



ПРОЕКТ «ВИЗУАЛИЗАЦИЯ ДВИЖЕНИЯ АНПА В ПЛОСКОСТИ ГОРИЗОНТА V2»

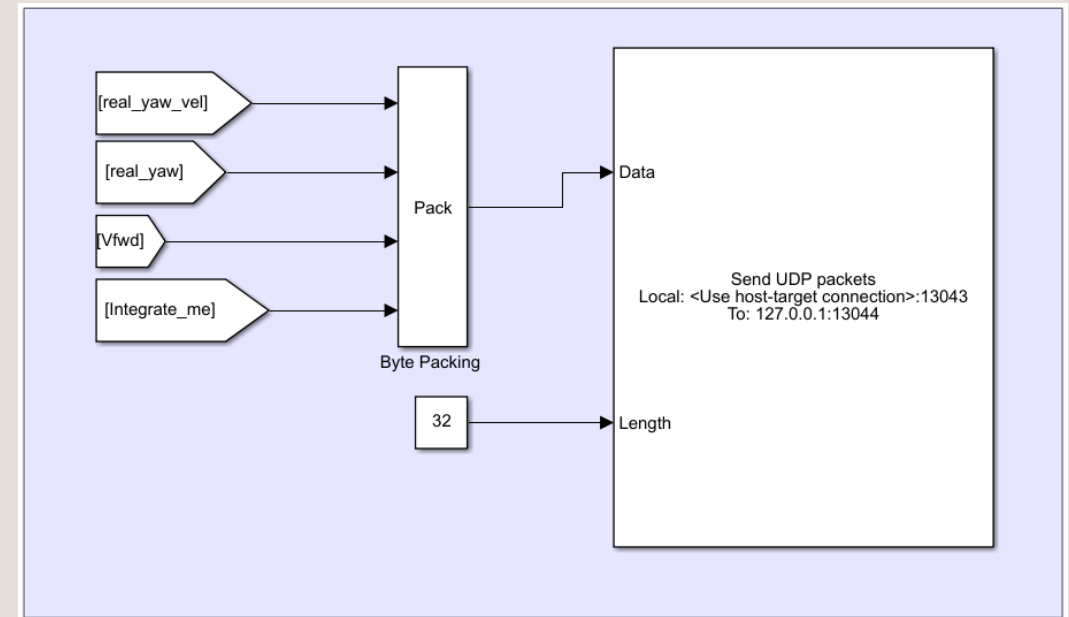
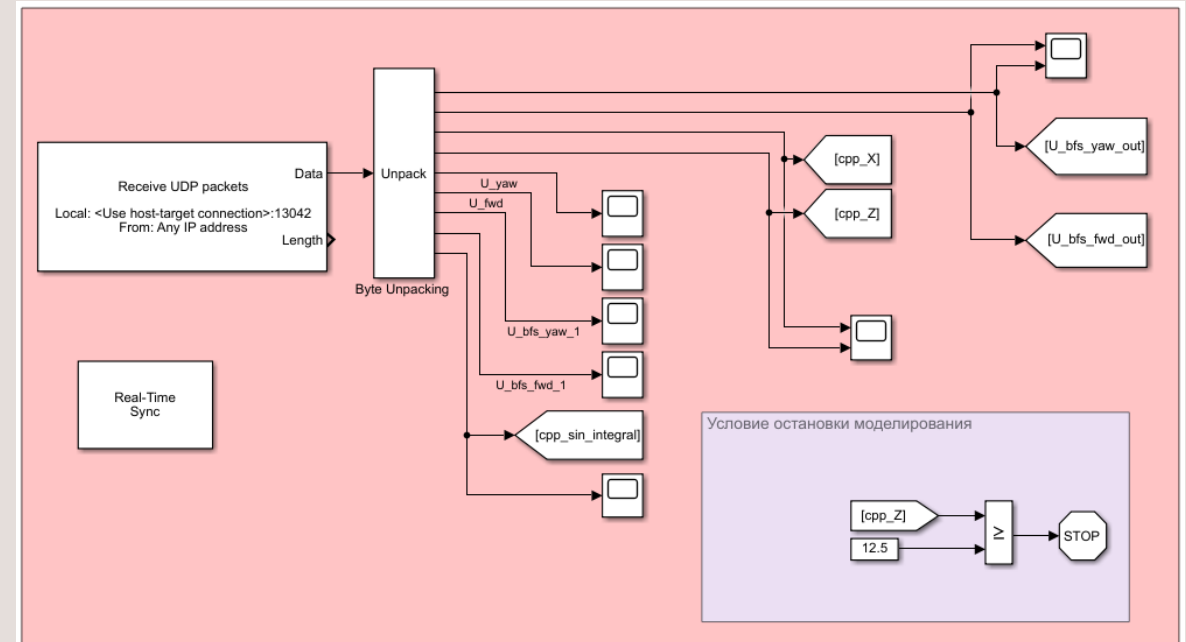
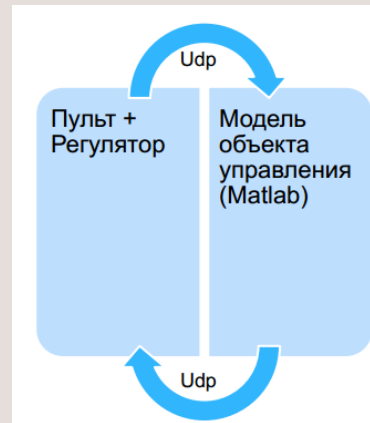
Андреев Е.В., СМ11-41М

