МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Интеллектуальные информационные технологии»

«К защите допускаю»

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Головко

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ** **КОНТРОЛЯ**

**ДОСТУПА С ПОДДЕРЖКОЙ ИЗМЕРЕНИЯ ТЕМПЕРАТУРЫ**

**НА ОСНОВЕ ПЛАТФОРМЫ CANTONK S02T**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ДИПЛОМНОМУ ПРОЕКТУ

**ДП.ИИ15.170124-06 81 00**

Листов

|  |  |
| --- | --- |
|  |  |
| Руководитель | В. В. Касьяник |
| Выполнил | А. А. Буров |
| Консультанты: |  |
| по основным разделам | В. В. Касьяник |
| по экономическому разделу | Э. Э. Ермакова |
| по ЕСПД | М.В. Хацкевич |
| Рецензент |  |

2021

**АННОТАЦИЯ**

67 с./ 70 c., 26 рис., 3 табл., 5 ист. лит., 2 прил., 6 л. граф. матер.

В дипломном проекте рассмотрены прямая и обратная кинематические задачи робота-манипулятора, приведены некоторые способы их решения. Описана теория генетических алгоритмов, а также применение их в задаче управления роботами-манипуляторами. Сформулированы требования к системе.

Выбрано два генетических алгоритма для решения обратной кинематической задачи, описана общая структура программных средств, представлены результаты их проектирования.

Разработано соответствующее программное обеспечение, реализованное на основе WPF-приложения. Для моделирования трехмерной модели робота-манипулятора, а также для отображения решения обратной кинематической задачи с помощью генетических алгоритмов использованы классы пространства имен System.Windows.Media.Media3D и библиотека OxyPlot (построение графиков). Реализован модуль для подключения к реальному роботу-манипулятору RoboArm и отправке ему полученного решения. Приведены результаты испытания и тестирования системы. Произведено сравнение использованных алгоритмов с другими алгоритмами для решения обратной кинематической задачи. Выполнена оценка экономической эффективности.

Форма № 17

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Факультет | | ЭИС | | | | | | | | Кафедра | | | ИИТ | | | |
|  | |  | | | | | | | |  | | |  | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | |
| Зав. кафедрой | | |  | | | | | | |  | | | | | | |
|  | | | (подпись) | | | | | | |  | | | | | | |
| « » |  | | | | | | | | 2021 г. | | | | | | | |
|  |  | | | | | | | |  | | | | | | | |
| ЗАДАНИЕ | | | | | | | | | | | | | | | | |
| по дипломному проектированию | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Студенту | | Бурову Александру Андреевичу | | | | | | | | | | | | | | |
| 1. Тема проекта | | | | Разработка программной системы контроля доступа с поддержкой измерения | | | | | | | | | | | | |
| температуры на основе платформы Cantonk S02T | | | | | | | | | | | | |
| (Утверждена приказом по вузу от | | | | | | | | | 20.04.2021 | | | | | № | 415-C ) | |
|  | | | | | | | | |  | | | | |  |  | |
| 2. Сроки сдачи студентом законченного проекта | | | | | | | | | | | 10.06.2021 г. | | | | | |
| 3. Исходные данные к проекту | | | | | | | |  | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Документация и описание робота-манипулятора RoboArm. | | | | | | | | | | | | | | | | |
| Описание прямой и обратной кинематической задачи. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Язык программирования: Python, Операционная система: Windows. | | | | | | | | | | | | | | | | |
| Особенности разработки: использовании генетических алгоритмов для решения обратной кинематической | | | | | | | | | | | | | | | | |
| задачи, визуализация полученного решения | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Система должна обеспечивать: | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 1. Трехмерное моделирование и визуализацию робота-манипулятора | | | | | | | | | | | | | | | | |
| 2. Ввод, редактирование параметров робота-манипулятора. | | | | | | | | | | | | | | | | |
| 3. Вычисление прямой задачи кинематики. | | | | | | | | | | | | | | | | |
| 4. Решение обратной задачи кинематики с помощью генетического алгоритма. | | | | | | | | | | | | | | | | |
| 5. Вывод результатов решения обратной задачи. | | | | | | | | | | | | | | | | |
| 6. Построение траектории движения на основе полученных решений ОКЗ. | | | | | | | | | | | | | | | | |
| 7. Взаимодействие с реальным роботом-манипулятором RoboArm. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | |
| 1. Анализ предметной области и постановка задачи | | | | | | | | | | | | | | | | |
| 2. Проектирование программных средств | | | | | | | | | | | | | | | | |
| 3. Реализация и испытание | | | | | | | | | | | | | | | | |
| 4. Технико-экономическое обоснование | | | | | | | | | | | | | | | | |
| 5. Энерго- и ресурсосбережение | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | |
| Список сокращений | | | | | | | | | | | | | | | | |
| Список использованных источников | | | | | | | | | | | | | | | | |
| Приложения А блок-схемы агоритмов | | | | | | | | | | | | | | | | |
| Приложения Б текст программы | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 5. Перечень графического материала (с точным указанием обязательных чертежей и | | | | | | | | | | | | | | | | |
| графиков) | |  | | | | | | | | | | | | | | |
| 1. Постановка задачи (плакат – формат А1) | | | | | | | | | | | | | | | | |
| 2. Структура системы (плакат – формат А1) | | | | | | | | | | | | | | | | |
| 3. Схема работы системы (чертеж “схема работы системы ” – формат А1) | | | | | | | | | | | | | | | | |
| 4. Схема программы (чертеж “схема программы” – формат А1) | | | | | | | | | | | | | | | | |
| 5. Результаты испытаний (плакат – формат А1) | | | | | | | | | | | | | | | | |
| 6. Экранные формы (плакат – формат А1) | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 6. Консультанты по проекту (с указанием относящихся к ним разделов проекта) | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| ст. преп. Касьяник В. В. – по основным разделам | | | | | | | | | | | | | | | | |
| ст. преп. Хацкевич М. В. – нормоконтроль | | | | | | | | | | | | | | | | |
| ст. преп. Ермакова Э. Э. – экономический раздел | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 7. Дата выдачи задания | | | | | | | 30.03.2021 г. | | | | | | | | | |
|  | | | | | | |  | | | | | | | | | |
| 8. Календарный график работы над проектом на весь период проектирования (с указанием | | | | | | | | | | | | | | | | |
| сроков выполнения и трудоемкость отдельных этапов) | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | |
| Раздел 1: 30.03 – 14.04: 20% | | | | | | | | | | | | | | | | |
| Раздел 2: 14.04 – 04.05: 30% | | | | | | | | | | | | | | | | |
| Раздел 3: 05.05 – 25.05: 30% | | | | | | | | | | | | | | | | |
| Раздел 4, 5: 26.05 – 01.06: 10% | | | | | | | | | | | | | | | | |
| Оформление проекта: 02.06 – 10.06: 10% | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | Руководитель | | | |  | | | | | |  |
|  | | | | | |  | | | | (подпись) | | | | | |  |
|  | | | | | | | | | | | | | | | | |
| Задание принял к исполнению (дата) | | | | | | | | | |  | | | | | |  |
|  | | | | | | | | | | | | | | | | |
| (подпись студента) | | | | |  | | | | | | | | | | |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc452712327)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc452712328)

[1.1 Описание кинематической модели робота-манипулятора 7](#_Toc452712329)

[1.2 Описание прямой и обратной кинематической задачи 11](#_Toc452712330)

[1.3 Применение генетических алгоритмов в задаче управления роботами-манипуляторами **Ошибка! Закладка не определена.**](#_Toc452712331)

[1.4 Постановка задачи 21](#_Toc452712332)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ 22](#_Toc452712333)

[2.1 Общее описание структуры ПС 22](#_Toc452712334)

[2.2 Проектирование ПС 25](#_Toc452712335)

[2.3 Генетический алгоритм решения обратной кинематической задачи 34](#_Toc452712336)

[2.4 Выбор средств программирования 38](#_Toc452712337)

[3 РЕАЛИЗАЦИЯ И ИСПЫТАНИЕ ПС 40](#_Toc452712338)

[3.1 Реализация ПС 40](#_Toc452712339)

[3.2 Испытание ПС 50](#_Toc452712340)

[4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 57](#_Toc452712341)

[4.1 Исходные данные для расчета экономического эффекта 57](#_Toc452712342)

[4.2 Расчет объема функций программного обеспечения 57](#_Toc452712343)

[4.3 Расчет полной себестоимости программного продукта **Ошибка! Закладка не определена.**](#_Toc452712344)

[4.4 Расчет цены и прибыли программного продукта 65](#_Toc452712345)

[5 ЭНЕРГО- И РЕСУРСОСБЕРЕЖЕНИЕ 68](#_Toc452712346)

[ЗАКЛЮЧЕНИЕ 71](#_Toc452712347)

[СПИСОК СОКРАЩЕНИЙ 72](#_Toc452712348)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 73](#_Toc452712349)

[БЛОК-СХЕМЫ АЛГОРИТМОВ 74](#_Toc452712350)

ВВЕДЕНИЕ

Система контроля доступа – совокупность программно-аппаратных технических средств контроля, имеющих целью регистрацию входа-выхода объектов (людей, транспорта) на заданной территории через «точки прохода»: двери, ворота, КПП.

Современные возможности систем контроля доступа – какие задачи выполняет оборудование:

* + Надежная защита объекта от попыток взлома, проникновения посторонних;
  + Учет количества людей, прошедших через КПП в обе стороны. Функциональное решение для учета рабочего времени сотрудников и регистрации посетителей;
  + Возможность разграничения доступа сотрудников на объект;
  + Ограничение доступа на парковку;
  + Контроль перемещения работников по предприятию, помогая наладить эффективную и рациональную деятельность сотрудников;
  + Возможность идентификации личности человека по фотографии, сетчатке глаза, отпечаткам пальцев.

Сферы применения систем контроля доступа разнообразны:

* офисы компаний, бизнес-центры;
* банки;
* учреждения образования (школы, техникумы, вузы);
* промышленные предприятия;
* охраняемые территории;
* автостоянки, парковки;
* места проезда автотранспорта;
* частные дома, жилые комплексы, коттеджи;
* гостиницы;
* общественные учреждения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

В данной главе введем основные понятия предметной области, опишем что представляет собой протокол MQTT и как работать с устройством S02T.

* 1. Описание платформы Cantonk S02T

Платформа Cantonk S02T – устройство для измерения температуры с функцией распознавания лица (см. рисунки 1.1 и 1.2). Данное устройство появилось совсем недавно, в 2020 году. В дальнейшем под платформой будем подразумевать ОС с программным обеспечение для устройства. Спецификация приведена в таблице 1.1.

