§4 Заключительный этап: командная часть

В командной части заключительного этапа участники должны решать прикладную задачу на орбите Земли с помощью программно-аппаратного комплекса, моделирующего полет и работу микроспутника на земной орбите. Продолжительность командной части — 3 дня (всего 18 астрономических часов).

4.1. Набор заданий

Команды могут выполнить любое из следующих заданий. Проверка задания выполняется на рабочем месте команды или специальном стенде по заданным объективным критериям. За каждое выполненное задание команда получает баллы. Возможны штрафы за превышение числа попыток и испорченное оборудование (5 баллов за каждый выведенный из строя блок спутника).

Задание	**Описание задачи**	**Баллы**	**Штрафы**
1. Проверка радиоканала	Передать сообщение через канал телеметрии	1	нет
2. Проверка магнитометра	Передать через канал телеметрии показания магнитометра (состояние датчика и показания индукции магнитного поля по трем векторам)	1	нет
3. Проверка датчика угловой скорости	Передать через канал телеметрии своего показания датчика угловой скорости (состояние прибора, угловая скорость по трем осям)	1	нет
4. Проверка солнечных датчиков	Передать через канал телеметрии сырые показания (два числа) всех четырех солнечных датчиков.	2	нет
5. Проверка камеры и передатчика	Сделать снимок с помощью камеры и передать его на наземный измерительный пункт по каналу высокопроизводительной связи	4	нет
6. Сборка аппарата и проверка систем	Установить все компоненты конструктора на корпус, сбалансировать его так, чтобы на подвесе он располагался параллельно полу, и передать показания со всех датчиков в телеметрию	3	нет
7. Проверка двигателя-маховика	Совершить вращение конструктора на подвесе с помощью двигателя-маховика	2	нет
8. Стабилизировать спутник	Стабилизировать конструктор спутника после заданного начального вращения	40 / число секунд (не более 1 мин)	нет
9. Ориентация спутника	Ориентировать конструктор спутника панелями солнечных батарей на имитатор солнца	150 / число секунд (не более 1 мин)	нет

10. Составление отчета о магнитном поле Земли	Получить модуль вектора магнитного поля в зависимости от координаты на поверхности	считается ср. отклонение от идеального графика: 15 * exp(-Δ)	3 далее штраф в 1 балл за попытку
11. Съемка точки на поверхности Земли и передача снимка на НИП	Сфотографировать заданную точку на поверхности глобуса и передать снимок на наземный измерительны пункт	15 за любой снимок Земли или 35 за снимок требуемой точки	5 далее штраф в 1 балл за попытку