Цели и задачи

- Получить зачёт по курсу «Проектирование ПО ПРТС»
- Попутно узнать что-нибудь новенькое для себя

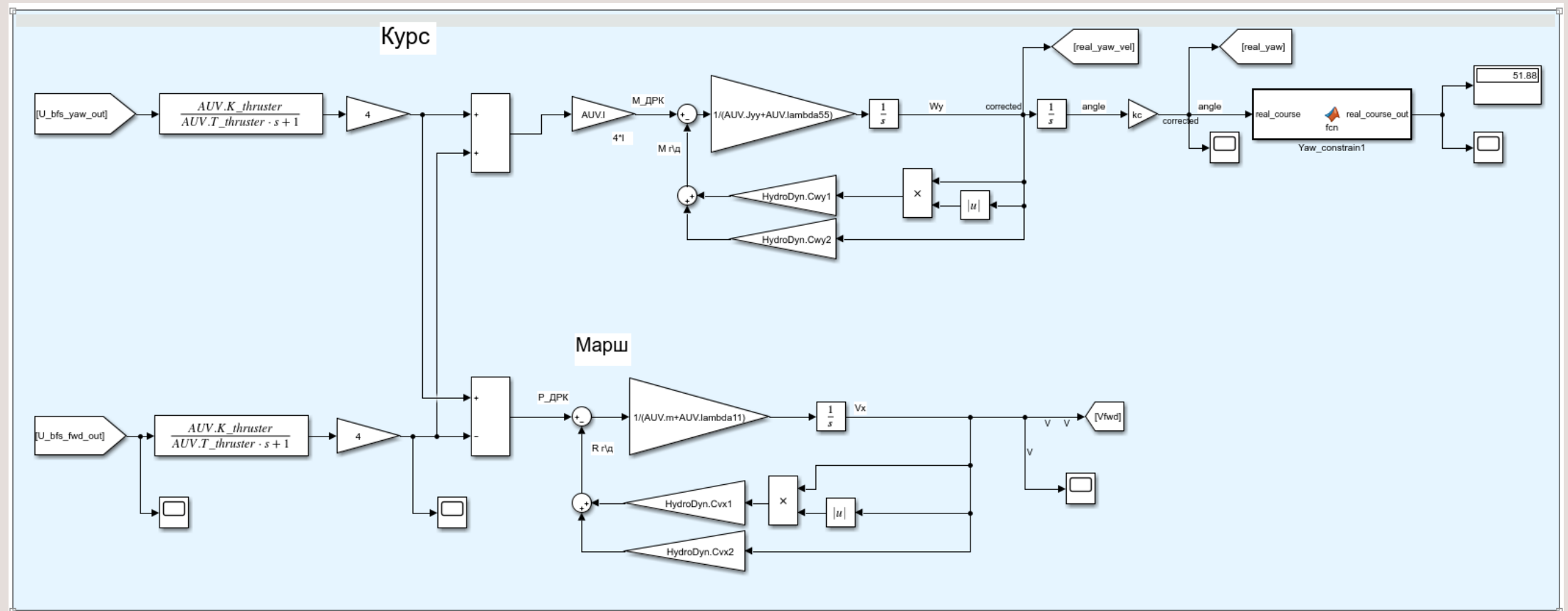
Избранный путь

- Обмен по UDP с периодом в 10 мс
- Получаем в модели
 - напряжение двигателей по маршу и курсу
 - Вычисленные параметры положения АНПА для отладки
- Отправляем из модели
 - значение угла курса,
 - скорость поворота по курсу и
 - маршевую скорость



Общий вид модели

- Контур марша разомкнут, контур курса – замкнут
- Учтено взаимовлияние каналов
- Регуляторы **k1** & **k2** реализованы в коде
- Порядок модели – 3-й
- Параметры мат. модели заботливо спрятаны в matlab-скрипте