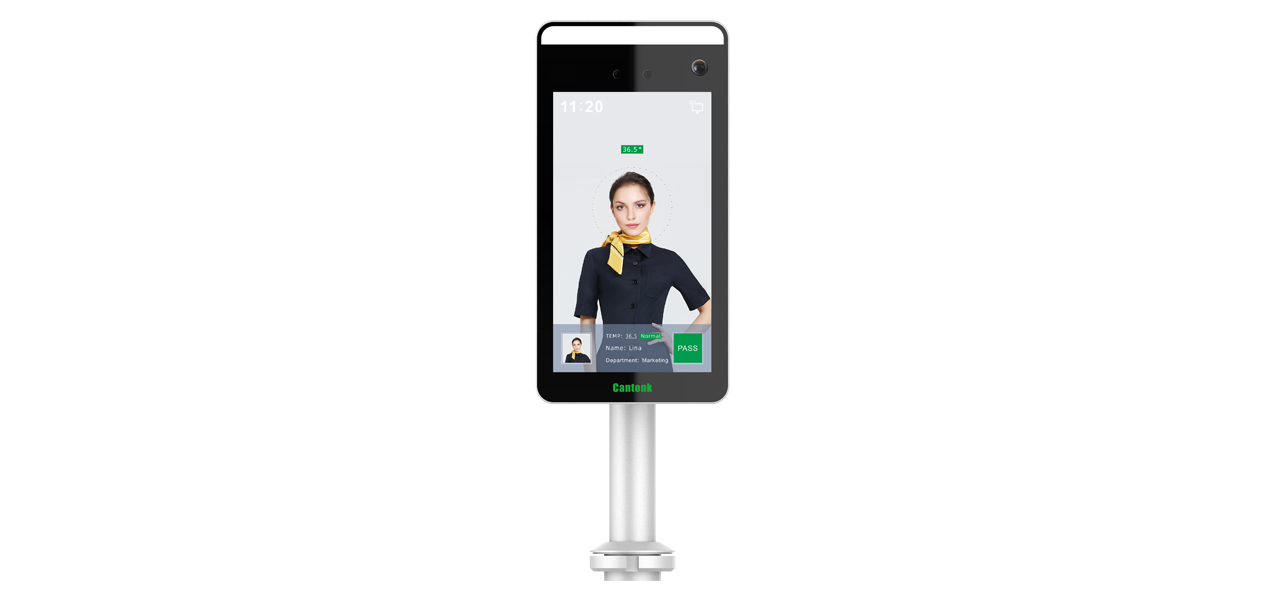


Рисунок 1.1. – Внешний вид устройства S02T

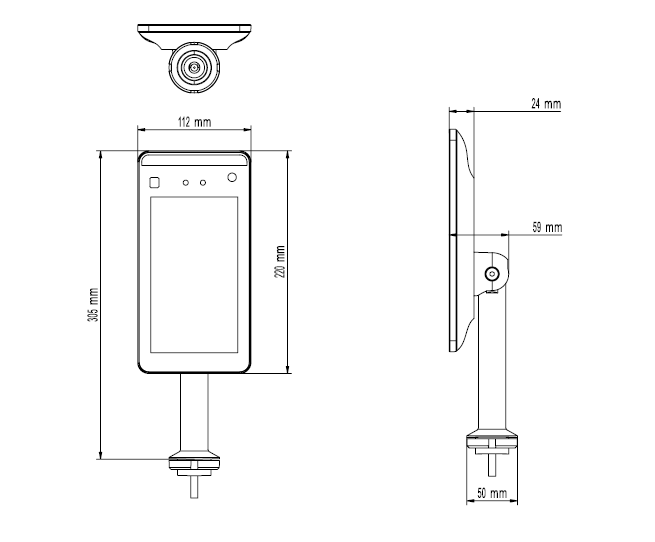


Рисунок 1.2. – Схема устройства S02T

Таблица 1.1. – Спецификация устройства S02T

|  |  |
| --- | --- |
| **Система** | |
| Главный процессор | MX3520 |
| ОС | Встраиваемый LINUX |
| ОЗУ | DDR3 512 MB |
| ПЗУ | EMMC 8 GB |
| **Экран** | |
| Размер | 7-дюймовый ЖК-экран IPS HD |
| Разрешение | 600\*1024 |
| **Камера** | |
| Тип | 2MP камера |
| Сенсор (датчик) | 1/5" GC2145 |
| Разрешение | NIR 800\*600 15fps |
| Объектив | 2.4 мм |
| **Камера** | |
| Тип | Ежедневная камера |
| Сенсор (датчик) | 1/2.8" SONY Starvis Back-illuminated CMOS sensor IMX307 |
| Разрешение | Цветное 1920\*1080 30fps |
| Объектив | 4.5 мм |
| **Определение температуры тела** | |
| Диапазон температуры | 35℃ – 42℃ |
| Расстояние измерения температуры | 0.5 – 1.3 м, 0.75 м – лучше |
| Точность измерения температуры | ± 0.3℃ |
| **Распознавание лица** | |
| Тип обнаружения | Только обнаружение маски |
| Расстояние распознавания лица | 0.5 – 2.5 м, 0.5 – 1.2 м (включение обнаружение температуры) |
| Вместимость БД лиц | Максимальная поддержка 30 000 лиц |
| Поза лица | Вертикально 58° – 60°,  горизонтально 35° |
| Преграда | Обычные очки и короткая челка никак не влияют на узнаваемость |
| Выражение лица | В обычных обстоятельствах легкое выражение лица не влияет на распознавание |
| Скорость отклика | Около 200 мс |
| Экспозиция лица | Поддерживается |
| Локальное хранилище | Поддерживается хранение 25 000 записей |
| Зона распознавания | Полноэкранное распознавание |
| Метод загрузки | TCP, HTTP, MQTT |
| **Сетевые функции** | |
| Сетевой протокол | IPv4, TCP/IP, HTTP, HTTPS, MQTT, RTSP, SMTP, NTP, Onvif |
| Протокол интерфейса | Частный протокол |
| Безопасный режим | Авторизованный пароль |
| Взаимосвязь событий | Голосовая трансляция, загрузка аномальных событий на платформу, тревожный вывод |
| Обновление системы | Поддержка удаленного обновления |
| **Аксессуары** | |
| Дополнительный свет | IR-свет, светодиодный белый свет |
| Модуль идентификации | / |
| Аудио выход | Встроенный динамик поддерживает голосовую трансляцию после успешного распознавания |
| Сетевой модуль | / |
| **Интерфейс** | |
| Сетевой интерфейс | RJ45 10M / 100M сетевая адаптация |
| Тревожный вход | / |
| Тревожный выход | Поддержка 1 канала |
| Релейный выход | Поддерживается |
| Интерфейс RS485 | Поддерживается |
| Интерфейс Wiegand | Поддержка Wiegand 26, 34 протоколов |
| Слот для карт TF | / |
| HDMI | / |
| Кнопка сброса | Поддерживается |
| Переключатель, предотвращающий разборку | / |
| **Общее** | |
| Корпус | Алюминиевый сплав, ультрабелая панель из закаленного стекла, IP66 |
| Рабочая температура | 10 ° C ~ 40° C |
| Рабочая влажность | 10% – 90% без конденсации |
| Температура хранения | -40 ° C ~ 70 ° C |
| Влажность хранения | 5% – 95% без конденсации |
| Уровень защиты | / |
| Источник питания | DC12V |
| Рассеиваемая мощность (максимум) | ≤ 12 W |
| Габаритные размеры (мм) | 220 (W) \* 112 (H) \* 24 (T) |
| Размер кронштейна (мм) | φ27\*189 |
| Способ установки | Настольный монтаж / Напольный монтаж / Монтаж ворот / Установка на колонне / Настенный монтаж |

* 1. Описание протокола MQTT

Одним из способов взаимодействия с устройством является протокол MQTT. MQTT (Message Queuing Telemetry Transport) — упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщений между устройствами по принципу издатель-подписчик (см. рисунок 1.3). Данный протокол был создан еще в 1999 году, но сейчас он особенно актуален благодаря росту IoT: и потребительские, и промышленные устройства внедряют распределённые сети и граничные вычисления (edge computing), а устройства с постоянной трансляцией данных становятся частью повседневной жизни.

Важные термины в MQTT:

* **Издатель (Publisher)**: тот, который публикует сообщения для внешнего мира;
* **Подписчик (Subscriber)**: тот, который получает сообщения, предназначенные для него;
* **Клиент (Client)**: клиент может быть издателем, подписчиком или и тем, и другим. То есть клиент публикует сообщение и одновременно получает другое сообщение;
* **Сервер/Брокер (Server/Broker)**: тот, кто первым получает сообщения, опубликованные издателем, даже до подписчика. Затем сервер публикует сообщения для подписчиков после фильтрации сообщений. И сервер имен, и брокер означают одно и то же;
* **Тема (Topic)**: строка, используемая клиентами и серверами для отправки и получения сообщений. Например: sensors/altimeter/1.

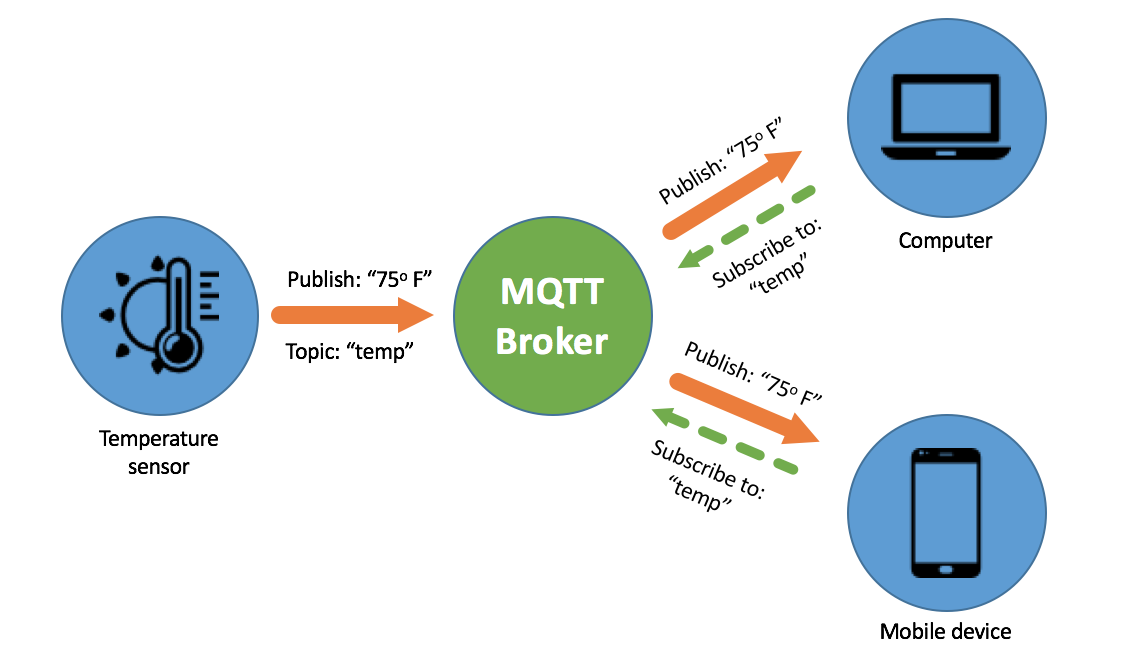


Рисунок 1.3. – Схема работы MQTT

Формат сообщения MQTT (см. рисунок 1.4):

* Байт 1: содержит тип сообщения (запрос клиента на подключение, подтверждение подписки, запрос ping и т. д.), флаг дублирования, инструкции для сохранения сообщений и информацию об уровне качества обслуживания (QoS);
* Байт 2: содержит информацию об оставшейся длине сообщения, включая полезную нагрузку и любые данные в заголовке необязательной переменной.

