



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»

ИНСТИТУТ МИРОВОГО ОКЕАНА
Департамент автоматики и робототехники

Данилов Даниил Дмитриевич

КУРСОВАЯ РАБОТА

Разработка информационно-управляющей системы для динамического
объекта
Вариант 4

Студент группы

Б11119-15.03.06мхрб

Данилов Д.Д. _____
(подпись)

Руководитель

д.т.н., доцент

Юхимец Д.А. _____
(подпись)

Регистрационный № _____

Оценка _____

подпись И.О.Фамилия

подпись И.О.Фамилия

«_____» _____ 20__г.

«_____» _____ 20__г.

г. Владивосток
2022

СОДЕРЖАНИЕ

Цель работы	6
Задание на курсовую работу	7
Техническое задание	8
Содержание	10
1 Введение	12
1.1 Наименование	12
1.2 Область применения	12
2 Основания для разработки	12
3 Назначение разработки	13
3.1 Функциональное назначение	13
3.2 Эксплуатационное назначение	13
4 Требования к программе	13
4.1 Требования к функциональным характеристикам	13
4.1.1 Требование к составу выполняемых функций	13
4.1.2 Требования к организации входных и выходных данных ..	14
4.1.3 Требования к временным характеристикам	14
4.2 Требования к надежности	15
4.2.1 Требования к обеспечению надежного (устойчивого)	
функционирования программы	15
4.2.2 Время восстановления после отказа	15
4.2.3 Отказы из-за некорректных действий оператора	16
4.3 Условия эксплуатации	16
4.3.1 Климатические условия эксплуатации	16
4.3.2 Требование к видам обслуживания	16
4.3.3 Требование к численности и квалификации персонала	16
4.4 Требования к составу и параметрам технических средств	17
4.5 Требования к информационной и программной совместимости	17

4.5.1 Требования к информационным структурам и методам решения	17
4.5.2 Требования к исходным кодам и языкам программирования	17
4.5.3 Требование к защите информации и программ	18
4.6 Требования к маркировке и упаковке	18
4.7 Требования к транспортированию и хранению	18
4.8 Специальные требования	18
5 Требования к программной документации	18
6 Техничко-экономические показатели	19
7 Стадии и этапы разработки	19
7.1 Стадии разработки	19
7.2 Этапы разработки	19
7.3 Содержание работ по этапам	19
8 Порядок контроля и приемки	21
8.1 Виды испытаний	21
8.2 Общие требования к приемке работы	21
Описание программы	22
Аннотация	24
1. Общие сведения	26
1.1 Обозначение и наименование программы	26
1.2 Программное обеспечение, необходимое для функционирования программы	26
1.3 Языки программирования, на которых написана программа	26
2. Функциональное назначение	26
2.1 Краткое описание функций	26
2.2 Состав функций	27
2.3 Сведения о функциональных ограничениях на применение	27
3. Описание логической структуры	28
3.1 Структура программы	28

3.2 Алгоритм программы	Ошибка! Закладка не определена.
4. Используемые технические средства	28
5. Вызов и загрузка	39
6. Входные данные.....	40
7. Выходные данные	41
Лист регистрации изменений.....	42
Руководство системного программиста	43
Аннотация	45
1. Общие сведения о программе.....	47
1.1 Назначение программы	47
1.2 Функции программы.....	47
1.3 Сведения о технических и программных средствах	47
2. Структура программы	48
2.1 Сведения о структуре программы.....	48
2.2 Взаимодействие с другими программами	48
3. Настройка на состав программных средств	48
4. Проверка программы	49
Лист регистрации изменений.....	50
Руководство оператора	51
1. Общие сведения о программе.....	55
1.1 Назначение программы	55
1.2 Функции программы.....	55
2. Условия выполнения программы	55
2.1 Требования к аппаратным средствам	55
2.2 Требования к программным средствам	56
2.3 Требования к оператору	56
3. Выполнение программы.....	56
3.1 Запуск программы.....	56
3.2 Начало работы	67
3.3 Настройка начальных параметров	67

3.4 Запуск работы системы	69
Лист регистрации изменений.....	70
Текст программы.....	71
1. Листинг модуля GUI.....	73
2. Листинг модуля ControlApp.....	75
3. Листинг модуля ControlSys	78
4. Листинг модуля Camera	81
5. Листинг модуля NavigationSys	82
6. Листинг модуля PID	84
Лист регистрации изменений.....	88
Результаты работы	89
Испытание 1.....	90
Испытание 2.....	91
Испытание 3.....	92
Испытание 4.....	93
Вывод.....	95

Цель работы

Целью курсовой работы является разработка информационно-управляющей системы (ИУС) для динамического объекта, разработка архитектуры и программной реализации алгоритмов управления. Разработанная ИУС должна реализовывать движение в заданную точку с заданной скоростью, движение в точку по прямой с заданной скоростью, движение по криволинейной траектории, задаваемой сплайном третьего порядка с заданной скоростью. Система управления должна содержать ПИД-регуляторы, обеспечивающие роботу выход в заданную точку пространства. Движение динамического объекта в штатном режиме происходит носом вперед. Миссия робота описывает траекторию его движения, проходящую через последовательность базовых точек. На каждом участке траектории робот может двигаться в одном из двух режимов движения, описанных выше. В процессе движения робот должен иметь возможность двигаться на заданной высоте от поверхности. Робот должен выполнять одну из следующих функций: поиск, картографирование. ИУС робота должна обладать интерфейсом пользователя, позволяющим задать миссию робота и отобразить процесс ее выполнения. Для объектов управления моделируются силы вязкого трения, для подводного робота гидростатические силы.

Задание на курсовую работу

В таблице 1 приведены исходные данные для выполнения курсовой работы.

Таблица 1 – Исходные данные

Вариант	Объект управления	Архитектура ИУС	Функция
5	Подводный робот	Объектно-ориентированная	Поиск

Из представленной таблицы видно, что объектом управления, для которого должна быть разработана ИУС, является подводный робот с функцией поиска объекта с заданным цветом, расположенного в заранее неизвестном месте карты. Сенсор для поиска – бортовая видеокамера. Результат поиска – координаты объекта. Робот обладает комплексом движителей, формирующие тягу для движения.

УТВЕРЖДАЮ

Д.т.н., доцент департамента
автоматики и робототехники

_____ Юхимец Д.А.

«_____» _____ 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

Информационно-управляющая система «AUV - 2022»

Техническое задание

ЛИСТ УТВЕРЖДЕНИЯ

RU.02067942.0005–01 ТЗ 13-01 ЛУ

Подпись и	
Инв. №	
Взам.	
Подпись и	
Инв. №	

Руководитель разработки

Д.т.н., доцент

_____ Юхимец Д.А.

«_____» _____ 2022

Исполнитель

Студент группы

Б11119-15.03.06мхрб

_____ Данилов Д.Д

«_____» _____ 2022

2022

УТВЕРЖДЕН

RU.02067942.0005–01 ТЗ 13-01

Информационно-управляющая система «AUV - 2022»

Техническое задание

RU.02067942.0005–01 ТЗ 13-01

Листов 10

Инв. №	Подпись и	Взам.	Инв. №	Подпись и

2022

СОДЕРЖАНИЕ

1 Введение.....	12
1.1 Наименование.....	12
1.2 Область применения.....	12
2 Основания для разработки	12
3 Назначение разработки.....	13
3.1 Функциональное назначение	13
3.2 Эксплуатационное назначение	13
4 Требования к программе	13
4.1 Требования к функциональным характеристикам	13
4.1.1 Требование к составу выполняемых функций.....	13
4.1.2 Требования к организации входных и выходных данных ..	14
4.1.3 Требования к временным характеристикам.....	14
4.2 Требования к надежности	15
4.2.1 Требования к обеспечению надежного (устойчивого)	
функционирования программы	15
4.2.2 Время восстановления после отказа	15
4.2.3 Отказы из-за некорректных действий оператора	16
4.3 Условия эксплуатации.....	16
4.3.1 Климатические условия эксплуатации	16
4.3.2 Требование к видам обслуживания.....	16
4.3.3 Требование к численности и квалификации персонала.....	16
4.4 Требования к составу и параметрам технических средств.....	17
4.5 Требования к информационной и программной совместимости	17
4.5.1 Требования к информационным структурам и методам	
решения	17
4.5.2 Требования к исходным кодам и языкам программирования	
.....	17
4.5.3 Требование к защите информации и программ	18
4.6 Требования к маркировке и упаковке	18

4.7 Требования к транспортированию и хранению	18
4.8 Специальные требования	18
5 Требования к программной документации	18
6 Техничко-экономические показатели	19
7 Стадии и этапы разработки	19
7.1 Стадии разработки	19
7.2 Этапы разработки	19
7.3 Содержание работ по этапам	19
8 Порядок контроля и приемки	21
8.1 Виды испытаний	21
8.2 Общие требования к приемке работы	21

1 ВВЕДЕНИЕ

1.1 Наименование

Информационно-управляющая система (ИУС) «AUV - 2022».

1.2 Область применения

ИУС предназначена для применения в подводных аппаратах с функцией поиска, имеющих движительный комплекс, формирующий тягу для управления движением робота. Сенсор для поиска – бортовая видеокамера.

2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Документы, на основании которых ведется разработка: Д.А. Юхимец – учебное электронное издание «Разработка информационно-управляющих систем для робототехнических объектов», ГОСТ 19.201-78, ГОСТ 19.402-78, ГОСТ 19.503-78.

Основанием для разработки является задание на курсовую работу по дисциплине «Автоматизированные информационно-управляющие системы» от 22.02.2022 года;

Наименование темы разработки – «Разработка информационно-управляющей системы для динамического объекта». Условное обозначение разработки – «AUV - 2022».

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

3.1 Функциональное назначение

Функциональным назначением является предоставление пользователю (человеку-оператору) возможности осуществления навигации робота по заданной пользователем траектории и поиск объектов, расположенных в заранее неизвестном месте на карте.

3.2 Эксплуатационное назначение

ИУС предназначена для эксплуатации на стационарном компьютере или ноутбуке.

4 ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1 Требования к функциональным характеристикам

4.1.1 Требование к составу выполняемых функций

Информационно-управляющая система должна обеспечивать возможность выполнения перечисленных ниже функций:

- 1) Движение робота в заданную точку с заданной скоростью (от 0.2 м/с до 1.5 м/с);
- 2) Движение робота по прямой в заданную точку с заданной скоростью (от 0.2 м/с до 1.5 м/с);
- 3) Движение робота по криволинейной траектории, описанной сплайном 3-го порядка, с заданной скоростью (от 0.2 м/с до 1.5 м/с);
- 4) Движение робота на заданной высоте от поверхности (от 0.6 до 5 м);

- 5) Вывод данных о движении (скорость, положение) в пользовательский интерфейс, также вывод координат найденных объектов
- 6) Обнаружение и обход роботом препятствий;
- 7) Задание пользователем миссии робота через интерфейс: по последовательности точек с минимальной дистанцией друг от друга 2 метра;
- 8) Выбор пользователем одного из трёх режимов движения, описанных выше;
- 9) Выбор пользователем скорости движения робота, расстояния от поверхности дна;

4.1.2 Требования к организации входных и выходных данных

Входными данными являются сигнала с датчиков робота, имеющих тип float, установленные пользователем значения параметров робота и миссии движения. Выходными данными являются управляющие сигналы для двигателей робота, а также данные в терминале программы о координатах найденных объектов.

4.1.3 Требования к временным характеристикам

Требования к временным характеристикам программы не предъявляются.

4.2 Требования к надежности

4.2.1 Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- 1) организацией бесперебойного питания технических средств;
- 2) использованием лицензионного программного обеспечения;
- 3) регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
- 4) регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов;
- 5) соблюдение условий эксплуатации, указанным в настоящем техническом задании.

4.2.2 Время восстановления после отказа

Время восстановления после отказа системы в случаях перебоев с электропитанием и по иным причинам, не повлекшим к критическим (каким-либо влияющим на работу) повреждениям систем, составляет не более 5-ти минут.

4.2.3 Отказы из-за некорректных действий оператора

В случае отказа системы из-за некорректных действий оператора требуется перезапуск всей системы.

4.3 Условия эксплуатации

4.3.1 Климатические условия эксплуатации

Климатические условия эксплуатации должны соответствовать условиям эксплуатации ЭВМ, на которой данный продукт установлен. ЭВМ должен обладать высокой степенью защищенности от влаги; низких температур. Нормальная функциональность системы будет при следующих условиях:

- 1) температура воздуха - $-10...+30$ градусов С;
- 2) относительная влажность воздуха - 11 – 81%;

4.3.2 Требование к видам обслуживания

Программа не требует проведения каких-либо видов обслуживания

4.3.3 Требование к численности и квалификации персонала

Для корректной работы программы требуется один оператор. Конечный пользователь программы (оператор) должен обладать практическими навыками использования персонального компьютера, а также перед работой с программой должен ознакомиться с соответствующим руководством оператора. В задачи оператора входит: установка параметров работы

программы, запуск и отключение программы. Права административного доступа к ПО оператору не предоставляется.

4.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя:

- 1) Процессор Core i5 с максимальной тактовой частотой от 3.6 ГГц;
- 2) Оперативную память объемом от 16 Гб;
- 3) Твердотельный накопитель от 100 Гб;
- 4) Свободное место на твердотельном накопителе - от 50 Гб;
- 5) Видеокартой с объемом видео памяти от 4 Гб и тактовой частотой от 1 МГц;
- 6) Операционную систему Windows 10;
- 7) Устройства управления и ввода: клавиатура, мышь.
- 8) Устройство вывода: монитор с разрешением от 1920x1080

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к информационным структурам и методам решения

Требований к информационной структуре входных и выходных данных, а также к методам решения не предъявляются.

4.5.2 Требования к исходным кодам и языкам программирования

Исходные коды программы, должны быть реализованы на выбор на следующих языках программирования: Python 3.9. Требований к средам разработки не предоставляется.

4.5.3 Требование к защите информации и программ

Требования к защите информации и программ не предъявляются.

4.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

4.7 Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

4.8 Специальные требования

Специальные требования не предъявляются.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать в себя:

- 1) техническое задание;
- 2) руководство системного программиста;
- 3) руководство оператора;
- 4) описание программы.

