Работа призера заключительного этапа

командной инженерной олимпиады школьников

Олимпиада Национальной технологической инициативы

Профиль «Системы связи и дистанционного зондирования Земли»

Попов Максим Федорович

Класс: 10

Город: Москва

Школа: ГБОУ Лицей «Вторая школа»

Регион: Москва

Уникальный номер участника:
шифр: 625439
stepik ID: 34836125

Команда на заключительном этапе: Л2Ш

Параллель: 10-11

Результаты заключительного этапа:

ФИО	Командный балл max(215)	Физика max(60)	Информатика max(100)	Общий балл max(100)
Попов Максим Федорович	115	14,6	0	39,428

Индивидуальная часть

Физика

Персональный лист участника с номером 625439:

Олимпиада НТИ

ФИО Попов Максим Фёдорович

Город Москва

Школа № Лицей „Вторая Школа"

Командная инженерная олимпиада «Олимпиада НТИ»

Направление ССиФЗЗ

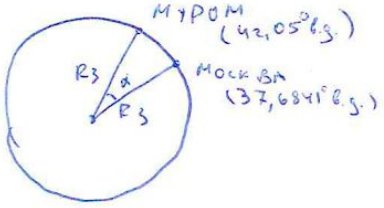
Предмет Физика

Номер участника 625 439

9,6 → 14,6

Спутниковая фотография.

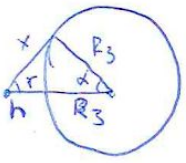
№ 5.



$\alpha = 37,05 - 37,6841 = 4,366$

$X = \sqrt{R_3^2 + (R_3+h)^2 - 2R_3(R_3+h)\cdot\cos\alpha} =$

$= \sqrt{(6371)^2 + 7071^2 - 2\cdot 7071\cdot 6371\cdot\cos\alpha} =$



$\frac{X}{\sin\alpha} = \frac{R_3}{\sin\gamma} \Rightarrow \sin\gamma = \frac{R_3}{X}\cdot\sin\alpha =$

$\gamma = 43^\circ \approx 0,75$ рад.

$\frac{\gamma}{2} = \frac{\varepsilon t^2}{2} \Rightarrow t = \sqrt{\frac{\gamma}{\varepsilon}} = \sqrt{\frac{0,6}{0.0314}} \approx 4,9$ с

Общее время поворота: 9,8 с.

№ 1.

$\frac{3500\text{ км}}{7500\times 3500} = \frac{1\text{ см}}{7500\text{ тыс.}} = 0,0004$ см/тысячу 0,2

Динамика спутников.

$V = \sqrt{g\cdot R} = \sqrt{9,222\cdot 6371000} = 7665$ м/с 0,4

$g = 9,222$ м/с (из параметров орбиты)

Командная инженерная олимпиада «Олимпиада НТИ»

Направление ССиФ33

Предмет Физика

Номер участника 625439

Ориентация и навигация.

$g = G \cdot \frac{M_3}{(h+R_3)^2} = 7,9639\ м/с^2$

№1

$V = \sqrt{g \cdot (R_3+h)} = 7,9639 \cdot 70$

$\omega = \sqrt{\frac{g}{R_3+h}} = \sqrt{\frac{7,9639}{6371 \cdot 10^3 + 700 \cdot 10^3}} = \sqrt{\frac{7,9639}{7071000}} \approx$ 0,00106 рад/сек 1

№2

$V_{спутника} = \sqrt{g \cdot (R_3+h)} = \sqrt{7,9639 \cdot 7071000} = 7504\ м/с$

$V_{земли} = \frac{2\pi R}{T} = \frac{2\pi \cdot 7071000}{((23 \cdot 60)+54) \cdot 60 + 4)s} = 516\ м/с$ 0

$V_{точки} = V_{спутника} - V_{земли} \approx$ 6990 м/с

№3

$\omega_{спутника} = 0,00106$ рад/сек

$T_{спутника} = \frac{2\pi}{\omega} = 5927$ сек

$k = \frac{\theta}{360^\circ} = \frac{5,7}{360} = 0,0158(3)$

$k \cdot T \approx$ 95 сек – длительность видео

№4.

Будем сначала разгонять спутник, а потом тормозить. Тогда попробуем разогнать спутник пока он не повернётся на $\frac{\varphi T}{2}$.

$\frac{\varphi T}{2} = \frac{\varepsilon \cdot t_1^2}{2} \Rightarrow \varphi T = \varepsilon \cdot t_1^2 \Rightarrow t_1^2 = \frac{\varphi T}{\varepsilon} \Rightarrow t_1 = \sqrt{\frac{\varphi T}{\varepsilon}} = \sqrt{\frac{3,14}{0,0314}} = \sqrt{100} = 10$ с.

А скорость в этот момент будет равна $\varepsilon \cdot t = 0,314$ рад/с, чего не может быть. Тогда разгонимся до ω_{max}, подождём и затормозим.

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Ф33

Предмет Физика

Номер участника 615439

№4 (продолж.)

$$\begin{cases} \omega_{max} = \varepsilon \cdot t_1 \\ \alpha = \frac{\varepsilon t_1^2}{2} \end{cases} \Leftrightarrow \begin{cases} \alpha = \frac{\omega_{max}^2}{2\varepsilon} = \frac{0{,}157^2}{2 \cdot 0{,}314} = 0{,}03925 \text{ рад} \\ t_1 = \frac{\omega_{max}}{\varepsilon} = 5 \text{ с} \end{cases}$$

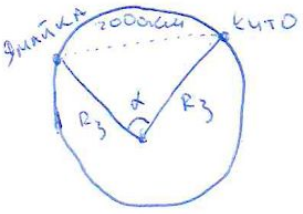
α - угол, на который повернут спутник при разгоне

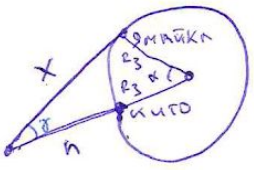
Тогда:

$$2\alpha = \omega_{max} \cdot t_2 \Rightarrow t_2 = \frac{\pi - 2\alpha}{\omega_{max}} \approx 19{,}5 \text{ с.}$$

Тогда общее время равно $2t_1 + t_2 \approx 30$ с. ~~2~~ 4.