4.2. Описание конструктора

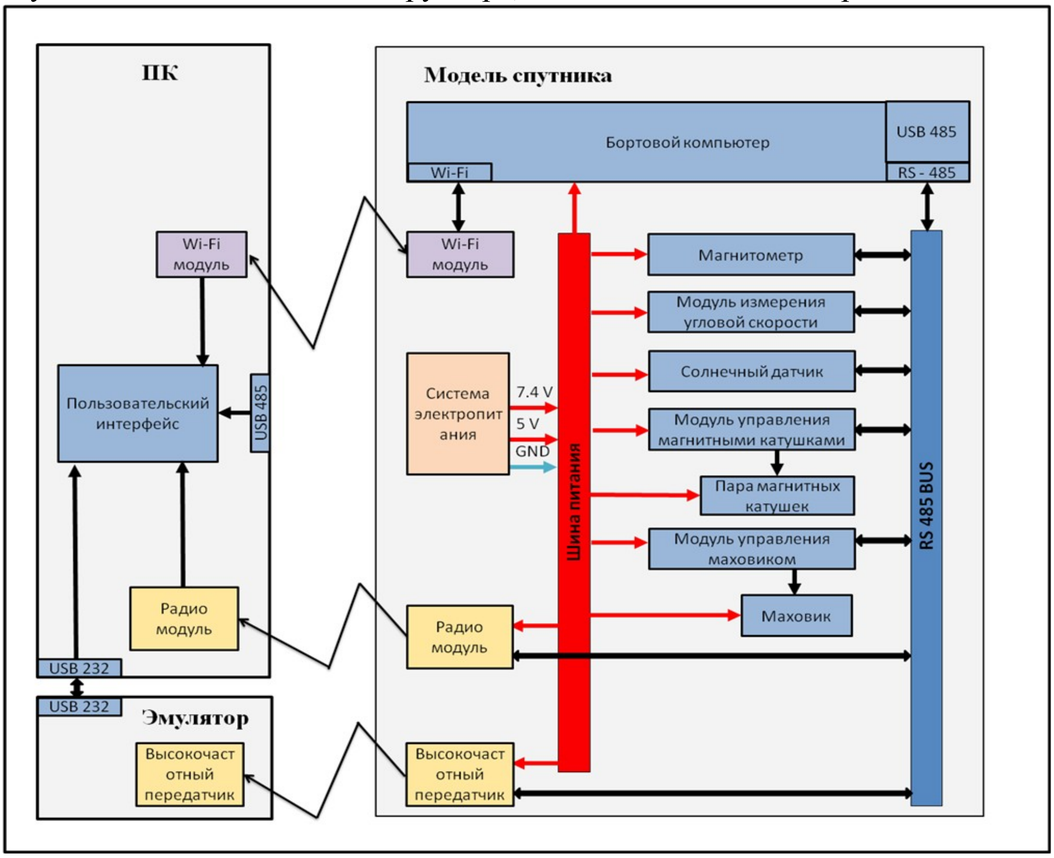
Каждая команда получает получает следующий набор для конструирование и инструменты:

- конструктор «Орбисат», включающий в себя:
  - 4 боковых, 1 верхняя и 1 нижняя крепежных панели;
  - блок управления;
  - система электропитания с аккумуляторами и выносным блоком управления;
  - двигатель-маховик;
  - панель солнечных батарей;
  - солнечный датчик (4 шт.);
  - датчик магнитного поля;
  - датчик угловой скорости;
  - УКВ-передатчик;
  - высокопроизводительный (направленный) передатчик;
  - наземный УКВ-приемник;
  - наземный приемник высокопроизводительной связи.
- крестовая отвертка;
- устройство для обжимания разъемов DB-9F и DB-9M;
- шлейф 9-жильный DS1057-09-30;
- устройство для тестирования шлейфа;
- подвес и настольная лампа;
- соединительный кабель USB;
- ноутбук с установленным программным обеспечением.

Собранная из этого конструктора функциональная модель спутника может совершать управляемый полет, управлять вращением вокруг одной оси, получать информацию от датчиков. Полет программируется на языке Python или C++.

В виде отдельных компонентов и в собранном виде конструктор представлен на следующих рисунках:

Функциональная схема конструктора, подключенного к ПК представлена на схеме:

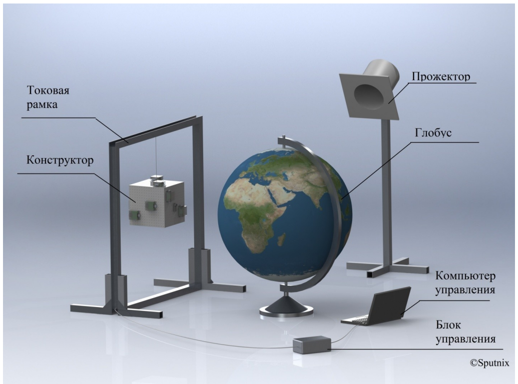


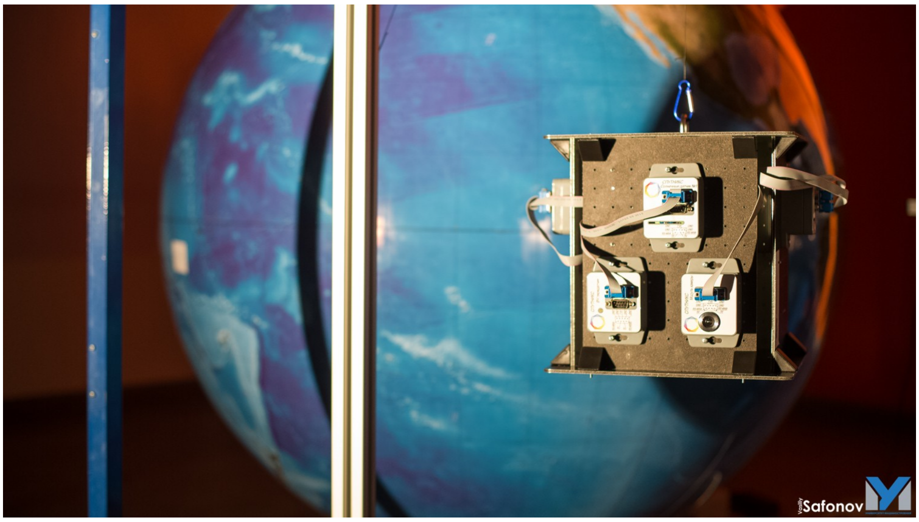
4.3. Описание стенда для испытаний

Стенд для испытаний представляет собой модель движения спутника по низкой экваториальной орбите с учетом освещения и воздействия магнитного поля Земли. Стенд включает в себя:

- вращающуюся модель земного шара со встроенными индикаторами наземных измерительных пунктов (НИП) и пунктами приема высокочастотных передач со спутника;
- магнитную рамку, имитирующую изменение магнитного поля Земли;
- подвес для спутника;
- прожектор;
- ноутбук с программой управления стендом и выводом информации для участников («Центр управления полетом»);
- баннер с регулярной квадратной разметкой.

Принципиальная схема стенда и фотография стенда в работающем состоянии представлены на следующих рисунках:





Стенд устанавливается в затемненной комнате. Перед началом испытаний жюри производит измерения уровней освещенности и естественного магнитного поля Земли.

4.4. Программирование полета аппарата

Программирование полета аппарата осуществляется на языках Python или C++ с использованием специальной библиотеки, включающей основные команды управления подсистемами аппарата.

Библиотека для языка C++: http://wiki.orbicraft.ru/doku.php?id=libschsat

Библиотека на языке Python: http://wiki.orbicraft.ru/doku.php?id=libschsat-py

4.5. Условие финальной задачи

Наиболее сложные задания 10 и 11 должны быть выполнены на описанном стенде. Для выполнения задания 11 каждая команда получает уникальное условие следующего вида.

Вам необходимо сделать снимки следующих точек (в скобках указан её меридиан):

- 1 (165°)
- 5 (55°)
- 9 (-60°)

Одна попытка — это один полный оборот глобуса (период обращения 5 мин).

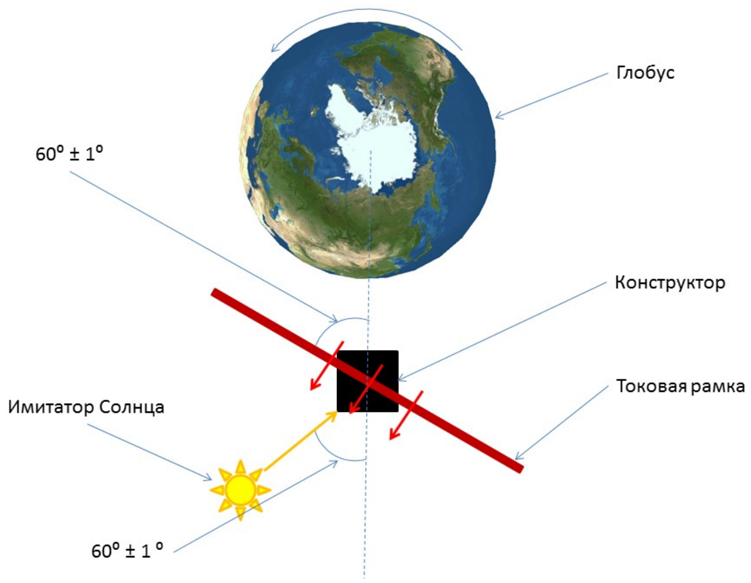
Спутник начинает «движение» с меридиана -100°, и «движется» с запада на восток.

