§4 Заключительный этап: командная часть

Задание
День первый 24.02.2018

2035 год. Вы запускаете миссию по исследованию экзопланеты «Терра-2035» открытой вами в рамках проектной смены в далёком 2018. Тогда вы обнаружили, что она очень напоминает нашу Землю. Вы решили собрать аппарат, который будет собирать информацию о планете и передавать её вам.

Сборка подобных аппаратов трудоёмкий процесс. Вы выбираете путь собрать его как конструктор, используя максимальное число модулей которые вам могут подойти. В вашем распоряжении есть набор деталей и ряд дополнительных модулей, которые могут помочь вам сократить время сборки и отладки аппарата до 3 дней. Итак начнем..

Задача 1 Входной контроль

Первой задачей является проверка функциональности базового набора.

Соберите его максимально аккуратно. Нужно продумать компоновку модулей так чтобы она не противоречила их функциональности, обеспечивая минимизацию момента инерции для лучшей управляемости.

После проработки концепции и сборки, вам необходимо отладить работу Системы Электро Питания (СЭП) бортового вычислительного модуля (БКУ) системы передачи тедеметрии (УКВ) датчиков угловой скорости, магнитного поля, направления на ближайщую звезду (солнечных датчиков) и вспомогательных модулей.

Вы должны собрать спутник, руководствуясь wiki на сайте http://orbicraft.sputnix.ru/ и проверить функциональность его компонентов, проведя последовательную их проверку.

Ваша задача продемонстрировать функциональность модулей, путем запуска тестовых программ. После этого вы собираете тестовые программы в единый код и по выбору пользователя из web интерфейса демонстрируете функциональность модулей.

Входной контроль должен показать что все детали набора «Орбикрафт» корректно работают по отдельности и в сборе. Для этого вы должны написать программу которая по запросу из сервисной консоли веб-интерфейса опрашивает модуль и выдает его текущий статус (в работе он или нет) и если он в работе то его основные характерные параметры.

Задача 2 Алгоритм стабилизации

После того, как вы убедитесь что аппарат собран и работоспособен, необходимо написать и запустить алгоритм остановки вращения спутника. Аппарат оказавшись вблизи планеты может иметь начальное вращение. Для того, чтобы выполнить вашу миссию необходимо это вращение остановить. Вы должны написать алгоритм, который сделает это максимально быстро и плавно, по возможности избежав рыскания. Вам необходимо использовать датчик угловой скорости и маховик.

Задача 3 Канал связи

Параллельно вы можете вести работы по наладке канала связи в оптическом диапазоне. Ваша задача передать битовый массив длиной до 200 элементов с одного компьютера на другой. Массив принимается через USB-UART от одного компьютера и бесконтактно передается на другой компьютер по средствам USB-UART. Передачу данных по ик нужно осуществить используя имеющееся у вас оборудование

Для помощи в решении вы можете использовать методичку с заданиями прошлого года. Она находится на сайте http://nti-contest.ru/profiles/space/

Задача 4 «Звездный датчик»

Для точной ориентации спутника используются звездные датчики. Вам предлагается создать подобный им модуль. Данная вам камера может различать яркие объекты на темном фоне и выдавать их координаты. Вам нужно запрограммировать камеру на слежение за объектами (светодиодами) и продемонстрировать результат в консоли USB-UART.

После того, как вы увидите координаты отслеживаемых объектов вам предстоит создать рисунок из светодиодов так, чтобы по нему можно было как можно более точно определить текущий угол поворота спутника. Изучив возможности камеры подумайте над тем, какой рисунок наиболее оптимально подходит для поставленной задачи.

http://charmedlabs.com/default/pixy-cmucam5/

Таблица 1. Критерии первого дня.

Тест систем аппарата		
Работа консоли веб-интерфейса	1	
Работа датчика угловых скоростей	1	
Работа солнечных датчиков	1	
Работа маховика	1	
Программа проверки всех перечисленных выше систем написана одним кодом, можно выбирать прибор для проверки, вводя команду через консоль	2	
Сборка и балансировка		
Спутник не перекошен более чем на 3 градуса	2	
Правильно установлено оборудование	2	
Момент инерции спутника минимизирован	1	
Аккуратность сборки	1	
Стабилизация		
Спутник держит стабилизацию не менее 20 секунд	10	
Стабилизированный спутник компенсирует внешнее возмущение	10	
Перед достижение стабильного состояния не меняется направления вращения	10	
Оптический канал		
Реализована передача бинарного массива с ардуино на ардуино по ИК-каналу	3	
Звездный датчик		
Камера подключена к ардуино. Система получает координаты тестовых объектов, имитирующих	5	

звезды		
ИТОГО	50 баллов	

Решение заданий первого дня

Задача 1 Входной контроль

Программа, которая позволяет проверить работоспособность всех перечисленных устройств:

```
#include <stdio.h>
#include "libschsat.h"
void control(void)
{
	puts("Start! Press any key");
	getc();  /* magnetometer test */
	int i;
	const int num = 1;  /* magnetometer #1 */
	printf("Enable magnetometer #%d\n", num);
	magnetometer_turn_on(num);
	printf("Get RAW data from magnetometer #%d\n", num);
	for (i = 0; i < 10; i++) {
		int16_t x, y, z;
		if (LSS_OK == magnetometer_request_raw(num, &x, &y, &z)) {
			printf("%d: x=%d y=%d z=%d\n", i, x, y, z);
		} else {
			puts("Fail!");
		}
		Sleep(1);
	}
	printf("Disable magnetometer #%d\n", num);
	magnetometer_turn_off(num);
	puts("Press any key");

	getc(); /* Hyro test */
	printf("Enable hyro #%d\n", num);
	hyro_turn_on(num);
	printf("Get RAW data from hyro #%d\n", num);
	for (i = 0; i < 10; i++) {
		int16_t x, y, z;
		if (LSS_OK == hyro_request_raw(num, &x, &y, &z)) {
			printf("%d: x=%d y=%d z=%d\n", i, x, y, z);
		} else {
			puts("Fail!");
		}
		Sleep(1);
	}
	printf("Disable hyro #%d\n", num);
	hyro_turn_off(num);
	puts("Press any key");
```

```
getc(); /* Sun sensor test */
printf("Enable sensor #%d\n", num);
sun_sensor_turn_on(num);
printf("Get RAW data from sun sensor #%d\n", num);
for (i = 0; i < 10; i++) {
        uint16_t value1;
        uint16_t value2;
        if (LSS_OK == sun_sensor_request_raw(num, &value1, &value2)) {
                printf("%d: raw=%d ; %d\n", i, value1, value2);
        } else {
                puts("Fail!");
        }
        Sleep(1);
}
printf("Disable sensor #%d\n", num);
sun_sensor_turn_off(num);
puts("Press any key");
getc(); /* motor test */
int16_t temp;
int16_t rpm = -3000;  /* -3000 ... +3000 */
printf("Enable motor #%d\n", num);
motor_turn_on(num);
printf("Manage speed motor #%d\n", num);
while (rpm <= 3000) {
        printf("<<< Set speed to %d\n", rpm);
        if (LSS_OK == motor_set_speed(num, rpm, &temp)) {
                if (temp == rpm)
                        printf("\t%d confirmed\n", rpm);
        }
        Sleep(1);
        if (LSS_OK == motor_request_speed(num, &temp)) {
                printf("Got speed %d >>>\n", temp);
        } else {
                puts("Fail! >>>");
        }
        rpm += 500;
}
printf("<<< Set speed to 0\n");
if (LSS_OK == motor_set_speed(num, 0, &temp)) {
        if (temp == 0)
                printf("\t%d confirmed\n", 0);
}
Sleep(1);
if (LSS_OK == motor_request_speed(num, &temp)) {
        printf("Got speed %d >>>\n", temp);
} else {
        puts("Fail! >>>");
}
Sleep(1);
motor_set_speed(num, 0, &temp);
Sleep(1);
printf("Disable motor #%d\n", num);
```

```
	motor_turn_off(num);
	puts("All tests done!");
	getc();
}
```

Задача 2 Алгоритм стабилизации

Код, отвечающий за стабилизацию конструктора спутника:

```
#include "libschsat.h"
const int Motor_num = 1;  /* motor #1 */
const int Hyro_num = 1;   /* Hyro sensor #1 */
const int SpeedCoef = 360  /* Hyro sensor coefficient */
void control(void)
{
	bussetup();
	mSleep(5000);
	motor_turn_on(Motor_num);
	mSleep(5000);
	hyro_turn_on(Hyro_num);
	mSleep(5000);
	int16_t set_speed = 0;
	int16_t MotorAns = 0;
	int16_t x, y, z;
	int i = 0;
	for (i = 0; i < 100; i++) {
		hyro_request_raw(num, &x, &y, &z));
		if (abs(z) < 1000) {
			set_speed = -z*SpeedCoef;
		}
		motor_set_speed(Motor_num, 0, &MotorAns);
		while (abs(set_speed - MotorAns) > 10){
			motor_request_speed (Motor_num, MotorAns);
			mSleep (200);
		}
		mSleep(5000);
	}
motor_turn_off(Motor_num);
hyro_turn_off(Hyro_num);
}
```

