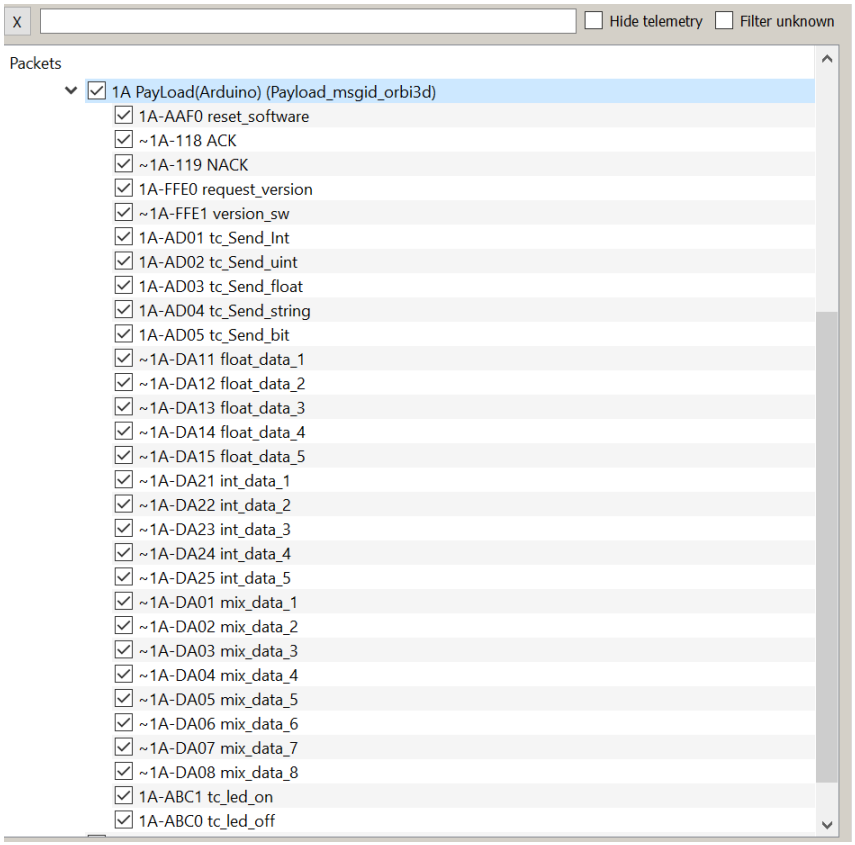
Указания к формированию пакета

В пакете данных, передаваемых с IntroSat необходимо передать следующую информацию:

- бортовое время в миллисекундах;
- название команды;
- данные с гироскопа;
- данные с датчика температуры и цвета с полезной нагрузки.

Получив данные с IntroSat в модуль полезной нагрузки ОрбиКрафт 3D с помощью канала связи, построенном на nrf24l01, конструктор посылает данные по CAN-шине в бортовой вычислительный модуль и отправляет их по УКВ в GNU Radio.

Для передачи данных по CAN-шине в ОрбиКрафт 3D используются команды с адреса полезной нагрузки.



Для выбора формата читайте таблицу с командами и их описанием: https://disk.yandex.ru/i/PE8sTLiSGFJiVw.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

Схемотехник

День 1

В течение первого и второго дней необходимо разработать плату полезной нагрузки для проведения биоэксперимента.

Технические требования к плате полезной нагрузки для проведения биоэксперимента

Плата должна позволять реализовать считывание показаний с датчиков и контроль температуры и освещенности колбы с объектом эксперимента.

Для проведения эксперимента снимаются показания температурного датчика DS18B20, а также спектральные характеристики объекта эксперимента с помощью датчика цвета TCS34725.

Контроль температуры должен осуществляться посредством регулирования напряжения между двумя контактами, предназначенными для крепления нити накаливания, с помощью широтно-импульсной модуляции.

Контроль освещенности производится также с помощью широтно-импульсной модуляции посредством белых светодиодов.

Для реализации полезной нагрузки на практике используется макетная плата конструктора «IntroSat». Микроконтроллер и приемопередатчик в этом случае находятся снаружи (на макетной плате их размещать не нужно).

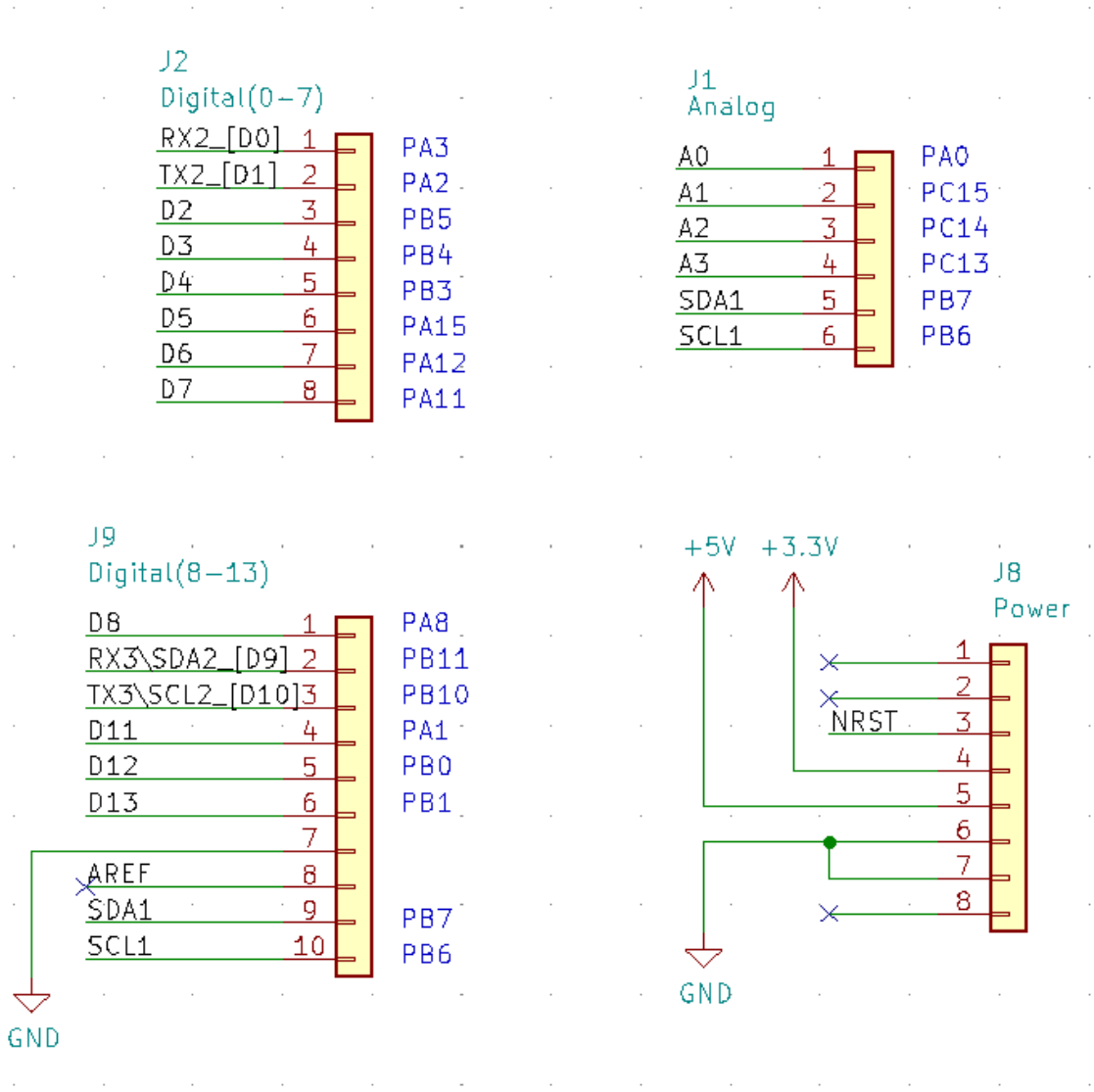
Задачей первого дня является составление в KiCAD (или аналоге, аналог согласовать с организаторами) электрической принципиальной схемы подключения полезной нагрузки для дальнейшей пайки на макетной плате. Также при выполнении схемы можно приступать к пайке при наличии времени.

Состав платы с экспериментом

1. Температурный датчик DS18B20.
2. Датчика цвета с ИК-фильтром TCS34725.
3. Белые светодиоды GNL-5053UWC.
   Управление освещенностью осуществляется с помощью ШИМ-сигнала.
4. Нихромовая нить.

Используется в качестве нагревательного элемента. Управление нагревом должно осуществляться с помощью ШИМ-сигнала, а ток, пропускаемый через нить, обеспечивается внешним источником напряжения 5 В. На схеме можно обозначить только место для подключения нихромовой нити в виде разъема.

На макетную плату приходит внешнее напряжение 3,3 В и 5 В. Макетная плата соединяется с материнской платой конструктора «IntroSat», на которой установлен микроконтроллер STM32F103C8. Выводы микроконтроллера, к которым можно получить доступ через макетную плату, представлены на рисунке.



На электрической принципиальной схеме также необходимо указать, с какими выводами микроконтроллера соединяется каждое устройство, входящее в полезную нагрузку. Если во встроенной библиотеке KiCAD отсутствует какой-либо компонент, его УГО необходимо создать самостоятельно в редакторе символов.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

В качестве материалов необходимо загрузить архив с проектом в KiCAD.

День 2

В течение дня необходимо:

- спаять прототип платы полезной нагрузки;
- доработать схему платы полезной нагрузки.

Создание прототипа

Необходимо, используя предоставленные компоненты, собрать с помощью паяльников на макетной плате прототип устройства. В результате работы прототип устрой-

ства должен выполнять весь ранее описанный функционал:

- регулирование освещенности светодиода посредством ШИМ;
- регулирование напряжения на нагревающем элементе посредством ШИМ;
- снятие показаний с датчика температуры;
- снятие показаний с датчика цвета.

Доработка схемы электрической принципиальной

Необходимо доработать электрическую принципиальную схему платы полезной нагрузки, добавив на нее следующие устройства:

1. Микроконтроллер STM32F103C8.

Необходимо подвести все необходимые управляющие сигналы от микроконтроллера к устройствам, а также подключить всю требующуюся для правильной работы микроконтроллера обвязку (обеспечить верную схему подключения питания, тактирование и т. д.).

Для выполнения миссии необходимо знать точное время, используя RTC модуль контроллера, вне зависимости от наличия бортового питания.

2. Заменить датчик цвета TCS34725 на более компактную версию TCS3472.

Питание (3,3 В и 5 В) остается внешним (приходит на специальный разъем).

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

В качестве материалов необходимо загрузить архив с проектом в KiCAD.

День 3

В течение дня необходимо произвести трассировку платы полезной нагрузки.

Требования к плате

- Размер платы полезной нагрузки не должен превышать 78×78 мм.
- Колба для размещения биокультуры — нетиповой компонент для KiCAD, поэтому необходимо создать для нее посадочное место в редакторе.

Габариты и форму колбы вы определяете самостоятельно, однако нельзя выходить за габариты платы. Минимальный объем колбы: 4000 мм^3; максимальная высота колбы: 6 мм.

- Источники излучения должны располагаться в непосредственной близости от колбы, таким образом, чтобы освещать ее.
- Датчик температуры должен располагаться в непосредственной близости от колбы, таким образом, чтобы измерять ее температуру.
- Датчик цвета должен располагаться в непосредственной близости от колбы, таким образом, чтобы определять спектр свечения.
- Контакты для подключения нихромовой нити должны располагаться в непосредственной близости от колбы, таким образом, чтобы производить ее нагрев.
- Необходимо создать посадочное место для подключения нихромовой нити, где место прохождения нити указано шелкографией
- Также может потребоваться создать посадочные места для других устройств,

отсутствующих в библиотеке.

Отчетность

Необходимо все наработки за день выгрузить в соответствующее событие на платформе https://orbita.education/.

В качестве материалов необходимо загрузить архив с проектом в KiCAD.

Физик-баллистик

День 1

Условие

Группа энтузиастов вывела на околоземные орбиты группировку из 4-х спутников. После отделения от ракеты-носителя каждый из спутников совершил небольшой маневр, чтобы разойтись по разным орбитам. Однако в ходе этого маневра связь с одним из спутников была потеряна. Между спутниками в группировке существует внутренняя связь. Когда потерянный спутник находится в зоне видимости трех остальных, то можно определить его местоположение в глобальной системе координат. Известно, что искомый спутник ближе к земле, чем спутники позиционирования.

Для решения задачи необходимо написать программу управления спутником, чтобы определить его местоположение. Продолжительность моделирования: 1000 секунд.

Ссылка на вебинар https://www.youtube.com/watch?v=CxW0AnmzfhA.

Ниже приведен шаблон с получением, преобразованием и отправкой данных.

```
// Размер пакета, приходящего с внешнего спутника:
↪  Float32Array([1,2,3,4]).byteLength +1
var plength = 4*4+1;
var positioning_system_bus;
var positioning_system_receiver;

function setup() {
  positioning_system_bus = spacecraft.devices[0].functions[0];
  positioning_system_receiver = spacecraft.devices[1].functions[0];
}

function loop() {
  // Прием данных от всех спутников
  let p1 = positioning_system_receiver.read(plength);
  let p2 = positioning_system_receiver.read(plength);
  let p3 = positioning_system_receiver.read(plength);

  // Проверка, что данные были в текущей итерации выполнения кода
  if (p1.length != plength || p2.length != plength || p3.length != plength) {
    return
  }

  // Преобразование Uint8Array в Float32Array для удобного использования данных
  ↪  со спутников
```

```
  let p = [new Float32Array(p1.slice(1).buffer), new
  ↪  Float32Array(p2.slice(1).buffer), new Float32Array(p3.slice(1).buffer)];

  // Элементы полученного массива выглядят как: p[0] = [X0, Y0, Z0, distance0]

  // Подготовка координат [0,1,2] к отправке
  var data = new Float32Array([0,1,2]);
  var byteView = new Uint8Array(data.buffer);
  // Отправка
  positioning_system_bus.transmit(byteView);
}
```

Описание входных параметров

Раз в 10 секунд в `positioning_system_receiver` приходит три массива `Uint8Array` длинной `Float32Array([x,y,z,d]).byteLength+1 = 4*4+1` (по массиву с каждого из трех спутников группировки), где в каждом массиве:

- первый байт — номер спутника,
- второй-пятый байты — x-координата типа `Float32Array` в системе ECEF,
- шестой-девятый байты — y-координата типа `Float32Array` в системе ECEF,
- десятый-тринадцатый байты — z-координата типа `Float32Array` в системе ECEF,
- четырнадцатый-семнадцатый байты — расстояние до спутника типа `Float32Array`.

Описание начисления очков

Правила начисления очков

Каждые 10 секунд начисляется

$$score = \frac{32,0/100}{r+1},$$

где r — евклидово расстояние от посчитанной точки до реальной.

Максимально возможный балл за задачу — 20.