Снимки возможно сбросить на один из трех наземных измерительных пунктов (НИП), которые установлены в следующих координатах (указан меридиан НИПа):

- НИП1 105°
- НИП2 15°
- НИП3 -80°

Все точки для съемки и НИПы расположены на параллели - 5° и спутники участников будут подвешены организаторами так, чтобы ось камеры лежала в плоскости этой параллели.

Вы можете ориентироваться по магнитному полю и Солнечному излучению, взаимная ориентация которых соответствует схеме:



4.6. Что требует решение задачи

Для решения поставленной задачи (начиная с самого простого задания №1 и до самого сложного задания №11) учащиеся должны обладать следующими знаниями и навыками:

- Расчет параметров системы с учетом известных характеристик подсистем и выбранной конструкции, в том числе:
    - расчет потребления электроэнергии;
    - влияния компоновки на момент инерции аппарата;
    - расположение датчиков на корпусе аппарата в связи с требованиями к измерениям.
- Монтаж электрических схем, включая обжимание и тестирование кабелей.
- Последовательное тестирование подсистем в сборке (выполнение заданий 1-7).
- Решение задачи стабилизации аппарата при управлении вращением вокруг одной (вертикальной оси).
- Расчет положения аппарата на орбите. Решение задачи ориентации по положению солнца и по магнитному полю.
- Программирование полета аппарата на языках Python или C++, отладка программы по результатам анализа телеметрии полета.

4.7. Критерии оценки

Оценка работы происходит следующим образом:

1. Участники собирают аппаратное решение (конструктор спутника или его компоненты)

и программируют его поведение на выполнение одного из заданий.

2. Участники передают работу (аппаратно-программное решение) судье, который фиксирует попытку команды.
3. Судья, не раскрывая авторов работы, передает работу жюри на стенд для проведения испытаний.
4. Оценка работы на стенде производится автоматически или членами (в зависимости от задания) в соответствии с заданными алгоритмами проверки, которые указаны далее.
5. Судья доносит итоги до команды вместе с протоколом испытаний, фиксирует баллы команды на табло.

Далее приводятся критерии оценки для всех задач командной части заключительного этапа.

Задача 4.7.1. «Проверка радиоканала»

Критерий проверки: Аппарат передает в телеметрии своего полёта сообщение, которое включает в себя название команды.

Результат фиксируется автоматически в программном комплексе стенда.

Задача 4.7.2 «Проверка магнитометра»

Критерий проверки: Аппарат непрерывно в течение не менее 10 с передает в телеметрии своего полёта показания магнитометра (состояние датчика и показания индукции магнитного поля по трем векторам). Для моделирования магнитного поля используется откалиброванная рамка стенда. Допускается отклонение от эталонного значения магнитного поля на 10%.

Результат фиксируется автоматически в программном комплексе стенда.

Задача 4.7.3 «Проверка датчика угловой скорости»

Критерий проверки: Аппарат непрерывно передает в течение не менее 10 с в телеметрии своего полёта показания датчика угловой скорости (состояние прибора, угловая скорость по трем осям). Перед измерением аппарату задается фиксированная угловая скорость с помощью специальной программы, запускаемой жюри.

Результат фиксируется автоматически в программном комплексе стенда.

Задача 4.7.4 «Проверка солнечных датчиков»

Критерий проверки: Аппарат непрерывно передает в течение не менее 10 с в телеметрии своего полета сырые показания (два числа) всех четырех солнечных датчиков. Жюри проверяет два режима: темная комната и моделирование солнца с помощью прожектора.