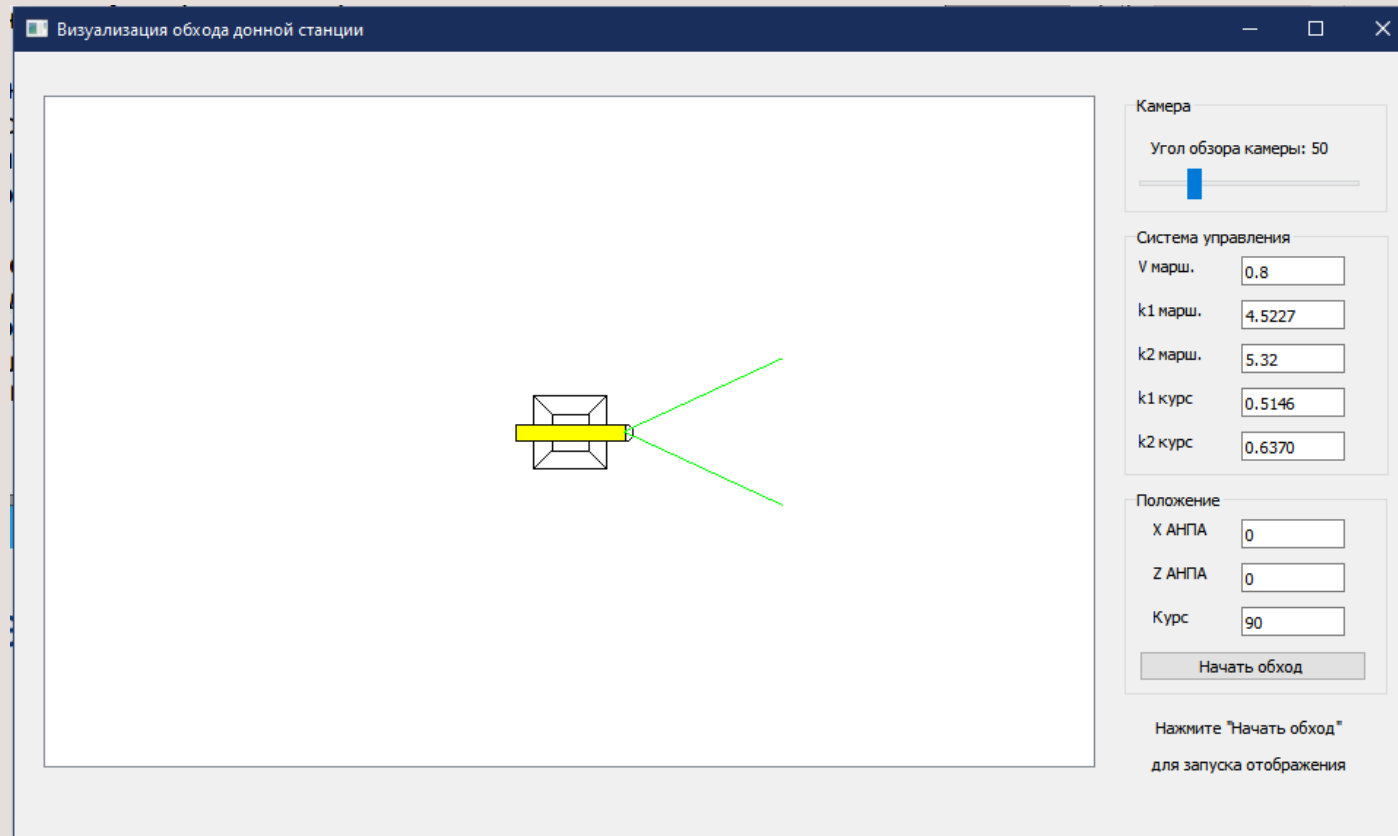


Что делает код

Генерирует точки траектории (заранее)	<code>generate_path_points()</code>
Получает данные по UDP	<code>this->real_yaw = udp.getData().real_yaw;</code>
Вычисляет текущее положение АНПА	<code>calc_position();</code>
Проверяет не достигнута ли текущая траекторная точка	<code>check_distance();</code>
Осуществляет выборку следующей траекторной точки	<code>this->X1 = x_final[dot_number]; this->Z1 = z_final[dot_number]</code>
Определяет направление движения по маршруту	<code>calc_dir();</code>
Рассчитывает желаемый курс	<code>calc_desired_yaw(); constrain_yaw();</code>
Рассчитывает напряжения двигателей с учётом насыщений и взаимовлияния каналов	<много кода>
Отправляет данные по UDP	
Отрисовывает вычисленную траекторию на главном экране	