Задача 3 Канал связи

Код отвечающий за передачу для платы arduino mega, ик передатчик подключен к 11 порту платы:

```
#include <stdint.h>
const uint8_t DIODE_PIN = 11;
const unsigned long UART_BAUD_RATE = 600;
const unsigned long UART_DELAY_US = 1000000 / UART_BAUD_RATE;
const unsigned long CARRIER_FREQUENCY = 38000;
```

```
const unsigned long CARRIER_DELAY_CYCLES = F_CPU / (CARRIER_FREQUENCY * 2);

static void initPWM()
{
	TCCR1A = 0;
	TCCR1B = 0;
	TCCR1C = 0;
	TIMSK1 = 0;
}

static void startPWM()
{
	TCCR1B = 0;
	TCCR1A = 0;
	TCNT1 = 0;
	OCR1A = CARRIER_DELAY_CYCLES - 1;
	TCCR1A = (0 << COM1A1) | (1 << COM1A0) | (1 << WGM11) | (1 << WGM10);
	TCCR1B = (1 << WGM13) | (1 << WGM12) | (0 << CS12) | (0 << CS11) | (1 << CS10);
}

static void stopPWM()
{
	TCCR1B = 0;
	TCCR1A = 0;
}

static void sendZero()
{
	startPWM();
	delayMicroseconds(UART_DELAY_US);
}

static void sendOne()
{
	stopPWM();
	delayMicroseconds(UART_DELAY_US);
}
void setup()
{
	pinMode(DIODE_PIN, OUTPUT);
	digitalWrite(DIODE_PIN, LOW);
	initPWM();
}

void sendByte(uint8_t byte)
{
	sendZero();
	for (int i = 0; i < 8; i++) {
		Serial.println(byte & 1);
		if ((byte & 1) != 0) {
			sendOne();
		} else {
```

```
                sendZero();
            }
            byte >>= 1;
        }
        sendOne();
}

void loop()
{
        for (int i = 0; i < 256; i++) {
            sendByte(i);
            delay(100);
        }
}
```

Код отвечающий за прием для платы arduino leonardo, ик передатчик подключен к 1 порту платы:

```
#include <stdint.h>
void setup()
{
        Serial.begin(115200);
        Serial1.begin(600);
}

void loop()
{
        if (Serial1.available() > 0) {
            uint8_t received_byte = Serial1.read();
            Serial.println(String(received_byte));
        }
}
```

Задача 4 «Звездный датчик»

```
#include <SPI.h>
#include <Pixy.h>
Pixy StarSensor;

void setup()
{
        Serial.begin(115200);
        Serial.print("Start\n");
        StarSensor.init();
}

void loop()
{
        static int i = 0;
        int j;
        uint16_t blocks;
```

```
        char buf[32];

        blocks = StarSensor.getBlocks();
        if (blocks)
        {
                i++;
                if (i%20==0)
                {
                        sprintf(buf, "Detected %d:\n", blocks);
                        Serial.print(buf);
                        for (j=0; j<blocks; j++)
                        {
                                sprintf(buf, "  block %d: ", j);
                                Serial.print(buf);
                                StarSensor.blocks[j].print();
                        }
                }
        }
}
```

Задание
День второй 25.02.2018

Вы собрали спутник, но он до сих пор работает нестабильно. Вас это беспокоит, пришло время отладки алгоритмов…

Несмотря на то, что критерии оценки первого дня сгорают, новые критерии учитывают накопленный вами опыт и наработки прошлых дней.

Код всех заданий предполагающих программирование необходимо в конце дня сохранять на сервер, иначе комиссия может обнулить выставленный за задание бал. Сегодня коды необходимо сохранять в папку соответствующую второму дню.

Задача 1 Остановка вращения аппарата

Чтобы выполнить миссию аппарат должен остановить своё вращение. К сожалению вчера это не удалось, но не теряем надежду.

Задача 2 Ориентация

Не менее важной задачей является ориентация аппарата. Мы предлагаем вам попробовать откалибровать модуль магнитометра и научиться ориентировать аппарат.

Задача 3 Калибровка «звездного датчика»

Ваш аппарат может видеть звезды, но пока нет оборудования которое поможет извлечь информацию из их положения. Ваша задача – обучить камеру распознавать звезды и сориентироваться по ним

Задача 4 Настройка оптического канала связи

Оптический канал связи поможет вам передать изображения с КА на Землю. Давайте попробуем разобраться в наработках прошлых лет, и сделаем передачу сигнала стабильной

Таблица 2.1. Критерии второго дня.

Критерий	Максимальный балл	Оценка
Стабилизация		
Спутник держит стабилизацию не менее 20 секунд	5	
Стабилизированный спутник компенсирует внешнее возмущение	5	
Перед достижением стабильного состояния не меняется направление вращения	5	
Проверка: залить на БКУ программу с именем **stabilization.zip**		
Ориентация		
Калибровка модуля «магнитометр», см. таблицу №2.2	10	
Ориентация аппарата по магнитометру: разворот изначально неподвижно висящего спутника на 180 градусов и стабилизация его в этом положении ***Проверка:*** залить на БКУ программу с именем **magnetometer_orientation.zip** загрузить программу в папку на сервере «Калибровка магнитометра»	5	
Калибровка модуля «звёздный датчик»		
Система получает координаты тестовых объектов, имитирующих звезды ***Проверка:*** быть готовым предоставить результаты в любой момент после 17.00	2.5	
Сконфигурировать датчик таким образом, чтобы он выдавал текущую ориентацию КА ***Проверка:*** создать папку на сервере с названием «Звездный датчик», положить в него файл **orientation.ino** Оставить на столе Arduino, прошитую этим файлом (подписать на бумажке)	5	
Настройка оптического канала связи		
Реализована передача бинарного массива с ардуино на ардуино по ИК-каналу ***Проверка:*** быть готовым предоставить результаты в любой момент после 17.00 В папку на сервере с названием «Звездный датчик», положить файлы **ir_transmitter.ino и ir_receiver.ino** Оставить на столе две Arduino, прошитые этими файлами (подписать на бумажке)	2.5	

Переданные данные корректны при кратковременных прерываниях сигнала (не более 10% от общего времени передачи) ***Проверка:*** В папку на сервере «Звездный датчик», положить файлы **ir_transmitter_reliable.ino и ir_receiver_reliable.ino**	10	
ИТОГО	50	

Таблица 2.2. Критерии второго дня.

Критерий	**Максимальный балл**	**Оценка**
Тест магнитометра, магнитометр выдает показания ***Проверка:*** залить на БКУ программу проверки с именем **magnetometer_test.zip** Создать папку на сервере «Калибровка магнитометра», положить в нее файл **magnetometer_test.c** или **magnetometer_test.py**	1	
Магнитометр опрашивается 1000 раз для сбора облака точек (интервал времени =100 мс) ***Проверка:*** залить на БКУ программу с именем **magnetometer_1000_dots.zip** Файл **magnetometer_1000_dots.c** или **magnetometer_1000_dots.py** в папку на сервере «Калибровка магнитометра»	2	
Облако собранных точек визуализировано в любой удобной программе (написать свою или использовать готовую) ***Проверка:*** в папку на сервере «Калибровка магнитометра» положить файл с данными для калибровки “1000_dots”, файл с построением облака точек “data_collected”, картинку (скрин), отображающую облако точек	2	
Получена матрица трансформации в программе magneto, с ее использованием построена сфера полученных точек аналогично предыдущему пункту. ***Проверка:*** в папке «Калибровка магнитометра» должен лежать файл построения облака откалиброванных точек “data_calibrated” и картинка (скрин), отображающая сферу точек	1	
Реализована программа вычисления угла ориентации по показаниям магнитометра ***Проверка:*** залить на БКУ программу с именем **magnetometer_angle.zip** Файл **magnetometer_angle.c** или **magnetometer_angle.py** в папку на сервере «Калибровка магнитометра»	3	

Выполнено определение нулевого положения магнитометра, определена погрешность измерения ***Проверка:*** на столе перед уходом оставить распечатанный транспортир с правильным расположением координат. На том же листе с транспортиром написать погрешность измерения	1	
ИТОГО	10	