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Ориентировочная экономическая эффективность и предполагаемая годовая потребность не рассчитываются.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

7.1 Стадии разработки

Разработка должна быть проведена в три стадии:

- 1) разработка технического задания;
- 2) рабочее проектирование;
- 3) внедрение.

7.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

- 1) разработка программы;
- 2) разработка программной документации;
- 3) испытания программы.

7.3 Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- 1) постановка задачи;
- 2) определение и уточнение требований к техническим средствам;
- 3) определение требований к программе;
- 4) определение стадий, этапов и сроков разработки программы и документации на неё;
- 5) выбор языков программирования;
- 6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (написанию) и отладке программы. На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- 1) разработка, согласование и утверждение программы и методики испытаний;
- 2) проведение приемо-сдаточных испытаний;
- 3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

8 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1 Виды испытаний

Приемо-сдаточные испытания должны проводиться на объекте Заказчика в оговорённые сроки.

Приемо-сдаточные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком программы и методик испытаний.

Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в протоколе проведения испытаний.

8.2 Общие требования к приемке работы

На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывают Акт приемки-сдачи программы в эксплуатацию

УТВЕРЖДАЮ

Д.т.н., доцент департамента
автоматики и робототехники

_____ Юхимец Д.А.

«_____» _____ 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

Информационно-управляющая система «AUV - 2022»

Описание программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.02067942.0005–01 ОП 13-01 ЛУ

Руководитель разработки
Д.т.н., доцент

_____ Юхимец Д.А.

«_____» _____ 2022

Исполнитель

Студент группы

Б11119-15.03.06мхрб

_____ Данилов Д.Д.

«_____» _____ 2022

Подпись и	
Инв. №	
Взам.	
Подпись и	
Инв. №	

2022

УТВЕРЖДЕН

RU.02067942.0005–01 ОП 13-01

Информационно-управляющая система «AUV - 2022»

Описание программы

RU.02067942.0005–01 ОП 13-01

Листов 29

Инв. №	Подпись и	Взам.	Инв. №	Подпись и

2022

АННОТАЦИЯ

В документе «описание программы» приведено описание ИУС «AUV - 2022», предназначенной для применения в подводных аппаратах с функцией поиска объектов: функциональное назначение, описание логики программы и условия её применения. Документ был оформлен согласно требованиям ЕСПД ГОСТ 19.402-78. Разработка программного обеспечения велась на основании технического задания под регистрационным номером RU.02067942.0005–01 ТЗ 13-01.

СОДЕРЖАНИЕ

1. Общие сведения	26
1.1 Обозначение и наименование программы	26
1.2 Программное обеспечение, необходимое для функционирования программы	26
1.3 Языки программирования, на которых написана программа	26
2. Функциональное назначение	26
2.1 Краткое описание функций.....	26
2.2 Состав функций.....	27
2.3 Сведения о функциональных ограничениях на применение	27
3. Описание логической структуры	28
3.1 Структура программы	28
3.2 Алгоритм программы	Ошибка! Закладка не определена.
4. Используемые технические средства.....	28
5. Вызов и загрузка	39
6. Входные данные	40
7. Выходные данные	41
Лист регистрации изменений.....	42

1. ОБЩИЕ СВЕДЕНИЯ

1.1 Обозначение и наименование программы

Наименование программы: «AUV - 2022». Программа предназначена для применения в подводных аппаратах с функцией поиска объектов

1.2 Программное обеспечение, необходимое для функционирования программы

Данное программное обеспечение может использоваться на следующих операционных системах: Microsoft Windows 10. Дополнительное ПО не используется.

1.3 Языки программирования, на которых написана программа

Программа реализована на языке программирования Python 3.9.4 с использованием среды Atom.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1 Краткое описание функций

Функцией данной программы является осуществление навигации робота по заданной пользователем траектории и поиск объектов, расположенных в неизвестном месте на карте, с выводом координат этих объектов.

2.2 Состав функций

Программа имеет возможность осуществлять движение робота с обходом препятствий в заданную точку с заданной скоростью на заданной высоте от поверхности; вывод координат найденных объектов в пользовательское приложение. Пользователь, используя интерфейс, имеет следующие возможности: отслеживать выполнение миссии; задавать параметры миссии; запуск и отключение робота.

2.3 Сведения о функциональных ограничениях на применение

1) Функциональные ограничения:

- Невозможность выполнения заявленных функций при критическом состоянии модуля системы.

2) Ограничения, накладываемые на область применения:

- Несоответствие условиям эксплуатации технических средств клиентской части.

3. ОПИСАНИЕ ЛОГИЧЕКОЙ СТРУКТУРЫ

3.1 Структура программы

Программа имеет объектно-ориентированную архитектуру. Диаграмма классов, описывающая структуру ИУС, представлена на рисунке 1.

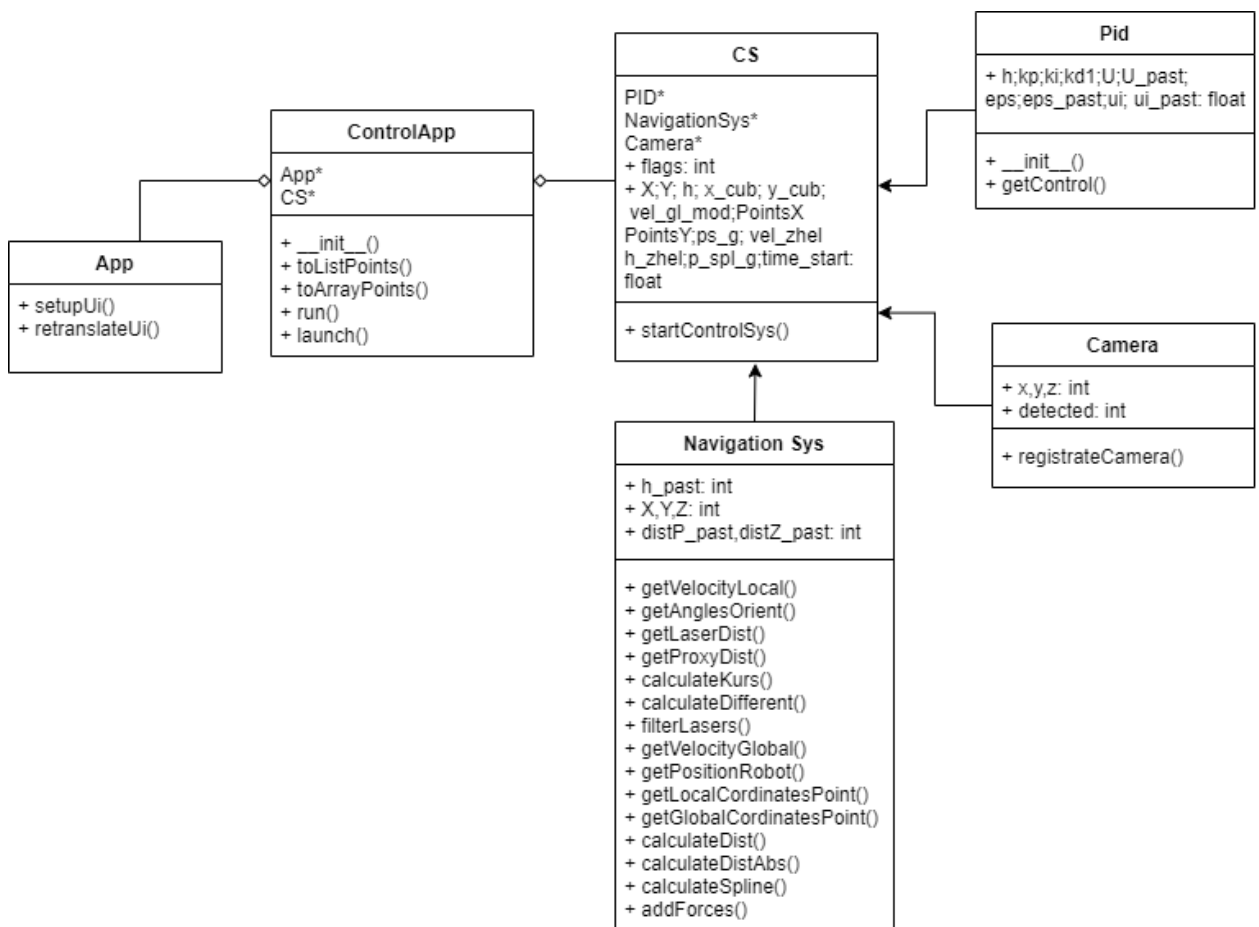


Рисунок 1 – Диаграмма классов.

1) Класс App: данный класс описывает структуру приложения: поля, строки ввода, надписи и т.д. Данный класс был сгенерирован с помощью приложения Qt Designer, где был разработан дизайн программы, при этом

понадобилась библиотека `ruuic5`. Логика обновления и считывания данных была описана в объекте `ControlApp`.

2) Класс `ControlApp`: данный класс содержит в себе объекты `Registration(App.Ui_MainWindow)` и `AupdateData(QThread)`, первый отвечает за запуск приложения, его инициализацию, которую в случае необходимости можно скорректировать (например если в классе, описывающем структуру `App` будет располагаться ненужное или некорректное поле его можно будет удалить или скорректировать при инициализации), также в данном классе если метод `launch()`, который запускает метод `run()` класса `AupdateData(QThread)`, который обновляет данные приложения и работает в отдельном потоке, так как основной поток будет занят окном приложения. Методы класса `AupdateData(QThread)` такие как `toArrayPoints()` и `toListPoints()` переводят из значений `string`, полученных с окна приложения, в массивы и листы соответственно (необходимо для обновления параметров в статическом классе `CS`). Для данного класса необходимы импорты библиотек `PyQt5` и `threading`.

3) Класс `CS`: данный класс описывает логику работы всей системы. В нём хранятся переменные, для трёх видов движений, а также системы условий для корректной работы. Режим движения робота в данном классе зависит от параметра `Mode`, если 0 – движение по прямой с заданной скоростью на заданной высоте от поверхности, 1 – миссия робота по заданным точкам, с заданной скоростью на заданной высоте от поверхности, с целью обнаружения объекта (куба) красного цвета, 2 – движение по сплайну с заданной скоростью на заданной высоте от поверхности, обновление данного режима происходит при в классе `ControlSys` после нажатия кнопки “Start”. Работает данный класс в отдельном потоке и запускается с `ControlApp`. Метод `startControlSys()`

отвечает за запуск модуля CS. В нём содержатся объекты классов PID, Camera, NavigationSys, также данный класс запускает работу отдельного потока камеры.

4) Класс NavigationSys: данный класс содержит в себе методы принятия и отправки данных с CoppeliaSim, также содержит методы для расчёта положения робота, его ориентации, методы для расчёта угла курса и дифферента, метод построения сплайна, фильтр дистанций с лазеров, методы получения координат точки в локальной системе координат, метод расчёта расстояния до точки. Работает от потока объекта CS. Необходимая библиотека для соединения с Coppelia Sim – ZMQ RemoteAPIClient. Опишем основные структуры расчёта:

Данные об углах приходят с Coppelia Sim в связанной системе координат, отправляющим датчиком является датчик угловых скоростей, где реализован метод вычисления углов путём интегрирования по соответствующим формулам:

$$\begin{aligned}\dot{\theta} &= \omega_y; \\ \dot{\psi} &= \frac{\omega_z}{\cos \theta}; \\ \dot{\varphi} &= \omega_x + \tan \theta \omega_z.\end{aligned}$$

Для перевода из одной системы координат в другую необходимо использовать элементарные матрицы поворотов. Каждая из этих элементарных матриц описывает поворот объекта относительно соответствующей оси и выглядят следующим образом:

$$A_{x\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix},$$

$$A_{y\beta} = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix},$$

$$A_{z\gamma} = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

где α, β, γ – углы поворота вокруг соответствующих осей.

Резльтирующая матрица поворота, как отмечалось ранее, представляет собой перемножение элементарных матриц, которая примет вид:

$$A = A_{z\gamma}A_{y\beta}A_{x\alpha}.$$

Так как углы с датчика угловых скоростей приходят в связанной с роботом системе координат, чтобы перевести локальный вектор скорости/точки в абсолютную, необходимо умножить матрицу поворота на данный вектор, а если наоборот перевести из глобального вектора скорости/точки в локальную необходимо умножить транспонированную матрицу на данный вектор.

Положение робота при движении находим с использованием датчика линейных скоростей, при этом первоначально переведем вектор скорости в абсолютную систему координат:

$$X = X_0 + Vt$$

Расчёт координат сплайна производится методом `calculateSpline()` где происходит интерполяция заданной точками функции с помощью сплайна

третьего порядка вида:

$$\begin{aligned} S(x) &= S_i(x) = \\ &= a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \\ &x_{i-1} \leq x \leq x_i \end{aligned}$$

Для определения коэффициентов a_i, b_i, c_i, d_i на всех n элементарных отрезках необходимо получить $4n$ уравнений. Данные уравнения вытекают из условия прохождения кубического сплайна, таких, как прохождения сплайна через заданные точки и условие гладкости второго порядка, то есть:

$$\begin{aligned} S_i(x_{i-1}) &= y_{i-1}; \\ S_i(x_i) &= y_i; \\ S'_i(x_i) &= S'_{i+1}(x_i); \\ S''_i(x_i) &= S''_{i+1}(x_i); \end{aligned}$$

Подставляя полученные выражения, получаем следующие уравнения для определения неизвестных коэффициентов:

$$\begin{aligned} a_i &= y_{i-1}; \\ d_i &= \frac{c_{i+1} - c_i}{3h_i}; \\ b_i &= \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i); \end{aligned}$$

Видно, что коэффициенты b_i и d_i зависят от c_i , составим следующую систему уравнений для коэффициента c_i :

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_ic_{i+1} = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right);$$

При свободном закреплении концов сплайна можно приравнять нулю кривизну линии в точках закрепления. Получаемая таким образом функция называется свободным кубическим сплайном. Отсюда следует, что:

$$\begin{aligned} S'_i(x_0) &= b_1 = k_1, & S'_i(x_n) &= b_{n+1} = k_2; \\ S''_i(x_0) &= 2c_1 = m_1 = 0, & S''_i(x_n) &= 2c_{n+1} = m_2 = 0; \end{aligned}$$

Таким образом можно составить систему линейных уравнений, решаемую с помощью метода обратной матрицы, вида $Hc = F$, где:

$$H = \begin{bmatrix} 2(h_{i-1} + h_i) & h_i & 0 & \dots & 0 \\ h_{i-1} & 2(h_{i-1} + h_i) & h_i & \dots & 0 \\ 0 & h_{i-1} & 2(h_{i-1} + h_i) & \dots & 0 \\ \dots & \dots & \dots & \dots & h_i \\ 0 & 0 & 0 & h_{i-1} & 2(h_{i-1} + h_i) \end{bmatrix};$$

$$c = \begin{bmatrix} c_2 \\ c_3 \\ c_4 \\ \dots \\ c_n \end{bmatrix};$$

$$F_i = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right);$$

5) Класс Camera: данный класс необходим для получения изображения с камеры робота. Данный класс работает на отдельном потоке и отдельном Remote Api, так как ZMQ Remote Api плохо передаёт данные в большом объёме (при больших расширениях камеры). Изображение с Coppelia Sim приходит в формате BGR, что необходимо преобразовать в RGB, а также в перевёрнутом формате, который нужно отзеркалить. Для обработки изображения используется HSV фильтр, предварительно настроенный ползунками под поиск красного цвета, который он переводит в белый, остальные же цвета становятся чёрными. Используя функцию cv2.moments(thresh1, 1) библиотеки OpenCV возможно получить положение объекта на картинке, с помощью чего рассчитать положение кубика в связанной системе координат используя высоту над поверхностью.