№5





$$\alpha = \frac{2000000}{2\pi R_3} = \frac{2000}{2\pi \cdot 6371} \approx 0{,}5 \text{ рад} \approx 30^\circ$$

$$X = \sqrt{R_3^2 + (R_3 + h)^2 - 2R_3(R_3 + h) \cdot \cos\alpha} =$$

$$= \sqrt{(6371)^2 + (7071)^2 - 2 \cdot 7071 \cdot 6371 \cdot \frac{\sqrt{3}}{2}} = 3544 \text{ км}$$

$$\frac{3544}{\sin\alpha} = \frac{X}{\sin\alpha} = \frac{R_3}{\sin\gamma} \Rightarrow \sin\gamma = \frac{R_3 \cdot \sin\alpha}{X} =$$

6 ~~2~~ 9

$$= \frac{6371 \cdot \sin 30}{3544} \approx 0.9 \Rightarrow \gamma \approx 65^\circ = 1{,}1344 \text{ рад}$$

$$\frac{\gamma}{2} = \frac{\varepsilon t^2}{2} \Rightarrow t = \sqrt{\frac{\gamma}{\varepsilon}} = \sqrt{\frac{1.1344}{0.0314}} \approx 6\text{с}$$

Ответ: общее время $= 2t = 12$ с.

Индивидуальная часть

Информатика

Решения участника с stepik ID 34836125

Задача №1.1

```
#include "stdio.h"
#include "math.h"

int main()
   {
   int line[20000][2] = {};
   int waste[20000][3] = {};

   int WN = 0;
   int LN = 0;

   scanf ("%d", &LN);
   scanf ("%d", &WN);

   for (int i = 0; i < LN; i++)
      {
      scanf ("%d", &line[i][0]);
      scanf ("%d", &line[i][1]);
      }

    for (int i = 0; i < WN; i++)
      {
      scanf ("%d", &waste[i][0]);
      scanf ("%d", &waste[i][1]);
      }


   for (int i = 0; i < LN; i++)
      {
      for (int j = 0; j < WN; j++)
         {
         int lx = (waste[j][0]-line[i][0]);
         int ly = (waste[j][1]-line[i][1]);

         //printf ("waste[%d][0]=%d, line[%d][0]=%d\n", j, waste[j][0], i, line[i][0]);

         int l = sqrt(lx*lx+ly*ly);

         if (i == 0)
            {
            waste[j][2] = l;
            }
         else if (waste[j][2] > l) waste[j][2] = l;
```

```
            //printf ("lx=%d, ly=%d, l=%d; waste=%d\n", lx, ly, l, waste[j][2]);
            }
        }

    int max = waste[0][2];

    for (int i = 1; i < WN; i++)
        {
        if (max < waste[i][2]) max = waste[i][2];
        }

    printf ("%d", max);
    return 0;
    }
```

Задача №2.1

```
#include "stdio.h"
#include "math.h"

int main()
    {
    float line[20000][2] = {};
    float waste[20000][3] = {};

    int WN = 0;
    int LN = 0;

    scanf ("%d", &LN);
    scanf ("%d", &WN);

    //printf ("LN=%d; WN=%d\n", LN, WN);

    for (int i = 0; i < LN; i++)
        {
        scanf ("%f", &line[i][0]);
        scanf ("%f", &line[i][1]);

        //printf ("line0=%f, line1=%f; i = %d\n", line[i][0], line[i][1], i);
        }

     for (int j = 0; j < WN; j++)
        {
        float tmp = 0;
        scanf ("%f", &tmp);
        waste[j][0] = tmp;
        scanf ("%f", &waste[j][1]);

        //printf ("waste[0][0]=%f\n", waste[j][0]);
        }
```

```
    for (int i = 0; i < LN; i++)
        {
        for (int j = 0; j < WN; j++)
            {
            float lx = (waste[j][0]-line[i][0]);
            float ly = (waste[j][1]-line[i][1]);

            //printf ("waste[%f][0]=%f, line[%f][0]=%f\n", j, waste[j][0], i, line[i][0]);

            float l = sqrt(lx*lx+ly*ly);

            if (i == 0)
                {
                waste[j][2] = l;
                }
            else if (waste[j][2] > l) waste[j][2] = l;

           // printf ("lx=%f, ly=%f, l=%f; waste=%f\n", lx, ly, l, waste[j][2]);
            }
        }

    float max = waste[0][2];

    for (int i = 1; i < WN; i++)
        {
        if (max < waste[i][2]) max = waste[i][2];
        }

    printf ("%f\n", max);

    //cout << max;
    return 0;
    }
```

