Второй отборочный этап

Индивидуальная часть

Связист

Тест посвящен работе с STM32CubeIDE — средой разработки для микроконтроллеров семейства STM32. С данной IDE вам предстоит более плотно работать на финале. С помощью нее необходимо будет реализовать алгоритмы управления приемопередатчиком CC1101, управляемым с помощью микроконтроллера STM32F303CCT6.

Для того, чтобы пройти данный тест, необходимо скачать и установить на ПК программное обеспечение для работы с микроконтроллерами семейства STM32 — STM32CubeIDE. ПО бесплатное, однако требует регистрации на сайте. Скачать ПО можно по ссылке:

https://www.st.com/en/development-tools/stm32cubeide.html.

Тест сделан по версии STM32CubeIDE v1.11.

Задания-тесты сделаны в большей степени в виде гайда, который мы вам предлагаем пройти, чтобы познакомиться с ПО. Тесты не являются сложными, но чтобы ответить на вопросы, необходимо выполнить несколько последовательных действий в самой IDE. Чтобы прохождение гайдов не вызывало больших затруднений желательно иметь представлении о программировании на C/C++ и начальное понимание о работе микроконтроллеров.

Полезным будет ознакомиться также с практикумом прошлого года, в котором рассматривается работа с платой Nucleo на базе микроконтроллера F303RE:

https://drive.google.com/file/d/10J6GTLQtct9am54CZvkGh2V4ciAoF2cG/view?usp=sharing

Если вы никогда не работали с микроконтроллерами, то советуем вам для начала немного ознакомиться с программированием микроконтроллеров на примере плат Arduino. В интернете можно найти много курсов на эту тему, вот один из них: https://stepik.org/course/69511/syllabus

Если вам не знаком язык программирования C, то также можно пройти любой курс, например: https://stepik.org/course/80538/promo

Или вы можете попробовать во всем разобраться сходу, такое тоже возможно.

Задача II.1.1.1. Пробная задача (0 баллов)

Описание Спутник может быть размещен на произвольной орбите 15 декабря 2021 года в 16:00 по UTC. Задайте параметры орбиты через кеплеровы элементы так, чтобы 16-го, 17-го и 18-го декабря в 8:00 по UTC каждого дня аппарат прошел над Владивостоком.

Координатами Владивостока считать GPS — координаты: 43.116418 по широте, 131.882475 по долготе.

Начальные данные

Радиус Земли, м: 6371008.8

Гравитационный параметр, $\nu = G \cdot M$, м^3/с^2: $3.986004418 \cdot 1014$

Средняя скорость вращения Земли, об/сут: 1.00273781191135448

Начальный угол вращения Земли: 24.66°

Критерии оценки

Задача считается выполненной, если аппарат в любой из дней проходит с отклонением до 25 км от цели и до 60 мин от нужного времени.

Максимальное число баллов за задачу: 0.

В этой задаче есть такое понятие как "очки успеха". Очки успеха не являются значащими баллами. Они показывают, насколько успешно решена задача.

Правила начисления очков успеха:

- Аппарат в любой из дней проходит с отклонением до 25 км от цели и до 60 мин от нужного времени — 2 очка успеха;
- Аппарат в любой из дней проходит с отклонением до 25 км от цели и до 15 мин от нужного времени — 3 очка успеха;
- При выполнении нескольких условий очки успеха суммируются;
- Максимальное число очков успеха: 30.

Для решения задачи можно воспользоваться видеоразбором заданий предыдущего года:

- типы орбит, часть 1:
  https://www.youtube.com/watch?v=xGjOmDieZCc
- типы орбит, часть 2:
  https://www.youtube.com/watch?v=XRWXfEw3Byg
- разбор задачи «Расчет орбиты»:
  https://www.youtube.com/watch?v=BsL_jPl0Kj4

Задача II.1.1.2. Задача по физике (2 балла)

Для некоторых спутников связи лучше подходит не геостационарная, а высокоэллиптическая орбита. Она является не круговой, а нарочито вытянутой, эллиптической. Центр Земли, соответственно, находится в одном из фокусов этого эллипса. Высокоэллиптическая орбита хорошо подходит для внутригосударственной связи и телевещания. Таким образом, спутник быстро пролетает по «низкой» части орбиты и долго находится над страной-пользователем системы связи, двигаясь по «высокой» части орбиты. Естественно, длительное нахождение над обслуживаемой страной получается ценой очень большого расстояния от Земли до апогея орбиты, и, соответственно, необходимостью обеспечить высокую мощность передатчика. Для наземного наблюдателя спутник при прохождении апогея кажется двигающимся в небе с очень маленькой скоростью, что удобно для наземных приемных станций: на протяжении нескольких часов пролета по «высокой» части орбиты не требуется поворот антенн,

осуществляющих обмен информацией со спутником. Часто параметры орбиты подбираются так, чтобы период обращения спутника составлял ровно 24 часа. Тогда апогей орбиты всегда находится над одной и той же точкой Земли.

Спутник связи летает по высокоэллиптической орбите, имеющей высоту перигея h = 500 км, находящейся в экваториальной плоскости и периодом обращения ровно 24 часа. Спутник обеспечивает передачу данных между устройствами на Земле. Частота передатчика на спутнике составляет 435 МГц. Оценить разность частоты передатчика и частоты, воспринимаемой наземной антенной станцией в момент, когда спутник находится на половине пути от апогея к перигею. Орбита спутника построена так, что в момент прохождения апогея наземная станция находится точно под ним на экваторе.

Справка о законах Кеплера:

Математик Кеплер установил, что если тело движется по эллиптической орбите, то центр Земли находится в одном из фокусов этого эллипса (первый закон Кеплера). Частным случаем эллипса является окружность, в этом случае оба фокуса эллипса находятся в одной и той же точке. Также для тела, движущегося по эллиптической орбите, справедливо равенство (второй закон Кеплера):

$$V(t_1) \cdot r(t_1) \cdot sin\Theta_1 = V(t_2) \cdot r(t_2) \cdot sin\Theta_2$$

Где t_1 и t_2 — произвольные моменты времени,

r — расстояние от тела до центра Земли,

V — орбитальная скорость тела,

Θ — угол между вектором скорости тела и направлением на центр Земли.

Третий закон Кеплера позволяет связать периоды обращения T_1 и T_2 по любым двум околоземным эллиптическим орбитам с бОльшими полуосями эллипсов орбит a_1 и a_2. В частном случае окружности полуосью эллипса является ее радиус.

$$\frac{T_2}{T_1} = (\frac{a_2}{a_1})^{\frac{3}{2}}$$

Эллипс с большим диаметром $2 \cdot a$ и малым диаметром $2 \cdot b$ описывается уравнением:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

При этом начало координат находится в центре эллипса. Обычно $a > b$, то есть большой диаметр эллипса расположен вдоль оси X. Тогда фокусы эллипса находятся на оси X и имеют координаты $(e \cdot a; 0)$ и $(-e \cdot a; 0)$, где e — эксцентриситет эллипса, $0 \leqslant e < 1$.

Эксцентриситет описывает «вытянутость» эллипса. У окружности e=0 , так как большой и малый диаметры совпадают. С вытягиванием эллипса e стремится к 1.

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

Начальные данные:

Масса Земли: $6 \cdot 10^{24}$ кг

Радиус Земли: 6370 км

Гравитационная постоянная: $G = 6.67 \cdot 10^{-11}$ Н·м^2 / кг^2

Радиус круговой орбиты, имеющей период обращения 24 часа: $R_{\text{гсо}} = 42168$ км (геостационарная орбита)

Скорость света: $3 \cdot 10^8$ м/с

Система оценки

За правильный ответ дается 2 балла. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 1.5 на 3-ей попытке, до 1 балла на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 0.5 баллов и далее не убывает.

Решение

Изменение воспринимаемой частоты обусловлено эффектом Доплера. Так как период обращения составляет 24 часа, то спутник вращается синхронно с Землей. Значит, в силу симметрии, спутник будет находиться точно над станцией как при проходе апогея, так и при проходе перигея. Разумеется, между этими моментами времени прямая «Центр Земли — спутник» НЕ будет проходить через наземную станцию.

Найдем сначала размеры орбиты. По третьему закону Кеплера, радиусы геостационарной орбиты и большая полуось эллиптической орбиты одинаковы:

$$R_{\text{гсо}} = a$$

Тогда для высоты спутника в апогее H и перигее h:

$$2 \cdot R_{\text{гсо}} = H + h + 2 \cdot R_{\text{зем}} = r_{\text{ап}} + r_{\text{пер}}$$

Отсюда находим высоту H спутника в апогее:

$$H = 2 \cdot R_{\text{гсо}} - 2 \cdot R_{\text{зем}} - h = 2 \cdot 42168 - 2 \cdot 6370 - 500 = 71096\text{км}$$

Эллипс орбиты является сильно вытянутым, его эксцентриситет почти равен 1:

$$e = \frac{R_{\text{гсо}} - r_{\text{пер}}}{R_{\text{гсо}}} = \frac{42168 - (500 + 6370)}{42168} = 0.837$$

Тогда определим расстояние от центра Земли до спутника в нужный нам момент прохождения половины пути $r_{0.5}$:

$$r_{0.5} = \sqrt{b^2 + (R_{\text{гсо}} - r_{\text{пер}})^2} = \sqrt{R_{\text{гсо}}^2 \cdot (1 - e^2) + (R_{\text{гсо}} - r_{\text{пер}})^2} =$$

$$= \sqrt{41268^2 \cdot (1 - 0.837^2) + (42168 - 6370 - 500)^2} = 42170.8\text{км}$$

Определим угол $\Theta_{0.5}$ между скоростью $V_{0.5}$ в момент прохождения половины пути и направлением на центр Земли $r_{0.5}$:

$$cos_{\Theta_{0.5}} = \frac{R_{\text{гсо}} - r_{\text{пер}}}{r_{0.5}} = \frac{42168 - 6370 - 500}{42170.8} = 0.837$$

Осталось найти скорость спутника в момент прохождения половины пути. По первому закону Кеплера и закону сохранения энергии составим систему из двух уравнений:

$$\begin{cases} V_{\text{перигей}} \cdot r_{\text{пер}} = V_{0.5} \cdot r_{0.5} \cdot sin\Theta_{0.5}, \\ \frac{V^2_{\text{перигей}}}{2} - \frac{G \cdot M_{\text{зем}}}{r_{\text{пер}}} = \frac{V^2_{0.5}}{2} - \frac{G \cdot M_{\text{зем}}}{r_{0.5}} \end{cases}$$

В реальности нужно найти скорость на половине пути $V_{0.5}$, поэтому избавляемся от скорости в перигее:

$$\begin{cases} V_{0.5} \cdot \frac{r_{0.5}}{r_{\text{пер}}} \cdot sin\Theta_{0.5} = V_{\text{перигей}}, \\ \frac{V^2_{0.5}}{2} - \frac{G \cdot M_{\text{зем}}}{r_{0.5}} = \frac{(V_{0.5} \cdot \frac{r_{0.5}}{r_{\text{пер}}} \cdot sin\Theta_{0.5})^2}{2} - \frac{G \cdot M_{\text{зем}}}{r_{\text{пер}}} \end{cases}$$

Получаем уравнение:

$$\frac{V^2_{0.5}}{2} \cdot ((\frac{r_{0.5}}{r_{\text{пер}}} \cdot sin\Theta_{0.5})^2 - 1) = G \cdot M_{\text{зем}} \cdot (\frac{1}{r_{\text{пер}}} - \frac{1}{r_{0,5}})$$

Окончательно:

$$V_{0.5} = \sqrt{\frac{2 \cdot G \cdot M_{\text{зем}} \cdot (\frac{1}{r_{\text{пер}}} - \frac{1}{r_{0.5}})}{(\frac{r_{0.5}}{r_{\text{пер}}} \cdot cos\Theta_{0.5})^2 - 1}} = \sqrt{\frac{2 \cdot G \cdot M_{\text{зем}} \cdot (\frac{1}{r_{\text{пер}}} - \frac{1}{r_{0.5}})}{(\frac{r_{0.5}}{r_{\text{пер}}})^2 \cdot (1 - (cos\Theta_{0.5})^2) - 1}} =$$

$$= \sqrt{\frac{2 \cdot 6.67 \cdot 10^{-11} \cdot 6 \cdot 10^{24} \cdot 10^{-3} \cdot (\frac{1}{6370+500} - \frac{1}{42170.8})}{(\frac{42170.8}{6370+500})^2 \cdot (1 - 0,837^2) - 1}} = 3079.7 \text{ м/с}$$

Смещение воспринимаемой частоты будет положительным, так как спутник движется К наблюдателю. Найдем смещение частоты.

$$v_1 = v_{\text{спутника}} \cdot \frac{\sqrt{1 - (\frac{V_{0.5}}{c})^2}}{1 - \frac{V_{0.5}}{c} \cdot cos(\beta)} \approx \frac{v_{\text{спутника}}}{1 - \frac{V_{0.5}}{c} \cdot cos(\beta)}$$

Где β — угол «Центр Земли — Спутник — Станция».

Угол β оценим в пренебрежении смещением станции вследствие вращения Земли за время пролета спутником четверти орбиты:

$$cos\beta \approx cos\Theta_{0.5} = \frac{R_{\text{гсо}} - r_{\text{пер}}}{r_{0.5}} = \frac{42168 - 500 - 6370}{42170.8} = 0.837$$

Соответственно, для отклонений частоты получим:

$$\delta_+ = v_1 - v_{\text{спутника}} = v_{\text{спутника}} \cdot (\frac{1}{1 - \frac{V_{0.5}}{c} \cdot cos(\beta)} - 1) == 435 \cdot 10^6 \cdot (\frac{1}{1 - \frac{3079.7}{3 \cdot 10^8} \cdot 0.837} - 1) =$$

$$= 3738 \text{ Гц} = 3,8 \text{ КГц}$$

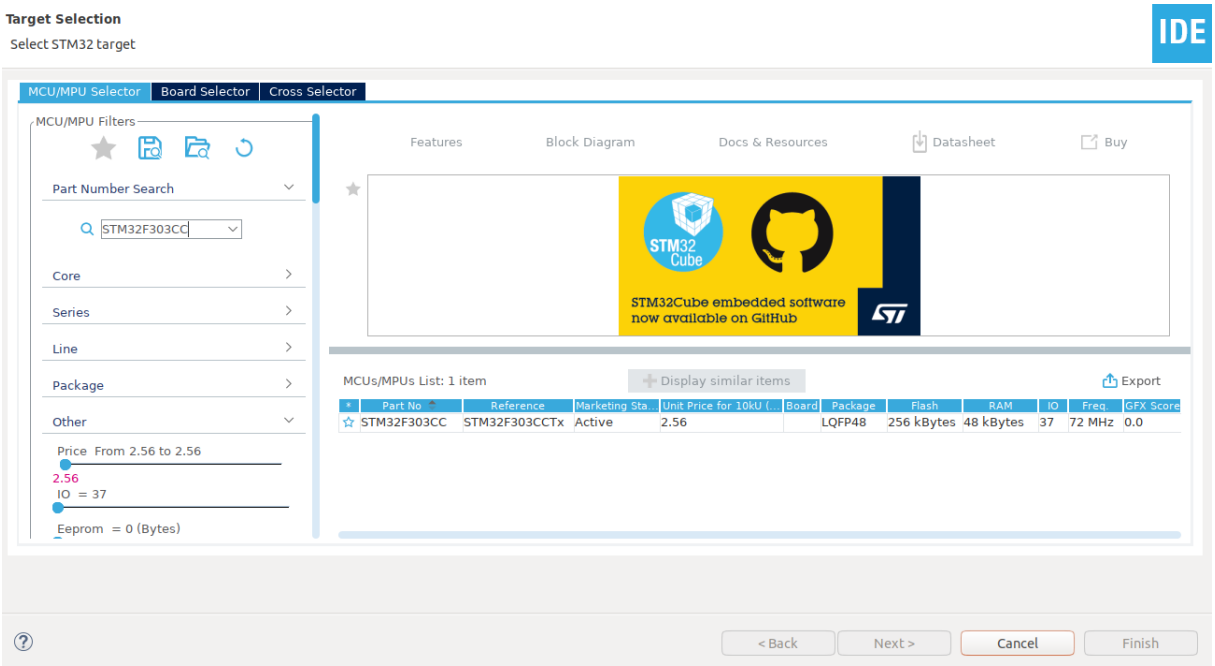
Ответ: 3,8 КГц.

Задача II.1.1.3. Инициализация портов ввода/вывода, знакомство с GPIO (2 балла)

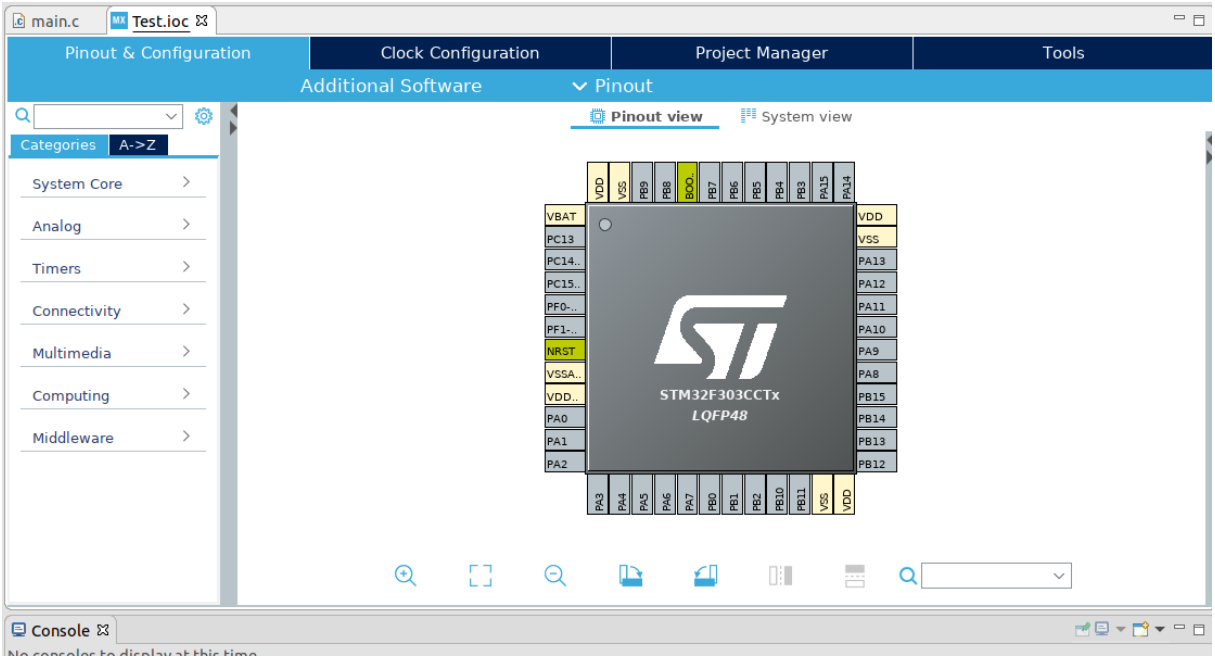
Инициализация портов ввода / вывода, знакомство с GPIO

В STM32CubeIDE в верхнем меню выберите File → New → STM32 Project.

В открывшемся окне найдите микроконтроллер STM32F303CCTx и выберите его.



Нажмите кнопку Next. Необходимо будет ввести имя проекта и нажать кнопку Finish, после чего откроется файл с расширением .ioc, в котором будет представлен выбранный микроконтроллер.



Почти каждый вывод микроконтроллера (пин) на представленной схеме можно

настроить. Для этого нужно кликнуть по выводу левой кнопкой мыши и выбрать нужный режим настройки вывода.

Предположим, что мы имеем простую систему, в которой по сигналу с концевика происходит включение вибромотора на 10 секунд. Логику работы системы можно также описать следующим образом:

- если концевик в разомкнутом положении, то вибромотор находится в выключенном состоянии;
- если концевик в сомкнутом положении, то вибромотор включается на 10 секунд.

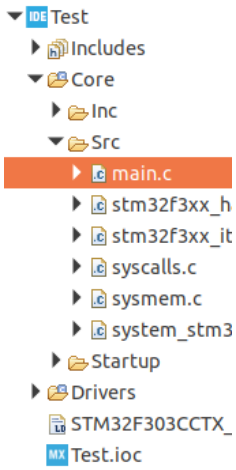
В данном случае сигнал с концевика является входным сигналом, а включение вибромотора — выходным.

Для реализации необходимой инициализации пинов воспользуемся режимом GPIO — ввода/вывода данных.

Более подробно о GPIO можно прочитать, например, здесь: http://mypractic.ru/urok-6-porty-vvoda-vyvoda-stm32.html

Необходимо задать PB0 как входной сигнал (GPIO_Input), PB1 как выходной (GPIO_Output). После этого в слева в меню в категории System Core в пункте GPIO отобразятся выбранные пины.

После инициализации пинов необходимо сгенерировать код (значок шестеренки в верхней панели меню), и открыть файл main.c в дереве проекта.



В файл main.c генерируется код, который содержит инициализацию необходимых пинов, интерфейсов и пр, а также функцию «int main(void)», которая задает работу микроконтроллера.

Найдите функцию «static void MX_GPIO_Init(void)».В ней есть строки, отвечающие за инициализацию пинов PB0 и PB1.

```
/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);

/*Configure GPIO pin : PB0 */
GPIO_InitStruct.Pin = GPIO_PIN_0;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : PB1 */
GPIO_InitStruct.Pin = GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
```

По описанным настройкам видно, что режим работы пина в качестве входа или выхода определяется полем Mode. Кликните по присваиваемому значению данного поля (GPIO_MODE_INPUT или GPIO_MODE_OUTPUT_PP) правой кнопкой мыши и выберите «Open Declaration».

Откроется файл с описанием всех модов, которые можно задать для пина, работающего в режиме GPIO. Сколько таких модов всего?

Критерий оценки: За правильный ответ начисляется 2 балла.

Всего доступно 2 попытки.

Решение

После выполнения всех указанных действий в STM32CubeIDE откроется в новой вкладке файл, в котором можно найти инициализацию режимов GPIO:

```
#define  GPIO_MODE_INPUT                  (0x00000000U)   /*!< Input Floating Mode                   */
#define  GPIO_MODE_OUTPUT_PP              (0x00000001U)   /*!< Output Push Pull Mode                 */
#define  GPIO_MODE_OUTPUT_OD              (0x00000011U)   /*!< Output Open Drain Mode                */
#define  GPIO_MODE_AF_PP                  (0x00000002U)   /*!< Alternate Function Push Pull Mode     */
#define  GPIO_MODE_AF_OD                  (0x00000012U)   /*!< Alternate Function Open Drain Mode    */
#define  GPIO_MODE_ANALOG                 (0x00000003U)   /*!< Analog Mode  */
#define  GPIO_MODE_IT_RISING              (0x10110000U)   /*!< External Interrupt Mode with Rising edge trigger detection          */
#define  GPIO_MODE_IT_FALLING             (0x10210000U)   /*!< External Interrupt Mode with Falling edge trigger detection         */
#define  GPIO_MODE_IT_RISING_FALLING      (0x10310000U)   /*!< External Interrupt Mode with Rising/Falling edge trigger detection  */
#define  GPIO_MODE_EVT_RISING             (0x10120000U)   /*!< External Event Mode with Rising edge trigger detection               */
#define  GPIO_MODE_EVT_FALLING            (0x10220000U)   /*!< External Event Mode with Falling edge trigger detection              */
#define  GPIO_MODE_EVT_RISING_FALLING     (0x10320000U)   /*!< External Event Mode with Rising/Falling edge trigger detection       */
```

Как можно видеть, всего таких режимов 12.

Ответ: 12.

Задача II.1.1.4. Использование GPIO (2 балла)

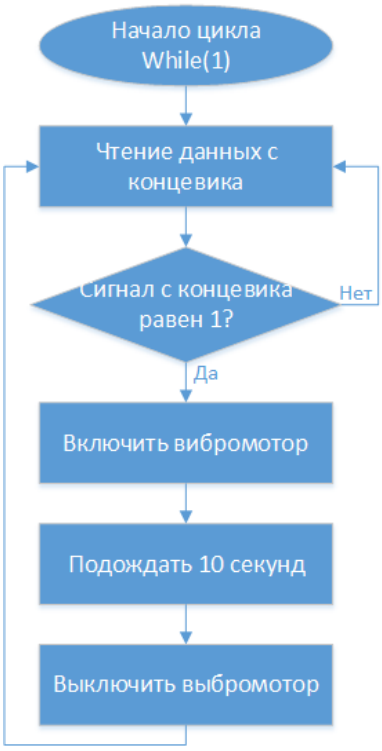
Основная работа микроконтроллера задается функцией «int main(void)» в файле main.c. Обратите внимание, что изначально в ней не так много кода, большая часть — это комментарии (выделяются как правило зеленым или серым цветом), помогающие сориентироваться. Можно заметить, что есть ряд комментариев, содержащих слова USER CODE BEGIN и USER CODE END. Это те места в функции, в которых желательно писать код. Тогда при возврате к распиновке микроконтроллера и последующей генерации кода код не будет потерян. Иначе написанный вами код может затираться каждый раз при новой генерации кода.

Структура функции «int main(void)» задана таким образом, что все, что необходимо задать изначально, задается до бесконечного цикла while(1). В самом же цикле задается программа микроконтроллера, которая повторяется до тех пор, пока микроконтроллер не будет обесточен. Если сравнивать со структурой скетча Arduino IDE, то все до цикла while(1) можно поставить в соответствие функции void setup(), а сам цикл функции void loop().

Обратите внимание, что в функции int main(void) есть следующие строки:

1. HAL_Init();
   Эта строчка вызывает инициализацию функций HAL — типовых функций, которыми мы будем пользоваться.
2. SystemClock_Config();
   Эта строчка вызывает функцию инициализации системных настроек, таких как источники тактирования и частоты работы различных интерфейсов.
3. MX_GPIO_Init();
   Уже известная вам функция, которая инициализирует настройку пинов, работающих в режиме GPIO.

Вернемся к логике работы системы. Ее можно описать следующей блок-схемой:



Действие «Чтение сигнала с датчика» в нашем случае будет задаваться типовой функцией HAL_GPIO_ReadPin. Единице соответствует состояние, когда на пине PB0 есть напряжение (т. е. уровень напряжения 3.3 В), а нулю — когда напряжения нет (т. е. уровень напряжения — 0 В). Напряжение на пине появляется тогда, когда концевик находится в замкнутом положении, а отсутствие напряжения соответствует разомкнутому положению концевика.

Включение и выключение вибромотора происходит за счет подачи на пин PB1 напряжения. Например, при напряжении в 3.3 В на пине PB1 вибромотор будет включаться, а при напряжении 0 В выключаться. Чтобы подать напряжение на пин можно воспользоваться функцией HAL_GPIO_WritePin.

Код управления можно задать следующим образом:

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

  /* USER CODE BEGIN 3 */
    if (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0) == GPIO_PIN_SET)
    {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_Delay(10000);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);
    }


}
```

Если кликнуть на функцию, определяющую задержку, HAL_Delay() правой кнопкой мыши и выбрать «Open Description», то откроется документ с описанием функции. Внутри функции используется HAL_MAX_DELAY.

В какой-то момент концевик переключили на пин PC6.

a. Как поменяются аргументы функции HAL_GPIO_ReadPin? b. Чему равно значение HAL_MAX_DELAY?

Варианты ответа:

- HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6)
- HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_6)
- HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_0)
- HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0)

Критерий оценки:

- За правильный ответ начисляется 2 балла.
- Всего доступно 2 попытки.

Решение

a. Пин PC6 принадлежит группе портов C, следовательно в его инициализации будет использоваться «GPIOC». По аналогии с PB0 и PB1 номер пина будет определяться как «GPIO_PIN_6». Тогда правильный ответ — 1.

b. Проделав описанные действия, увидим, что значение HAL_MAX_DELAY задается следующим образом:

Переведя 0xFFFFFFFF из шестнадцатеричной системы в десятичную, получим значение 4294967295.

Ответ: 1 — HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_6); 2 — HAL_MAX_DELAY: 4294967295.

Задача II.1.1.5. Интерфейс SPI (2 балла)

В сложных системах, содержащих множество устройств, осмысленно использование единой логики работы для всех устройств. Реализуется это с помощью исполь-

зования различных интерфейсов и протоколов. Одним из распространенных интерфейсов для управления с помощью микроконтроллера одновременно несколькими устройствами является SPI.

Serial Peripheral Interface (SPI) — последовательный интерфейс для обмена данными между микроконтроллером и подчиненными ему периферийными устройствами.

В интерфейсе есть одно управляющее устройство и от одного до нескольких подчиненных. Управляющее устройство называется также master, а подчиненные — slave. Именно управляющее устройство контролирует обмен данными по интерфейсу. В нашем случае в качестве управляющего устройства выступает микроконтроллер STM32F303CCTx.

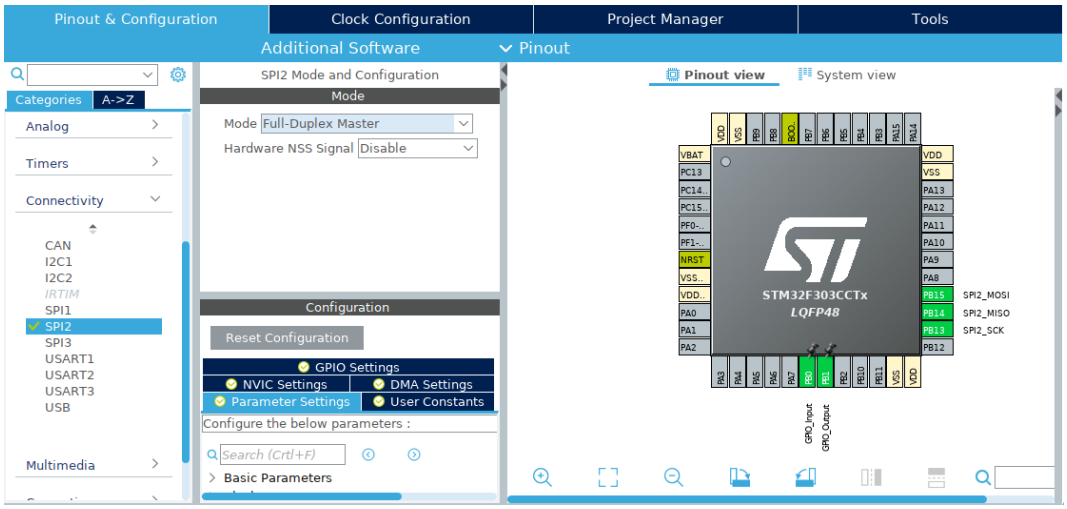
Данный интерфейс использует четыре линии для своей работы, из которых три первых являются общими для всех устройств, а последний — индивидуальным для каждого.

1. MOSI — master output, slave input — линия данных, по которой данные идут от управляющего устройства к подчиненному. Например, команда от микроконтроллера на отображение данных на экране передается по этой линии.
2. MISO — master input, slave output — линия данных, по которой данные поступают к управляющему устройству от подчиненного. Например, чтение данных с датчика происходит по этой линии.
3. SCK — линия, отвечающая за синхронизацию управляющего и подчиненного устройств.
4. CS или SS — chip select или slave select — выбор устройства. Как правило, каждое подчиненное устройство подключается отдельным проводом к микроконтроллеру. Если микроконтроллер на линии CS задает напряжение 0 В (т. е. логический 0), то устройство считается выбранным для обмена данными. На остальных устройствах при этом на их линии CS будет 3.3 В (т. е. логическая 1).

Более подробно об SPI можно прочитать здесь: http://easystm32.ru/interfaces/43-spi-interface-part-1/

Чтобы задать SPI с помощью STM32CubeIDE, нужно вернуться к вкладке с распиновкой микроконтроллера. Если вы ее закрыли, то дважды кликните на файл с расширением .ioc слева в дереве проекта.

Перейдите в меню слева в пункт Connectivity и выберите SPI2. В Mode выберите Full-Duplex Master.



Определятся пины с подписями SPI2_MOSI, SPI2_MISO и SPI2_SCK. Для того, чтобы была возможна передача данных по SPI нужно задать еще один пин, который будет отвечать за выбор устройства.

a. Определите по коду, пин под каким номером определен как линия CS для устройства. В ответе укажите только цифру.

b. Определите по коду, какое действие он выполняет.