6) Класс PID: данный класс необходим для расчёта управляющих сигналов, на основе входного и желаемого значений. Инициализирующая функция __init__ устанавливает коэффициенты в зависимости от регулятора, сами же коэффициенты настраивались обнулением k_i, k_d изменением k_p и последующим анализом увеличения k_i, k_d для каждого регулятора индивидуально, с учётом того, что k_p - пропорциональная составляющая пытается компенсировать разницу заданного значения и фактического, k_i - устраняет статическую ошибку, но добавляет колебательность системе, k_d - противодействует предполагаемым отклонениям, вызванными возмущениями системы или запаздыванием. Расчёт управляющего сигнала в дискретной форме происходит по формуле:

$$U(kh) = P \cdot e(kh) + u_i((k-1)h) + I \cdot h \cdot e(kh) + D \cdot \frac{e(kh) - e((k-1)h)}{h},$$

где P, I, D – коэффициенты регулятора, k – номер дискретного шага, e – значение ошибки для регулирования.

3.2 Структура программы

В процессе работы ИУС может находиться в следующих состояниях, представленных на рисунке 5:

Первое состояние – INIT. Состояние, в котором система инициализирует все параметры, классы.

В случае успешного подключения к Coppelia Sim система переходит в состояние Ready - состояние после инициализации, в нем система ожидает команды для начала работы (главное меню программы). В случае безуспешного подключения, система переходит в состояние FAIL, где программа выводит сообщение об ошибке подключения, затем выход из системы.

После получения команды start, система переходит в состояние Start, в котором происходит выбор типа работы для перехода в состояние WORK. Это состояние является основным для ИУС, и в нем происходят все действия и расчеты, связанные с работой подводного робота, т.е. состояние, когда система выполняет работу по управлению роботом. В данном состоянии считываются сигналы датчиков, вычисляется местоположение препятствий, осуществляется формирование выходных управляющих сигналов для движителей.

В состоянии WORK система может переходить в состояния:

- 1) MOVE 1. Состояние, во время которого происходит движение робота по прямой в точку с заданной скоростью.
- 2) MOVE 2. Состояние, отвечающее за миссию робота.

3) MOVE 3. Состояние, в течение которого происходит движение робота в точку по сплайну с заданной скоростью.

Работа прекращается в случае неисправности системы, наличия критической ошибки.

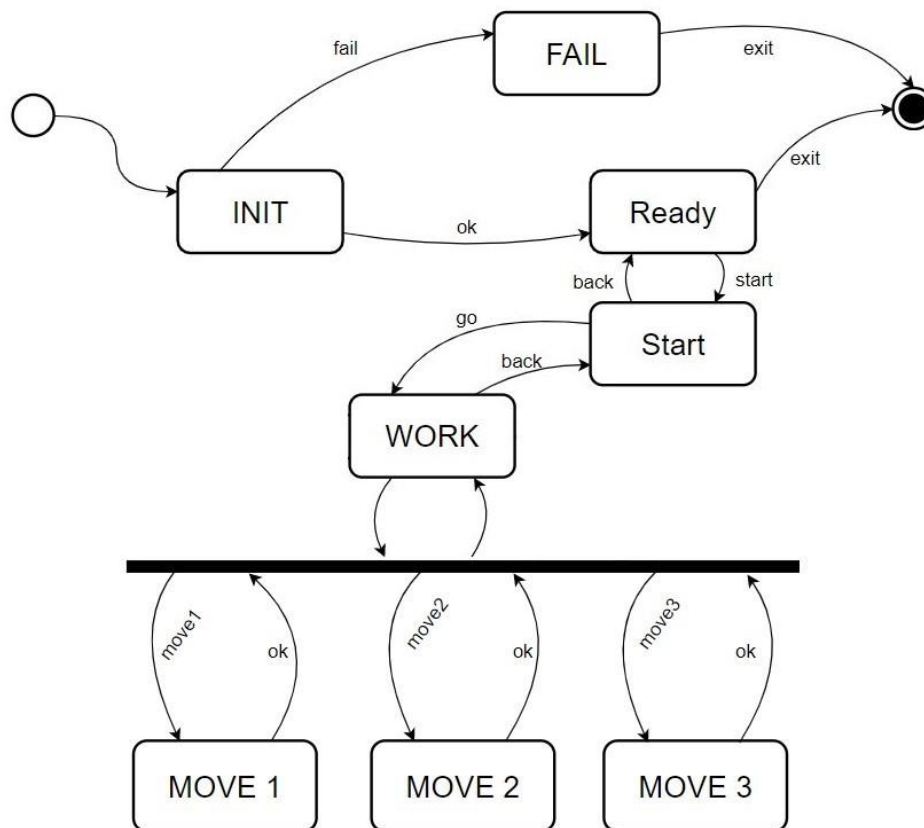


Рисунок 2 – Диаграмма состояний ИУС

Диаграмма последовательности состояния WORK представлена на рисунке 3. Как видно, из рисунка работа программы происходит поэтапно. Для начала, пользователь, вводя значения параметров робота и тип движения, отправляет это на объект класса Control_System. Следующим этапом идёт получения данных о положении робота объектами классов камеры и системы управления. Объект класса навигации же, в свою очередь, для отправки данных о положении робота, получает информацию с датчиков навигации и, обрабатывая её, посылает обратно. Далее система управления получает

данные от датчиков расстояния (препятствий и высоты) для получения сигналов, на основе которых будет проводиться регулирование параметрами робота. Далее происходит подача желаемого и реального значения в пид регулятор, который производит расчёт управляющего сигнала. Затем метод из класса системы навигации отправляет данные на двигатели робота.

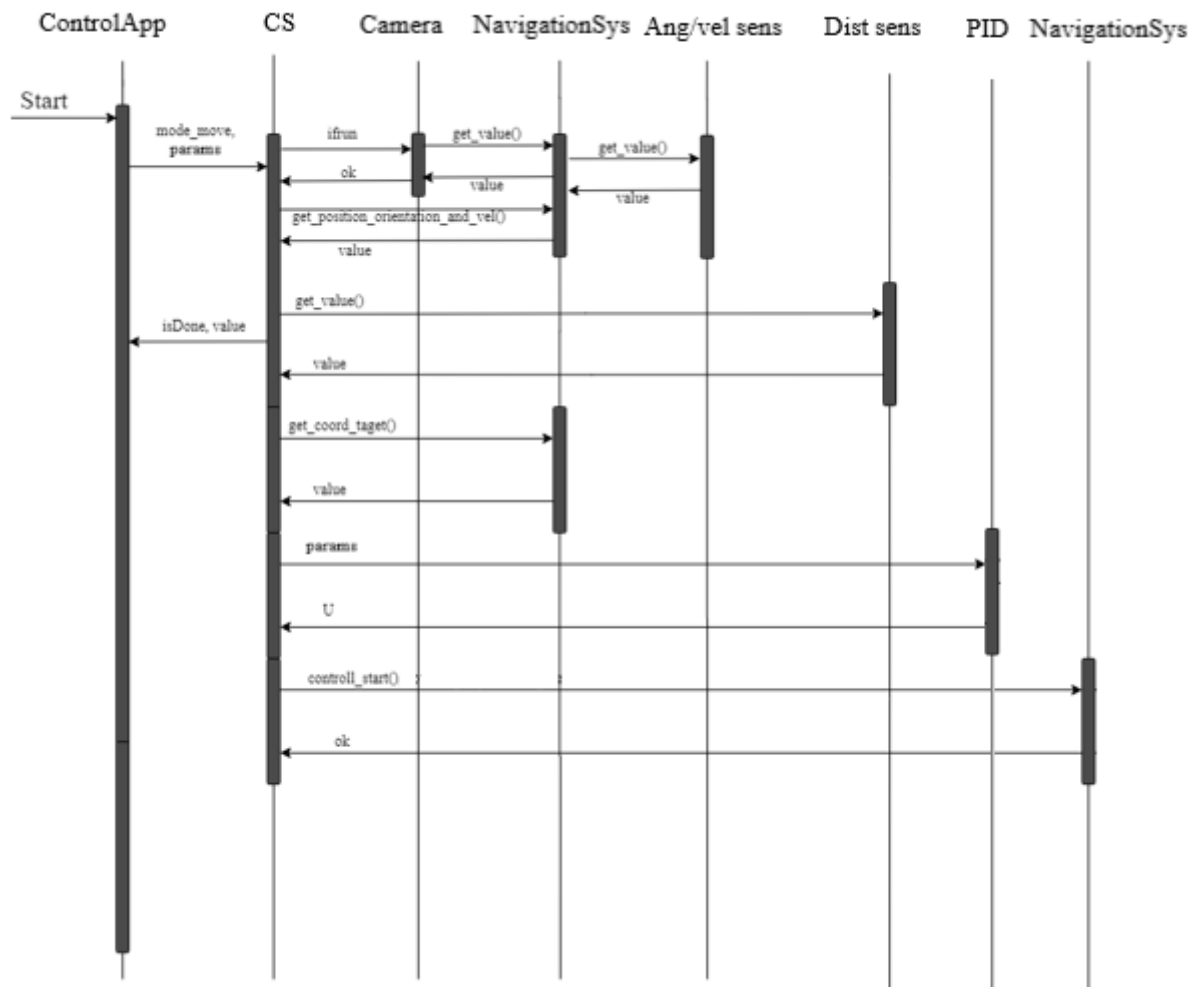


Рисунок 3 – Диаграмма последовательности

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

В состав технических средств должен входить компьютер, включающий в себя:

- 1) Процессор Core i5 с максимальной тактовой частотой от 3.6 ГГц;
- 2) Оперативную память объемом от 16 Гб;
- 3) Твердотельный накопитель от 100 Гб;
- 4) Свободное место на твердотельном накопителе - от 50 Гб;
- 5) Видеокартой с объемом видео памяти от 4 Гб и тактовой частотой от 1 МГц;
- 6) Операционную систему Windows 10;
- 7) Устройства управления и ввода: клавиатура, мышь.
- 8) Устройство вывода: монитор с разрешением от 1920x1080

5. ВЫЗОВ И ЗАГРУЗКА

Запуск программы осуществляется посредством открытия главного исполняемого файла «ControlApp.py». Старт работы робота происходит после нажатия кнопки «Start» в окне приложения.

Запуск исполняемых файлов происходит с помощью среды разработки Atom.

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для программы являются:

- Вводимые пользователем значения: массив точек, задающих траекторию движения робота, требуемую скорость движения.
- Данные с датчиков, сенсоров, тип float.

7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными являются:

- Управляющие сигналы для двигателей робота, отправляемые на сами двигатели.

[illegible]

УТВЕРЖДАЮ

Д.т.н., доцент департамента
автоматики и робототехники

_____ Юхимец Д.А.

«_____» _____ 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

Информационно-управляющая система «AUV - 2022»

Руководство системного программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.02067942.0005–01 РСП 13-01 ЛУ

Подпись и	
Инв. №	
Взам.	
Подпись и	
Инв. №	

Руководитель разработки

Д.т.н., доцент

_____ Юхимец Д.А.

«_____» _____ 2022

Исполнитель

Студент группы

Б11119-15.03.06мхрб

_____ Данилов Д.Д

«_____» _____ 2022

УТВЕРЖДЕН

RU.02067942.0005–01 РСП 13-01 ЛУ

Информационно-управляющая система «AUV - 2022»

Руководство системного программиста

RU.02067942.0005–01 РСП 13-01 ЛУ

Листов 6

Инд. №	Подпись и
Взам.	Инд. №
Подпись и	Взам.
Инд. №	Подпись и

2022

АННОТАЦИЯ

В данном программном документе приведено руководство системного программиста для программы «AUV - 2022», предназначенной для применения в подводных аппаратах с функцией поиска объектов. Программа реализована на языке программирования Python 3.9.4.

В данном программном документе описаны общие сведения о программе, настройка программы на условия конкретного применения, способы проверки, позволяющие дать общее заключение о работоспособности программы.

Оформление программного документа «Руководство системного программиста» произведено по требованиям ГОСТ 19.503-79.

СОДЕРЖАНИЕ

1. Общие сведения о программе	47
1.1 Назначение программы	47
1.2 Функции программы.....	47
1.3 Сведения о технических и программных средствах	47
2. Структура программы	48
2.1 Сведения о структуре программы.....	48
2.2 Взаимодействие с другими программами	48
3. Настройка на состав программных средств	48
4. Проверка программы	49
Лист регистрации изменений.....	50

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначение программы

Программа предназначена для осуществления навигации робота по заданной пользователем траектории или в одном из трёх видов движения, а также поиска объектов по заданному маршруту движения.