Задача №2.3

```
#include "stdio.h"
#include "math.h"

int main()
    {
    float line[20000][2] = {};
    float waste[20000][3] = {};

    int WN = 0;
    int LN = 0;

    scanf ("%d", &LN);
    scanf ("%d", &WN);

    //printf ("LN=%d; WN=%d\n", LN, WN);
```

```
for (int i = 0; i < LN; i++)
    {
    scanf ("%f", &line[i][0]);
    scanf ("%f", &line[i][1]);

    //printf ("line0=%f, line1=%f; i = %d\n", line[i][0], line[i][1], i);
    }

 for (int i = 0; i < WN; i++)
    {
    scanf ("%f", &waste[i][0]);
    scanf ("%f", &waste[i][1]);

    //printf ("waste[0][0]=%f\n", waste[0][0]);
    }


for (int i = 0; i < LN; i++)
    {
    for (int j = 0; j < WN; j++)
        {
        float lx = (waste[j][0]-line[i][0]);
        float ly = (waste[j][1]-line[i][1]);

        //printf ("waste[%f][0]=%f, line[%f][0]=%f\n", j, waste[j][0], i, line[i][0]);

        float l = sqrt(lx*lx+ly*ly);

        if (i == 0)
            {
            waste[j][2] = l;
            }
        else if (waste[j][2] > l) waste[j][2] = l;

       // printf ("lx=%f, ly=%f, l=%f; waste=%f\n", lx, ly, l, waste[j][2]);
        }
    }

float max = waste[0][2];

for (int i = 1; i < WN; i++)
    {
    if (max < waste[i][2]) max = waste[i][2];
    }

printf ("%f\n", max);

//cout << max;
return 0;
}
```

Командная часть

Результаты были получены в рамках выступления команды: Л2Ш

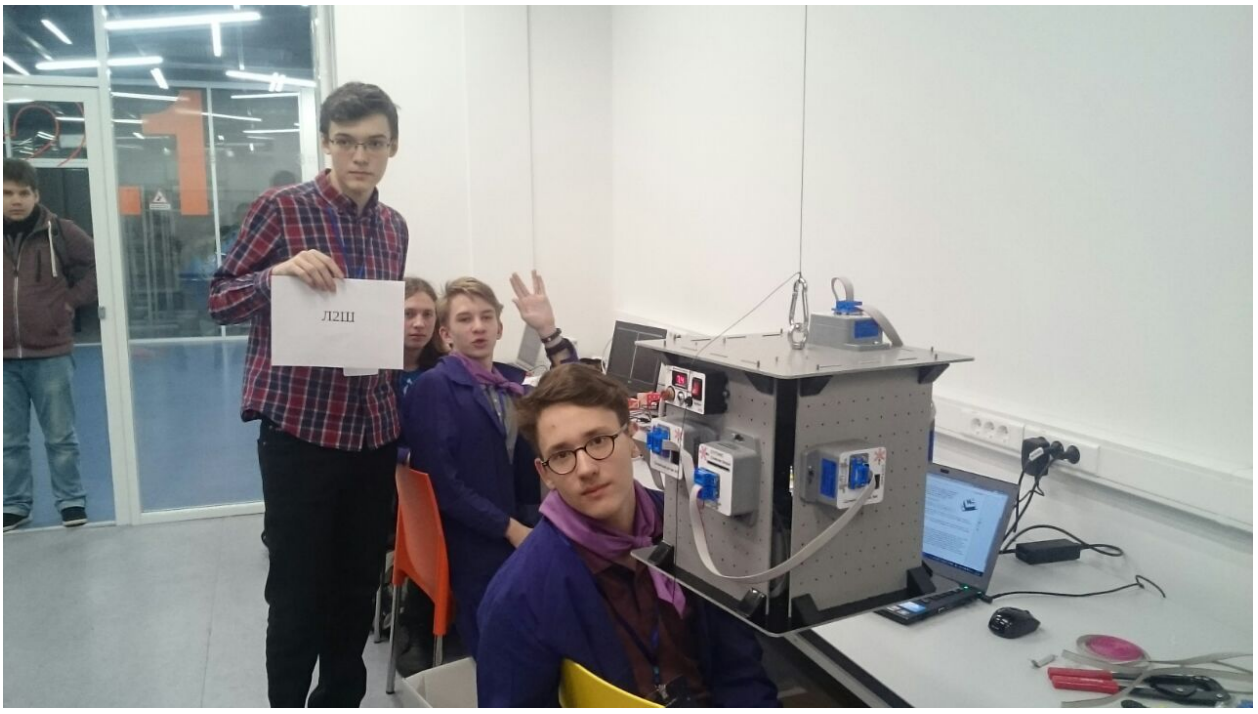
Личный состав команды:

- Попов Максим Фёдорович
- Заичкин Иван Николаевич
- Юровский Владимир Олегович
- Галустов Артемий Львович

Фото команды:

1. Сборка типовой платформы

На фото команда с собранным устройством в процессе работы:



2. Финальный код, написанный командой, и загружаемый на устройство БКУ (включает в себя подзадачи по тестированию устройств, калибровке магнитометра, задаче стабилизации и задаче интеграции):

```
#include "libschsat.h"

#define MAGNETOMETER_ID 1
#define MOTOR_ID 1
#define HYRO_ID 1

void Init ();

void StartStop (int16_t* speed);

void Main (int16_t* speed, int16_t* FinishAngle);

void magnetometerCorrection(float *x, float *y, float *z);

float magnetometerAngel (); //РљРsРґ РsС€РёР±РєРё -404, РёРSР°С‡Рµ С‡РёСЃР»Рs >= 0

int TakeHyroData (int16_t HyroData[3]); //РљРsРґ РsС€РёР±РєРё -404, РёРSР°С‡Рµ С‡РёСЃР»Рs >= 0

int StopIt (); //РљРsРґС‹ РsС€РёР±РsРє:-404, -303, -202, РёРSР°С‡Рµ С‡РёСЃР»Рs >= 0

int sabs (int16_t num);
```

```
int SpeedControl (int16_t *speed);

void sendingTime ();

//////////////////////////////////////////////////////////////////////
uint8_t data[1670] = {66, 77, 126, 6, 0, 0, 0, 0, 0, 0, 62, 0, 0, 0, 40, 0, 0, 0, 100, 0, 0, 0,
100, 0
, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 64, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, 0, -1, -1, -1, -1, -1, -8, 63, -1, -1, -1,
-1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -128, 15, -1, -1, -1, -1, -1, -16, 0,
0, 0, -1, -1, -1, -1, -2, 0, 7, -1, -1, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1
, -8, 0, 7, -1, -1, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -16, 0, 7, -1, -1,
 -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -64, 31, -125, -1, -1, -1, -1, -1, -16, 0, 0, 0, -1, -1,
-1, -1, -128, 127, -113, -1, -1, -1, -1, -1, -16, 0, 0, 0, -
1, -1, -1, -1, 1, -1, -65, -1, -1, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -2, 7,
-1, -1, -16, 0, 63, -1, -1, -16, 0, 0, 0, -1, -1, -1, -2, 15, -1, -1, -128, 0, 7
, -1, -1, -16, 0, 0, 0, -1, -1, -1, -4, 15, -1, -4, 0, 0, 1, -1, -1, -16, 0, 0,
0, -1, -1, -64, -4, 31, -1, -16, 0, 0, 0, 127, -1, -16, 0, 0, 0, -1, -1, -128, 124, 63, -1, -64,
3, -2, 0, 31, -1, -16, 0, 0, 0, -1, -1, 0, 62, 127, -1, 0, 31,
-1, -32, 15, -1, -16, 0, 0, 0, -1, -2, 0, 62, -1, -4, 0, -1, -1, -8, 7, -1, -16,
 0, 0, 0, -1, -4, 12, 31, -1, -16, 3, -1, -1, -2, 3, -1, -16, 0, 0, 0, -1, -8, 28, 31, -1, -32, 15,
-1, -1, -1, -127, -1, -16, 0, 0, 0, -1, -8, 62, 15, -1, -128
, 63, -1, -1, -1, -64, -1, -16, 0, 0, 0, -1, -16, 127, 7, -2, 0, -1, -49, -1, -1
, -32, 127, -16, 0, 0, 0, -1, -16, 127, 7, -8, 3, -2, 1, -1, -1, -16, 127, -16,
0, 0, 0, -1, -32, -1, -125, -16, 15, -2, 0, 63, -1, -8, 127, -16, 0, 0, 0, -1, -
63, -1, -125, -64, 31, -1, 0, 7, -1, -8, 127, -16, 0, 0, 0, -1, -63, -1, -63, 0,
 127, -1, -32, 0, -1, -8, 127, -16, 0, 0, 0, -1, -61, -1, -64, 1, -1, -1, -8, 0,
 31, -2, 127, -16, 0, 0, 0, -1, -125, -1, -32, 7, -1, -1, -1, 0, 3, -1, -1, -16,
 0, 0, 0, -1, -121, -1, -16, 15, -1, -1, -1, -32, 0, 127, -1, -16, 0, 0, 0, -1,
-121, -1, -4, 63, -1, -1, -1, -4, 0, 15, -1, -16, 0, 0, 0, -1, 7, -2, -1, -1, -1
, -1, -1, -1, -64, 3, -1, -16, 0, 0, 0, -1, 7, -4, 127, -1, -1, -1, -1, -1, -8,
1, -1, -16, 0, 0, 0, -1, 15, -4, 63, -1, -1, -1, -1, -1, -2, 0, 127, -16, 0, 0,
0, -1, 15, -4, 63, -1, -1, -1, -1, -1, -1, -64, 31, -16, 0, 0, 0, -1, 15, -8, 63
, -1, -1, -1, -1, -1, -1, -16, 15, -16, 0, 0, 0, -1, 15, -8, 63, -1, -1, -1, -1,
 -1, -97, -4, 7, -16, 0, 0, 0, -1, 15, -8, 127, -1, -1, -1, -1, -1, -125, -2, 3,
 -16, 0, 0, 0, -1, 15, -8, 127, -1, -1, -1, -1, -1, -63, -1, -127, -16, 0, 0, 0,
 -1, 15, -16, 127, -1, -1, -1, -1, -1, -63, -1, -64, -16, 0, 0, 0, -1, 15, -16,
127, -1, -1, -1, -1, -1, -32, -1, -32, -16, 0, 0, 0, -1, 7, -16, -1, -1, -1, -1,
 -1, -1, -32, 127, -16, 112, 0, 0, 0, -1, 7, -32, -1, -1, -1, -1, -1, -1, -16, 63, -8, 48, 0, 0, 0,
-1, -121, -32, -1, -1, -1, -1, -1, -1, -8, 63, -8, 48, 0, 0,
 0, -1, -121, -63, -1, -1, -1, -1, -1, -1, -4, 31, -4, 48, 0, 0, 0, -1, -113, -63, -1, -1, -1, -1,
-1, -1, -4, 15, -4, 16, 0, 0, 0, -1, -1, -61, -1, -1, -1, -1,
 -1, -1, -2, 15, -2, 16, 0, 0, 0, -1, -1, -125, -1, -1, -1, -1, -1, -1, -1, 7, -2, 16, 0, 0, 0, -1,
-1, 7, -1, -1, -1, -1, -1, -1, -1, 3, -2, 0, 0, 0, 0, -1, -4
, 7, -1, -1, -1, -1, -1, -1, -1, -125, -2, 0, 0, 0, 0, -1, -128, 15, -1, -1, -1,
 -1, -1, -1, -1, -49, -1, 0, 0, 0, 0, -32, 0, 31, -1, -1, -1, -1, -1, -1, -1, -1
, -1, 0, 0, 0, 0, -64, 0, 63, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -128, 1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -64, 0, 0, 0, 0, -128, 31, -1, -1, -1
, -1, -1, -1, -1, -4, 0, 16, 0, 0, 0, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -16, 0, 16, 0, 0, 0, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -64, 0, 48, 0, 0, 0