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.5 «Проверка камеры и передатчика»

Критерий проверки: Аппарат производит снимок и передает его на наземный измерительный пункт (с использованием наземного модуля высокопроизводительной связи).

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.6 «Сборка аппарата и проверка систем»

Критерий проверки: Все компоненты конструктора прикреплены к корпусу и соединены шлефами. Аппарат прикрепляется к подвесу и висит так, что его вехняя и нижняя грань параллельны полу с отклонением не более 5 градусов от горизонтали. Проверка осуществляется с помощью закрепленной на аппарате лазерной указки и специального баннера с квадратной разметкой, на котором указаны допустимые отклонения. Конструктор после запуска непрерывно в течение не менее 10 с передает показания со всех установленных датчиков в телеметрию.

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.7 «Проверка двигателя-маховика»

Критерий проверки: Аппарат закрепляется на подвесе. Двигатель-маховик включился, аппарат совершил вращения по часовой стрелке в течение 10 секунд (с точностью до 1 сек), после чего совершил вращения против часовой стрелке 10 секунд (с точностью до 1 сек) и выключился.

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.8 «Стабилизировать спутник»

Критерий проверки: Аппарат закрепляется на подвесе. Перед включением аппарату задается фиксированная угловая скорость с помощью специальной программы, запускаемой жюри. Аппарат должен прекратить вращение как можно скорее и продержать его в стабильном положении не менее 5 секунд. Стабильное положение аппарата определяется с помощью лазерной указки, прикрепленной к корпусу, и баннера с разметкой. Программный комплекс стенда измеряет время, за которое произведена стабилизация аппарата.

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.9 «Ориентация спутника»

Критерий проверки: Аппарат закрепляется на подвесе. Перед включением аппарату задается фиксированная угловая скорость с помощью специальной программы, запускаемой жюри. Аппарат должен стабилизироваться и изменить угловое положение так, чтобы солнечные батареи были направлены в сторону имитатора Солнца, после остановки вращения аппарата допускается отклонение не более чем на 10 градусов в течение 5 секунд. Ориентация аппарата определяется с помощью лазерной указки, прикрепленной к корпусу, и баннера с разметкой. Программный комплекс стенда измеряет время, за которое произведена ориентация аппарата.

Результат фиксируется членами жюри в соответствии с описанным критерием.

Задача 4.7.10 «Составление отчета о магнитном поле Земли»

Критерий проверки: Аппарат закрепляется на подвесе. Стенд производит вращение глобуса в течение 5 мин, одновременно с этим магнитная рамка меняет создаваемое искусственно магнитное поле по заданной ранее функции. Аппарат измеряет изменение модуля вектора магнитного поля в зависимости от градуса долгота. Результаты измерений автоматически сравниваются с эталонной функцией моделируемого магнитного поля, создаваемого откалиброванной рамкой. Численно рассчитывается функция среднего линейного отклонения, результат которой определяет число получаемых за задачу баллов.

Результат фиксируется автоматически в программном комплексе стенда.

Задача 4.7.11 «Съемка точки на поверхности Земли и передача снимка на НИП»

Критерий проверки: Аппарат закрепляется на подвесе. Перед включением аппарату задается фиксированная угловая скорость с помощью специальной программы, запускаемой жюри. Стенд производит вращение глобуса в течение 5 мин, одновременно с этим магнитная рамка меняет создаваемое искусственно магнитное поле по заданной ранее функции. Аппарат должен сфотографировать заданную точку на поверхности, координаты которой выдаются команде в задании. Маркер цели должен полностью попасть на снимок. Снимок должен быть передан на НИМ по высокопроизводительному каналу связи.

Результат фиксируется автоматически в программном комплексе стенда.