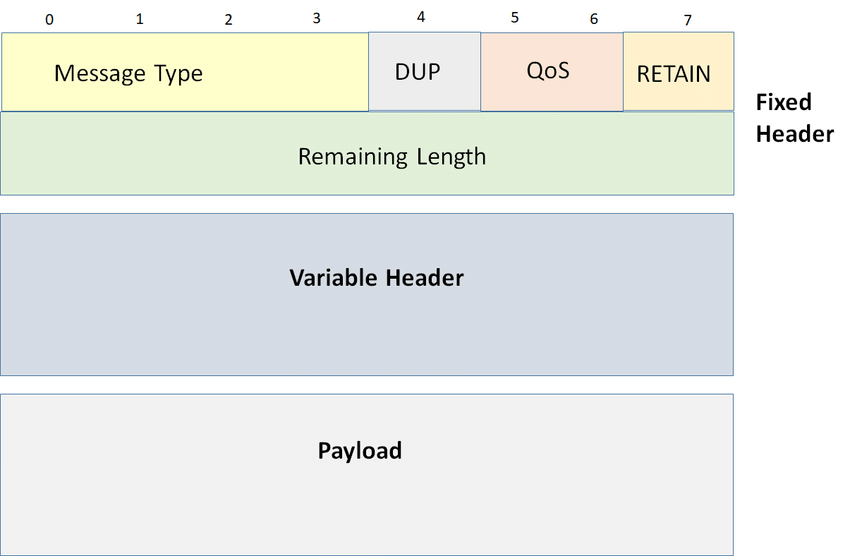


Рисунок 1.4 – Формат сообщения MQTT

Флаги QoS содержат следующие значения, основанные на намерении и срочности сообщения:

* 0 – не более одного раза: сервер срабатывает и забывает. Сообщения могут быть потеряны или продублированы;
* 1 – по крайней мере один раз: получатель подтверждает доставку. Сообщения могут дублироваться, но доставка гарантирована;
* 2 – ровно один раз: сервер обеспечивает доставку. Сообщения поступают точно один раз без потери или дублирования.
  1. Взаимодействие с тепловизором

Программное обеспечение устройства и платформы (Windows) управляется однозначной привязкой. Когда устройство не связано с каким-либо программным обеспечением платформы, информация о персонале в устройстве будет удалена, а информацию о персонале необходимо будет повторно загрузить на устройство после привязки. Обмен данными с устройством осуществляется с помощью текстового формата JSON. Чтобы связаться с устройством нужно отправить следующую информацию ему:

{

"mqtt\_cmd": 1,

"mqtt\_operate\_id": 10,

"device\_id": "7101384284372",

"tag": "platfrom define",

"bind\_ctrl": 1

}

Описание параметров

*mqtt\_cmd* – принимает значение 1 или 2, где:

* 1 – платформа получает системные параметры устройства;
* 2 – платформа настраивает системные параметры устройства.

*mqtt\_operate\_id* – принимает значение 1–10, которое определяет тип команды

*device\_id* – серийный номер устройства

*tag* – тег сообщения, устройство вернет тот же тег, чтобы различать тип операции и тип данных, возвращаемый устройством

*bind\_ctrl* – принимает значение 0 или 1, где:

* 0 – команда на развязывание с устройством;
* 1 – команда на связывание с устройством.

Устройство вернет одно из следующих сообщений, если:

* подключение успешно:

{

"code": 0,

"msg": "mqtt bind ctrl success",

"device\_id": "7101384284372",

"dev\_cur\_pts": 1584053128,

"tag": "platfrom define",

"datas": {

"device\_token":"1057628122"}

}

* ошибка подключения, если устройство уже связано с другой платформой:

{

"code": -1,

"msg": "The device has been bound! ip:192.168.1.88 platfrom:2",

"device\_id": "7101384284372",

"dev\_cur\_pts": 1583988351,

"tag": "platfrom define",

"datas": "nodata"

}

После привязки устройства получим *device\_token*, который служит меткой привязки и нужен для отправки остальных команд:

* добавление информации о пользователе(-ях) и его (их) фотографии:

Платформа отправляет сообщение:

{

"mqtt\_cmd":1,

"mqtt\_operate\_id":7,

"device\_token":"1057628122",

"device\_id":"7101384284372",

"tag":"platfrom define",

"piclib\_manage":0,

"param":

{

"lib\_name":"Smart Park",

"lib\_id":"8",

"server\_ip":"192.168.1.2",

"server\_port":7777,

"pictures":

[{

"active\_time":"2020/01/1 00:00:01",

"user\_id":"11",

"user\_name":"Zhang San",

"end\_time":"2020/12/30 23:59:59",

"p\_id":"null",

"picture":"/192952.JPG"

},

{

"active\_time":"2020/01/1 00:00:01",

"user\_id":"22",

"user\_name":"LI SI",

"end\_time":"2020/12/30 23:59:59",

"p\_id":"null",

"picture":"/linxiaofei.jpg"

}]

}

}

Описание параметров

*server\_ip* – адрес http-сервера, на котором хранится изображение, является IP-адрес или доменным именем.

*active\_time* – начальное время, когда людям разрешено проходить, формат времени: гггг/мм/дд.

*end\_time* – последнее время, когда людям разрешено проходить, формат времени: гггг/мм/дд.

*picture* – абсолютный путь, по которому хранится изображение на http-сервере.

Устройство вернет следующее сообщение:

{

"code":0,

"msg":"download PicLib status",

"device\_id":"7101384284372",

"dev\_cur\_pts":1057628122,

"tag":"platfrom define",

"datas":

[{

"pic\_url":"/192952.JPG",

"picture\_statues":10,

"user\_id":"11"

},

{

"pic\_url":"/linxiaofei.jpg",

"picture\_statues":10,

"user\_id":"22"

}]

}

Если параметр *picture\_statues* имеет значение 10, значит изображение успешно было загружено на устройство, если 20 – ошибка загрузки изображения.

* удаление информации о пользователе(-ях):

Платформа отправляет сообщение:

{

"mqtt\_cmd":1,

"mqtt\_operate\_id":7,

"device\_token":"1057628122",

"device\_id":"7101384284372",

"tag":"platfrom define",

"piclib\_manage":3,

"param":{

"users":[

{

"user\_id":"11"

},

{

"user\_id":"22"

}]

}

}

Устройство вернет следующее сообщение:

{

"code":0,

"msg":"delete users piclib success",

"device\_id":"7101389947744",

"dev\_cur\_pts":1584092989,

"tag":"platfrom define",

"datas":[

{

"user\_id":"11",

"status":0

},

{

"user\_id":"22",

"status":0

}]

}

Если параметр *status* имеет значение 0, значит пользователь успешно был удален, если 1 – ошибка удаления пользователя.

* удаление информации всех пользователей:

Платформа отправляет сообщение:

{

"mqtt\_cmd":1,

"mqtt\_operate\_id":7,

"device\_token":"1057628122",

"device\_id":"7101384284372",

"tag":"platfrom define",

"piclib\_manage":1

}

Устройство вернет одно из следующих сообщений, если:

* удаление успешно:

{

"code":0,

"msg":"delete all piclib success!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583986800,

"tag":"platfrom define",

"datas":"no data"

}

* ошибка удаления:

{

"code":-1,

"msg":"mqtt param erro!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583987981,

"tag":"platfrom define",

"datas":"no data"

}

* запрос информации из БД устройства:

Платформа отправляет сообщение:

{

"mqtt\_cmd":1,

"mqtt\_operate\_id":7,

"device\_token":"1057628122",

"device\_id":"7101384284372",

"tag": "platform define",

"piclib\_manage":4,

"page":0

}

Описание параметров

*page* – значение, равное:

* -1 – запросить общее количество записей информации о пользователях;
* не менее 0 – запросить соответствующую страницу, и на одной странице содержится 100 единиц данных. Например, чтобы запросить страницу данных 501-600, чтобы заполнить 5.

Устройство вернет одно из следующих сообщений, если:

* *page* имеет значение -1:

{

"code":0,

"msg":"mqtt query success!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1590480998,

"tag":"platfrom define",

"datas":{

"total\_num":30102

}

}

* *page* имеет не менее 0:

{

"code":0,

"msg":"mqtt query success!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583987504,

"tag":"platfrom define",

"datas":[

{

"lib\_id":"8",

"lib\_name":"Smart Park",

"user\_id":"1",

"user\_name":"Zhang San",

"active\_time":"2020/01/1 00:00:01",

"end\_time":"2020/12/30 23:59:59",

},

{

"lib\_id":"9",

"lib\_name":"Smart Park",

"user\_id":"2",

"user\_name":"Li Si",

"active\_time":"2020/01/1 00:00:01",

"end\_time":"2020/12/30 23:59:59",

}

]

}

* настройка параметров контроля температуры:

Платформа отправляет сообщение:

{

"mqtt\_cmd":1,

"mqtt\_operate\_id":6,

"device\_token":"1057628122",

"device\_id":"7101384284372",

"tag":"platfrom define",

"temperature\_fun":

{

"temp\_dec\_en":false,

"stranger\_pass\_en":false,

"make\_check\_en":false,

"alarm\_temp":37.4,

"temp\_comp":1.1,

"record\_save\_time":24,

"save\_record":true,

"save\_jpeg":true

}

}

Устройство вернет одно из следующих сообщений, если:

* параметры контроля приняты:

{

"code":0,

"msg":"funtable param set success",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583987966,

"tag":"platfrom define",

"datas":"no data"

}

* параметры контроля не приняты:

{

"code":-1,

"msg":"mqtt param erro!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583987981,

"tag":"platfrom define",

"datas":"no data"

}

* и иные команды.

Также устройство будет возвращать сообщение, после аутентификации и идентификации людей / незнакомцев:

{

"code":0,

"msg":"Upload Person Info!",

"device\_id":"7101384284372",

"dev\_cur\_pts":1583996959,

"tag":"UploadPersonInfo",

"datas":

{

"msgType":"1",

"similar":"0.98",

"user\_id":"77",

"name":"xiaoming",

"time":"2020/05/20 20:00:00",

"temperature":"36.60",

"matched":"1",

"imageFile":"data:image/jpg;base64,/9j/4........."

}

}

Описание параметров

*msgType* – тип сообщения, еще не определен, можно позволить устройству возвращает указанное значение.

*similar* – процент схожести с пользователем.

*user\_id* – пользовательский идентификатор, если незнакомец (нет человека в БД устройства), то значение будет пустым.

*name* – имя пользователя, если незнакомец (нет человека в БД устройства), то значение будет пустым.

*time* – дата и время прохода человека.

*temperature* – температура человека.

*matched* – определяет нахождение человека в БД устройства (1 – в БД находится, 0 – незнакомец).

*imageFile* – снимок человека, зашифрованный с помощью base64. Данный параметр появляется только если на устройстве параметр *save\_jpeg* имеет значение true [1].

* 1. Постановка задачи

В рамках данного дипломного проекта требуется разработать ПО для контроля доступа с поддержкой измерения температуры.

ПО должно обеспечивать:

1. Взаимодействие с устройствами.

2. Изменение настроек устройства.