Правила дисконтирования

Количество решений без штрафа — 1 на участника команды. За каждое последующее решение будет накладываться штраф 10% до убывания в 20%. Максимальная разница попыток между участниками команды — 3. При наличии ошибки в синтаксисе попытка не считается потраченной.

День 2

Условие

Группа энтузиастов вывела на околоземные орбиты группировку из 4-х спутников. После отделения от ракеты-носителя каждый из спутников совершил небольшой маневр, чтобы разойтись по разным орбитам. Однако в ходе этого маневра связь с одним из спутников была потеряна. Между спутниками в группировке существует внутренняя связь. Когда потерянный спутник находится в зоне видимости трех остальных, то можно определить его местоположение в глобальной системе координат.

Для решения задачи необходимо написать программу управления спутником, чтобы определить кеплеровы элементы орбиты аппарата по двум или более точек положения. Продолжительность моделирования: 1000 секунд.

Указание: для решения задачи можно считать, что орбита искомого аппарата круговая.

В результате работы программы необходимо передать на Землю через передатчик один раз значения кеплеровых элементов в формате: `Float32Array(["inclination", "semimajor-axis "ascending-node", "periapsis-argument "true-anomaly"])`.

Описание входных параметров

Раз в 10 секунд в `positioning_system_receiver` приходит три массива `Uint8Array` длинной `Float32Array([x,y,z,d]).byteLength+1 = 4*4+1` (по массиву с каждого из трех спутников группировки), где в каждом массиве:

- первый байт — номер спутника,
- второй-пятый байты — x-координата типа `Float32Array` в системе ECEF,
- шестой-девятый байты — y-координата типа `Float32Array` в системе ECEF,
- десятый-тринадцатый байты — z-координата типа `Float32Array` в системе ECEF,
- четырнадцатый-семнадцатый байты — расстояние до спутника типа `Float32Array`.

Описание начисления очков

Правила начисления очков

За каждый правильно посчитанный параметр начисляется по 6 баллов. Балл убывает на 50% за каждые 5% точности.

Максимально возможный балл за задачу — 30.

Правила дисконтирования

Количество решений без штрафа — 1 на участника команды. За каждое последующее решение будет накладываться штраф 10% до убывания в 20%. Максимальная разница попыток между участниками команды — 3. При наличии ошибки в синтаксисе попытка не считается потраченной.

День 3

Группа энтузиастов еще только выводит на околоземную орбиту группировку из 4-х спутников с позывными SC0, SC1, SC2 и SC3.

Пусковой контейнер выпустит аппараты в 16 марта 2023 года в 0 часов 0 минут 0 секунд UTC на орбиты, отличающиеся только истинной аномалией на 0.01 последовательно друг от друга (то есть они будут лететь последовательно друг за другом с небольшим отставанием).

Спутник SC0 является научным. Он должен быть оснащен, как минимум, оборудованием связи и устройством типа «Источник данных».

Три спутника связи SC1..SC3 имеют оборудование связи и могут иметь по одной (но не более) газореактивной двигательной установке производства ОКБ Факел.

Необходимо создать проект спутниковой группировки, включающий вышеуказанные аппараты и не более 3 наземных станций, расположенных в каких-либо населенных пунктах из списка: Вороново (в составе Новой Москвы), Санкт-Петербург,

Новосибирск, Благовещенск, Анадырь.

При этом:

1. Названия, количество, порядок и начальные орбиты аппаратов, также как и положение наземных станций, должны соответствовать условию задачи.
2. Состав группировки, параметры аппаратов и наземных станций, и прочие данные проекта должны соответствовать требованиям, описанным в разделе «Описание входных параметров».
3. Начальную ориентацию аппаратов и их вращение можно задать любые (для упрощения считать, что аппараты самостоятельно ориентируются сразу после выхода на орбиту за счет имеющихся на борту, но не моделируемых исполнительных устройств начального заряда дополнительного аккумулятора), после чего на время симуляции поддерживают постоянное вращение (или его отсутствие).
4. Обеспечить как можно больший совокупный период возможной связи научного аппарата с наземными станциями проекта. Связь возможна при соблюдении хотя бы одного из условий:
   - Научный аппарат находится в зоне видимости одной из станций;
   - Спутник связи находится в зоне видимости одной из станций и, одновременно, в зоне видимости научного аппарата.
5. (опционально) Используя не более половины топлива в двигателе на каждом из спутников связи, добиться максимального попарного расстояния между аппаратами (между SC0 и SC1, между SC1 и SC2 и между SC2 и SC3) на 17 марта 2023 года в 0 часов 0 минут 0 секунд.

При необходимости усилить свое решение, можно задать программу управления одному или нескольким аппаратам. Краткая справка по API приведена, например, здесь: https://orbita.education/materials/188/2604.

(Обратите внимание: чтобы двигатель начал работу, нужно не только включить устройство, но и установить значение тяги)

Описание входных параметров

Общие данные симуляции:

- Время начала симуляции: 16 марта 2023 года 0:00:00 UTC
- Время окончания симуляции: 17 марта 2023 года 00:00:01 UTC
- Шаг моделирования: 0,1 с
- Воздействие гравитации Солнца и Луны: Да
- Модель гравитации Земли: pz-90–11, точечные массы.
- Модель атмосферы: есть, простая
- Модель вращения Земли: Iau-2006-2000a
- Прецессию Земли считать отсутствующей.

Состав группировки и ее наземного сегмента:

- Один научный аппарат (должен называться SC0 и идти первым в списке)
- Три аппарата ретрансляции (должны называться SC1, SC2 и SC3 и идти после научного аппарата в порядке номеров)
- Не более трех наземных станций, каждая из которых расположена в каких-либо населенных пунктах из списка: Вороново (в составе Новой Москвы), Санкт-

Петербург, Новосибирск, Благовещенск, Анадырь. Станцию размещать в условном центре города (приведены ниже), с точностью до 4 знака после запятой.

Общие требования к аппаратам:

- Для простоты все спутники считать шарообразными и равномерными по массе, а все их устройства — размещенными в центре. Ориентаций устройств может быть произвольной.
- Однако размер и масса аппарата не должны превышать нормативов для формата CubeSat 3U (не более 34 см в длину и не более 4 кг вес)
- Все функции «Передатчик» должны иметь имя переменной Transmitter
- Все функции «Приемник» должны иметь имя переменной Reciever

Требование к оборудованию научного аппарата SC0:

- Аппарат оснащен полезной нагрузкой типа Источник данных, с именем BIOLAB, потребляющей не менее 5 Вт. и работающей все время.
- Аппарат оснащен передатчиком типа Передатчик с именем TRN, потребляющей не менее 5 Вт, когда включен.
- Аппарат может использовать только солнечные батареи совокупной площадью не более 1200 см^2 с КПД не более 30%, а также аккумуляторы начальным совокупной зарядом не более 50000 Дж.

Требование к оборудованию и программе управления спутников связи SC1..SC3:

- Наличие приемника и передатчика.
- Оборудование для обеспечения энергобаланса — на свое усмотрение (допускается нереалистичное, в предположении, что энергобалансом этих аппаратов занимаетесь не вы).
- Не более одной газореактивной двигательной установки производства ОКБ Факел — устройство из Базы Данных `GRDU_Fakel`.

Любой аппарат группировки может передавать данные на наземную станцию или другой аппарат группировки, если выполнены все условия:

- Он находится в зоне прямой видимости
- Но не далее, чем в 1000 км от наземной станции (или другого аппарата соответственно),
- У аппарата включен Передатчик (и Приемник, если это спутник связи).

Допустимые для размещения наземных станций населенные пункты.

Город	**Координаты условного центра**
Вороново (Москва)	55,320888, 37,158425
Санкт-Петербург	59,935848, 30,305252
Новосибирск	55,016684, 82,923375
Благовещенск	50,249266, 127,553278
Анадырь	64,735814, 177,518913

Описание начисления очков

Решения оцениваются экспертами по достигнутым показателям.

Количество решений не ограничено. Дисконта за количество попыток нет.

По умолчанию **оценивается только последнее отправленное командой на расчет решение**, завершенное без ошибок и сбоев. До завершения рабочего времени

или в течение 10 минут после его завершения команда может предложить жюри оценить вместо этого другое отправленное решение, но только одно.

Система оценивания

Общее описание

Комплексная задача финала декомпозируется на 4 направления и также декомпозируется по дням. За решение ряда подзадач в каждом из 4-х направлений можно набрать одинаковое количество командных баллов. Все задачи постепенно сливаются в одну комплексную.

В декомпозиции по дням количество баллов распределено следующим образом:

- день 1: максимум 20 баллов за выполнение подзадач в направлении;
- день 2: максимум 30 баллов за выполнение подзадач в направлении;
- день 3: максимум 50 баллов за выполнение подзадач в направлении.

Суммарно команда по дням может заработать 80, 120 и 150 баллов соответственно в течение первого, второго и третьего дней. Максимальный балл за командную часть — 400. Это значение затем нормируется вместе с учетом предметной индивидуальной части. Максимальный балл за задачи определенного дня можно заработать только в течение соответствующего дня. На следующий день задачи предыдущего дня можно досдать с дисконтом в 50%. В третьем дне досдача задач первого дня недоступна.

Если команды испытывают трудности в решении первостепенных задач (задач первого дня), организаторами может быть выдана часть авторского решения после закрытия возможности их сдачи.

В рекомендованном составе команды 4 человека (допустимо 3) и 4 направления работы. Как правило, участник команды отвечает за выбранное направление, однако возможны смена ролей и передача подзадач внутри направления друг другу.

Ниже приведены критерии для каждого направления в соответствии с разделением подзадач по дням.

Критерии оценки для роли Программист МК

Критерий оценки	**Макс. балл**
1. Задачи первого дня	**20**
1.1. Реализован обмен данными по UART и беспроводная прошивка	2.5
1.2. Реализовано считывание показаний с датчика позиционирования и вывод данных в UART	2.5
1.3. Реализовано вращение спутника с помощью маховика	2.5
1.4. Реализовано вращение спутника с помощью маховика	2.5
1.5. Осуществлен расчет и намотка катушек	10
2. Задачи второго дня	**30**
2.1. Реализовано автоматическое вращение спутника с помощью катушек	10
2.2. Реализована разгрузка маховика с использованием магнитных катушек	20
3. Задачи третьего дня	**50**

Критерий оценки	**Макс. балл**
3.1. Реализован алгоритм стабилизации с помощью магнитных катушек (алгоритм B-dot)	10
3.2. Произведена интеграцию радиомодуля и полезной нагрузки в сборку спутника	5
3.3. Циклограмма спутника содержит последовательное выполнение следующих подзадач:	
• реализована стабилизация с помощью магнитных катушек	5
• реализованы нагрев и освещение емкости, расположенной в полезной нагрузке	5
• реализован сбор данных с датчиков полезной нагрузки для последующего формирования пакета	10
• отправка пакета на Orbicraft 3D	5
• все этапы циклограммы выполняются корректно в одной программе в правильной последовательности	10

Критерии оценки для роли Схемотехник

Критерий оценки	**Макс. балл**
1. Задача первого дня: итоговая схема устройства разработана и работоспособна	**20**
2. Задачи второго дня	**30**
2.1. Спаянное устройство полностью работоспособно	12
2.2. Качество пайки (отсутствуют «сопли», проводные соединения надежны и пр.)	6
2.3. Обновленная схема работоспособна и не содержит грубых ошибок	12
3. Задачи третьего дня	**50**
3.1. Все компоненты размещены и соединены на печатной плате	20
3.2. Соблюдены правила трассировки печатных плат	20
3.3. Корректно созданы посадочные места для нетиповых компонентов	10

Критерии оценки для роли Инженер связи

Критерий оценки	**Макс. балл**
1. Задача первого дня	**20**
1.1. Осуществлена передача с научного спутника (IntroSat) на спутник-ретранслятор (Orbicraft 3D)	10
1.2. Реализовано управление маховиком на спутнике-ретрансляторе (Orbicraft 3D)	10
2. Задачи второго дня	**30**
2.1. Осуществлена передача данных с IntroSat по УКВ-сигналу Orbicraft	10

Критерий оценки	Макс. балл
2.2. Построенный граф выводит пакет AX.25 на блок Message Debug PDU Vectors	20
3. Задачи третьего дня	**50**
3.1. Полноценная сборка графа декодирует данные до первоначального вида в режиме реального времени	15
3.2. Пакет данных сформирован согласно требованиям, содержит всю указанную информацию. Пакет успешно ретранслируется на SDR-приемник в режиме реального времени	30
3.3. Ретрансляция пакета данных успешно интегрирована в циклограмму спутника	5

Критерии оценки для роли баллистик

Критерий оценки	Макс. балл
1. Задача первого дня: каждые 10 секунд симуляции начисляется 0,3 балла при разнице между реальным и расчетным положением не более 1 м	**20**
2. Задача второго дня: за каждый правильно посчитанный параметр орбиты начисляется по 6 баллов	**30**
3. Задача третьего дня	**50**
3.1. Совокупное количество пакетов научных данных, переданных на любую из наземных станций проекта, не менее 500	10
3.2. Лучший среди всех команд ненулевой результат совокупного количества пакетов научных данных, переданных на любую из наземных станций проекта (каждый следующий в порядке убывания в вышеуказанном рейтинге не нулевой результат команды — на 1.5 балла меньше предыдущего)	20
3.3. Расстояние между парами аппаратов (SC0 и SC1, SC1 и SC2, SC2 и SC3) к концу симуляции более 100 000 м	2 за каждую пару
3.4. Расстояние между парами аппаратов (SC0 и SC1, SC1 и SC2, SC2 и SC3) за время симуляции более 100 000 м, но не более 1 000 000 м (дополнительно)	2 за каждую пару
3.5. Аккумулятор научного аппарата имел начальный заряд не более 100 000 Дж и не был разряжен за время симуляции	4
3.6. Дополнительно, если передано не менее 500 пакетов, а начальный заряд аккумулятора научного аппарата не превышал 50 000 Дж.	4

Решение задачи

Программист МК

День 1

Код управления

В соответствующие блоки необходимо добавить следующий код.

```
/* USER CODE BEGIN Includes */
#include <AK8963.h>
#include <CoilFlyWheel.h>
#include <Gyroscope.h>
```

```
#include <MotorFlyWheel.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

#include "IS_Bluetooth.h"

/* USER CODE END Includes */

/* USER CODE BEGIN 2 */
using namespace IntroSatLib;

MotorFlyWheel motor(&hi2c1, 0x38);
CoilFlyWheel coil(&hi2c1, 0x38);

AK8963 mag(&hi2c1, 0x0C);
Gyroscope mpu1(&hi2c1, 0x68);

while (motor.Init()) {}
HAL_Delay(250);
while (coil.Init()) {}
HAL_Delay(250);

while (mpu1.Init()) {}
HAL_Delay(250);

while (mag.Init()) {}
HAL_Delay(250);

int16_t speed = 32000;
char str;
int16_t mx = 0;
int16_t my = 0;
// int16_t mz = 0;

/* USER CODE END 2 */

while (1) {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  HAL_UART_Receive(&huart1, (uint8_t *)&str, 1, 1000);
  if (str == 'b') {
    str = '\n';
    HAL_UART_Transmit(&huart1, (uint8_t *)"Bootloader Mode\n\r",
                      strlen("Bootloader Mode\n\r"), 1000);
    enter_bootloader();
  }

  motor.NeedSpeed(3000);
  HAL_Delay(1000);
  motor.NeedSpeed(0);
  HAL_Delay(1000);
  motor.NeedSpeed(-3000);
  HAL_Delay(1000);
  motor.NeedSpeed(0);
  HAL_Delay(1000);
  coil.NeedSpeed(speed);

  mag.Read();
```

```
    mx = mag.RawX();
    my = mag.RawY();
    HAL_Delay(100);

    char buf[20];

    sprintf(buf, "MagX: %d \t", mx);
    HAL_UART_Transmit(&huart1, (uint8_t *)buf, strlen(buf), 1000);

    sprintf(buf, "MagY: %d \n\r", my);
    HAL_UART_Transmit(&huart1, (uint8_t *)buf, strlen(buf), 1000);
  }
  /* USER CODE END 3 */
```

Данный пример кода подробно разбирается в видео-инструкции по удаленному подключению: https://youtu.be/zBBec1O_W_A.