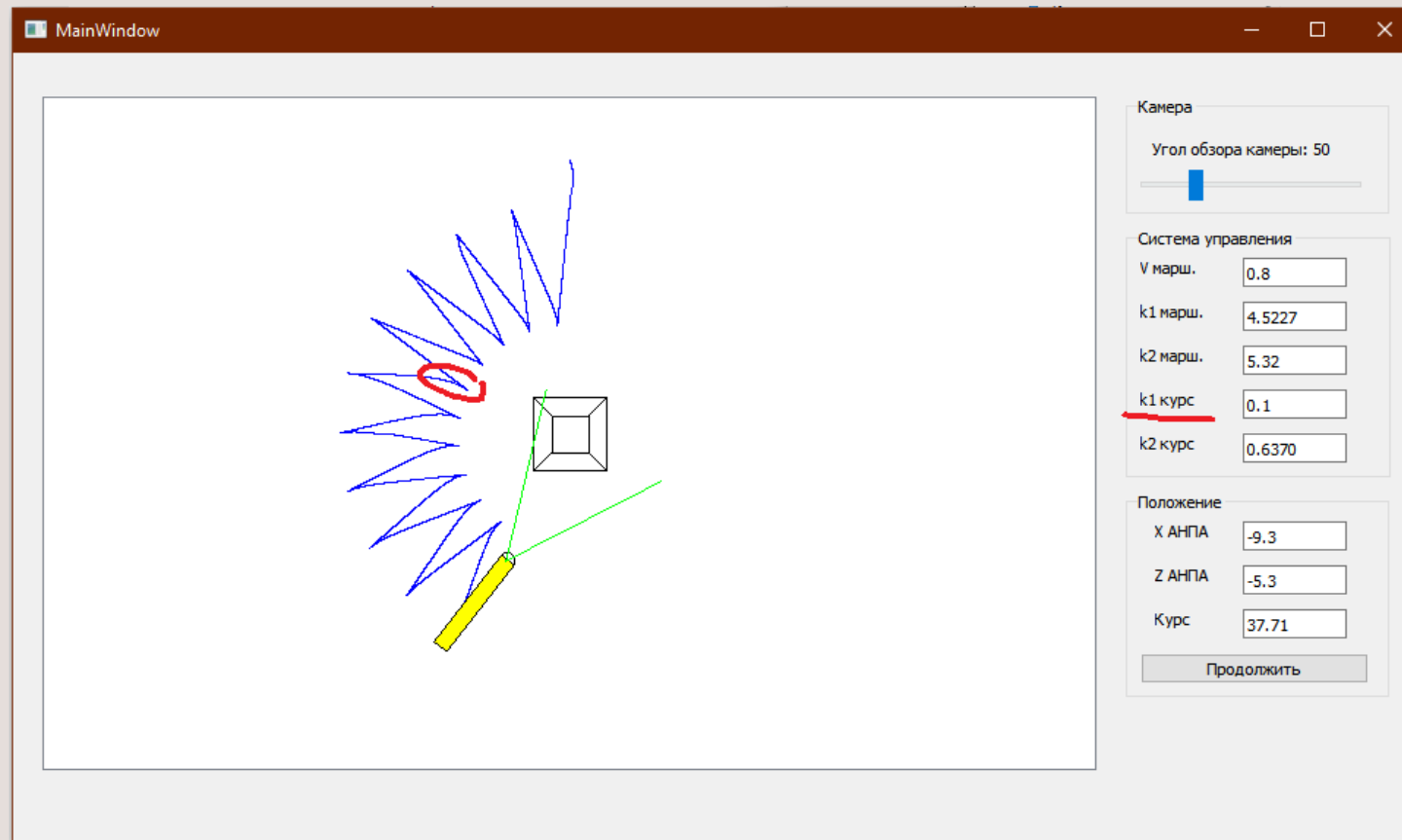
Графический интерфейс

Начальное состояние GUI программы



Процесс визуализации

Был произведён останов моделирования, уменьшение k_1 курса и возобновление расчёта