3. Ввод, редактирование данных о человеке.

4. Отправка данных о человеке на устройство.

5. Сбор статистики человека и наглядная демонстрация ее.

6. Формирование отчетов по статистике.

1. ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ

В этой главе спроектируем ПС. Для этого опишем их общую структуру, а также более подробно рассмотрим проектирование подсистемы визуализации. Определим генетический алгоритм для решения ОКЗ. Кроме того, выберем средства программирования для реализации наших спроектированных ПС.

* 1. Общее описание структуры ПС

Необходимо спроектировать такую структуру ПС, которая удовлетворяла бы всем поставленным задачам. Для этого последовательно выделим все подсистемы, способные их решить.

Вся работа основана на взаимодействии с устройствами, поэтому нам необходимо иметь подсистему «Управление Устройствами», в задачи которой будет входить хранение характеристик обнаруженных устройств, поиск устройств, возможность их изменения и получения.

Поскольку взаимодействие устройства происходит с помощью протокола MQTT, выделим подсистему «Платформа Публикации», которая нужна для отправки сообщений устройству, а также подсистему «Платформа Подписки», которая нужна для сбора сообщений: статистика, представляющая собой аутентификацию и идентификацию людей / незнакомцев, результаты отправки сообщений на устройства.

Для хранения и извлечения всех данных об устройствах, статистике, пользователях нужно иметь БД, для этого выделим подсистему «Управление БД».

Статистика должна быть визуализирована, поэтому выделим подсистему «Визуализатор БД». Данная подсистема будет взаимодействовать с подсистемой «Управление БД»: получать нужные данные для отображения.

Взаимодействие с ПО будет происходить с помощью подсистемы «Визуализатор», которая будет отображать действия с подсистемами: «Управление БД», «Управление Устройствами», «Визуализатор БД».

Взаимодействие этих подсистем представлено на рисунке 2.1.

Далее необходимо выделить подсистемы для решения ОКЗ с помощью генетических алгоритмов.

Для решения ОКЗ введем подсистему «Решатель ОКЗ», которая будет взаимодействовать с подсистемой «Рука-Манипулятор», т.е. получать и изменять характеристики робота-манипулятора (его пространственное положение).

Также нам необходимо знать конечную точку, которую должен достичь манипулятор, и алгоритм, с помощью которого будет решаться ОКЗ. Для этого выделим подсистему «Ввод данных», которая будет доступна из подсистемы «Визуализатор».



Рисунок 2.1 – Взаимодействие подсистем «Рука-Манипулятор», «Симулятор», «Визуализатор» и «Редактирование параметров»

Поскольку стоит задача решения ОКЗ с помощью ГА, наше ПО должно включать подсистему «ГА», данные которой будет передавать «Решатель ОКЗ». В свою очередь, подсистема «ГА» будет передавать результат подсистеме «Решатель ОКЗ».

Для преобразования данных, необходимых для работы ГА, выделим подсистему «Конвертер для ГА».

При работе с ГА важно увидеть статистические данные, которые будут показывать работоспособность нашего подхода при решении ОКЗ. Поэтому выделим подсистему «Сбор статистики».

Для тестирования алгоритмов введем подсистему «Тестирование». Тестирование будет заключаться в запуске ГА заданное количество раз со сбором статистики по каждому запуску. Будем осуществлять тестирование алгоритмов с получением данных о времени работы алгоритма и ошибки вычислений, а также тестирование зависимости времени выполнения алгоритма от желаемой точности вычислений.

Уточненная схема взаимодействия подсистем ПО представлена на рисунке 2.2.

Интересной является задача движения робота-манипулятора по заданной траектории. Чтобы проверить возможность использования ГА в данной задаче, выделим подсистему «Динамика», где будем рассчитывать движение по прямой и по кривой, обозначающей знак бесконечности.

Поскольку для движения необходимо разбивать заданную траекторию на точки, выделим подсистему «Конвертер траектории».



Рисунок 2.2 – Уточненная схема взаимодействия подсистем ПО

Взаимодействие с роботом-манипулятором RoboArm обеспечим через подсистему «Управление RoboArm». В ее задачи будет входить преобразование решения ОКЗ в набор команд, понятных роботу-манипулятору, передача этих команд сервомоторам, а также настройка соответствия сервомоторов и звеньев модели робота-манипулятора. Подключение к RoboArm осуществляется через COM-порт. Для этой задачи выделим подсистему «Подключение через COM-порт».

Итоговая схема взаимодействия подсистем ПО представлена на рисунке 2.3.

Рисунок 2.3 – Схема взаимодействия подсистем ПО

* 1. Проектирование ПС

Необходимо обеспечить взаимодействие пользователя с нашим ПО. Для этого спроектируем интерфейс системы.

При запуске приложения будет появляться главное окно. Модель робота-манипулятора будет строиться на полный размер окна. Пользователь должен иметь возможность вводить данные (желаемую точку схвата), поэтому наше главное окно должно содержать поля ввода этой точки. Кроме того, нужно обеспечить выбор алгоритма для решения ОКЗ. Результат решения ОКЗ будет выводиться в главном окне. Для этого добавим поля для его вывода. Причем будем выводить как полученные углы наклонов звеньев, так и конечную точку робота-манипулятора. Также после работы ГА должна выводиться статистическая информация и график изменения ошибки на каждой итерации ГА. Поскольку вся эта информация будет выводиться в главном окне, обеспечим возможность сокрытия каждого блока при желании пользователя. Для этого добавим пункт меню «Вид», в котором можно будет отметить все желаемые для отображения блоки данных. Статистическая информация будет скрываться/появляться при нажатии на графический элемент рядом с ней.

Для того, чтобы графические элементы не закрывали модель робота-манипулятора, добавим к ним свойства прозрачности.

В панель меню добавим пункты «Тестирование» (тестирования ГА), «Настройка» (редактирование параметров робота-манипулятора), «Движение по траектории» (задание траектории движения робота-манипулятора), «RoboArm» (подключение к RoboArm, отправка команд).

Дизайн главного окна приложения представлен на рисунке 2.4.

Пункт меню «Тестирование» будет включать два подпункта: «Время, итерации», «Время/точность».

При нажатии на подпункт «Время, итерации» будет появляться окно для тестирования времени работы ГА и количества итераций, потребовавшихся на решение ОКЗ (рисунок 2.5). В нем будут содержаться поля ввода конечной точки манипулятора, количества тестов; выбор ГА; вывод статистических данных. Кроме того, в данном окне можно будет изменить параметры тестируемого манипулятора, что не будет влиять на манипулятор, представленный в главном окне приложения. Также обеспечим возможность просмотра подробного отчета по каждому из тестов (рисунок 2.6).

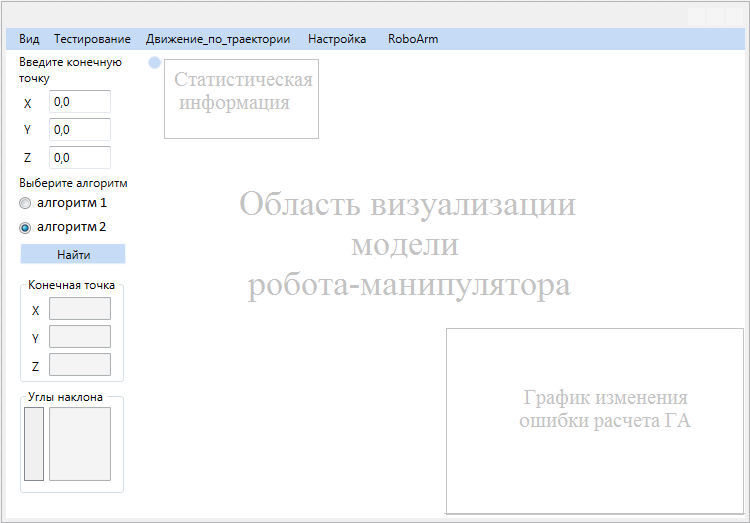


Рисунок 2.4 – Дизайн главного окна приложения

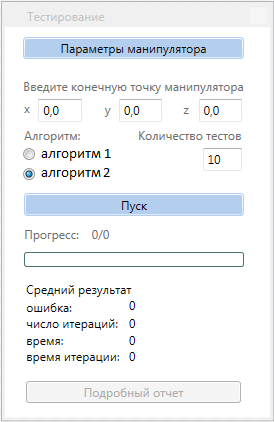


Рисунок 2.5 – Дизайн окна тестирования «Тестирование»

При нажатии на подпункт «Время/точность» будет появляться окно для тестирования зависимости времени работы ГА от точности вычисления ОКЗ (рисунок 2.7). В нем должна быть обеспечена возможность ввода конечной точки манипулятора, количества тестов, а также выбор ГА.

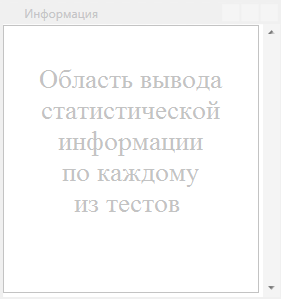


Рисунок 2.6 – Дизайн окна вывода информации по каждому из тестов

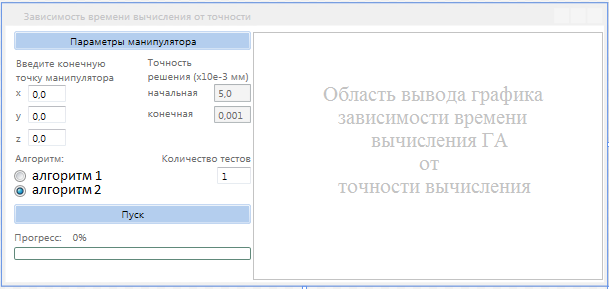


Рисунок 2.7 – Дизайн окна тестирования «Зависимость времени вычисления от точности»

Пункт меню «Настройка» будет включать в себя подпункт «Параметры манипулятора», при нажатии на который будет выводиться окно редактирования параметров робота-манипулятора (рисунок 2.8). В нем будут отображаться текущие характеристики робота-манипулятора. Кроме того, данное окно будет содержать поля для добавления, изменения и удаления звеньев, а также возможность изменения точности вычисления ОКЗ.

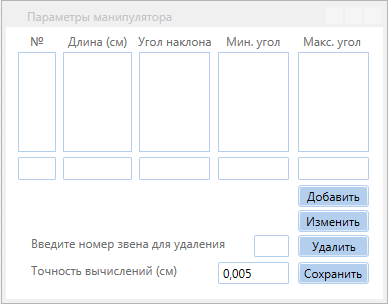


Рисунок 2.8 – Дизайн окна редактирования параметров манипулятора «Параметры манипулятора»

При нажатии на пункт меню «Движение по траектории» будет отображаться окно «Параметры движения» для настройки параметров движения робота-манипулятора RoboArm (рисунок 2.9). Данное окно позволит выбрать траекторию (прямая, знак бесконечности), а также задать начальную (для обеих траекторий) и конечную (для прямой) точки. Также обеспечим возможность выбора количество точек, на которые будет разбиваться траектория.