, 7, -1, -97, -1, -1, -1, -1, -1, -1, -1, 0, 1, -16, 0, 0, 0, 7, -2, 15, -1, -1,
 -1, -1, -1, -1, -2, 0, -1, -16, 0, 0, 0, -121, -2, 15, -1, -1, -1, -1, -1, -1,
-4, 7, -1, -16, 0, 0, 0, -121, -1, 7, -1, -1, -1, -1, -1, -1, -4, 15, -1, -16, 0, 0, 0, -125, -1,
```

```
-125, -1, -1, -1, -1, -1, -1, -8, 63, -1, -16, 0, 0, 0, -61, -1, -125, -1, -1, -1, -1, -1, -1, -8,
63, -33, -16, 0, 0, 0, -63, -1, -63, -1, -1
, -1, -1, -1, -1, -8, 126, 31, -16, 0, 0, 0, -63, -1, -63, -1, -1, -1, -1, -1, -1, -8, 126, 15,
-16, 0, 0, 0, -32, -1, -32, -1, -1, -1, -1, -1, -1, -16, 126, 15
, -16, 0, 0, 0, -32, 127, -16, 127, -1, -1, -1, -1, -1, -16, 127, 15, -16, 0, 0,
 0, -16, 63, -16, 127, -1, -1, -1, -1, -1, -16, 127, 15, -16, 0, 0, 0, -8, 63, -8, 63, -1, -1, -1,
-1, -1, -16, -1, 15, -16, 0, 0, 0, -4, 15, -8, 63, -1, -1, -1
, -1, -1, -16, -1, 15, -16, 0, 0, 0, -4, 7, -4, 31, -1, -1, -1, -1, -1, -16, -1,
 15, -16, 0, 0, 0, -2, 3, -1, 31, -1, -1, -1, -1, -1, -32, -1, 15, -16, 0, 0, 0,
 -1, 0, -1, -1, -1, -1, -1, -1, -1, -32, -1, 15, -16, 0, 0, 0, -1, -64, 63, -1,
-1, -1, -1, -1, -1, -31, -1, 15, -16, 0, 0, 0, -1, -32, 15, -1, -1, -1, -1, -1,
-1, -31, -1, 15, -16, 0, 0, 0, -1, -16, 3, -1, -1, -1, -1, -1, -1, -31, -1, 15,
-16, 0, 0, 0, -1, -4, 0, 127, -1, -1, -1, -1, -1, -13, -2, 15, -16, 0, 0, 0, -1,
 -1, 0, 15, -1, -1, -1, -1, -29, -5, -2, 15, -16, 0, 0, 0, -1, -1, -64, 0, -1, -1, -1, -1, -128, -1,
-2, 31, -16, 0, 0, 0, -1, -1, -8, 0, 15, -1, -1, -2, 0, 127, -2, 31, -16, 0, 0, 0, -1, -1, -2, 0, 1,
-1, -1, -8, 0, 63, -4, 31, -16, 0, 0,
0, -1, -29, -1, -32, 0, 127, -1, -16, 0, 31, -4, 63, -16, 0, 0, 0, -1, -32, -1,
-4, 0, 15, -1, -64, 28, 31, -8, 63, -16, 0, 0, 0, -1, -31, -1, -1, -64, 7, -1, -128, 126, 15, -8,
127, -16, 0, 0, 0, -1, -32, -1, -1, -8, 3, -2, 1, -2, 15, -16,
 127, -16, 0, 0, 0, -1, -32, 127, -1, -1, 15, -8, 3, -1, 7, -32, -1, -16, 0, 0,
0, -1, -16, 63, -1, -1, -1, -32, 15, -1, 7, -64, -1, -16, 0, 0, 0, -1, -8, 31, -1, -1, -1, -64, 63,
-1, -125, -127, -1, -16, 0, 0, 0, -1, -4, 7, -1, -1, -1, 0,127, -1, -127, 3, -1, -16, 0, 0, 0, -1,
-2, 1, -1, -1, -8, 1, -1, -9, -64, 3, -1
, -16, 0, 0, 0, -1, -1, 0, 63, -1, -32, 7, -1, -29, -32, 7, -1, -16, 0, 0, 0, -1
, -1, -128, 3, -2, 0, 31, -1, -61, -32, 15, -1, -16, 0, 0, 0, -1, -1, -32, 0, 0,
 0, 127, -1, -63, -16, 63, -1, -16, 0, 0, 0, -1, -1, -8, 0, 0, 1, -1, -1, -125,-1, -1, -1, -16, 0,
0, 0, -1, -1, -1, 0, 0, 15, -1, -1, 3, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -16, 0, 127, -1, -2, 7,
-1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -4, 15, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1,
-1, -1, -1, -1, -97, -16, 31, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -2, 31, -64, 63, -1, -1, -1,
-16, 0, 0, 0, -1, -1, -1, -1, -1, -2, 12, 0, 127, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -2,
0, 0, -1, -1, -1, -1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -1, 0, 3, -1, -1, -1, -1, -16, 0, 0, 0, -1,
-1, -1, -1, -1, -1, -1, 0, 15, -1, -1, -1,-1, -16, 0, 0, 0, -1, -1, -1, -1, -1, -1, -64, -1, -1, -1, -1,
-1, -16, 0, 0, 0
};

void sendPackage(uint8_t *buffer, int size) {

	transceiver_send(3, 1, buffer, size);
	printf("Debug : sending %i bytes : ", size);

	int i;
	for(i = 0; i < size; i++) {

	printf("%i ", buffer[i]);

	}

	printf("\n");

}

//////////////////////////////////////////////////////////////////
void control(void)
```

```
        {
        Init ();

        int16_t FinishAngle = 199;

        printf ("FinishAngle = %d\n", FinishAngle);

        int16_t speed = 0;
        StartStop (&speed);

        printf ("MAIN BEGIN\n");
        Main (&speed, &FinishAngle);

        sendingTime ();

        int16_t wasteVar = 0;
        motor_set_speed(MOTOR_ID, 0, &wasteVar);

        mSleep(5000);

        motor_turn_off(MOTOR_ID);
        hyro_turn_off(HYRO_ID);
        magnetometer_turn_off(MAGNETOMETER_ID);

        return;
        }
void StartStop (int16_t* speed)
        {
        int i;
        for (i = 0; i < 50; i++)
        {
        StopIt ();

        motor_request_speed (1, speed);

        mSleep (200);
        }
        }

void Init ()
        {
        motor_turn_on(MOTOR_ID);
        hyro_turn_on(HYRO_ID);
        magnetometer_turn_on(MAGNETOMETER_ID);

        int16_t StartAngle = magnetometerAngel ();

        while (StartAngle < 0)
        {
        printf ("Error: trying to take angel\n");
        StartAngle = magnetometerAngel ();
```

```
        mSleep (1000);
        }

        printf ("StartAngle = %d\n", StartAngle);
        }

void Main (int16_t* speed, int16_t* FinishAngle)
        {