1.2 Функции программы

Программа имеет возможность осуществлять движение робота в заданную точку с обходом препятствий с заданной скоростью на заданной высоте от поверхности; нахождение объектов с помощью камер и отображение координат объектов в пользовательском приложении.

1.3 Сведения о технических и программных средствах

Программа должна эксплуатироваться на персональном компьютере включающего в себя:

- 1) Процессор Core i5 с максимальной тактовой частотой от 3.6 ГГц;
- 2) Оперативную память объемом от 16 Гб;
- 3) Твердотельный накопитель от 100 Гб;
- 4) Свободное место на твердотельном накопителе - от 50 Гб;
- 5) Видеокартой с объемом видео памяти от 4 Гб и тактовой частотой от 1 МГц;
- 6) Операционную систему Windows 10;
- 7) Устройства управления и ввода: клавиатура, мышь.
- 8) Устройство вывода: монитор с разрешением от 1920x1080

2. СТРУКТУРА ПРОГРАММЫ

2.1 Сведения о структуре программы

Подробное описание структуры и работы программы приведено в программном документе «Описание программы» настоящего (см. стр. 19)

2.2 Взаимодействие с другими программами

Для работы программы необходима виртуальная среда моделирования CoppeliaSim, в сцену которого загружен необходимый объект управления (подводный робот) с установленным набором сенсоров.

3. НАСТРОЙКА НА СОСТАВ ПРОГРАММНЫХ СРЕДСТВ

Запуск программы осуществляется посредством открытия главного исполняемого файла «ControlApp.py». Старт работы робота происходит после нажатия кнопки «Start» в окне приложения. Соединение с Coppelia Sim реализовано на двух Remote Api: ZMQ Remote Api служит для получения данных с датчиков линейных и угловых скоростей и отправки значения сил на движители, а Remote Api по порту для получения значений RGB массива с камеры. Такое разделение необходимо, потому что ZMQ Remote Api плохо передаёт массивы больших размерностей.

Запуск исполняемых файлов происходит с помощью среды разработки Atom.

4. ПРОВЕРКА ПРОГРАММЫ

Изначально необходимо запустить работу окна приложения через объект ControlApp. После нажатия кнопки «Start» произойдёт запуск потока, отвечающего за обновление данных в приложении, также запуск потока работы системы контроля ControlSys и запуск потока считывания камеры Camera. При этом от потока ControlSys будет зависеть соединение по ZMQ Remote Api серверу и от потока Camera соединение по порту. Для нормальной работы программы необходимо убедиться в корректности соединения с Coppelia Sim в обоих случаях. В первом функция RemoteAPIClient(), во втором sim.simxStart('127.0.0.1',19997,True,True,5000,5) вернут минус 1, если что-то пошло не так.

[illegible]

УТВЕРЖДАЮ

Д.т.н., доцент департамента
автоматики и робототехники

_____ Юхимец Д.А.

«_____» _____ 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

Информационно-управляющая система «AUV - 2022»

Руководство оператора

ЛИСТ УТВЕРЖДЕНИЯ

RU.02067942.0005–01 РО 13-01 ЛУ

Подпись и	
Инв. №	
Взам.	
Подпись и	
Инв. №	

Руководитель разработки

Д.т.н., доцент

_____ Юхимец Д.А.

«_____» _____ 2022

Исполнитель

Студент группы

Б11119-15.03.06мхрб

_____ Данилов Д.Д

«_____» _____ 2022

УТВЕРЖДЕН

RU.02067942.0005–01 РО 13-01 ЛУ

Информационно-управляющая система «AUV - 2022»

Руководство оператора

RU.02067942.0005–01 РО 13-01

Листов 11

Инв. №	Подпись и
Взам.	Инв. №
Подпись и	Взам.
Инв. №	Подпись и

2022

АННОТАЦИЯ

В данном программном документе приведено руководство оператора для программы «AUV - 2022», предназначенной для применения в подводных аппаратах с функцией поиска объектов. Программа реализована на языке программирования Python 3.9.4.

В данном программном документе описаны общие сведения о программе; условия, необходимые для выполнения программы; последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды; тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора.

Оформление программного документа «Руководство оператора» произведено по требованиям ГОСТ 19.505-79.

СОДЕРЖАНИЕ

1. Общие сведения о программе	55
1.1 Назначение программы	55
1.2 Функции программы.....	55
2. Условия выполнения программы	55
2.1 Требования к аппаратным средствам	55
2.2 Требования к программным средствам	56
2.3 Требования к оператору	56
3. Выполнение программы.....	56
3.1 Запуск программы.....	56
3.2 Начало работы	57
3.3 Настройка начальных параметров	58
3.4 Запуск работы системы	59
Лист регистрации изменений.....	60

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначение программы

Программа предназначена для осуществления навигации робота по одному из режимов функционирования: движение в точку с заданной скоростью, движение в точку по сплайну с заданной скоростью, движение в точку по прямой, выполнение миссии (проход последовательно-установленных точек с заданным типом движения), поиск объектов на карте.

1.2 Функции программы

Программа имеет возможность осуществлять движение робота в заданную точку с обходом препятствий с заданной скоростью и на заданной высоте от поверхности; находить объекты красного цвета, расположенного в заранее неизвестном месте карты.

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 Требования к аппаратным средствам

Программа эксплуатируется на персональном компьютере включающего в себя:

- 1) Процессор Core i5 с максимальной тактовой частотой от 3.6 ГГц;
- 2) Оперативную память объемом от 16 Гб;
- 3) Твердотельный накопитель от 100 Гб;
- 4) Свободное место на твердотельном накопителе - от 50 Гб;

- 5) Видеокартой с объемом видео памяти от 4 Гб и тактовой частотой от 1 МГц;
- 6) Операционную систему Windows 10;
- 7) Устройства управления и ввода: клавиатура, мышь.
- 8) Устройство вывода: монитор с разрешением от 1920x1080

2.2 Требования к программным средствам

Для работы программы необходима виртуальная среда моделирования CorreliaSim, в сцену которого загружен необходимый объект управления (подводный робот).

2.3 Требования к оператору

Требуется навык использования персонального компьютера, иметь опыт работы с графическими интерфейсами, иметь опыт работы в среде Atom.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1 Запуск программы

Запуск программы осуществляется посредством открытия главного исполняемого файла «ControlApp.py». Старт работы робота происходит после нажатия кнопки «Start» в окне приложения.

3.2 Начало работы

При запуске файла “ControlApp.py” пользователь увидит окно приложения с подписями и местами для заполнения.

The screenshot shows a window titled 'MainWindow' with three main sections for data entry:

- Moving to point:** Includes input fields for 'Point:', 'Velocity:', and 'Height:'.
- Mission:** Includes input fields for 'Points:', 'Velocity:', 'Position robot: X:', 'Y:', 'Z:', 'Velocity:', and 'Position goal: x:'.
- Splain:** Includes input fields for 'Points Y:', 'Velocity:', and 'h:'.

A 'Start' button is located at the bottom center of the window.

Рисунок 5 – Общий вид «Окна приложения»

Пропуски около подписей означают, что там будут отображаться значения, места для заполнения отмечены белой рамкой.

3.3 Настройка начальных параметров

Для запуска конкретного режима необходимо ввести нужные значения скорости и координат точек в формате X,Y с отделением пробелом для каждой последующей при этом остальные оставить пустыми, высота указывается для всех режимов в одном блоке.

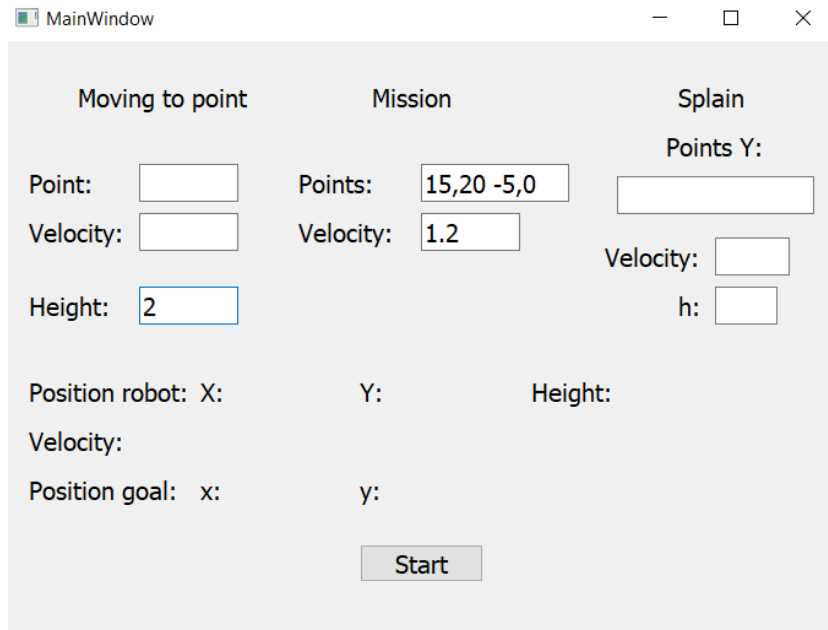
При работе со сплайном h обозначает шаг, либо 1, либо -1, что означает перемещение в положительную или отрицательную область по оси x , при этом точки по Y задаются перечислением через запятую.

The screenshot shows a software window titled 'MainWindow' with three main sections for configuring movement parameters:

- Moving to point:**
 - Point:
 - Velocity:
 - Height:
- Mission:**
 - Points:
 - Velocity:
 - Start:
- Splain:**
 - Points Y:
 - Velocity:
 - h:

Below these sections, there are labels for 'Position robot: X:', 'Y:', 'Z:', 'Velocity:', 'Position goal: x:', and 'y:', but no corresponding input fields are visible in the image.

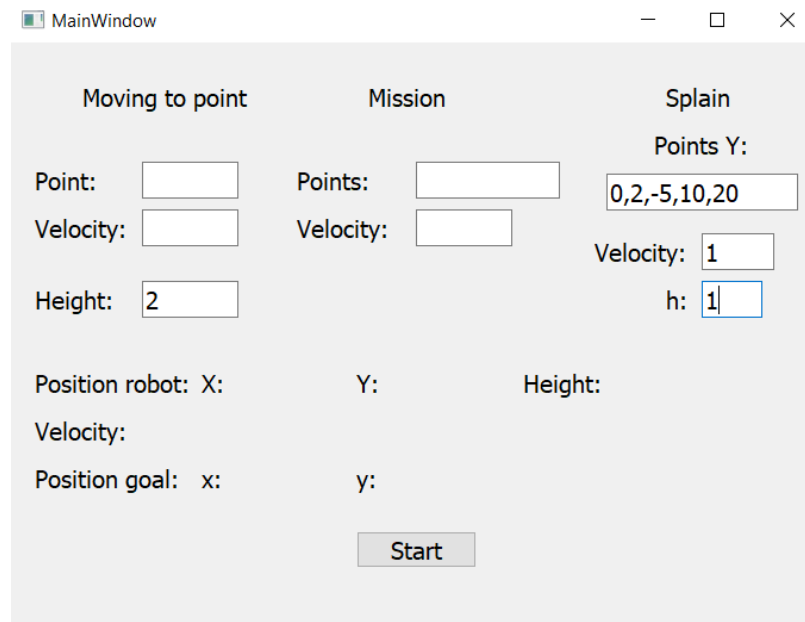
Рисунок 6 – Пример заполнения формы в окне приложения для движения в точку



MainWindow

Moving to point	Mission	Splain
Point: <input type="text"/>	Points: <input type="text" value="15,20 -5,0"/>	Points Y: <input type="text"/>
Velocity: <input type="text"/>	Velocity: <input type="text" value="1.2"/>	Velocity: <input type="text"/>
Height: <input type="text" value="2"/>		h: <input type="text"/>
Position robot: X:	Y:	Height:
Velocity:		
Position goal: x:	y:	
<input type="button" value="Start"/>		

Рисунок 7 – Пример заполнения формы в окне приложения для миссии робота



MainWindow

Moving to point	Mission	Splain
Point: <input type="text"/>	Points: <input type="text"/>	Points Y: <input type="text" value="0,2,-5,10,20"/>
Velocity: <input type="text"/>	Velocity: <input type="text"/>	Velocity: <input type="text" value="1"/>
Height: <input type="text" value="2"/>		h: <input type="text" value="1"/>
Position robot: X:	Y:	Height:
Velocity:		
Position goal: x:	y:	
<input type="button" value="Start"/>		

3.4 Запуск работы системы

После заполнения окна приложения для желаемого режима необходимо нажать кнопку “Start”. Пустые поля будут обновляться в соответствии с положением робота, скоростью и обнаружением объекта красного цвета.