Расчет катушек

1. Магнитная индукция, создаваемая кольцами Гельмгольца

Индукция в центре колец Гельмгольца при условии, что расстояние между кольцами равно их радиусу, может быть рассчитана по формуле:

$$B = \mu_0 \left(\frac{2}{\sqrt{5}}\right)^3 \frac{IN}{R},$$

где $\mu_0 = 1,25664 \cdot 10^{-6}$ Гн/м — коэффициент, магнитная постоянная, к этой величине можно относиться как к константе;

$I = 1,35$ — сила тока, подаваемая на кольца, отображается на экране (с нагревом колец сила тока будет уменьшаться);

$N = 120$ — количество витков на кольцах; в данной конструкции 12 слоев, в каждом слое по 10 проводов, с помощью платы соединяемых последовательно и образующих витки;

$R = 0,15$ м — радиус колец.

Подставив числа, получим, что $B = 0,00097$ Тл.

2. Момент сил, действующий на катушки спутника

Момент сил, действующий на рамку с n витков проводника с током I и имеющей площадь $S_{\text{кат}}$, плоскость которой наклонена на угол α к вектору напряженности B постоянного магнитного поля, вычисляется по формуле (используются единицы измерения СИ):

$$M = B \cdot I \cdot S_{\text{кат}} \cdot n \cdot \sin(\alpha).$$

На катушки подается напряжение в 5 В, а сопротивление двух катушек, подключенных параллельно, не должно быть меньше 10 Ом, чтобы обеспечить работоспособность платы. Можно принять данное сопротивление цепи с катушками как типовое и отсюда определить примерное значение силы тока на катушках: $I = 5/10 = 0,5$ А.

Для расчета площади катушек для простоты можно принять, что их форма соответствует кругу. Радиус катушек равен 3,9 см или 0,039 м. Тогда для площади круга:

$$S_{\text{кат}} = \pi \cdot R^2 = \pi \cdot 0,039^2 = 0,00478 \text{ м}^2.$$

С помощью своих витков катушки должны создавать такой момент, чтобы:

- придать спутнику на подвесе необходимый момент вращения;
- преодолеть момент трения подвеса.

3. Момент вращения

Момент вращения твердого тела вокруг оси определяется как произведение момента инерции тела J на угловое ускорение ε:

$$M_{\text{вр}} = J \cdot \varepsilon.$$

С некоторым приближением можно предположить, что момент инерции определяется для спутника как для куба:

$$J = \frac{1}{6} m a^2,$$

где $m = 0,781$ кг — масса спутника;

$a = 10$ см $= 0,1$ м — сторона куба.

Таким образом, момент инерции спутника $J = 0,0013$ кг/м^2.

Угловое ускорение будет меняться в зависимости от положения спутника относительно вектора магнитной индукции. Когда катушки будут вставать параллельно кольцам, синус угла между вектором магнитной индукции и силой тока в катушке будет равен нулю. А следовательно, в таком положении катушки не будут создавать момента сил для вращения спутника. Но если спутник будет иметь небольшое ускорение на малом угле, то сможет провернуться через зону, где катушками не создается момент вращения по инерции.

Исходя из вышеописанных соображений, для упрощения расчета мы можем сами предположить значения углового ускорения ε и угла поворота α, которые были бы достаточными для того, чтобы на спутник почти всегда действовал момент сил, создаваемый катушками спутника. Итак, угол, при котором должен создаваться момент, должен быть относительно небольшим, предположим, что $\alpha = 10^\circ$. При этом угле ε также должна быть отлична от нуля. Пусть $\varepsilon = 0,1^\circ/\text{с}^2$.

4. Момент сопротивления подвеса

Одним из моментов, который противодействует вращению спутников, является момент трения подшипника в подвесе. Этот момент зависит от веса спутника и характеристик самого подшипника. Осевую нагрузку на подшипник можно определить по формуле:

$$M_{\text{под}} = f \cdot F \cdot d/2,$$

где $f = 0,025$ — коэффициент подшипника;

$F = mg$ — вес подвешенного спутника;

$d = 0,004$ м — диаметр вала подшипника.

5. Итоговое количество витков

Момент, создаваемый катушками, должен быть равен сумме момента вращения, который мы хотим придать спутнику, и момента трения подшипника подвеса:

$$M = M_{\text{вр}} + M_{\text{под}}.$$

$$B \cdot I \cdot S_{\text{кат}} \cdot n \cdot \sin(\alpha) = J \cdot \varepsilon + mg \cdot f \cdot d/2.$$

Отсюда мы можем определить необходимое количество витков n:

$$n = \frac{J \cdot \varepsilon + mg \cdot f \cdot d/2}{B \cdot I \cdot S_{\text{кат}} \cdot \sin(\alpha)}.$$

Подставив значения в формулу, можем определить, что $n = 957$. При расчете важно перевести градусы в радианы, а полученное значение округлить до целых в большую сторону.

Это общее количество витков для двух катушек.

Тогда для одной катушки: $957/2 = 479$ витков.

День 2

В соответствующие блоки необходимо добавить следующий код.

```
/* USER CODE BEGIN Includes */
#include <AK8963.h>
#include <CoilFlyWheel.h>
#include <Gyroscope.h>
#include <MotorFlyWheel.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

#include "IS_Bluetooth.h"

/* USER CODE END Includes */

/* USER CODE BEGIN 2 */
using namespace IntroSatLib;

MotorFlyWheel motor(&hi2c1, 0x38);
CoilFlyWheel coil(&hi2c1, 0x38);

AK8963 mag(&hi2c1, 0x0C);
Gyroscope mpu1(&hi2c1, 0x68);

while (motor.Init()) {
}
HAL_Delay(250);
while (coil.Init()) {
}
HAL_Delay(250);

while (mpu1.Init()) {
}
HAL_Delay(250);

while (mag.Init()) {
}
HAL_Delay(250);

int16_t speed = 3000;
char str;
int16_t mx = 0;
int16_t my = 0;
int16_t gz = 0;
```

```
// int16_t mz = 0;

// Начальная закрутка маховика
motor.NeedSpeed(speed);
HAL_Delay(30000);

/* USER CODE END 2 */

while (1) {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
  HAL_UART_Receive(&huart1, (uint8_t *)&str, 1, 1000);
  if (str == 'b') {
    str = '\n';
    HAL_UART_Transmit(&huart1, (uint8_t *)"Bootloader Mode\n\r",
                      strlen("Bootloader Mode\n\r"), 1000);
    enter_bootloader();
  }

  // Включение катушек
  coil.NeedSpeed(32000);

  // Чтение данных угловой скорости по вертикальной оси
  gz = mpu1.RawX();

  // Сброс скорости маховика, если угловая скорость соответствует состоянию
  // покоя
  if (gz < 50) {
    if (speed > 40) speed -= 40;
    motor.NeedSpeed(speed);
  }
}
```

День 3

```
/* USER CODE BEGIN Includes */
#include <AK8963.h>
#include <Accelerometer.h>
#include <CoilFlyWheel.h>
#include <DS18B20.h>
#include <Gyroscope.h>
#include <MotorFlyWheel.h>
#include <TCS34725.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

#include "IS_Bluetooth.h"
#include "NRF24.h"
/* USER CODE END Includes */

/* USER CODE BEGIN PV */
char str1[20] = {0};
uint8_t buf1[20] = {0};

// Разбиваем данные на два пакета и задаем структуры, соответствующие
// кодификатору
struct __attribute__((packed)) data_pack1 {
  float data1;
```

```
  float temp;
  uint16_t flight_time;
  int16_t gyrX;
  int16_t gyrY;
  int16_t gyrZ;
  char str[20];
};

struct __attribute__((packed)) data_pack2 {
  float data1;
  float data2;
  uint16_t Col1;
  uint16_t Col2;
  uint16_t Col3;
  uint16_t Col4;
  char str[20];
};

/* USER CODE END PV */

/* USER CODE BEGIN 2 */
using namespace IntroSatLib;

MotorFlyWheel motor(&hi2c1, 0x38);
CoilFlyWheel coil(&hi2c1, 0x38);

AK8963 mag(&hi2c1, 0x0C);
Gyroscope mpu1(&hi2c1, 0x68);

while (motor.Init()) {
}
HAL_Delay(250);
while (coil.Init()) {
}
HAL_Delay(250);

while (mpu1.Init()) {
}
HAL_Delay(250);

while (mag.Init()) {
}
HAL_Delay(250);

DS18B20Init(&huart2);
TCS34725Init(&hi2c1);

int16_t speed = 3000;
char str;

int16_t mx = 0;
int16_t my = 0;
int16_t mz = 0;

int16_t my_prev = 0;

int16_t gx = 0;
int16_t gy = 0;
int16_t gz = 0;

```

```
85  int16_t t;
86  int16_t t_prev = 0;
87
88  // Начальная закрутка маховика
89  motor.NeedSpeed(speed);
90  HAL_Delay(30000);
91  motor.NeedSpeed(0);
92
93  NRF24_ini();
94  NRF24_WriteReg(RF_CH, 76);
95
96  NRF24_PacketLen(32);
97
98  /* USER CODE END 2 */
99
100 while (1) {
101   /* USER CODE END WHILE */
102   /* USER CODE BEGIN 3 */
103   HAL_UART_Receive(&huart1, (uint8_t *)&str, 1, 1000);
104   if (str == 'b') {
105     str = '\n';
106     HAL_UART_Transmit(&huart1, (uint8_t *)"Bootloader Mode\n\r",
107                       strlen("Bootloader Mode\n\r"), 1000);
108     enter_bootloader();
109   }
110
111   mag.Read();
112   mx = mag.RawX();
113   my = mag.RawY();
114
115   /*           B-dot                */
116   t = HAL_GetTick();
117   int dt = t - t_prev;
118   int16_t speed_1 = ceil(-2000 * (my - my_prev) / dt);
119
120   if (speed_1 > 32000)
121     speed_1 = 32000;
122   else if (speed_1 < -32000)
123     speed_1 = -32000;
124
125   coil.NeedSpeed(speed_1);
126   HAL_Delay(100);
127
128   my_prev = my;
129   /*           B-dot                */
130
131   // Считываем данные с гироскопа
132   gx = mpu.RawX();
133   gy = mpu.RawY();
134   gz = mpu.RawZ();
135
136   // Считываем данные с датчика цвета
137   uint16_t r, g, b, c;
138   r = TCS34725Red();
139   g = TCS34725Green();
140   b = TCS34725Blue();
141   c = TCS34725Clear();
142
143   // Считываем данные с датчика температуры
144   float a = DS18B20GetTemperature();
```

```
145
146     struct data_pack1 pack1;
147     struct data_pack2 pack2;
148     uint8_t data[32];
149
150     // Заполняем поля структуры данными
151     pack1.flight_time = t;
152     pack1.gyrX = gx;
153     pack1.gyrY = gy;
154     pack1.gyrZ = gz;
155     pack1.temp = a;
156
157     pack2.Col1 = r;
158     pack2.Col2 = g;
159     pack2.Col3 = b;
160     pack2.Col4 = c;
161
162     // Выбираем код подходящего формата данных по кодификатору
163     uint16_t msgId = 0xDA01;
164
165     // Заполняем первые элементы пакета данных в соответствии с требованиями
166     data[0] = sizeof(data);
167     data[1] = 0x1;
168     data[2] = msgId >> 8;
169     data[3] = msgId & 0xFF;
170
171     // Заполняем два первых элемента данных в соответствии с типами структуры
172     data[4] = 0;
173     data[5] = *((uint8_t *)&pack1.temp);
174
175     // Заполняем следующие 8 бит данных, преобразуя данные побайтово
176     data[6] = pack1.flight_time >> 8;
177     data[7] = pack1.flight_time & 0xFF;
178     data[8] = pack1.gyrX >> 8;
179     data[9] = pack1.gyrX & 0xFF;
180     data[10] = pack1.gyrY >> 8;
181     data[11] = pack1.gyrY & 0xFF;
182     data[12] = pack1.gyrZ >> 8;
183     data[13] = pack1.gyrZ & 0xFF;
184
185     // Добавляем к формируемому пакету название команды
186     strcat(pack1.str, "My Team");
187     for (int i = 0; i < strlen(pack1.str); i++) {
188       data[i + 14] = *((uint8_t *)&pack1.str[i]);
189     }
190
191     // Отправляем сформированный пакет данных
192     NRF24L01_Send(data);
193
194     // Обновляем данные для передачи второго пакета данных
195     data[5] = 0;
196     data[6] = pack2.Col1 >> 8;
197     data[7] = pack2.Col1 & 0xFF;
198     data[8] = pack2.Col2 >> 8;
199     data[9] = pack2.Col2 & 0xFF;
200     data[10] = pack2.Col3 >> 8;
201     data[11] = pack2.Col3 & 0xFF;
202     data[12] = pack2.Col4 >> 8;
203     data[13] = pack2.Col4 & 0xFF;
204
```

```
205   // Отправляем второй пакет данных
206   NRF24L01_Send(data);
207
208   HAL_Delay(2000);
209 }
```