```
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
```

Критерий оценки:

1. За правильный ответ начисляется 2 балла.
2. Всего доступно 2 попытки.

Варианты ответа:

- Код инициализирует интерфейс SPI.
- Код выбирает устройство для передачи данных по SPI на подчиненное устройство.
- Код выбирает устройство для чтения данных по SPI с подчиненного устройства.
- Код выбирает устройство для работы по SPI. Возможно как чтение данных, так и передача данных.

Решение

a. В коде идет управление пинами PB13, PB15 и PA8. Пины PB13 и PB15 являются линией тактирования интерфейса SPI и линий обмена данными MOSI. Тогда логично, что PA8 — это пин, отвечающий за выбор подчиненного устройства, то есть CS.

b. Строка кода

«HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);»

выбирает для обмена данными подчиненное устройство. В последствии при использовании определенных функций возможно как чтение, так и запись данных.

Ответ: a — 8; b — 4.

Задача II.1.1.6. Передатчик CC1101 (2 балла)

Одним из устройств, с которым вам предстоит работать на финале — это программируемый передатчик CC1101. Данный передатчик работает по интерфейсу SPI.

Работа с передатчиком включает в себя несколько обязательных пунктов, описанных ниже.

1. Конфигурация передатчика
   Конфигурация передатчика определяет то, на какой частоте будет передача сигнала, с какой мощностью, какой тип модуляции будет использоваться при передачи и т. п. Конфигурация задается записью в регистры передатчика определенных значений. Эти значения можно узнать из документации на передатчик.
2. Инициализация режима работы передатчика
   Так как передатчик может работать как на прием (RX), так и на передачу (TX), то необходимо устанавливать режим его работы. Режим работы передатчика задается, так называемыми, стробами — командами, разрешающими и запрещающими определенные действия. Чтобы установить строб — необходимо передать передатчику по SPI значение данного строба.

Ознакомиться с некоторыми особенностями работы с передатчиком можно по задаче прошлого года:

https://drive.google.com/file/d/1GfYrdlM-1cYsoOCt4ITmjKw5uEjWKwy6/view?usp=sharing

Если упростить весь приводимый по ссылке код и опустить инициализацию передатчика, то получится следующее:

```
char* buffer = "Test Test Test"; // данные, которые передаем в сообщении

uint8_t tx[2];
tx[0] = CC1101_TXFIFO; // CC1101_TXFIFO определяет работу передатчика на передачу с использованием буфера
tx[1] = strlen(buffer); // находим длину строки передаваемых данных
uint8_t rx[2]; rx[0]='\0';rx[1]='\0'; // задаём вспомогательную переменную
HAL_SPI_TransmitReceive(&hspi2, tx, rx, 2, HAL_MAX_DELAY); // устанавливаем длину сообщения для передачи


uint8_t addr;
addr = CC1101_TXFIFO | WRITE_BURST;  // определяем режим работы передатчика

HAL_SPI_Transmit(&hspi2, addr, 1, HAL_MAX_DELAY); // готовим передатчик к записи данных,
                                                  //предназначенных для передачи
HAL_SPI_Transmit(&hspi2, buffer, strlen(buffer), HAL_MAX_DELAY); // записываем в передатчик данные,
                                                                 //предназначенные для передачи