        int16_t wasteVar = 0;
        int i;
        for (i = 0; i < 600; i++)
        {
        printf ("--------------------------------------------------\n\n");

        int16_t curAngel = magnetometerAngel ();
        while (curAngel < 0)
                {
                curAngel = magnetometerAngel ();
                mSleep(50);
                }

        int16_t rotAngel = (*FinishAngle - curAngel);
        //printf ("tmp: rotAngel=%d\n", rotAngel);
        if (abs(rotAngel) > 180) rotAngel -= 360*sabs(rotAngel);


        int16_t tmp = sabs(rotAngel);


        int16_t angleSpeed = -300*tmp;
        int16_t curAngelSpeed [3] = {};
        while (TakeHyroData (curAngelSpeed) < 0)
                {
                mSleep(50);
                }


        int16_t deltaAngelSpeed = curAngelSpeed[2]-angleSpeed;

        *speed -= deltaAngelSpeed*0.1;

        SpeedControl (speed);

        //printf ("tmp=%d\n", tmp);
        //printf ("curAngel = %d\n", curAngel);
        printf ("rotAngel = %d\n", rotAngel);

        if (abs(rotAngel) < 2)
                {
                printf ("FALSE1\n");
                int flag = StopIt();

                while (flag < 0)
```

```
                {
                flag = StopIt();
                mSleep (50);
                }

            printf ("FALSE2\n");

            if (flag == 100) break;
            }

        else
            {
            motor_set_speed(MOTOR_ID, *speed, &wasteVar);
            printf ("TRUE\n");
            }


        int16_t curSpeed = 0;
        motor_request_speed (MOTOR_ID, &curSpeed);

        printf ("curAngelSpeed=%d\n", curAngelSpeed[2]);
        //printf ("deltaAngelSpeed=%d\n\n", deltaAngelSpeed);
        //printf ("MAIN: speed=%d\n", speed);
        printf ("MAIN: curspeed=%d\n\n", curSpeed);

        mSleep (200);
        }
        }

void magnetometerCorrection(float *x, float *y, float *z)
        {
        float Ainv[3][3] = {{1.759203, 0.042054, 0.004133}, {0.042054, 1.632513,
0.114138}, {0.004133, 0.114138, 1.871844}};
        float b[3] = {93.819656, 49.502959, 31.326686};

        float 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]);
        }

float magnetometerAngel ()
        {
        int16_t x, y, z;
        float fx, fy, fz;

        if (magnetometer_request_raw(MAGNETOMETER_ID, &x, &y, &z) == LSS_OK)
        {
        fx = x;
```

```
        fy = y;
        fz = z;

        magnetometerCorrection(&fx, &fy, &fz);

        float Angel = (atan2(fx, fy) * 180 / 3.1415926)+180;

        return Angel;
        }

        else
        {
        printf("Magnetometer Fail\n");
        return -404;
        }
        }

int StopIt ()
        {
        int16_t HyroData[3] = {};
        if (TakeHyroData (HyroData) < 0) return -404;

        printf ("StopFunc: Gyro data z=%d\n", HyroData[2]);

        int16_t curSpeed = 0;
        if (motor_request_speed (1, &curSpeed) != LSS_OK) return -303;

        int16_t tmp = HyroData[2]*3;

        if (abs (HyroData[2]) <= 50) tmp = HyroData[2]/2;
        else if (abs (HyroData[2]) <= 150) tmp = HyroData[2];

        int16_t speed = curSpeed-tmp;

        SpeedControl (&speed);

        if (motor_set_speed(1, speed, &speed) != LSS_OK) return -202;

        //printf ("StopFunc: speed=%d\n", speed);
        printf ("StopFunc: curspeed=%d\n\n", curSpeed);

        if (HyroData[2] <= 150) return 100;

        return 0;
        }

int TakeHyroData (int16_t HyroData[3])
        {
        printf("Take hyro data #%d\n", HYRO_ID);

        if (hyro_request_raw (HYRO_ID, &HyroData[0], &HyroData[1], &HyroData[2]) ==
LSS_OK)
        {
```

```
        return 100;
        }
        else
        {
        printf ("Gyro Fail!\n");
        return -404;
        }
        }

int sabs (int16_t num)
        {
        if (num != 0) return (abs(num)/num);
        else return num;
        }

int SpeedControl (int16_t *speed)
        {
        if (*speed > 5000)
        {
        *speed = 5000;
        printf ("StopFunc: So big speed\n");
        return 1;
        }

        else if (*speed < -5000)
        {
        *speed = -5000;
        printf ("StopFunc: So small speed\n");
        return 1;
        }

        return 0;
        }

void sendingTime ()
        {
        printf ("SENDING\n");

        transceiver_turn_on(3);

        int i;
        for (i = 0; i < 300; i++)
        {
        int packNum = (i/3)*16;
        if ((i%3) == 0) sendPackage(&(data[packNum]), 16);
        StopIt ();
        mSleep (200);
        printf ("SENDING\n");
        }
        }
```