[illegible][illegible]

УТВЕРЖДАЮ

Д.т.н., доцент департамента
автоматики и робототехники

_____ Юхимец Д.А.

«_____» _____ 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

Информационно-управляющая система «AUV - 2022»

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.02067942.0005–01 ТП 13-01 ЛУ

Руководитель разработки
Д.т.н., доцент

_____ Юхимец Д.А.

«_____» _____ 2022

Исполнитель
Студент группы
Б11119-15.03.06мхрб

_____ Данилов Д.Д

«_____» _____ 2022

Подпись и	
Инв. №	
Взам.	
Подпись и	
Инв. №	

2022

УТВЕРЖДЕН

RU.02067942.0005–01 ТП 13-01 ЛУ

Информационно-управляющая система «AUV - 2022»

Текст программы

RU.02067942.0005–01 ТП 13-01

Листов 23

Инов. №	Подпись и
Взам.	Инов. №
Подпись и	Взам.
Инов. №	Подпись и

2022

СОДЕРЖАНИЕ

1. Листинг модуля App	64
2. Листинг модуля ControlApp.....	69
3. Листинг модуля NavigationSys	72
4. Листинг модуля CS	78
5. Листинг модуля Camera	85
6. Листинг модуля PID	85
Лист регистрации изменений.....	88

1. Листинг модуля App

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(682, 491)
        font = QtGui.QFont()
        font.setPointSize(12)
        MainWindow.setFont(font)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(60, 30, 141, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_2.setGeometry(QtCore.QRect(110, 100, 81, 31))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(300, 30, 81, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(550, 30, 61, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_4.setFont(font)
        self.label_4.setTextFormat(QtCore.Qt.AutoText)
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(160, 270, 21, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_5.setFont(font)
        self.label_5.setTextFormat(QtCore.Qt.AutoText)
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(290, 270, 21, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_6.setFont(font)
        self.label_6.setTextFormat(QtCore.Qt.AutoText)
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(self.centralwidget)
        self.label_7.setGeometry(QtCore.QRect(430, 270, 71, 31))
```



```
font = QtGui.QFont()
font.setPointSize(12)
self.label_7.setFont(font)
self.label_7.setTextFormat(QtCore.Qt.AutoText)
self.label_7.setObjectName("label_7")
self.label_8 = QtWidgets.QLabel(self.centralwidget)
self.label_8.setGeometry(QtCore.QRect(20, 310, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_8.setFont(font)
self.label_8.setTextFormat(QtCore.Qt.AutoText)
self.label_8.setObjectName("label_8")
self.label_10 = QtWidgets.QLabel(self.centralwidget)
self.label_10.setGeometry(QtCore.QRect(510, 430, 51, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_10.setFont(font)
self.label_10.setText("")
self.label_10.setTextFormat(QtCore.Qt.AutoText)
self.label_10.setObjectName("label_10")
self.label_13 = QtWidgets.QLabel(self.centralwidget)
self.label_13.setGeometry(QtCore.QRect(20, 100, 51, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_13.setFont(font)
self.label_13.setTextFormat(QtCore.Qt.AutoText)
self.label_13.setObjectName("label_13")
self.label_14 = QtWidgets.QLabel(self.centralwidget)
self.label_14.setGeometry(QtCore.QRect(20, 140, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_14.setFont(font)
self.label_14.setTextFormat(QtCore.Qt.AutoText)
self.label_14.setObjectName("label_14")
self.label_15 = QtWidgets.QLabel(self.centralwidget)
self.label_15.setGeometry(QtCore.QRect(240, 100, 61, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_15.setFont(font)
self.label_15.setTextFormat(QtCore.Qt.AutoText)
self.label_15.setObjectName("label_15")
self.label_16 = QtWidgets.QLabel(self.centralwidget)
self.label_16.setGeometry(QtCore.QRect(240, 140, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_16.setFont(font)
self.label_16.setTextFormat(QtCore.Qt.AutoText)
self.label_16.setObjectName("label_16")
self.lineEdit_3 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_3.setGeometry(QtCore.QRect(110, 140, 81, 31))
self.lineEdit_3.setObjectName("lineEdit_3")
self.lineEdit_4 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_4.setGeometry(QtCore.QRect(340, 100, 121, 31))
self.lineEdit_4.setObjectName("lineEdit_4")
```

```
self.lineEdit_5 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_5.setGeometry(QtCore.QRect(340, 140, 81, 31))
self.lineEdit_5.setObjectName("lineEdit_5")
self.label_17 = QtWidgets.QLabel(self.centralwidget)
self.label_17.setGeometry(QtCore.QRect(540, 70, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_17.setFont(font)
self.label_17.setTextFormat(QtCore.Qt.AutoText)
self.label_17.setObjectName("label_17")
self.lineEdit_6 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_6.setGeometry(QtCore.QRect(500, 110, 161, 31))
self.lineEdit_6.setObjectName("lineEdit_6")
self.label_18 = QtWidgets.QLabel(self.centralwidget)
self.label_18.setGeometry(QtCore.QRect(20, 350, 191, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_18.setFont(font)
self.label_18.setTextFormat(QtCore.Qt.AutoText)
self.label_18.setObjectName("label_18")
self.label_19 = QtWidgets.QLabel(self.centralwidget)
self.label_19.setGeometry(QtCore.QRect(190, 270, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_19.setFont(font)
self.label_19.setText("")
self.label_19.setTextFormat(QtCore.Qt.AutoText)
self.label_19.setObjectName("label_19")
self.label_20 = QtWidgets.QLabel(self.centralwidget)
self.label_20.setGeometry(QtCore.QRect(20, 270, 151, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_20.setFont(font)
self.label_20.setTextFormat(QtCore.Qt.AutoText)
self.label_20.setObjectName("label_20")
self.label_21 = QtWidgets.QLabel(self.centralwidget)
self.label_21.setGeometry(QtCore.QRect(320, 270, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_21.setFont(font)
self.label_21.setText("")
self.label_21.setTextFormat(QtCore.Qt.AutoText)
self.label_21.setObjectName("label_21")
self.label_22 = QtWidgets.QLabel(self.centralwidget)
self.label_22.setGeometry(QtCore.QRect(500, 270, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_22.setFont(font)
self.label_22.setText("")
self.label_22.setTextFormat(QtCore.Qt.AutoText)
self.label_22.setObjectName("label_22")
self.label_23 = QtWidgets.QLabel(self.centralwidget)
self.label_23.setGeometry(QtCore.QRect(120, 310, 81, 31))
font = QtGui.QFont()
```

```
font.setPointSize(12)
self.label_23.setFont(font)
self.label_23.setText("")
self.label_23.setTextFormat(QtCore.Qt.AutoText)
self.label_23.setObjectName("label_23")
self.label_24 = QtWidgets.QLabel(self.centralwidget)
self.label_24.setGeometry(QtCore.QRect(190, 350, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_24.setFont(font)
self.label_24.setText("")
self.label_24.setTextFormat(QtCore.Qt.AutoText)
self.label_24.setObjectName("label_24")
self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(290, 410, 101, 31))
self.pushButton.setObjectName("pushButton")
self.label_9 = QtWidgets.QLabel(self.centralwidget)
self.label_9.setGeometry(QtCore.QRect(160, 350, 21, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_9.setFont(font)
self.label_9.setTextFormat(QtCore.Qt.AutoText)
self.label_9.setObjectName("label_9")
self.label_11 = QtWidgets.QLabel(self.centralwidget)
self.label_11.setGeometry(QtCore.QRect(290, 350, 21, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_11.setFont(font)
self.label_11.setTextFormat(QtCore.Qt.AutoText)
self.label_11.setObjectName("label_11")
self.label_25 = QtWidgets.QLabel(self.centralwidget)
self.label_25.setGeometry(QtCore.QRect(320, 350, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_25.setFont(font)
self.label_25.setText("")
self.label_25.setTextFormat(QtCore.Qt.AutoText)
self.label_25.setObjectName("label_25")
self.label_26 = QtWidgets.QLabel(self.centralwidget)
self.label_26.setGeometry(QtCore.QRect(20, 200, 71, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_26.setFont(font)
self.label_26.setTextFormat(QtCore.Qt.AutoText)
self.label_26.setObjectName("label_26")
self.lineEdit_7 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_7.setGeometry(QtCore.QRect(110, 200, 81, 31))
self.lineEdit_7.setObjectName("lineEdit_7")
self.label_27 = QtWidgets.QLabel(self.centralwidget)
self.label_27.setGeometry(QtCore.QRect(550, 200, 21, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_27.setFont(font)
self.label_27.setTextFormat(QtCore.Qt.AutoText)
```

```

self.label_27.setObjectName("label_27")
self.lineEdit_8 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_8.setGeometry(QtCore.QRect(580, 200, 51, 31))
self.lineEdit_8.setObjectName("lineEdit_8")
self.lineEdit_9 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_9.setGeometry(QtCore.QRect(580, 160, 61, 31))
self.lineEdit_9.setObjectName("lineEdit_9")
self.label_28 = QtWidgets.QLabel(self.centralwidget)
self.label_28.setGeometry(QtCore.QRect(490, 160, 81, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.label_28.setFont(font)
self.label_28.setTextFormat(QtCore.Qt.AutoText)
self.label_28.setObjectName("label_28")
MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
    self.label_2.setText(_translate("MainWindow", "Moving to
point"))
    self.label_3.setText(_translate("MainWindow", "Mission"))
    self.label_4.setText(_translate("MainWindow", "Splain"))
    self.label_5.setText(_translate("MainWindow", "X:"))
    self.label_6.setText(_translate("MainWindow", "Y:"))
    self.label_7.setText(_translate("MainWindow", "Height:"))
    self.label_8.setText(_translate("MainWindow", "Velocity:"))
    self.label_13.setText(_translate("MainWindow", "Point:"))
    self.label_14.setText(_translate("MainWindow", "Velocity:"))
    self.label_15.setText(_translate("MainWindow", "Points:"))
    self.label_16.setText(_translate("MainWindow", "Velocity:"))
    self.label_17.setText(_translate("MainWindow", "Points Y:"))
    self.label_18.setText(_translate("MainWindow", "Position
goal:"))
    self.label_20.setText(_translate("MainWindow", "Position
robot:"))
    self.pushButton.setText(_translate("MainWindow", "Start"))
    self.label_9.setText(_translate("MainWindow", "x:"))
    self.label_11.setText(_translate("MainWindow", "y:"))
    self.label_26.setText(_translate("MainWindow", "Height:"))
    self.label_27.setText(_translate("MainWindow", "h:"))
    self.label_28.setText(_translate("MainWindow", "Velocity:"))

```

2. Листинг модуля ControlApp

```

import App
import threading
from PyQt5 import QtWidgets, QtCore
from PyQt5.QtCore import QThread
import time
import threading
import CS
import random
import numpy as np

class UpdateData(QThread):

    flag = 0

    def __init__(self,mainwindow,parent=None):
        super().__init__()
        self.mainwindow = mainwindow

    def toListPoints(self,PointsY):

        count = PointsY.count(',')
        out = np.zeros(count+1)
        to_array = [char for char in PointsY]
        num = ""
        u = 0

        for i in range(len(to_array)):

            if to_array[i] == ",":
                out[u] = float(num)
                u = u + 1
                num = ""

            if to_array[i] != "," and to_array[i] != " ":
                num = num + str(to_array[i])

            if to_array[i] == " " or i == len(to_array)-1:
                out[u] = float(num)
                u = u + 1
                num = ""

        return out.tolist()

    def toArrayPoints(self,Points):

        o = 0
        u = 0
        num = ""

```

```

count = Points.count(',')
out = np.zeros((count,4))
to_array = [char for char in Points]
print(to_array)

for i in range(len(to_array)):

    if to_array[i] == ",":
        out[o][u] = float(num)
        u = u + 1
        num = ""

    if to_array[i] != "," and to_array[i]!=" ":
        num = num + str(to_array[i])

    if to_array[i] == " " or i == len(to_array)-1:
        out[o][u] = float(num)
        o = o + 1
        u = 0
        num = ""

for i in range(count):
    out[i][3] = 1

print(out)
return out

def run(self):
    while True:

        if self.flag == 0:

            #первый режим
            Point = self.mainwindow.lineEdit_2.text()
            Velocity0 = self.mainwindow.lineEdit_3.text()

            #второй режим
            Points = self.mainwindow.lineEdit_4.text()
            Velocity1 = self.mainwindow.lineEdit_5.text()

            #третий режим
            PointsY = self.mainwindow.lineEdit_6.text()
            Velocity2 = self.mainwindow.lineEdit_9.text()
            h = self.mainwindow.lineEdit_8.text()

            #высота
            Height = self.mainwindow.lineEdit_7.text()

            if(Points!=""):

                CS.ps_g = self.toArrayPoints(Points)
                CS.vel_zhel = float(Velocity1)
                CS.h_zhel = float(Height)
                CS.Mode = 1

```

```

    if(Point!=""):

        ar = self.toArrayPoints(Point)
        ar[0,2] = 2
        CS.ps_g = ar[0]
        CS.vel_zhel = float(Velocity0)
        CS.h_zhel = float(Height)
        CS.Mode = 0

    if (PointsY!=""):

        if(h == "-1"): CS.PointsX = [0,-20,-40,-60,-80]
        if(h == "1"): CS.PointsX = [0,20,40,60,80]
        CS.PointsY = self.toListPoints(PointsY)
        CS.vel_zhel = float(Velocity2)
        CS.h_zhel = float(Height)
        CS.Mode = 2

    threading.Thread(target=CS.startControlSys).start()
    self.flag = 1

    self.mainwindow.label_19.setText(str(round(CS.X,3)))
    time.sleep(0.001)
    self.mainwindow.lineEdit_2.text()
    time.sleep(0.001)
    self.mainwindow.label_21.setText(str(round(CS.Y,3)))
    time.sleep(0.001)
    self.mainwindow.label_22.setText(str(round(CS.h,3)))
    time.sleep(0.001)

    self.mainwindow.label_23.setText(str(round(CS.vel_gl_mod,3)))
    time.sleep(0.001)
    self.mainwindow.label_24.setText(str(round(CS.x_cub,3)))
    time.sleep(0.001)
    self.mainwindow.label_25.setText(str(round(CS.y_cub,3)))
    time.sleep(0.001)

class Registration(QtWidgets.QMainWindow, App.Ui_MainWindow):

    def __init__(self):
        super(Registration,self).__init__()
        self.setupUi(self)

        self.pushButton.clicked.connect(self.launch)
        self.Aupdate_instance = AupdateData(mainwindow = self)

    def launch(self):
        self.Aupdate_instance.start()

App = QtWidgets.QApplication([])
window = Registration()
window.show()
App.exec()

```

3. Листинг модуля NavigationSys

```

import time
from zmqRemoteApi import RemoteAPIClient
import numpy as np

class NavigationSys():

    h_past = 2
    distP_past = 2 - 0.133
    distZ_past = 2 - 0.133

    X = 0
    Y = 0
    Z = 2

    sim = 0

    def getVelocityLocal(self, past_velocity):

        vel = self.sim.getStringSignal("velocity")
        if(vel!=None):

            vel = self.sim.unpackFloatTable(vel)
            sumVel = np.sqrt(vel[0]**2 + vel[1]**2 + vel[2]**2)

            if(sumVel<0.00001): return [0,0,0]
            else: return vel

        return past_velocity

    def getAnglesOrient(self, past_angles):

        ang = self.sim.getStringSignal("angles")
        if(ang!=None): angles = self.sim.unpackFloatTable(ang); return
angles

        return past_angles

    def getLaserDist(self, past_laser_dist):

        las = self.sim.getStringSignal("Lasers_distance")
        if(las!=None): las_dist = self.sim.unpackFloatTable(las);
return las_dist

        return past_laser_dist

    def getProxyDist(self):

        proxy_detect = self.sim.getStringSignal("Proxy_distance")

```



```

    if(proxy_detect!=None): proxy_detect =
self.sim.unpackFloatTable(proxy_detect); return
proxy_detect[0],proxy_detect[1],proxy_detect[2]

```