uint8_t rx[1]; rx[0]='\0';
HAL_SPI_TransmitReceive(&hspi2, CC1101_STX, rx, 1, HAL_MAX_DELAY); // с помощью строба разрешаем передачу
```

Такой код инициализирует передачу текстовой информации с передатчика.

Как видно из кода, работа построена на последовательном обмене данных между микроконтроллером и передатчиком по SPI.

Опираясь на код выше, определите, какие из действий являются обязательными при формировании передачи.

Варианты ответа:

1. Определение режима работы передатчика.
2. Передача сообщений с использованием буфера.
3. Передача сообщения фиксированной длины.
4. Использование команды-строба.
5. Использование SPI для управления передатчиком.
6. Инициализация передатчика.

Ответ: 1; 4; 5; 6.

Система оценки

1. В вопросе №1 нужно указать в поле ввода число. За правильный ответ начисляется 2 балла.
2. В вопросе №2 нужно ответить на два пункта. За правильный ответ за каждый вопрос начисляется 1 балл.
3. В вопросе №3 нужно ответить на два пункта. За правильный ответ за каждый вопрос начисляется 1 балл.
4. В вопросе №4 нужно выбрать верное утверждение для каждого пункта. За правильный ответ начисляется 2 балла.

Максимум можно заработать 8 баллов. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 6-ти на 3-ей попытке, до 4-х на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 2 баллов и далее не убывает.

Схемотехник

Тест посвящен работе в среде KiCad. KiCad — это бесплатное ПО, предназначенное для проектирования печатных плат. На финале вам предстоит работать с этим ПО, и чтобы познакомиться с ним, данный тест сделан скорее в виде гайда, нежели в виде теста. Чтобы пройти его — советуем скачать само ПО. Скачать KiCad можно по ссылке: https://kicad.org/

В ходе теста рассматривается создание простой платы. Чтобы познакомиться с функционалом KiCad можно использовать:

- видеокурс, посвященный азам работы в данном ПО: `https://www.youtube.com/playlist?list=PLo4HNRLqCxb41tCJky3JtFFKl1m5aKhng`
- руководство по ПО: `https://docs.kicad.org/5.1/ru/getting_started_in_kicad/getting_started_in_kicad.html`

В ходе теста разбирается процесс создания простой платы с микроконтроллером и передатчиком, которые работают друг с другом по интерфейсу SPI. Эти элементы

могут являться частью пикоспутника, которым вам предстоит заниматься на финале.

Для лучшего понимания данного гайда не лишним будет знать основы схемотехники и понимать принципы построения электрических схем.

Задача II.1.2.1. Задача по физике (2 балла)

С какой относительной погрешностью изготовления α можно применить конденсатор и индуктивность в колебательном контуре радиоприемника? В колебательном контуре соединены последовательно конденсатор C = 1 мкФ и катушка L = 1 мкГн, омическим сопротивлением пренебречь. Принять, что отклонение собственной частоты контура от частоты сигнала не должно составлять более $\Delta v = 628$ Гц.

Относительной погрешностью величины А называют значение:

α = (Аноминал — Афактическое) / Аноминал , Где

Аноминал — номинальное значение величины,

Афактическое — значение величины, полученное при измерениях

Система оценки

За правильный ответ дается 2 балла. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 1.5 на 3-ей попытке, до 1 балла на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 0.5 баллов и далее не убывает.

Решение

По формуле Томсона, в колебательном контуре с индуктивностью L и емкостью C собственная частота колебаний составит:

$$v_{\text{собств}} = \frac{1}{2\pi} \cdot \frac{1}{\sqrt{LC}} = \frac{1}{2 \cdot 3.14} \cdot \frac{1}{\sqrt{10^{-6} \cdot 10^{-6}}} = 159 \text{ кГц}$$

Сравнивая величины $v_{\text{собств}}$ и Δv, видим, что они отличаются в 250 раз. Тогда:

$$\frac{v_{\text{собств}} - \Delta v}{v_{\text{собств}}} \approx \frac{v_{\text{собств}} + \Delta v}{v_{\text{собств}}} = \frac{\sqrt{LC}}{\sqrt{(1-\alpha) \cdot L \cdot (1-\alpha) \cdot C}}$$

$$1 + \frac{\Delta v}{v_{\text{собств}}} = \frac{1}{1-\alpha}$$

Окончательно,

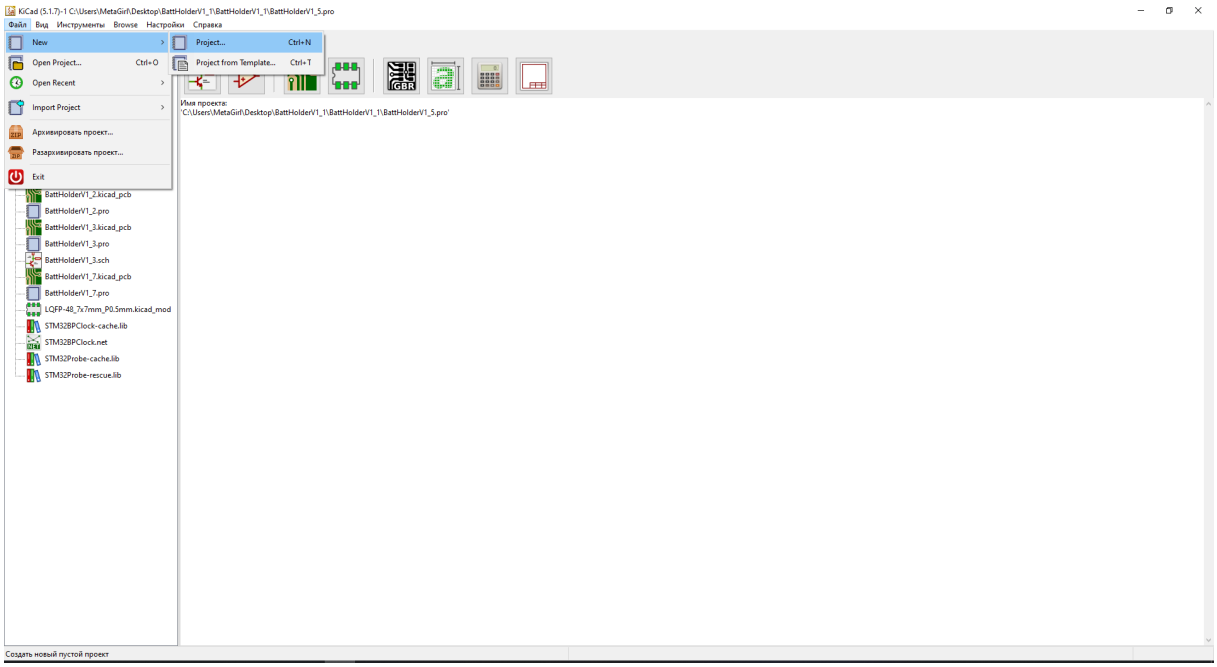
$$\alpha = 1 - \frac{v_{\text{собств}}}{v_{\text{собств}} + \Delta v} = 1 - \frac{\frac{1}{2\pi} \cdot \frac{1}{\sqrt{LC}}}{\frac{1}{2\pi} \cdot \frac{1}{\sqrt{LC}} + \Delta v} = 1 - \frac{1}{1 + \Delta v \cdot 2\pi \cdot \sqrt{LC}} ==$$

$$== 1 - \frac{1}{1 + 628 \cdot 6,28 \cdot \sqrt{10^{-6} \cdot 10^{-6}}} = 1 - \frac{1}{1 + 39 \cdot 10^{-4}} \approx 4 \cdot 10^{-3}$$

Ответ: $\alpha = 4 \cdot 10^{-3}$.

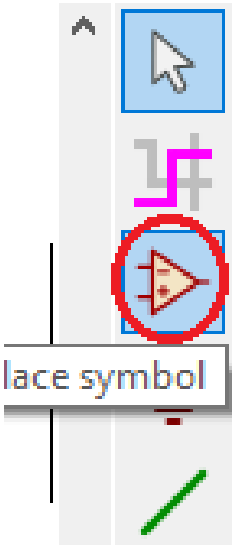
Задача II.1.2.2. Начало работы. Создание принципиальной схемы. (2 балла)

При установке KiCad устанавливается сразу несколько программ. Файл После установки запустите KiCad. В открывшемся окне выберите Файл → New → Project.

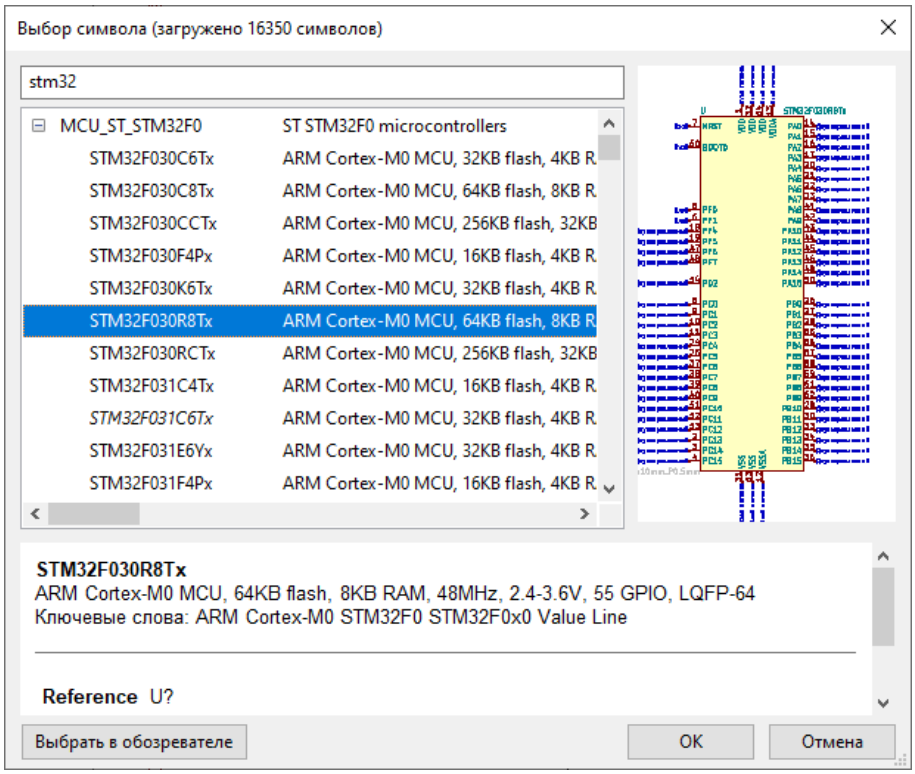


После этого нужно будет ввести название проекта. Далее кликните по иконке проектировщика принципиальных схем.

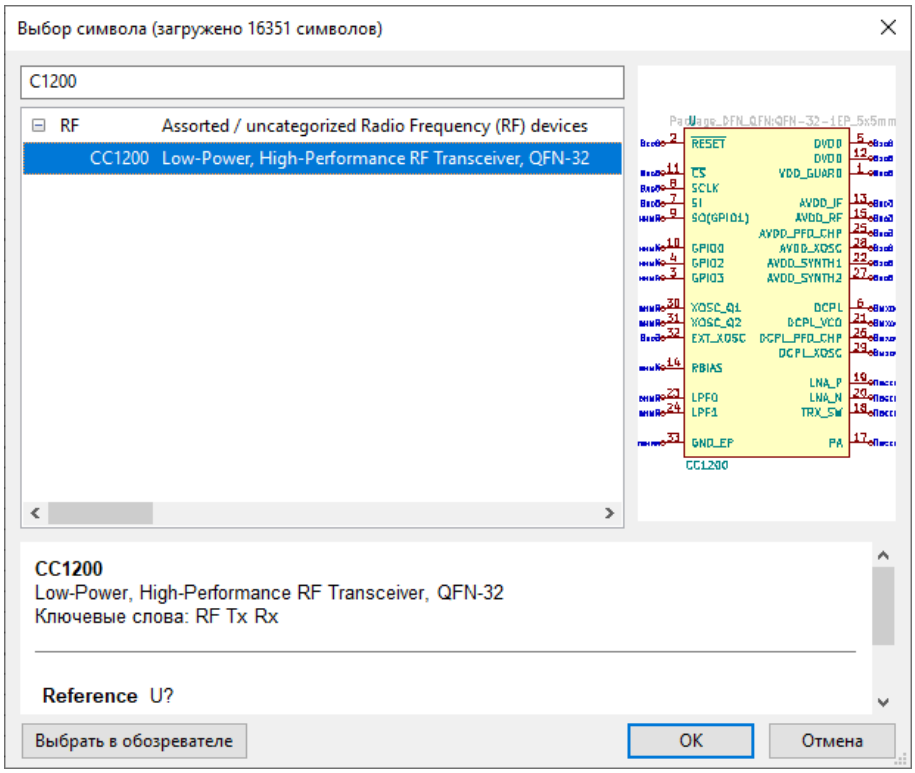
Откроется новый документ. В этом документе можно начать создавать принципиальную схему будущей платы. Справа в вертикальном меню выберите значок операционного усилителя.



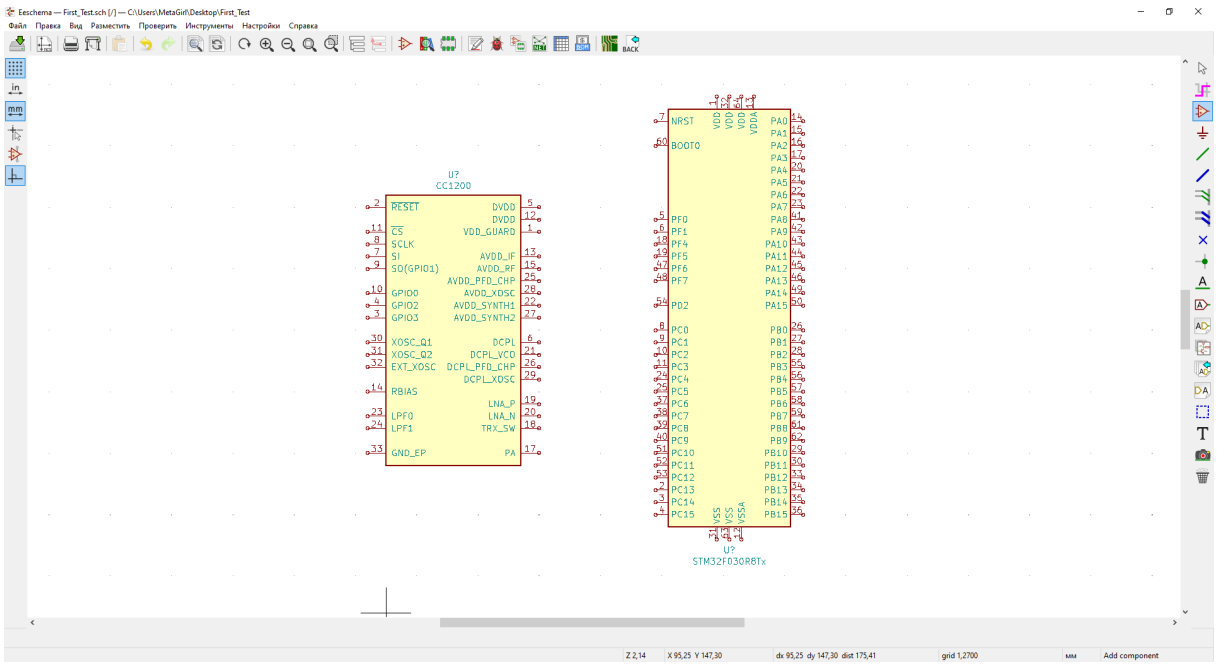
Эта опция позволяет разместить в любом месте документа элемент. Кликните на любое место на листе. После этого возможно потребуется подождать, пока загрузятся библиотеки. В открывшемся окне выберите схему микроконтроллера STM32F030R8Tx.



Далее кликните на листе там, где хотите расположить схему выбранного микроконтроллера. Этот микроконтроллер будет управляющим звеном в проектируемой схеме. Добавим еще один элемент на схему — чип приемопередатчика CC1200.



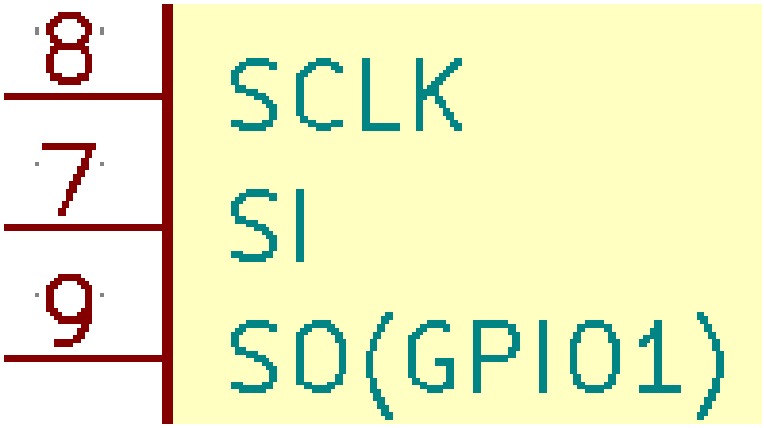
На самом деле на финале вам предстоит работать с похожим чипом CC1101, однако его по умолчанию нет в библиотеках KiCad, поэтому для простоты можно пока использовать CC1200. Также кликните на место на листе, где хотите расположить схему элемента.



У каждой из схем есть множество выводов. Для нас важно сейчас то, что чип CC1200 управляется по интерфейсу SPI. У этого интерфейса 3 линии, по которым он работает:

1. MOSI — master output, slave input — линия данных, по которой данные идут от управляющего устройства к подчиненному. Например, команда от микроконтроллера на отображение данных на экране передается по этой линии.
2. MISO — master input, slave output — линия данных, по которой данные поступают к управляющему устройству от подчиненного. Например, чтение данных с датчика происходит по этой линии.
3. SCK — линия, отвечающая за синхронизацию управляющего и подчиненного устройств.

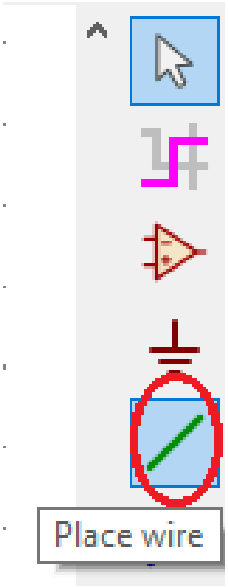
Похожие подписи можно увидеть на схеме чипа CC1200, где SCLC — это SCK, SI — это MOSI, а SO — MISO.



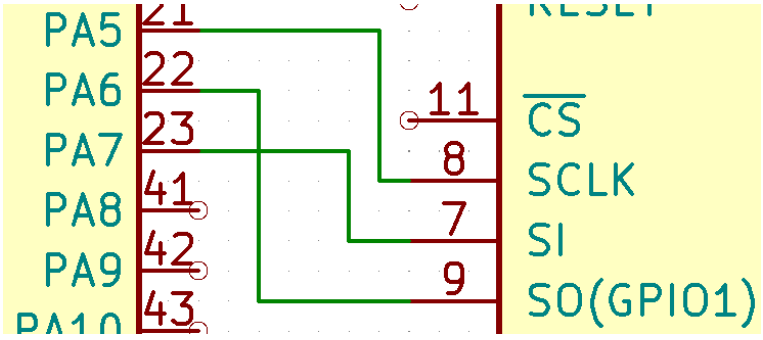
Интерфейс SPI на микроконтроллере инициализируется программно на определенных выводах (пинах). Информация об этом находится в документации на микроконтроллеры (datasheet). Ниже приведены скриншоты из datasheet на микроконтроллер STM32F030R8Tx.

PA4	I/O	TTa	-	SPI1_NSS, USART1_CK(2) USART2_CK(3)(5), TIM14_CH1, USART6_TX(5)
PA5	I/O	TTa	-	SPI1_SCK, USART6_RX(5)
PA6	I/O	TTa	-	SPI1_MISO, TIM3_CH1, TIM1_BKIN, TIM16_CH1, EVENTOUT USART3_CTS(5)
PA7	I/O	TTa	-	SPI1_MOSI, TIM3_CH2, TIM14_CH1, TIM1_CH1N, TIM17_CH1, EVENTOUT

SCK — это пин PA5, MISO — PA6, а MOSI — PA7. Теперь можно соединить соответствующие пины двух микросхем, чтобы обеспечить их соединение по SPI. Для этого справа в вертикальном меню выберите зеленую линию.

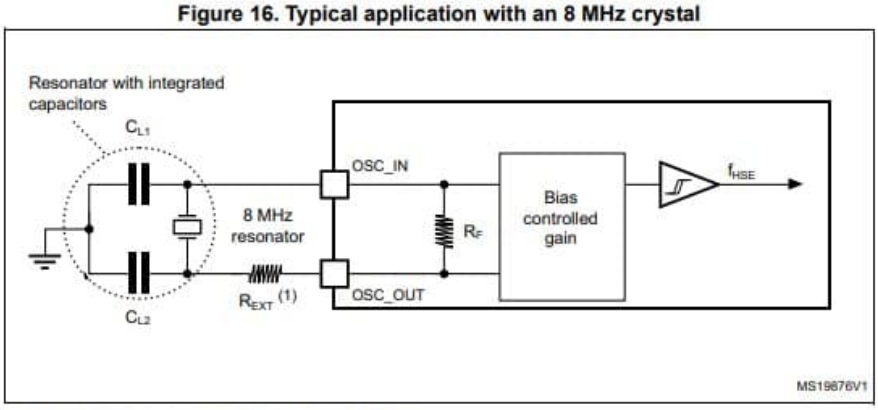


Теперь можно соединить соответствующие пины между собой.



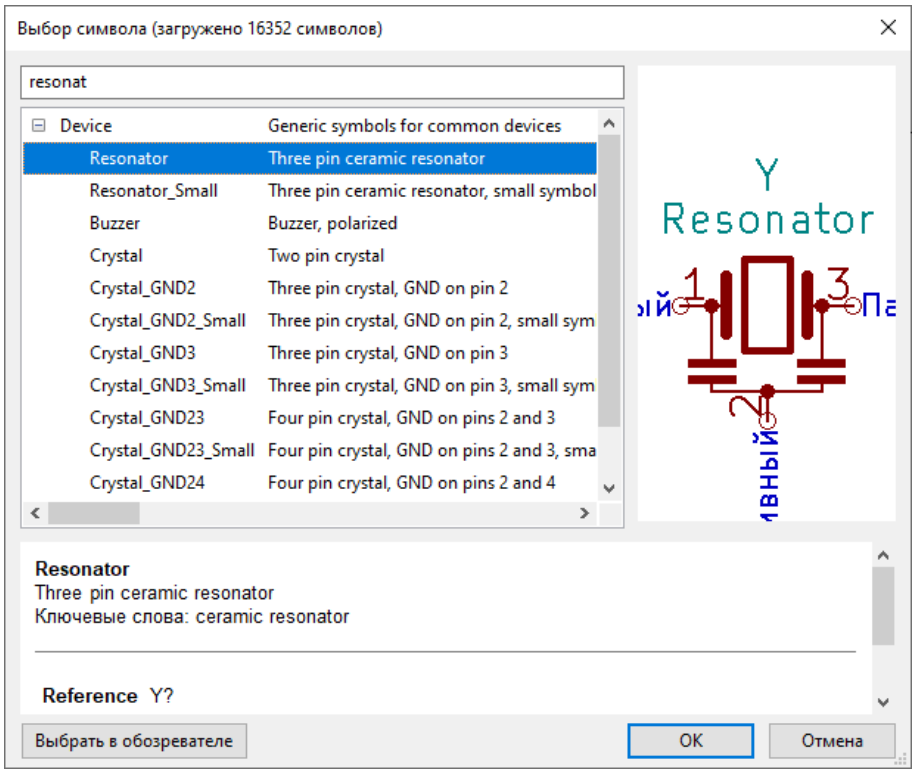
Соединение по SPI обеспечено. Однако для самого микроконтроллера нужны дополнительные компоненты для корректной работы. Прежде всего — кварцевый резонатор для обеспечения точного тактирования. В том же datasheet есть информация о том, какой резонатор необходим для микроконтроллера. Схема резонатора показана на скрине.

Figure 16. Typical application with an 8 MHz crystal

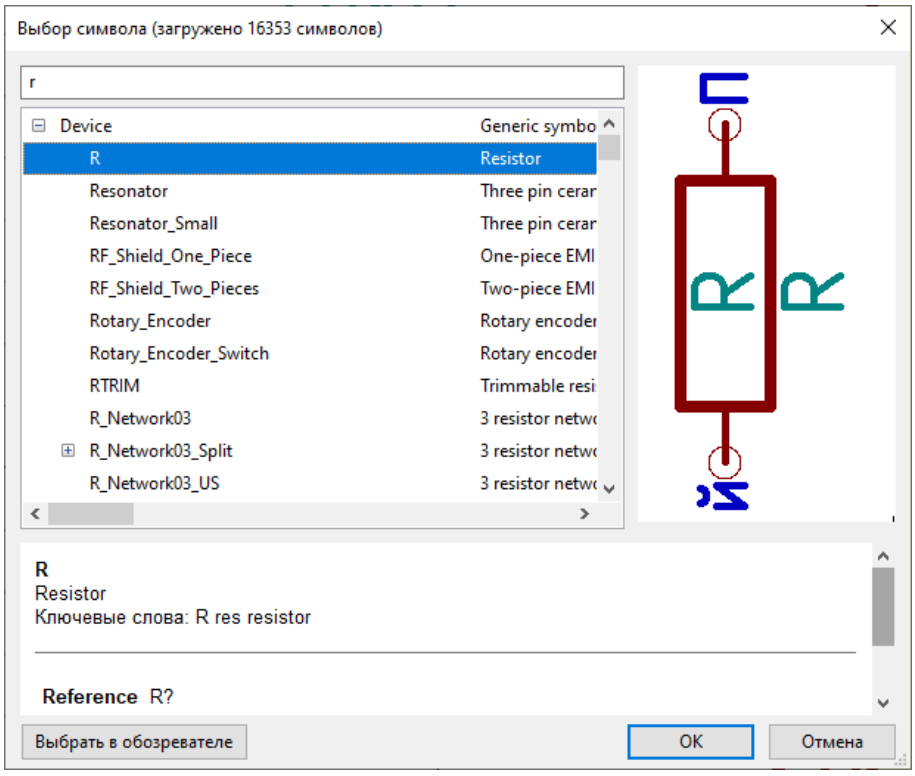


1. R_{EXT} value depends on the crystal characteristics.

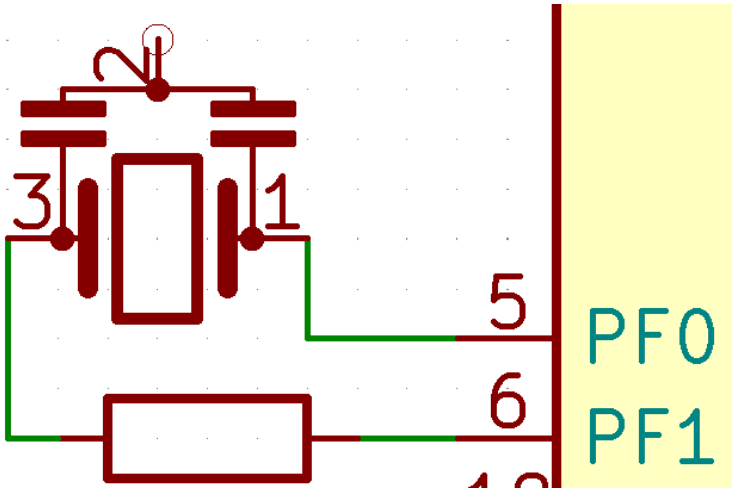
Здесь OSC_IN и OSC_OUT — пины PF0 и PF1 соответственно. Снова переключитесь на значок операционного усилителя и кликните на любую точку на листе документа. В появившемся окне найдите резонатор.



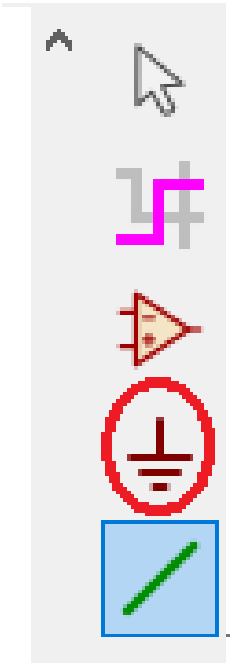
Добавьте резонатор на лист. Помимо резонатора в схеме из datasheet отмечен резистор Rext. Для размещения резистора повторите операцию по размещению элемента, в поиске впишите «R» и выберите стандартное обозначение сопротивления.



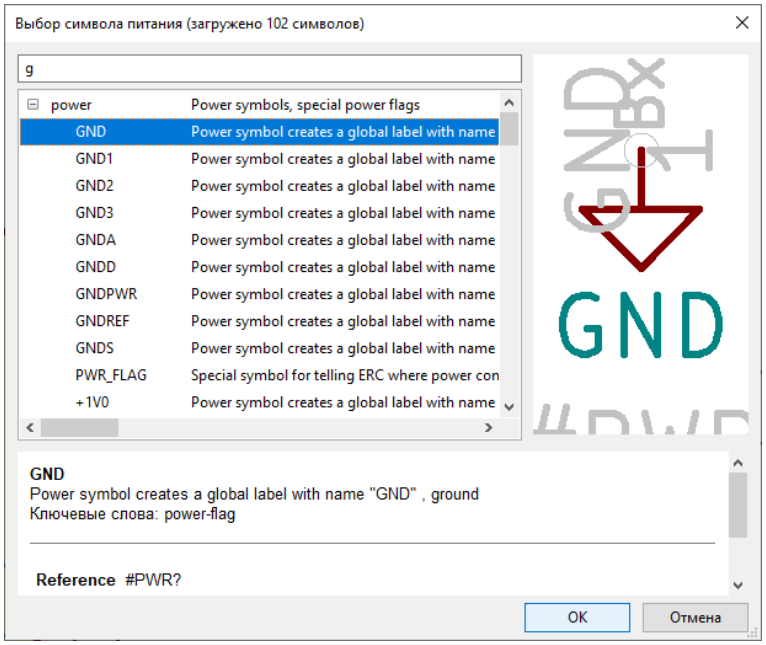
После размещения всех элементов можно повторить схему из datasheet.



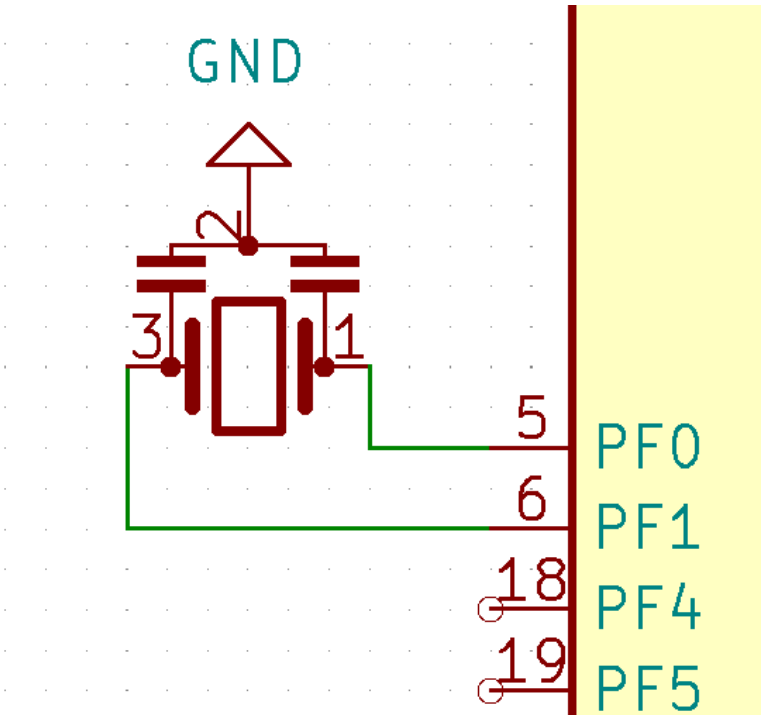
Не будем сейчас проставлять номиналы конденсаторов и резисторов, однако нужно понимать, что это неотъемлемая часть при создании платы. Выход 2 необходимо заземлить в соответствии со схемой из datasheet. Для этого справа в вертикальном меню выберите значок заземления.



Кликните также на любую точку документа и в открывшемся окне в поиске напишите «GND».

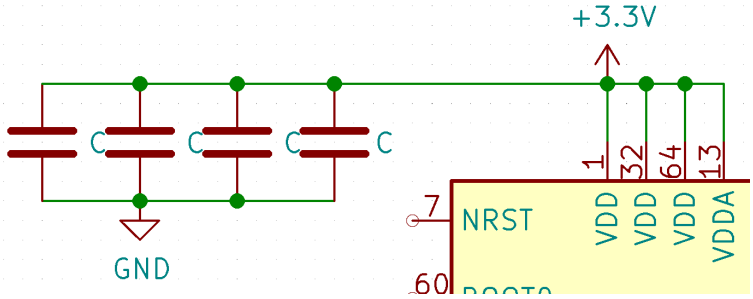


Выберите обозначение заземления и дополните схему.

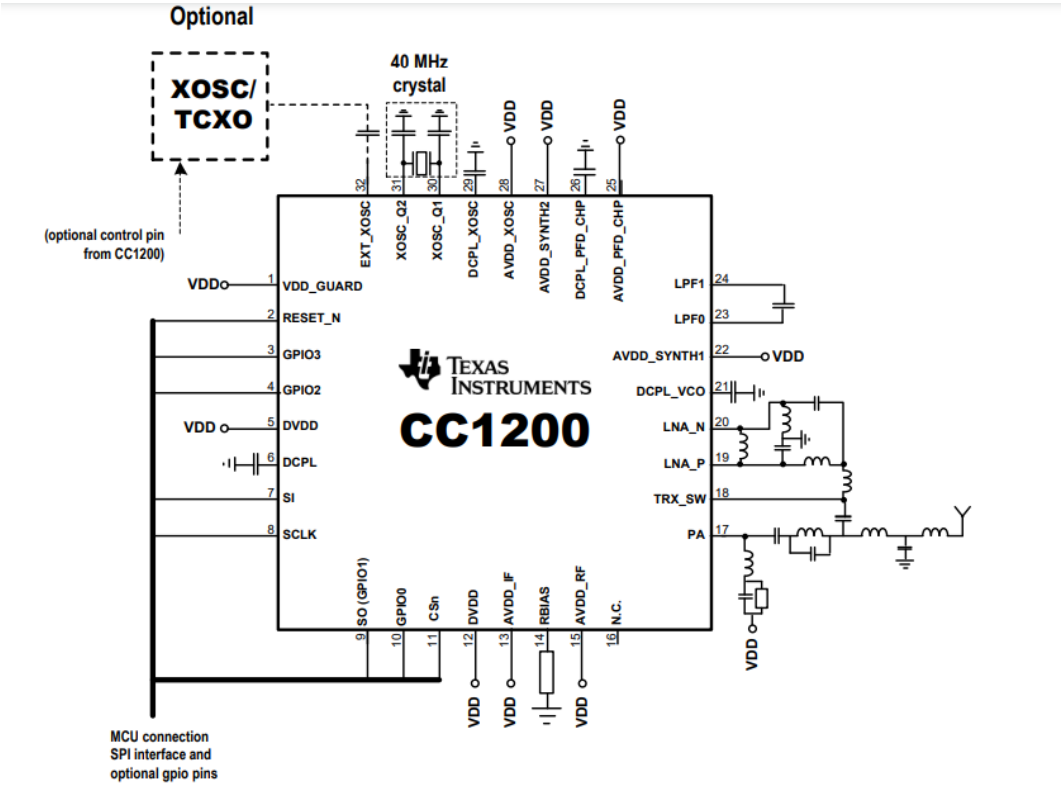


Помимо этого, необходимо подсоединить к земле все пины на схемах элементов, подписанные GND или VSS. А все пины с обозначением VDD — к питанию 3.3 V. Подключение к питанию осуществляется тем же образом, что и подключение к GND, только элемент в этом случае нужно выбрать 3.3 V.

Перед каждым выводом к питанию необходимо разместить конденсатор.



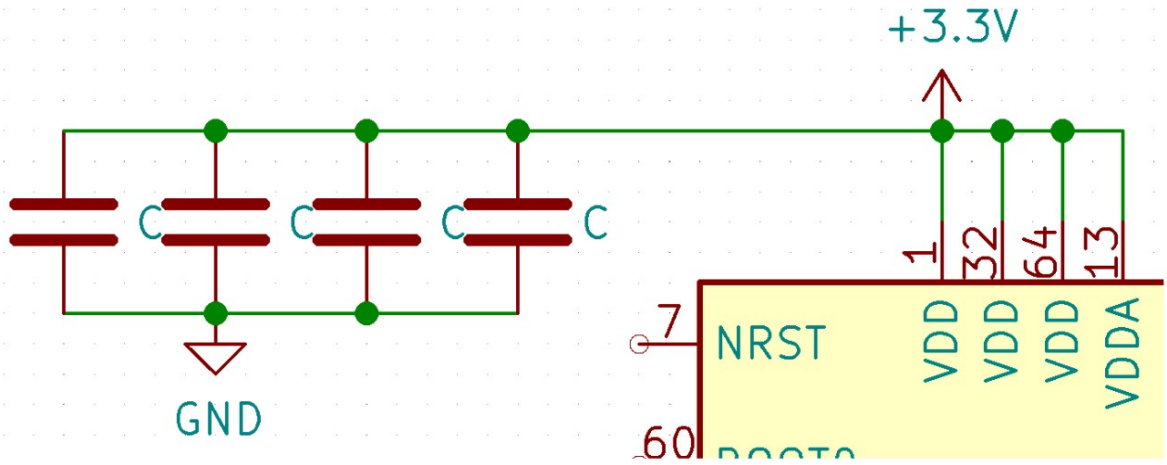
Ниже приведена обвязка для чипа CC1200.

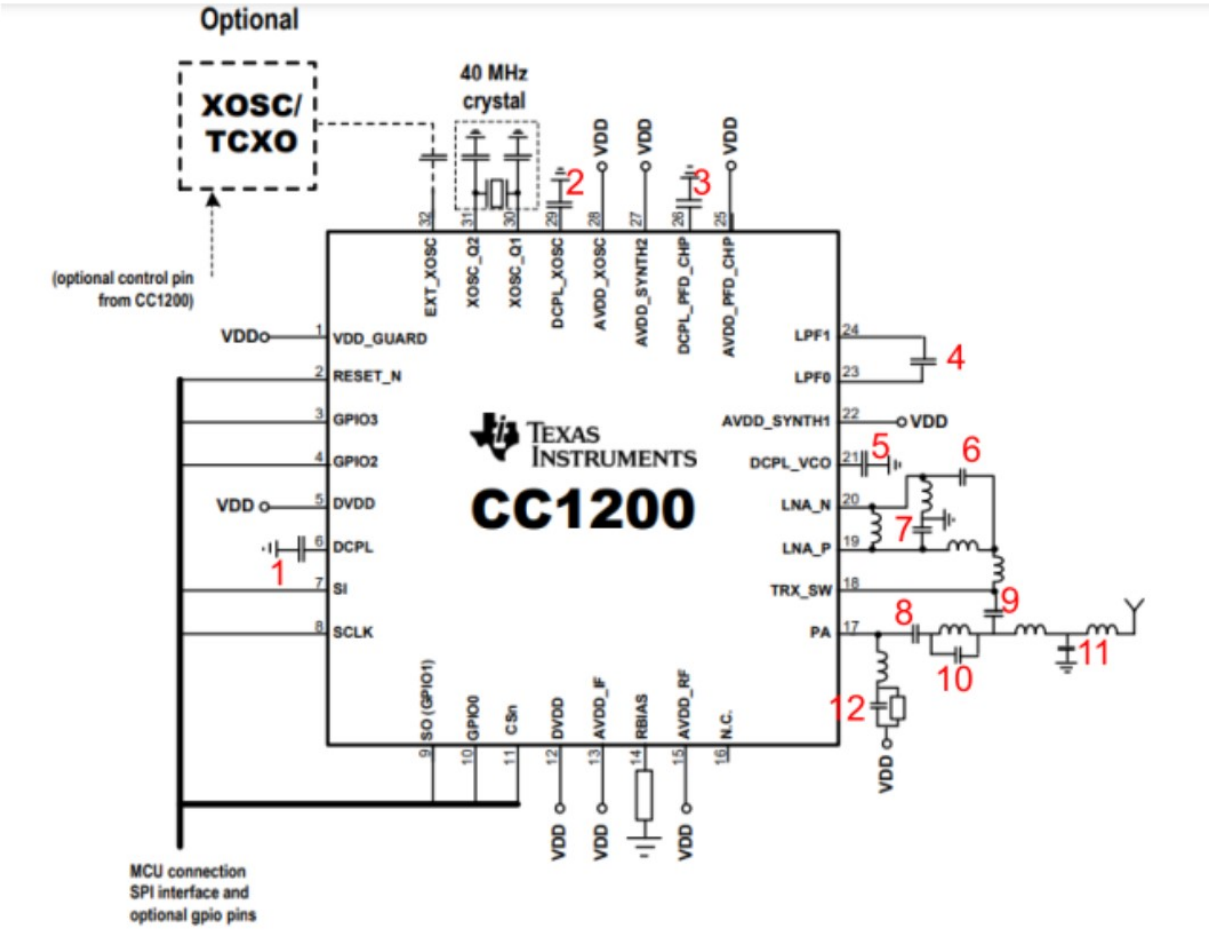


Сколько на схеме необходимо разместить конденсаторов?

Решение

На схеме микроконтроллера к питанию подключается 4 конденсатора, как показано на скриншоте:

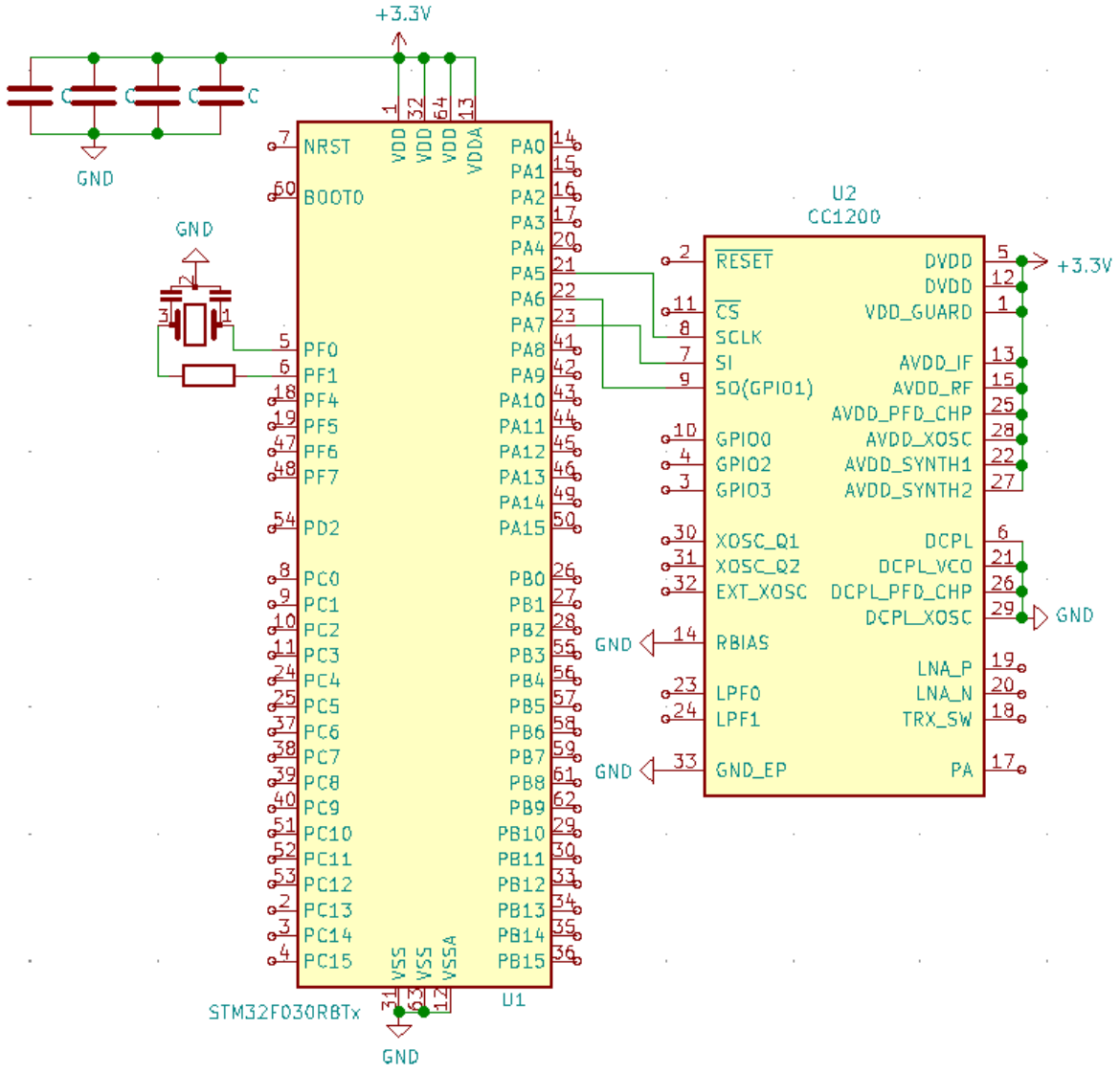




Ответ: 16.

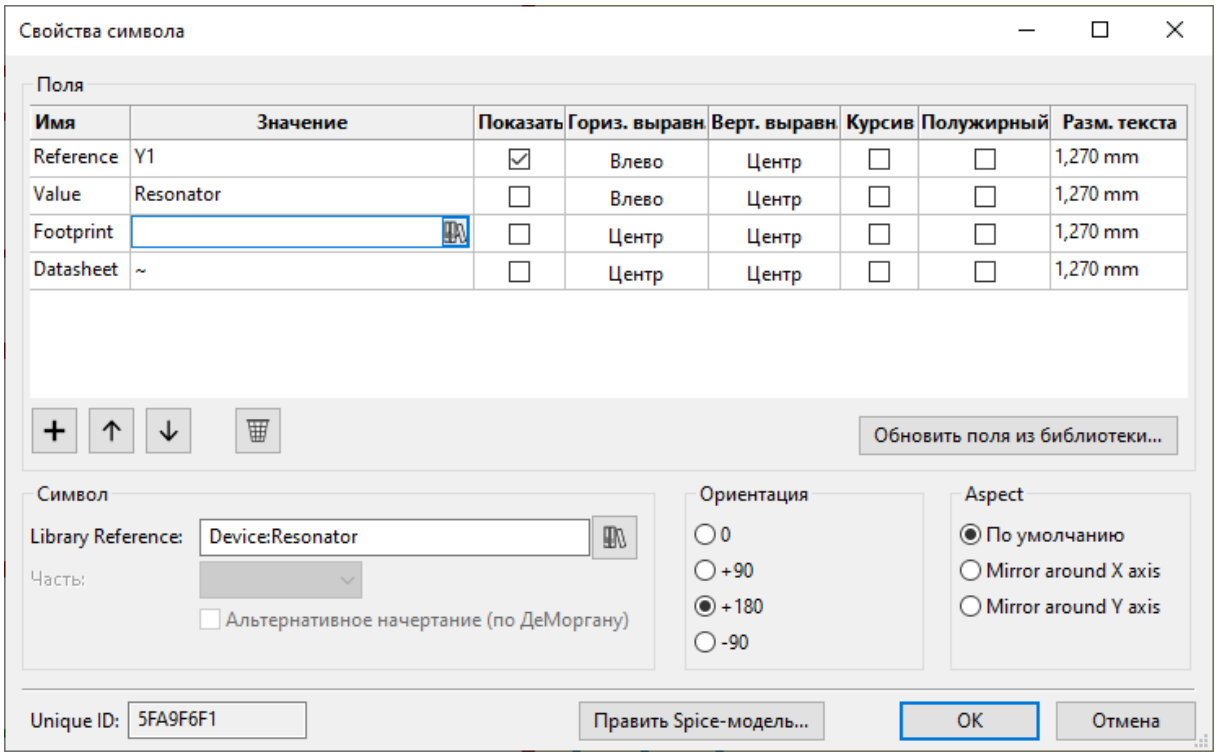
Задача II.1.2.3. Посадочные места (2 балла)

Мы не будем на текущем этапе размещать повторять всю необходимую обвязку для каждого из элементов, однако необходимо понимать, что при проектирование платы этот шаг является неотъемлемым. Соедините как показано на скриншоте пины питания и земли.

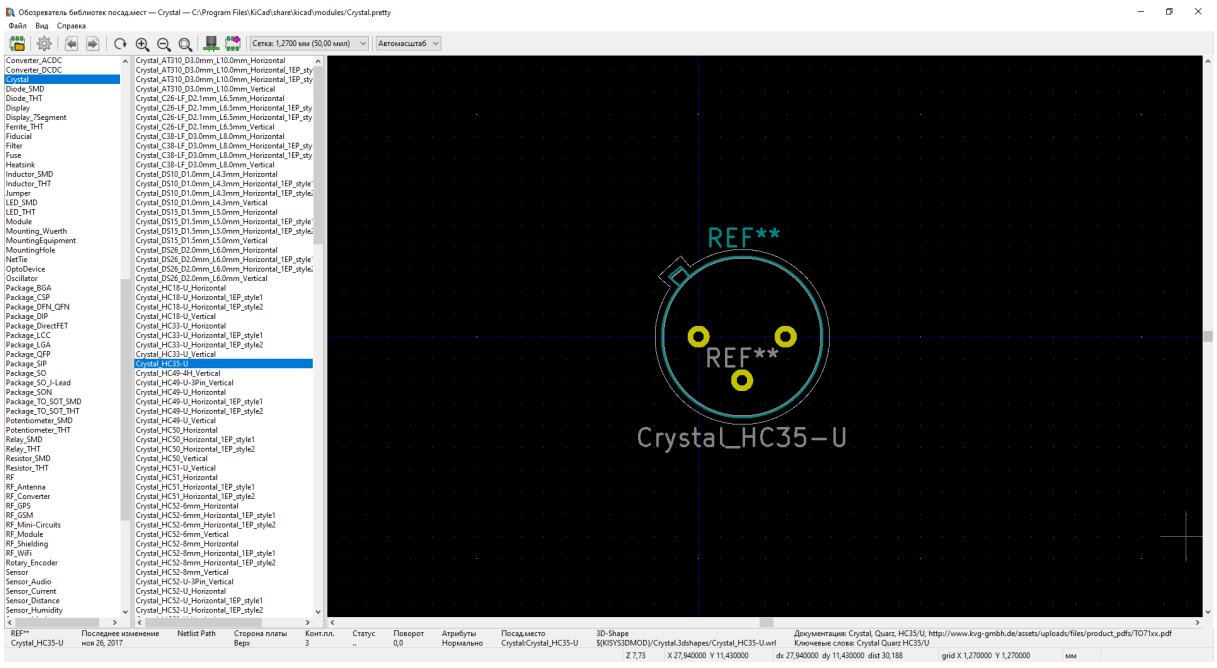


После создания принципиальной схемы следующим этапом является выбор посадочных мест компонентов. Каждый использующийся в схеме элемент может изготавливаться в различных корпусах. Например, микроконтроллер может изготавливаться в трех разных корпусах. При проектировании платы определяется, в каком именно корпусе будет использоваться элемент при монтаже. Контур этого элемента будет использоваться также при прокладывании дорожек платы.

Если дважды кликнуть на изображение резонатора, то откроется новое окно. Посадочное место определяет поле Footprint.

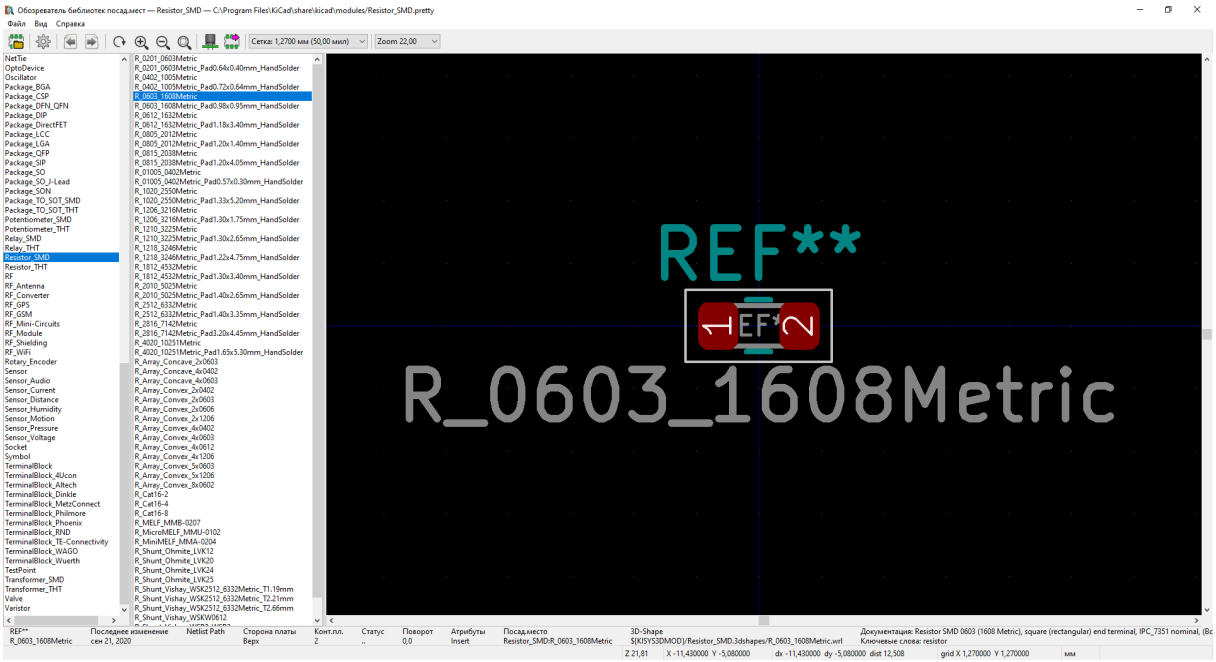


Кликните по значку со стопкой книг. Снова откроется новое окно, в котором будет список посадочных мест, имеющихся в KiCad по умолчанию. В самом левом списке найдите наименование Crystal и кликните на него. Откроется еще один список.



Во втором списке найдите посадочное место с названием «Crystal:Resonator_SMD _muRata_CSTxExxV-3Pin_3.0x1.1mm». Дважды кликните по его названию, чтобы добавить посадочное место к элементу.

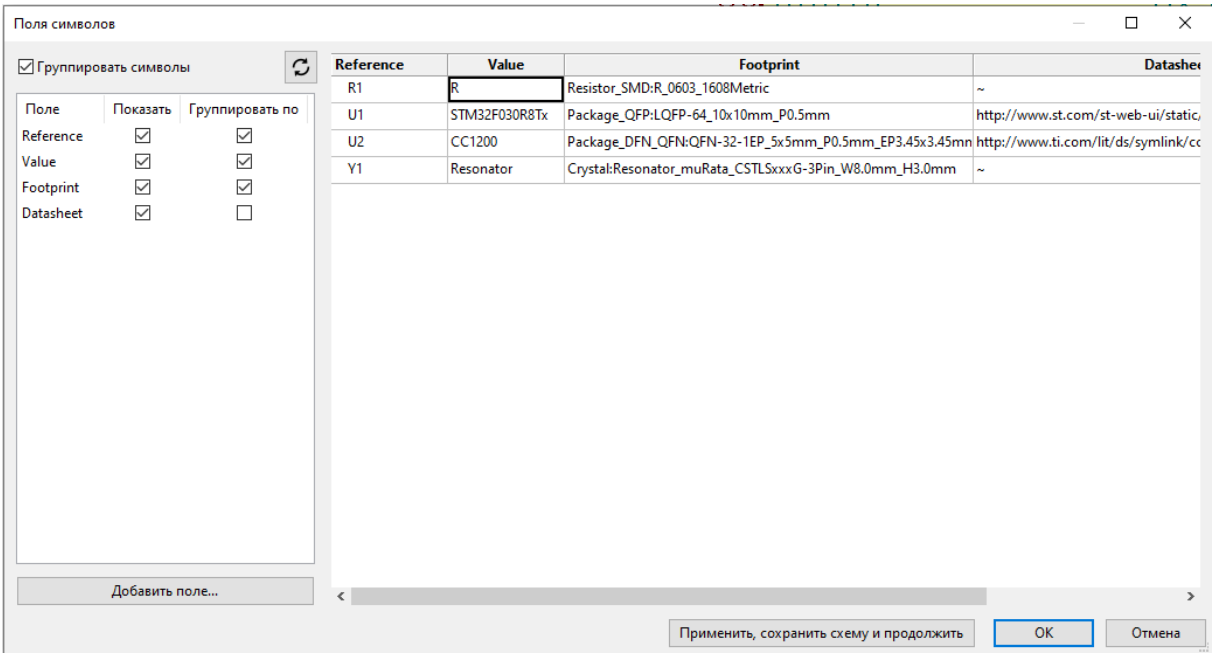
Повторите операцию для резистора.



Выберите посадочное место под названием «Resistor_SMD:R_0603_1608Metric». Также дважды кликните по названию посадочного места.

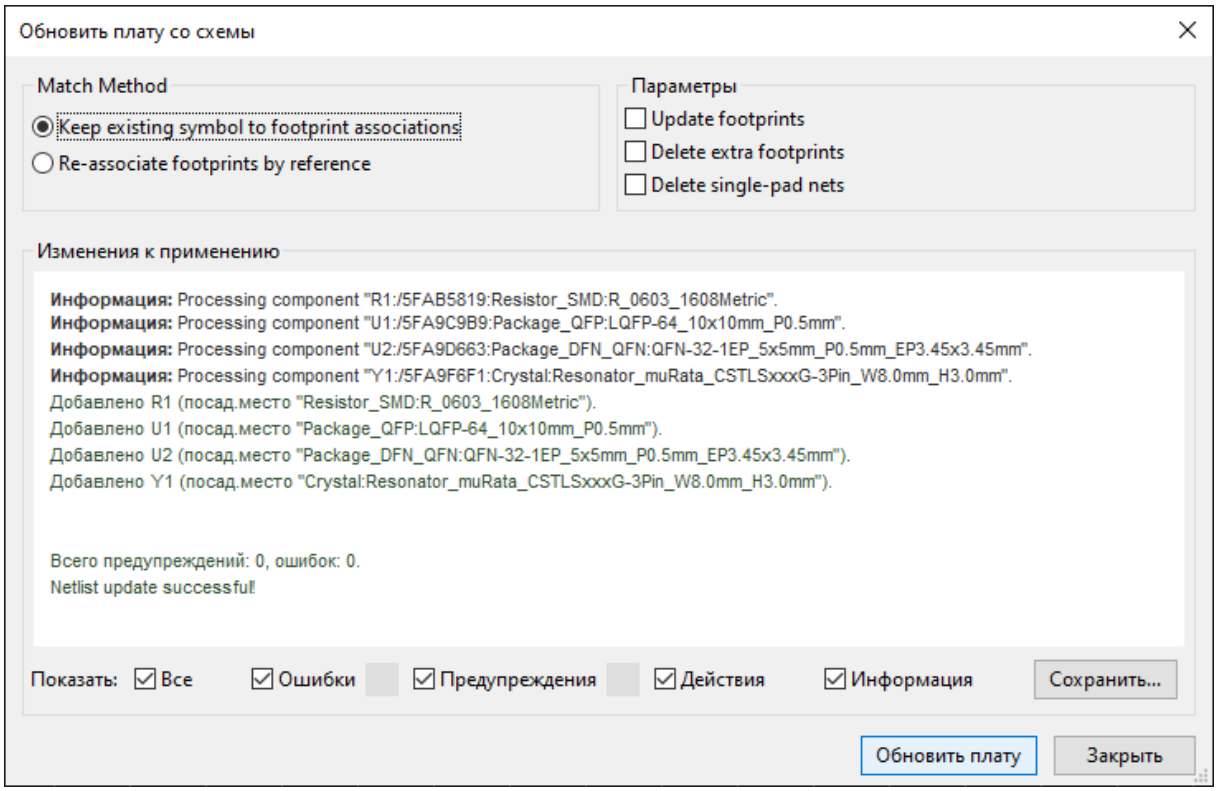
Для конденсаторов определите место «Capacitor_SMD:C_0201_0603Metric» (в категории Capacitor).

Для чипа CC1200 и микроконтроллера посадочные места определены по умолчанию. Вы можете проверить это, кликнув на значок



Перед следующим шагом нужно убедиться, что поле Footprint заполнено у каждого элемента.

Далее нажмите на значок в верхней панели меню. Откроется новое окно — редактор печатных плат. В новом окне в верхнем меню нажмите на значок. Появится новое окно.

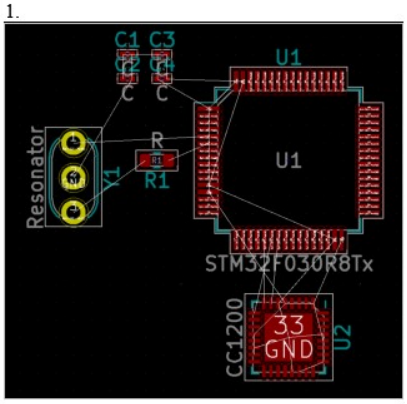


Нажмите «Обновить плату». После этого компоненты появятся на листе. Кликните в любое место, чтобы разместить их. Изначально, все элементы расположены кучно. Необходимо немного разнести их для дальнейшей работы.

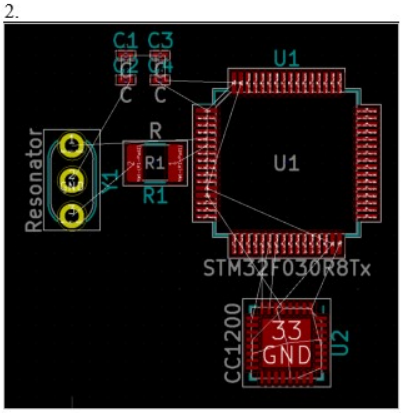
Какой из видов схемы соответствует обозначенным посадочным местам?

Варианты ответа:

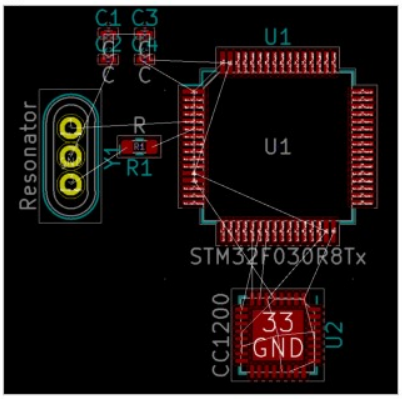
1.



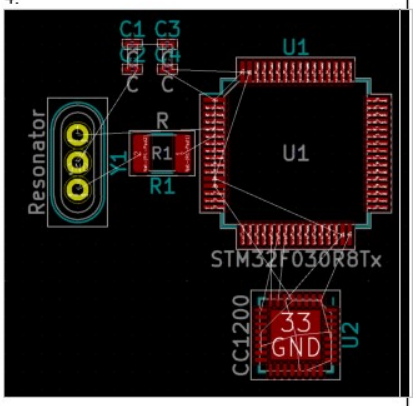
2.



3.

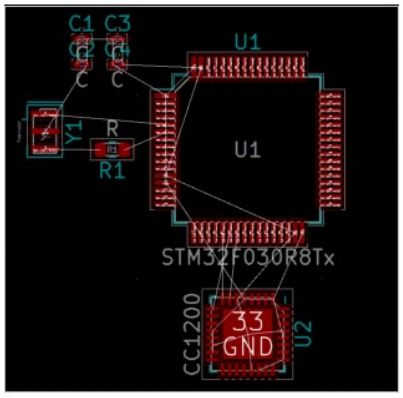


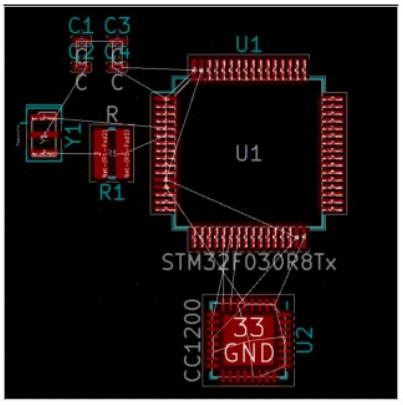
4.



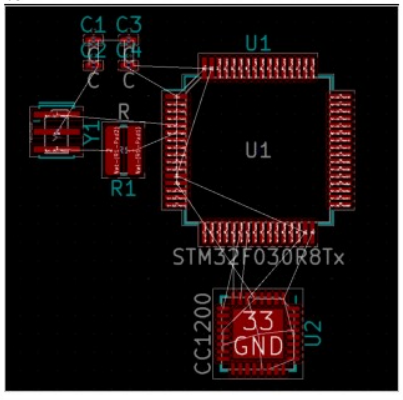
5.

6.

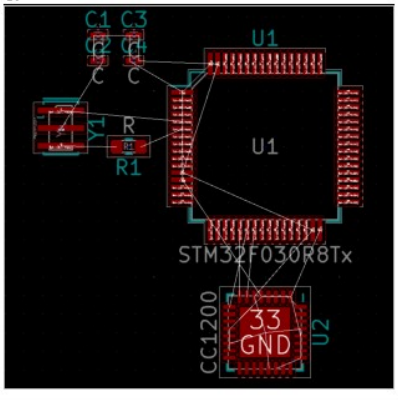




7.



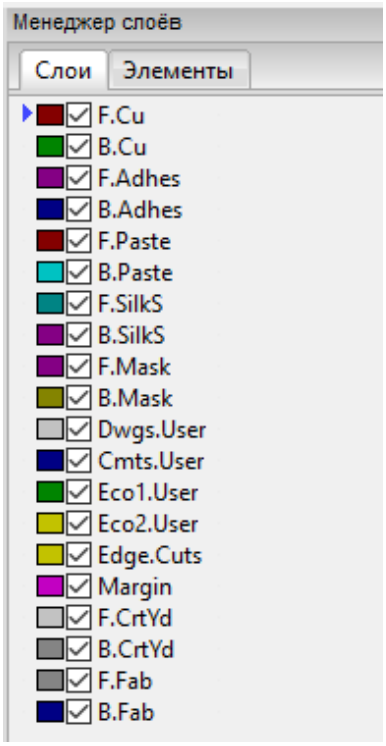
8.



Ответ: 5.