РЕШЕНИЕ:

Пример программы на языке С++, обеспечивающей решение последнего задания

(№11):

```
#include <stdio.h>
#include <stdint.h>
#include "libschsat.h"
#include "math.h"
#include <time.h>

#define MAX_WRATE 3000
#define SUNSENSORS_COUNT 4
#define Z_WRATE_INDEX 0
#define MINIMAL_SUN 200
#define REQUESTED_SIDE 1
#define DELTA_STOPPING 500
#define KwStab 0.5
#define STAB_DELAY 20000

uint16_t sun[SUNSENSORS_COUNT][2];
int16_t hyro[3];
int16_t speed;
int16_t speed_confirm;
int ret;
int sun_index = 0;
double confirmation = 0;
double start_time;
struct timeval tv;
int passed[100];


double ms_from_start (void)
{
 gettimeofday(&tv, NULL);
 return(tv.tv_sec*1000+tv.tv_usec/1000-start_time);
}


void stabilize_speed(int16_t value)
{
 int16_t delta;
 delta = (value - hyro[Z_WRATE_INDEX])*KwStab;
 if (delta > DELTA_STOPPING)
       delta = DELTA_STOPPING;

 motor_set_speed(1,speed+delta,&speed_confirm);
}
void slow_down(void)
{
 if (hyro[Z_WRATE_INDEX]>0)
 {
       motor_set_speed (1, speed-DELTA_SOPPING,&speed_confirm);
 }
 else
 {

      motor_set_speed (1, speed+DELTA_STOPPING,&speed_confirm);
 }
}

void stabilize_angle (void)
{
 double Kw = -0.1;
```

```
 double Kprop = -0.01;
 double delta  = 0;
 delta += Kw * hyro[Z_WRATE_INDEX];
 delta += Kprop * (sun[sun_index-1][0]-sun[sun_index-1][1]);
 motor_set_speed (1, speed+delta,&speed_confirm);

}

void control (void)
{
 int i;
 int j;
 printf("Entering main loop!\n");
 gettimeofday(&tv, NULL);
 start_time = tv.tv_sec*1000+tv.tv_usec/1000;

 hyro_turn_on(1);
 for (i=0; i<SUNSENSORS_COUNT; i++)
 {
       sun_sensor_turn_on(i+1);
 }
 motor_turn_on(1);

 for (i=0; i<100; i++)
       passed[i]=0;


 mSleep(1000);


 while (1)
 {
       ret = hyro_request_raw(1, &(hyro[0]), &(hyro[1]), &(hyro[2]));
       if (ret != LSS_OK)
             printf("Hyro fail\n");
       for (i=0; i<SUNSENSORS_COUNT; i++)
       {
             ret |= sun_sensor_request_raw(i+1, &(sun[i][0]), &(sun[i][1]));
             if (ret != LSS_OK)
                   printf("Sun %d fail\n",i+1);
       }
       ret |= motor_request_speed(1, &speed);
       if (ret != LSS_OK)
             printf("Motor fail\n");
       if (ret == LSS_OK)
       {
             //Let's find the most sunny side
             int16_t MaxRAW = 0;
             for (i=0; i<SUNSENSORS_COUNT; i++)
                   for ( j = 0; j<2; j++)
                         if    (sun[i][j] > MaxRAW)
                         {
                               MaxRAW = sun[i][j];
                               sun_index = i+1;
                         }

             if (sun_index == 0) //really dark
                   continue;

             if (abs(hyro[Z_WRATE_INDEX])>MAX_WRATE)
```

```
{