Инженер связи

День 1

В соответствующие блоки необходимо добавить следующий код.

```
1  /* USER CODE BEGIN Includes */
2  #include "NRF24.h"
3  #include "stdio.h"
4  /* USER CODE END Includes */
5
6  /* USER CODE BEGIN PV */
7  char str1[20] = {0};
8  uint8_t buf1[20] = {0};
9
10 struct __attribute__((packed)) data_rpm {
11   float rpm;
12 };
13
14 /* USER CODE END PV */
15
16 /* USER CODE BEGIN 2 */
17
18 NRF24_ini();
19 NRF24_WriteReg(RF_CH, 76);
20
21 struct data_rpm d_data_rpm;  // зададим переменную, соответствующую заданной
22                              // выше структуре данных
23 uint8_t data[32];  // зададим массив для данных заданного размера
24
25 // Определим значение элементов структуры
26 d_data_rpm.rpm = 2000.0;  // Число в качестве примера
27
28 uint16_t msgId = 0xA01;  // Зададим переменную команды, которую хотим отправить
29                          // на Орбикрафт 3D
30 data[0] = sizeof(data);  // Определим размер передаваемых данных и поместим
31                          // значение в первый элемент массива
32 data[1] = 0xB;  // Зададим адрес устройства Орбикрафт 3D, которому хотим
33                 // отправить команду Преобразуем в 2 байта значение команды
34 data[3] = msgId & 0xFF;
35 data[2] = msgId >> 8;
36
37 // Запишем в массив указатели на данные структуры
38 for (int i = 0; i < sizeof(d_data_rpm); i++) {
39   data[i + 4] = *((uint8_t *)&d_data_rpm + i);
40 }
41 NRF24_PacketLen(32);  // Зададим в настройках передатчика длину пакета данных,
42                       // который хотим отправить
43 /* USER CODE END 2 */
44
45 /* Infinite loop */
46 /* USER CODE BEGIN WHILE */
47 while (1) {
```

```
48    /* USER CODE END WHILE */
49
50    /* USER CODE BEGIN 3 */
51    HAL_Delay(2000);
52    NRF24L01_Send(data);
53  }
54  /* USER CODE END 3 */
```

День 2

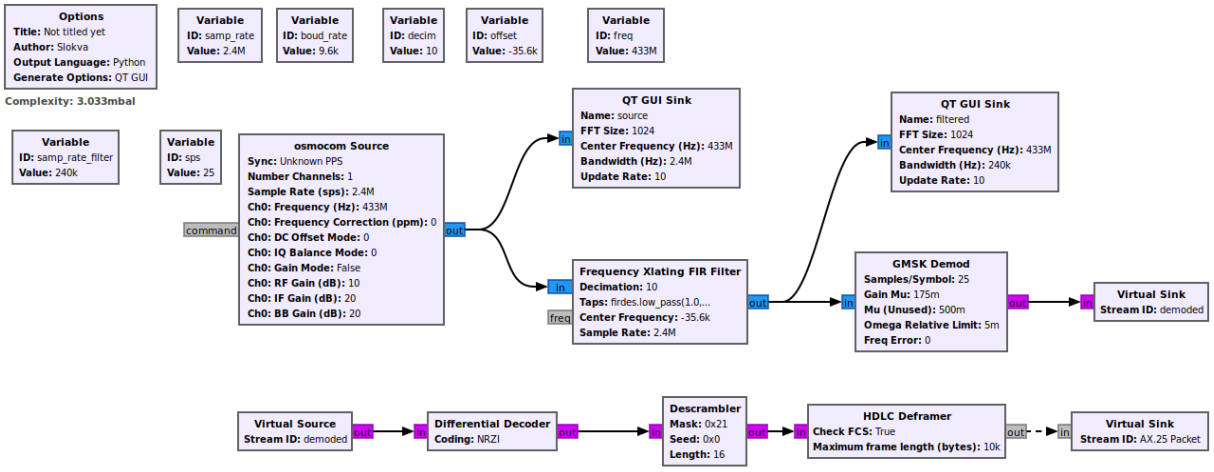
Код для управления передатчиком

В соответствующие блоки необходимо добавить следующий код.

```
1   /* USER CODE BEGIN Includes */
2   #include "NRF24.h"
3   #include "stdio.h"
4   /* USER CODE END Includes */
5
6   /* USER CODE BEGIN PV */
7   char str1[20] = {0};
8   uint8_t buf1[20] = {0};
9
10  struct __attribute__((packed)) data_float {
11    float data1;
12    float null0;
13    float null1;
14    float null2;
15  };
16  /* USER CODE END PV */
17
18  /* USER CODE BEGIN 2 */
19
20  NRF24_ini();
21  NRF24_WriteReg(RF_CH, 76);
22
23  struct data_float d_data_float;
24  uint8_t data[sizeof(d_data_float) + 3 + 1];
25
26  d_data_float.data1 = HAL_GetTick();
27  d_data_float.null0 = 0;
28  d_data_float.null1 = 0;
29  d_data_float.null2 = 0;
30
31  uint16_t msgId = 0xDA11;
32  data[0] = sizeof(data);
33  data[1] = 0x1;
34  data[3] = msgId & 0xFF;
35  data[2] = msgId >> 8;
36
37  for (int i = 0; i < sizeof(d_data_float); i++) {
38    data[i + 4] = *((uint8_t *)&d_data_float + i);
39  }
40
41  NRF24_PacketLen(32);
42
43  /* USER CODE END 2 */
44
45  /* Infinite loop */
46  /* USER CODE BEGIN WHILE */
```

```
47  while (1) {
48    /* USER CODE END WHILE */
49
50    /* USER CODE BEGIN 3 */
51    HAL_Delay(2000);
52    NRF24L01_Send(data);
53  }
54  /* USER CODE END 3 */
```

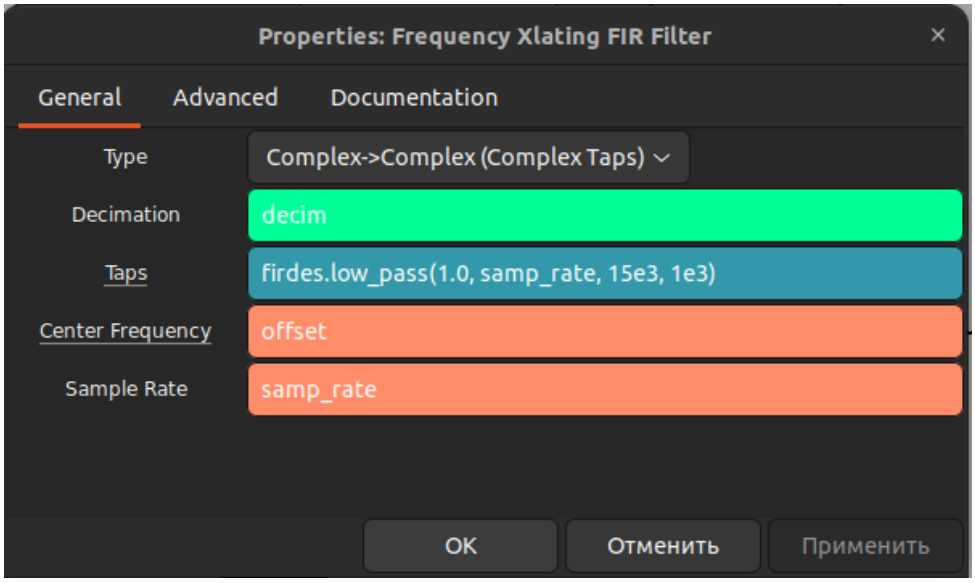
Граф для декодирования сигнала



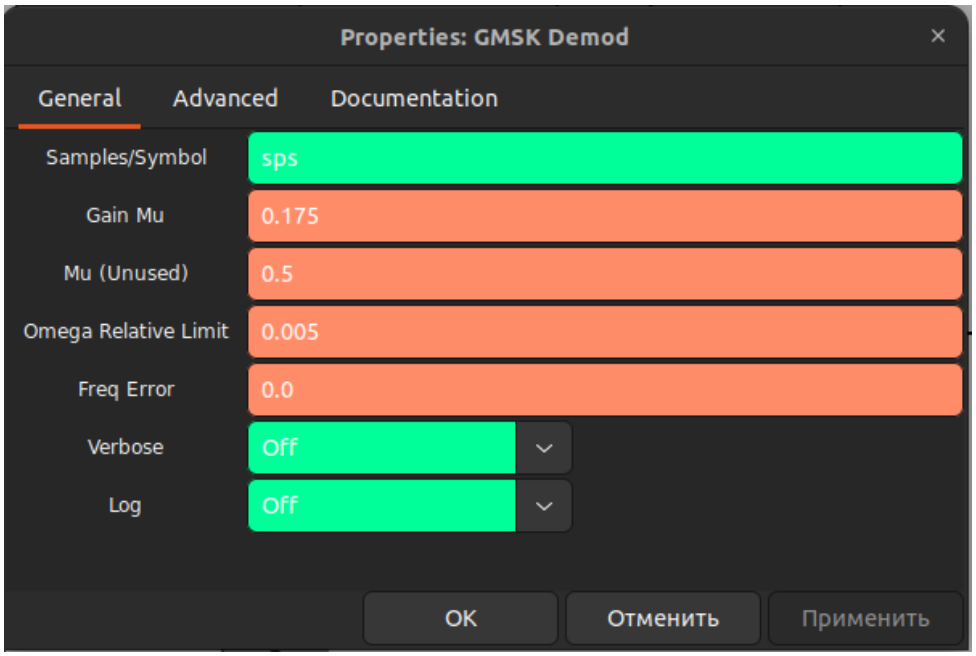
Рекомендуемое значение количества отсчетов (`samp_rate`): 2400000. Данное значение кратно скорости передачи, равной 9600, что облегчает последующее декодирование сигнала. Помимо этого, данное количество отсчетов достаточно велико, чтобы избежать потерь полезной информации в сигнале.

В качестве источника сигнала в графе рекомендуется использовать `osmocom Source`, так как он поддерживает настройку необходимого значения `samp_rate`.

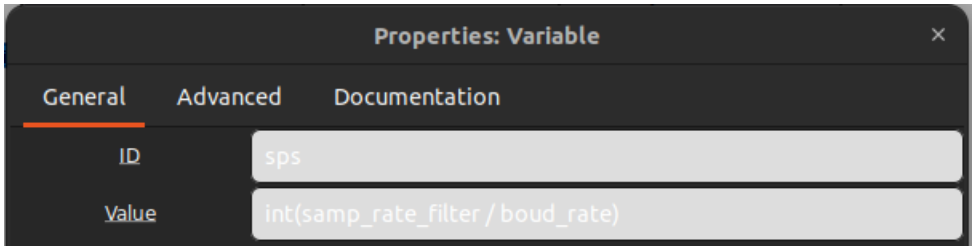
После получения сигнала необходимо провести его фильтрацию. Для этого может подойти несколько фильтрующих блоков. В примере используется один из них: Frequency FIR Filter. В фильтре используются следующие настройки (`decim = 10`).

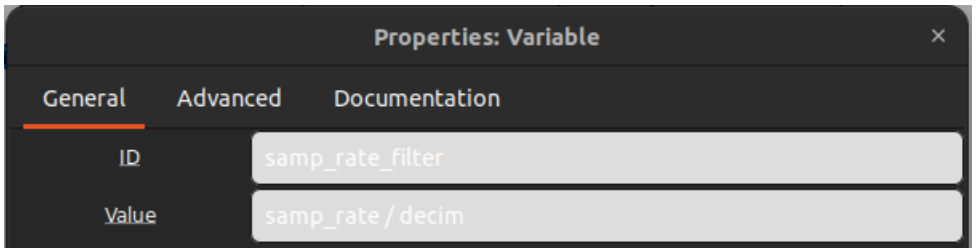


После фильтрации сигнала идет этап демодуляции. Так как модуляция сигнала — GMSK, то можно воспользоваться соответствующим блоком GMSK Demod.

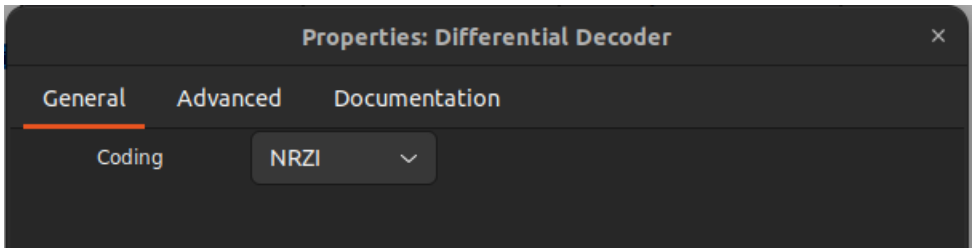


В качестве первого параметра данного блока используется рассчитанное значение сэмплов, приходящихся на символ.

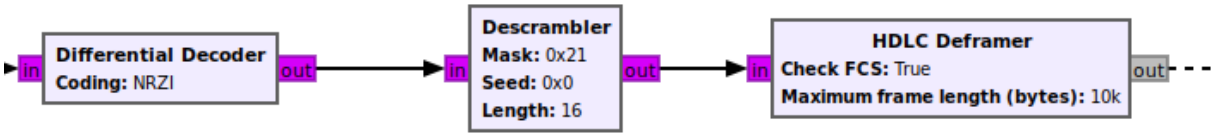




На выходе с демодулирующего блока сигнал будет представлен в виде последовательности бит. Так как используется протокол AX.25, то последовательность бит будет представлена в виде NRZI (более подробно можно прочитать по ссылке: https://notblackmagic.com/bitsnpieces/ax.25/). Для представления последовательности бит в исходном порядке следует воспользоваться соответствующим декодером.



Так как помимо использования NRZI также используется скрэмблирование, то при декодировании необходимо провести обратную операцию, используя блок Descrambler. На выходе данного блока получится фрэйм HDLC, из которого можно извлечь пакет протокола AX.25, используя блок HDLC Deframer.



Конечным результатом работы представленного графа является недекодированный пакет протокола AX.25.

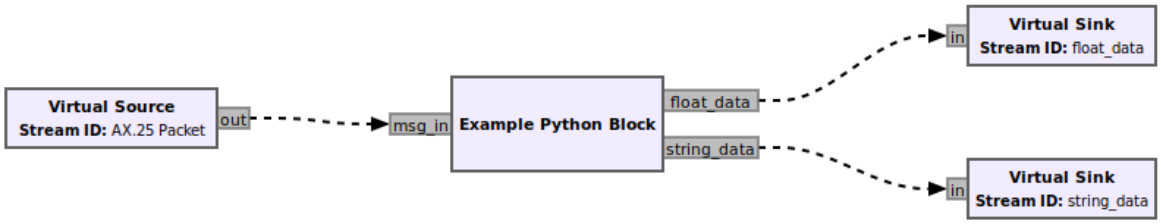
День 3

Код для управления передатчиком

Часть кода, относящаяся к формированию пакета, приведена в решении по направлению Программист МК для третьего дня.

Граф для декодирования сигнала

К представленному в решении второго дня графу необходимо добавить несколько блоков.



Для обработки пакета, собираемого участниками самостоятельно в коде для передатчика, логично воспользоваться Python-блоком, в котором можно будет выдавать данные в необходимом формате.