```

    return 0,0,0

```

#Курс дифферент расчитываются для ССК

```

def calculateKurs(self,x,y,z):

```

```

    if(x==0 and y==0):

```

```

        return 0

```

```

    if (x>0 and y>0) or (x>0 and y<0):

```

```

        alpha = np.arctan(y/x)*(180/np.pi)

```

```

    if x<0 and y>0 :

```

```

        alpha = 180 + np.arctan(y/x)*(180/np.pi)

```

```

    if x<0 and y<0 :

```

```

        alpha = -180 + np.arctan(y/x)*(180/np.pi)

```

```

    if y==0 and x<0 :

```

```

        alpha = 180

```

```

    if x == 0 and y>0:

```

```

        alpha = 90

```

```

    if y==0 and x>0 :

```

```

        alpha = 0

```

```

    if x==0 and y<0:

```

```

        alpha = -90

```

```

    return alpha

```

```

def calculateDifferent(self,dist_P,dist_Z):

```

```

    betta = np.arctan((dist_P-dist_Z)/7.7196)

```

```

    return betta*(180/np.pi)

```

```

def filterLasers(self,h,distP,distZ):

```

```

    h1 = 0

```

```

    dP = 0

```

```

    dZ = 0

```

```

    eps = h - self.h_past

```

```

    if(np.abs(eps)>0.1):

```

```

        h1 = self.h_past

```

```

    else :

```

```

        h1 = h

```

```

    self.h_past = h1

```

```

    epsP = distP - self.distP_past

```

```

    if(np.abs(epsP)>0.1):

```

```

        dP = self.distP_past

```

```

    else:

```

```

        dP = distP

```

```

self.distP_past = dP

epsZ = distZ - self.distZ_past
if(np.abs(epsZ)>0.1):
    dZ = self.distZ_past
else:
    dZ = distZ
self.distZ_past = dZ

return h1,dP, dZ

def getVelocityGlobal(self,a,b,c,vel_local):

    A = np.matrix([ [np.cos(a)*np.cos(b),
np.cos(a)*np.sin(b)*np.sin(c)-
np.cos(c)*np.sin(a),np.sin(a)*np.sin(c)+np.cos(a)*np.cos(c)*np.sin(b),
0],
                    [np.cos(b)*np.sin(a),
np.cos(a)*np.cos(c)+np.sin(a)*np.sin(b)*np.sin(c),
np.cos(c)*np.sin(a)*np.sin(b)-np.cos(a)*np.sin(c),0],
                    [-np.sin(b), np.cos(b)*np.sin(c),
np.cos(b)*np.cos(c),0],
                    [0,0,0,1]])

    vel_0 = [0,0,0,1]
    vel_0 [0] = vel_local[0]
    vel_0 [1] = vel_local[1]
    vel_0 [2] = vel_local[2]

    vel_global = np.dot(A,vel_0)

    vel_gl_mod = np.sqrt(vel_global[0,0]**2 + vel_global[0,1]**2 +
vel_global[0,2]**2)

    return vel_global,vel_gl_mod

def getPositionRobot(self,vel_global):

    self.X = self.X + vel_global[0,0]*0.1
    self.Y = self.Y + vel_global[0,1]*0.1
    self.Z = self.Z + vel_global[0,2]*0.1

    return self.X,self.Y,self.Z

def getLocalCoordinatesPoint(self,a,b,c,X,Y,Z,point_global):

    A = np.matrix([ [np.cos(a)*np.cos(b),
np.cos(a)*np.sin(b)*np.sin(c)-
np.cos(c)*np.sin(a),np.sin(a)*np.sin(c)+np.cos(a)*np.cos(c)*np.sin(b),
X],
                    [np.cos(b)*np.sin(a),
np.cos(a)*np.cos(c)+np.sin(a)*np.sin(b)*np.sin(c),
np.cos(c)*np.sin(a)*np.sin(b)-np.cos(a)*np.sin(c),Y],
                    [-np.sin(b), np.cos(b)*np.sin(c),
np.cos(b)*np.cos(c),Z],
                    [0,0,0,1]])

```

```

        [-np.sin(b), np.cos(b)*np.sin(c),
np.cos(b)*np.cos(c), Z],
        [0,0,0,1]])

    point_local = np.dot(np.linalg.inv(A),point_global)

    return point_local

    def getGlobalCoordinatesPoint(self,a,b,c,X,Y,Z,point_local):

        A = np.matrix([ [np.cos(a)*np.cos(b),
np.cos(a)*np.sin(b)*np.sin(c)-
np.cos(c)*np.sin(a),np.sin(a)*np.sin(c)+np.cos(a)*np.cos(c)*np.sin(b),
X],
                        [np.cos(b)*np.sin(a),
np.cos(a)*np.cos(c)+np.sin(a)*np.sin(b)*np.sin(c),
np.cos(c)*np.sin(a)*np.sin(b)-np.cos(a)*np.sin(c),Y],
                        [-np.sin(b), np.cos(b)*np.sin(c),
np.cos(b)*np.cos(c),Z],
                        [0,0,0,1]])

        point_global = np.dot(A,point_local)

        return
[point_global[0,0],point_global[0,1],point_global[0,2],point_global[0,
3]]

    def calculateDist(self,point_global,point_local,X,Y):

        vec_point = np.sqrt(point_global[0]**2 + point_global[1]**2)
        vec_robot = np.sqrt(X**2 + Y**2)

        if(vec_point>=vec_robot):

            disToPoint = np.sqrt(point_local[0,0]**2 +
point_local[0,1]**2)

        else:

            disToPoint = -np.sqrt(point_local[0,0]**2 +
point_local[0,1]**2)

        return disToPoint

    def calculateDistAbs(self,point_local):

        disToPoint = np.sqrt(point_local[0,0]**2 +
point_local[0,1]**2)

        return disToPoint

    def calculateSpline(self,x,y,Z):

```

```

lenght = len(y) - 2
#заполняем массив шагов
h = np.zeros(len(x)-1)
for i in range(len(h)):
    h[i] = x[i+1]-x[i]
#Создаем матрицу коэффициентов H и вектор F.
u0 = np.zeros(lenght)
u1 = np.zeros(lenght)
u2 = np.zeros(lenght)
for i in range(lenght):
    u0[i] = h[i+1]
    u1[i] = h[i]
    u2[i] = 2*(h[i+1]+h[i])
#Расставляем диагональки
P = np.diag(u0[:len(u0)-1], k =1)
P1 = np.diag(u1[:len(u1)-1],k =-1)
P2 = np.diag(u2, k = 0)
H = P+P1+P2
# F вектор
F = np.zeros((lenght,1))
i1 = 2
for i in range(lenght):
    F[i,0] = 3*((y[i1]-y[i1-1])/h[i1-1] - (y[i1-1]-y[i1-
2])/h[i1-1])
    i1 = i1+1
# Матрица с коэффициентами C
C = np.dot(np.linalg.inv(H),F)
#Заполняем массив коэффициентов
a = np.zeros((lenght+1,1)); c = np.zeros((lenght+2,1))
b = np.zeros((lenght+2,1)); d = np.zeros((lenght+1,1))
i1=0
for i in range(lenght+1):
    if i < (lenght):
        c[i+1,0] = C[i]
        b[i,0] = (y[i+1] - y[i]) / h[i] - h[i] / 3 * (c[i+1] + 2 *
c[i])

        d[i,0] = (c[i+1] - c[i]) / (3 * h[i])
        a[i,0] = y[i]

#Заполняем 2 массива для графика
yForGraph = np.zeros((lenght+1)*10 + 1)
xForGraph = np.zeros((lenght+1)*10 + 1)
hDel = 0
i2 = 0
for i in range(lenght+1):

    if(i==3):
        for i1 in range(11):
            yForGraph[i2] = a[i,0] + b[i,0]*(hDel-x[i]) +
c[i,0]*((hDel-x[i])**2) + d[i,0]*((hDel-x[i])**3)
            xForGraph[i2] = hDel
            hDel = hDel + h[i]/10
            i2 = i2 + 1

```

```

        else:
            for i1 in range(10):
                yForGraph[i2] = a[i,0] + b[i,0]*(hDel-x[i]) +
c[i,0]*((hDel-x[i])**2) + d[i,0]*((hDel-x[i])**3)
                xForGraph[i2] = hDel
                hDel = hDel + h[i]/10
                i2 = i2 + 1
            points_global = np.zeros((len(xForGraph),4))

            for i in range(len(xForGraph)-1):
                points_global[i] = [xForGraph[i],yForGraph[i],Z,1]

            dist_between = np.zeros(len(xForGraph)-1)
            for i in range(len(dist_between)):
                dist_between[i] = np.sqrt((xForGraph[i+1]-xForGraph[i])**2
+ (yForGraph[i+1]-yForGraph[i])**2)

            return points_global,dist_between # КОЛ-ВО их на i-1

def addForces(self,alpha,betta,U,U1,U2,U3,U4):

self.sim.setStringSignal("addForces",self.sim.packFloatTable([alpha,bet
tta,U,U1,U2,U3,U4]))

```

4. Листинг модуля CS

```
import time
from zmqRemoteApi import RemoteAPIClient
import time
from zmqRemoteApi import RemoteAPIClient
import numpy as np
import Camera
import threading
import NavigationSys
import PID

client = RemoteAPIClient();
sim = client.getObject('sim')
client.setStepping(True)

Cam = Camera.Camera()
threading.Thread(target=Cam.registerCamera, args = (str(2))).start()

past_velocity = np.zeros(3)
past_angles = np.zeros(3)
past_laser_dist = np.zeros(2)
kos = 0
k1 = 0

PID_height = PID.PID(0.05, 8, 0, 15.5)
PID_different = PID.PID(0.05, 30, 0.5, 8)
PID_velocity = PID.PID(0.05, 100, 60, 0.1)
PID_distance = PID.PID(0.05, 10, 0.5, 28)
PID_kurs = PID.PID(0.05, 3, 0.08, 0.01)

NavigationSys = NavigationSys.NavigationSys()
NavigationSys.sim = sim
time_start = time.time()

p_spl_g, dist_between = 0, 0
dist_pastAbs = 0
o = 0
end = 0
i = 0
Mode = 0
vel_gl_mod_past = 0
flag_1 = 0
flag_2 = 0
flag_4 = 0
fl = 3
addPoint = 0
X = 0; Y = 0; h = 0
x_cub = 0; y_cub = 0
vel_gl_mod = 0
PointsX = 0
PointsY = 0

ps_g = 0
vel_zhel = 0
```

```

h_zhel = 0

def startControlSys():

    global client, sim, past_velocity, past_angles, past_laser_dist, p_g,
ps_g
    global kos, k1, PID_height, PID_different, PID_velocity, vel_zhel
    global PID_distance, PID_kurs, NavigationSys, time_start;
    global dist_pastAbs, o, end, i, Mode, vel_gl_mod_past, PointsX, PointsY
    global p_spl_g, dist_between, flag_1, flag_4, fl, addPoint
    global X, Y, h, x_cub, y_cub, vel_gl_mod, h_zhel, flag_2

    sim.startSimulation()

    while True:

        if (Mode == 0):

            velocity = NavigationSys.getVelocityLocal(past_velocity)
            las_dist = NavigationSys.getLaserDist(past_laser_dist)
            a, b, c = NavigationSys.getAnglesOrient(past_angles)
            detected_P, detected_L, detected_R =
NavigationSys.getProxyDist()

            vel_global, vel_gl_mod =
NavigationSys.getVelocityGlobal(a, b, c, velocity)
            X, Y, Z = NavigationSys.getPositionRobot(vel_global)

            if (Cam.detected > 0 and Cam.detected < 8):
                coord =
NavigationSys.getGlobalCoordinatesPoint(a, b, c, X, Y, 0, [Cam.x, Cam.y, 0, 1])
                x_cub = coord[0]
                y_cub = coord[1]

            if flag_1 == 0 :

                point_global = ps_g

            else :

                point_global = addPoint

            point_local =
NavigationSys.getLocalCoordinatesPoint(a, b, c, X, Y, Z, point_global)

            dist =
NavigationSys.calculateDist(point_global, point_local, X, Y)
            distAbs = NavigationSys.calculateDistAbs(point_local)

```

```

x = point_local[0,0]
y = point_local[0,1]
z = point_local[0,2]

alpha = NavigationSys.calculateKurs(x,y,z)

h,distP,distZ = NavigationSys.filterLasers((las_dist[0] +
las_dist[1] + 0.266)/2,las_dist[0],las_dist[1])
betta = NavigationSys.calculateDifferent(distP,distZ)

if (distAbs<1 and flag_4 == 1):

    flag_1 = 0; flag_4 = 0

    if (((detected_P == 1 and detected_L == 1 and detected_R
== 1) or (detected_R == 1 and detected_P == 1)) and flag_4 == 0):
        flag_1 = 1; flag_4 = 1
        addPoint =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,Z,[8,4,0,1])

    if (detected_L == 1 and detected_P == 1 and flag_1 == 0
and flag_4 == 0):

        flag_1 = 1; flag_4 = 1
        addPoint =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,Z,[8,-4,0,1])

eps_time = time.time() - time_start

if(eps_time>25):

    U2 = PID_velocity.getControl(vel_gl_mod,vel_zhel)
    U = PID_kurs.getControl(alpha,0)

    if(distAbs<5 and flag_1 == 0 ): U2 = 0; k1 = 1

else:

    U = PID_kurs.getControl(alpha,0)
    U2 = 0

U1 = PID_different.getControl(betta,0)
U3 = PID_height.getControl(h_zhel,h)
U4 = 0

if vel_gl_mod<0.01 and k1 == 1:
    U2 = 0
    U = 0
    kos = 1

```



```

if kos == 0:
    NavigationSys.addForces(alpha,betta,U,U1,U2,U3,U4)
else:
    if(distAbs<0.3): U4 = 0
    else: U4 = PID_distance.getControl(dist,0)
    NavigationSys.addForces(alpha,betta,0,U1,0,U3,U4)

if(Mode == 1):