3. Работа с оптическим каналом (включает в себя все подзадачи,

связанные с оптическим каналом)

Код для передатчика:

```
//--------------MESSAGE-MAP----------

#define INIT_TRANS_ 0x2213
#define TRANSMIT_   0x2203
#define RECIVE_REQ  0x2223
#define RECIVE_ANS  0x2233

//--------------ADDRESS-MAP----------

#define MY_ADDR 0x0203

//-----------------END-------------

#include <stdint.h>

const uint8_t DIODE_PIN = 11;
const unsigned long UART_BAUD_RATE = 1000;
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 sendByte(uint8_t byte)
  {
  sendZero();

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

    byte >>= 1;
    }

  sendOne();
  }

int findZero(char *data, int start, int size) {

    for(int i = start; i < size; i++) {

        if(data[i] == 0) {

            return i;

        }

    }

    return -1;

}

bool formCOBS(char *data, int size, char *buffer) { //buffer size must not be less than size
+ 2

    int nextZero = findZero(data, 0, size);
```

```
if(nextZero == -1) {

    buffer[0] = 1 + size;

    for(int i = 0; i < size; i++) {

        buffer[i + 1] = data[i];

    }

    buffer[size + 1] = 0;

    return true;

} else {

    int lastZero = 0; // First free byte

    buffer[0] = 1 + nextZero;

    for(int i = 1; i <= nextZero; i++) {

        buffer[i] = data[i - 1];

    }

    lastZero = nextZero + 1;
    nextZero = findZero(data, lastZero, size);

    while(nextZero != -1) {

        buffer[lastZero] = 1 + nextZero - lastZero;

        for(int i = lastZero + 1; i <= nextZero; i++) {

            buffer[i] = data[i - 1];

        }

        lastZero = nextZero + 1;
        nextZero = findZero(data, lastZero, size);

    }

    buffer[lastZero] = 1 + size - lastZero;

    for(int i = lastZero + 1; i <= size; i++) {

        buffer[i] = data[i - 1];

    }

    buffer[size + 1] = 0;
```

```
    }

}


bool parseCOBS(char *package, int size, char *buffer)
  {
    int bytesWritten = 0;
    int iter = 0;

    while(package[iter + package[iter]] != 0) {

        for(int i = iter + 1; i < iter + package[iter]; i++) {

            if(bytesWritten == size) {

                return false;

            }

            buffer[bytesWritten] = package[i];
            bytesWritten ++;

        }

        if(bytesWritten == size) {

                return false;

        }

        buffer[bytesWritten] = 0;
        bytesWritten ++;
        iter = iter + package[iter];


        if(iter >= size) {

            return false;

        }

    }

    for(int i = iter + 1; i < iter + package[iter]; i++) {

        if(bytesWritten == size) {

            return false;

        }
```

```
      buffer[bytesWritten] = package[i];
      bytesWritten ++;

   }

   return true;

}

void setup()
 {
 // put your setup code here, to run once:
 Serial.begin (115200);
 Serial2.begin(115200);
 // в шилде системная шина подключена на Serial2
 pinMode(13, OUTPUT);     // определяет режим прием/передача
 pinMode(12, OUTPUT);
 pinMode(DIODE_PIN, OUTPUT);

 digitalWrite(13, LOW);   // режим прием
 digitalWrite(12, LOW);   // режим прием
 digitalWrite(DIODE_PIN, LOW);

 initPWM();
 }

String buf;

void SendMsg (char* msg);
void IR_Send(char* test);

void loop()
 {
 // put your main code here, to run repeatedly:

 if (Serial2.available() > 0)
  {
  char test = Serial2.read();
  buf += String(test);

  if (test == 0)
   {
   int length_ = buf.length();

   char* buf_char = new char[length_];
   char* decoded  = new char[length_];

   buf.toCharArray(buf_char, length_);

   Serial.println("I recieved package");

   if (parseCOBS (buf_char, length_, decoded))
    {
```

```
        Serial.println ("R:" + String(decoded));

        buf = "";

        if (int(decoded[3]) << 8 | decoded[2] == MY_ADDR)
          switch (int(decoded[1]) << 8 | decoded[0])
            {
            case INIT_TRANS_: Serial.println ("Enabling..."); digitalWrite (12, HIGH); break;
            case TRANSMIT_:   IR_Send(decoded + 8); buf = "";
break;//Serial.println(String((char*) decoded + 8)); break;

            //case RECIVE_REQ: SendMsg(); break;

            default: /*Serial.println("R_M: " + String(decoded));*/ break;
            }
          else
           {
           //Serial.println("R: " + String(decoded));
           }
        Serial.println("\n-----------------------------------------\n");
        }

      delete buf_char;
      delete decoded;
      }
    }
  }

void SendMsg ()
  {
  char string[] = {RECIVE_ANS & 0xFF, (RECIVE_ANS >> 8) & 0xFF, 0x01, 0x00, MY_ADDR
& 0xFF, (MY_ADDR >> 8) & 0xFF, '1', '2', '3', 0};

  char encoded[sizeof(string) + 2] = {};

  formCOBS (string, sizeof(string), encoded);

  Serial.write(encoded);
  }

void IR_Send(char* test)
  {
  for (int i = 0; i < 16; i++)
    {
    Serial.write(test[i]);

    sendByte(test[i]);
    sendByte(test[i]);
    sendByte(test[i]);

    delay(30);
    }
```

```
}
```