Пример кода для Python-блока.

```
1  """
2  Embedded Python Blocks:
3
4  Each time this file is saved, GRC will instantiate the first class it finds
```

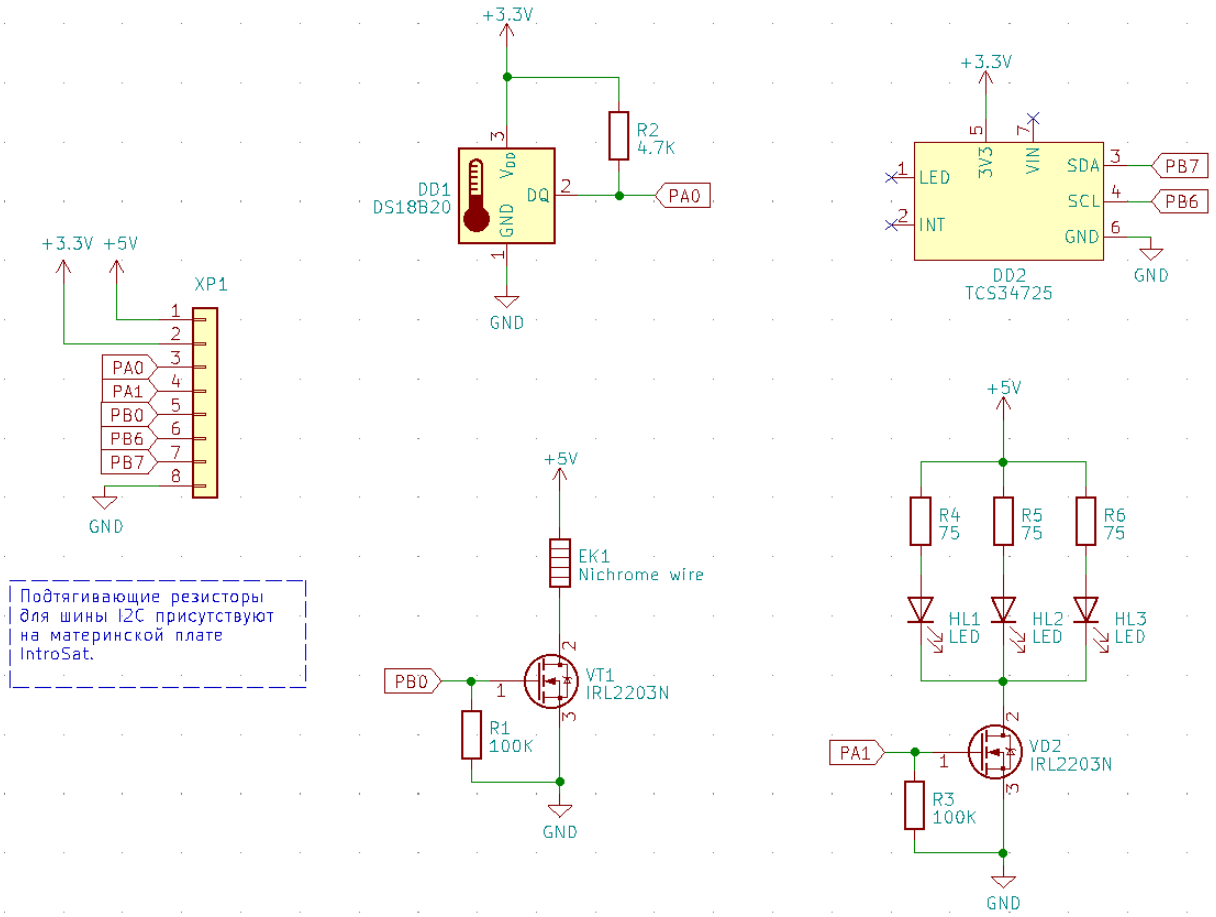
```
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

from gnuradio import gr
import struct
import pmt


class blk(gr.basic_block):  # other base classes are basic_block, decim_block,
    interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.basic_block.__init__(
            self,
            name='Example Python Block',   # will show up in GRC
            in_sig=None,
            out_sig=None
        )

        self.message_port_register_out(pmt.intern('string_data'))
        self.message_port_register_out(pmt.intern('float_data'))
        self.message_port_register_in(pmt.intern('msg_in'))
        self.set_msg_handler(pmt.intern('msg_in'), self.handle_msg)

    def handle_msg(self, msg):
         msg_decoded = bytes(pmt.to_python(msg)[1])
         ret = self.proceed(msg_decoded)
         resultFloat = pmt.PMT_NIL
         for i in ret:
            if (type(ret[i]) == str):
                self.message_port_pub(pmt.intern("string_data"),
                 pmt.cons(pmt.intern(i), pmt.intern(ret[i])))
            else:
                resultFloat = pmt.list_add(resultFloat, pmt.list2(pmt.intern(i),
                 pmt.make_f32vector(1, ret[i])))
         self.message_port_pub(pmt.intern('float_data'), resultFloat)

    def proceed(self, data):
        #TODO: Здесь должен быть ваш код
        rez = dict()
        callsigns = bytearray()
        for i in data:
            callsigns.append(i >> 1)
            if (i & 1):
                break
        other = data[len(callsigns):]
        rez["dst"] = str(callsigns[:6], "ASCII").strip()
        rez["dstSSID"] = str(callsigns[6] & 0x0F)
        rez["type"] = "command" if (callsigns[6] >> 6) else "response"
        rez["src"] = str(callsigns[7:7+6], "ASCII").strip()
        rez["srcSSID"] = str(callsigns[7+6] & 0x0F)
        rez["control"] = hex(other[0])
        rez["PID"] = hex(other[1])

        CANPacket = other[2:]
        CANunpacked = struct.unpack(f"HBxBxH{len(CANPacket) - 8}s",
         bytes(CANPacket))
```

```
61          rez["MsgId"], rez["srcCAN"], rez["dstCAN"] = list(map(hex,
            ↪  CANunpacked[:-2]))
62          msgId = CANunpacked[0]
63          rez["lenCAN"] = str(CANunpacked[-2])
64
65
66          value = CANunpacked[-1]
67          if (msgId == 0xDA11):
68              rez["gx"], rez["gy"], rez["gz"] = struct.unpack("3f4x", value)
69          elif (msgId == 0xDA01):
70              rez["time"], teamName = struct.unpack("H20s", value)
71              rez["teamName"] = str(teamName, "cp1251")
72          elif (msgId == 0xDA12):
73              rez["r"], rez["g"], rez["b"], rez["t"] = struct.unpack("4f", value)
74
75          return rez
```

Схемотехник

День 1

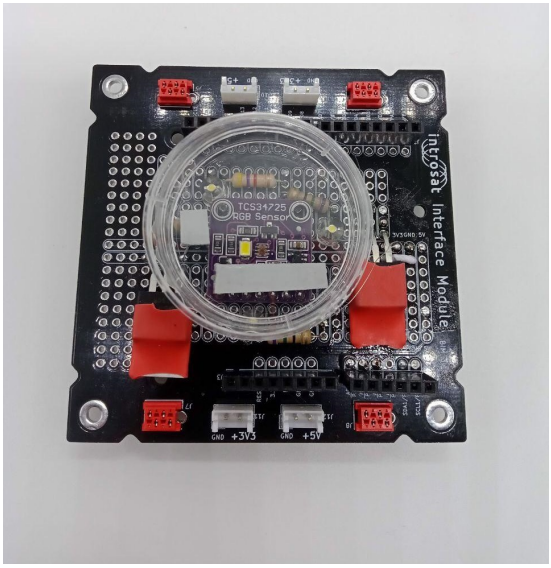
Электрическая принципиальная схема выполнена в KiCAD. Условное графическое обозначение (УГО) датчика цвета отсутствует в стандартной библиотеке KiCAD, поэтому оно было создано вручную. Температурный датчик подключается к микроконтроллеру по интерфейсу 1-wire, причем шину данных необходимо подтянуть к питанию через резистор 4,7 кОм. В качестве УГО для нихромовой нити выбран нагревательный элемент. Нихромовая нить подключается к питанию 5В через полевой транзистор n-типа в ключевом режиме, чтобы можно было управлять нагревом. Светодиоды требуют напряжения питания 3,5В и тока 20 мА на каждый, поэтому их нельзя соединять с выводами микроконтроллера — они также подключены через транзистор и ограничительные резисторы. Сопротивление резисторов вычислено по закону Ома. Затвор транзисторов подтянут через резисторы 100 кОм на землю во избежание случайного срабатывания ключа. Каждый вывод, который должен быть подключен к микроконтроллеру, создан на схеме в виде глобальной метки с обозначением, соответствующим названию вывода микроконтроллера. На схеме обозначен разъем, к которому также подключаются эти глобальные метки, что сигнализирует о том, что будет осуществляться соединение с внешним устройством. На этот разъем также приходят линии питания.

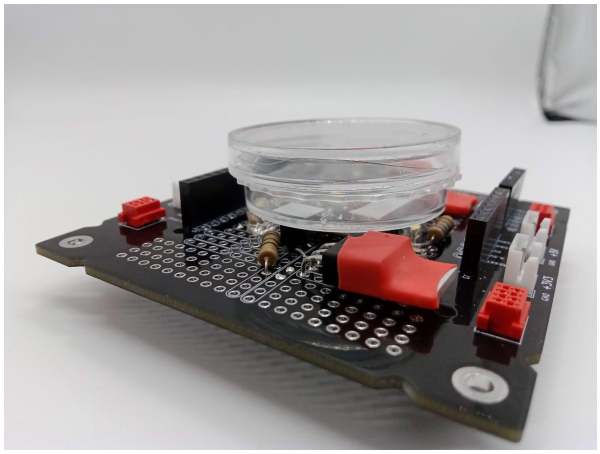


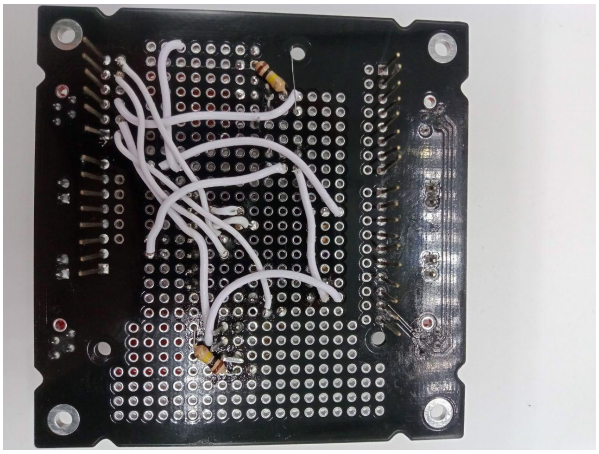
День 2

Создание прототипа

Датчик цвета необходимо расположить под колбой, чтобы можно было определить цвет жидкости. Светодиоды расположены под 120 градусов по периметру колбы под ней для равномерного освещения. Датчик температуры также расположен под колбой и касается ее — также допустимо располагать его снаружи, но вплотную к колбе. Корпуса транзисторов обернуты термоусадкой, чтобы не произошло короткое замыкание при касании металлической части транзистора и платы прототипирования. Нихромовая нить опоясывает колбу.

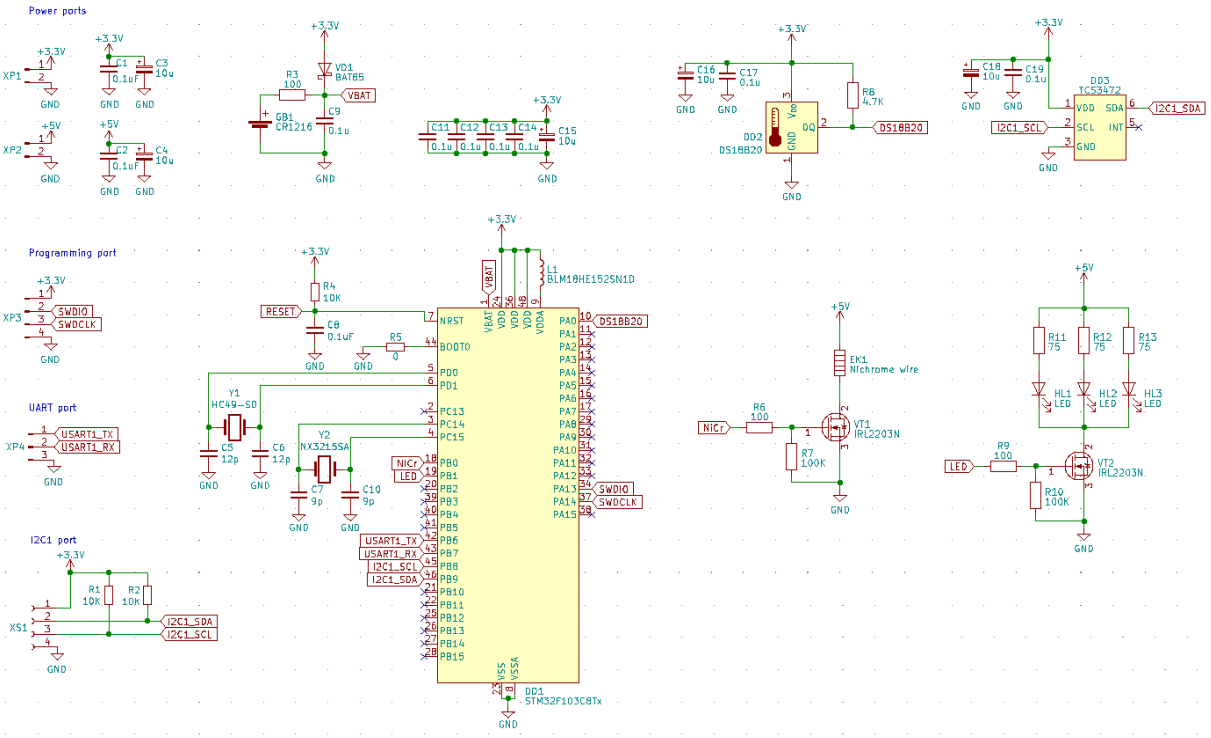






Доработка схемы

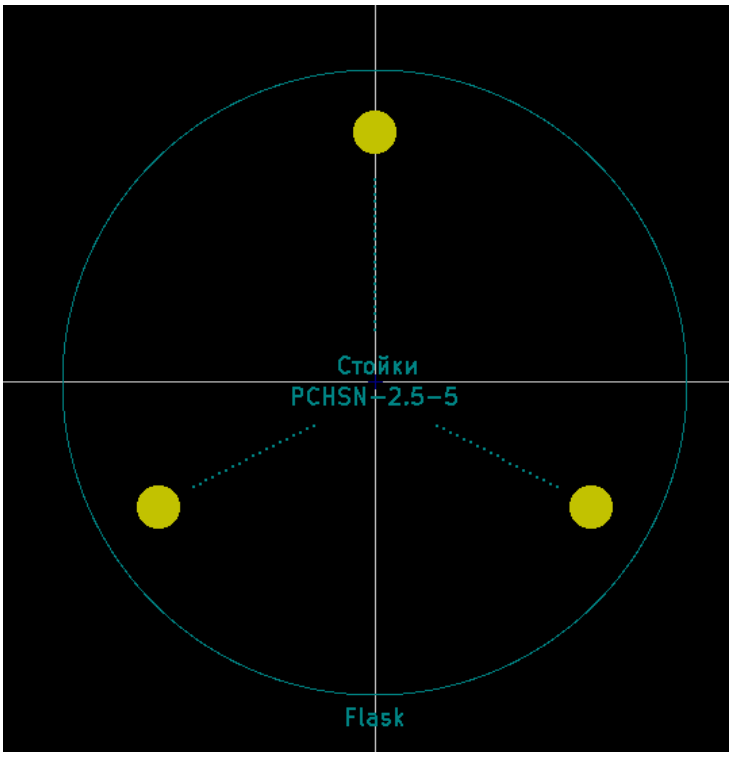
На схему добавлены микроконтроллер, разъемы и обвязка для всех устройств. Датчик цвета заменен на компактную версию TCS3472, для него создано новое УГО, так как он имеет другой набор выводов. К датчику температуры, датчику цвета, микроконтроллеру и разъемам питания подключены фильтрующие конденсаторы по питанию (керамический 0,1 мкФ для устранения высокочастотных помех и танталовый 10 мкФ — для низкочастотных). К затворам транзисторов добавлены резисторы 100 Ом для ограничения тока. Осуществлена возможность работы внутренних RTC часов микроконтроллера при сбое внешнего питания через батарейку CR1216. К микроконтроллеру также подключены внешние высокочастотный и низкочастотные резонаторы; вывод NRST подтянут к плюсу питания (на него необходимо подать минус питания, чтобы осуществить сброс микроконтроллера). Созданы три разъема для обмена данными с микроконтроллером: порт программирования, UART порт и I2C порт. Шины интерфейса I2C подтянуты через резисторы 10 кОм к плюсу питания.