Было, 1

```

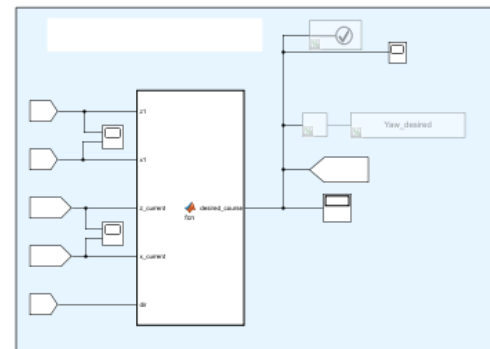
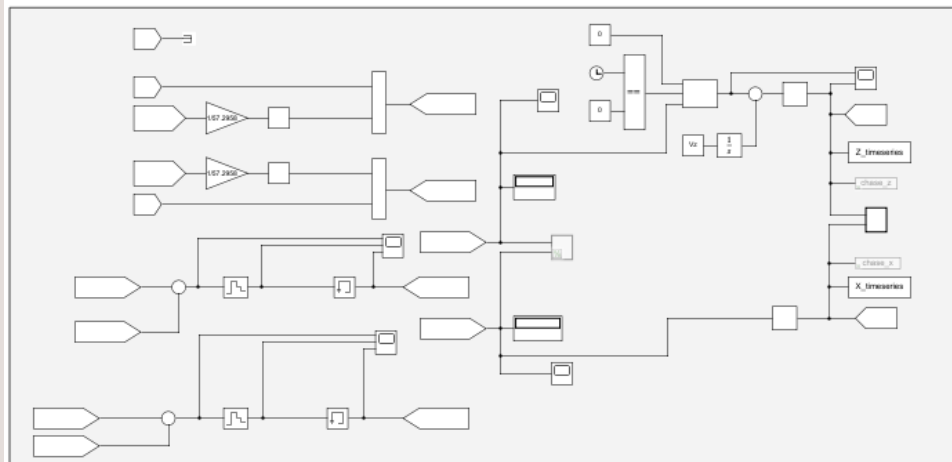
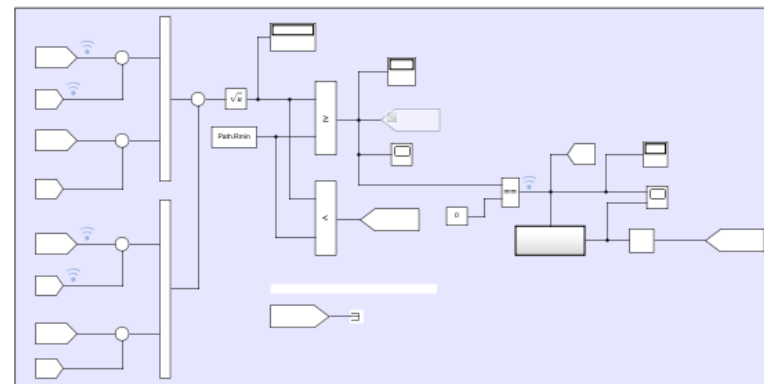
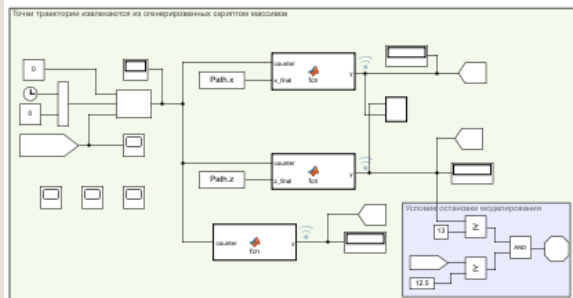
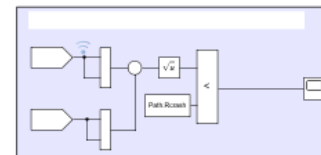
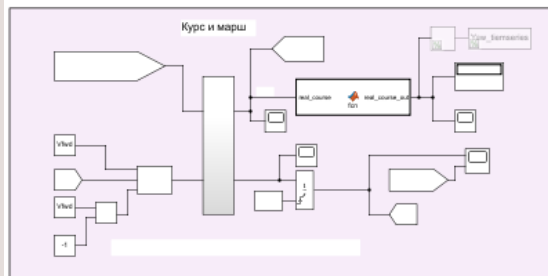
1 function real_course_out = fcn(real_course)
2
3 % Приводим к стандарту
4 if real_course < 0
5     real_course_out = 360 - abs(real_course);
6 elseif real_course >= 360
7     real_course_out = real_course - 360;
8 else
9     real_course_out = real_course;
10 end
11
12 end
    
```

```

1 function deflection_constrained = fcn(real_course, desired_course)
2
3 % Предлагаем повернуться более чем на 180 градусов
4 % Только поворачивать курс не имеет никакого значения,
5 % важно лишь конечное положение.
6 % На должен поворачивать менее чем на 180 градусов
7
8 if abs(desired_course - real_course) > 180
9     % выполняем ограничение
10     if desired_course > 180
11         deflection_constrained = desired_course - 360;
12     elseif desired_course < -180