	//too fast, just need to stop rotating
	slow_down();
}
else if ((sun[sun_index-1][0] < MINIMAL_SUN) && (sun[sun_index-1][1] < MINIMAL_SUN))
{
	//Ok even brightest spot is too dark, do nothing
	printf ("    Unknown position\n");
}
else
{
	if (sun_index == REQUESTED_SIDE)
	{
		if ((sun[REQUESTED_SIDE-1][0] > MINIMAL_SUN) && (sun[REQUESTED_SIDE-1][1] > MINIMAL_SUN))
		{
		//ok now we could stabilize angle, not speed

			confirmation = fabs(sun[REQUESTED_SIDE-1][0]/sun[REQUESTED_SIDE-1][1]);
			//printf ("Confirmation = %d\n",sun[REQUESTED_SIDE-1][0]-sun[REQUESTED_SIDE-1][1] );
			stabilize_angle();
			if (fabs(confirmation-1)<0.005)
			{
				printf ("Pointed!\n");
			}
		}
		else
		{
		//we have to try really hard to stay here
			if (sun[REQUESTED_SIDE-1][0]>(sun[REQUESTED_SIDE-1][1]))
				stabilize_speed(-1000);
			else
				stabilize_speed(1000);
		}
	}
	else
	{
	//wrong sensor at all, let's keep rotating
		if (sun_index == 2)
			stabilize_speed(2000);
		else if (sun_index == 4)
			stabilize_speed(-2000);
		else
			printf("Test point 3\n");
	}
}
//stabilization finished, need to work

double t;
t = ms_from_start();
if ((t > STAB_DELAY) && (t < STAB_DELAY+1000) && (passed[0]==0))
{
	passed[0]=1;
	printf ("Stage 1 all fine!\n");
}
else if ((t > STAB_DELAY+10000) && (t < STAB_DELAY+11000) &&
```

```
(passed[1]==0))
            {
                passed[1]=1;
                if (confirmation < 0.005)
                    printf ("Stage 2, taking photo, stable\n");
                else
                    printf ("Stage 2, taking photo, unstable\n");
                camera_turn_on();
                camera_take_photo(1);
            }
            else if ((t > STAB_DELAY+14000) && (t < STAB_DELAY+15000) && (passed[2]==0))
            {
                passed[2]=1;
                transmitter_turn_on(1);
                printf("Stage 3, powering HF transmitter\n");
            }
            else if ((t > STAB_DELAY+16000) && (t < STAB_DELAY+17000) && (passed[3]==0))
            {
                passed[3]=1;
                if (confirmation < 0.01)
                    printf ("Stage 4, transmitting photo, to earth\n");
                else
                    printf ("Stage 4, transmitting photo, to space\n");
                transmitter_transmit_photo(1,1);
            }

            else if ((t > STAB_DELAY+16500) && (t < STAB_DELAY+17500) && (passed[4]==0))
            {
                passed[4]=1;
                transmitter_turn_on(1);
                printf("Stage 4+, switching off HF transmitter\n");
                transmitter_turn_off(1);
                camera_turn_off();
            }


            mSleep(100);
        }
        else
            printf("Failed\n");


    }


}
```