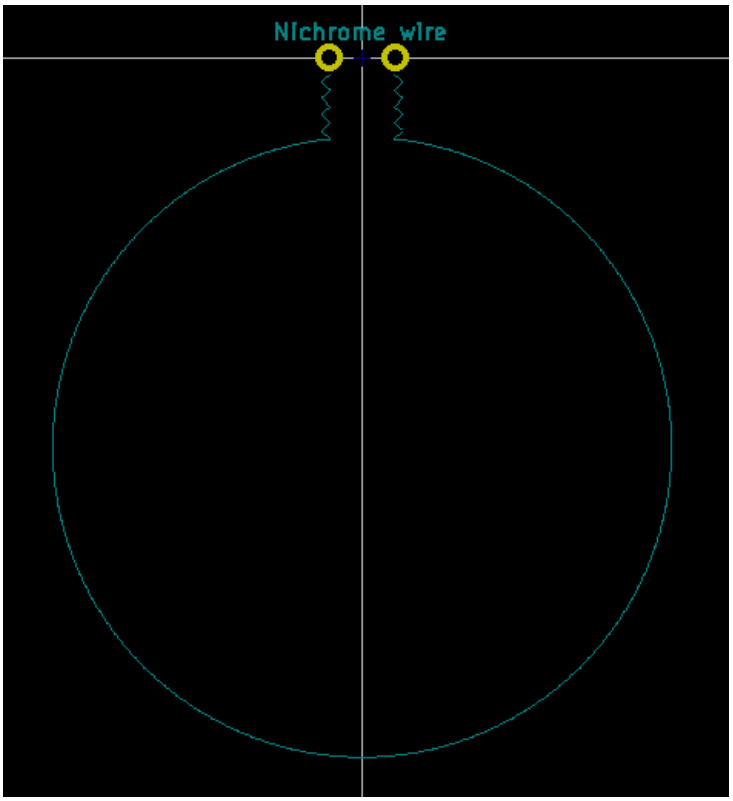
День 3

Созданы посадочные места под нетиповые компоненты схемы:

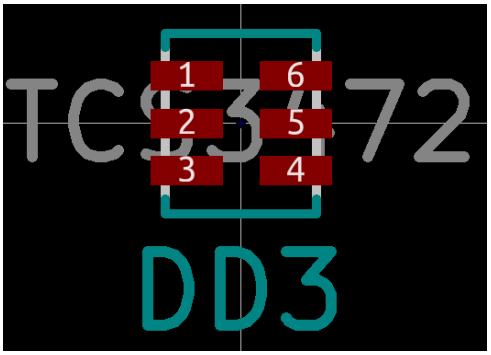
1. Колба с экспериментом: предусмотрены монтажные отверстия для установки колбы на латунные стойки PCHSN-2.5-5.



2. Нихромовая нить: опоясывает колбу изнутри. Два вывода для подключения к 5 В и стоку управляющего транзистора.



3. Датчик цвета: посадочное место создано в соответствие с datasheet на него.

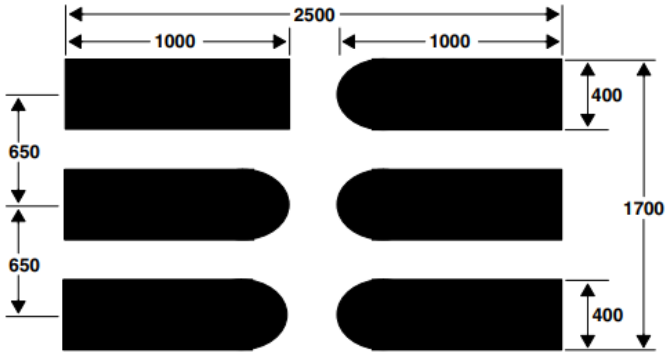


APPLICATION INFORMATION: HARDWARE

PCB Pad Layout

Suggested PCB pad layout guidelines for the Dual Flat No-Lead (FN) surface mount package are shown in Figure 11.

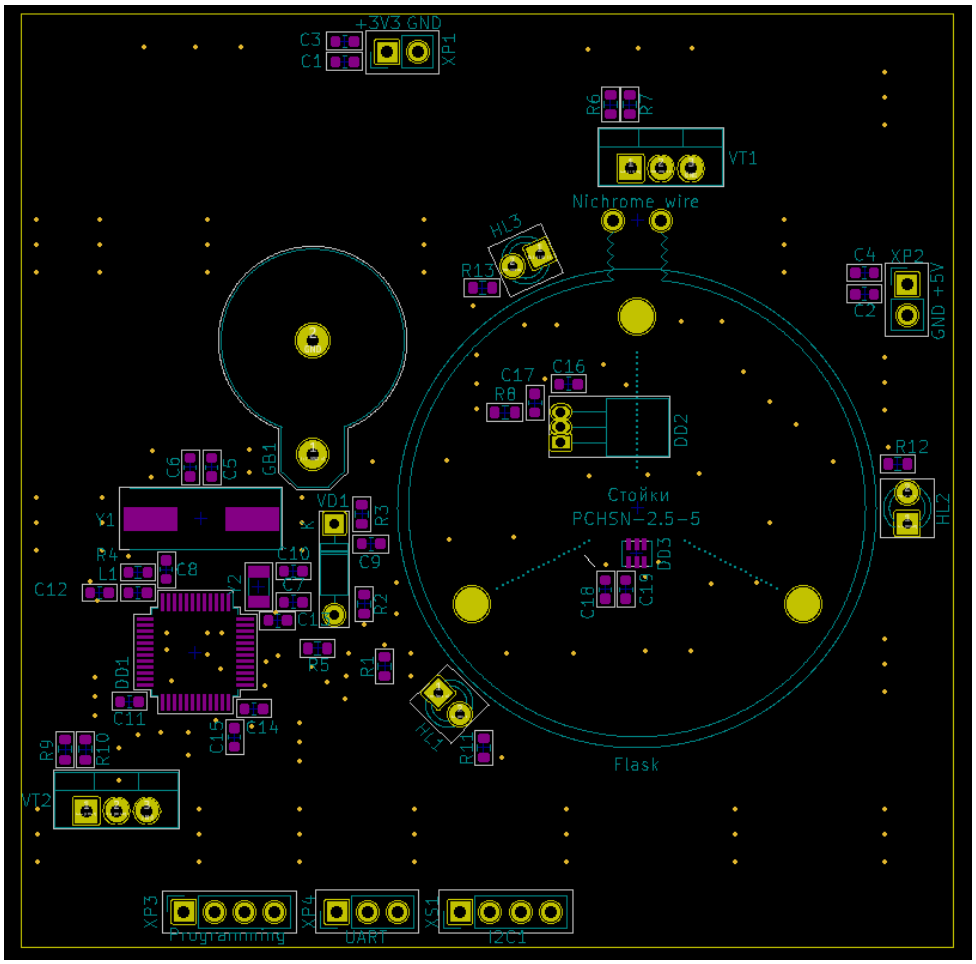
Note: Pads can be extended further if hand soldering is needed.

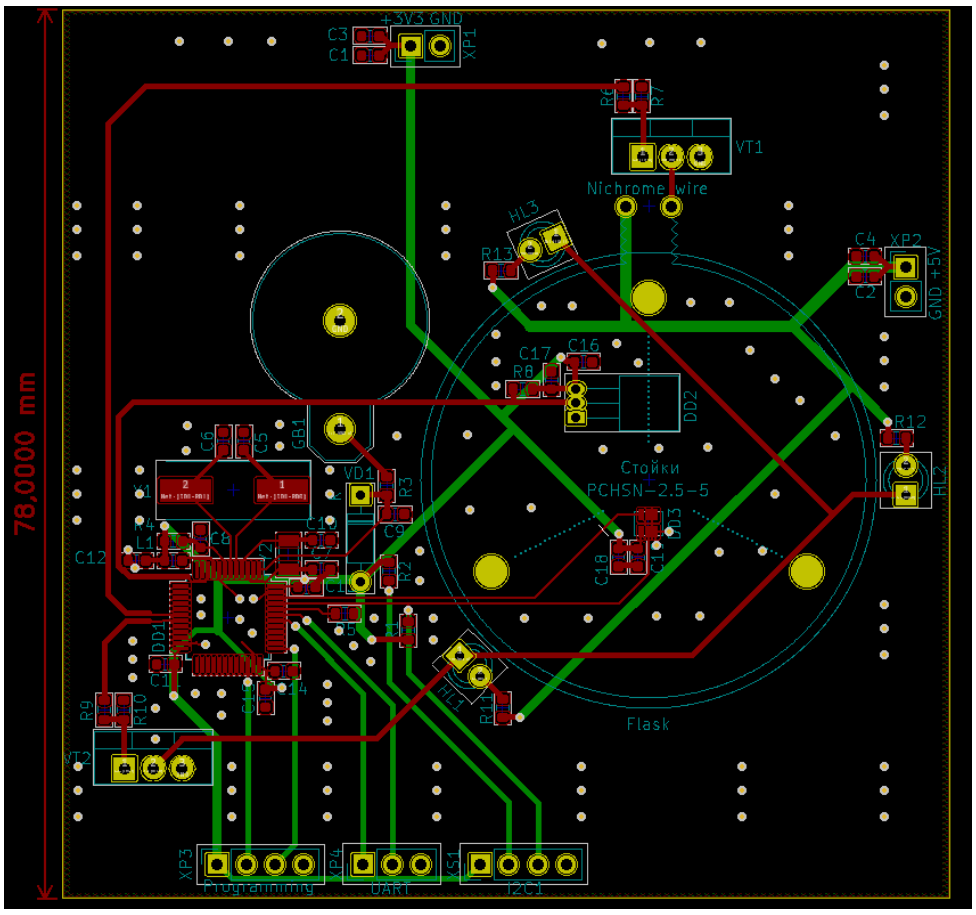


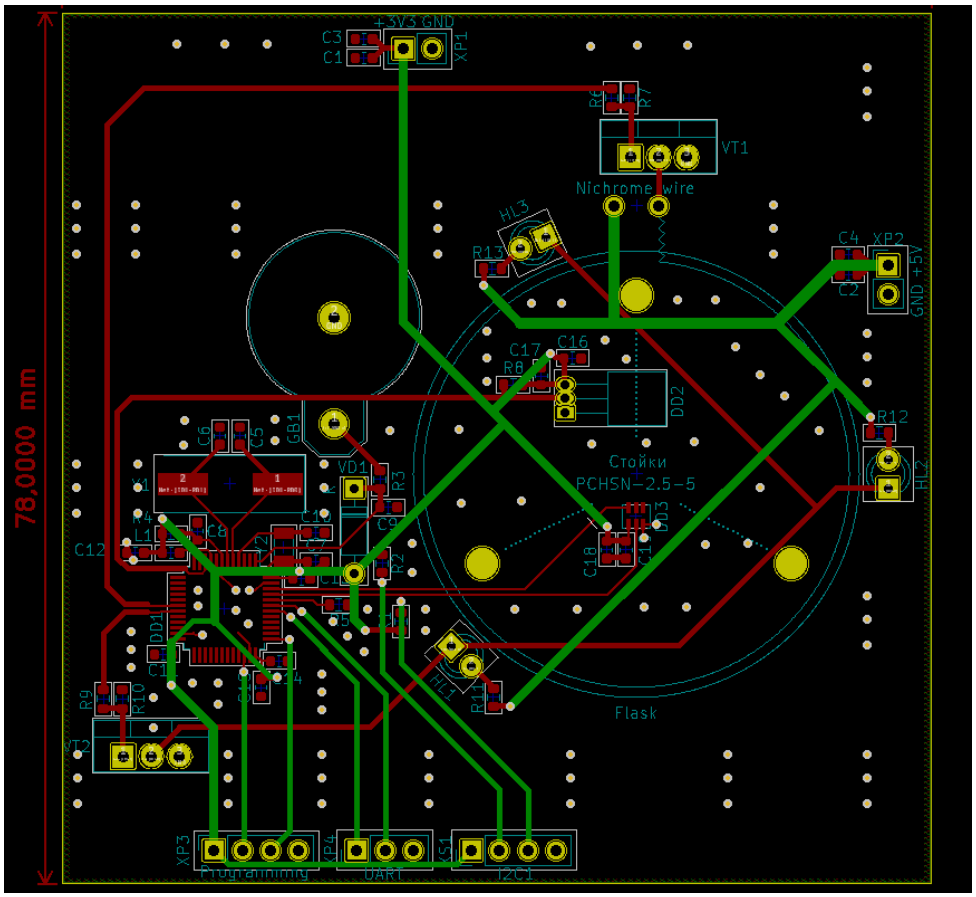
NOTES: A. All linear dimensions are in micrometers.
B. This drawing is subject to change without notice.