Решение заданий второго дня

Задача 1 Остановка вращения аппарата

```
include "libschsat.h"
const int Motor_num = 1;  /* motor #1 */
const int Hyro_num = 1;   /* Hyro sensor #1 */
const int SpeedCoef = 0.1  /* Hyro sensor coefficient */

void control(void)
{
	bussetup();
	mSleep(5000);
	motor_turn_on(Motor_num);
	mSleep(5000);
	hyro_turn_on(Hyro_num);
	mSleep(5000);

	int16_t set_speed = 0;
	int16_t MotorAns = 0;
	int16_t x, y, z;
	int i = 0;

	for (i = 0; i < 100; i++) {
		hyro_request_raw(num, &x, &y, &z));
		if (abs(z) < 1000) {
			set_speed = set_speed - z*SpeedCoef;
		}
		motor_set_speed(Motor_num, 0, &MotorAns);
		while (abs(set_speed - MotorAns) > 10){
			motor_request_speed (Motor_num, MotorAns);
			mSleep (200);
		}
		mSleep(5000);
	}
	motor_turn_off(Motor_num);
	hyro_turn_off(Hyro_num);
}
```

Код для получения данных с магнитометра:

```
#include "libschsat.h"

void control(void)
{
	int i;
	const int num = 1;	/* magnetometer #1 */
	printf("Enable magnetometer #%d\n", num);
	magnetometer_turn_on(num);
	printf("Get RAW data from magnetometer #%d\n", num);
	for (i = 0; i < 1000; i++) {
		int16_t x, y, z;
		if (LSS_OK == magnetometer_request_raw(num, &x, &y, &z)) {
			printf("%d: x=%d y=%d z=%d\n", i, x, y, z);
		} else {
			puts("Fail!");
		}
		mSleep(100);
	}
	printf("Disable magnetometer #%d\n", num);
	magnetometer_turn_off(num);
}
```

Код для калибровки магнитометра:

```
#include "libschsat.h"
#include "math.h"

float mag_angle(float x, float y)
{
	float alpha = 0;
	if (x>0){
		alpha = atan(y/x);
	} else if ((x<0)&&(y>0)) {
		alpha = atan(y/x)+M_PI;
	} else if ((x<0)&&(y<0)) {
		alpha = atan(y/x)-M_PI;
	} else if ((x=0)&&(y>0)) {
		alpha = M_PI_2;
	} else if ((x=0)&&(y<0)) {
		alpha = - M_PI_2;
	}
	return(alpha);
}

void mag_calibration (float *x, float *y, float *z)
{
	float Ainv[3][3] = {{ 1.199705, -0.046626, 0.002081}, { -0.046626, 11.588773, -0.452128 }, {
```

```
0.002081, -0.452128, 1.218523 }};
        float b[3] = { 35.845771, -10916.825690, 208.109724 };
        float xin, yin, zin;
        xin = *x;
        yin = *y;
        zin = *z;
        *x = (Ainv[0][0])*(xin-b[0])+(Ainv[0][1])*(yin-b[1])+(Ainv[0][2])*(zin-b[2]);
        *y = (Ainv[1][0])*(xin-b[0])+(Ainv[1][1])*(yin-b[1])+(Ainv[1][2])*(zin-b[2]);
        *z = (Ainv[2][0])*(xin-b[0])+(Ainv[2][1])*(yin-b[1])+(Ainv[2][2])*(zin-b[2]);
}

/*
** Lab 2: get a raw data from a magnetometer
*/
void control(void)
{
        int i;
        const int num = 1;      /* magnetometer #1 */
        printf("Enable magnetometer #%d\n", num);
        magnetometer_turn_on(num);
        printf("Get RAW data from magnetometer #%d\n", num);
        for (i = 0; i < 100; i++) {
                int16_t x, y, z;
                float alpha;
                if (LSS_OK == magnetometer_request_raw(num, &x, &y, &z)) {
                        //printf("%d: x=%d y=%d z=%d\n", i, x, y, z);
                        float xf, yf, zf;
                        xf = x;
                        yf = y;
                        zf = z;
                        mag_calibration(&xf, &yf, &zf);
                        //printf("                              after calibtation: x=%f y=%f z=%f\n", xf, yf, zf);
                        alpha = mag_angle(xf,zf);
                        alpha = alpha/M_PI*180;
                        printf("                              angle: alpha=%f\n", alpha);
                } else {
                        puts("Fail to request data from magnetometer!");
                }
                mSleep(100);
        }
        printf("Disable magnetometer #%d\n", num);
        magnetometer_turn_off(num);
}
```

Код ориентации аппарата с помощью магнитометра:

```
import math

kd = 200
kp = -100
time_step = 0.2
```

```
mtr_num = 1
hyr_num = 1
mag_num = 1

def mag_calibrated(magx,magy,magz):
        magx_cal = 1.04*magx - 0.26*magy + 0.05*magz - 68.76
        magy_cal = 0.24*magx + 1.04*magy + 0.29*magz + 256.92
        magz_cal = -0.09*magx - 0.19*magy + 0.77*magz + 159.41
        return magx_cal, magy_cal, magz_cal

def angle_transformation(alpha, alpha_goal):
        if alpha<(alpha_goal - 180):
                alpha = alpha + 360
        elif alpha>(alpha_goal +180):
                alpha = alpha - 360
        return alpha

def motor_new_speed_PD(mtr_speed, alpha, alpha_goal, omega, omega_goal):
        mtr_new_speed = int(mtr_speed + kp*(alpha-alpha_goal) + kd*(omega-omega_goal))
        return mtr_new_speed
        # return 0

def initialize_all():
        print "Enable motor №", mtr_num
        motor_turn_on(mtr_num)
        sleep(1)

        print "Enable angular velocity sensor №", hyr_num
        hyro_turn_on(hyr_num)
        sleep(1)

        print "Enable magnetometer", mag_num
        magnetometer_turn_on(mag_num)
        sleep(1)

def switch_off_all():
        print "Finishing..."
        print "Disable angular velocity sensor №", hyr_num
        hyro_turn_off(hyr_num)
        print "Disable magnetometer", mag_num
        magnetometer_turn_off(mag_num)
        motor_set_speed(mtr_num, 0)
        sleep (1)
        motor_turn_off(mtr_num)
        print "Finish program"

def control():
        initialize_all()
        mtr_state = 0
        hyro_state = 0
        mag_state = 0
```

```
        alpha_goal = 0
        omega_goal = 0

        for i in range(1000):
                mag_state, magx_raw, magy_raw, magz_raw = magnetometer_request_raw(mag_num)
                hyro_state, gx_raw, gy_raw, gz_raw = hyro_request_raw(hyr_num)
                mtr_state, mtr_speed = motor_request_speed(mtr_num)

                if not mag_state:
                        magx_cal, magy_cal, magz_cal =
mag_calibrated(magx_raw,magy_raw,magz_raw)
                        magy_cal = - magy_cal
                        mag_alpha = math.atan2(magy_cal, magx_cal)/math.pi*180
                        mag_alpha = angle_transformation(mag_alpha, alpha_goal)
                        print "magx_cal =", magx_cal, "magy_cal =", magy_cal, "magz_cal =", magz_cal
#
                        print "mag_alpha atan2= ", mag_alpha
                elif mag_state == 1:
                        print "Fail because of access error, check the connection"
                elif mag_state == 2:
                        print "Fail because of interface error, check your code"

                if not hyro_state:
                        gx_degs = gx_raw * 0.00875
                        gy_degs = gy_raw * 0.00875
                        gz_degs = gz_raw * 0.00875
                        omega = gz_degs
                        print "gx_degs =", gx_degs, "gy_degs =", gy_degs, "gz_degs =", gz_degs
                elif hyro_state == 1:
                        print "Fail because of access error, check the connection"
                elif hyro_state == 2:
                        print "Fail because of interface error, check your code"

                if not mtr_state:
                        print "Motor_speed: ", mtr_speed
                        mtr_new_speed =
motor_new_speed_PD(mtr_speed,mag_alpha,alpha_goal,gz_degs,omega_goal)
                        motor_set_speed(mtr_num, mtr_new_speed)
                sleep(time_step)
        switch_off_all()
```

Задача 3 Калибровка «звездного датчика»

```
#include <SPI.h>
#include <Pixy.h>

// This is the main Pixy object
```

```
Pixy StarSensor;

void setup()
{
	Serial.begin(9600);
	Serial.print("Starting...\n");
	StarSensor.init();
}

void loop()
{
	unsigned int  x1, x2, x3, y1, y2, y3, ab, bc,ca, mini, maxi, midl;
	int xx, xx1, yy, yy1;
	static int i = 0;
	int j;
	uint16_t blocks;
	char buf[32];

	// grab blocks!
	blocks = StarSensor.getBlocks();

	// If there are detect blocks, print them!
	if (blocks)
	{
		i++;
		if (i%30==0)
		{
			x1 = StarSensor.blocks [0] .x;
			y1 = StarSensor.blocks [0] .y;
			x2 = StarSensor.blocks [1] .x;
			y2 = StarSensor.blocks [1] .y;
			x3 = StarSensor.blocks [2] .x;
			y3 = StarSensor.blocks [2] .y;

			ab=sqrt(((x1-x2)*(x1-x2)) + ((y1-y2)*(y1-y2)));
			bc=sqrt(((x2-x3)*(x2-x3)) + ((y2-y3)*(y2-y3)));
			ca=sqrt(((x1-x3)*(x1-x3)) + ((y1-y3)*(x1-y3)));

			if (ab > ca && ab > bc){maxi = ab;xx= x1; xx1=x2; yy=y1; yy1=y2;}
			if (bc > ca && bc > ab){maxi = bc;xx= x2; xx1=x3; yy=y2; yy1=y3;}
			if (ca > ab && ca > bc){maxi = ca;xx= x1; xx1=x3; yy=y1; yy1=y3;}

			if (ab < ca && ab < bc){maxi = ab;}
			if (bc < ca && bc < ab){maxi = bc;}
			if (ca < ab && ca < bc){maxi = ca;}

			if ((mini != ab) && (maxi != ab)){midl = ab; }
			if ((mini != bc) && (maxi != bc)){midl = bc; }
			if ((mini != ca) && (maxi != ca)){midl = ca; }

			int x =(xx-xx1);
			int y =(yy-yy1);
```

```
                    double  angl = (atan2(x,y))*180/3.14159625358;
                    Serial.println(abs(180 - angl));
                }
        }
}
```