Пример программы на языке Python, обеспечивающей решение последнего задания (№11):

```
import numpy as np
LSS_OK = 0
LSS_ERROR = 1
import time as tm
import numpy as numpy

MAX_WRATE = 3000
```

```
SUNSENSORS_COUNT = 4
Z_WRATE_INDEX = 0
MINIMAL_SUN = 200
REQUESTED_SIDE = 1
DELTA_STOPPING = 500
KwStab = 0.5
STAB_DELAY = 20000

sun = np.ndarray((3,4), dtype=int)
hyro = np.ndarray((1,3), dtype=int)
passed = np.ndarray((1,3), dtype=int)
sun_index = 0
confirmation = 0
INIT_TIME = tm.time()

def ms_from_start ():
    return tm.time() - INIT_TIME

def stabilize_speed (value):
    delta = (value - hyro[Z_WRATE_INDEX])*KwStab
    if delta > DELTA_STOPPING:
        delta = DELTA_STOPPING

    speed_confirm = motor_set_speed(1,speed+delta)
    return speed_confirm

def slow_down ():
    if hyro[Z_WRATE_INDEX] > 0: speed_confirm = motor_set_speed(1, DELTA_SOPPING)
    else: speed_confirm = motor_set_speed(1, DELTA_STOPPING)
    return speed_confirm

def stabilize_angle ():
    Kw = -0.1
    Kprop = -0.01
    delta = 0
    delta += Kw*hyro[Z_WRATE_INDEX]
    delta += Kprop*(sun[sun_index,0] - sun[sun_index-1,1])
    speed_confirm = motor_set_speed (1, speed+delta)
    return speed_confirm

def control ():
    INIT_TIME = tm.time()
    print ("Entering main loop!\n")

    hyro_turn_on(1)
    for i in range(SUNSENSORS_COUNT): sun_sensor_turn_on(i+1)
    motor_turn_on(1)

    for i in range(100): passed[i] = 0

    tm.sleep(1)

    while 1:
        ret_hyro = hyro_request_raw(1)
        ret = ret_hyro[0]
        if ret != LSS_OK: print("Hyro fail!\n")
        for i in range(SUNSENSORS_COUNT):
            sun[i] = sun_sensor_request_raw(i+1)
            ret |= sun[i][0]
            if ret != LSS_OK: print("sun %d fail\n", (i+1))
```

```
        ret_m, speed = motor_request_speed(1)
        ret |= ret_m
        if ret != LSS_OK: print("Motr fail!\n")
        if ret == LSS_OK:
            # Let's find most sunny side!
            MaxRAW = 0
            for i in range(SUNSENSORS_COUNT):
                for j in range(2):
                    if sun[i,j] > MaxRAW:
                        MaxRAW = sun[i,j]
                        sun_index = i+1
                    if sun_index == 0: continue
                    if abs(hyro[Z_WRATE_INDEX]) > MAX_WRATE: slow_down()
                    elif sun[sun_index-1,0] < MINIMAL_SUN and sun[sun_index-1,1] < MINIMAL_SUN: print ("Unknown position!\n")
                    else:
                        if sun_index == REQUESTED_SIDE:
                            if sun[REQUESTED_SIDE-1,0] > MINIMAL_SUN and sun[REQUESTED_SIDE-1,1] > MINIMAL_SUN:
                                confirmation = abs(sun[REQUESTED_SIDE-1,0]/sun[REQUESTED_SIDE-1,1])
                                stabilize_angle()
                                if abs(confirmation-1) < 0.005: print ("Pointed!\n!")
                            else:
                                #try hard to stay there
                                if sun[REQUESTED_SIDE-1,0] > sun[REQUESTED_SIDE-1,1]: stabilize_speed(-1000)
                                else: stabilize_speed(1000)
                        else:
                            if sun_index == 2: stabilize_speed(2000)
                            elif sun_index == 4: stabilize_speed(-2000)
                            else: print ("Test point 3\n")
                    # stabilization finished
                    t = ms_from_start()
                    if t > STAB_DELAY and t < STAB_DELAY+1000 and passed[0]==0:
                        passed[0]=1
                        print ("Stage 1 all fine!\n")
                    elif t > STAB_DELAY+10000 and t < STAB_DELAY+11000 and passed[1]==0:
                        passed[1]=1
                        if confirmation < 0.005: print ("Stage 2. taking photo, stable\n")
                        else: print ("Stage 2, taking photo, unstable\n")
                        camera_turn_on()
                        sleep(0.5)
                        camera_take_photo(1)
                    elif t > STAB_DELAY+14000 and t < STAB_DELAY+15000 and passed[2]==0:
                        passed[2] = 1
                        transmitter_turn_on(1)
                        print ("Stage 3, powering HF transmitter\n")
                    elif t > STAB_DELAY+16000 and t < STAB_DELAY+17000 and passed[3]==0:
                        passed[3]=1
                        if confirmation < 0.01: print ("Stage 4, transmitter photo to earth\n")
                        else: print ("Stage 4, transmitter photo to space\n")
                        transmitter_transmit_photo(1,1)
```

```
elif t > STAB_DELAY+16500 and t < STAB_DELAY+17500 and passed[4]==0:
    passed[4]=1
    transmitter_turn_on(1)
    print ("Stage 4+, switching off HF transmitter\n")
    transmitter_turn_off(1)
    camera_turn_off()
    sleep(10)
else: print ("Failed\n")
```