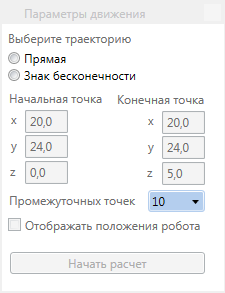


Рисунок 2.9 – Дизайн окна «Параметры движения»

Добавим окно «Графики изменения углов наклона звеньев» (рисунок 2.10) для визуализации изменения углов наклона и возможности их анализа.

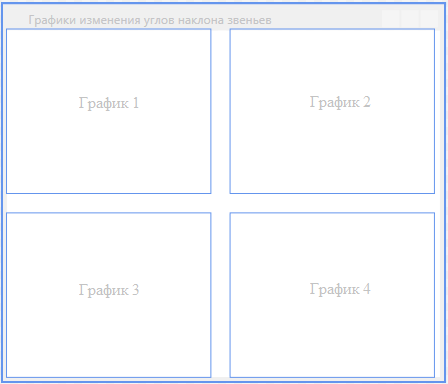


Рисунок 2.10 – Дизайн окна «Графики изменения углов наклона звеньев»

В пункт меню RoboArm включим подпункт «Подключиться через COM-порт», при нажатии на который откроется окно подключения к RoboArm (рисунок 2.11). В нем можно будет настроить параметры порта.

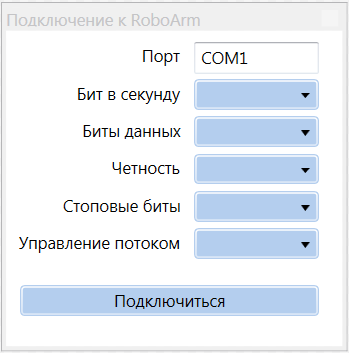


Рисунок 2.11 – Дизайн окна «Подключение к RoboArm»

При удачном подключении отобразится окно «Управление RoboArm» (рисунок 2.12). В нем будет обеспечена возможность манипулирования звеньями RoboArm. Также в данном окне будут отображаться команды, которые будут посылаться на контроллер.

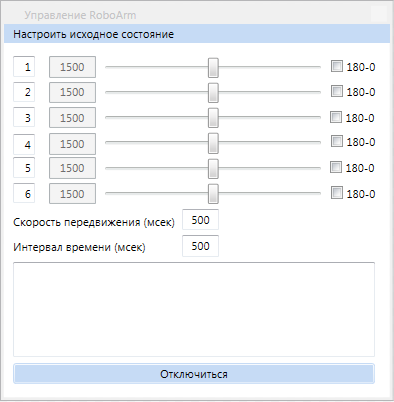


Рисунок 2.12 – Дизайн окна «Управление RoboArm»

Приведем варианты использования системы в виде диаграммы вариантов использования. Эта диаграмма описывает взаимоотношения и зависимости между группами вариантов использования и действующими лицами, которые принимают участие в процессе. Действующее лицо – это роль, какую пользователь играет по отношению к системе. Диаграмма вариантов использования данной системы представлена на рисунке 2.13.

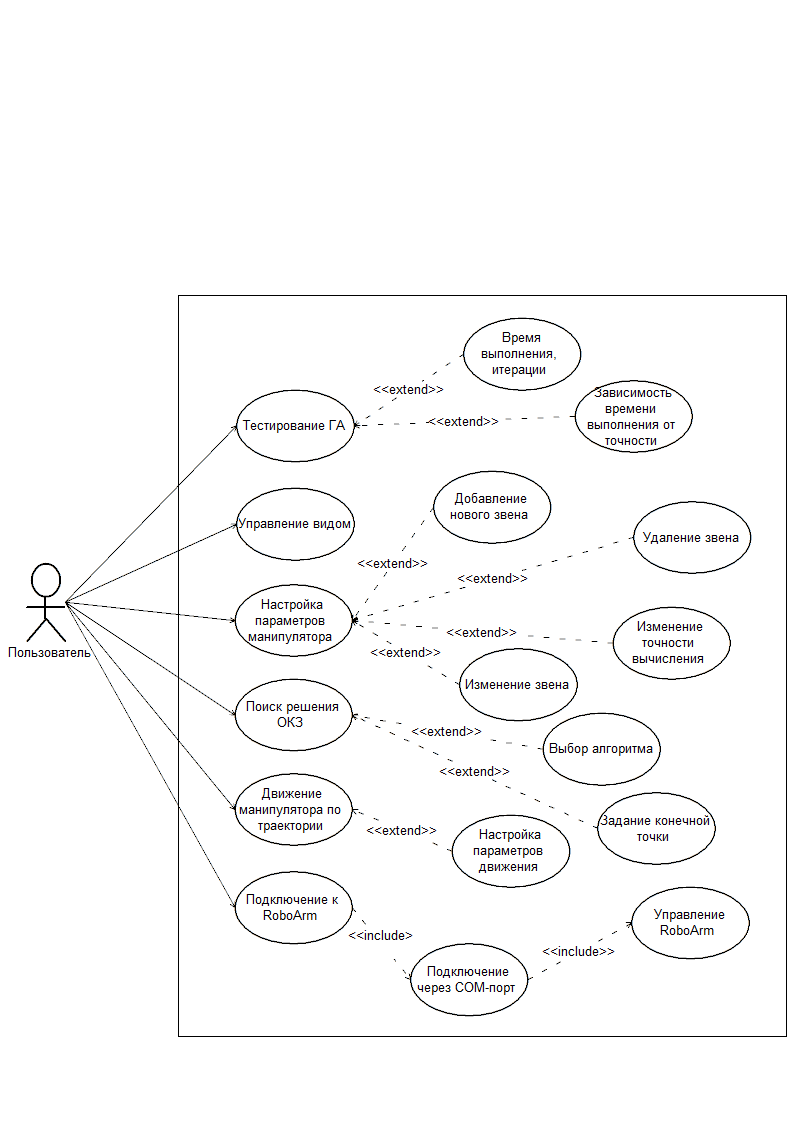


Рисунок 2.13 – Диаграмма вариантов использования

Приведем текстовое описание основных прецедентов:

Прецедент *«Поиск решения ОКЗ»:*

1. Пользователь вводит желаемую конечную точку робота-манипулятора и выбирает алгоритм решения ОКЗ (если данные некорректны – п. 1.1, иначе п. 2).
   1. Выводится сообщение о некорректном вводе.
2. Система решает ОКЗ, перестраивает модель робота-манипулятора и выводит результат.

Прецедент *«Настройка параметров манипулятора»:*

1. Пользователь выбирает пункт меню «Настройка» подпункт «Параметры манипулятора».
2. Выводится окно настройки параметров манипулятора.
3. Пользователь вводит данные для добавления (если данные некорректны – п. 3.1, если корректны – п. 3.2).
   1. Выводится сообщение о некорректном вводе.
   2. Система добавляет новое звено к манипулятору.
4. Пользователь изменяет параметры звена манипулятора (если данные некорректны – п. 4.1, если корректны – п. 4.2).
   1. Выводится сообщение о некорректном вводе.
   2. Система изменяет параметры звена манипулятора.
5. Пользователь вводит номер звена для удаления (если данные некорректны – п. 5.1, если корректны – п. 5.2).
   1. Выводится сообщение о некорректном вводе.
   2. Система удаляет звено из цепи робота-манипулятора.
6. Пользователь вводит новую точность вычисления решения ОКЗ и сохраняет ввод.

Прецедент *«Тестирование ГА»:*

1. Пользователь выбирает пункт меню «Тестирование» подпункт «Время, итерации».
2. Выводится окно «Тестирование».
   1. Пользователь вводит конечную точку манипулятора и количество тестов (если ввод некорректен – п. 2.1.1, иначе – п. 2.2).
      1. Выводится сообщение о некорректном вводе.
   2. Система проводит тесты, выводит результат пользователю.
   3. Пользователь выбирает просмотр подробной информации о тестах.
   4. Выводится окно с информацией о тестах.
3. Пользователь выбирает пункт меню «Тестирование» подпункт «Время/точность».
4. Выводится окно «Зависимость времени вычисления от точности».
   1. Пользователь вводит конечную точку манипулятора и количество тестов (если ввод некорректен – п. 4.1.1, иначе – п. 4.2).
      1. Выводится сообщение о некорректном вводе.
   2. Система проводит тесты, выводит результат пользователю.

Прецедент *«Управление видом»:*

1. Пользователь выбирает пункт меню «Управление видом».
2. Пользователь выбирает подпункт «Конечная точка» (если пункт был выбран до этого – п. 2.1, если нет – п. 2.2).
   1. С главного окна скрывается отображение конечной точки манипулятора.
   2. На главном окне отображается конечная точка манипулятора.
3. Пользователь выбирает подпункт «Углы наклона» (если пункт был выбран до этого – п. 3.1, если нет – п. 3.2).
   1. С главного окна скрывается отображение углов наклона звеньев манипулятора.
   2. На главном окне отображается углы наклона звеньев манипулятора.
4. Пользователь выбирает подпункт «График ошибки» (если пункт был выбран до этого – п. 4.1, если нет – п. 4.2).
   1. С главного окна скрывается поле для вывода графика ошибки решения ОКЗ в процессе работы ГА.
   2. На главном окне отображается поле для вывода графика ошибки решения ОКЗ в процессе работы ГА.

Прецедент *«Движение по траектории»:*

1. Пользователь выбирает пункт меню «Движение по траектории».
2. Открывается окно настройки параметров движения.
3. Пользователь выбирает траекторию.
4. Пользователь вводит начальную и/или конечную точки (если ввод некорректный – п. 4.1), выбирает число точек, на которые будет разбиваться траектория и нажимает кнопку «Начать расчет».
   1. Выводится сообщение о некорректном вводе.
5. Система решает ОКЗ для движения по траектории, выводит результат в главное окно.

Прецедент *«Подключение к RoboArm»:*

1. Пользователь выбирает пункт меню «RoboArm» подпункт «Подключиться через COM-порт».
2. Открывается окно подключения.
3. Пользователь вводит имя порта, настраивает подключение и нажимает «Подключиться» (если не удалось подключиться – п. 3.1).
   1. Выводится сообщение о том, что подключиться к введенному порту не удалось.
4. Отображается окно «Управление RoboArm».
5. С помощью графических элементов пользователь управляет роботом-манипулятором RoboArm.
6. Пользователь выбирает пункт меню «Настроить исходное состояние».
7. С помощью графических элементов настраивает исходное состояние RoboArm и нажимает кнопку «Сохранить как исходное состояние».

При помощи диаграммы классов представим структуру модели системы в терминологии объектно-ориентированного программирования. Иерархия классов интерфейса приложения представлена на рисунке 2.14.