    #Скорость локальную, дистанцию датчиков, ориентацию углов
    velocity = NavigationSys.getVelocityLocal(past_velocity)
    las_dist = NavigationSys.getLaserDist(past_laser_dist)
    a,b,c = NavigationSys.getAnglesOrient(past_angles)
    detected_P,detected_L,detected_R =
NavigationSys.getProxyDist()

    points_global = ps_g
    if(o == 0): dist_pastAbs = np.sqrt(points_global[0][0]**2
+ points_global[0][1]**2)

    if(dist_pastAbs<1 and fl == 3):
        i = i + 1
        if(i==len(points_global)):
            end = 1

    if(end==0 and flag_1 == 0): point_global =
points_global[i]
    if(end==0 and flag_1 == 1): point_global = addPoint

    #Скорость в глобальной вектором и модулем, координаты
робота
    vel_global,vel_gl_mod =
NavigationSys.getVelocityGlobal(a,b,c,velocity)
    X,Y,Z = NavigationSys.getPositionRobot(vel_global)

    #Получение координат кубика
    if(Cam.detected>0 and Cam.detected<8):
        coord =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,0,[Cam.x,Cam.y,0,1])
        x_cub = coord[0]
        y_cub = coord[1]

    point_local =
NavigationSys.getLocalCoordinatesPoint(a,b,c,X,Y,Z,point_global)
    dist =
NavigationSys.calculateDist(point_global,point_local,X,Y)
    distAbs = NavigationSys.calculateDistAbs(point_local)

    x = point_local[0,0]
    y = point_local[0,1]

```

```

z = point_local[0,2]

alpha = NavigationSys.calculateKurs(x,y,z)

h,distP,distZ = NavigationSys.filterLasers((las_dist[0] +
las_dist[1] + 0.266)/2,las_dist[0],las_dist[1])
betta = NavigationSys.calculateDifferent(distP,distZ)

if (distAbs<1 and flag_4 == 1):

    flag_1 = 0; flag_4 = 0; fl = 2

if (distAbs>1 and flag_4 == 0):

    fl = 3

    if (((detected_P == 1 and detected_L == 1 and detected_R
== 1) or (detected_R == 1 and detected_P == 1)) and flag_4 == 0):
        fl = 1
        flag_1 = 1; flag_4 = 1
        addPoint =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,Z,[8,4,0,1])

    if (detected_L == 1 and detected_P == 1 and flag_1 == 0
and flag_4 == 0):
        fl = 1
        flag_1 = 1; flag_4 = 1
        addPoint =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,Z,[8,-4,0,1])

#тут желаемые в пиды идут
U2 = PID_velocity.getControl(vel_gl_mod,vel_zhel)
U1 = PID_different.getControl(betta,0)
U3 = PID_height.getControl(h_zhel,h)
U = PID_kurs.getControl(alpha,0)

U4 = 0

if(end==1):U = 0; U1 = 0; U2 = 0; U4 = 0

NavigationSys.addForces(alpha,betta,U,U1,U2,U3,U4)

dist_pastAbs = distAbs

if (Mode == 2):

    if(flag_2 == 0):
        p_spl_g,dist_between =
NavigationSys.calculateSpline(PointsX,PointsY,2)
        flag_2 = 1

#Скорость локальную, дистанцию датчиков, ориентацию углов
velocity = NavigationSys.getVelocityLocal(past_velocity)

```

```

las_dist = NavigationSys.getLaserDist(past_laser_dist)
a,b,c = NavigationSys.getAnglesOrient(past_angles)

if(o == 0):
    dist_pastAbs = dist_between[0]

if(i>=len(dist_between)):
    dist_bet = dist_between[len(dist_between)-1]
else:
    dist_bet = dist_between[i]

if(dist_pastAbs<dist_bet*0.5):

    if(i==len(p_spl_g)-1):

        i = len(p_spl_g)-1
        if (dist_pastAbs<2): end = 1

    else: i = i + 1

if(end==0): point_global = p_spl_g[i]

vel_global,vel_gl_mod =
NavigationSys.getVelocityGlobal(a,b,c,velocity)
X,Y,Z = NavigationSys.getPositionRobot(vel_global)

if(Cam.detected>0 and Cam.detected<8):
    coord =
NavigationSys.getGlobalCoordinatesPoint(a,b,c,X,Y,0,[Cam.x,Cam.y,0,1])
    x_cub = coord[0]
    y_cub = coord[1]

    point_local =
NavigationSys.getLocalCoordinatesPoint(a,b,c,X,Y,Z,point_global)

dist =
NavigationSys.calculateDist(point_global,point_local,X,Y)
distAbs = NavigationSys.calculateDistAbs(point_local)

x = point_local[0,0]
y = point_local[0,1]
z = point_local[0,2]

alpha = NavigationSys.calculateKurs(x,y,z)

h,distP,distZ = NavigationSys.filterLasers((las_dist[0] +
las_dist[1] + 0.266)/2,las_dist[0],las_dist[1])
beta = NavigationSys.calculateDifferent(distP,distZ)

eps_time = time.time() - time_start

```

```
if (eps_time > 20):
    U2 = PID_velocity.getControl(vel_gl_mod, vel_zhel)
else:
    U2 = 0

U1 = PID_different.getControl(betta, 0)
U3 = PID_height.getControl(h_zhel, h)
U = PID_kurs.getControl(alpha, 0)
U4 = 0

if (end == 1): U = 0; U1 = 0; U2 = 0; U4 = 0

NavigationSys.addForces(alpha, betta, U, U1, U2, U3, U4)

dist_pastAbs = distAbs

o = o + 1
past_laser_dist = las_dist
past_velocity = velocity

client.step()

sim.stopSimulation()
```

5. Листинг модуля Camera

```
import sim
import sys
import time
import math
import cv2
import numpy as np
from PIL import Image

class Cam:

    x, y = 0, 0
    detected = 0
    z = 2

    def registrateCamera(self, z):

        self.z = float(z)
        sim.simxFinish(-1)
        clientID = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)

        if clientID == -1:
            print('Ничего не работает')

        error, camera =
sim.simxGetObjectHandle(clientID, "VS", sim.simx_opmode_one-shot_wait)
        error, resolution, image =
sim.simxGetVisionSensorImage(clientID, camera, 0, sim.simx_opmode_streami
ng)

        while (True):

            error, resolution, image =
sim.simxGetVisionSensorImage(clientID, camera, 0, sim.simx_opmode_buffer)
            if error == sim.simx_return_ok:

                ImageBGR = np.array(image, dtype=np.uint8)
                ImageBGR.resize(resolution[0], resolution[1], 3)
                ImageRGB_rotated =
cv2.cvtColor(ImageBGR, cv2.COLOR_BGR2RGB)

                flip_image = cv2.flip(ImageRGB_rotated, 0)
                #cv2.imshow("Flip image", flip_image)

                ImageRGB_rotated = flip_image

                hsv = cv2.cvtColor(ImageRGB_rotated, cv2.COLOR_RGB2HSV
)

                h1_min = np.array((115, 0, 0), np.uint8)
                h1_max = np.array((255, 255, 255), np.uint8)
                thresh1 = cv2.inRange(hsv, h1_min, h1_max)
                #cv2.imshow('re', thresh1)

                moments = cv2.moments(thresh1, 1)
```

```

dM01 = moments['m01']
dM10 = moments['m10']
dArea = moments['m00']
# будем реагировать только на те моменты,
# которые содержат больше 100 пикселей

    if dArea > 10 and self.detected>=0 and
self.detected<8:
        self.detected = self.detected + 1
        x = resolution[0]/2 - int(dM10 / dArea); Y = x
        y = resolution[0]/2 - int(dM01 / dArea); X = y #
X;Y координаты в пикселях относительно центра робота

        z = 2
        x_local = (X/resolution[0])*(self.z - 0.134)*2
        y_local = (Y/resolution[0])*(self.z - 0.134)*2
        self.x = x_local; self.y = y_local
        if dArea<10:self.detected = 0; cv2.waitKey(3)

```

6. Листинг модуля PID

```
class PID:

    h = 0; kp = 0; ki = 0; kd1 = 0;
    U = 0; U_past = 0; eps = 0; eps_past = 0;
    ui = 0; ui_past = 0;

    def __init__(self, h, kp, ki, kd):

        self.h = h
        self.kp = kp
        self.ki = ki
        self.kd = kd

    def getControl(self, param_input, param_zhelaem):

        self.eps = param_zhelaem - param_input
        self.ui = self.ui_past + self.ki*self.h*self.eps
        self.U = self.kp*self.eps + self.ui + self.kd*(self.eps-
self.eps_past)/self.h

        self.U_past = self.U
        self.ui_past = self.ui
        self.eps_past = self.eps

    return self.U
```

[illegible]

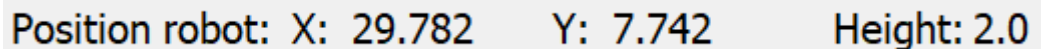
РЕЗУЛЬТАТЫ РАБОТЫ

Далее проведем различные испытания для определения качества работы ИУС управления подводным роботом.

Испытание 1

На тестовом полигоне зададим координату целевой точки – [30,7.8] и оценим полученные результаты: координаты робота по достижении точки и скорость. Параметры режима работы:

- Заданная точка: [30,7.8];
- Заданная скорость: 1.5 м/с;
- Заданная высота: 2м;
- Режим движения: движение по прямой с заданной скоростью.



Position robot: X: 29.782 Y: 7.742 Height: 2.0

Рисунок 6 – Позиция робота в приложении



x: +29.8075 y: +7.7445 z: +2.0003

Рисунок 7 – Позиция робота в CoppeliaSim

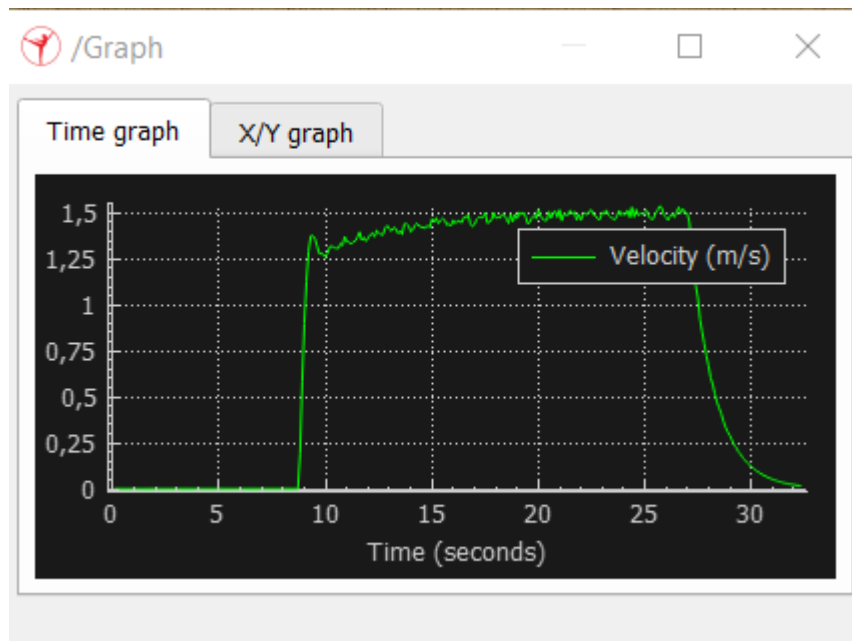


Рисунок 8 – График скорости робота в CoppeliaSim

Из рисунков 6-7 можно заключить, что робот приходит в заданную точку с ошибкой около 10см, что является приемлемым значением. Как видно из рисунка 8, график скорости соответствует заданному значению. Начало движения робота происходит в момент 8с, поэтому время переходного процесса составит около 10с, на моменте 26с робот начинает тормозить перед точкой.

Испытание 2

В ходе второго испытания зададим миссию роботу с координатами $[10,10]$, $[-10,10]$, $[-5,-5]$, при этом добавим цель красного цвета (куб) в заранее неизвестном для робота положении. Параметры режима работы:

- Заданные точки: $[10,10]$, $[-10,10]$, $[-5,-5]$
- Заданная скорость: 1.5 м/с;
- Заданная высота: 2м;
- Режим движения: миссия.

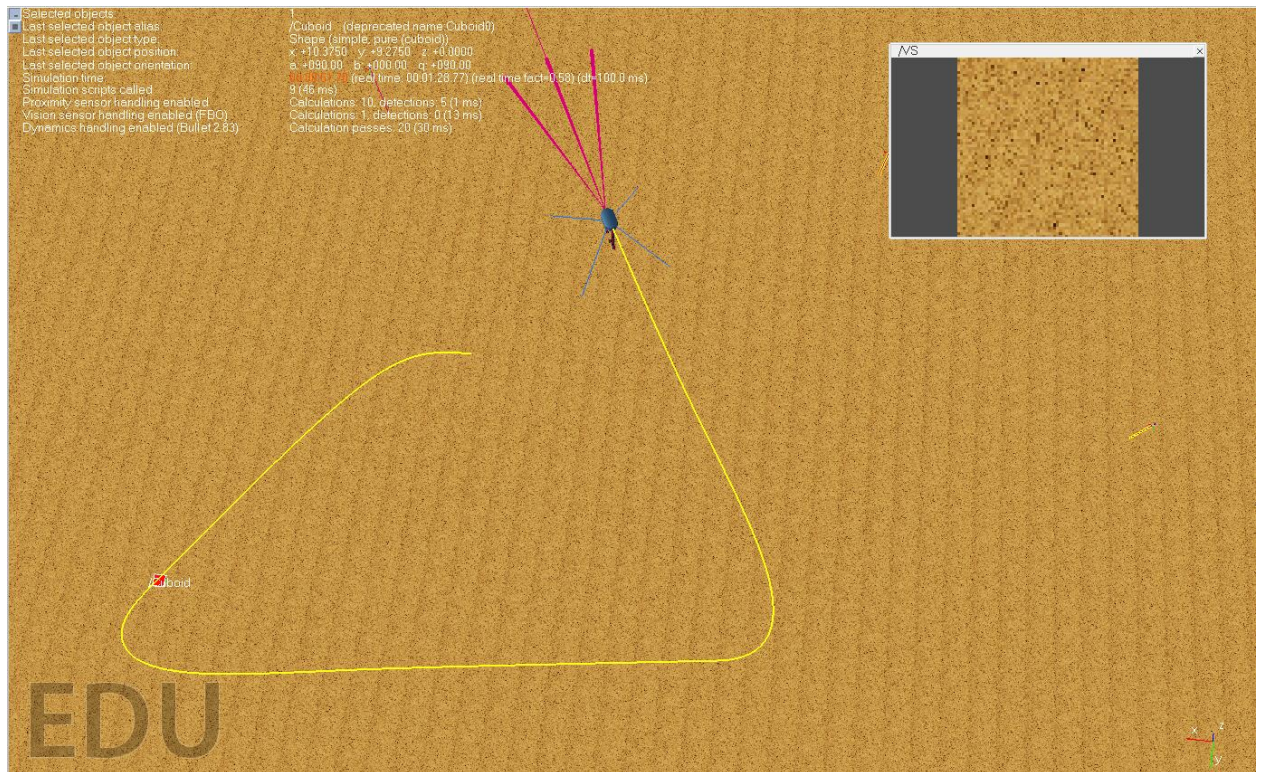


Рисунок 9 – Траектория движения робота

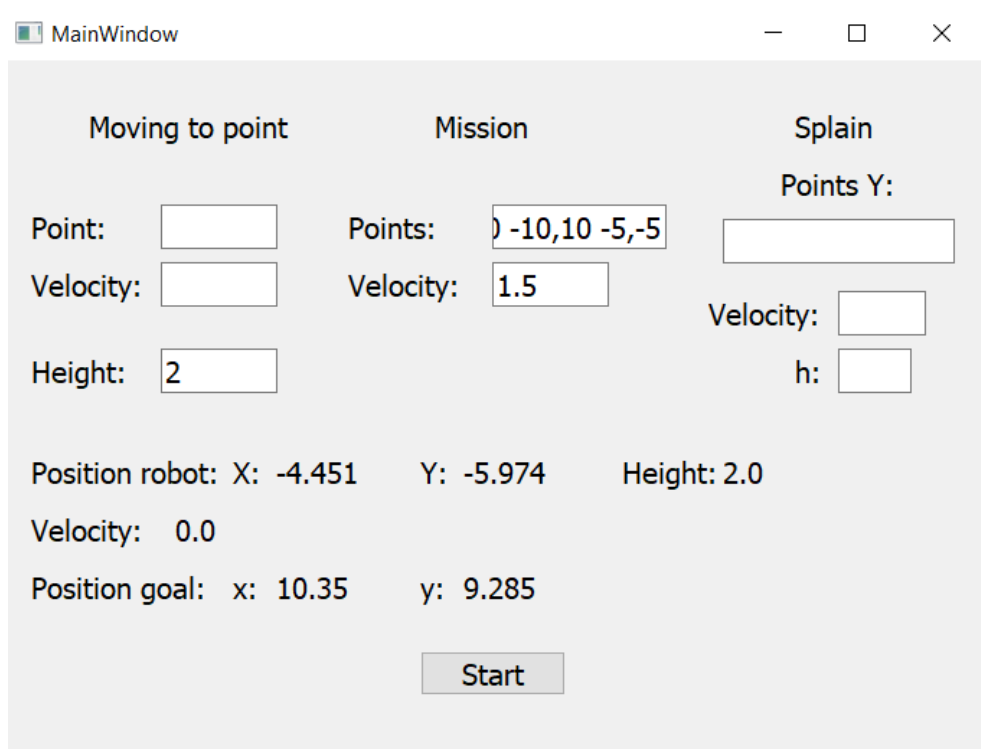


Рисунок 10 – Окно приложения