Задача 4 Настройка оптического канала связи

```
#include <stdint.h>
const uint8_t DIODE_PIN = 11;
const unsigned long UART_BAUD_RATE = 600;
const unsigned long UART_DELAY_US = 1000000 / UART_BAUD_RATE;
const unsigned long CARRIER_FREQUENCY = 38000;
const unsigned long CARRIER_DELAY_CYCLES = F_CPU / (CARRIER_FREQUENCY * 2);

static void initPWM()
{
        TCCR1A = 0;
        TCCR1B = 0;
        TCCR1C = 0;
        TIMSK1 = 0;
}

static void startPWM()
{
        TCCR1B = 0;
        TCCR1A = 0;
        TCNT1 = 0;
        OCR1A = CARRIER_DELAY_CYCLES - 1;
        TCCR1A = (0 << COM1A1) | (1 << COM1A0) | (1 << WGM11) | (1 << WGM10);
        TCCR1B = (1 << WGM13) | (1 << WGM12) | (0 << CS12) | (0 << CS11) | (1 << CS10);
}

static void stopPWM()
{
        TCCR1B = 0;
        TCCR1A = 0;
}

static void sendZero()
{
        startPWM();
        delayMicroseconds(UART_DELAY_US);
}

static void sendOne()
{
        stopPWM();
        delayMicroseconds(UART_DELAY_US);
```

```
}

void setup()
{
	Serial.begin(115200);
	pinMode(DIODE_PIN, OUTPUT);
	digitalWrite(DIODE_PIN, LOW);
	initPWM();
}

void sendByte(uint8_t byte)
{
	sendZero();

	for (int i = 0; i < 8; i++) {
		Serial.println(byte & 1);
		if ((byte & 1) != 0) {
			sendOne();
		} else {
			sendZero();
		}
		byte >>= 1;
	}
	sendOne();
}

void loop()
{
	if (Serial.available()){
		char buf = Serial.read();
		sendByte(buf);
		delay(0);
		sendByte(~buf);
		delay(0);
	}
}
```

Задание
День третий 26.02.2018

Мы всё больше и больше понимаем о нашем спутнике. Приходит время сделать что-то действительно новое. Такое чего до вас никто не делал..

Код всех заданий предполагающих программирование необходимо в конце дня сохранять на сервер, иначе комиссия может обнулить выставленный за задание бал. Сегодня коды необходимо сохранять в папку соответствующую третьему дню. Критерии оценок содержат часть элементов прошлых дней. Их могут сдать только команды, не сдававшие их в прошлые дни.

Задача 1 Настройка оптического канала связи

Настало время передать массив данных побольше. Похожий на тот, что нам действительно нужен. А заодно проверить насколько корректно он передается. На сервере лежит программа которая визуализирует принятые данные. Вы можете использовать её или написать собственную на том языке который вы предпочитаете.

Задача 2 Настройка звездного датчика.

Настраиваем звездный датчик так, чтобы его показания по возможности наиболее точно соответствовали текущему углу. Ваша задача определить погрешность его показаний. На основании этого вы сможете довести его точность до желаемой.
Кроме того, стоит задача интеграции - передача показаний звездного датчика на БКУ. См подробнее пункт «Интеграция».

Задача 3 Ориентация и остановка вращения

Аппарат должен вести себя предсказуемо при выполнении различных маневров. Нужно научится выполнять повороты аппарата на желаемый угол для выполнения задачи миссии.

Задача 4 Интеграция с системной шиной через Шилд

Аппарат построен на основе системной шины. В этой шине принят определенный протокол передачи данных. Задача состоит в том, чтобы научить устройство слышать нужные пакеты и реагировать на них.

Задача 5 Интеграция

Звездный датчик должен иметь интерфейс позволяющий ему работать с бортовой шиной. Ваша задача объединить две части задачи в одно целое.

Таблица 3. Критерии третьего дня.

Критерий	**Максимальный балл**	**Оценка**
Оптический канал		
Переданные данные корректны при кратковременных прерывания сигнала (не более 10% от общего времени передачи) ***Проверка:*** *в папку “Звездный датчик” на сервере положить файлы ir_transmitter_reliable.ino и*	5	

ir_receiver_reliable.ino		
Время передачи тестового массива не превышает одной минуты ***Проверка:*** быть готовым продемонстрировать результат после 16:30	10	
Визуализация тестового массива ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Загрузить скриншот переданных данных и программу визуализации в папку “Звездный датчик»	5	
Звёздный датчик		
Точность определения угла по ЗД не хуже чем 5 градусов в диапазоне от 0 до 360 градусов ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы star_sensor_orientation.ino в папку “Звездный датчик»	5	
Ориентация и стабилизация		
Ориентация аппарата по магнитометру: разворот неподвижного спутника на 180 градусов и стабилизация его в этом положении ***Проверка:*** залить на БКУ программу **magnitometer_orientation.zip** в папку “Калибровка магнитометра” на сервере Быть готовым продемонстрировать результат после 16:30	10	
Шилд (Создать папку “Шилд” на сервере)		
Показать сообщение в шине конструктора в мониторе порта Arduino ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы shield_test.ino в папку “Шилд».	1	
Построковый вывод сообщений с использованием разделителя, указанного в документации ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы shield_test_00.ino в папку “Шилд».	2	
Включение светодиода по высланному сигналу с БКУ ***Проверка:*** быть готовым продемонстрировать	2	

результат после 16:30 Залить файл программы led_test.ino в папку “Шилд».		
Передача текстового сообщения из консоли БКУ в консоль Arduino ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы shield_text_receive.ino в папку “Шилд».	10	
Передача числа с шилда на БКУ и вывод в веб-консоль БКУ ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы shield_number_transmit.ino в папку “Шилд».	10	
Интеграция		
Вывод показаний звездного датчика в консоль веб-интерфейса БКУ ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы shield_star_data_transmit.ino в папку “Шилд».	10	
ИТОГО максимальный балл	**70**	
ИТОГО максимальный балл по заданиям текущего дня	**65**	

Решение заданий третьего дня

Задача 1 Настройка оптического канала связи

Код принимающей части на Компьютере:

```
import processing.serial.*;

PImage img;
Serial myPort;
int val;
int dec;
int count = 0;
int i = 0;

void setup() {
        size(100, 100);
        printArray(Serial.list());
        String portName = Serial.list()[0];
        myPort = new Serial(this, portName, 115200);
        img = createImage(100, 100, ALPHA);
```

```
}

void draw() {
        background(0);
        while ( myPort.available() > 0) {  // If data is available,
                val = myPort.read();          // read it and store it in val
                for(i=0; i<8; i++) {
                        img.pixels[count] = color(((val>>i)&1));
                        count++;
                }
                if (count >=  img.pixels.length) {
                        count = 0;
                }
        }
        img.updatePixels();
        image(img, 0, 0);
}
```

Код для передатчика см. в заданиях 4 и 5 данного дня.