13         deflection_constrained = 360 + desired_course;
14     else
15         deflection_constrained = desired_course;
16     end
17 % иначе просто ограничиваем
18 else
19     deflection_constrained = desired_course;
20 end
21 end
    
```

```

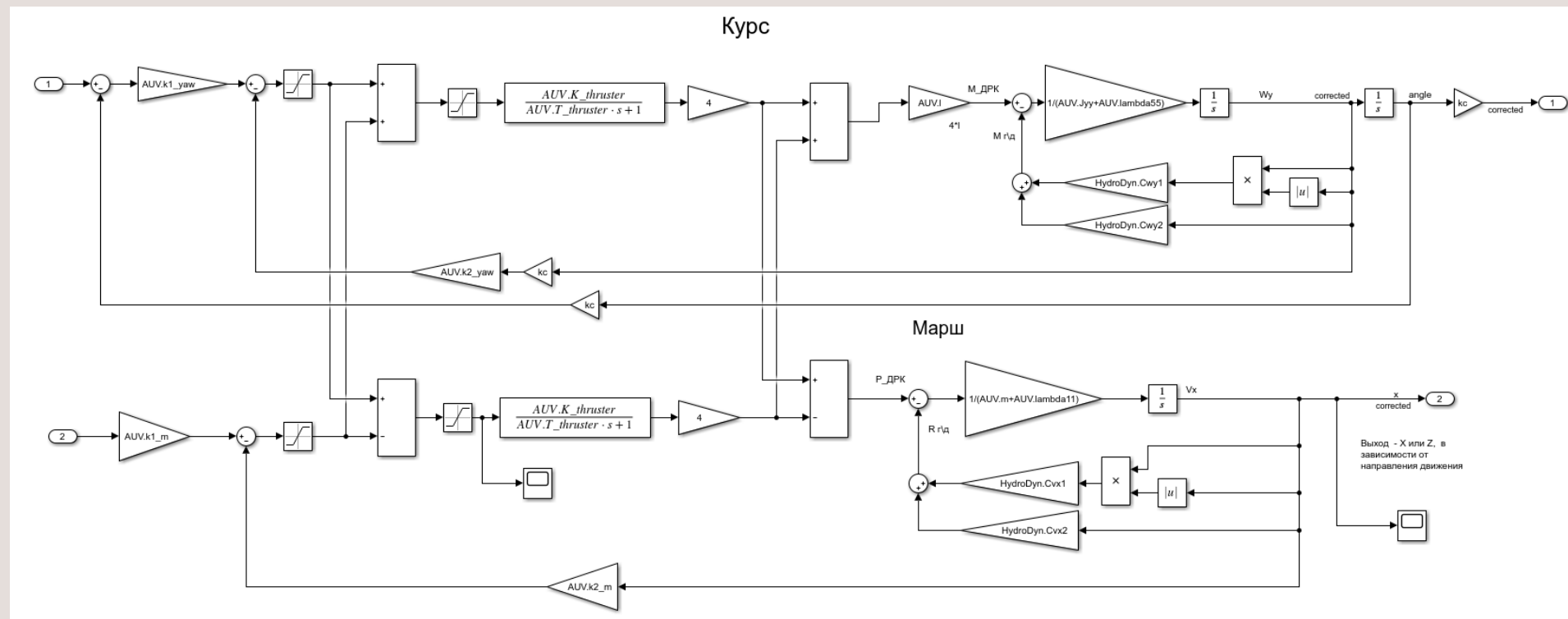
1 function desired_course = fcn(z1, x1, z_current, x_current, dir)
2
3 delta_x = x1 - x_current;
4 delta_z = z1 - z_current;
5
6 %1
7 if (delta_x >= 0) && (delta_z >= 0)
8     desired_course = 57.2958*atan(delta_z / delta_x);
9 %2
10 elseif (delta_x >= 0) && (delta_z < 0)
11     desired_course = 360 - 57.2958*atan(abs(delta_z) / delta_x);
12 %3
13 elseif (delta_x < 0) && (delta_z < 0)
14     desired_course = 180 + 57.2958*atan(abs(delta_z) / abs(delta_x));
15 %4
16 elseif (delta_x < 0) && (delta_z >= 0)
17     desired_course = 180 - 57.2958*atan(abs(delta_z) / abs(delta_x));
18
19 else
20     desired_course = -1;
21 end
22
23
24 % При движении назад отнимаем от 360-и
25 if dir == -1
26     desired_course = desired_course - 180;
27 end
28
29 end
    
```