```

/Cuboid (deprecated name:Cuboid0)
Shape (simple, pure (cuboid))
x: +10.3440 y: +9.1240 z: +0.0500
a: +090.00 b: +000.00 q: +090.00

```

Рисунок 11 – Координаты красного куба

Как видно из рисунков 10-11 координаты красного куба отличаются в пределах 10см, что является приемлемым, следовательно, камера работает корректно, ошибка по координатам составляет около 1.2м, что является допустимым при данной скорости, также высота в конечной точке равна заданной, что говорит о корректной работе пид регулятора высоты.

Испытание 3

В ходе третьего испытания проведём проверку объезда роботом препятствия. Зададим объект “Cylinder” с размерами 3м в диаметре и 3м в высоту. Параметры режима работы:

- Заданные точки: [15,15]
- Заданная скорость: 1 м/с;
- Заданная высота: 2м;
- Режим движения: движение по прямой с заданной скоростью.

Результаты работы приведены на рисунке 18.

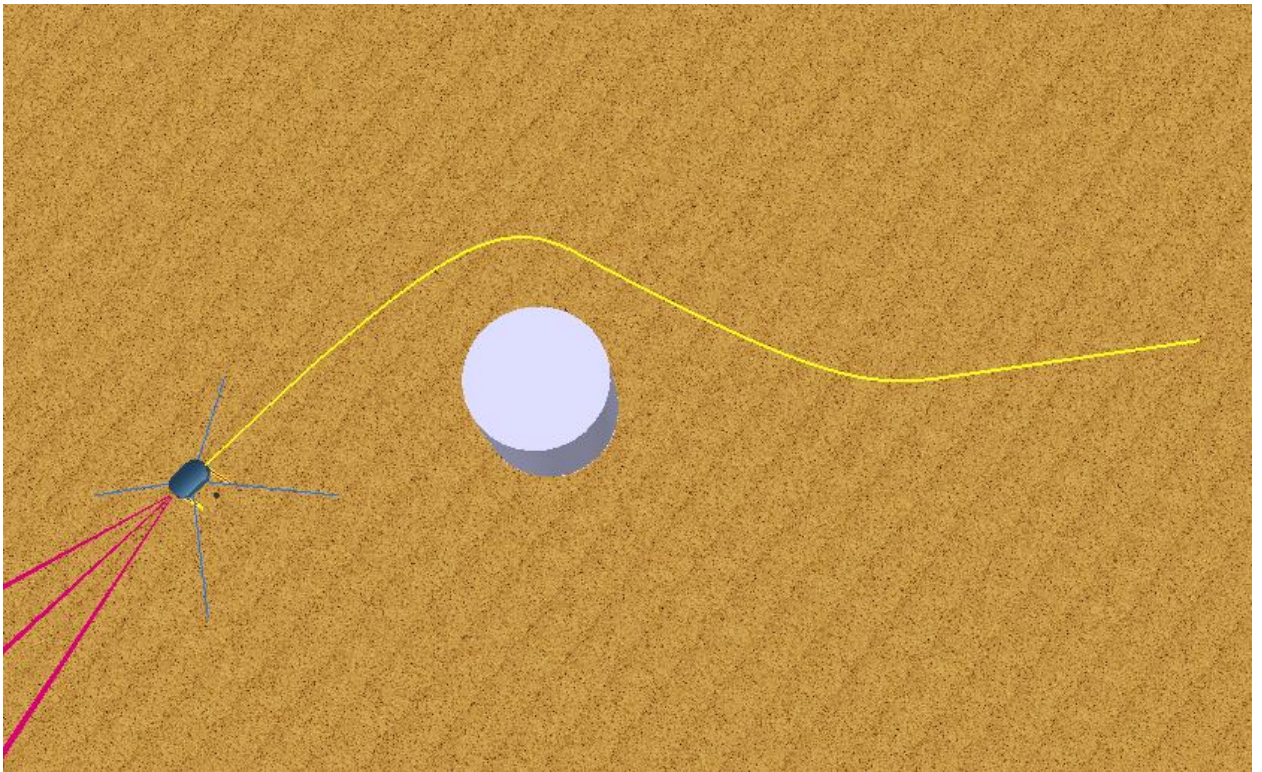


Рисунок 12 – Объезд препятствия роботом

Из результатов работы, представленных на рисунке 12 видно, что обход препятствий выполняется корректно.

Испытание 4

В ходе четвертого испытания проведем исследование прохождения по кривой, задаваемой сплайном третьего порядка. Параметры режима работы:

- Заданные точки по Y: [0,5,-10,10,-15]
- Заданные точки по X: [0,-20,-40,-60,-80]
- Заданная скорость: 1 м/с;
- Заданная высота: 2м;
- Режим движения: движение по сплайну.

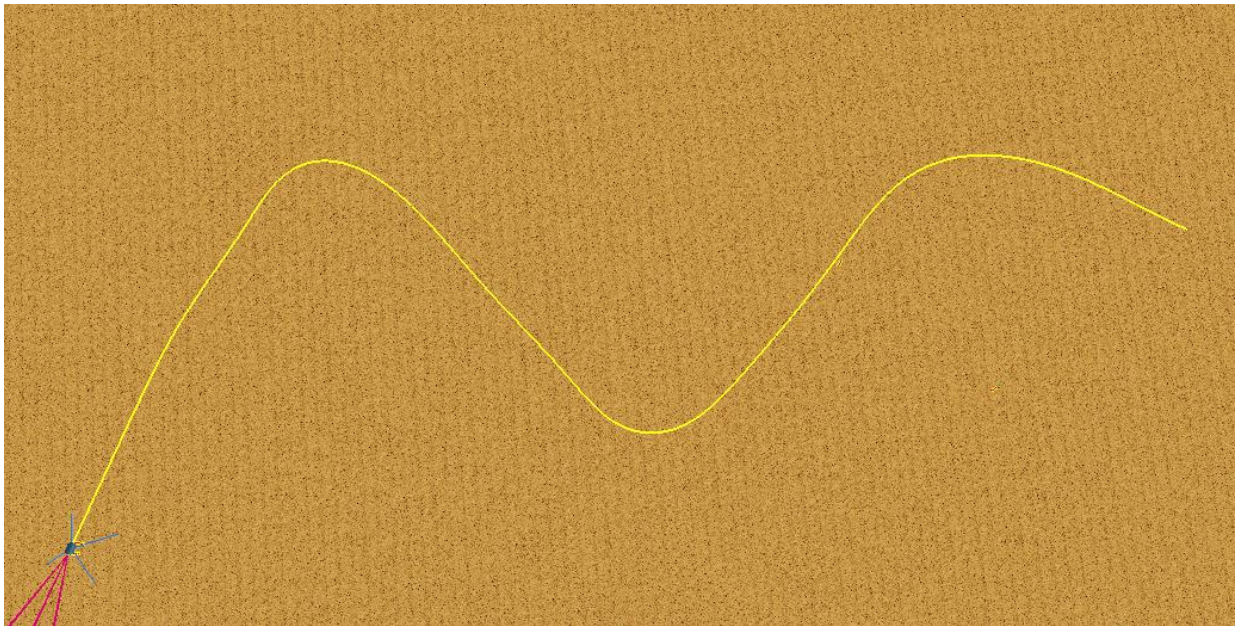


Рисунок 13 – Движение по сплайну

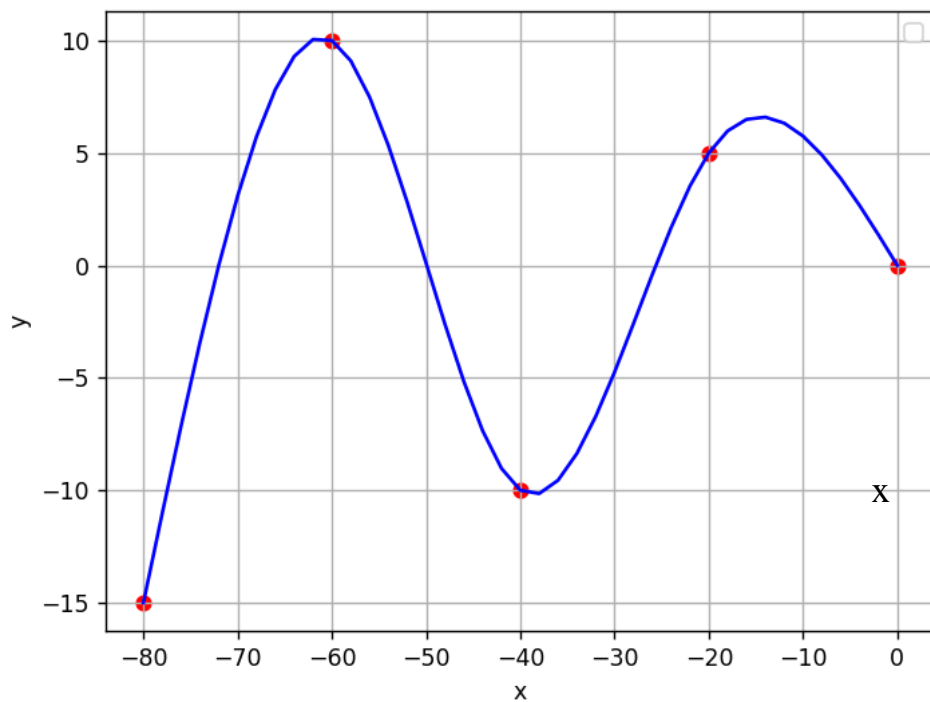


Рисунок 14 – Заданный сплайн третьего порядка

По результатам работы, представленным на рисунках 13-14 видно, что траектория движения робота схожа с заданным сплайном. Отклонения возникают из-за того, что робот переключает точки сплайна в диапазоне 0.5

метров, из-за чего не успевает развернуться на точки, особенно на перегибах. Повысить точность прохода роботом точек можно уменьшив скорость, а затем расстояние переключения.

ВЫВОД

В ходе выполнения курсовой работы были получены навыки создания автоматизированной информационно-управляющей системы для подводного робота с функцией поиска объектов, расположенных в заранее неизвестных местах. Была разработана система управления подводным роботом на языке программирования Python с использованием объектно-ориентированной архитектуры. Также был разработан пользовательский интерфейс позволяющий отслеживать выполнение миссии, а также задавать и отслеживать параметры робота.

Разработана следующая документация: техническое задание, описание программы, руководство системного программиста, руководство оператора и текст программы.

Было произведено тестирование полученной информационно-управляющей системы, в ходе которой она успешно справилась с поставленными задачами.