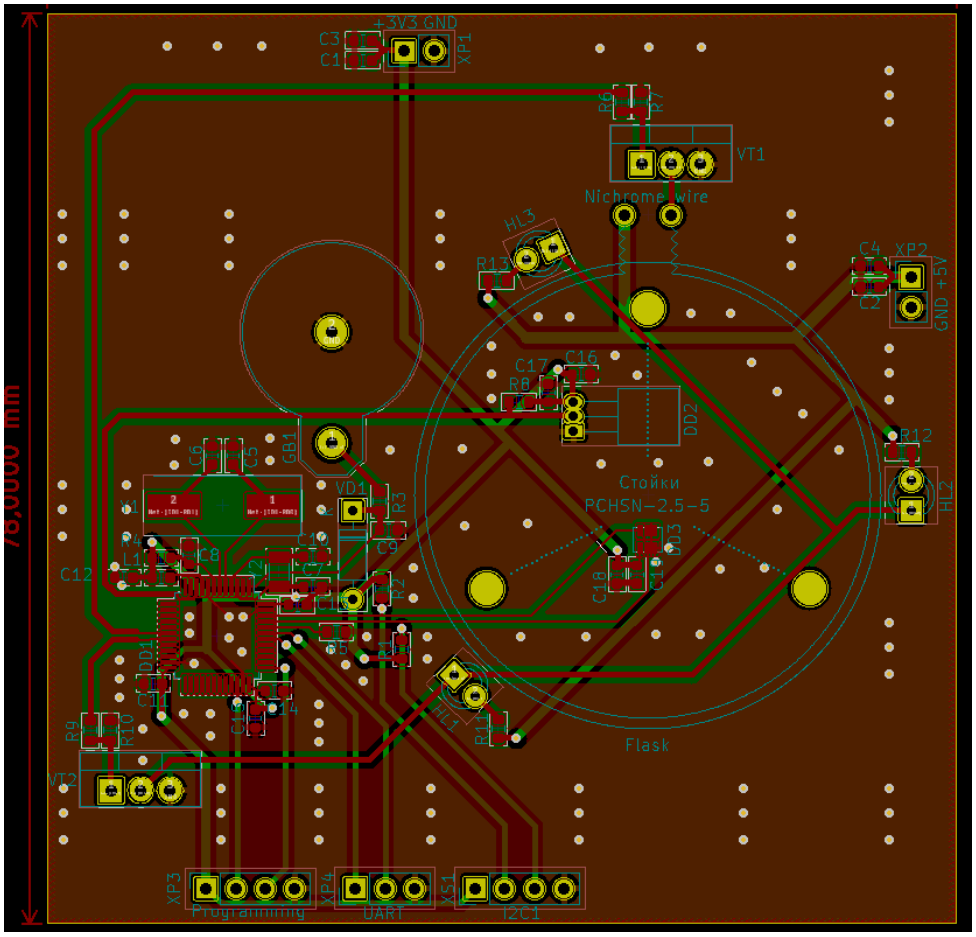
Figure 11. Suggested FN Package PCB Layout

Трассировка платы осуществлена в KiCAD. Датчик цвета и датчик температуры расположены под колбой, нихромовая нить по периметру внутри колбы, светодиоды — по периметру снаружи, под 120 градусов для равномерного освещения. При трассировки учтены основные правила трассировки, такие, как: ширина дорожек меньше ширины контактных площадок для хорошего механического крепления; длинные дорожки по возможности частично сделаны толще, чтобы уменьшить сопротивление; силовые дорожки также широкие. Питание разведено «звездочкой» — не допускаются петли. Блокировочные конденсаторы расположены близко к микросхемам. Подключение конденсаторов к резонатором симметрично, и они расположены близко к микроконтроллеру. Созданы земляные полигоны на верхнем и нижнем слоях платы; они соединены через дополнительные переходные отверстия.


+3V3 GND
XP1
C3
C1
R6
R7
VT1
Nichrome wire
HL3
R13
C4
XP2
C2
GND +5V
C17
C16
R8
DD2
R12
HL2
Стойки
PCH5N-2.5-5
C6
C5
GB1
VD1
R3
Y1
C10
C9
R4
L1
C8
C12
C7
C1
R2
DD3
C18
C19
R5
R1
DD1
C11
C15
C14
HL1
R11
R9
R10
VT2
Flask
XP3
Programming
XP4
UART
XS1
I2C1


78,0000 mm
+3V3 GND
XP1
C3
C1
R6
R7
VT1
Nichrome wire
HL3
R13
C4
XP2
C2
GND +5V
C17
C16
R8
DD2
R12
HL2
Стойки
PCH5N-2.5-5
C6
C5
GB1
VD1
R3
Y1
C10
C9
R4
L1
C8
C12
C7
C1
R2
DD3
C18
C19
R5
R1
DD1
C11
C15
C14
HL1
R11
R9
R10
VT2
Flask
XP3
Programming
XP4
UART
XS1
I2C1


+3V3 GND
XP1
VT1
Nichrome wire
HL3
R13
C4
XP2
GND +5V
C16
C17
R8
DD2
R12
HL2
Стойки
PCHSN-2.5-5
GB1
VD1
R3
C9
C12
C18
DD3
R2
HL1
R14
Flask
R9
R10
XS2
XP3
Programming
XP4
UART
XS1
I2C1
78,0000 mm


+3V3 GND
XP1
VT1
Nichrome wire
HL3
R13
C4
XP2
GND +5V
C16
C17
R8
DD2
R12
HL2
Стойки
PCHSN-2.5-5
GB1
VD1
R3
C9
C12
C18
DD3
R2
HL1
R14
Flask
R9
R10
XS2
XP3
Programming
XP4
UART
XS1
I2C1
78,0000 mm

Баллистик

День 1

```
// Функция расчета местоположения методом трилатерации
function trilateration(P1, P2, P3, r1, r2, r3) {
  // Приведение центра координат к центру одной из окружностей
  const p1 = [0, 0, 0];
  const p2 = [P2[0] - P1[0], P2[1] - P1[1], P2[2] - P1[2]];
  const p3 = [P3[0] - P1[0], P3[1] - P1[1], P3[2] - P1[2]];
  const v1 = p2.map((x, i) => x - p1[i]);
  const v2 = p3.map((x, i) => x - p1[i]);

 // Вектор расстояния между первым и вторым спутником делим на его длину
  const Xn = v1.map((x) => x / Math.sqrt(v1.reduce((sum, xi) => sum + xi ** 2,
  ↪  0)));

 // Векторное произведение векторов v1 и v2
  const tmp = [
    v1[1] * v2[2] - v1[2] * v2[1],
    v1[2] * v2[0] - v1[0] * v2[2],
    v1[0] * v2[1] - v1[1] * v2[0],
  ];

 // Полученное векторное произведение делим на его длину
  const Zn = tmp.map((x) => x / Math.sqrt(tmp.reduce((sum, xi) => sum + xi ** 2,
  ↪  0)));

 // Векторное произведение векторов Xn и Zn
  const Yn = [
    Xn[1] * Zn[2] - Xn[2] * Zn[1],
    Xn[2] * Zn[0] - Xn[0] * Zn[2],
    Xn[0] * Zn[1] - Xn[1] * Zn[0],
  ];

 // Векторы нормали для перехода в новую систему координат
  const i = Xn.reduce((sum, xi, j) => sum + xi * v2[j], 0);
  const d = Xn.reduce((sum, xi, j) => sum + xi * v1[j], 0);
  const j = Yn.reduce((sum, yi, k) => sum + yi * v2[k], 0);

 // Вычисление координат
  const X = ((r1 ** 2) - (r2 ** 2) + (d ** 2)) / (2 * d);
  const Y = (((r1 ** 2) - (r3 ** 2) + (i ** 2) + (j ** 2)) / (2 * j)) - ((i / j) *
  ↪  X);
  const Z1 = Math.sqrt(Math.max(0, r1 ** 2 - X ** 2 - Y ** 2));
  const Z2 = -Z1;

  const K1 = [P1[0] + X * Xn[0] + Y * Yn[0] + Z1 * Zn[0],
    P1[1] + X * Xn[1] + Y * Yn[1] + Z1 * Zn[1],
    P1[2] + X * Xn[2] + Y * Yn[2] + Z1 * Zn[2]];
  const K2 = [P1[0] + X * Xn[0] + Y * Yn[0] + Z2 * Zn[0],
    P1[1] + X * Xn[1] + Y * Yn[1] + Z2 * Zn[1],
    P1[2] + X * Xn[2] + Y * Yn[2] + Z2 * Zn[2]];

  return [K1, K2];
}

var transmitter;
var producer;

```

```
var plength = 0

// Инициализация устройств аппарата согласно API и вспомогательных переменных
function setup() {
  var tmp = new Float32Array([0,0,0,0]);
  plength = tmp.byteLength +1;
  transmitter = spacecraft.devices[0].functions[0];
  producer = spacecraft.devices[1].functions[0];
}


var errors = 0
function loop() {
  // Чтение данных о местоположении сторонних спутников
  let p1 = producer.read(plength);
  let p2 = producer.read(plength);
  let p3 = producer.read(plength);

  // Проверка на ошибки чтения
  if (p3.length != plength || p1.length != plength || p2.length != plength) {
    errors += 1
    return
  }

       // Преобразование полученных данных в массив
  let p = [new Float32Array(p1.slice(1).buffer), new
  ↪  Float32Array(p2.slice(1).buffer), new Float32Array(p3.slice(1).buffer)]

  // Расчет местоположения методом трилатерации
  let a = trilateration(p[0],p[1],p[2],p[0][3],p[1][3],p[2][3])

  // Передача данных в эфир
  var data = new Float32Array(a[1]);
  var byteView = new Uint8Array(data.buffer);
  transmitter.transmit(byteView)
}
```

День 2

```
// helper function to calculate magnitude of a vector
function magnitude(v) {
  return Math.sqrt(v[0] * v[0] + v[1] * v[1] + v[2] * v[2]);
}

function scalarProduct(vector, scalar) {
  return vector.map((coord) => coord * scalar);
}

// helper function to scale a vector by a scalar
function scale(s, v) {
  return [s * v[0], s * v[1], s * v[2]];
}

function justProduct(a, b) {
  return scale(b, a);
}

// helper function to calculate cross product of two vectors
function crossProduct(v1, v2) {
```

```
  let x = v1[1] * v2[2] - v1[2] * v2[1];
  let y = v1[2] * v2[0] - v1[0] * v2[2];
  let z = v1[0] * v2[1] - v1[1] * v2[0];
  return [x, y, z];
}

// helper function to calculate dot product of two vectors
function dotProduct(v1, v2) {
  return v1[0] * v2[0] + v1[1] * v2[1] + v1[2] * v2[2];
}

function ecefToKeplerian(position1, position2, time) {
  const mu = 3.986004418e14; // gravitational parameter for Earth (m^3/s^2)
  const J2 = 1.082626925638815e-3; // second zonal coefficient for Earth
  const R = 6.378137e6; // radius of Earth (m)
  const omega_e = 7.2921159e-5; // angular velocity of Earth (rad/s)

  // Calculate the position vector and velocity vector from the two ECEF coordinates
  const r1 = position1.map((coord) => coord * 1000);
  const r2 = position2.map((coord) => coord * 1000);
  const dt = time[1] - time[0];
  const v1 = r1.map((coord, i) => (r2[i] - coord) / dt);
  const v2 = r2.map((coord, i) => (coord - r1[i]) / dt);

  // Calculate the specific angular momentum vector
  const h = crossProduct(r1, v1);

  // Calculate the inclination
  const i = Math.acos(h[2] / magnitude(h));

  // Calculate the right ascension of the ascending node
  const n = crossProduct([0, 0, 1], h);
  const raan = (n[1] >= 0 ? Math.acos(n[0] / magnitude(n)) : 2 * Math.PI - Math.acos(n[0] / magnitude(n)));

  // Calculate the eccentricity vector
  const crossVH = crossProduct(v1, h);

  const e = scalarProduct(crossVH, 1 / mu).map((coord, i) => coord - r1[i] / magnitude(r1));

  // Calculate the eccentricity
  const ecc = magnitude(e);

  // Calculate the argument of periapsis
  // const w = (e[2] >= 0 ? Math.acos(dotProduct(n, e) / (magnitude(n) * ecc)) : 2 * Math.PI - Math.acos(dotProduct(n, e) / (magnitude(n) * ecc)));

  // const w = (e[2] >= 0 ? Math.acos(dotProduct(n, e) / (magnitude(n) * magnitude(e))) : 2 * Math.PI - Math.acos(dotProduct(n, e) / (magnitude(n) * magnitude(e))));
  const w = Math.acos(dotProduct(n, e) / (magnitude(n) * magnitude(e)));

  // const w = Math.atan2(e[2], Math.sqrt(e[0]**2 + e[1]**2));

  // Calculate the true anomaly
  const E = Math.atan2(dotProduct(e, crossProduct(h, r1)) / (mu * ecc), (dotProduct(r1, e) / mu) - 1 / ecc);
  const v = 2 * Math.atan(Math.sqrt((1 + ecc) / (1 - ecc)) * Math.tan(E / 2));
```

```
74
75    // Calculate the semi-major axis
76    const a = 1 / (2 / magnitude(r1) - magnitude(v1) ** 2 / mu);
77
78    // Calculate the mean motion
79    const n_0 = Math.sqrt(mu / Math.pow(a, 3));
80
81    // Calculate the mean anomaly
82    // const M_0 = E - ecc * Math.sin(E);
83    // true anomaly
84    const M_0 = v - ecc * Math.sin(v);
85
86    // Calculate the time since periapsis
87    const t_p = -M_0 / n_0;
88
89    // Calculate the time of periapsis passage
90    const t_0 = time[0] - t_p;
91
92    // Return the Keplerian orbit elements
93    return { semiMajorAxis: a, eccentricity: ecc, inclination: i, raan: raan,
      ↪  argumentOfPeriapsis: w, meanAnomaly: M_0, tAnomaly: v,
      ↪  timeOfPeriapsisPassage: t_0 };
94  }
95
96  function trilateration(P1, P2, P3, r1, r2, r3) {
97    const p1 = [0, 0, 0];
98    const p2 = [P2[0] - P1[0], P2[1] - P1[1], P2[2] - P1[2]];
99    const p3 = [P3[0] - P1[0], P3[1] - P1[1], P3[2] - P1[2]];
100   const v1 = p2.map((x, i) => x - p1[i]);
101   const v2 = p3.map((x, i) => x - p1[i]);
102
103   const Xn = v1.map((x) => x / Math.sqrt(v1.reduce((sum, xi) => sum + xi ** 2,
      ↪  0)));
104
105   const tmp = [
106     v1[1] * v2[2] - v1[2] * v2[1],
107     v1[2] * v2[0] - v1[0] * v2[2],
108     v1[0] * v2[1] - v1[1] * v2[0],
109   ];
110
111   const Zn = tmp.map((x) => x / Math.sqrt(tmp.reduce((sum, xi) => sum + xi ** 2,
      ↪  0)));
112
113   const Yn = [
114     Xn[1] * Zn[2] - Xn[2] * Zn[1],
115     Xn[2] * Zn[0] - Xn[0] * Zn[2],
116     Xn[0] * Zn[1] - Xn[1] * Zn[0],
117   ];
118
119   const i = Xn.reduce((sum, xi, j) => sum + xi * v2[j], 0);
120   const d = Xn.reduce((sum, xi, j) => sum + xi * v1[j], 0);
121   const j = Yn.reduce((sum, yi, k) => sum + yi * v2[k], 0);
122
123   const X = ((r1 ** 2) - (r2 ** 2) + (d ** 2)) / (2 * d);
124   const Y = (((r1 ** 2) - (r3 ** 2) + (i ** 2) + (j ** 2)) / (2 * j)) - ((i / j) *
      ↪  X);
125   const Z1 = Math.sqrt(Math.max(0, r1 ** 2 - X ** 2 - Y ** 2));
126   const Z2 = -Z1;
127
128   const K1 = [P1[0] + X * Xn[0] + Y * Yn[0] + Z1 * Zn[0],
```

```
    P1[1] + X * Xn[1] + Y * Yn[1] + Z1 * Zn[1],
    P1[2] + X * Xn[2] + Y * Yn[2] + Z1 * Zn[2]];
  const K2 = [P1[0] + X * Xn[0] + Y * Yn[0] + Z2 * Zn[0],
    P1[1] + X * Xn[1] + Y * Yn[1] + Z2 * Zn[1],
    P1[2] + X * Xn[2] + Y * Yn[2] + Z2 * Zn[2]];

  return [K1, K2];
}


var plength = 0

function rad2deg(radians) {
  return radians * (180/Math.PI);
}

function setup() {
    Math.sqrt(NaN)
  plength = 4*4+1;
  positioning_system_bus = spacecraft.devices[0].functions[0];
  positioning_system_receiver = spacecraft.devices[1].functions[0];
}
let state = 0;
let cord1;
let cord2;
function loop() {
  let p1 = positioning_system_receiver.read(plength);
  let p2 = positioning_system_receiver.read(plength);
  let p3 = positioning_system_receiver.read(plength);

  if (p3.length != plength || p1.length != plength || p2.length != plength) {
    return
  }

  if (p1[0] != 0 || p2[0] != 1 || p3[0] != 2) {
    throw new Error('unsynk id' + p1 + "=" + p2);
  }

  let p = [new Float32Array(p1.slice(1).buffer), new
  ↪  Float32Array(p2.slice(1).buffer), new Float32Array(p3.slice(1).buffer)]
  let pp = trilateration(p[0],p[1],p[2],p[0][3],p[1][3],p[2][3])

  p = pp[1];

  if (state == 0) {
    cord1 = p;
    state+=1;
    return;
  }
  if (state == 1) {
    cord2 = p;
    state+=1;
    return;
  }


  let a = ecefToKeplerian([cord1[0]/1000,
  ↪  cord1[1]/1000,cord1[2]/1000],[cord2[0]/1000, cord2[1]/1000, cord2[2]/1000],
  ↪  [0, 10]);