Задача 2 Настройка звездного датчика.

```
#include <math.h>
#include <SPI.h>
#include <Pixy.h>

struct vector {
  int zx = 0;
  int zy = 0;

  int lx = 0;
  int ly = 0;

  float lAngle = 0;

  float angle = 0.00;
  boolean control = false;
  };

const float kFilter = 0.50;
const uint8_t frame = 6;

Pixy pixy;
vector horizontal;

boolean GetVectors();
void SettingZeroPos();
float FilterAngle();
boolean GetAngle(float * angle);

void setup()
```

```
  {
  Serial.begin(115200);
  while (!Serial)
    {
    ;
    }
  Serial.print("Starting...\n");
  pixy.init();
  SettingZeroPos();
  }

void loop()
  {
  float angle = 0;
  if(GetAngle(&angle))Serial.println(angle);
  else
    {
    #ifdef test
    Serial.println("error of get angle");
    #endif
    }

  }

boolean GetAngle(float * angle)
  {
  int sumX = 0;
  int sumY = 0;
  uint8_t  nVector = 0;
  for(uint8_t i = 0; i < frame; i++)
    {
    if(GetVectors())
      {
      sumX += horizontal.lx;
      sumY += horizontal.ly;
      nVector++;
      }
    }

  if(nVector > 0)
    {
    horizontal.lx /= nVector;
    horizontal.ly /= nVector;
    *angle = FilterAngle();
    return true;
    }
  else return false;
  }

float FilterAngle()
  {
```

```
  horizontal.angle = 0;

  horizontal.angle =  kFilter*(atan2(horizontal.zy, horizontal.zx) - atan2(horizontal.ly,
horizontal.lx))/3.1415926535*180 + (1 - kFilter)*horizontal.lAngle;
  horizontal.lAngle = horizontal.angle;
  #ifdef test
  Serial.print( (atan2(horizontal.zy, horizontal.zx) - atan2(horizontal.ly,
horizontal.lx))/3.1415926535*180);
  Serial.print("\t,\t ");
  Serial.println(horizontal.angle);

  #endif
  if(horizontal.angle < 0)
    {
    return -horizontal.angle;
    }
  if(horizontal.angle >= 0)
    {
    return 360 - horizontal.angle;
    }

  }


boolean GetVectors()
  {
  horizontal.control = false;

  uint16_t blocks = 0;

  for(int i = 0; i < 500 && blocks != 2; i++)
    {
    blocks = pixy.getBlocks();

    if(blocks == 2 && pixy.blocks[1].signature == 2 && pixy.blocks[0].signature == 1)
      {
      horizontal.lx = pixy.blocks[0].x - pixy.blocks[1].x;
      horizontal.ly = pixy.blocks[0].y - pixy.blocks[1].y;

      horizontal.control = true;
      return true;
      }
    }

  return false;
  }


void SettingZeroPos()
  {
  while(!GetVectors())
    {
```

```
  ;
  }

 horizontal.zx = horizontal.lx;
 horizontal.zy = horizontal.ly;
 }
```

Задача 3 Ориентация и остановка вращения

Код стабилизации на C:

```
#include "libschsat.h"
void magCalibration(int xin, int yin, int zin, float *x, float *y, float *z)
{
        float A[3][3] = {{-0.081, -0.007, -2.017},
                                {0.917, 1.155, -0.844},
                                {-0.692, 0.002, -0.119}};
        float b[3] = {2.289, 1.203, -4.32};
        //float xin = *x, yin = *y, zin = *z;
        *x = (A[0][0])*(xin-b[0])+(A[0][1])*(yin-b[1])+(A[0][2])*(zin-b[2]);
        *y = (A[1][0])*(xin-b[0])+(A[1][1])*(yin-b[1])+(A[1][2])*(zin-b[2]);
        *z = (A[2][0])*(xin-b[0])+(A[2][1])*(yin-b[1])+(A[2][2])*(zin-b[2]);
}

double convert(double value,double From1,double From2,double To1,double To2)
{
    return (value-From1)/(From2-From1)*(To2-To1)+To1;
}

double correction(double a)
{
        double b;
        if(a > 0 && a < 120)
        {
                b = a + (0.0153*a*a-1.0677*a-3.8721);
                b = b < 180 ? b : 179.9999;
        }
        else if(a >= 120)
        {
                b = convert(a, 120, 180, -180, -150);
        }
        else if(a > -110)
        {
                b = a + (-0.0104*a*a-1.1699*a-0.3628);
        }
        else
        {
                b = 0.5149*a-62.567;
                b = b > -180 ? b : -179.9999;
        }
```

```
	return b;
}
void control(void)
{
	const int motorNum = 1, gyroNum = 1, magNum = 1;
	int16_t temp;
	//int16_t rpm = 0;

	int u = 0, err = 0, errold = 0, errsum = 0, speed = 0;
	int zerospeed = 0;
	int aerr = 0, aerrold = 0, aerrsum = 0;
	double kp = 10, ki = 0.005, kd = -1;
	double kps = 150, kis = 0, kds = -10;

	double alpha = 0, target = 0;

	if(LSS_OK != magnetometer_turn_on(magNum)) {printf("Motor connection fail!"); return;}
	if(LSS_OK != motor_turn_on(motorNum)) {printf("Motor connection fail!"); return;}
	if(LSS_OK != hyro_turn_on(gyroNum)) {printf("Gyroscope connection fail!"); return;}

	printf("Connection complete!");

	int16_t xg, yg, zg;

	Sleep(1);
	int counter = 0;
	Sleep(2);

	float xf, yf, zf;
	int16_t x, y, z;

	while(1)
	{
		if (LSS_OK == hyro_request_raw(gyroNum, &x, &y, &z)) {
			//printf("x=%d y=%d z=%d", x, y, z);
		}
		else
		{
			/*x = 0;
			y = 0;
			z = 0;*/
			puts("Fail! Gyro");
		}

		err = -z;
		errsum += err;

		u = kp*err + ki * errsum + kd * (err - errold);

		if(u > 5000) u = 5000;
		if(u < -5000) u = -5000;
```

```
                motor_set_speed(motorNum, u, &temp);
                //printf("          Motor speed: %d\n", temp);

                errold = err;
                Sleep(0.01);
                if(abs(err) < 10) {zerospeed = temp; break;}
        }
        magnetometer_request_raw(magNum, &x, &y, &z);
        magCalibration(x, y, z, &xf, &yf, &zf);
        alpha = atan2(xf, zf)*180.0/3.1415926;
        alpha = correction(alpha);

        target = alpha + 180 > 180 ? alpha - 180 : alpha + 180;

        while(1)
        {
                if (LSS_OK == magnetometer_request_raw(magNum, &x, &y, &z)) {
                        magCalibration(x, y, z, &xf, &yf, &zf);
                        alpha = atan2(xf, zf)*180.0/3.1415926;
                        alpha = correction(alpha);
                        printf("%f\n", alpha);
                        //printf("%d: x=%d y=%d z=%d\n", i, x, y, z);
                } else {puts("Fail! Mag");}

                if (LSS_OK == hyro_request_raw(gyroNum, &xg, &yg, &zg)) {
//printf("x=%d y=%d z=%d", x, y, z);
                }
                else{/*x = 0;y = 0;z = 0;*/puts("Fail! Gyro");}

                aerr = target - alpha;
                aerr = aerr < -180 ? aerr + 360 : aerr > 180 ? aerr - 360 : aerr;
                aerrsum += aerr;

                speed = aerr * kps + kis * aerrsum + kds * (aerr - aerrold);

                if(speed > 2000) speed = 2000;
                if(speed < -2000) speed = -2000;

                err = speed-zg;
                errsum += err;

                u = kp*err + ki * errsum + kd * (err - errold);

                if(u > 5000) u = 5000;
                if(u < -5000) u = -5000;

                motor_set_speed(motorNum, zerospeed + u, &temp);
                //printf("          Motor speed: %d\n", temp);
                aerrold = aerr;
                errold = err;
                Sleep(0.01);
                //if(abs(err) < 50) break;
```

```
            if(counter > 3000) break;
            counter++;
        }
        //Sleep(10);
        motor_set_speed(motorNum, 0, &temp);
        printf("Disable motor #%d\n", motorNum);
        motor_turn_off(motorNum);
}
```