Задача II.1.2.4. Трассировка, часть 1 (2 балла)

Следующим этапом является трассировка платы. Основной материал платы — текстолит. Материалом, проводящим электричество является медь. Все соединения определяются проведенными на текстолите медными дорожками. Медных слоев, из которых формируются дорожки, на плате может быть несколько. По умолчанию в KiCad их задано два — верхний и нижний. Слои платы задаются отдельными проектировочными слоями, которые называются F.Cu и B.Cu. Вы можете их увидеть справа, за вертикальным меню.



При наведении курсора на слой пишется его назначение.

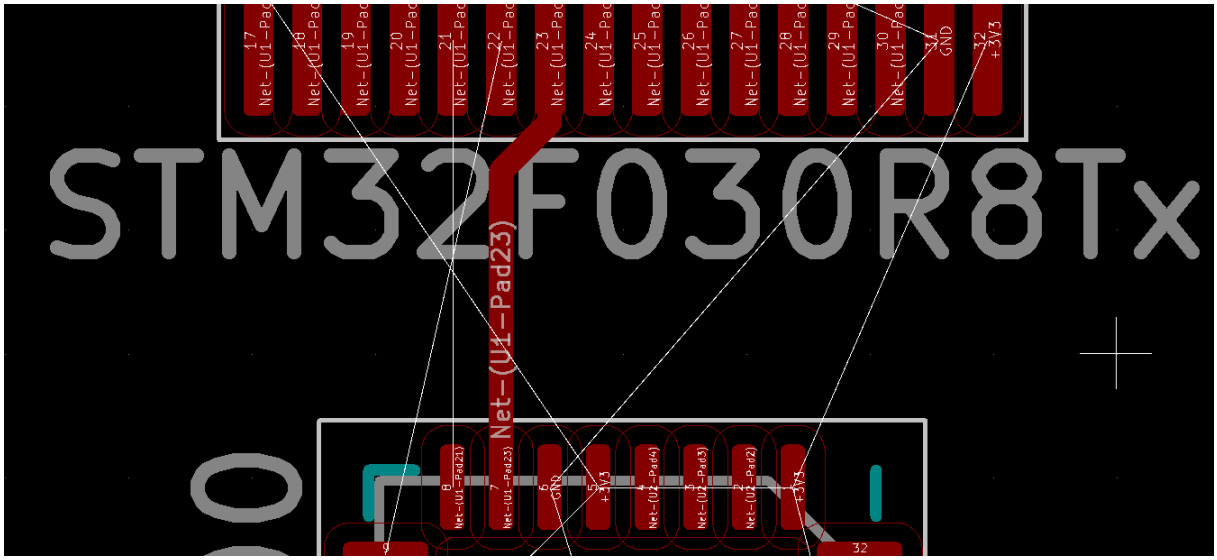
Какой по счету слой отвечает за шелкографию (надписи на плате) сверху платы?

Ответ: 7.

Задача II.1.2.5. Трассировка, часть 2 (2 балла)

Пока нас интересуют только медные слои. В редакторе печатных плат все заданные на принципиальной схеме соединения на данный момент показаны тонкими белыми линиями. Чтобы начать разводить медные дорожки выберите в правом вертикальном меню функцию прокладывания дорожек.

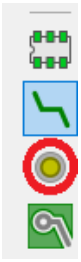
Кликните на один из пинов микроконтроллера, определенных под SPI. Появится линия и подсветится пин на CC1200, с которым необходимо соединить выбранный пин микроконтроллера. Проведите дорожку.



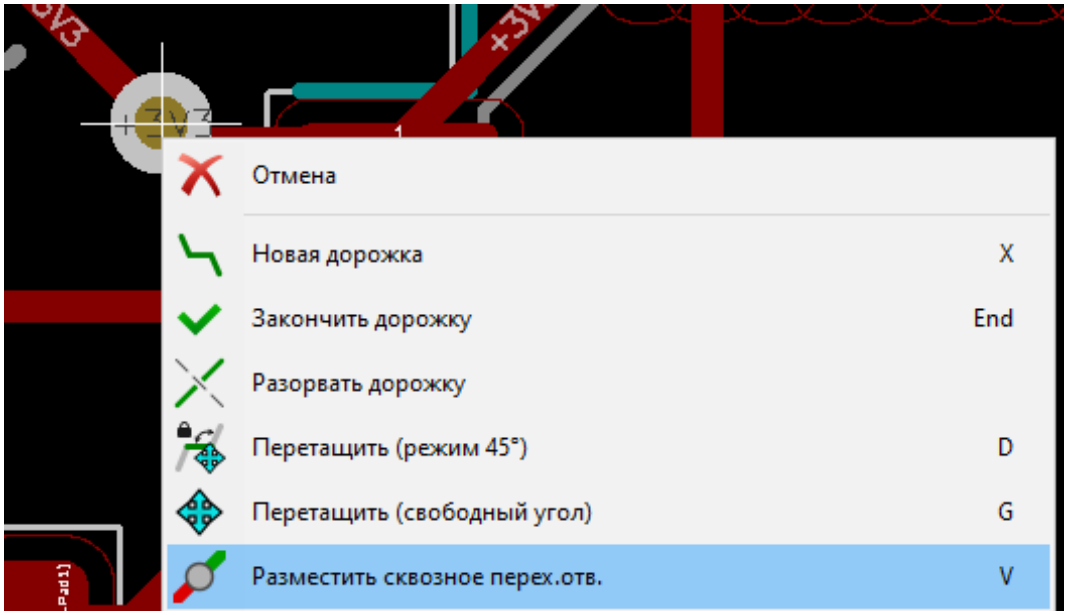
Повторите это действие для двух других пинов SPI. Можно увидеть, что пара неразведенных соединений пересекается. Если после выбора первого пина сразу навести курсор на второй, то редактор сам предложит вариант прокладывания дорожки. Допустимо прокладывание дорожки в контуре элемента, что решает проблему с пересечением.

Соедините подобным образом резонатор и конденсаторы с микроконтроллером.

При разведении дорожек питания и заземления вы можете столкнуться с тем, что это трудно, а местами невозможно сделать. Здесь на помощь приходит второй медный слой. Существует такое понятие, как переходное отверстие (via). Такое отверстие позволяет перевести соединение на другой слой платы. Чтобы разместить переходное отверстие, выберите соответствующий значок в вертикальном меню справа.

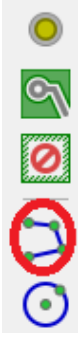


Затем кликните на то место, где хотите его расположить. Чтобы провести линию на другом медном слое кликните правой кнопкой мыши по переходному отверстию и выберите пункт «Разместить сквозное переходное отверстие».



При разведении контактов заземления обратите внимание, что на чипе CC1200 есть точка, к которой можно заземлить контакты.

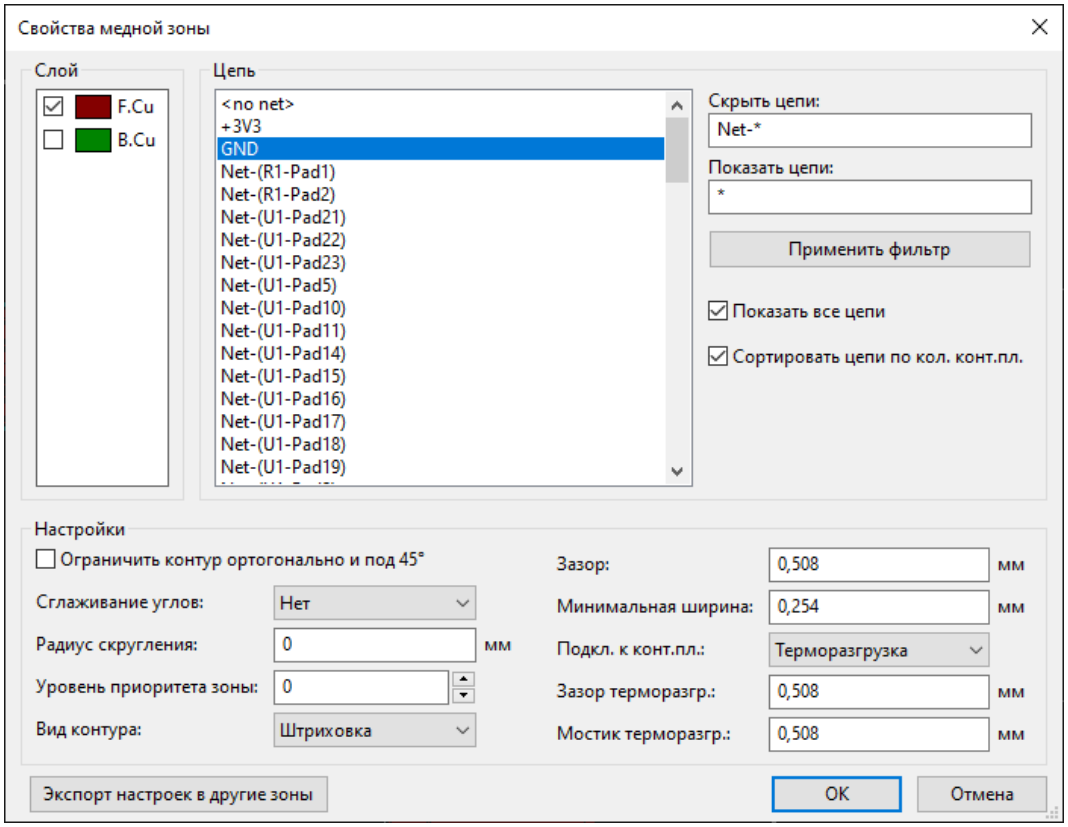
Когда вы закончите с разведением дорожек, следующим действием необходимо обрисовать контур платы. Это можно сделать с помощью соответствующего инструмента, который также можно выбрать в вертикальном меню справа.



Выберите его и обрисуйте замкнутый контур.

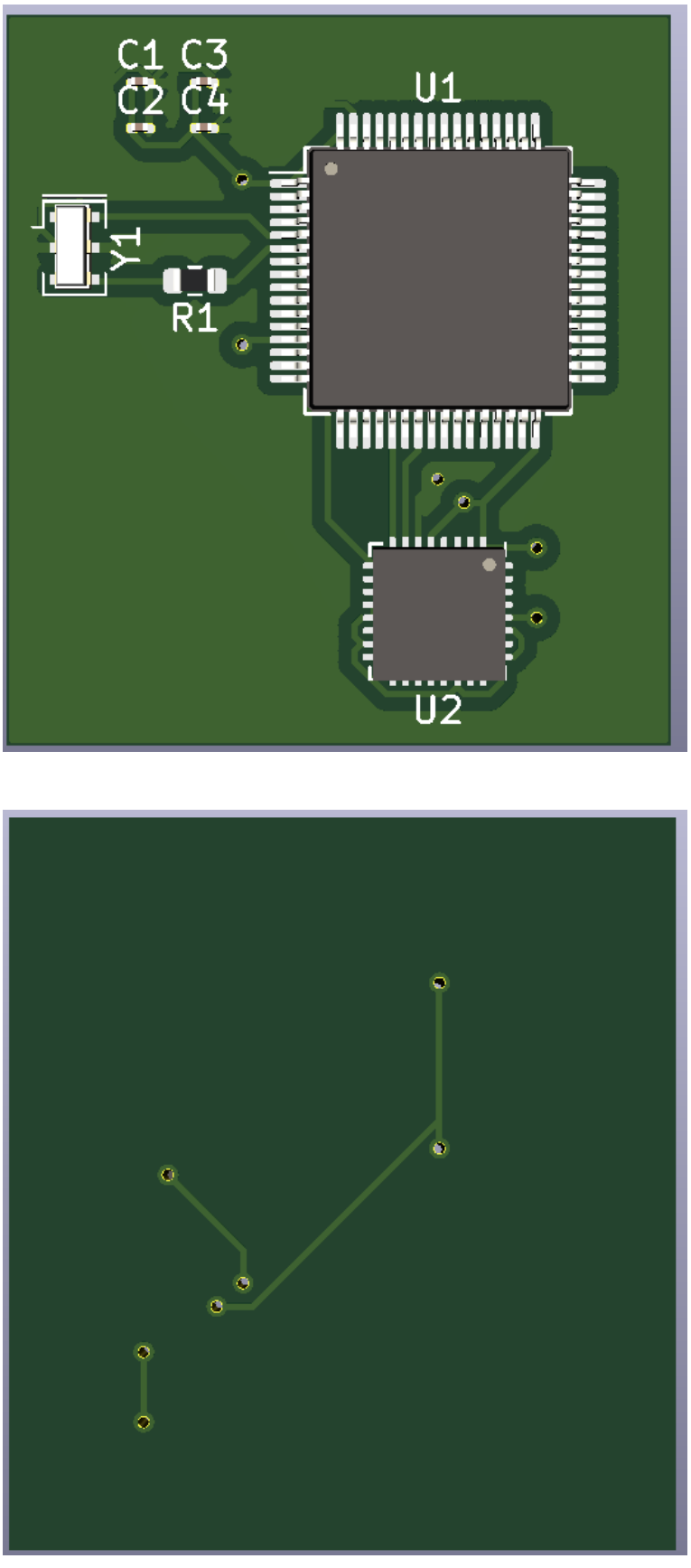
После этого можно сделать так называемый полигон. Полигон — это обширный слой меди, который используют для создания зоны питания или заземления. Выберите в вертикальном меню справа значок для создания полигона.

Затем кликните на один из углов контура платы. Появится окно, в котором необходимо указать слой, на котором будет создаваться полигон и тип соединения (GND или 3.3V).



Обратите внимание, что контактная площадка с заземлением на чипе CC1200 должна соединяться с полигоном.

После всех действий можно посмотреть в 3d-визуализации на получившийся результат. Для этого в верхней панели меню перейдите Вид →3D Viewer. В чем ошибка на приведенном 3D-виде?



Варианты ответа:

1. Нет одного из компонентов.
2. Неправильно используются переходные отверстия.
3. Нет соединения CC1200 с полигоном.

4. Полигон размещен не на том слое платы.
5. Ошибки нет, все правильно.
6. Полигон заходит на дорожки.
7. Неправильно подключены конденсаторы к пинам питания.

Ответ: 7.

Система оценки

1. В вопросе №1 нужно указать в поле ввода число. За правильный ответ начисляется 2 балла.
2. В вопросе №2 нужно выбрать правильный ответ. За правильный ответ начисляется 2 балла.
3. В вопросе №3 нужно указать в поле ввода число. За правильный ответ начисляется 2 балла.
4. В вопросе №4 нужно выбрать правильный ответ. За правильный ответ начисляется 2 балла.

Максимум можно заработать 8 баллов. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 6-ти на 3-ей попытке, до 4-х на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 2 баллов и далее не убывает.

Спутникостроитель

Тест посвящен работе с STM32CubeIDE — средой разработки для микроконтроллеров семейства STM32. С данной IDE вам предстоит более плотно работать на финале. С помощью нее необходимо будет реализовать алгоритмы управления спутником, работающим на базе микроконтроллера STM32F303CCT6.

Для того, чтобы пройти данный тест, необходимо скачать и установить на ПК программное обеспечение для работы с микроконтроллерами семейства STM32 — STM32CubeIDE. Скачать ПО можно по ссылке:

https://www.st.com/en/development-tools/stm32cubeide.html

ПО бесплатное, однако требует регистрации на сайте.

Задания-тесты сделаны в большей степени в виде гайда, который мы вам предлагаем пройти, чтобы познакомиться с ПО. Тесты не являются сложными, но чтобы ответить на вопросы, необходимо выполнить несколько последовательных действий в самой IDE. Чтобы прохождение гайдов не вызывало больших затруднений желательно иметь представлении о программировании на C/C++ и начальное понимание о работе микроконтроллеров.

Полезным будет ознакомиться также с практикумом прошлого года, в котором рассматривается работа с платой Nucleo на базе микроконтроллера F303RE:

https://drive.google.com/file/d/10J6GTLQtct9am54CZvkGh2V4ciAoF2cG/view?usp=sharing

Если вы никогда не работали с микроконтроллерами, то советуем вам для начала

немного ознакомиться с программированием микроконтроллеров на примере плат Arduino. В интернете можно найти много курсов на эту тему, вот один из них:

https://stepik.org/course/69511/syllabus

Если вам не знаком язык программирования C, то также можно пройти любой курс, например:

https://stepik.org/course/80538/promo

Или вы можете попробовать во всем разобраться сходу, такое тоже возможно.

Задача II.1.3.1. Задача по физике (2 балла)

Спутник, снимающий Землю, обладает способностью перенацеливаться на интересующие объекты на Земле. Для осуществления вращения он оснащен гиродинами (гироскопами-маховиками). Раскрутка маховика приводит к вращению спутника вокруг его оси. Торможение маховика влечет за собой, соответственно, замедление вращения спутника. Маховик раскручивается электродвигателем, питающимся от аккумулятора. Торможение маховика осуществляется переключением электродвигателя в соответствующий режим и не требует затрат энергии. Какой запас энергии в аккумуляторе нужно иметь к началу раскрутки маховика, чтобы спутник мог перенацелиться на L=50 км по Земле за T=10 секунд? К моменту начала и окончания перенацеливания оптическая ось камеры не должна вращаться. Спутник массой M=1000 кг можно принять однородным кубом со стороной a=1 м. Высота орбиты H=500 км. Вращение осуществляется вокруг оси, проходящей через центр спутника и параллельной вектору его орбитальной скорости. Принять, что электродвигатель раскручивает маховик с постоянным угловым ускорением [радиан/с2]. После достижения маховиком максимальной угловой скорости электродвигатель сразу же переключается в режим торможения, которое осуществляется с таким же (по модулю) угловым ускорением, что и раскрутка. Маховик имеет форму однородного цилиндра радиусом r=10 см массой m=10 кг, вращающегося вокруг своей продольной оси. КПД электродвигателя маховика принять 100%.

Система оценки

За правильный ответ дается 2 балла. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 1.5 на 3-ей попытке, до 1 балла на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 0.5 баллов и далее не убывает.

Решение

Оптическая ось камеры, как и сам спутник, должна повернуться на малый угол α:

$$\alpha = \frac{L}{H} = \frac{50}{500} = 0,1 \text{ радиан} = 5,73^\circ$$

Так как угловая скорость вращения спутника не ограничена, то перенацеливание можно осуществлять в два этапа: сначала разгон (вращение с положительным угловым ускорением) до поворота на угол $0.5 \cdot \alpha$, а затем торможение с момента достижения угла$0.5 \cdot \alpha$ до остановки при достижении угла α. Следовательно, можно найти

требуемое угловое ускорение спутника ε по аналогии с формулами для линейного ускорения:

$$\frac{\alpha}{2} = \frac{1}{2} \cdot \varepsilon_{\text{спутн}} \cdot (\frac{T}{2})^2$$

И тогда:

$$\varepsilon_{\text{спутн}} = \frac{4\alpha}{T^2}$$

При вращении спутника момент импульса для спутника будет равен моменту импульса крутящегося маховика. Тогда в любой момент времени t:

$$I_{\text{спутн}} \cdot \omega(t) = I_{\max} \cdot \Omega(t)$$

При вращении спутник достигнет максимальной угловой скорости ω_m:

$$\omega_m = \varepsilon_{\text{спутн}} \cdot \frac{T}{2} = \frac{4\alpha}{T^2} \cdot \frac{T}{2} = \frac{2 \cdot \alpha}{T}$$

Соответственно, маховик достигнет предельной угловой скорости Ω_m:

$$\Omega_m = \frac{I_{\text{спутн}}}{I_{max}} \cdot \omega_m$$

В результате раскрутки маховика вся система «спутник без маховика+маховик» за время $0,5 \cdot T$ приобретает кинетическую энергию вращения:

$$K = \frac{1}{2} \cdot (I_{\text{спутн}} \cdot \omega_m^2 + I_{max} \cdot \Omega_m^2) = \frac{1}{2} \cdot (I_{\text{спутн}} \cdot (\frac{2\alpha}{T})^2 + I_{max} \cdot (\frac{I_{\text{спутн}}}{I_{max}})^2 \cdot (\frac{2\alpha}{T})^2) =$$

$$= \frac{1}{2} \cdot (\frac{2\alpha}{T})^2 \cdot (I_{\text{спутн}} + I_{max} \cdot (\frac{I_{\text{спутн}}}{I_{max}})^2) = \frac{2\alpha^2}{T^2} \cdot I_{\text{спутн}} \cdot (1 + \frac{I_{\text{спутн}}}{I_{max}})$$

Именно эту энергию и нужно иметь в аккумуляторе. При торможении маховика вся она будет отдана обратно аккумулятору (при допущении 100% КПД в режиме генератора). Раскроем моменты инерции в финальной формуле

$$K = \frac{2 \cdot \alpha^2}{T^2} \cdot I_{\text{спутн}} \cdot (1 + \frac{I_{\text{спутн}}}{I_{max}}) = \frac{2 \cdot \alpha^2}{T^2} \cdot \frac{(M-m) \cdot a^2}{6} \cdot (1 + \frac{\frac{(M-m) \cdot a^2}{6}}{\frac{m \cdot r^2}{2}}) =$$

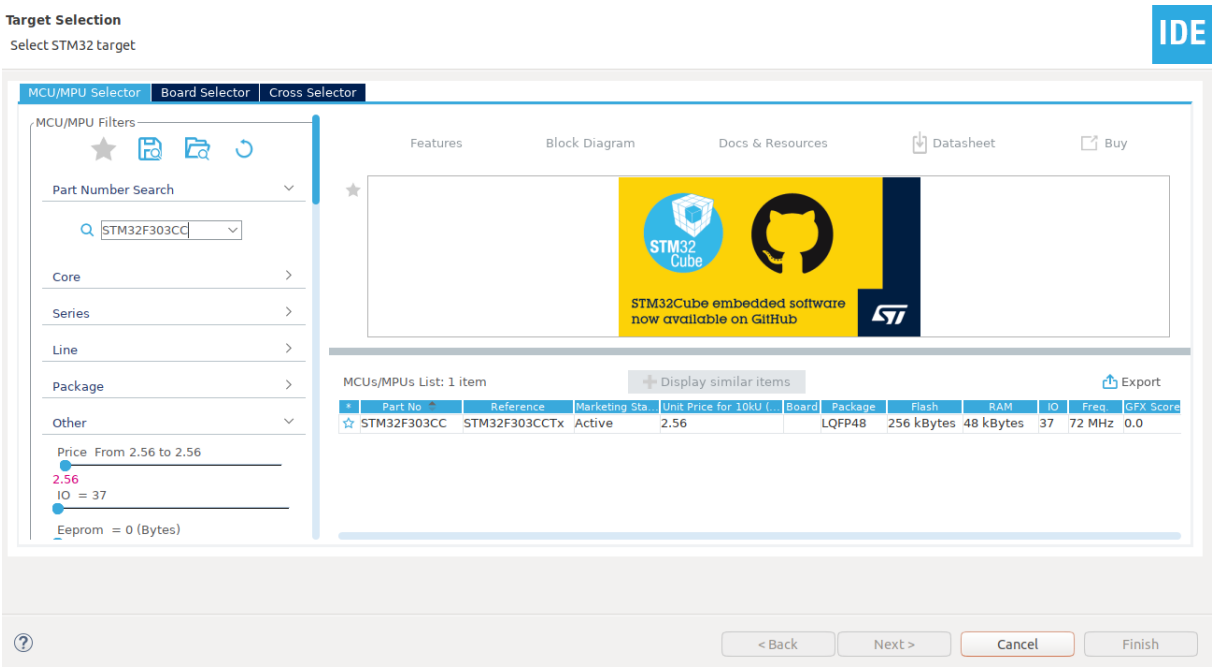
$$= \frac{\alpha^2}{T^2} \frac{(M-m) \cdot a^2}{3} \cdot (1 + \frac{(M-m) \cdot a^2}{3mr^2}) == \frac{0.01}{100} \cdot \frac{(1000-10)}{3} \cdot (1 + \frac{1000-10}{3 \cdot 10 \cdot 0,01}) = 109 \text{Дж}$$

Ответ: 109 Дж.

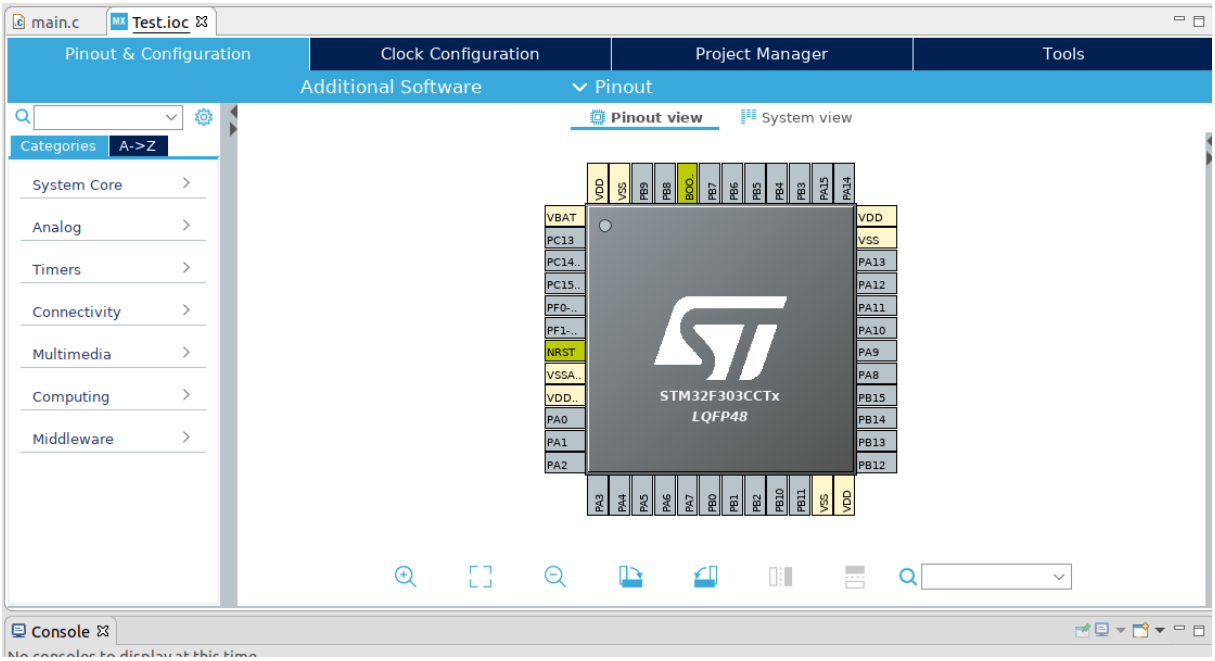
Задача II.1.3.2. Инициализация портов ввода/вывода, знакомство с GPIO (2 балла)

После установки STM32CubeIDE, запустите его. В верхнем меню выберите File → New → STM32 Project.

В открывшемся окне найдите микроконтроллер STM32F303CCTx и выберите его.



Нажмите кнопку Next. Необходимо будет ввести имя проекта и нажать кнопку Finish, после чего откроется файл с расширением .ioc, в котором будет представлен выбранный микроконтроллер.



Почти каждый вывод микроконтроллера (пин) на представленной схеме можно настроить. Для этого нужно кликнуть по выводу левой кнопкой мыши и выбрать нужный режим настройки вывода.

Предположим, что мы имеем простую систему, в которой в зависимости от дискретного сигнала (принимающего значения 0 или 1) фоторезистора происходит переключение состояния реле (вкл или выкл). Проще говоря, логика работы следующая:

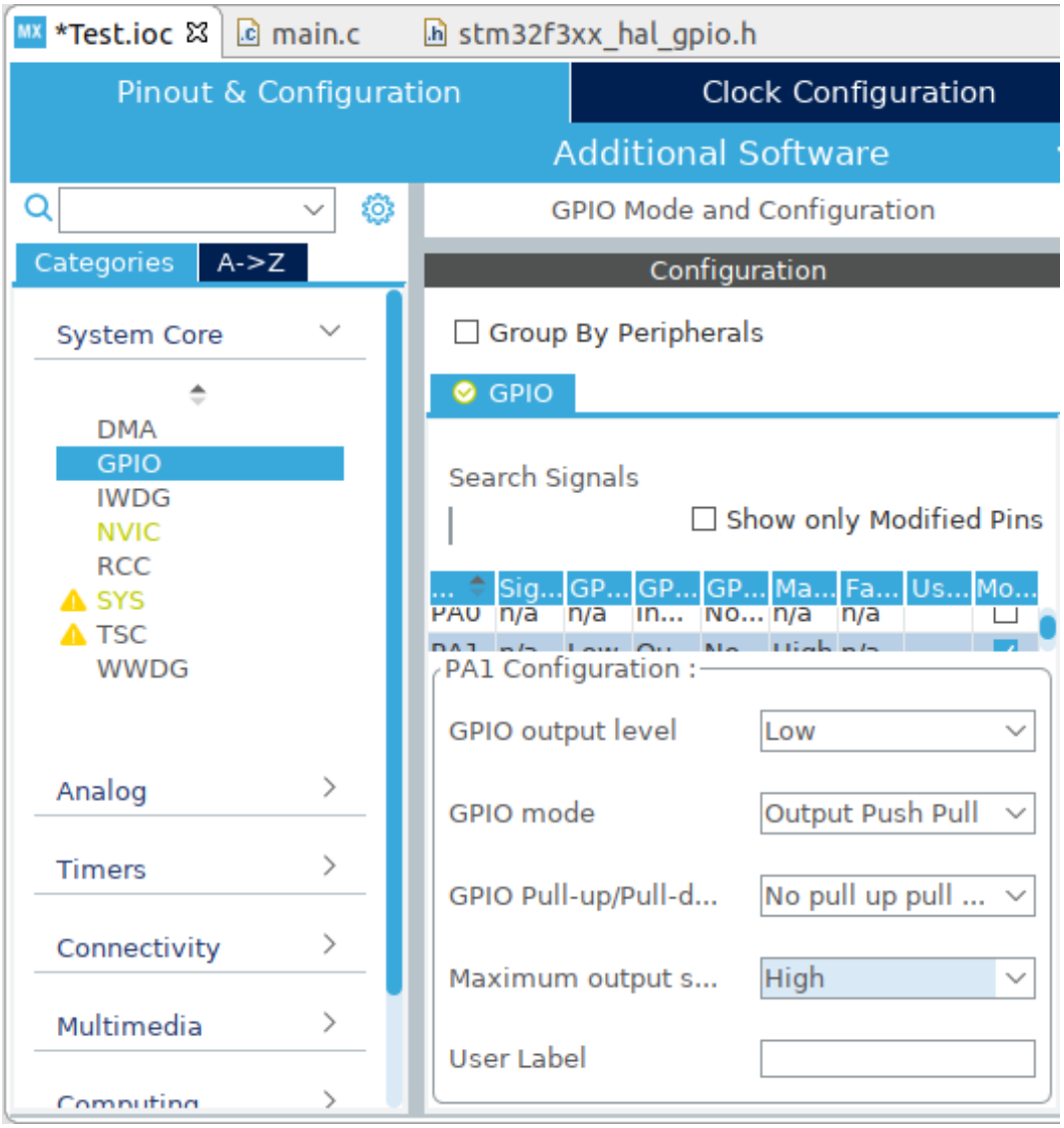
- если на фоторезистор падает свет, то он выдает значение, равное 1;
- если на фоторезистор находится в тени, то он выдает значение, равное 0;

- если сигнал с фоторезистора равен 1, то реле переходит в состояние «вкл»;
- если сигнал с фоторезистора равен 0, то реле переходит в состояние «выкл».

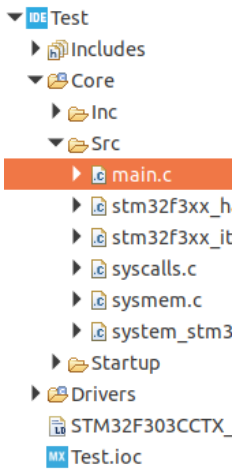
В данном случае фоторезистор выступает в качестве источника входного сигнала для микроконтроллера, а контакт управления реле является выходным сигналом. Для реализации необходимой инициализации пинов воспользуемся режимом GPIO — ввода/вывода данных.

Более подробно о GPIO можно прочитать, например, здесь: http://mypractic.ru/urok-6-porty-vvoda-vyvoda-stm32.html

Необходимо задать PA0 как входной сигнал (GPIO_Input), PA1 как выходной (GPIO_Output). После этого в слева в меню в категории System Core в пункте GPIO отобразятся выбранные пины. Выберите PA1 и в появившемся окне PA1 Configuration в поле Maximum output speed измените значение на HIGH.



После инициализации пинов необходимо сгенерировать код (значок шестеренки в верхней панели меню), и открыть файл main.c в дереве проекта.



В файл main.c генерируется код, который содержит инициализацию необходимых пинов, интерфейсов и пр, а также функцию «int main(void)», которая задает работу микроконтроллера.

Найдите функцию «static void MX_GPIO_Init(void)».В ней есть строки, отвечающие за инициализацию пинов PA0 и PA1.