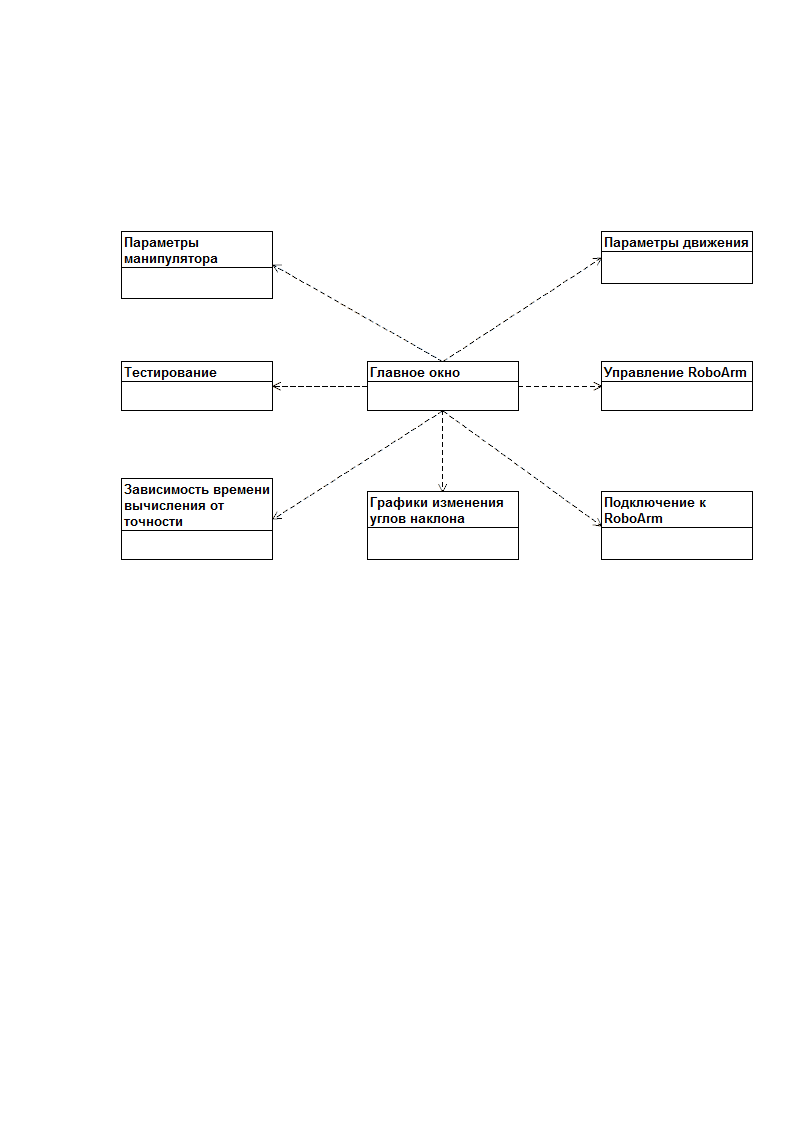


Рисунок 2.14 – Иерархия классов интерфейса приложения

Подсистему «Рука-манипулятор» спроектируем в виде классов HandManipulator и Link. Link будет описывать звено манипулятора. Его атрибутами будут длина (length), максимальный угол наклона (maxTheta), минимальный угол наклона (minTheta), текущий угол наклона (theta). HandManipulator будет описывать непосредственно сам манипулятор. Для этого его атрибутом будут звенья (links). Кроме того, в нем будут содержаться методы для добавления (add(link)), изменения (changeAt(index, link)) и удаления звена (removeAt(index)). Также данный класс будет предоставлять решение прямой кинематической задачи (endPoint()). Диаграмма классов данной подсистемы изображена на рисунке 2.15.



Рисунок 2.15 – Первоначальная диаграмма классов подсистемы «Рука-манипулятор»

Подсистему «Симулятор» представим в виде класса RoboArmSimulator. Его структура будет зависеть от выбранных средств программирования. В общем случае, данный класс должен содержать методы для перемещения манипулятора (move()) и получения построенной модели (visualModel()). Диаграмма данного класса изображена на рисунке 2.16.



Рисунок 2.16 – Первоначальная диаграмма классов подсистемы «Симулятор»

Подсистему «Решатель ОКЗ» представим в виде класса InverseKinematicProblem. Данный класс должен содержать метод решения ОКЗ по заданному алгоритму с заданной точностью для одной точки (solve(point, algorithm, accuracy)), а также метод решения ОКЗ для перемещения манипулятора по траектории (solveDynamic(lineToPoints, accuracy)). Диаграмма данного класса изображена на рисунке 2.17.



Рисунок 2.17 – Первоначальная диаграмма классов подсистемы «Решатель ОКЗ»

* 1. Генетический алгоритм решения обратной кинематической задачи

Решать ОКЗ будем с помощью двух генетических алгоритмов. Первый из них – традиционный (стандартный) генетический алгоритм. Далее опишем некоторые подробности работы этого алгоритма.

Хромосомы представим в виде генов, являющихся углами наклона звеньев руки-манипулятора, т.е. хромосома – набор вещественных чисел. Начальная популяция создается случайным образом. В отборе родителей для скрещивания участвует половина лучших особей, в соответствии с функцией пригодности каждой из них. Пригодность особи – евклидово расстояние между конечной точкой руки-манипулятора (точкой схвата) и целевой точкой. Конечная точка – решение прямой кинематической задачи для текущего пространственного положения манипулятора. Перед отбором родителей популяция сортируется в соответствии с функцией пригодности.

В качестве оператора скрещивания будем использовать дискретную рекомбинацию. Дискретная рекомбинация в основном применяется к хромосомам с вещественными генами. Ее основными способами являются собственно дискретная рекомбинация, промежуточная, линейная и расширенно линейная рекомбинация.

Поскольку хромосома состоит из вещественных чисел, будем использовать промежуточную рекомбинацию, которая применяется только к вещественным переменным. В данном методе предварительно определяется числовой интервал значений генов потомков, который должен содержать значения генов родителей. Потомки создаются по следующему правилу:

Потомок1 = Родитель1 + α\*(Родитель2 – Родитель1);

Потомок2 = Родитель2 + α\*(Родитель1 – Родитель2),

где множитель α – случайное число на отрезке [-d, 1 + d], d ≥ 0.

Как отмечают сторонники этого метода, наиболее оптимальное воспроизведение получается при d = 0,25. Для каждого гена создаваемого потомка выбирается отдельный множитель α. При промежуточной рекомбинации возникают значения генов, отличные от значения генов особей родителей. Это приводит к возникновению новых особей, пригодность которых может быть лучше, чем пригодность родителей. В литературе такой оператор рекомбинации иногда называется дифференциальным скрещиванием.

Мутацию генов осуществлять не будем. После применения оператора скрещивания к двум родителям выберем одного наиболее пригодного потомка.

Отбор и скрещивание родителей происходит столько раз, какого размера должна быть новая популяция (n). В нашем случае размер популяции изменяться не будет. Лучшим решением всегда будет первая особь в популяции, поскольку популяция сортируется.

Алгоритм работает до тех пор, пока пригодность лучшей особи (bestFit) не будет меньше некоторого заданного значения ошибки (err) или пока общее количество популяций (pps) не достигнет определенного значения (lim).

Блок-схема работы данного алгоритма представлена на рисунке 2.18.



Рисунок 2.18 – Схема работы стандартного ГА

Второй алгоритм – Hill climbing – поиск с восхождением к вершине. В нем постоянно происходит перемещение в направлении возрастания некоторого значения, т.е. подъем.  Работа этого алгоритма заканчивается после достижения "пика", в котором ни одно из соседних состояний не имеет более высокого значения. В нем не осуществляется прогнозирование за пределами состояний, которые являются непосредственно соседними по отношению к текущему состоянию.

Поиск с восхождением к вершине иногда называют жадным локальным поиском, поскольку в процессе его выполнения происходит захват самого хорошего соседнего состояния без предварительных рассуждений о том, куда следует отправиться дальше. Во время такого поиска зачастую происходит очень быстрое продвижение в направлении решения, т. к. обычно бывает чрезвычайно легко улучшить плохое состояние.

С точки зрения ГА, этот алгоритм заключается в постоянной мутации одной особи. Мутация принимается, если приспособленность особи улучшается, в противном случае, особь остается в первоначальном состоянии (состоянии до мутации). Данный алгоритм быстро приближается к целевой точке, однако может долго блуждать в ее окрестностях.

В отличие от стандартного алгоритма, будем представлять хромосому в двоичном виде. Размер хромосомы будет рассчитываться следующим образом: угол наклона каждого звена представляется в двоичном виде, причем 9 разрядов (генов) выделяется под целую часть (достаточно для представления 360◦) и 30 разрядов под вещественную часть (точность 1\*10-9). Таким образом, длина хромосомы – N\*39, где N – количество звеньев руки-манипулятора.

На каждой итерации хромосома мутирует, пока ее приспособленность не улучшится или пока количество мутаций (mCounter) не достигнет заданного значения (nм). Это ограничение вводится для того, чтобы алгоритм не зацикливался.

Алгоритм работает до тех пор, пока пригодность особи (fit) не будет меньше некоторого заданного значения ошибки (err). Также для ограничения времени работы алгоритма будем останавливать его, если количество поколений (pps) достигнет определенного значения (lim).

Данный алгоритм требует больших вычислительных затрат, поскольку необходимо преобразовывать хромосому, представленную в двоичном виде, в вещественные числа для определения приспособленности.

Схема работы данного алгоритма представлена в приложении А на рисунке А.1.

Для решения задачи перемещения манипулятора по траектории будем использовать традиционный ГА. Для каждой точки, на которые будет разбита траектория, решим ОКЗ. Причем, для того чтобы движения манипулятора были более плавными, будем добавлять дополнительные ограничения на углы звеньев для следующей точки, на основе того, в каких четвертях находились углы звеньев для предыдущей точки. Если решить ОКЗ для текущей точки с дополнительными ограничениями не удастся, будем их убирать. Блок-схема для решения данной задачи представлена в приложении А на рисунке А.2.

* 1. Выбор средств программирования

В качестве языка программирования был выбран язык C#. C# — типобезопасный объектно-ориентированный язык, предназначенный для разработки разнообразных приложений, выполняемых в среде .NET Framework.  .NET Framework – интегрированный компонент Windows, содержащий виртуальную систему выполнения (среда CLR) и унифицированный набор библиотек классов. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения баз данных и т. д.

Среда программирования – Microsoft Visual Studio 2015 (MVS), включающая [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8" \o "Интегрированная среда разработки) программного обеспечения и ряд других инструментальных средств. MVS позволяет разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F" \o "Текстовый интерфейс пользователя) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F" \o "Графический интерфейс пользователя), в том числе с поддержкой технологии [WPF](https://ru.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms) (Windows Presentation Foundation). MVS включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0" \o "Редактор исходного кода) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense" \o "IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3" \o "Рефакторинг). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)" \o "Класс (программирование)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85" \o "Схема базы данных). MVS позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD" \o "Плагин)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8" \o "Система управления версиями), добавление новых наборов инструментов или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F).

Для визуализации данных будем использовать WPF. WPF ― это система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С помощью WPF можно создавать широкий спектр как автономных, так и размещенных в браузере приложений. В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, в том числе Extensible Application Markup Language, элементами управления, привязкой данных, макетом, 2-D- и 3-D-графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. WPF входит в состав Microsoft .NET Framework и позволяет создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

Для построения и визуализации модели робота-манипулятора воспользуемся встроенными средствами WPF приложений – классами, содержащимися в пространстве имен System.Windows.Media.Media3D.

Для построения графиков воспользуемся библиотекой OxyPlot.