Задача 4 Интеграция с системной шиной через Шилд

```
uint8_t buf[254], cnt = 0;
bool transmit = 0, strtflag = 0;
char to_addr_str[100], from_addr_str[100], msg_type_str[100];

uint8_t COBS_decode(uint8_t * buf)
{
        uint8_t i = 0xFF, next_null = 0;

        while (true)
        {
            if (Serial2.available())
            {
                uint8_t in_byte = Serial2.read();
                //Serial.write(in_byte);
                if (!in_byte || i == 0xFE)
                    break;
                if (i == 0xFF)
                    next_null = in_byte - 1;
                else if (i == next_null)
                {
                    next_null = i + in_byte;
                    buf[i] = 0;
                }
                else
                    buf[i] = in_byte;
                ++i;
            }
        }
        //Serial.write(0xAA);
        return (i < 0xFE) ? i : 0xFF;
}

int msg_type_decode(char * res_str, uint16_t msg_type)
{
        char *msg_type_str;
        //Serial.println("checking ");
        //Serial.println(msg_type, HEX);
        switch (msg_type)
        {
            case (0x0000): msg_type_str="RPI_OBC_BASE"; break;
            case (0x0200): msg_type_str="SENS_LIGHT_REQ_RAW"; break;
```

```
                case (0x0208): msg_type_str="SENS_LIGHT_CMD_RESET"; break;
                case (0x0210): msg_type_str="SENS_LIGHT_REQ_MAXRAW"; break;
                case (0x0220): msg_type_str="SENS_LIGHT_SET_MINVALUE"; break;
                case (0x0228): msg_type_str="SENS_LIGHT_SET_CALIBRATE"; break;
                case (0x0280): msg_type_str="SENS_LIGHT_RESP_RAW"; break;
                case (0x0288): msg_type_str="SENS_LIGHT_RESP_MAXRAW"; break;
                case (0x0300): msg_type_str="SENS_MAG_REQ_RAW"; break;
                case (0x0304): msg_type_str="SENS_MAG_CMD_RESET"; break;
                case (0x0380): msg_type_str="SENS_MAG_RESP_RAW"; break;
                case (0x0400): msg_type_str="SENS_HYRO_REQ_RAW"; break;
                case (0x0404): msg_type_str="SENS_HYRO_CMD_RESET"; break;
                case (0x0480): msg_type_str="SENS_HYRO_RESP_RAW"; break;
                case (0x0500): msg_type_str="SENS_SUN_REQ_RAW"; break;
                case (0x0508): msg_type_str="SENS_SUN_CMD_RESET"; break;
                case (0x0510): msg_type_str="SENS_SUN_REQ_MAXRAW"; break;
                case (0x0518): msg_type_str="SENS_SUN_SET_MINVALUE"; break;
                case (0x0580): msg_type_str="SENS_SUN_RESP_RAW"; break;
                case (0x0588): msg_type_str="SENS_SUN_RESP_MAXRAW"; break;
                case (0x0600): msg_type_str="SENS_ACCEL_REQ_RAW"; break;
                case (0x0604): msg_type_str="SENS_ACCEL_CMD_RESET"; break;
                case (0x0680): msg_type_str="SENS_ACCEL_RESP_RAW"; break;
                case (0x0D00): msg_type_str="CONTROL_WHEEL0_SET_SPEED"; break;
                case (0x0D04): msg_type_str="CONTROL_WHEEL0_CMD_RESET"; break;
                case (0x0D08): msg_type_str="CONTROL_WHEEL0_REQ_SPEED"; break;
                case (0x0D10): msg_type_str="CONTROL_WHEEL0_RESP_SPEED_CONFIRM";
break;
                case (0x0D0C): msg_type_str="CONTROL_WHEEL0_RESP_SPEED"; break;
                case (0x0E00): msg_type_str="CONTROL_FAN0_SET_SPEED"; break;
                case (0x0E04): msg_type_str="CONTROL_FAN0_CMD_RESET"; break;
                case (0x0E08): msg_type_str="CONTROL_FAN0_REQ_SPEED"; break;
                case (0x0E10): msg_type_str="CONTROL_FAN0_RESP_SPEED_CONFIRM"; break;
                case (0x0E0C): msg_type_str="CONTROL_FAN0_RESP_SPEED"; break;
                case (0x0F00): msg_type_str="CONTROL_COIL_SET_VAL"; break;
                case (0x0F10): msg_type_str="CONTROL_COIL_CMD_RESET"; break;
                case (0x0F20): msg_type_str="CONTROL_COIL_RESP_VAL_CONFIRM"; break;
                case (0x1000): msg_type_str="CONTROL_ENGINE_SET_VAL"; break;
                case (0x1010): msg_type_str="CONTROL_ENGINE_CMD_RESET"; break;
                case (0x1020): msg_type_str="CONTROL_ENGINE_RESP_VAL_CONFIRM"; break;
                case (0x2000): msg_type_str="MISC_LASER_CMD_ON"; break;
                case (0x2004): msg_type_str="MISC_LASER_CMD_OFF"; break;
                case (0x2008): msg_type_str="MISC_LASER_CMD_RESET"; break;
                case (0x200C): msg_type_str="MISC_LASER_RESP_ON_CONFIRM"; break;
                case (0x2010): msg_type_str="MISC_LASER_RESP_OFF_CONFIRM"; break;
                case (0x2100): msg_type_str="MISC_HF_REQ_STATE"; break;
                case (0x2110): msg_type_str="MISC_HF_CMD_RESET"; break;
                case (0x2120): msg_type_str="MISC_HF_RESP_STATE"; break;
                case (0x2200): msg_type_str="MISC_UHF_CMD_SEND"; break;
                case (0x2210): msg_type_str="MISC_UHF_CMD_RESET"; break;
                case (0x2220): msg_type_str="MISC_UHF_REQ_BUFFER"; break;
                case (0x2230): msg_type_str="MISC_UHF_RESP_BUFFER"; break;
                case (0x2300): msg_type_str="MISC_SUN_REQ_RAW"; break;
                case (0x2308): msg_type_str="MISC_SUN_CMD_RESET"; break;
```

```
                case (0x2310): msg_type_str="MISC_SUN_REQ_MAXRAW"; break;
                case (0x2380): msg_type_str="MISC_SUN_RESP_RAW"; break;
                case (0x2388): msg_type_str="MISC_SUN_RESP_MAXRAW"; break;
                case (0x2400): msg_type_str="MISC_EARTHFIELD_CMD_SET"; break;
                case (0x2408): msg_type_str="MISC_EARTHFIELD_CMD_AMPRESET"; break;
                case (0x2410): msg_type_str="MISC_EARTHFIELD_CMD_RESET"; break;
                case (0x2500): msg_type_str="MISC_GROUNDLEDS_CMD_SET"; break;
                case (0x2510): msg_type_str="MISC_GROUNDLEDS_CMD_RESET"; break;
                case (0x2600): msg_type_str="MISC_MOTOGLOBE_CMD_GO_SLOW"; break;
                case (0x2610): msg_type_str="MISC_MOTOGLOBE_CMD_GO_FAST"; break;
                case (0x2620): msg_type_str="MISC_MOTOGLOBE_CMD_START"; break;
                case (0x2630): msg_type_str="MISC_MOTOGLOBE_CMD_STOP"; break;
                case (0x2640): msg_type_str="MISC_MOTOGLOBE_CMD_ENABLEINT"; break;
                case (0x2650): msg_type_str="MISC_MOTOGLOBE_CMD_DISABLEINT"; break;
                case (0x2660): msg_type_str="MISC_MOTOGLOBE_CMD_RESET"; break;
                case (0x2700): msg_type_str="MISC_ARM_CMD_SET_STATE"; break;
                case (0x2710): msg_type_str="MISC_ARM_CMD_RESET"; break;
                case (0x2720): msg_type_str="MISC_ARM_RESP_STATE"; break;
                default: return 1; break;
        }
        strcpy(res_str, msg_type_str);
        return 0;
}

int addr_decode(char * res_str, uint16_t addr)
{
        char *addr_str;
        //Serial.println("checking ");
        //Serial.println(addr, HEX);
        switch (addr)
        {
                case (0x0001): addr_str="OBC_ADDRESS"; break;
                case (0x0010): addr_str="SENS_LIGHT_ADDRESS"; break;
                case (0x0018): addr_str="SENS_MAG_ADDRESS"; break;
                case (0x001C): addr_str="SENS_HYRO_ADDRESS"; break;
                case (0x0020): addr_str="SENS_SUN_ADDRESS"; break;
                case (0x0028): addr_str="SENS_ACCEL_ADDRESS"; break;
                case (0x0100): addr_str="CONTROL_WHEEL0_ADDRESS"; break;
                case (0x0110): addr_str="CONTROL_FAN0_ADDRESS"; break;
                case (0x0120): addr_str="CONTROL_COIL_ADDRESS"; break;
                case (0x0130): addr_str="CONTROL_ENGINE_ADDRESS"; break;
                case (0x0200): addr_str="MISC_LASER_ADDRESS"; break;
                case (0x0210): addr_str="MISC_HF_ADDRESS"; break;
                case (0x0220): addr_str="MISC_UHF_ADDRESS"; break;
                case (0x0230): addr_str="MISC_SUN_ADDRESS"; break;
                case (0x0240): addr_str="MISC_EARTHFIELD_ADDRESS"; break;
                case (0x0250): addr_str="MISC_GROUNDLEDS_ADDRESS"; break;
                case (0x0260): addr_str="MISC_MOTOGLOBE_ADDRESS"; break;
                case (0x0270): addr_str="MISC_ARM_ADDRESS"; break;
                case (0xFFFF): addr_str="MISC_ADDRESS_BROADCAST"; break;
                default: return 1; break;
        }
```

```
        strcpy(res_str, addr_str);
        return 0;
}

void setup() {
        Serial.begin(115200);
        Serial2.begin(115200);
        pinMode(13,OUTPUT);
        digitalWrite(13,LOW);
}

void loop() {
        while (Serial.available() > 0) {
                digitalWrite(13,HIGH);
                delay(0);
                Serial2.write(Serial.read());
                digitalWrite(13,LOW);
        }

        if (Serial2.available()) {
                Serial.println();
                Serial.print("Get:");
                uint8_t cnt = COBS_decode(buf);
                for (size_t i = 0; i < cnt; ++i)
                {
                        Serial.print(buf[i], HEX);
                        Serial.print(' ');
                }
                uint16_t msg_type = ((uint16_t*)buf)[0],
                to_addr = ((uint16_t*)buf)[1],
                from_addr = ((uint16_t*)buf)[2];
                Serial.println();
                uint16_t c_msg_type = msg_type, c_to_addr = to_addr, c_from_addr = from_addr;
                bool aflag = true;

                while (msg_type - c_msg_type < 5 && msg_type_decode(msg_type_str, c_msg_type) ||
(aflag = false) == true)
                {
                        --c_msg_type;
                        //aflag = false;
                }
                if (!aflag)
                {
                        Serial.print("type ");
                        Serial.print(msg_type_str);
                        Serial.print(' ');
                        Serial.print(msg_type - c_msg_type);
                        Serial.println();
                }
                else
                        Serial.println("msg_type_fail");
                aflag = true;
```

```
                while (to_addr - c_to_addr < 5 && addr_decode(to_addr_str, c_to_addr) || (aflag = false)
== true)
                {
                        --c_to_addr;
                        //aflag = false;
                }
                if (!aflag)
                {
                        Serial.print("to ");
                        Serial.print(to_addr_str);
                        Serial.print(' ');
                        Serial.print(to_addr - c_to_addr);
                        Serial.println();
                }
                else
                        Serial.println("to_addr_fail");
                aflag = true;
                while (from_addr - c_from_addr < 15 && addr_decode(from_addr_str, c_from_addr) ||
(aflag = false) == true)
                {
                        --c_from_addr;
                        //aflag = false;
                }
                if (!aflag)
                {
                        Serial.print("from ");
                        Serial.print(from_addr_str);
                        Serial.print(' ');
                        Serial.print(from_addr - c_from_addr);
                        Serial.println();
                }
                else
                        Serial.println("from_addr_fail");
                Serial.print("msg ");
                for (size_t i = 6; i < cnt; ++i)
                {
                        Serial.print(buf[i], HEX);
                        Serial.print(' ');
                }
                Serial.println();
        }
        delay(0);
}
```