```
/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

/*Configure GPIO pin : PA0 */
GPIO_InitStruct.Pin = GPIO_PIN_0;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : PA1 */
GPIO_InitStruct.Pin = GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

По описанным настройкам видно, что режим работы пина в качестве входа или выхода определяется полем Mode. Кликните по присваиваемому значению данного поля (GPIO_MODE_INPUT или GPIO_MODE_OUTPUT_PP) правой кнопкой мыши и выберите «Open Declaration».

Сколько режимов GPIO всего можно задать?

Критерий оценки:

- За правильный ответ начисляется 2 балла.
- Всего доступно 2 попытки.

Решение

После выполнения всех указанных действий в STM32CubeIDE откроется в новой вкладке файл, в котором можно увидеть все инициализируемые номера пинов.

```
#define GPIO_PIN_0                 ((uint16_t)0x0001U)  /* Pin 0 selected    */
#define GPIO_PIN_1                 ((uint16_t)0x0002U)  /* Pin 1 selected    */
#define GPIO_PIN_2                 ((uint16_t)0x0004U)  /* Pin 2 selected    */
#define GPIO_PIN_3                 ((uint16_t)0x0008U)  /* Pin 3 selected    */
#define GPIO_PIN_4                 ((uint16_t)0x0010U)  /* Pin 4 selected    */
#define GPIO_PIN_5                 ((uint16_t)0x0020U)  /* Pin 5 selected    */
#define GPIO_PIN_6                 ((uint16_t)0x0040U)  /* Pin 6 selected    */
#define GPIO_PIN_7                 ((uint16_t)0x0080U)  /* Pin 7 selected    */
#define GPIO_PIN_8                 ((uint16_t)0x0100U)  /* Pin 8 selected    */
#define GPIO_PIN_9                 ((uint16_t)0x0200U)  /* Pin 9 selected    */
#define GPIO_PIN_10                ((uint16_t)0x0400U)  /* Pin 10 selected   */
#define GPIO_PIN_11                ((uint16_t)0x0800U)  /* Pin 11 selected   */
#define GPIO_PIN_12                ((uint16_t)0x1000U)  /* Pin 12 selected   */
#define GPIO_PIN_13                ((uint16_t)0x2000U)  /* Pin 13 selected   */
#define GPIO_PIN_14                ((uint16_t)0x4000U)  /* Pin 14 selected   */
#define GPIO_PIN_15                ((uint16_t)0x8000U)  /* Pin 15 selected   */
#define GPIO_PIN_All               ((uint16_t)0xFFFFU)  /* All pins selected */
```

Максимальный номер — 15.

Ответ: 15.

Задача II.1.3.3. Использование GPIO (2 балла)

Основная работа микроконтроллера задается функцией «int main(void)» в файле main.c. Обратите внимание, что изначально в ней не так много кода, большая часть — это комментарии (выделяются как правило зеленым или серым цветом), помогающие сориентироваться. Можно заметить, что есть ряд комментариев, содержащих слова USER CODE BEGIN и USER CODE END. Это те места в функции, в которых желательно писать код. Тогда при возврате к распиновке микроконтроллера и последующей генерации кода код не будет потерян. Иначе написанный вами код может затираться каждый раз при новой генерации кода.

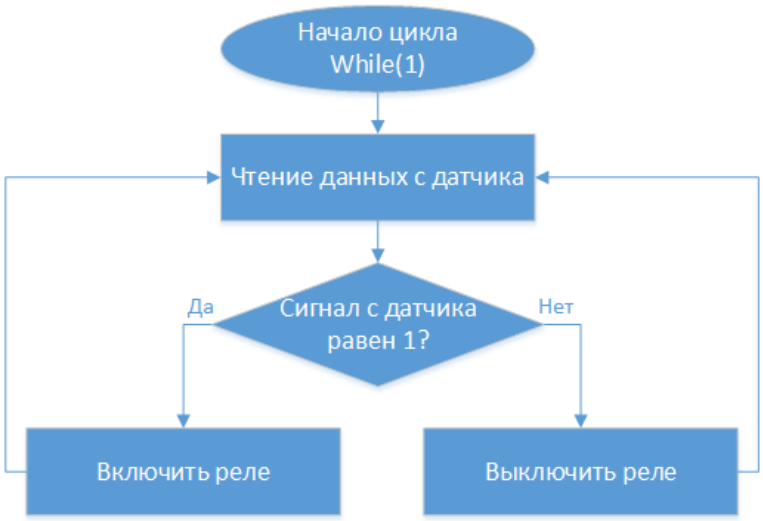
Структура функции «int main(void)» задана таким образом, что все, что необходимо задать изначально, задается до бесконечного цикла while(1). В самом же цикле задается программа микроконтроллера, которая повторяется до тех пор, пока микроконтроллер не будет обесточен. Если сравнивать со структурой скетча Arduino IDE, то все до цикла while(1) можно поставить в соответствие функции void setup(), а сам цикл функции void loop().

Обратите внимание, что в функции int main(void) есть следующие строки:

1. HAL_Init();
   Эта строчка вызывает инициализацию функций HAL — типовых функций, которыми мы будем пользоваться.
2. SystemClock_Config();
   Эта строчка вызывает функцию инициализации системных настроек, таких как источники тактирования и частоты работы различных интерфейсов.

3. MX_GPIO_Init();
   Уже известная вам функция, которая инициализирует настройку пинов, работающих в режиме GPIO.

Вернемся к логике работы системы. Ее можно описать следующей блок-схемой:



Действие «Чтение сигнала с датчика» в нашем случае будет задаваться типовой функцией HAL_GPIO_ReadPin. Единице соответствует состояние, когда на пине PA0 есть напряжение (т. е. уровень напряжения 3.3 В), а нулю — когда напряжения нет (т. е. уровень напряжения — 0 В)

Включение и выключение реле происходит за счет подачи на пин PA1 напряжения. Например, при напряжении в 3.3 В на пине PA1 реле будет включаться, а при напряжении 0 В выключаться. Чтобы подать напряжение на пин можно воспользоваться функцией HAL_GPIO_WritePin.

Код управления можно задать следующим образом:

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

  /* USER CODE BEGIN 3 */

    if( HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET ) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
    }
    else {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
    }

}
```

GPIOA указывает обеим функциям, что используются порты из группы A, к которой и принадлежат пины PA0 и PA1, это указано в их названии (например, для пинов PB0 и PB1 нужно было бы указать GPIOB). Переменная, содержащая в названии сочетание «GPIO_PIN_» указывает номер пина в группе. Состояние пина GPIO_PIN_SET соответствует наличию напряжения на нем, а GPIO_PIN_RESET — его отсутствию.

Существует еще одна полезная функция для управления состоянием пинов — HAL_GPIO_TogglePin. Эта функция производит инверсию состояния пина, то есть сменяет SET на RESET.

Опираясь на приведенную информацию, определите, какое управление системой задает код ниже.

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

  /* USER CODE BEGIN 3 */

    if( HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET ) {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
    }

}
```

Варианты ответа:

1. При наличии света на фоторезисторе реле включается, а при отсутствии света — выключается.
2. При наличии света на фоторезисторе реле выключается, а при отсутствии света — включается.
3. Состояние реле стало зафиксированным.
4. При отсутствии света на фоторезисторе реле включается, а при отсутствии света — выключается.
5. При отсутствии света на фоторезисторе реле выключается, а при отсутствии света — включается.
6. При наличии света на фоторезисторе реле меняет свое состояние на противоположное.
7. При отсутствии света на фоторезисторе реле меняет свое состояние на противоположное.
8. Логика работы системы не изменилась.

Критерий оценки:

- За правильный ответ начисляется 2 балла.
- Всего доступно 2 попытки.

Ответ: 6.

Задача II.1.3.4. Вывод данных с помощью UART (2 балла)

Когда речь идет о работе с датчиками, то часто возникает необходимость в выводе данных, которые с него приходят. Для того, чтобы данные с датчика, которые считывает микроконтроллер, отобразились на экране монитора, необходимо настроить обмен данным. Часто обмен идет по так называемому протоколу UART. По нему, например, данные можно передать через подключение по USB-порту.

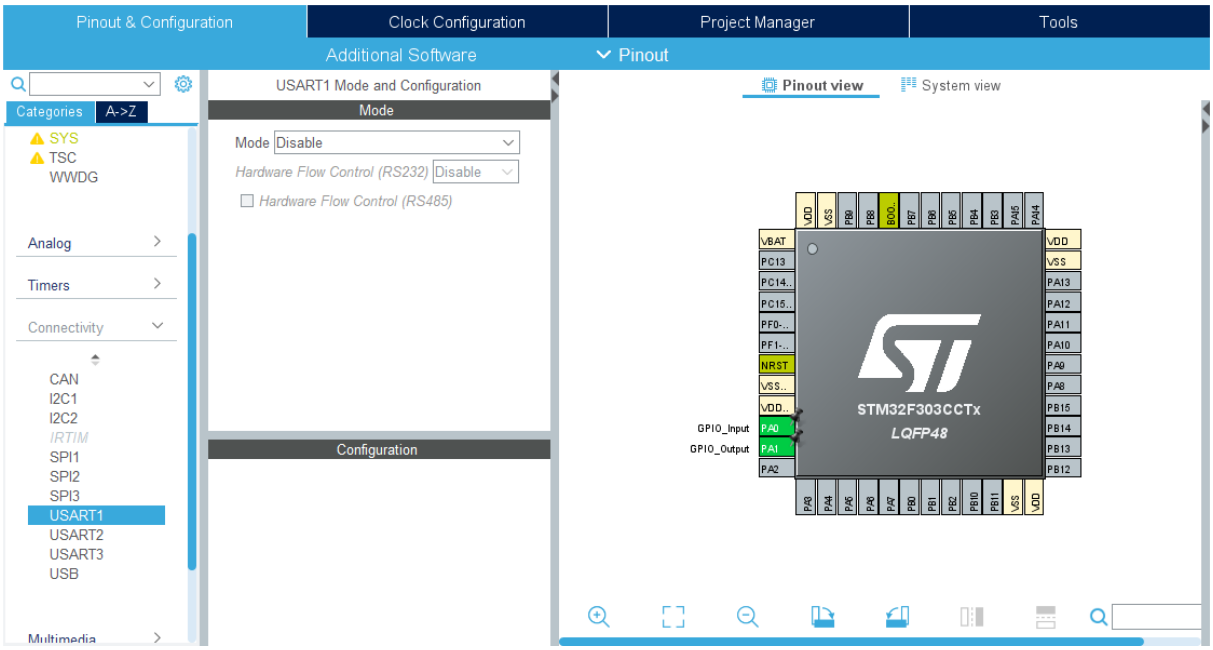
Более подробно с UART можно ознакомиться здесь:

http://easystm32.ru/interfaces/14-uart-in-stm32-part-0/

У микроконтроллера STM32F303CCTx есть возможность задействовать несколько каналов UART, однако чаще всего достаточно одного. UART задается двумя каналами: Rx, который принимает данные, и Tx, который отправляет данные.

Чтобы задать UART с помощью STM32CubeIDE, нужно вернуться к вкладке с распиновкой микроконтроллера. Если вы ее закрыли, то дважды кликните на файл с расширением .ioc слева в дереве проекта.

Перейдите в меню слева в пункт Connectivity и выберите USART1.



В появившемся меню настроек выберите Mode →Asynchronous. После этого можете наблюдать, как еще два пина инициализируются на схеме микроконтроллера (обычно PA9 и PA10). У пинов также появятся надписи USART1_RX и USART1_TX.

Для передачи данных по UART используется функция HAL_UART_Transmit(). Если написать эту функцию в файле main.c и кликнуть правой кнопкой мыши, то можно также открыть ее описание («Open Declaration»).

a. Укажите, какая скорость (поле Baud Rate) задается в конфигурациях интерфейса USART по умолчанию.

b. По коду ниже определите, сколько байт передается по UART.

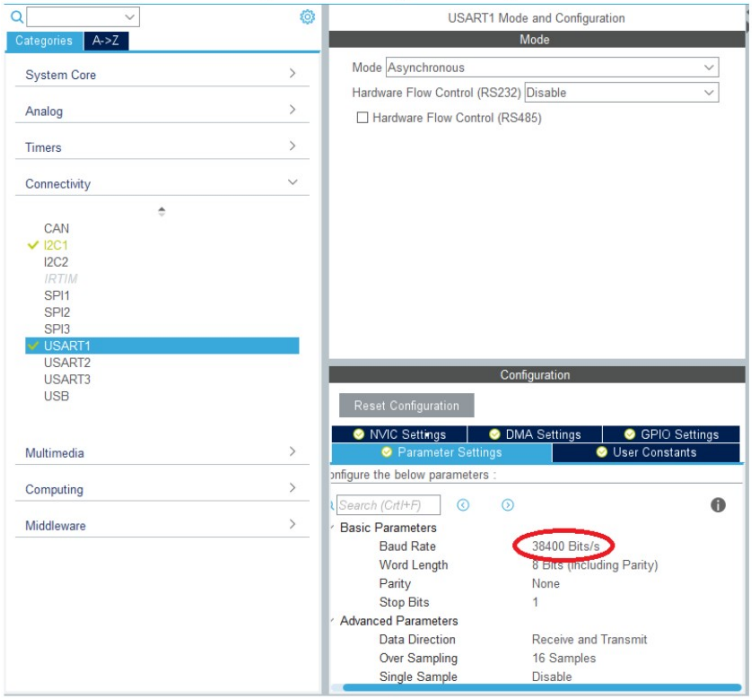
```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

      HAL_UART_Transmit(&huart1, str, 2, 10);
      HAL_Delay(100);
  }
  /* USER CODE END 3 */
}
```

Критерий оценки:

- За каждый пункт при правильном ответе начисляется 1 балл.
- Всего доступно 2 попытки.

Решение

a. При инициализации UART можно увидеть скорость по умолчанию:



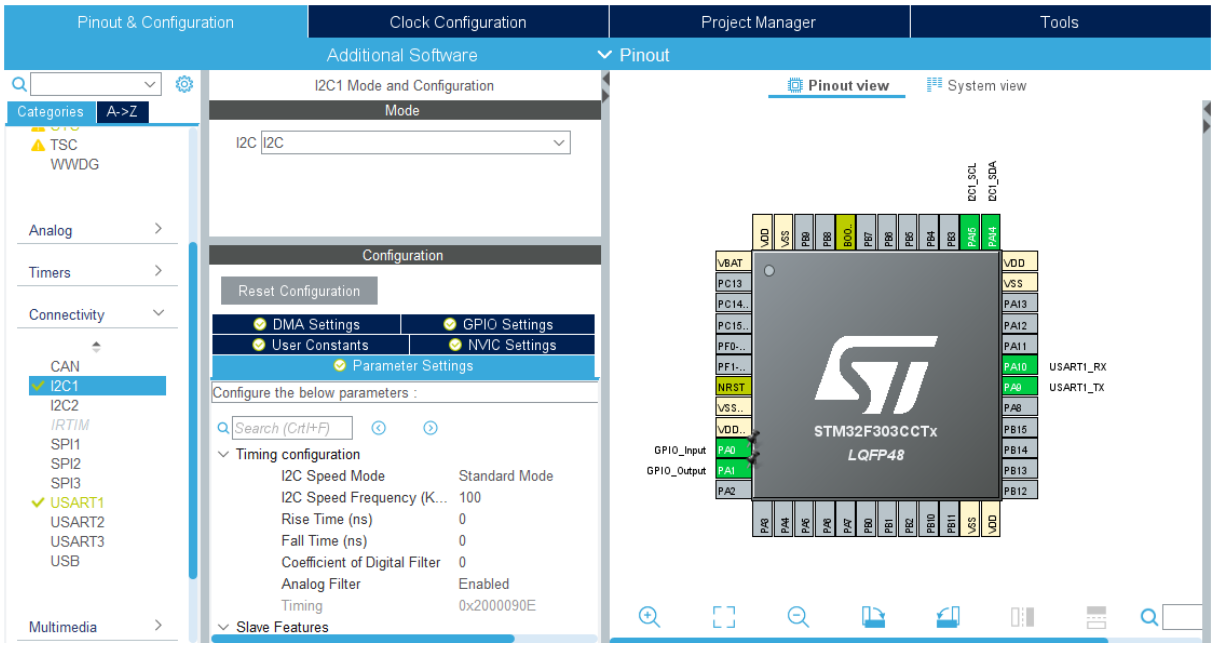
Это 38400 бит/сек.

b. У функции «HAL_UART_Transmit» в качестве третьего аргумента указывается количество передаваемых байт. В примере указано, что это 2 байта.

Ответ: a — 38400; b — 2.

Задача II.1.3.5. Шина I2C (2 балла)

Не уходите с вкладки с распиновкой микроконтроллера. В пункте Connectivity выберите также I2C1. В настройках Mode выберите I2C → I2C.



Аналогично с USART, можно увидеть, что инициализируются два пина с подписями I2C1_SDA и I2C1_SCL.

В сложных системах, содержащих множество устройств, осмысленно использование единой логики работы для всех устройств. Реализуется это с помощью использования различных интерфейсов и протоколов. I2C — один из самых простых и удобных интерфейсов, позволяющих с помощью одного микроконтроллера организовать работу множества устройств.

Шина I2C позволяет подключить одновременно до 127 устройств, которые смогут управляться от одного микроконтроллера. Управляющее устройство шины также принято называть master, а зависимые — slave. В нашем случае плата с микроконтроллером STM32F303CCT6 — master, а остальные устройства — slave.

Шина I2C состоит из двух линий: SCL и SDA. Линия SCL является линией тактирования и отвечает за синхронизацию устройств между собой по частоте работы. Линия SDA является шиной данных и отвечает за их прием и передачу.

Каждое устройство, подключенное к шине I2C имеет свой адрес. Сообщение от микроконтроллера идет по шине, и каждое устройство проверяет, относится ли данное сообщение к нему, сравнивая свой адрес и адрес, указанный в сообщении. Обычно данный адрес задается производителем устройства и указывается в datasheet устройства.

Адрес является семибитным числом, что и накладывает ограничение в 127 устройств (т. к. 1111111 в двоичной системе — это 127).

Более подробно о шине I2C можно прочитать здесь: https://blog.radiotech.kz/stm32/i2c-chast-1-perevod-iz-knigi-mastering-stm32/

Сгенерируйте код. Найдите функцию static void MX_I2C1_Init(void). Эта функция отвечает за инициализацию I2C.

Также как и для UART, для I2C существует библиотека HAL с типовыми функциями, позволяющими построить обмен данными между устройствами. Например, функция HAL_I2C_Master_Receive() позволяет считывать данные с подчиненного микроконтроллеру устройства.

Функция чтения данных с датчика освещенности выглядит следующим образом:

```
uint16_t BH1750_readLightLevel(I2C_HandleTypeDef *I2C, uint8_t addr) {

    uint16_t level; //текущее измеренное значение
    uint8_t buf_in[2];

    while (HAL_I2C_GetState(I2C) != HAL_I2C_STATE_READY); //ждем готовность шины I2C
    HAL_I2C_Master_Receive(I2C, addr << 1, &buf_in, 2, 10); //Читаем 2 байта по адресу датчика

    level = buf_in[0]; //загружаем старший байт результата
    level  <<= 8; //сдвигаем его вверх
    level |= buf_in[1];//добавляем младший байт результата

    return level;
}
```

Переменная addr хранит I2C-адрес, заданный шестнадцатеричным числом. Чтобы получить семибитный адрес необходимо осуществить побитовый сдвиг на 1 бит, что задается выражением add « 1.

Данная функция далее вызывается в цикле while(1):

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

    /* USER CODE BEGIN 3 */
      Sensor = BH1750_readLightLevel(&hi2c1, 0x23);
      HAL_Delay(100);
  }