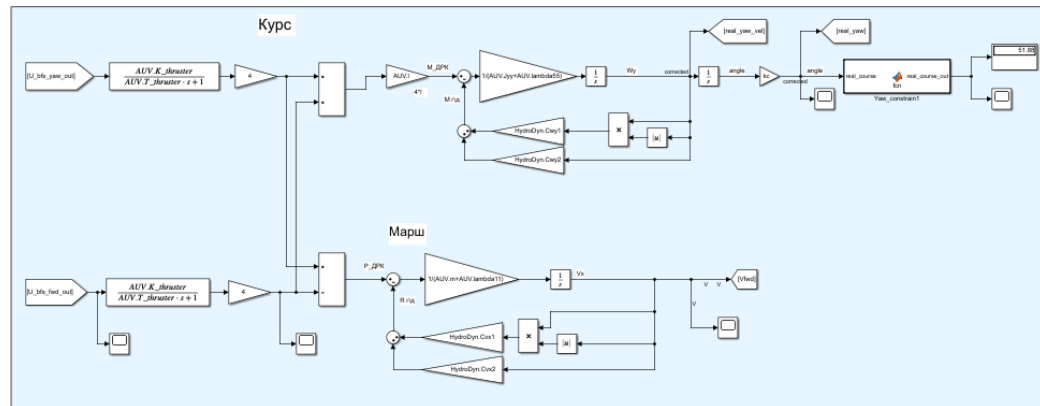
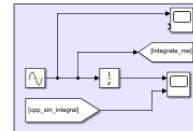
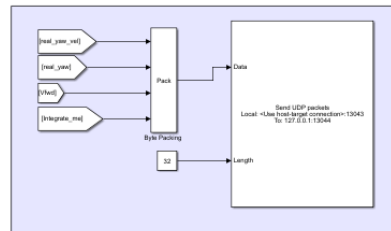
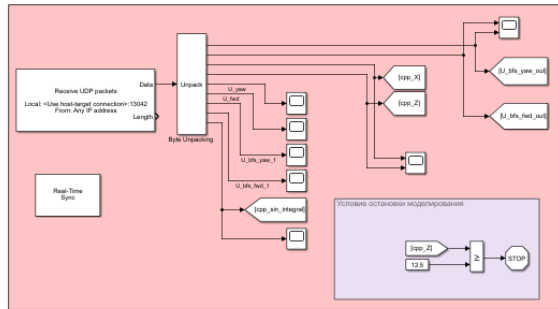
```

22 z_outer = zeros(1, int32(Path.N_points));
23 x_outer = zeros(1, int32(Path.N_points));
24 z_inner = zeros(1, int32(Path.N_points));
25 x_inner = zeros(1, int32(Path.N_points));
26
27 for i=1:Path.N_points
28     %disp( class(i) ); disp( i );
29
30     [p1_z, p1_x] = pol2cart(deg2rad(Path.angle), Path.D_mass_center_max);
31     [p2_z, p2_x] = pol2cart(deg2rad(Path.angle + Path.angular_delta/2), Path.D_mass_center_max);
32
33     %disp(p1_z); disp(p1_x);
34
35     z_outer(int32(i)) = p1_z;
36     x_outer(int32(i)) = p1_x;
37
38     z_inner(int32(i)) = p2_z;
39     x_inner(int32(i)) = p2_x;
40
41     Path.angle = Path.angle + Path.angular_delta;
42
43 end
44
45 % Инициализируем нулями массивы для точек итоговой траектории
46 Path.z = zeros(1, int32(Path.N_points*2));
47 Path.x = zeros(1, int32(Path.N_points*2));
48 Path.direction = zeros(1, int32(Path.N_points*2));
49
50 j = 1;
51 for i=1:Path.N_points
52     Path.z(j) = z_outer(i);
53     Path.x(j) = x_outer(i);
54     Path.direction(j) = 1;
55     j = j + 1;
56
57     Path.z(j) = z_inner(i);
58     Path.x(j) = x_inner(i);
59     Path.direction(j) = -1;
60     j = j + 1;
61 end
62
63 clear p1_z p1_x p2_z p2_x i j z_inner x_inner z_outer x_outer;
    
```