Задача 5 Интеграция

```
uint8_t buf[254], cnt = 0;
bool transmit = 0, strtflag = 0;
char to_addr_str[100], from_addr_str[100], msg_type_str[100];

uint8_t COBS_decode(uint8_t * buf)
{
	uint8_t i = 0xFF, next_null = 0;

	while (true)
	{
		if (Serial2.available())
		{
			uint8_t in_byte = Serial2.read();
			//Serial.write(in_byte);
			if (!in_byte || i == 0xFE)
				break;
			if (i == 0xFF)
				next_null = in_byte - 1;
			else if (i == next_null)
			{
				next_null = i + in_byte;
				buf[i] = 0;
			}
			else
				buf[i] = in_byte;
			++i;
		}
	}
	//Serial.write(0xAA);
	return (i < 0xFE) ? i : 0xFF;
}

int COBS_encode(uint8_t * buf, size_t sz = 254)
{
	if (sz > 254)
		return 1;
	size_t last_null = 0;

	for (size_t i = 0; i <= sz; ++i)
	{
		if (i == sz || !buf[i])
		{
			size_t cnt = i - last_null + 1;
		Serial2.write(cnt);
		if (last_null != i)
		{
			for (size_t j = last_null; j < i; ++j)
			{
				Serial2.write(buf[j]);}
			}
			last_null = i + 1;
		}
```

```
        }
        Serial2.write(0);
        return 0;
};

void setup() {
        Serial.begin(115200);
        Serial2.begin(115200);
        pinMode(13,OUTPUT);
        digitalWrite(13,LOW);
}

void loop() {
        unsigned int  x1, x2, x3, y1, y2, y3, ab, bc,ca, mini, maxi, midl;
        int xx, xx1, yy, yy1;
        static int i = 0;
        int j;
        uint16_t blocks;
        char buf[32];

        // grab blocks!
        blocks = StarSensor.getBlocks();

        // If there are detect blocks, print them!
        if (blocks)
        {
                i++;


                if (i%30==0)
                {
                        x1 = StarSensor.blocks [0] .x;
                        y1 = StarSensor.blocks [0] .y;
                        x2 = StarSensor.blocks [1] .x;
                        y2 = StarSensor.blocks [1] .y;
                        x3 = StarSensor.blocks [2] .x;
                        y3 = StarSensor.blocks [2] .y;

                        ab=sqrt(((x1-x2)*(x1-x2)) + ((y1-y2)*(y1-y2)));
                        bc=sqrt(((x2-x3)*(x2-x3)) + ((y2-y3)*(y2-y3)));
                        ca=sqrt(((x1-x3)*(x1-x3)) + ((y1-y3)*(x1-y3)));

                        if (ab > ca && ab > bc){maxi = ab;xx= x1; xx1=x2; yy=y1; yy1=y2;}
                        if (bc > ca && bc > ab){maxi = bc;xx= x2; xx1=x3; yy=y2; yy1=y3;}
                        if (ca > ab && ca > bc){maxi = ca;xx= x1; xx1=x3; yy=y1; yy1=y3;}

                        if (ab < ca && ab < bc){maxi = ab;}
                        if (bc < ca && bc < ab){maxi = bc;}
                        if (ca < ab && ca < bc){maxi = ca;}

                        if ((mini != ab) && (maxi != ab)){midl = ab; }
                         if ((mini != bc) && (maxi != bc)){midl = bc;
```

```
                    if ((mini != ca) && (maxi != ca)){midl = ca; }
                    int x =(xx-xx1);
                    int y =(yy-yy1);
                     double  angl = (atan2(x,y))*180/3.14159625358;
                    buf = char(angl);
                    COBS_encode(buf, sizeof(buf))
                    digitalWrite(13,HIGH);
                    Serial2.println(abs(180 - angl));
                    digitalWrite(13,LOW);
                }
        }
}
```

Задание
День четвертый 27.02.2018

Близится наилучший момент для запуска. Пора проводить финальные испытания. Всё ли у вас готово для того, чтобы аппарат работал как единое целое?

Код всех заданий предполагающих программирование необходимо в конце дня сохранять на сервер, иначе комиссия может обнулить выставленный за задание бал. Сегодня коды необходимо сохранять в папку соответствующую третьему дню. Критерии оценок содержат часть элементов прошлых дней. Их могут сдать только команды, не сдававшие их в прошлые дни. Такие элементы выделены *курсивом*.

Сегодня мы проводим испытания на стенде. Ваш аппарат должен отработать миссию и передать данные в цуп.

Таблица 4. Критерии оценки четвертого дня.

Критерий	**Максимальный балл**	**Оценка**
Ориентация		
Ориентация и стабилизация спутника с использованием «звездного датчика» ***Проверка:*** залить файл программы star_orientation.exe на БКУ, подготовить “звездное небо”, быть готовым продемонстрировать результат после 16:30	10	
Шилд		
Передача числа с шилда на БКУ и вывод в веб-консоль БКУ ***Проверка:*** быть готовым продемонстрировать результат после 13:00 Залить файл программы shield_number_transmit.ino в папку “Шилд”.	5	
Интеграция		

Вывод показаний звездного датчика в консоль веб-интерфейса БКУ ***Проверка:*** быть готовым продемонстрировать результат после 13:00 Залить файл программы shield_star_data_transmit.ino в папку “Шилд”.	5	
Передача файла, лежащего на БКУ через оптический канал в программу ЦУП ***Проверка:*** быть готовым продемонстрировать результат после 16:30 Залить файл программы на сервер с названием transmitter	10	
Передача данных с “БКУ” висящего на подвесе спутника с использованием направленного передатчика на “ЦУП” через “НИП” находящийся на “планете” Планета неподвижна, “НИП” доступен в течение 1 минуты ***Проверка:*** быть готовым продемонстрировать результат на стенде после 16:30	20	
Передача данных с “БКУ” висящего на подвесе спутника с использованием направленного передатчика на “ЦУП” через “НИП” находящийся на “планете” Планета вращается. ***Проверка:*** быть готовым продемонстрировать результат на стенде после 16:30	30	
ИТОГО	**80**	

Решение заданий четвертого дня

Задача 1 Интеграция

Решением задания четвертого дня является интеграция ранее полученных результатов в единое целое.

```
void control(void)
{
	const uint16_t tx_num = 2;
	const uint16_t rx_num = 1;
	const uint8_t hello[] = {
0x42, 0x4D, 0x7E, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x00, 0x28, 0x00,
0x00, 0x00, 0x64, 0x00, 0x00, 0x00, 0x64, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x40, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xF8, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x80, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFE, 0x00, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xF8, 0x00, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xF0, 0x00, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xC0, 0x1F, 0x83, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
```