```

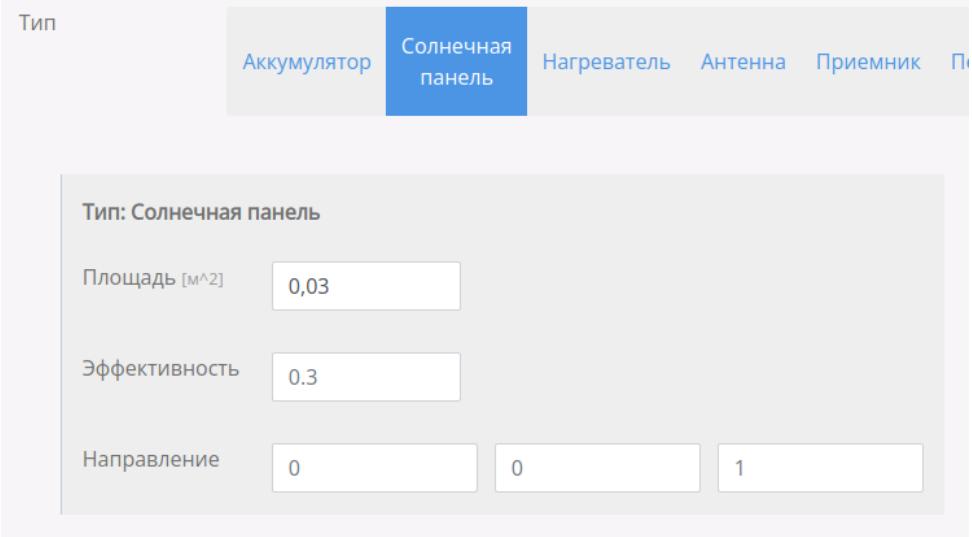
```
    var data = new Float32Array([rad2deg(a.inclination), a.semiMajorAxis,
    ↪  rad2deg(a.raan), rad2deg(a.meanAnomaly), rad2deg(a.argumentOfPeriapsis)]);
    var byteView = new Uint8Array(data.buffer);
    positioning_system_bus.transmit(byteView);
}
```

День 3

Научный спутник SC0

Состав

Среди аппаратов формата CubeSat для проведения эксперимента в большей степени подойдет типоразмер 3U. Поэтому при дальнейшем моделировании можно опираться на это предположение. Исходя из предположения, что научный аппарат может быть представлен форм-фактором CubeSat 3U, можно предположить, что его четыре боковые панели покрыты солнечными батареями, площадь которых будет 0,03 м^2. Направление солнечных батарей будет задаваться кватернионами, например: $[1, 0, 0]$, $[-1, 0, 0]$, $[0, 1, 0]$ и $[0, -1, 0]$.



Программа управления

Ниже представлен код управления, который включает источник данных и передатчик при вхождении в зону видимости.

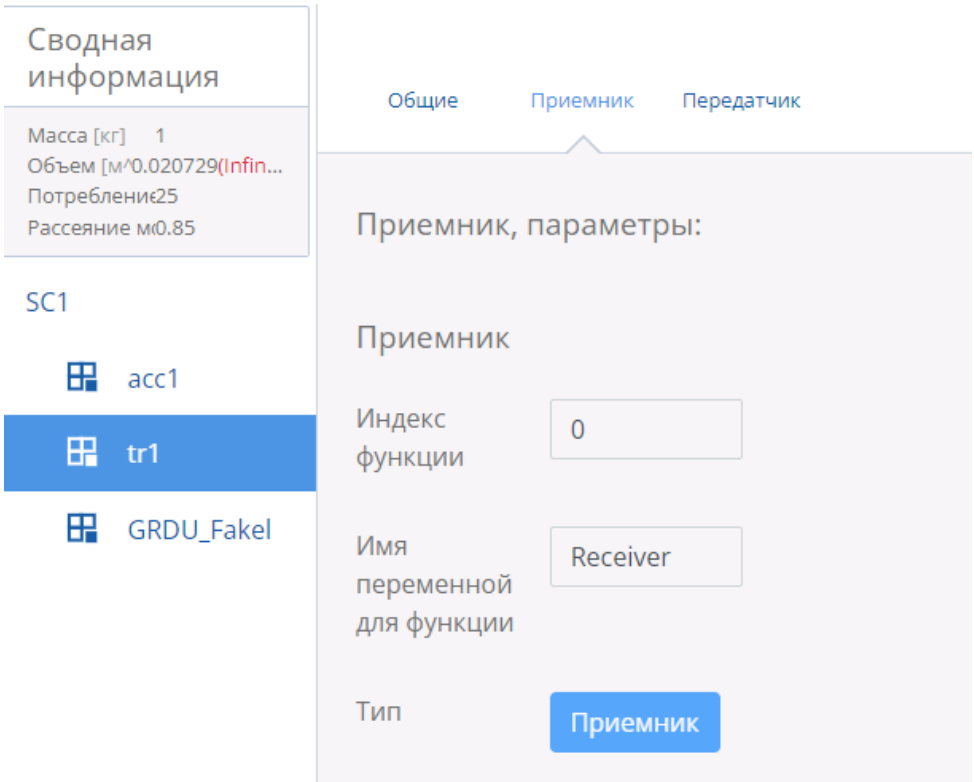
```
'use strict';
var transmitter;
var time;
var inzone;

function time_in(tm, h1,m1,h2,m2) {
    if (tm >= h1*3600+m1*60 && tm <= h2*3600+m2*60) {
        return true;
    } else {
        return false;
    };
}

function setup() {
    transmitter = spacecraft.devices[1].functions[0];
```

```
}
function loop() {
    time = spacecraft.flight_time;
    inzone = false;
    if(time_in(time,7,25,7,51)){
        inzone = true;
    };
    if(time_in(time,16,5,16,15)){
        inzone = true;
    };
    if(time_in(time,17,40,17,50)){
        inzone = true;
    };
    if(time_in(time,19,20,19,40)){
        inzone = true;
    };
    if(time_in(time,21,0,21,10)){
        inzone = true;
    };
    if(time_in(time,22,40,22,55)){
        inzone = true;
    };
    if(inzone){
        transmitter.enable();
    } else {
        transmitter.disable();
    };
}
```

Спутники связи SC1-3

Состав

Все устройства необходимо задать в соответствии с условием задачи.

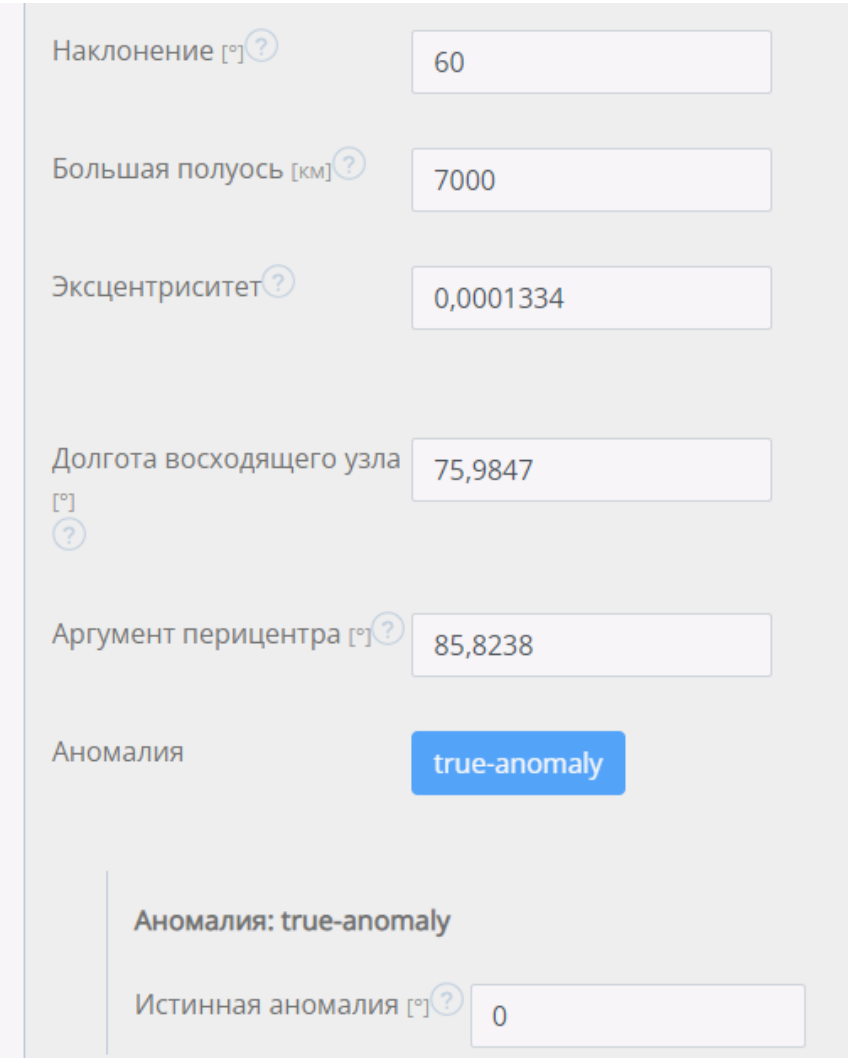


Программа управления

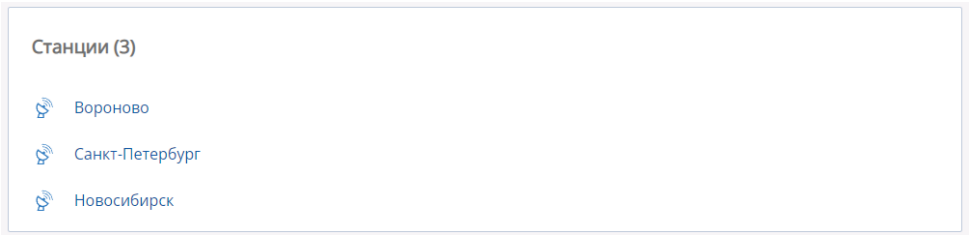
Программа управления для всех трех аппаратов аналогична за исключением моментов включения двигателей. Ниже приведен код для одного из аппаратов.

```
'use strict';

var thruster;
var receiver;
var transmitter;

let t1 = 0;
let dt = 400;

function setup() {
  thruster = spacecraft.devices[2].functions[0];
  receiver = spacecraft.devices[1].functions[0];
  transmitter = spacecraft.devices[1].functions[1];
  receiver.enable();
  transmitter.enable();
}

function loop() {

  if (spacecraft.flight_time < dt) {
      thruster.thrust = thruster.maximum_thrust;
  }
  else {
      thruster.thrust = 0
  }
}
```

Орбиты аппаратов Для выполнения поставленной задачи подойдет орбита, которая хотя бы раз пройдет над тремя станциями. Представленные ниже параметры орбиты проходят раз в сутки над указанными городами, например над Вороново, Новосибирск и Санкт-Петербургом.



Наземные станции



Результат моделирования

Материалы для подготовки

1. Ежесезонная доработка постоянного курса подготовки к профилю, расположенного по адресу https://orbita.education/events/13.
2. Задания инженерной части предметного тура, реализованные как индивидуальные курсы подготовки, в т. ч. например (ввиду закрытия событий приводятся прямые ссылки на просмотр):
   - Курс для программиста микроконтроллеров: http://orbita.education/events/532779cec855459d847d67f807473d08.
   - Курс для инженера-схемотехника: http://orbita.education/events/dc95b2aa59a64a3aaaa0d8e316fe90c4.
   - Курс для баллистика: http://orbita.education/events/5eda891e22aa4abea209703f7983b9b4.
   - Курс для инженера-радиотехника: http://orbita.education/events/54990cc2a8eb47e89c9e71857021168f.
3. Запись вебинара по баллистике: https://youtu.be/CxW0AnmzfhA.
4. Запись вебинара по GNU Radio: https://youtu.be/wsBSd6rZTNU.
5. Запись вебинаров по KiCad:
   - Вебинар сезона 22/23, https://youtu.be/8A8OGThgfr8.
   - Вебинар сезона 21/22, часть 1: https://youtu.be/SoM8YOO9hFU.
   - Вебинар сезона 21/22, часть 2: https://youtu.be/MqI50P1a8-s.