  /* USER CODE END 3 */
}
```

a. Какой I2C-адрес у микроконтроллера? Его можно увидеть в инициализации I2C после генерации кода. Ответ укажите одной цифрой.

b. Как будет выглядеть адрес датчика с учетом битового сдвига? В ответе укажите 7 бит без последнего бита после сдвига.

Критерий оценки:

1. За правильный ответ начисляется 2 балла.
2. Всего доступно 2 попытки.

Ответ: a — 0; b — 0100011.

Баллистик

Задача II.1.4.1. Задача по физике (2 балла)

Геостационарная орбита характеризуется «точкой стояния» — долготой места на экваторе Земли, находящегося точно под спутником, обращающимся вокруг Земли в плоскости экватора с периодом 24 часа синхронно с вращением планеты.

Спутник находится на геостационарной орбите в точке стояния 60° в.д. Его нужно перевести в точку стояния 120 ° в.д. Сделать это можно путем перевода спутника на «вложенную» эллиптическую орбиту, касающуюся круговой геостационарной орбиты внутренним образом. Период обращения по «вложенной орбите» будет меньше 24 часов, таким образом, при обратном переходе с «вложенной орбиты» на геостационарную, спутник окажется восточнее своей предыдущей «точки стояния». Переход с геостационарной на вложенную орбиту осуществляется двигателем спутника путем выдачи ему тормозного импульса, происходит уменьшение скорости спутника на величину ΔV. Точка выдачи этого импульса станет апогеем вложенной орбиты, таким образом, для орбитальных скоростей имеем $V_{\text{апогей}} = V_{\text{ГСО}} - \Delta V$. Наоборот, переход со вложенной орбиты на круговую осуществляется выдачей двигателем спутника разгонного импульса в апогее. Какова будет высота спутника над Землей в перигее вложенной «временной» орбиты?

Математик Кеплер установил, что если тело движется по эллиптической орбите, то центр Земли находится в одном из фокусов этого эллипса (первый закон Кеплера). Частным случаем эллипса является окружность, в этом случае оба фокуса эллипса находятся в одной и той же точке. Также для тела, движущегося по эллиптической орбите, справедливо равенство (второй закон Кеплера):

$$V(t_1) \cdot r(t_1) \cdot sin\Theta_1 = V(t_2) \cdot r(t_2) \cdot sin\Theta_2$$

Где t_1 и t_2 — произвольные моменты времени,

r — расстояние от тела до центра Земли,

V — орбитальная скорость тела,

Θ — угол между вектором скорости тела и направлением на центр Земли. В частности, для точек апогея и перигея вектор скорости перпендикулярен направлению на Землю, тогда $\Theta = 0$:

$$V_{\text{апогей}} \cdot r_{\text{апогей}} = V_{\text{перигей}} \cdot r_{\text{перигей}}$$

Третий закон Кеплера позволяет связать периоды обращения T_1 и T_2 по любым двум околоземным эллиптическим орбитам с бОльшими полуосями эллипсов орбит a_1 и a_2. В частном случае окружности полуосью эллипса является ее радиус.

$$\frac{T_2}{T_1} = (\frac{a_2}{a_1})^{\frac{3}{2}}$$

Начальные данные:

Масса Земли: $6 \cdot 10^{24}$ кг.

Радиус Земли: 6370 км.

Гравитационная постоянная: $G = 6.67 \cdot 10^{-11}$ Н$\cdot$ м^2 / кг^2.

Радиус круговой орбиты, имеющей период обращения 24 часа: $R_{\text{гсо}} = 42168$ км.

Система оценки

За правильный ответ дается 2 балла. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 25% (т. е. максимально возможный балл снижается до 1.5 на 3-ей попытке, до 1 балла на четвертой попытке). Начиная с 5-ой попытки максимальный балл снижается до 0.5 баллов и далее не убывает.

Решение

Для начала нужно найти необходимый период обращения по вложенной эллиптической орбите. Пусть α — долгота старой точки стояния, β — долгота новой точки стояния. Спутник движется с запада на восток. Период обращения по вложенной орбите меньше, чем по ГСО, поэтому Земля «отстает» от спутника. Значит, в момент возвращения в апогей подспутниковая точка будет восточнее по сравнению с исходной точкой стояния. Тогда при периоде обращения по геостационарной орбите $T_{\text{гсо}} = 24$ часа имеем:

$$T_{\text{элл}} = T_{\text{гсо}} \cdot \frac{360^\circ - (\beta - \alpha)}{360^\circ}$$

Значит, по третьему закону Кеплера, имеем соотношение между большими диаметрами орбит при высоте перигея h:

$$\frac{T_{\text{элл}}}{T_{\text{гсо}}} = (\frac{h + R_{\text{зем}} + R_{\text{гсо}}}{2R_{\text{гсо}}})^{\frac{3}{2}}$$

Отсюда имеем для искомой высоты перигея h:

$$h = (\frac{T_{\text{элл}}}{T_{\text{гсо}}})^{\frac{3}{2}} \cdot 2 \cdot R_{\text{гсо}} - R_{\text{зем}} - R_{\text{гсо}} = (\frac{360^\circ - (\beta - \alpha)}{360^\circ})^{\frac{2}{3}} \cdot 2 \cdot R_{\text{гсо}} - R_{\text{зем}} - R_{\text{гсо}} =$$

$$= (\frac{360^\circ - (120 - 60)}{360^\circ})^{\frac{2}{3}} \cdot 2 \cdot 42168 - 6370 - 42168 = 26145 \text{ км} \approx 26000 \text{ км}$$

Ответ: 26000 км.

Задача II.1.4.2. Орбита: подбор орбиты (8 баллов)

Группировка из 6-ти низкоорбитальных спутников обеспечивает связью земной шар. Все спутники расположены на круговых орбитах высотой 885 км. Элементы орбит спутников приведены в таблице.

Элемент орбиты	Спутник 1	Спутник 2	Спутник 3	Спутник 4	Спутник 5	Спутник 6
Большая полуось, км	7256	7256	7256	7256	7256	7256
Эксцентр.	0	0	0	0	0	0
Наклонение	70	70	70	70	70	70
ДВУ	0	60	120	180	240	300
АП	0	0	0	0	0	0
Истинная аномалия	0	120	240	60	180	300

В результате сильной магнитной бури спутники 2, 3 и 4 вышли из строя. У компании-производителя есть возможность запустить на замену трем спутникам

только один, но с улучшенной аппаратурой, позволяющей запустить его на более высокую орбиту, но с большой полуосью не более 20000 км.

Какой должна быть орбита нового спутника, чтобы обеспечить связь с городами Оттава, Рио-де-Жанейро, Москва, Пекин и Сидней? Для обеспечения связи необходимо, чтобы хотя бы раз в 5 часов над городом пролетал один из спутников. Считается, что спутник обеспечивает связь, если он находится на 30 градусах над уровнем горизонта и выше.

Сферические координаты городов:

- Рио-де-Жанейро: -22.9064, 316.88
- Оттава: 45.4112, 284.31
- Москва: 55.7522, 37.6155
- Пекин: 39.0907, 116.397
- Сидней: -33.8678, 151.20732

Время моделирования: с 01 января 2021 года 12:00 по 02 января 2021 года 12:00 по UTC.

Система оценки

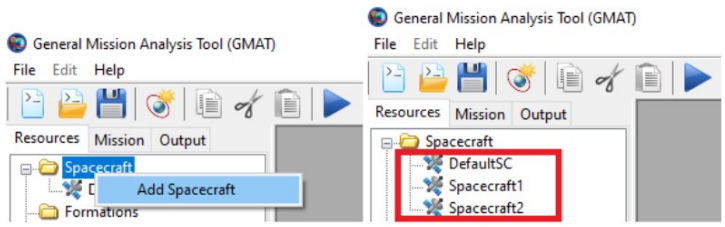
Всего образуется пять окон без связи, длительностью более 5-ти часов. За закрытие окна начисляется 1.6 балла.

Максимум можно заработать 8 баллов. Всего без штрафа доступно две попытки. После двух попыток действует штраф в 10% (т. е. максимально возможный балл снижается до 7.2 на 3-ей попытке, до 6.4 на четвертой попытке и т. д.). Начиная с 8-ой попытки максимальный балл снижается до 2 баллов и далее не убывает.

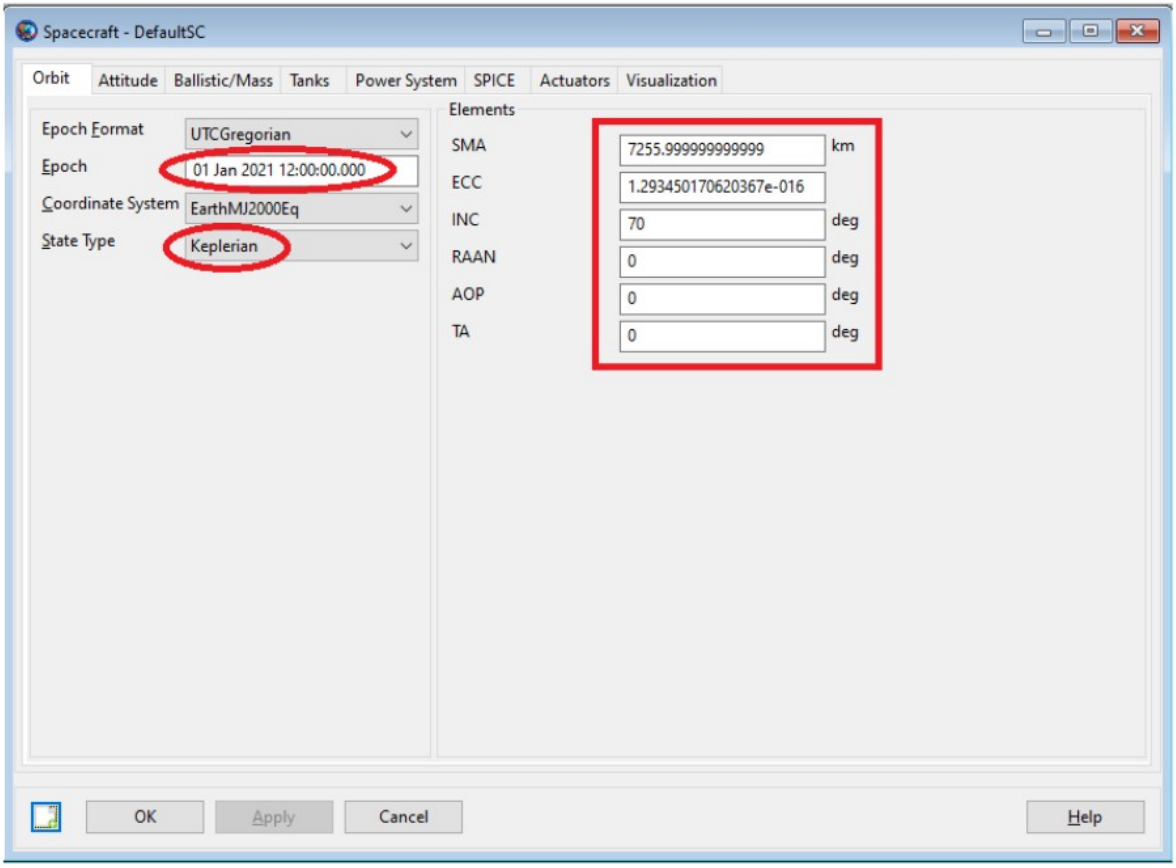
Время по UTC.

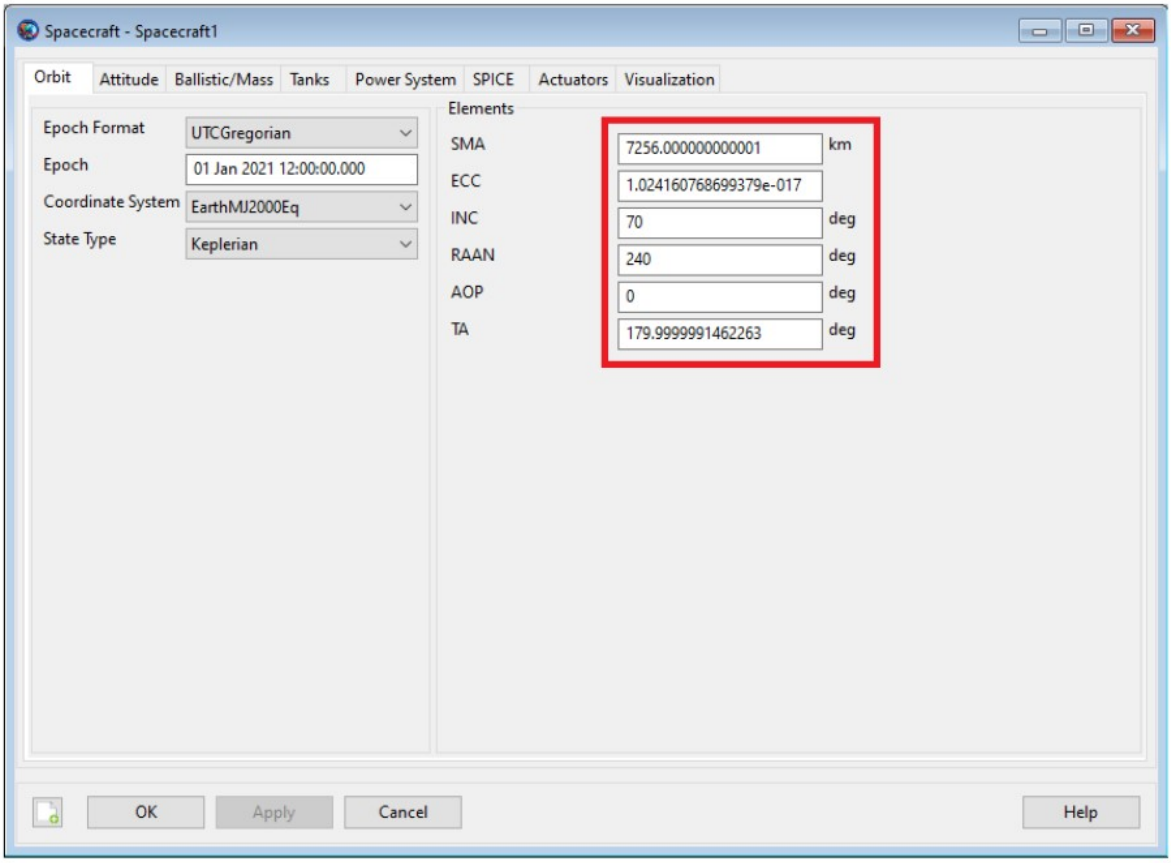
Решение

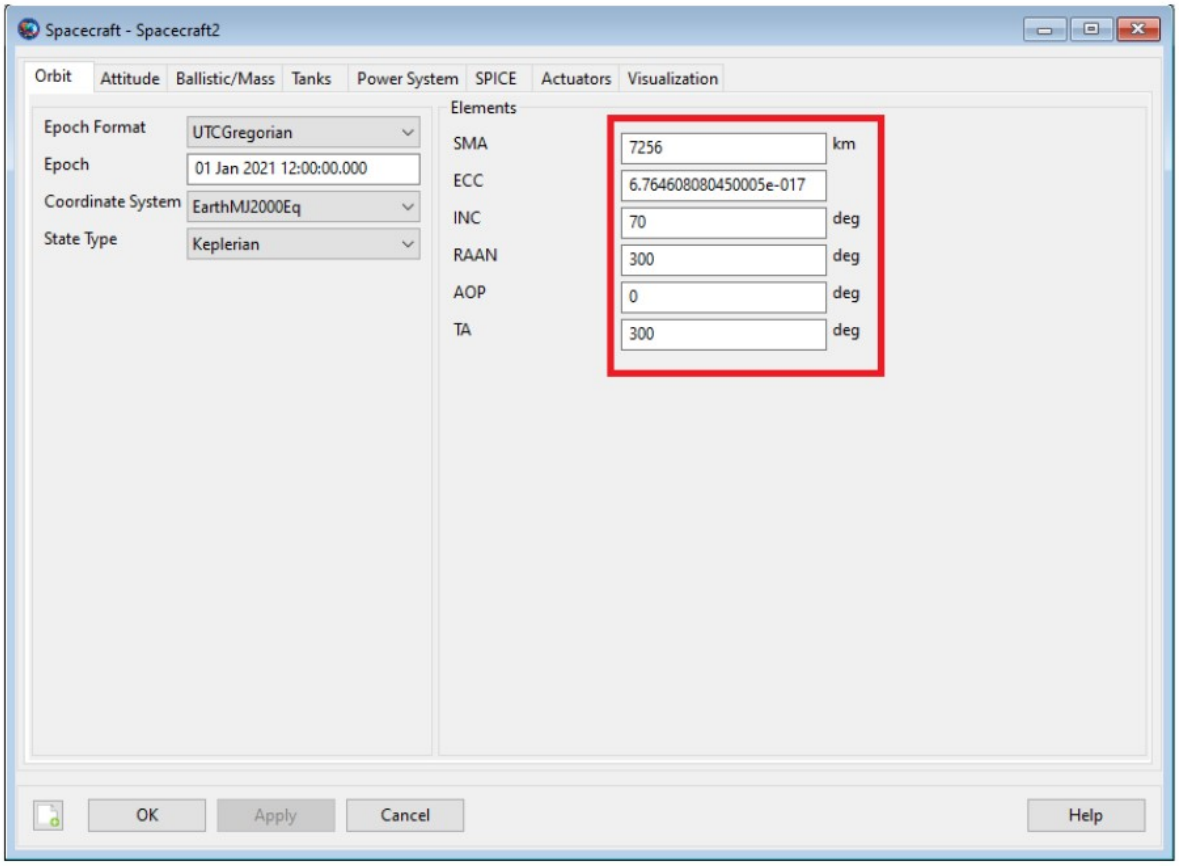
С помощью ПО GMAT можно определить образующиеся окна для пролета. Для этого необходимо задать орбиты оставшихся спутников. В дереве проекта слева в группе объектов Spacecraft необходимо задать еще 2 объекта:



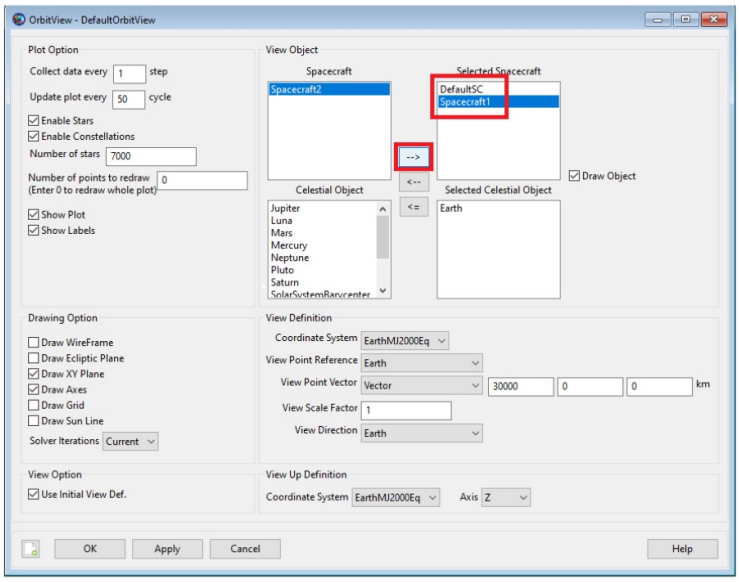
После этого каждому объекту необходимо задать время моделирования и соответствующие кеплеровы элементы орбиты:


Spacecraft - DefaultSC
Orbit
Attitude
Ballistic/Mass
Tanks
Power System
SPICE
Actuators
Visualization
Epoch Format
UTCGregorian
Epoch
01 Jan 2021 12:00:00.000
Coordinate System
EarthMJ2000Eq
State Type
Keplerian
Elements
SMA
7255.999999999999
km
ECC
1.293450170620367e-016
INC
70
deg
RAAN
0
deg
AOP
0
deg
TA
0
deg
OK
Apply
Cancel
Help

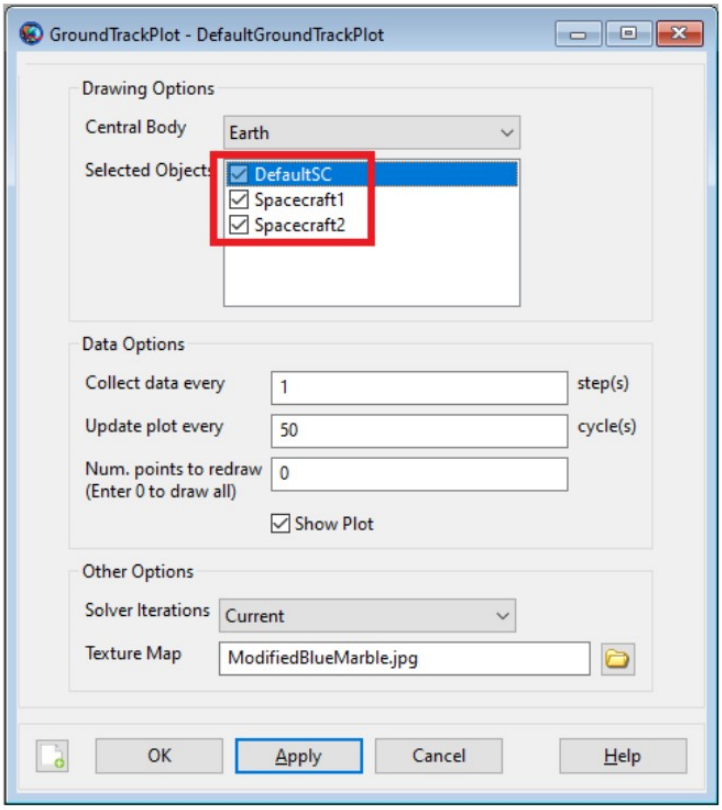

Spacecraft - Spacecraft1
Orbit
Attitude
Ballistic/Mass
Tanks
Power System
SPICE
Actuators
Visualization
Epoch Format
UTCGregorian
Epoch
01 Jan 2021 12:00:00.000
Coordinate System
EarthMJ2000Eq
State Type
Keplerian
Elements
SMA
7256.000000000001
km
ECC
1.024160768699379e-017
INC
70
deg
RAAN
240
deg
AOP
0
deg
TA
179.9999991462263
deg
OK
Apply
Cancel
Help



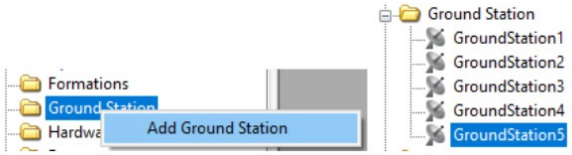
Далее в группе объектов типа Output в настройках DefaultOrbitView необходимо добавить отображение всех спутников:



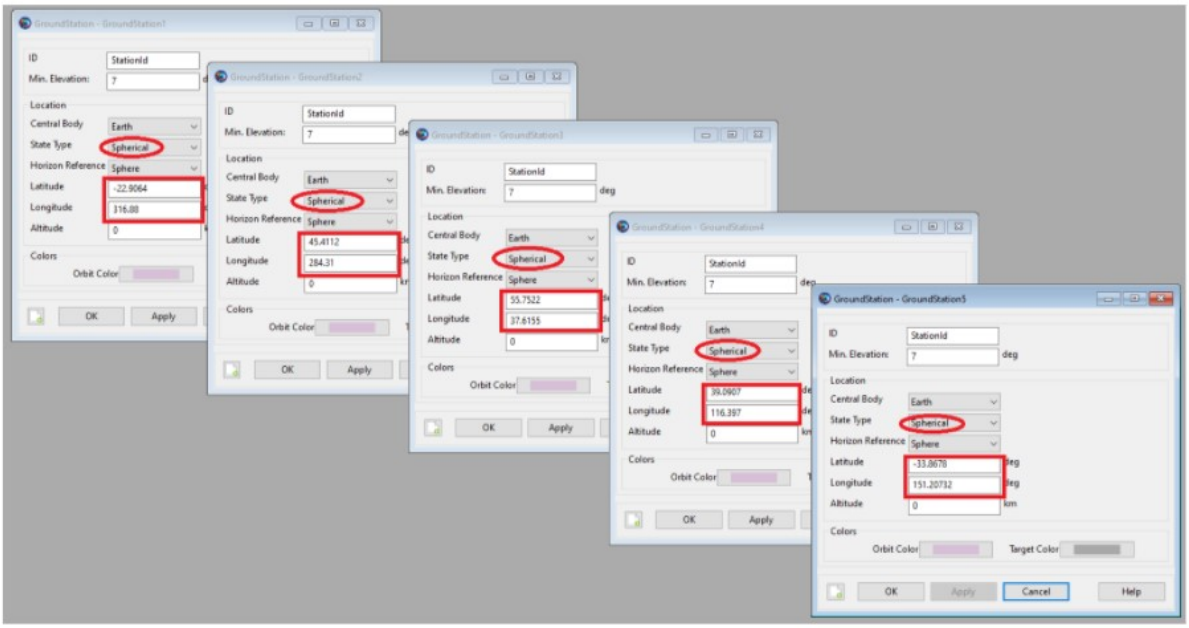
В настройках DefaultGroundTrackPlot также настроить отображение всех треков спутников:



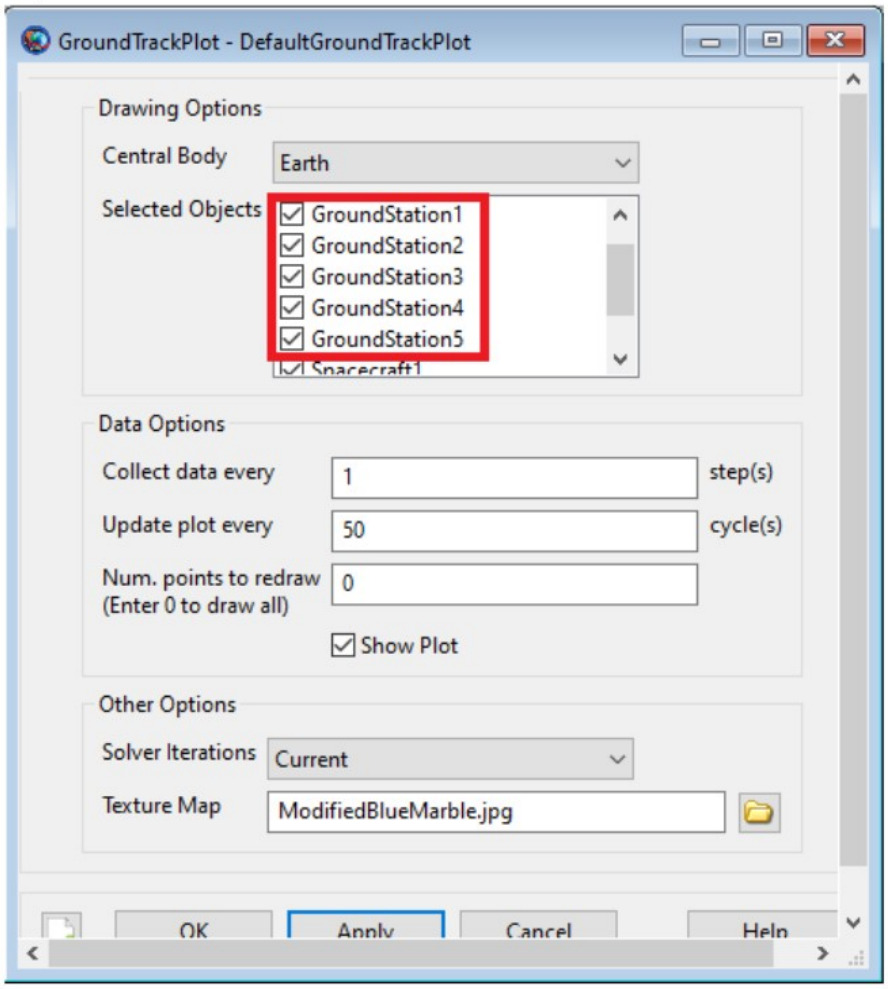
Далее необходимо добавить наземные станции, размещенные в указанных координатах городов. Для этого в группе объектов Ground Station необходимо задать 5 объектов:



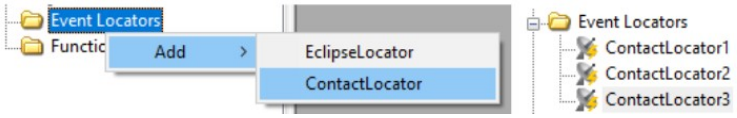
После этого необходимо каждому объекту задать сферические координаты соответствующего города:



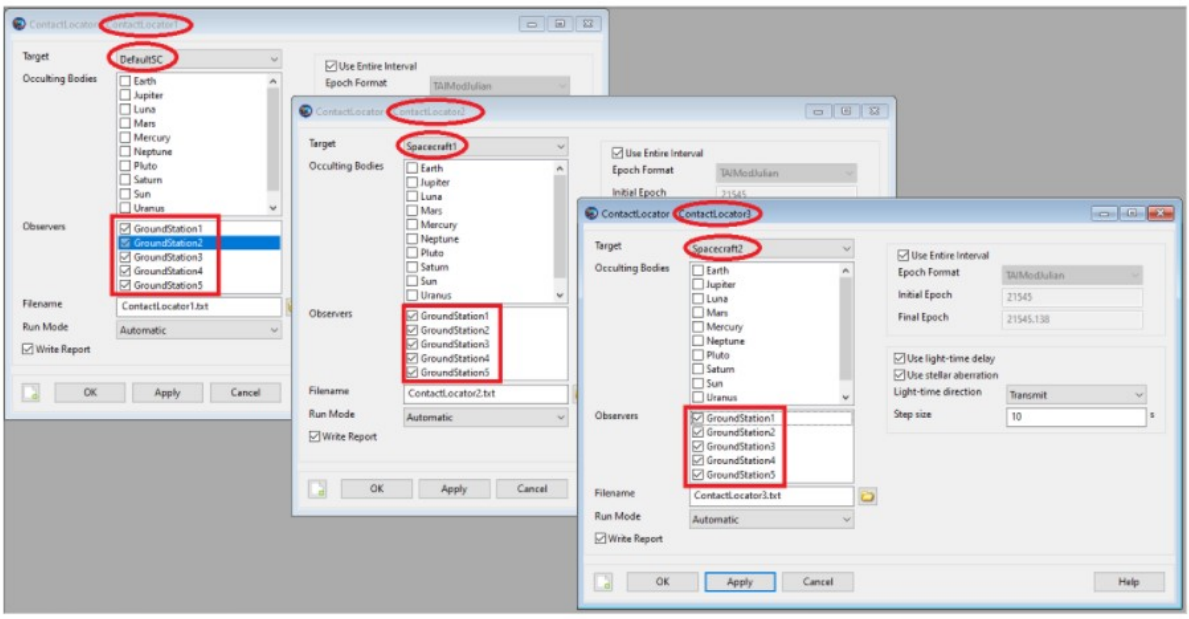
Снова откройте DefaultGroundTrackPlot и проставьте галочки объектам, соответствующим наземным станциям:



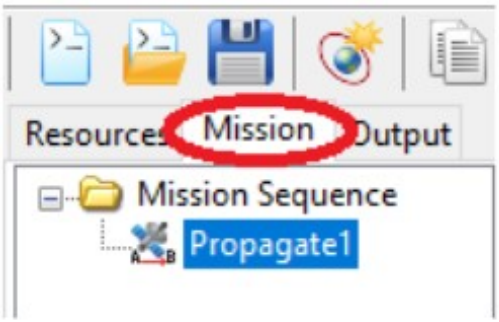
Затем, в группе объектов Event Locators добавить 3 объекта «Contact Locator»:



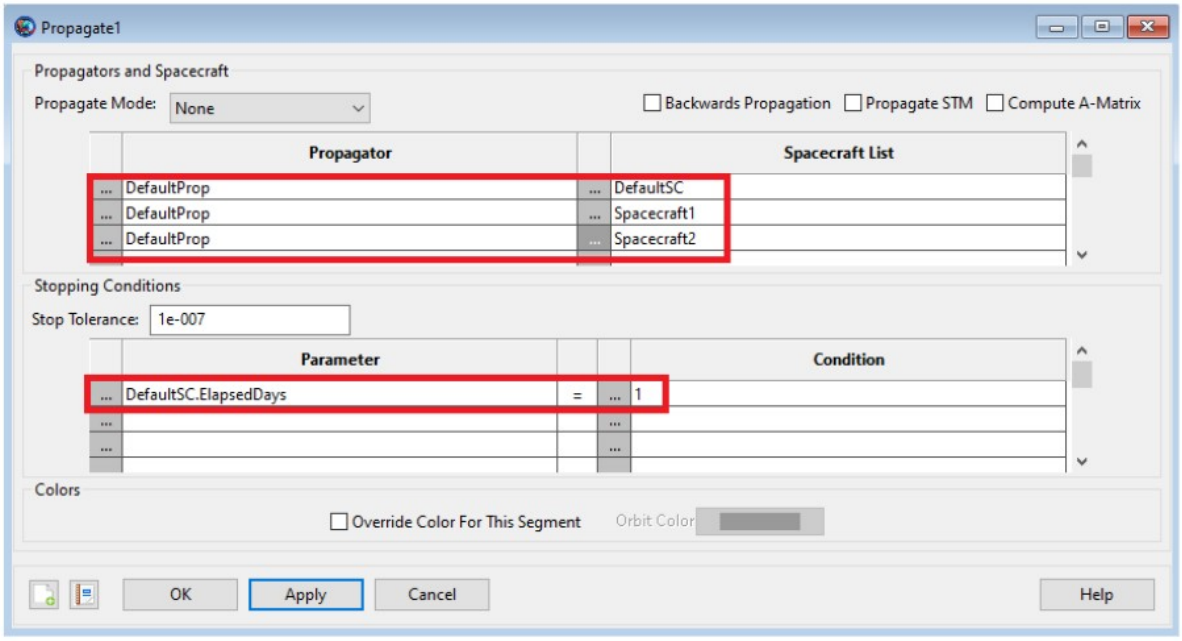
Каждому объекту ContactLocator поставьте в соответствие один из объектов Spacecraft и поставьте галочки для каждой наземной станции:



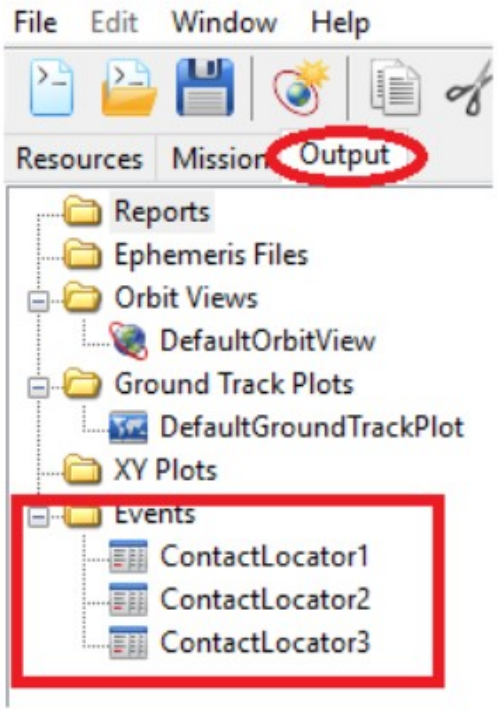
Переключитесь на вкладку Mission и откройте настройки Propagate1:

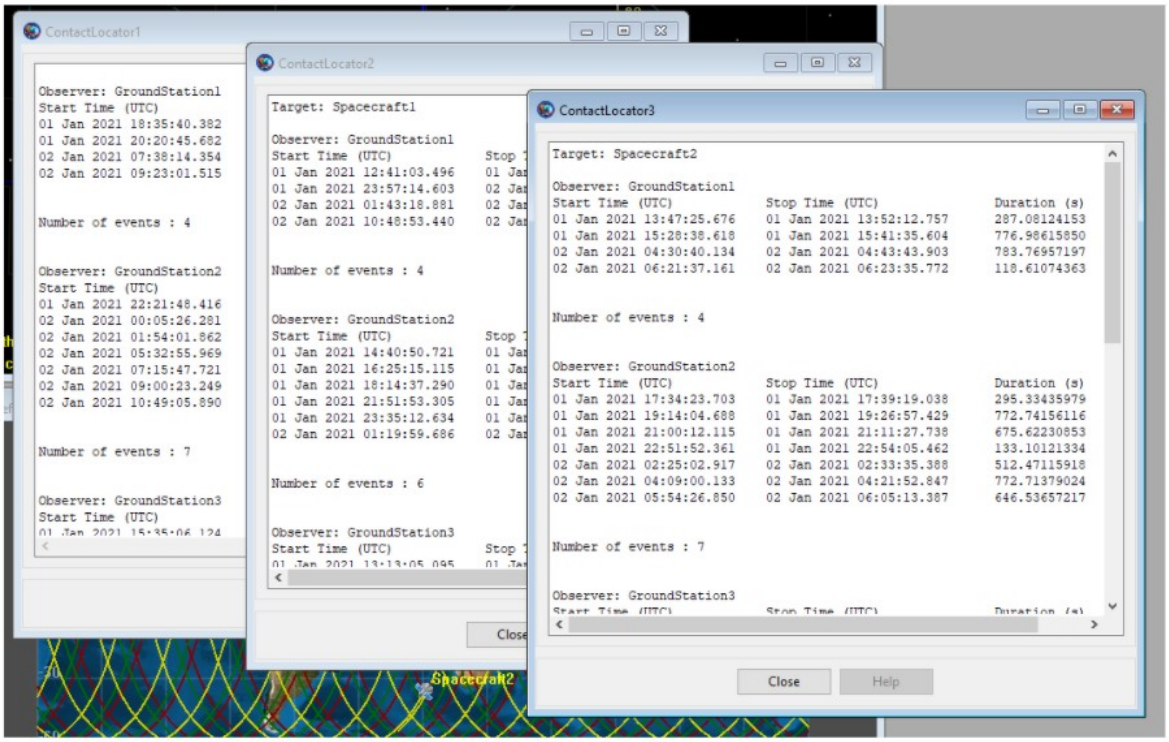


В открывшемся окне необходимо добавить все объекты Spacecraft, а также скорректировать время моделирования, выбрав свойство ElapsedDays и указав значение 1:



После этого запустите расчет нажав F5 или значок на верхней панели меню. Дождитесь окончания расчета. Переключитесь на вкладку Output. Здесь вы найдете 3 вывода данных с объектов типа ContactLocator.





Проанализировав полученные данные, можно узнать окна для каждого города, в которые он будет без связи:

1. Рио-де-Жанейро: 18:42-00:00, 04:40-10:52
2. Москва: 00:33-07:51
3. Пекин: 21:12-02:39
4. Сидней: 06:34-12:00

Расположим окна по мере возрастания времени:

1. Рио-де-Жанейро, 18:42-00:00
2. Пекин, 21:12-02:39
3. Москва, 00:33-07:51
4. Рио-де-Жанейро, 04:40-10:52
5. Сидней, 06:34-12:00

Некоторые полученные промежутки времени по длительности более четырех часов, соответственно, нужно уточнить временные рамки, в которые необходимо, чтобы спутник прошел над городом. Для этого к меньшему значению времени необходимо добавить 4 часа, а из большего — вычесть. Получим:

1. Рио-де-Жанейро, 19:00-23:42
2. Пекин, 21:39-02:12
3. Москва, 02:51-05:33
4. Рио-де-Жанейро, 05:52-09:40
5. Сидней, 07:00-11:34

После этого можно рассчитать параметры орбиты.

Зная координаты городов и временные промежутки, можно определить, с какой примерной угловой скоростью должен вращаться космический аппарат вокруг

центра Земли, чтобы успеть охватить все города.

Так как широта Москвы является наибольшей, то можно взять наклонение орбиты, приближенно равное ее широте: $i = 55°$. Также, для простоты можно рассмотреть случай с круговой орбитой. Тогда, ее эксцентриситет и аргумент перицентра будут равны нулю.

Рассчитаем приращение по координатам между городами:

Рассчитаем приращение по координатам между городами:

1. Рио-де-Жанейро — Пекин:$\Delta\phi = 61.9971, \Delta\lambda = -200.483$
2. Пекин — Москва: $\Delta\phi = 16.6615, \Delta\lambda = -78.7815$
3. Москва — Рио-де-Жанейро: $\Delta\phi = -78.6586, \Delta\lambda = 279.2645$
4. Рио-де-Жанейро — Сидней: $\Delta\phi = -10.9612, \Delta\lambda = -165.67268$

Определим допустимые временные промежутки, за которые спутник должен пролететь между городами:

1. Рио-де-Жанейро — Пекин: $t_1 = [2{:}30 — 2{:}39]$
2. Пекин — Москва: $t_2 = [3{:}21 — 5{:}12]$
3. Москва — Рио-де-Жанейро: $t_3 = [3{:}01 — 4{:}07]$
4. Рио-де-Жанейро — Сидней: $t_4 = [1{:}08 — 1{:}54]$

После этого можем определить необходимую угловую скорость по широте $\omega(\phi)$ и по долготе $\omega(\lambda)$:

1.
2. Рио-де-Жанейро — Пекин: $\omega(\phi) = 0.0069, \omega(\lambda) = -0.0223$ для$t_1 = 2:30$
3. Пекин — Москва: $\omega(\phi) = 0.0011, \omega(\lambda) = -0.0053$ для $t_2 = 4:09$ (промежуточное время, попадающее в допустимый промежуток)
4. Москва — Рио-де-Жанейро: $\omega(\phi) = -0.0068, \omega(\lambda) = 0.0241$ для $t_3 = 3:13$
5. Рио-де-Жанейро — Сидней: $\omega(\phi) = -0.0016, \omega(\lambda) = -0.0242$ для $t_4 = 1:54$

Необходимо скорректировать показания угловой скорости так, чтобы значения для разных городов были схожими. Сделать это можно, добавив к приращению координат городов 1 или 2 витка. При этом для $\Delta\phi$ приращение на виток будет равно $220°$, так как выбранное наклонение орбиты — $55°$, и широта подспутниковой точки будет изменяться от $-55°$ до $+55°$. Для $\Delta\lambda$ же приращение будет равно $360°$. Тогда получим:

1. Рио-де-Жанейро — Пекин: $\Delta\phi = 61.9971 + 220 = 281.9971, \Delta\lambda = -200.483 + 360 = 159.517$
2. Пекин — Москва: $\Delta\phi = 16.6615 + 220 \cdot 2 = 456.6615, \Delta\lambda = -78.7815 + 360 = 281.2185$
3. Москва — Рио-де-Жанейро:$\Delta\phi = -78.6586 + 220 \cdot 2 = 361.3414, \Delta\lambda = 279.2645$
4. Рио-де-Жанейро — Сидней:$\Delta\phi = -10.9612+220 = 209.0388, \Delta\lambda = -165.67268+ 360 = 194.32732$

Тогда новые значения угловой скорости будут:

1. Рио-де-Жанейро — Пекин:$\omega(\phi) = 0.0313, \omega(\lambda) = 0.0177$ для $t_1 = 2:30$
2. Пекин — Москва: $\omega(\phi) = 0.0306, \omega(\lambda) = 0.0188$ для $t_2 = 4:09$
3. Москва — Рио-де-Жанейро: $\omega(\phi) = 0.0312, \omega(\lambda) = 0.0241$ для $t_3 = 3:13$

4. Рио-де-Жанейро — Сидней: $\omega(\phi) = 0.0306, \omega(\lambda) = 0.0284$ для $t_4 = 1:54$

Тогда видно, что $\omega(\phi)$ можно принять примерно равным $0.0309°$ /сек , а $\omega(\lambda) - 0.0223°$/сек.

В приращение долготы $\omega(\lambda)$ входит также приращение за счет вращения Земли. Угловая скорость ее вращения $\omega 3 = 0.004178°$/сек. Так как выбранное наклонение орбиты меньше $90°$, то $\omega(\lambda)$ и $\omega 3$ совпадают по знаку.

После этого можем вычислить угловую скорость спутника.

$$\omega_{\text{спутника}} = \sqrt{\omega_\phi^2 - (\omega_\lambda - \omega_3)^2} = 0.036°/\text{сек}$$

Для движения по окружности справедливо:

$$v = \omega \cdot R$$

Космическая скорость вычисляется по формуле:

$$v = \sqrt{G \cdot \frac{M}{R}}$$

Тогда:

$$G \cdot \frac{M}{R} = \omega^2 R^2$$

Отсюда:

$$R = \sqrt{G \cdot \frac{M}{\omega^2}}$$

Подставив значения, получим: $R = 10079$ км. Это будет большой полуосью орбиты.

Начало времени моделирования — 12:00, и, предположим, мы хотим пройти первой точкой (город Рио-де-Жанейро) в 20:00. Тогда с начала времени моделирования до прохода над первой точкой пройдет 8 часов. Зная координаты Рио-де-Жанейро, можно вычислить истинную аномалию и долготу восходящего узла.

Определим начальный угол поворота Земли:

$$2 \cdot \pi \cdot (0.7790572732640 + 1.00273781191135448 \cdot 7671) = 281.09°$$

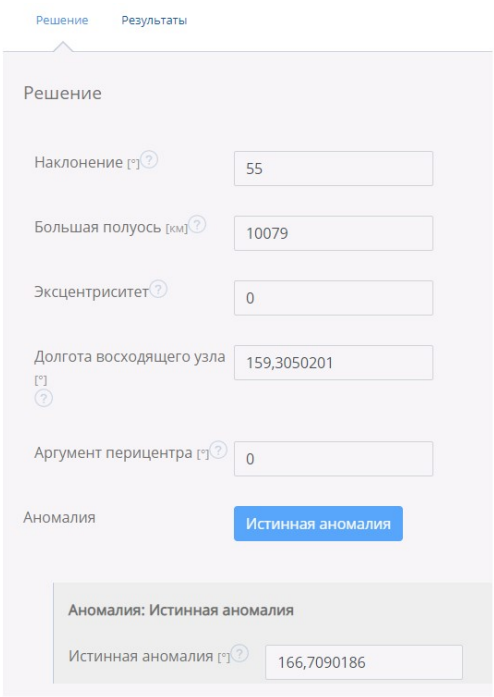
Где 1.00273781191135448 — скорость вращения Земли в оборот/сут, 7671 — количество дней, прошедших с эпохи J2000.0.

Для определения долготы восходящего узла прибавим начальный угол поворота Земли к долготе Рио-де-Жанейро и также прибавим угол, на который переместится спутник за 8 часов:

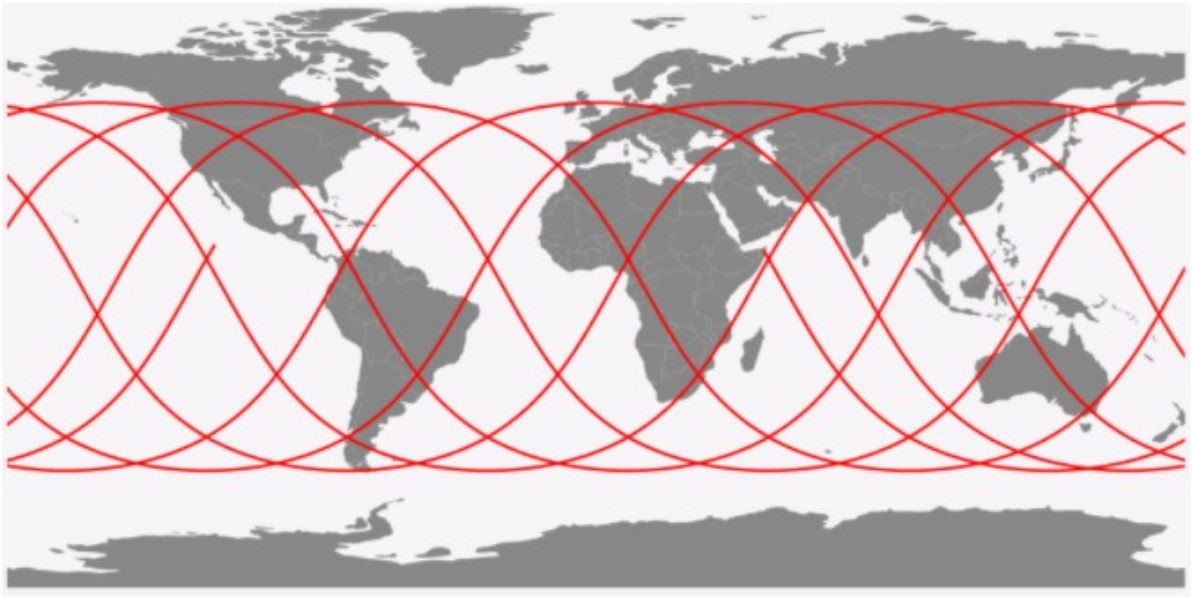
$$\Omega = 281.09 + \lambda_{\text{Рио}} + \omega_\lambda \cdot 8 \cdot 60 \cdot 60 = 159.31°$$

Для определения истинной аномалии от широты Рио-де-Жанейро также отнимем угол, на который переместится спутник за 8 часов:

$$M = \phi_{\text{Рио}} - \omega_\phi \cdot 8 \cdot 60 \cdot 60 = 166.71°$$


Решение
Результаты
Решение
Наклонение [°]
55
Большая полуось [км]
10079
Эксцентриситет
0
Долгота восходящего узла [°]
159,3050201
Аргумент перицентра [°]
0
Аномалия
Истинная аномалия
Аномалия: Истинная аномалия
Истинная аномалия [°]
166,7090186


Баллы: 8
Широта, °
Долгота, °
200
100
0
-100
-200
12:00 01-01
18:00 01-01
00:00 01-02
06:00 01-02
12:00 01-02
Высота, м
м
4,000,000
3,000,000
2,000,000
1,000,000
0



Орбита

Задача II.1.5.1. Орбита: энергобаланс и стабилизация (30 баллов)

Группа студентов из Канзас-Сити и Санкт-Петербурга планирует запустить космический аппарат с датчиком космической погоды на борту. Чтобы оба города могли получать данные с аппарата необходимо, чтобы аппарат пролетал над Канзас-Сити в промежуток с 13:00 по 15:00, над Санкт-Петербургом в промежуток с 15:00 по 18:00 по UTC каждый день.

Аппарат может быть расположен на произвольной орбите 20 февраля 2021 года.

Одной из подзадач аппарата является удержание его в положении, когда одна из его граней повернута на Солнце. Конструктивно на этой грани расположена солнечная батарея. Необходимо удерживать ориентацию солнечной батареи на Солнце по оси z, чтобы космическому аппарату хватало энергии на поддержание работы устройств.

Ёмкость аккумулятора: 10000 мАч

Начальный заряд аккумулятора: 70%

Расход с активной стабилизацией: 20мАч/с

Солнечная постоянная: 1367 Вт/м^2

Устройство	Потребление, мАч/с
Мотор маховика по одной оси	5
Навигатор	1
Полезная нагрузка	4

Координаты Канзас-Сити: 39.11417, 265.37254

Координаты Санкт-Петербурга: 59.93863, 30.31413

При заряде аккумулятора 0% аппарат выходит из строя и больше не является работоспособным.

Необходимо определить орбиту, на которой должен быть расположен аппарат, чтобы его работа была бесперебойной в течение 2 суток полета, а также обеспечить проход над городами в указанные промежутки времени.

Критерии оценки

1. $20/(2 \cdot 24 \cdot 60 \cdot 60)$ балла за секунду за ориентацию
2. 2.5 балла за попадание в указанный промежуток времени для каждого из городов.

Решение

Для простоты ориентации можно рассмотреть солнечно-синхронную орбиту, которая сохраняет свое положение относительно Солнца постоянным. В таком случае, спутник будет достаточно сориентировать, повернув вокруг оси Z всего один раз и затем сведя его угловую скорость к нулю.

Солнечно-синхронная орбита имеет наклонение около 98°. Также для простоты можно рассмотреть случай с круговой орбитой и принять, что ее эксцентриситет будет равен 0. Тогда аргумент перицентра также можно будет считать равным 0 для случая с круговой орбитой. Получаем, что три параметра орбиты уже определены.

Так как моделирование идет в течение двух суток, то аппарат должен совершать целое число витков за звездные сутки. Тогда будет верным соотношение:

$$T_{\text{орб}} = \frac{T_{ЗВ}}{N}$$

При этом между Канзас-Сити и Санкт-Петербургом желательно также иметь целое количество витков. Количество витков, которое аппарат сделает, пролетая над поверхностью Земли, расположенной между указанными городами, можно определить следующим образом:

$$T_{\text{орб}} \cdot k = \frac{\Delta\lambda}{\omega_\lambda}$$

Где скорость $\omega\lambda$ определяется как сумма скорости спутника и Земли:

$$\omega_\lambda = \omega_{\lambda\text{сп}} + \omega_{\lambda З} = \omega_{\text{сп}} \cdot cos98^\circ + \frac{360^\circ}{86164}$$

В свою очередь:

$$\omega_{\text{сп}} = \frac{360^\circ}{T_{\text{орб}}}$$

Таким образом, необходимо сделать несколько итераций расчетов, чтобы определить значения N и k. При N = 15 получаем k = 9.0053, что практически является целым числом. Тогда, приняв N = 15, можно определить период обращения спутника и значение большой полуоси:

$$a = \sqrt[3]{\mu(\frac{T_{\text{орб}}}{2\pi})^2} = 6932\text{км}$$

С начала времени моделирования до пролета над Канзас-Сити пройдет 13 часов. Чтобы определить долготу восходящего узла, необходимо к начальному углу поворота добавить долготу города Канзас-Сити и тот угол, на который провернется Земля за 13 часов:

$$\Omega = 331.36^\circ + \frac{1.00273781191135448 \cdot 360^\circ}{24/13} + 265.37254^\circ = 72.27^\circ$$

Где 331,36° — начальный угол поворота Земли, а 1.00273781191135448 — скорость вращения Земли в оборот/сут.

Истинная аномалия определяется как:

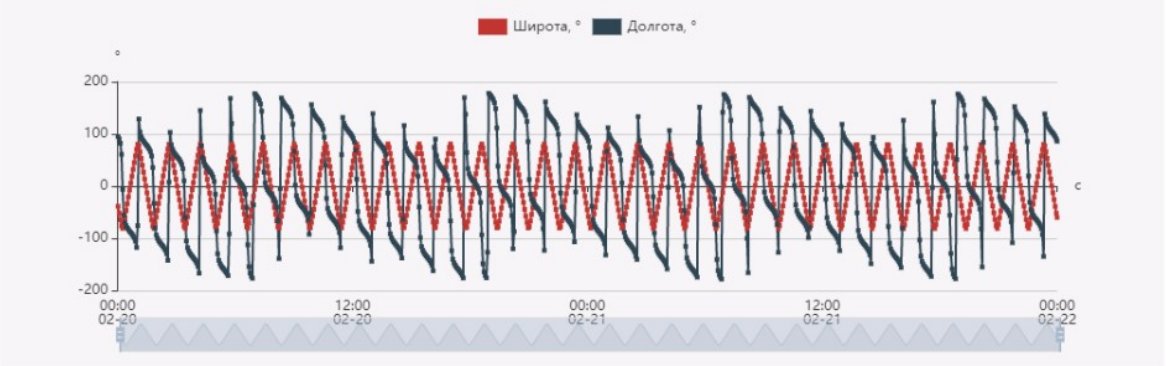
$$M = 0,147254073 \cdot 360 - - - \phi_{\text{Канзас}} = 336,97^\circ$$

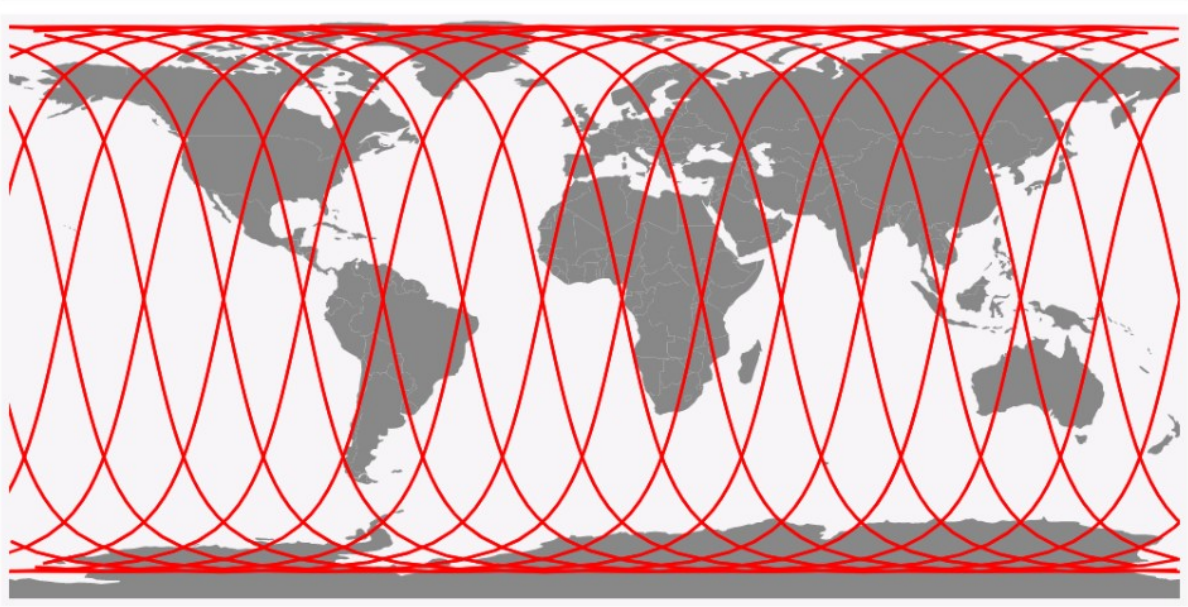
Где 0,147254073 — нецелая часть оборота спутника вокруг Земли при его движении по орбите в течение 13 часов. Пример программы ориентации:

```
var wheels;
var gyros;
var charge;

function setup() {
    wheels = spacecraft.devices[0];
    gyros = spacecraft.devices[1];
    charge = spacecraft.devices[3].functions[0].charge;
    wheels.functions[2].motor_torque = 0.0001;
}

var old_charge = charge;

function loop() {
    charge = spacecraft.devices[3].functions[0].charge;

    var i = 2;
    if ( old_charge - charge < (-1 + 7.5*wheels.enabled) &&
    ↪  Math.abs(gyros.functions[i].angular_velocity) > 0.00005 ) {
        wheels.enable();
        wheels.functions[i].motor_torque = 0;
        {
            wheels.functions[i].motor_torque = gyros.functions[i].angular_velocity *
            ↪  0.5;
        }
    } else if (old_charge - charge > 0 + 7.5*wheels.enabled) {
        wheels.enable();
        wheels.functions[i].motor_torque = -0.0002 +
        ↪  gyros.functions[i].angular_velocity * 0.01;
    } else {
        wheels.disable();
    }
    old_charge = charge;
}
```


Баллы: 30
Производство энергии Вт
Потребление Вт
21
18
15
12
9
6
3
0
00:00 02-20
12:00 02-20
00:00 02-21
12:00 02-21
00:00 02-22
c
Заряд Вт*с
10,000
8,000
6,000
4,000
2,000
0


Кватернион ориентации, элемент W
Кватернион ориентации, элемент X
Кватернион ориентации, элемент Y
Кватернион ориентации,
1/2
1.2
0.9
0.6
0.3
0
-0.3
-0.6
-0.9
-1.2
Угловая скорость X, рад/сек
Угловая скорость Y, рад/сек
Угловая скорость Z, рад/сек
рад/сек
0.018
0.015
0.012
0.009
0.006
0.003
-0.003





Задача II.1.5.2. Орбита: кодирование и декодирование (30 баллов)

Два спутника расположены на заданных орбитах (параметры задать, если необходимо). В определенные моменты времени спутник А передает данные спутнику Б. Периодически луч связи проходит сквозь верхние слои атмосферы, и часть данных может побиться. Чтобы этого не произошло, в программу управления каждым спутником нужно заложить алгоритм защиты и восстановления данных. Спутник А должен закодировать данные, включить передатчик и передать их в нужный момент времени. Спутник Б должен включить приемник и декодировать данные.

Необходимо написать программу управления для спутников А и Б.

На подумать:

- нужно ли ввести подбор орбиты тут;
- как задать время передачи, т. е. включения на спутнике А передатчика, а на спутнике Б приемника, или как определить условия для их нахождения;
- как портить данные;
- как задавать информацию на передачу.

Задачей спутника-ретранслятора является передача сообщений из пункта А в пункт Б. Сигнал должен в пункт Б должен приходить не позднее чем через 2 часа после прохода спутника над пунктом А.

Данные в ходе передачи повреждаются, поэтому со стороны пункта А нужно закодировать их с помощью помехоустойчивого кодирования, а на пункте Б раскодировать.

В течение 3 дней необходимо осуществить передачу 3 раза.

Необходимо подобрать орбиту для космического аппарата, а также:

1. задать программу для пункта А
2. задать программу спутника
3. задать программу для пункта Б

Начальные данные

Координаты пункта А:

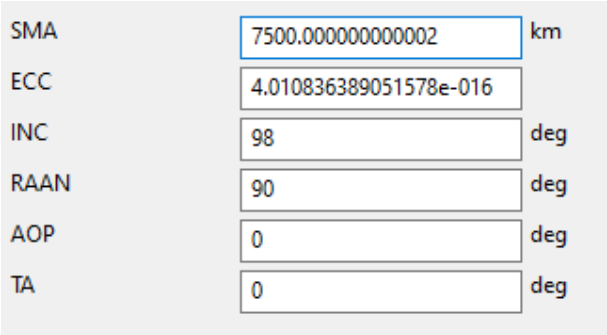
Координаты пункта Б:

Координаты Канзас-Сити: 39.11417, 265.37254

Координаты Санкт-Петербурга: 59.93863, 30.31413

Критерии оценки

- 2 балла за каждый прием данных в пункте Б.
- 3 балла за частично правильно переданные данные.
- 5 баллов за полностью правильно переданные данные.



Observer: GroundStation1

Start Time (UTC)	Stop Time (UTC)	Duration (s)
20 Feb 2021 01:55:44.492	20 Feb 2021 02:03:08.189	443.69628484
20 Feb 2021 14:59:29.800	20 Feb 2021 15:06:34.120	424.32049820
21 Feb 2021 01:16:27.300	21 Feb 2021 01:22:54.774	387.47455511
21 Feb 2021 14:19:40.256	21 Feb 2021 14:26:43.109	422.85289087
22 Feb 2021 02:22:56.197	22 Feb 2021 02:29:15.679	379.48166099
22 Feb 2021 13:41:39.399	22 Feb 2021 13:44:36.291	176.89191200
22 Feb 2021 15:26:51.050	22 Feb 2021 15:32:04.619	313.56937329

Number of events : 7

Observer: GroundStation2

Start Time (UTC)	Stop Time (UTC)	Duration (s)
20 Feb 2021 05:55:54.609	20 Feb 2021 06:02:10.516	375.90724679
20 Feb 2021 07:42:16.359	20 Feb 2021 07:48:45.532	389.17319023
20 Feb 2021 16:24:13.666	20 Feb 2021 16:31:12.262	418.59581277
20 Feb 2021 18:11:44.265	20 Feb 2021 18:17:05.086	320.82156156
21 Feb 2021 05:17:26.960	21 Feb 2021 05:20:01.486	154.52648909
21 Feb 2021 07:02:05.889	21 Feb 2021 07:09:26.224	440.33539135
21 Feb 2021 15:45:21.165	21 Feb 2021 15:50:48.527	327.36233465
21 Feb 2021 17:30:41.528	21 Feb 2021 17:37:46.711	425.18341786
22 Feb 2021 06:22:14.434	22 Feb 2021 06:29:24.407	429.97318424
22 Feb 2021 08:09:20.716	22 Feb 2021 08:14:28.530	307.81368603
22 Feb 2021 15:07:55.658	22 Feb 2021 15:09:19.563	83.905000915
22 Feb 2021 16:50:34.012	22 Feb 2021 16:57:57.921	443.90893708
22 Feb 2021 18:40:48.616	22 Feb 2021 18:41:55.906	67.289689207

Number of events : 13

Угол обзора станций: 30

Задачей спутника является сбор научных данных. За одну секунду спутник может собрать 5 пакетов данных размером 20 байт. Размер буфера спутника ограничен 1 Мбайтом. Данные необходимо передать на станцию с координатами (ШШШШ; ДДД). При передаче данных возможно их повреждение, поэтому желательно защитить их помехоустойчивым кодированием перед передачей со спутника и осуществить соответствующее декодирование на станции.

Для решения задачи необходимо:

- подобрать орбиту космического аппарата, которая будет позволять спутнику проходить над станцией для передачи данных;
- написать программу управления для спутника;
- написать программу управления для наземной станции для приема данных.

Аппарат может быть расположен на орбите 28 февраля 2021 года в 00:00 по UTC. В течение 24 часов на станцию необходимо передать как можно больше данных.

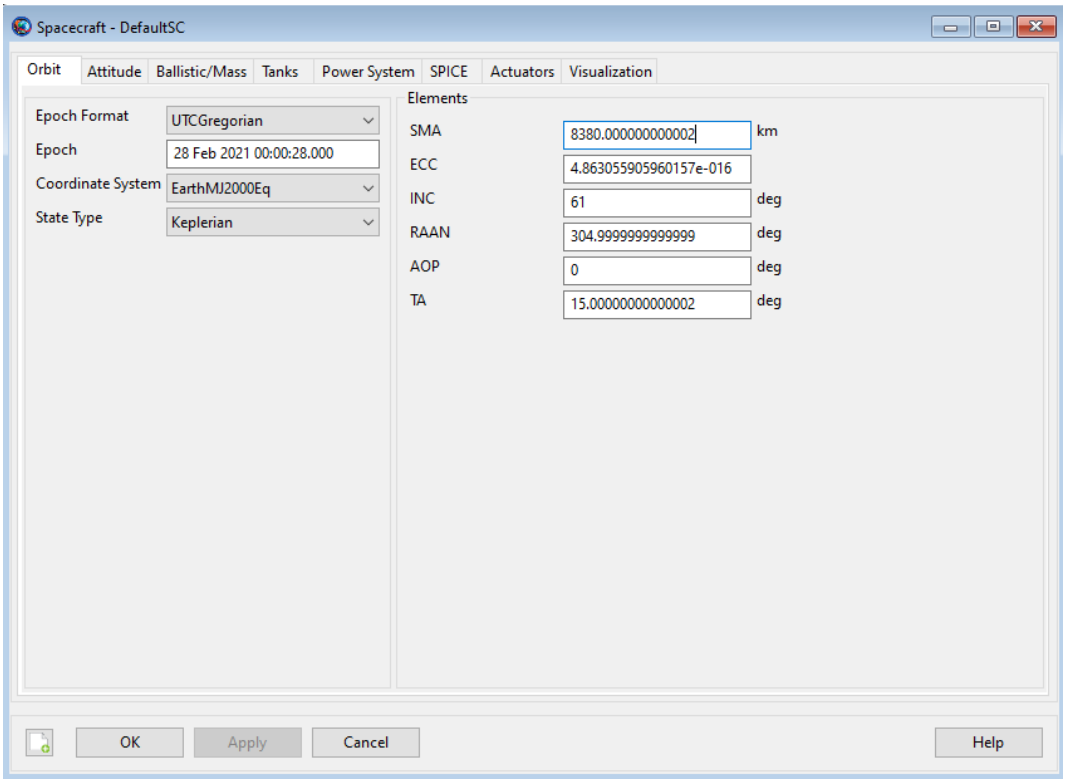
Скорость передачи данных: 3600 байт/сек

Угол над горизонтом, при котором возможен прием данных: 30

Критерий оценки:

- За каждые 225 правильно принятых пакетов начисляется 0.015625 балла. Максимально возможный балл — 30.

Ушуайя: -54.80722, 291.69556 Нанорталик: 60.14275, 314.76106



Observer: GroundStation1

Start Time (UTC)	Stop Time (UTC)	Duration (s)
28 Feb 2021 03:18:30.588	28 Feb 2021 03:32:37.870	847.28177022 (2)
28 Feb 2021 05:28:03.733	28 Feb 2021 05:45:56.632	1072.8991717 (3)
28 Feb 2021 07:41:39.865	28 Feb 2021 07:59:13.570	1053.7054573 (4)
28 Feb 2021 09:55:23.434	28 Feb 2021 10:13:15.651	1072.2171946 (5)
28 Feb 2021 12:08:19.917	28 Feb 2021 12:25:02.760	1002.8424184 (6)

Number of events : 5

Observer: GroundStation2

Start Time (UTC)	Stop Time (UTC)	Duration (s)
28 Feb 2021 00:35:51.258	28 Feb 2021 00:40:23.523	272.26444400 (1)
28 Feb 2021 12:55:42.825	28 Feb 2021 13:01:25.192	342.36741476 (7)
28 Feb 2021 15:00:17.049	28 Feb 2021 15:16:44.854	987.80442509 (8)
28 Feb 2021 17:11:30.501	28 Feb 2021 17:29:26.552	1076.0511593 (9)
28 Feb 2021 19:24:25.258	28 Feb 2021 19:42:22.675	1077.4164318 (10)
28 Feb 2021 21:37:08.201	28 Feb 2021 21:54:12.284	1024.0827553 (11)
28 Feb 2021 23:50:58.476	01 Mar 2021 00:00:27.988	569.51224833 (12)

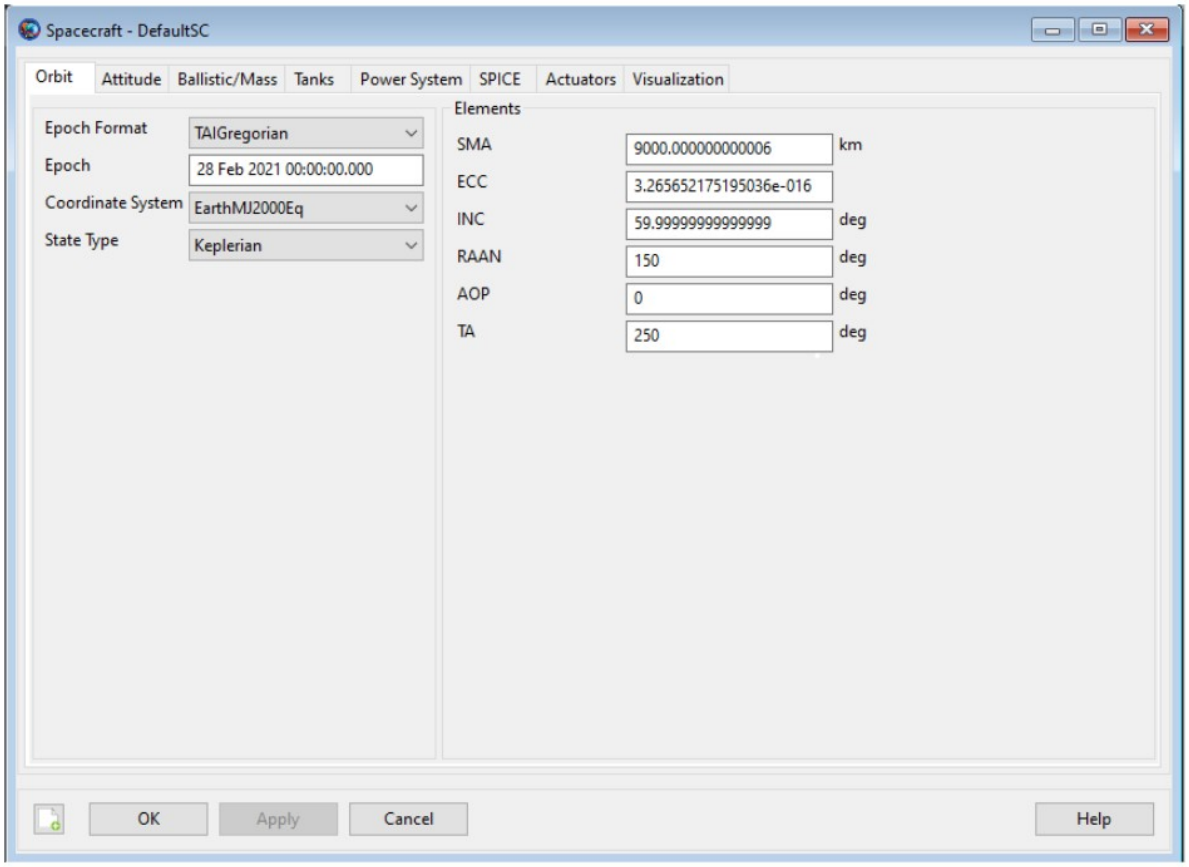
Number of events : 7

Решение

Для решения задачи целесообразно подобрать орбиту так, чтобы успело минимум передаться количество пакетов $N = 30/0.00048828125 = 61440$. Тогда будет возможным заработать максимальный балл.

Один пакет — это 100 байт, а скорость передачи — 3600 байт/сек. Отсюда можно определить минимальное количество времени, которое спутник должен быть на связи со станцией:

$$61440 \cdot 100/3600 \approx 1707 \text{ сек.}$$



Target: DefaultSC

Observer: GroundStation1

Start Time (UTC)	Stop Time (UTC)	Duration (s)
28 Feb 2021 00:02:47.215	28 Feb 2021 00:26:11.307	1404.0918092
28 Feb 2021 02:32:00.636	28 Feb 2021 02:52:08.704	1208.0680070
28 Feb 2021 16:16:57.767	28 Feb 2021 16:33:08.300	970.53249761
28 Feb 2021 18:39:59.726	28 Feb 2021 19:03:03.894	1384.1674149
28 Feb 2021 21:09:17.610	28 Feb 2021 21:32:33.298	1395.6876325
28 Feb 2021 23:39:17.618	28 Feb 2021 23:59:22.988	1205.3692582

Number of events : 6

Пример кода кодировщика:

```
let EARTH_RADIUS = 6371008.8;

let PACKET_SIZE = 100;

let produced_encode_pack = [];

function radians(angle) {
    return angle * Math.PI / 180;
```

```javascript
}

function make_coords(latitude, longitude) {
    return {latitude: radians(latitude), longitude: radians(longitude)};
}

function calculate_elevation(station_location, spacecraft_location) {
    let radius = EARTH_RADIUS;
    let height = spacecraft_location[2];
    let coords_1 = make_coords(station_location[0], station_location[1]);
    let coords_2 = make_coords(spacecraft_location[0], spacecraft_location[1]);
    let cos_angle = Math.cos(coords_1.latitude) * Math.cos(coords_2.latitude);
    cos_angle *= Math.cos(coords_1.longitude - coords_2.longitude);
    cos_angle += Math.sin(coords_1.latitude) * Math.sin(coords_2.latitude);
    let sin_angle = Math.sqrt(1.0 - cos_angle * cos_angle);
    let x = radius * sin_angle;
    let y = radius * (1.0 - cos_angle) + height;
    return [Math.atan2(y, x) - Math.acos(cos_angle), Math.sqrt(x * x + y * y)];
}

function check_visibility(station_location, spacecraft_location) {
    let elevation = calculate_elevation(station_location,spacecraft_location);
    if (elevation[0] < Math.PI / 9) {
        return false;
    }

    return true;
}

function setup() {
}

function enc(msg_in) {
    let gf_log = new Uint8Array([0, 0, 1, 25, 2, 50, 26, 198, 3, 223, 51, 238, 27,
    ↪  104, 199, 75, 4, 100, 224, 14, 52, 141, 239, 129, 28, 193, 105, 248, 200, 8,
    ↪  76, 113, 5, 138, 101, 47, 225, 36, 15, 33, 53, 147, 142, 218, 240, 18, 130,
    ↪  69, 29, 181, 194, 125, 106, 39, 249, 185, 201, 154, 9, 120, 77, 228, 114, 166,
    ↪  6, 191, 139, 98, 102, 221, 48, 253, 226, 152, 37, 179, 16, 145, 34, 136, 54,
    ↪  208, 148, 206, 143, 150, 219, 189, 241, 210, 19, 92, 131, 56, 70, 64, 30, 66,
    ↪  182, 163, 195, 72, 126, 110, 107, 58, 40, 84, 250, 133, 186, 61, 202, 94, 155,
    ↪  159, 10, 21, 121, 43, 78, 212, 229, 172, 115, 243, 167, 87, 7, 112, 192, 247,
    ↪  140, 128, 99, 13, 103, 74, 222, 237, 49, 197, 254, 24, 227, 165, 153, 119, 38,
    ↪  184, 180, 124, 17, 68, 146, 217, 35, 32, 137, 46, 55, 63, 209, 91, 149, 188,
    ↪  207, 205, 144, 135, 151, 178, 220, 252, 190, 97, 242, 86, 211, 171, 20, 42,
    ↪  93, 158, 132, 60, 57, 83, 71, 109, 65, 162, 31, 45, 67, 216, 183, 123, 164,
    ↪  118, 196, 23, 73, 236, 127, 12, 111, 246, 108, 161, 59, 82, 41, 157, 85, 170,
    ↪  251, 96, 134, 177, 187, 204, 62, 90, 203, 89, 95, 176, 156, 169, 160, 81, 11,
    ↪  245, 22, 235, 122, 117, 44, 215, 79, 174, 213, 233, 230, 231, 173, 232, 116,
    ↪  214, 244, 234, 168, 80, 88, 175])
```

```
43      let gf_exp = new Uint8Array([1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205,
        ↪  135, 19, 38, 76, 152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24, 48, 96,
        ↪  192, 157, 39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193, 159, 35,
        ↪  70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111, 222, 161, 95,
        ↪  190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30, 60, 120, 240, 253, 231,
        ↪  211, 187, 107, 214, 177, 127, 254, 225, 223, 163, 91, 182, 113, 226, 217, 175,
        ↪  67, 134, 17, 34, 68, 136, 13, 26, 52, 104, 208, 189, 103, 206, 129, 31, 62,
        ↪  124, 248, 237, 199, 147, 59, 118, 236, 197, 151, 51, 102, 204, 133, 23, 46,
        ↪  92, 184, 109, 218, 169, 79, 158, 33, 66, 132, 21, 42, 84, 168, 77, 154, 41,
        ↪  82, 164, 85, 170, 73, 146, 57, 114, 228, 213, 183, 115, 230, 209, 191, 99,
        ↪  198, 145, 63, 126, 252, 229, 215, 179, 123, 246, 241, 255, 227, 219, 171, 75,
        ↪  150, 49, 98, 196, 149, 55, 110, 220, 165, 87, 174, 65, 130, 25, 50, 100, 200,
        ↪  141, 7, 14, 28, 56, 112, 224, 221, 167, 83, 166, 81, 162, 89, 178, 121, 242,
        ↪  249, 239, 195, 155, 43, 86, 172, 69, 138, 9, 18, 36, 72, 144, 61, 122, 244,
        ↪  245, 247, 243, 251, 235, 203, 139, 11, 22, 44, 88, 176, 125, 250, 233, 207,
        ↪  131, 27, 54, 108, 216, 173, 71, 142, 1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116,
        ↪  232, 205, 135, 19, 38, 76, 152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24,
        ↪  48, 96, 192, 157, 39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193,
        ↪  159, 35, 70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111, 222,
        ↪  161, 95, 190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30, 60, 120, 240,
        ↪  253, 231, 211, 187, 107, 214, 177, 127, 254, 225, 223, 163, 91, 182, 113, 226,
        ↪  217, 175, 67, 134, 17, 34, 68, 136, 13, 26, 52, 104, 208, 189, 103, 206, 129,
        ↪  31, 62, 124, 248, 237, 199, 147, 59, 118, 236, 197, 151, 51, 102, 204, 133,
        ↪  23, 46, 92, 184, 109, 218, 169, 79, 158, 33, 66, 132, 21, 42, 84, 168, 77,
        ↪  154, 41, 82, 164, 85, 170, 73, 146, 57, 114, 228, 213, 183, 115, 230, 209,
        ↪  191, 99, 198, 145, 63, 126, 252, 229, 215, 179, 123, 246, 241, 255, 227, 219,
        ↪  171, 75, 150, 49, 98, 196, 149, 55, 110, 220, 165, 87, 174, 65, 130, 25, 50,
        ↪  100, 200, 141, 7, 14, 28, 56, 112, 224, 221, 167, 83, 166, 81, 162, 89, 178,
        ↪  121, 242, 249, 239, 195, 155, 43, 86, 172, 69, 138, 9, 18, 36, 72, 144, 61,
        ↪  122, 244, 245, 247, 243, 251, 235, 203, 139, 11, 22, 44, 88, 176, 125, 250,
        ↪  233, 207, 131, 27, 54, 108, 216, 173, 71, 142])
44
45      let nsym = 10
46      // regen for different nsym
47      let gen = new Uint8Array([0x01,0xd8,0xc2,0x9f,0x6f,0xc7,0x5e,0x5f,0x71,0x9d,0xc1])
48
49      let msg_out = new Uint8Array([...msg_in, ...Array(gen.length - 1)]) // may be
        ↪  optimised
50
51      let lgen = []
52      for (let j = 0; j < gen.length; j++) {
53          lgen.push(gf_log[gen[j]])
54      }
55
56      for (let i = 0; i < msg_in.length; i++) {
57          const coef = msg_out[i];
58          if (coef != 0) {
59              const lcoef = gf_log[coef]
60              for (let j = 1; j < gen.length; j++) {
61                  msg_out[i + j] ^= gf_exp[lcoef + lgen[j]]
62              }
63          }
64      }
65
66      for (let i = 0; i < msg_in.length; i++) {
67          msg_out[i] = msg_in[i]
68      }
69      return msg_out
70  }
71
```

```
let my_k = 10;

function loop() {
    for (var qwe = 0; qwe < 10; qwe++) {
        let produced_encode = [];
        let producer = spacecraft.devices[1].functions[0];
        let produced_packet = producer.read(my_k);


        if (produced_packet.length == 0) {
            return;
        }
        else{
            produced_encode = enc(produced_packet)
        }

        if (produced_packet.length != my_k) {
            runtime.debug('KS= ' + produced_packet.toString())
        }

        let  produced_encode_packet = new Uint8Array(produced_encode)
        produced_encode_pack.push(produced_encode_packet);

        let navigator = spacecraft.devices[2].functions[0];
        let spacecraft_location = navigator.location;
        let station_location = [-54.80722, -68.30444];
        if (!check_visibility(station_location, spacecraft_location)) {
            return;
        }

        let transmitter = spacecraft.devices[0].functions[0];

        let k = 0
        //transmitter.transmit(produced_encode_packet.shift());

        for (var interq = 0; interq < 12; interq++) {
            if (produced_encode_pack.length > 0){
                transmitter.transmit(produced_encode_pack.shift());
            }
        }
    }
}
```

Пример кода декодировщика:

```
function setup() {
}


// decoder
```

```
var gf_log = new Uint8Array([0, 0, 1, 25, 2, 50, 26, 198, 3, 223, 51, 238, 27, 104,
↪  199, 75, 4, 100, 224, 14, 52, 141, 239, 129, 28, 193, 105, 248, 200, 8, 76, 113,
↪  5, 138, 101, 47, 225, 36, 15, 33, 53, 147, 142, 218, 240, 18, 130, 69, 29, 181,
↪  194, 125, 106, 39, 249, 185, 201, 154, 9, 120, 77, 228, 114, 166, 6, 191, 139, 98,
↪  102, 221, 48, 253, 226, 152, 37, 179, 16, 145, 34, 136, 54, 208, 148, 206, 143,
↪  150, 219, 189, 241, 210, 19, 92, 131, 56, 70, 64, 30, 66, 182, 163, 195, 72, 126,
↪  110, 107, 58, 40, 84, 250, 133, 186, 61, 202, 94, 155, 159, 10, 21, 121, 43, 78,
↪  212, 229, 172, 115, 243, 167, 87, 7, 112, 192, 247, 140, 128, 99, 13, 103, 74,
↪  222, 237, 49, 197, 254, 24, 227, 165, 153, 119, 38, 184, 180, 124, 17, 68, 146,
↪  217, 35, 32, 137, 46, 55, 63, 209, 91, 149, 188, 207, 205, 144, 135, 151, 178,
↪  220, 252, 190, 97, 242, 86, 211, 171, 20, 42, 93, 158, 132, 60, 57, 83, 71, 109,
↪  65, 162, 31, 45, 67, 216, 183, 123, 164, 118, 196, 23, 73, 236, 127, 12, 111, 246,
↪  108, 161, 59, 82, 41, 157, 85, 170, 251, 96, 134, 177, 187, 204, 62, 90, 203, 89,
↪  95, 176, 156, 169, 160, 81, 11, 245, 22, 235, 122, 117, 44, 215, 79, 174, 213,
↪  233, 230, 231, 173, 232, 116, 214, 244, 234, 168, 80, 88, 175]);
var gf_exp = new Uint8Array([1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205, 135,
↪  19, 38, 76, 152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24, 48, 96, 192, 157,
↪  39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193, 159, 35, 70, 140, 5,
↪  10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111, 222, 161, 95, 190, 97, 194, 153,
↪  47, 94, 188, 101, 202, 137, 15, 30, 60, 120, 240, 253, 231, 211, 187, 107, 214,
↪  177, 127, 254, 225, 223, 163, 91, 182, 113, 226, 217, 175, 67, 134, 17, 34, 68,
↪  136, 13, 26, 52, 104, 208, 189, 103, 206, 129, 31, 62, 124, 248, 237, 199, 147,
↪  59, 118, 236, 197, 151, 51, 102, 204, 133, 23, 46, 92, 184, 109, 218, 169, 79,
↪  158, 33, 66, 132, 21, 42, 84, 168, 77, 154, 41, 82, 164, 85, 170, 73, 146, 57,
↪  114, 228, 213, 183, 115, 230, 209, 191, 99, 198, 145, 63, 126, 252, 229, 215, 179,
↪  123, 246, 241, 255, 227, 219, 171, 75, 150, 49, 98, 196, 149, 55, 110, 220, 165,
↪  87, 174, 65, 130, 25, 50, 100, 200, 141, 7, 14, 28, 56, 112, 224, 221, 167, 83,
↪  166, 81, 162, 89, 178, 121, 242, 249, 239, 195, 155, 43, 86, 172, 69, 138, 9, 18,
↪  36, 72, 144, 61, 122, 244, 245, 247, 243, 251, 235, 203, 139, 11, 22, 44, 88, 176,
↪  125, 250, 233, 207, 131, 27, 54, 108, 216, 173, 71, 142, 1, 2, 4, 8, 16, 32, 64,
↪  128, 29, 58, 116, 232, 205, 135, 19, 38, 76, 152, 45, 90, 180, 117, 234, 201, 143,
↪  3, 6, 12, 24, 48, 96, 192, 157, 39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119,
↪  238, 193, 159, 35, 70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111,
↪  222, 161, 95, 190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30, 60, 120, 240,
↪  253, 231, 211, 187, 107, 214, 177, 127, 254, 225, 223, 163, 91, 182, 113, 226,
↪  217, 175, 67, 134, 17, 34, 68, 136, 13, 26, 52, 104, 208, 189, 103, 206, 129, 31,
↪  62, 124, 248, 237, 199, 147, 59, 118, 236, 197, 151, 51, 102, 204, 133, 23, 46,
↪  92, 184, 109, 218, 169, 79, 158, 33, 66, 132, 21, 42, 84, 168, 77, 154, 41, 82,
↪  164, 85, 170, 73, 146, 57, 114, 228, 213, 183, 115, 230, 209, 191, 99, 198, 145,
↪  63, 126, 252, 229, 215, 179, 123, 246, 241, 255, 227, 219, 171, 75, 150, 49, 98,
↪  196, 149, 55, 110, 220, 165, 87, 174, 65, 130, 25, 50, 100, 200, 141, 7, 14, 28,
↪  56, 112, 224, 221, 167, 83, 166, 81, 162, 89, 178, 121, 242, 249, 239, 195, 155,
↪  43, 86, 172, 69, 138, 9, 18, 36, 72, 144, 61, 122, 244, 245, 247, 243, 251, 235,
↪  203, 139, 11, 22, 44, 88, 176, 125, 250, 233, 207, 131, 27, 54, 108, 216, 173, 71,
↪  142]);

var field_charac = 255;

function gf_pow(x, power) {
    return gf_exp[((field_charac*10 + gf_log[x] * power) % field_charac)];
}
function gf_add(x, y) {
    return (x ^ y);
}
function gf_sub(x, y) {
    return (x ^ y);
}
function gf_neg(x) {
    return x;
}
```

```
function gf_inverse(x) {
    return gf_exp[(field_charac - gf_log[x])];
}
function gf_mul(x, y) {
    if (((x === 0) || (y === 0))) {
        return 0;
    }
    return gf_exp[((field_charac*10 + gf_log[x] + gf_log[y]) % field_charac)];
}
function gf_div(x, y) {
    if ((y === 0)) {
        throw new ZeroDivisionError();
    }
    if ((x === 0)) {
        return 0;
    }
    return gf_exp[(((gf_log[x] + field_charac) - gf_log[y]) % field_charac)];
}

function gf_poly_scale(p, x) {
    let ret = []
    for (const i in p) {
        ret.push(gf_mul(p[i], x))
    }
    return ret
}

function gf_poly_add(p, q) {
    var r;
    r = Array(Math.max(p.length, q.length));
    r.splice(r.length - p.length, r.length, ...p)
    for (var i = 0; (i < q.length); i += 1) {
        r[((i + r.length) - q.length)] ^= q[i];
    }
    return r;
}

function gf_poly_mul(p, q) {
    var lp, lq, qj, r;
    r = Array(p.length + q.length - 1).fill(0);
    // lp = [gf_log[p[i]] for i in range(len(p))]
    lp = []
    for (const i in p) {
        lp.push(gf_log[p[i]])
    }

    for (var j = 0; (j < q.length); j += 1) {
        qj = q[j];
        if ((qj !== 0)) {
            lq = gf_log[qj];
            for (var i = 0, _pj_b = p.length; (i < _pj_b); i += 1) {
                if ((p[i] !== 0)) {
                    r[(i + j)] ^= gf_exp[(lp[i] + lq)];
                }
            }
        }
    }
    return r;
}
function gf_poly_mul_simple(p, q) {
```

```
    /* Multiply two polynomials, inside Galois Field */
    var r;
    r = Array(((p.length + q.length) - 1));
    for (var j = 0, _pj_a = q.length; (j < _pj_a); j += 1) {
        for (var i = 0, _pj_b = p.length; (i < _pj_b); i += 1) {
            r[(i + j)] ^= gf_mul(p[i], q[j]);
        }
    }
    return r;
}
function gf_poly_neg(poly) {
    /* Returns the polynomial with all coefficients negated. In GF(2^p), negation does not change the coefficient, so we return the polynomial as-is. */
    return poly;
}



function gf_poly_div(dividend, divisor) {
    /* Fast polynomial division by using Extended Synthetic Division and optimized for GF(2^p) computations (doesn't work with standard polynomials outside of this galois field). */
    var coef, msg_out, separator;
    msg_out = dividend.slice();
    for (var i = 0, _pj_a = (dividend.length - (divisor.length - 1)); (i < _pj_a); i += 1) {
        coef = msg_out[i];
        if ((coef !== 0)) {
            for (var j = 1, _pj_b = divisor.length; (j < _pj_b); j += 1) {
                if ((divisor[j] !== 0)) {
                    msg_out[(i + j)] ^= gf_mul(divisor[j], coef);
                }
            }
        }
    }
    separator = (- (divisor.length - 1));
    return [msg_out.slice(0, separator), msg_out.slice(separator)];
}
function gf_poly_square(poly) {
    /* Linear time implementation of polynomial squaring. For details, see paper: "A fast software implementation for arithmetic operations in GF (2n)". De Win, E., Bosselaers, A., Vandenberghe, S., De Gersem, P., & Vandewalle, J. (1996, January). In Advances in Cryptology - Asiacrypt'96 (pp. 65-76). Springer Berlin Heidelberg. */
    var k, length, out, p;
    length = poly.length;
    out = Array(((2 * length) - 1));
    for (var i = 0, _pj_a = (length - 1); (i < _pj_a); i += 1) {
        p = poly[i];
        k = (2 * i);
        if ((p !== 0)) {
            out[k] = gf_exp[(2 * gf_log[p])];
        }
    }
    out[((2 * length) - 2)] = gf_exp[(2 * gf_log[poly[(length - 1)]])];
    if ((out[0] === 0)) {
        out[0] = ((2 * poly[1]) - 1);
    }
    return out;
}
```

```
function gf_poly_eval(poly, x) {
    /* Evaluates a polynomial in GF(2^p) given the value for x. This is based on Horner's scheme for maximum efficiency. */
    var y;
    y = poly[0];
    for (var i = 1, _pj_a = poly.length; (i < _pj_a); i += 1) {
        y = (gf_mul(y, x) ^ poly[i]);
    }
    return y;
}
function rs_calc_syndromes(msg, nsym, fcr = 0, generator = 2) {
    /*Given the received codeword msg and the number of error correcting symbols (nsym), computes the syndromes polynomial.
    Mathematically, it's essentially equivalent to a Fourrier Transform (Chien search being the inverse).
    */
    //    [0] + [gf_poly_eval(msg, gf_pow(generator, i+fcr)) for i in range(nsym)]
    var re = [0]
    for (let i = 0; i < nsym; i++) {
        re.push(gf_poly_eval(msg, gf_pow(generator, i + fcr)));
    }
    return re;
}


function rs_forney_syndromes(synd, pos, nmess, generator = 2) {
    var erase_pos_reversed, fsynd, x;
    erase_pos_reversed = []

    for (const p in pos) {
        erase_pos_reversed.push(nmess - 1 - p);
    }

    fsynd = synd.slice(1);

    for (var i = 0, _pj_a = pos.length; (i < _pj_a); i += 1) {
        x = gf_pow(generator, erase_pos_reversed[i]);
        for (var j = 0, _pj_b = (fsynd.length - 1); (j < _pj_b); j += 1) {
            fsynd[j] = (gf_mul(fsynd[j], x) ^ fsynd[(j + 1)]);
        }
    }
    return fsynd;
}



function rs_find_error_locator(synd, nsym, erase_count = 0) {
    /* Find error/errata locator and evaluator polynomials with Berlekamp-Massey algorithm */
    var K, delta, err_loc, errs, new_loc, old_loc, synd_shift;
    err_loc = [1];
    old_loc = [1];

    synd_shift = 0;
    if ((synd.length > nsym)) {
        synd_shift = (synd.length - nsym);
    }
    for (var i = 0, _pj_a = (nsym - erase_count); (i < _pj_a); i += 1) {
        K = (i + synd_shift);
```

```
191
192         delta = synd[K];
193         for (var j = 1, _pj_b = err_loc.length; (j < _pj_b); j += 1) {
194             delta ^= gf_mul(err_loc.slice((- (j + 1)))[0], synd[(K - j)]);
195         }
196         old_loc.push(0);
197         if ((delta !== 0)) {
198             if ((old_loc.length > err_loc.length)) {
199                 new_loc = gf_poly_scale(old_loc, delta);
200                 old_loc = gf_poly_scale(err_loc, gf_inverse(delta));
201                 err_loc = new_loc;
202             }
203             err_loc = gf_poly_add(err_loc, gf_poly_scale(old_loc, delta));
204         }
205     }
206     while (err_loc.indexOf(0) === 0) {
207         err_loc.shift();
208     };
209     errs = (err_loc.length - 1);
210     if (((((errs - erase_count) * 2) + erase_count) > nsym)) {
211         // throw new SyntaxError("Too many errors to correct");
212     }
213
214     return err_loc;
215 }
216 function rs_find_errata_locator(e_pos, generator = 2) {
217     /* Compute the erasures/errors/errata locator polynomial from the
        ↪  erasures/errors/errata positions (the positions must be relative to the x
        ↪  coefficient, eg: "hello worldxxxxxxxx" is tampered to "h_ll_ worldxxxxxxxx"
        ↪  with xxxxxxxx being the ecc of length n-k=9, here the string positions are
        ↪  [1, 4], but the coefficients are reversed since the ecc characters are placed
        ↪  as the first coefficients of the polynomial, thus the coefficients of the
        ↪  erased characters are n-1 - [1, 4] = [18, 15] = erasures_loc to be specified
        ↪  as an argument. */
218     var e_loc;
219     e_loc = [1];
220     for (var i = 0, _pj_b = e_pos.length; (i < _pj_b); i += 1) {
221         e_loc = gf_poly_mul(e_loc, gf_poly_add([1], [gf_pow(generator, e_pos[i]),
            ↪  0]));
222     }
223     return e_loc;
224 }
225 function rs_find_errors(err_loc, nmess, generator = 2) {
226     /* Find the roots (ie, where evaluation = zero) of error polynomial by bruteforce
        ↪  trial, this is a sort of Chien's search (but less efficient, Chien's search is
        ↪  a way to evaluate the polynomial such that each evaluation only takes constant
        ↪  time). */
227     var err_pos, errs;
228     errs = (err_loc.length - 1);
229     err_pos = [];
230     for (var i = 0; (i < nmess); i += 1) {
231         if ((gf_poly_eval(err_loc, gf_pow(generator, i)) === 0)) {
232             err_pos.push(((nmess - 1) - i));
233         }
234     }
235     if ((err_pos.length !== errs)) {
236         // throw new SyntaxError("Too many (or few) errors found by Chien Search for
            ↪  the errata locator polynomial!");
237     }
238     return err_pos;
```

```
}
function rs_find_error_evaluator(synd, err_loc, nsym) {
    /* Compute the error (or erasures if you supply sigma=erasures locator polynomial,
    ↪  or errata) evaluator polynomial Omega from the syndrome and the
    ↪  error/erasures/errata locator Sigma. Omega is already computed at the same
    ↪  time as Sigma inside the Berlekamp-Massey implemented above, but in case you
    ↪  modify Sigma, you can recompute Omega afterwards using this method, or just
    ↪  ensure that Omega computed by BM is correct given Sigma. */
    var _, remainder;
    let a = Array(nsym + 2).fill(0);
    a[0] = 1;
    [_, remainder] = gf_poly_div(gf_poly_mul(synd, err_loc), a);
    return remainder;
}



function rs_correct_errata(msg_in, synd, err_pos, fcr = 0, generator = 2) {
    /* Forney algorithm, computes the values (error magnitude) to correct the input
    ↪  message. */
    var E, X, Xi, Xi_inv, Xlength, coef_pos, err_loc, err_loc_prime,
    ↪  err_loc_prime_tmp, l, magnitude, msg, y;
    msg = msg_in.slice();
    coef_pos = function () {
        var _pj_a = [], _pj_b = err_pos;
        for (var _pj_c = 0, _pj_d = _pj_b.length; (_pj_c < _pj_d); _pj_c += 1) {
            var p = _pj_b[_pj_c];
            _pj_a.push(((msg.length - 1) - p));
        }
        return _pj_a;
    }
        .call(this);
    err_loc = rs_find_errata_locator(coef_pos, generator);
    err_eval = rs_find_error_evaluator(synd.slice().reverse(), err_loc, err_loc.length
    ↪  - 1).slice().reverse()

    X = [];
    for (var i = 0, _pj_a = coef_pos.length; (i < _pj_a); i += 1) {
        l = (field_charac - coef_pos[i]);
        X.push(gf_pow(generator, (- l)));
    }
    E = Array(msg.length);
    Xlength = X.length;
    for (var i = 0, _pj_a = X.length; (i < _pj_a); i += 1) {
        Xi = X[i];
        Xi_inv = gf_inverse(Xi);
        err_loc_prime_tmp = [];
        for (var j = 0, _pj_b = Xlength; (j < _pj_b); j += 1) {
            if ((j !== i)) {
                err_loc_prime_tmp.push(gf_sub(1, gf_mul(Xi_inv, X[j])));
            }
        }
        err_loc_prime = 1;
        for (var coef, _pj_d = 0, _pj_c = err_loc_prime_tmp.length; (_pj_d < _pj_c);
        ↪  _pj_d += 1) {
            coef = err_loc_prime_tmp[_pj_d];
            err_loc_prime = gf_mul(err_loc_prime, coef);
        }
        if ((err_loc_prime === 0)) {
```

```
290             throw new SyntaxError("Decoding failed: Forney algorithm could not
                ↪  properly detect where the errors are located (errata locator prime is
                ↪  0).");
291         }
292 
293         y = gf_poly_eval(err_eval.slice().reverse(), Xi_inv)
294 
295         y = gf_mul(gf_pow(Xi, (1 - fcr)), y);
296         magnitude = gf_div(y, err_loc_prime);
297         E[err_pos[i]] = magnitude;
298     }
299     msg = gf_poly_add(msg.slice(), E);
300     return msg;
301 }
302 function dec(msg_in, nsym = 10, fcr = 0, generator = 2, only_erasures = false) {
303     var erase_pos, err_loc, fsynd, msg_out, synd;
304     if ((msg_in.length > field_charac)) {
305         throw new ValueError(("Message is too long (" + msg_in.length + " when max is
            ↪  " + field_charac + ")"));
306     }
307     msg_out = msg_in.slice();
308     erase_pos = [];
309     if ((erase_pos.length > nsym)) {
310         throw new SyntaxError("Too many erasures to correct");
311     }
312     synd = rs_calc_syndromes(msg_out, nsym, fcr, generator);
313     if ((Math.max(synd) === 0)) {
314         return [msg_out.slice(0, (- nsym)), msg_out.slice((- nsym)), []];
315     }
316     fsynd = rs_forney_syndromes(synd, erase_pos, msg_out.length, generator);
317     err_loc = rs_find_error_locator(fsynd, nsym, erase_pos.length);
318 
319     err_pos = rs_find_errors(err_loc.slice().reverse(), msg_out.length, generator)
320     if ((err_pos === null)) { // TODO ? Dont work
321         throw new SyntaxError("Could not locate error");
322     }
323     msg_out = rs_correct_errata(msg_out.slice(), synd, erase_pos.concat(err_pos), fcr,
        ↪  generator);
324     synd = rs_calc_syndromes(msg_out, nsym, fcr, generator);
325     if ((Math.max(synd) > 0)) {
326         throw new SyntaxError("Could not correct message");
327     }
328     return [msg_out.slice(0, (- nsym)), msg_out.slice((- nsym)), (erase_pos +
        ↪  err_pos)];
329 }
330 
331 let my_k = 10;
332 
333 function loop() {
334     let new_received = []
335     let receiver = station.devices[0].functions[0];
336     for (var iter1 = 0; iter1 < 280; iter1++) {
337 
338         let received_queue = receiver.receive(my_k+10);
339 
340 
341         if (received_queue.length == 0) {
342             return;
343         }
344         else{
```

```
345             new_received = dec(received_queue)[0]
346         }
347
348         if (received_queue.length != my_k+10) {
349             runtime.debug(received_queue.toString())
350         }
351
352         let consumer = station.devices[1].functions[0];
353
354         let  new_received_packet = new Uint8Array(new_received)
355         consumer.write(new_received_packet);
356     }
357 }
```

Решение № 1:

Баллы: 30