Код для приемника:

```
#include <SoftwareSerial.h>
#include <stdint.h>

SoftwareSerial mySerial(11, 10);

//--------------MESSAGE-MAP----------

#define INIT_TRANS_ 0x2213
#define TRANSMIT_   0x2203
#define RECIVE_REQ  0x2223

#define RECIVE_ANS  0x2233

//--------------ADDRESS-MAP----------

#define MY_ADDR 0x0201

//-----------------END-------------

int findZero(char *data, int start, int size) {

   for(int i = start; i < size; i++) {

      if(data[i] == 0) {

         return i;

      }

   }

   return -1;

}

bool formCOBS(char *data, int size, char *buffer) { //buffer size must not be less than size
+ 2

   int nextZero = findZero(data, 0, size);

   if(nextZero == -1) {

      buffer[0] = 1 + size;

      for(int i = 0; i < size; i++) {

         buffer[i + 1] = data[i];
```

```
        }

        buffer[size + 1] = 0;

        return true;

    } else {

        int lastZero = 0; // First free byte

        buffer[0] = 1 + nextZero;

        for(int i = 1; i <= nextZero; i++) {

            buffer[i] = data[i - 1];

        }

        lastZero = nextZero + 1;
        nextZero = findZero(data, lastZero, size);

        while(nextZero != -1) {

            buffer[lastZero] = 1 + nextZero - lastZero;

            for(int i = lastZero + 1; i <= nextZero; i++) {

                buffer[i] = data[i - 1];

            }

            lastZero = nextZero + 1;
            nextZero = findZero(data, lastZero, size);

        }

        buffer[lastZero] = 1 + size - lastZero;

        for(int i = lastZero + 1; i <= size; i++) {

            buffer[i] = data[i - 1];

        }

        buffer[size + 1] = 0;

    }

}

bool parseCOBS(char *package, int size, char *buffer)
  {
    int bytesWritten = 0;
```

```
    int iter = 0;

    while(package[iter + package[iter]] != 0) {

        for(int i = iter + 1; i < iter + package[iter]; i++) {

            if(bytesWritten == size) {

                return false;

            }

            buffer[bytesWritten] = package[i];
            bytesWritten ++;

        }

        if(bytesWritten == size) {

                return false;

        }

        buffer[bytesWritten] = 0;
        bytesWritten ++;
        iter = iter + package[iter];


        if(iter >= size) {

            return false;

        }

    }

    for(int i = iter + 1; i < iter + package[iter]; i++) {

        if(bytesWritten == size) {

            return false;

        }

        buffer[bytesWritten] = package[i];
        bytesWritten ++;

    }

    return true;

}
```

```
#define _HI(var) (((var) >> 8) & 0xFF)
#define _LO(var) ((var)& 0xFF)

void setup()
 {
 Serial.begin (115200);
 mySerial.begin(1000);
 }

uint8_t buf[3] = {};

int recv_count = 0;

bool find_true(uint8_t* answ);

uint8_t test = 0;
bool true_val = 0;

void loop()
 {
 if (mySerial.available() > 0)
  {
  if (recv_count < 2)
   {
   buf[recv_count] = mySerial.read();
   recv_count++;
   }
  else
   {
   buf[recv_count] = mySerial.read();
   recv_count = 0;

   if (find_true (&test))
    true_val = true;
   else
    true_val = false;
   }
  }

 if (true_val)
  {
  true_val = false;
  Serial.write(test);
  }

 /*if (Serial2.available() > 0)
  {
  char test = Serial2.read();
  buf += String(test);

  if (test == 0)
   {
   int length_ = buf.length();
```

```
    char* buf_char = new char[length_];
    char* decoded  = new char[length_];

    buf.toCharArray(buf_char, length_);

    if (parseCOBS (buf_char, length_, decoded))
      {
      //Serial.println (decoded);

      buf = "";

      if (int(decoded[3]) << 8 | decoded[2] == MY_ADDR1)
        {
        switch (int(decoded[1]) << 8 | decoded[0])
          {
          case INIT_TRANS_: //digitalWrite (12, HIGH); break;
          case RECIVE_REQ:  if (true_val) SendMsg (test); break;

          default: //Serial.print(String(decoded) + String(char(0xfe))); break;
          }
        }
      else
        {
        //Serial.print(String(decoded) + String(char(0xff)));
        }
      }

    delete buf_char;
    delete decoded;
    }
  }/**/

 delay(1);
 }

bool find_true(uint8_t* answ)
 {
 uint8_t test = 0;

 if (buf[0] == buf[1])
   {
   test = buf[0];
   }
 else if (buf[0] == buf[2])
   {
   test = buf[0];
   }
 else if (buf[1] == buf[2])
   {
   test = buf[1];
   }
 else
```

```
  {
  return false;
  }

 *answ = test;
 return true;
 }


void SendMsg (uint8_t byte)
 {
 uint8_t string[] = {_LO(RECIVE_ANS), _HI(RECIVE_ANS), _LO(0x0001), _HI(0x0001),
_LO(MY_ADDR), _HI(MY_ADDR), byte, 0};

 uint8_t encoded[sizeof(string) + 2] = {};

 formCOBS (string, sizeof(string), encoded);

 //Serial.(encoded);
 }
```

4. Код, написанный командой, для работы со звездным датчиком:

```
///ось X на камере направлена от эмблемы между камерой и мк
/// ось Y направлена от эмблемы ||линии  мк и оптике
#define test

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

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

 int dx = 0;
 int dy = 0;

 float dAngle = 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()
 {
 #ifdef test
 Serial.begin(9600);
 while(!Serial)
  {
  ; // леонардо, блин !!!
  }
 Serial.print("Starting...\n");
 #endif
 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))/3.14159265358*180 + (1 - kFilter)*horizontal.lAngle;
  horizontal.lAngle = horizontal.angle;
  #ifdef test
  Serial.print( (atan2(horizontal.zy, horizontal.zx) - atan2(horizontal.ly,
horizontal.lx))/3.14159265358*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 SettingZeroPose()
 {
 while(!GetVectors())
  {
  ;
  #ifdef test
  Serial.println("GET ZEROPos");
  #endif
  }


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