1. РЕАЛИЗАЦИЯ И ИСПЫТАНИЕ ПС
   1. Реализация ПС

Реализованное ПО запускается в ОС Windows 7 и выше с .NET версии не менее 4.5.2. Взаимодействие с реальным роботом-манипулятором может осуществляться через интерфейс RS-232. Расположение физических узлов системы изображено на рисунке 3.1.

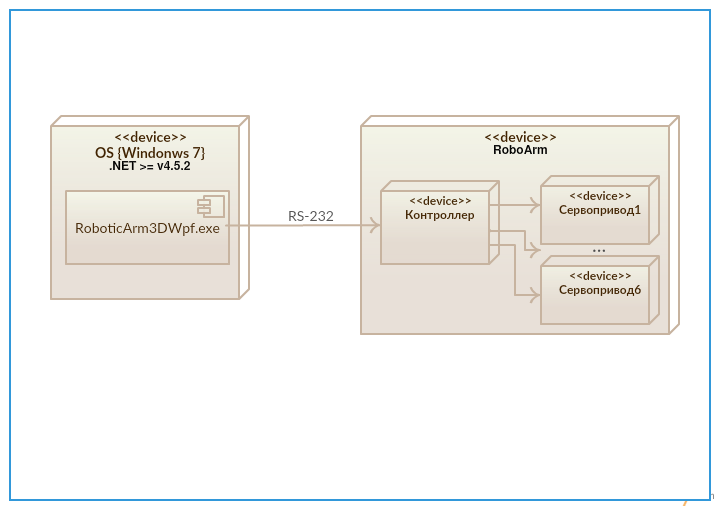


Рисунок 3.1 – Физические узлы системы

Покажем разбиение системы на структурные и функциональные компоненты и их развертывание на описанной инфраструктуре при помощи диаграммы компонентов (рисунок 3.2).

Компонент RoboArm3DWpf является исполняемым файлом, запустив который можно работать с системой. В его состав входят следующие компоненты:

* RoboticArmWpf – содержит в себе:
  + MainWindow – реализует главное окно приложения, с помощью которого осуществляется взаимодействие со всеми подсистемами.
  + RoboArmSimulator – построение 3D модели робота-манипулятора, а также визуализация работы ГА (отображение конечной точки робота-манипулятора в процессе решения ОКЗ).
  + InverseKinematicProblem – решение ОКЗ, взаимодействуя с ГА.
* TestArm – тестирование ГА.
* ManipulatorParameters – изменение параметров модели робота-манипулятора.
* Plot – построение графиков результатов работы ГА, расчета ОКЗ для движения по траектории, тестирования.
* Dynamic – настройка параметров движения робота-манипулятора по траектории.
* RoboArm – взаимодействие с реальным роботом-манипулятором RoboArm.

В виде библиотек реализованы компоненты:

* GeneticAlgorithmsLib – ГА для решения ОКЗ.
* StatisticGALib – сохранение и получение статистической информации о работе ГА.
* RoboticArmLib – описание робота-манипулятора.
* DataConverterLib – преобразование данных для работы ГА.

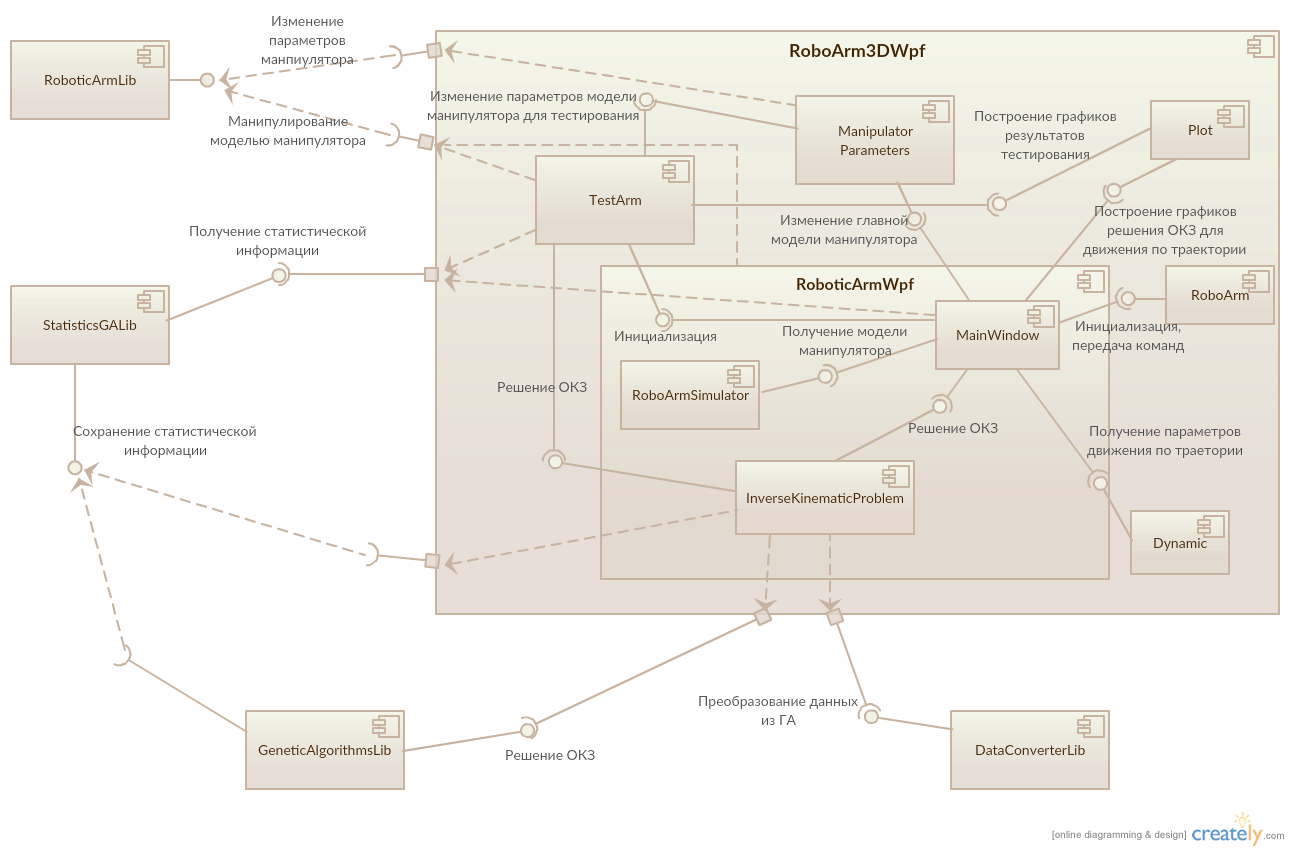
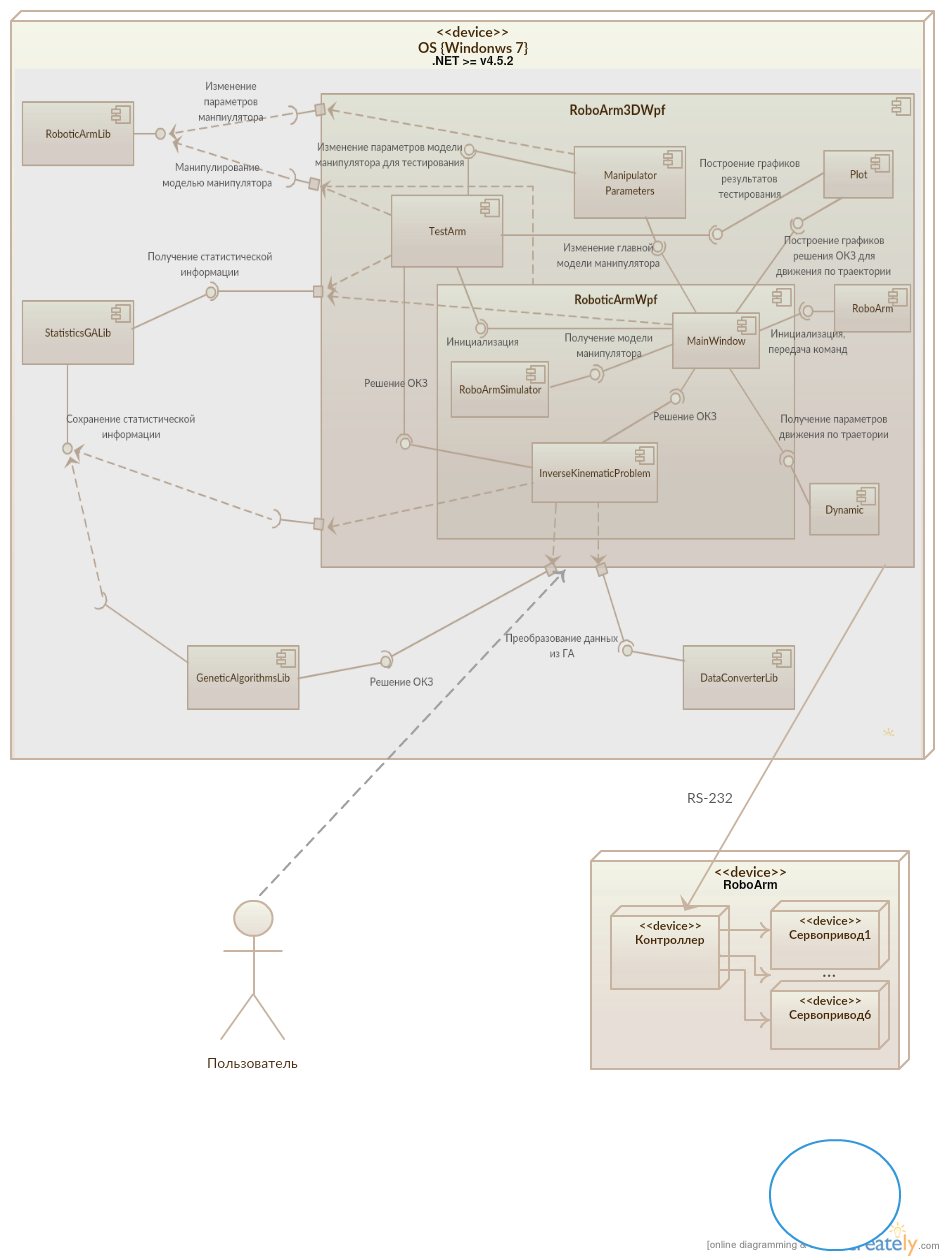


Рисунок 3.2 – Диаграмма компонентов

Отобразим наложение разработанных компонентов на физические узлы при помощи диаграммы развертывания (рисунок 3.3).

Рисунок 3.3 – Диаграмма развертывания

Опишем некоторые подсистемы, реализованные в виде изображенных выше компонентов.

Библиотека подсистемы «Рука-Манипулятор» (HandManipulatorLib.dll) реализована в виде двух классов: Link – для представления звена манипулятора, HandManipulator – для представления непосредственно руки-манипулятора. Диаграмма классов данной библиотеки показана на рисунке 3.4.

В языке C# все пользовательские классы наследуются от класса Object. Класс Object предоставляет базовые методы с возможностью из переопределения (такие, как ToString(), Equals(), GetHashCode() и др.). На диаграммах классом данный класс изображать не будем.

Опишем основные элементы данных классов.