0xFF, 0xFF, 0x80, 0x7F, 0x8F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0x01, 0xFF, 0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFE, 0x07, 0xFF, 0xFF, 0xF0, 0x00, 0x3F, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFE, 0x0F, 0xFF, 0xFF, 0x80, 0x00, 0x07, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFC, 0x0F, 0xFF, 0xFC, 0x00, 0x00, 0x01, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xC0, 0xFC, 0x1F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x80, 0x7C, 0x3F, 0xFF, 0xC0, 0x03, 0xFE, 0x00, 0x1F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x3E, 0x7F, 0xFF, 0x00, 0x1F, 0xFF, 0xE0, 0x0F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFE,
0x00, 0x3E, 0xFF, 0xFC, 0x00, 0xFF, 0xFF, 0xF8, 0x07, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFC,
0x0C, 0x1F, 0xFF, 0xF0, 0x03, 0xFF, 0xFF, 0xFE, 0x03, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF8,
0x1C, 0x1F, 0xFF, 0xE0, 0x0F, 0xFF, 0xFF, 0xFF, 0x81, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF8,
0x3E, 0x0F, 0xFF, 0x80, 0x3F, 0xFF, 0xFF, 0xFF, 0xC0, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0,
0x7F, 0x07, 0xFE, 0x00, 0xFF, 0xCF, 0xFF, 0xFF, 0xE0, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0,
0x7F, 0x07, 0xF8, 0x03, 0xFE, 0x01, 0xFF, 0xFF, 0xF0, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE0,
0xFF, 0x83, 0xF0, 0x0F, 0xFE, 0x00, 0x3F, 0xFF, 0xF8, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xC1,
0xFF, 0x83, 0xC0, 0x1F, 0xFF, 0x00, 0x07, 0xFF, 0xF8, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xC1,
0xFF, 0xC1, 0x00, 0x7F, 0xFF, 0xE0, 0x00, 0xFF, 0xF8, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xC3,
0xFF, 0xC0, 0x01, 0xFF, 0xFF, 0xF8, 0x00, 0x1F, 0xFE, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x83,
0xFF, 0xE0, 0x07, 0xFF, 0xFF, 0xFF, 0x00, 0x03, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x87,
0xFF, 0xF0, 0x0F, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x7F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x87,
0xFF, 0xFC, 0x3F, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x0F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x07,
0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x03, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x07,
0xFC, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x01, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xFC, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xFC, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF8, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF8, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x9F, 0xFC, 0x07, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF8, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x83, 0xFE, 0x03, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF8, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC1, 0xFF, 0x81, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF0, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC1, 0xFF, 0xC0, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x0F,
0xF0, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0xFF, 0xE0, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x07,
0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x7F, 0xF0, 0x70, 0x00, 0x00, 0x00, 0xFF, 0x07,
0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x3F, 0xF8, 0x30, 0x00, 0x00, 0x00, 0xFF, 0x87,
0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x3F, 0xF8, 0x30, 0x00, 0x00, 0x00, 0xFF, 0x87,
0xC1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x1F, 0xFC, 0x30, 0x00, 0x00, 0x00, 0xFF, 0x8F,
0xC1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x0F, 0xFC, 0x10, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xC3, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0x0F, 0xFE, 0x10, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x83, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0xFE, 0x10, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x03, 0xFE, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFC,
0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x83, 0xFE, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x80,
0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xCF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x00,
0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00,
0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x80, 0x01,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x80, 0x1F,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x10, 0x00, 0x00, 0x00, 0x07, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x10, 0x00, 0x00, 0x00, 0x07, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, 0x30, 0x00, 0x00, 0x00, 0x07, 0xFF,
0x9F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x00, 0x07, 0xFE,
0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x87, 0xFE,
0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x07, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x87, 0xFF,

```
0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x0F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x83, 0xFF,
0x83, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x3F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xC3, 0xFF,
0x83, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x3F, 0xDF, 0xF0, 0x00, 0x00, 0x00, 0xC1, 0xFF,
0xC1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x7E, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0xC1, 0xFF,
0xC1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x7E, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xE0, 0xFF,
0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x7E, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xE0, 0x7F,
0xF0, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x7F, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0x3F,
0xF0, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x7F, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xF8, 0x3F,
0xF8, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFC, 0x0F,
0xF8, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFC, 0x07,
0xFC, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFE, 0x03,
0xFF, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0x00,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xC0,
0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE0,
0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0,
0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFC,
0x00, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF3, 0xFE, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xE3, 0xFB, 0xFE, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xC0, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0xFF, 0xFE, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xF8, 0x00, 0x0F, 0xFF, 0xFF, 0xFE, 0x00, 0x7F, 0xFE, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFE, 0x00, 0x01, 0xFF, 0xFF, 0xF8, 0x00, 0x3F, 0xFC, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE3,
0xFF, 0xE0, 0x00, 0x7F, 0xFF, 0xF0, 0x00, 0x1F, 0xFC, 0x3F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE0,
0xFF, 0xFC, 0x00, 0x0F, 0xFF, 0xC0, 0x1C, 0x1F, 0xF8, 0x3F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE1,
0xFF, 0xFF, 0xC0, 0x07, 0xFF, 0x80, 0x7E, 0x0F, 0xF8, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE0,
0xFF, 0xFF, 0xF8, 0x03, 0xFE, 0x01, 0xFE, 0x0F, 0xF0, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xE0,
0x7F, 0xFF, 0xFF, 0x0F, 0xF8, 0x03, 0xFF, 0x07, 0xE0, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0,
0x3F, 0xFF, 0xFF, 0xFF, 0xE0, 0x0F, 0xFF, 0x07, 0xC0, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF8,
0x1F, 0xFF, 0xFF, 0xFF, 0xC0, 0x3F, 0xFF, 0x83, 0x81, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFC,
0x07, 0xFF, 0xFF, 0xFF, 0x00, 0x7F, 0xFF, 0x81, 0x03, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFE,
0x01, 0xFF, 0xFF, 0xF8, 0x01, 0xFF, 0xF7, 0xC0, 0x03, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x3F, 0xFF, 0xE0, 0x07, 0xFF, 0xE3, 0xE0, 0x07, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x80, 0x03, 0xFE, 0x00, 0x1F, 0xFF, 0xC3, 0xE0, 0x0F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xE0, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xC1, 0xF0, 0x3F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xF8, 0x00, 0x00, 0x01, 0xFF, 0xFF, 0x83, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0x03, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xF0, 0x00, 0x7F, 0xFF, 0xFE, 0x07, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x0F, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0x9F, 0xF0, 0x1F, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFE, 0x1F, 0xC0, 0x3F, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFE, 0x0C, 0x00, 0x7F, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00};
        printf("Enable transceiver #%d\n", tx_num);
        transceiver_turn_on(tx_num);
        motor_turn_on(1);
        Sleep(1);
        bus_setup();
        int temp;
```

```
        int i = 0;
        printf("Send data from #%d to #%d\n", tx_num, rx_num);
        transceiver_send(tx_num, rx_num, hello, sizeof(hello));
        printf("Send data from #%d to #%d\n", tx_num, rx_num);
        Sleep(100);
        printf("Disable transceiver #%d\n", tx_num);
        transceiver_turn_off(tx_num);
        return;
}
```

Задание
День пятый 28.02.2018

Сегодня день финальных испытаний. Вы должны продемонстрировать выполнение миссии вашим аппаратом. Ваши программы должны работать как единое целое.

Код всех заданий предполагающих программирование необходимо в конце дня сохранять на сервер в папке финального решения, иначе комиссия может обнулить выставленный за задание бал. Сегодня коды необходимо сохранять в папку соответствующую четвертому дню.

При сдаче рассматриваются две основных ситуации - «геостационарная» орбита спутника и спутник на экваториальной орбите. Вы должны сделать всё возможное, чтобы выполнить миссию в комплексе.

Таблица 5. День пятый. Критерии оценки.

Критерий	**Максимальный балл**	**Оценка**
Интеграция		
Вывод показаний звездного датчика в консоль веб-интерфейса БКУ ***Проверка:*** быть готовым продемонстрировать результат после 11:00 Залить файл программы shield_star_data_transmit.ino в папку “Шилд”.	15	
Передача файла, лежащего на БКУ через оптический канал в программу ЦУП ***Проверка:*** быть готовым продемонстрировать результат после 11:00 Залить файл программы на сервер с названием transmitter	5	
Передача данных с “БКУ” висящего на подвесе спутника с использованием направленного передатчика на “ЦУП” через “НИП” находящийся на “планете” Планета неподвижна, “НИП” доступен в течение 1 минуты ***Проверка:*** быть готовым продемонстрировать результат на	20	

стенде после 12:30 При отсутствии одного из условий реализации задачи за задачу будут сниматься баллы следующим образом: ориентация осуществлена с недостаточной точностью - 10 баллов данные переданы частично - 5 баллов данные не переданы - 10 баллов		
Передача данных с “БКУ” висящего на подвесе спутника с использованием направленного передатчика на “ЦУП” через “НИП” находящийся на “планете” Планета вращается. ***Проверка:*** быть готовым продемонстрировать результат на стенде после 12:30	30	
ИТОГО	**70**	

Решение заданий пятого дня

Пятый день предполагает очную сдачу комиссии отработку решений прошлых дней в связке между собой. Коды решений соответствуют приведенным для 1-4 дня.