Было, 2



Стало



```

182 void SU_ROV::tick()
183 {
184     // Получаем данные
185     this->real_yaw_vel = udp.getData().real_yaw_vel;
186     this->real_yaw = udp.getData().real_yaw;
187     this->real_V_fwd = udp.getData().real_V;
188
189     this->sin_curr = udp.getData().real_sin;
190     qDebug() << "sin_curr: " << sin_curr << " prev:" << sin_prev << " sin_summ: " << sin_summ << "\n";
191     // Вычисляем
192
193     //double temp =
194     this->sin_summ += ( (sin_prev + sin_curr)/2 * ( (double)timer_period / 1000) );
195     this->sin_prev = sin_curr;
196
197     calc_position();
198     check_distance();
199
200     assert(dot_number < x_final.size());
201
202     this->X1 = x_final[dot_number];
203     this->Z1 = z_final[dot_number];
204
205     calc_dir();
206     check_end_simulation();
207     calc_desired_yaw();
208     constrain_yaw();
209
210     this->U_fwd = this->dir * this->modelParams.Vfwd * this->k1_m;
211     this->U_yaw = (this->deflection_yaw_constrained - this->real_yaw) * this->k1_yaw;
212
213     this->U_bfs_yaw_1 = saturation_block(this->U_yaw - this->k2_yaw * qRadiansToDegrees(this->real_yaw_vel));
214     this->U_bfs_fwd_1 = saturation_block(U_fwd - this->k2_m * this->real_V_fwd);
215
216     this->U_bfs_yaw_out = saturation_block(this->U_bfs_yaw_1 + U_bfs_fwd_1, 12, -12);
217     this->U_bfs_fwd_out = saturation_block(this->U_bfs_yaw_1 - U_bfs_fwd_1, 12, -12);
218
219     emit sendComputedCoords(X_current, Z_current, real_yaw); // to Trajectory class
220
221     udp.send(this->U_bfs_yaw_out, U_bfs_fwd_out, this->X_current, this->Z_current,
222             this->U_yaw,
223             this->U_fwd,
224             this->U_bfs_yaw_1,
225             this->U_bfs_fwd_1,
226             this->sin_summ);
227 }
228

```

Возникшие проблемы

```
< > std_abs.h _x
59 #ifdef _GLIBCXX_USE_LONG_LONG
60     inline long long
61     abs(long long __x) { return __builtin_llabs (__x); }
62 #endif
63
64 // _GLIBCXX_RESOLVE_LIB_DEFECTS
65 // 2192. Validity and return type of std::abs(0u) is unclear
66 // 2294. <cstdlib> should declare abs(double)
67
68 #ifndef __CORRECT_ISO_CPP_MATH_H_PROTO
69     inline _GLIBCXX_CONSTEXPR double
70     abs(double __x)
71     { return __builtin_fabs(__x); }
72
73     inline _GLIBCXX_CONSTEXPR float
74     abs(float __x)
75     { return __builtin_fabsf(__x); }
76
77     inline _GLIBCXX_CONSTEXPR long double
78     abs(long double __x)
79     { return __builtin_fabsl(__x); }
80 #endif
81
82 #if defined(__GLIBCXX_TYPE_INT_N_0)
83     inline _GLIBCXX_CONSTEXPR __GLIBCXX_TYPE_INT_N_0
84     abs(__GLIBCXX_TYPE_INT_N_0 __x) { return __x >= 0 ? __x : -__x; }
85 #endif
86 #if defined(__GLIBCXX_TYPE_INT_N_1)
87     inline _GLIBCXX_CONSTEXPR __GLIBCXX_TYPE_INT_N_1
88     abs(__GLIBCXX_TYPE_INT_N_1 __x) { return __x >= 0 ? __x : -__x; }
89 #endif
90 #if defined(__GLIBCXX_TYPE_INT_N_2)
91     inline _GLIBCXX_CONSTEXPR __GLIBCXX_TYPE_INT_N_2
92     abs(__GLIBCXX_TYPE_INT_N_2 __x) { return __x >= 0 ? __x : -__x; }
93 #endif
94 #if defined(__GLIBCXX_TYPE_INT_N_3)
95     inline _GLIBCXX_CONSTEXPR __GLIBCXX_TYPE_INT_N_3
96     abs(__GLIBCXX_TYPE_INT_N_3 __x) { return __x >= 0 ? __x : -__x; }
97 #endif
98
99 #if !defined(__STRICT_ANSI__) && defined(_GLIBCXX_USE_FLOAT128)
100     inline _GLIBCXX_CONSTEXPR
101     __float128
102     abs(__float128 __x)
103     { return __x < 0 ? -__x : __x; }
104 #endif
105
```

Итого

- Обмен по UDP
- Регуляторы k_1 & k_2 марша и курса в коде
- Контур курса замкнут
- Реализовано следование заданной траектории: выборка путевой точки, расчёт желаемого угла курса, определение выхода в точку
- Формирование управляющих сигналов в коде
- Реализована система счисления пути
- GUI позволяет изменить скорость маршевого движения и ~~скорректировать~~ сломать регуляторы СУ
- Сигналы и слоты соединены

Пути улучшения ПО

- Добавить поля для задания IP-адреса и номера порта

С прошлого семестра:

- Добавить оси с подписями и отсчётами и координатную сетку.
- В случае перемещения в пространстве использовать модуль Qt Data Visualization.
- Встроить текстуру морского дна в виде фона.
- Добавить возможность масштабирования и перемещения по получившейся карте.
- Возможность запуска моделирования напрямую из C++-кода (сделано)

Спасибо за внимание)