Класс *Link*:

* double length – длина звена;
* double maxTheta – максимальный угол наклона звена;
* double minTheta – минимальный угол наклона звена;
* Point3D – точка начала соединения текущего звена;
* double theta – текущий угол наклона звена;
* double RotatingAngle – свойство для установления и получения значения угла поворота системы координат для преобразования двумерной системы в трехмерную и в наоборот;
* Vector3D RotatedVector(double prevTheta) – поворот звена на угол RotatedAngle с учетом значений углов поворота предыдущих звеньев для перехода из двумерного пространства в трехмерное;
* Vector3D Vector2D(double prevTheta) – представление звена в виде вектора в двумерном пространстве.

Класс *HandManipulator:*

* List<Link> links – звенья руки-манипулятора.
* void Add(Link link) – добавление звена к руке-манипулятору;
* void ChangeAt(int index, Link link) – замена звена с индексом index в List<Link> links на звено link;
* void ChangeThetas(List<double> thetas) – изменение углов наклона звеньев, где thetas – новые углы наклона;
* void RemoveAt(int index) – удаление звена из кинематической цепи манипулятора, где index – номер звена в links;
* void Rotate() – поворот манипулятора.
* Point3D rotatePoint(Point3D, double phi) – перенос точки на заданный угол;
* Point3D endPoint(List<Link> links, int index) – получение конечной точки звена с индексом index из links (решение прямой кинематической задачи).



Рисунок 3.4 – Диаграмма классов подсистемы «Рука-Манипулятор»

Библиотека подсистемы «ГА» (GeneticAlgorithmsLib.dll) состоит из четырех классов: Chromosome – описание особи для алгоритма HillClombing, класс HillClombingGA – содержит собственно реализацию этого алгоритма, класс StandartGA – реализация традиционного ГА. StandartGaDynamic – традиционный ГА для решения ОКЗ при движении по траектории. Диаграмма классов этой подсистемы представлена на рисунке 3.5.

Класс *Chromosome*:

* internal Fitness fitness – функция приспособленности, где Fitness – делегат вида public delegate double Fitness(char[] genes);

Делегат – это тип, который безопасно инкапсулирует метод.

* private char[] genes – гены хромосомы в двоичном виде.
* bool Equals(object obj) – переопределение метода Equals для сравнения двух объектов класса Chromosome.



Рисунок 3.5 – Диаграмма классов подсистемы «ГА»

Класс *HillClimbingGA*:

* Chromosome chromosome – экземпляр класса Chromosome для хранения хромосомы;
* Chromosome mutate(double probability) – мутация хромосомы с вероятностью probability.
* char[] Evolve(double accuracy) – эволюция, в результате которой находится решение ОКЗ; accuracy – точность решения.
* Chromosome shake() – изменение геном хромосомы при долгом блуждании возле «пика».

Класс *StandartGA*:

* Fitness fitness – делегат для расчета приспособленности особи;
* double[][] population – текущая популяция;
* double[][] restrictions – ограничения для генерации значений генов;
* double[] IntermediateRecombination (double[] firstParent, double[] secondParent) – реализация промежуточной рекомбинации для получения потомка;
* void select() – отбор родителей, создание новой популяции;
* void sortPopulation() – сортировка особей в соответствии с их приспособленностью;
* double[] Evolve(double accuracy) – эволюция, в результате которой находится решение ОКЗ; accuracy – точность решения.

Класс StandartGaDynamic переопределяет метод Evolve(), в нем изменяются некоторые параметры условий работы ГА.

Для удобства преобразования хромосомы, представленной массивом типа char, в массив типа double (массив углов поворота звеньев) используется библиотека DataConverter.lib, в которой содержится статический класс ConverterForGA, предоставляющий методы для такого преобразования как в прямом, так и в обратном порядке. Кроме того, в данной библиотеке содержится класс TrajectoryConverter для преобразования траектории в набор точек.



Рисунок 3.6 – Диаграмма классов подсистемы «Конвертер для ГА»

Подсистема «Сбор статистики» (StatisticsGALib.dll) представлена в виде библиотеки со статическим классом Statistics (рисунок 3.7), с помощью которого можно сохранять и получать статистику. Вся статистика сохраняется в файлах.

Подсистемы «Симулятор» и «Решатель ОКЗ» реализованы в виде классов InverseKinematicProblem (рисунок 3.8) и RoboArmSimulator (рисунок 3.9) соответственно.



Рисунок 3.7 – Диаграмма классов подсистемы «Статистика»

В задачи класса InverseKinematicProblem входит вызов выбранного пользователем ГА для решения ОКЗ для заданной точки (метод FindThetas() – для тестирования; метод Solve() для решения ОКЗ модели робота-манипулятора, представленной в главном окне приложения), а также для движения по траектории (метод SolveDynamic() – решение задачи движения путем решения ОКЗ для точек, на которые разбита заданная траектория, analiseQuaters() - анализ четвертей, в которых находятся углы наклона полученных решений, addRestrictions() – добавление дополнительных ограничений после анализа четвертей). Кроме того, метод данного класса (RotateEndPoint()) проецирует точку, заданную пользователем для трехмерного пространства, в двумерную плоскость.



Рисунок 3.8 – Диаграмма классов подсистемы «Решатель ОКЗ»

Класс RoboArmSimulator строит модель руки-манипулятора (метод VisualModel()), предоставляет возможности для удобного просмотра этой модели c помощью изменения позиции «камеры» (ChangeCameraPosition()). Также в этом классе к модели робота может добавляться облако точек решений, полученных ГА в процессе поиска искомого решения.



Рисунок 3.9 – Диаграмма классов подсистемы «Симулятор»

Подсистема «Тестирование» (рисунок 3.10) реализована классами TestTimeAccuracy, TestWindow и TextWindow(вспомогательный класс для вывода дополнительной информации о тестах TextWindow), предоставляющими возможности настройки тестов, их проведения, а также визуализации результатов.

Как отмечалось ранее, построение графиков осуществляется с помощью библиотеки OxyPlot. Для удобства их построения реализованы классы ErrorPlotModel (график изменения ошибки решения ОКЗ с помощью ГА – конечная точка манипулятора для лучшей особи популяции на каждой итерации), TimeVsAccuracyPlot (отображение результатов тестирования зависимости времени решения ОКЗ с помощью ГА от точности вычислений), DynamicPlotModel (графики изменения углов наклона звеньев манипулятора при перемещении по траектории. Построение происходит для модели манипулятора, соответствующей роботу-манипулятору RoboArm). Диаграмма этих классов представлена на рисунке 3.11.



Рисунок 3.10 – Диаграмма классов подсистемы «Тестирование»



3.11 – Диаграмма классов для построения графиков

Подсистема «Динамика» реализована в виде классов MovemetParametersWindow (настройка параметров движения оп траектории) и DynamicPlotsWindow (отображение графиков, построенных с помощью DynamicPlotModel). Диаграмма классов данной подсистемы представлена на рисунке 3.12.



Рисунок 3.12 – Диаграмма классов подсистемы «Динамика»

* 1. Испытание ПС

Изначально в приложение встроены параметры робота-манипулятора RoboArm. При его запуске появляется главное окно (рисунок 3.13).

После ввода данных (точки с координатами x, y, z, где ось Y направлена вверх), выбора алгоритма решения ОКЗ и нажатия кнопки «Найти» окно примет вид, представленный на рисунке 3.14.

Подсистема «Тестирование» была разработана специально для определения трудоемкости алгоритмов и сравнения их с другими алгоритмами. Проведем ряд тестов и определим конкурентоспособность наших алгоритмов. Для этого протестируем робота-манипулятора, состоящего из 10 звеньев и не имеющего ограничений на их углы наклона (параметры манипулятора представлены на рисунке 3.15).

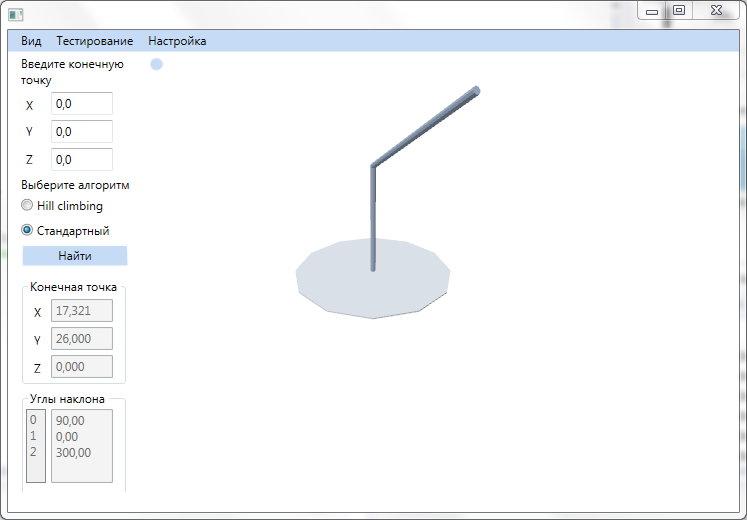


Рисунок 3.13 – Пример главного окна при запуске приложения

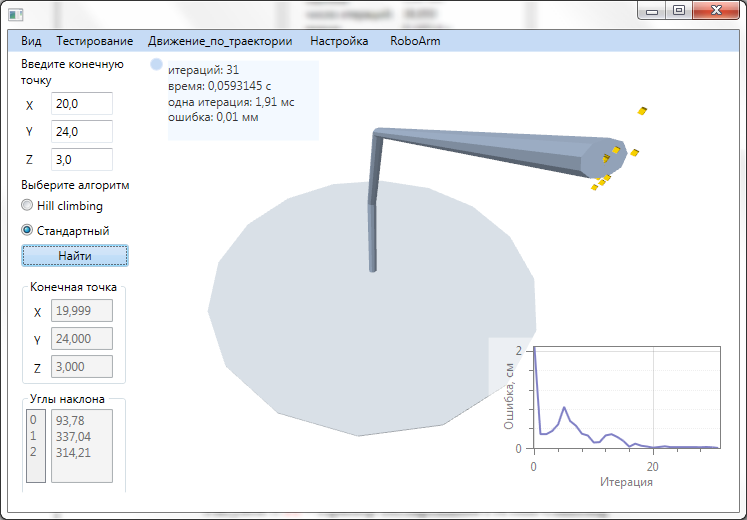


Рисунок 3.14 – Пример отображения решения ОКЗ

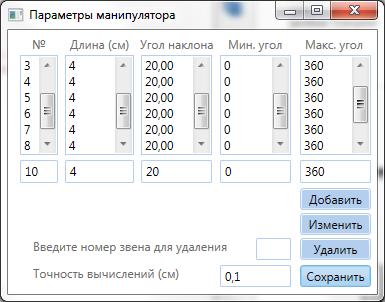


Рисунок 3.15 – Параметры манипулятора для тестирования

Проведем тесты для точки c координатами (20,0; 24,0; 3,0) для обоих алгоритмов. Результаты показаны на рисунках 3.16, 3.17.

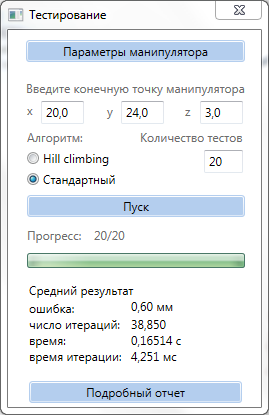


Рисунок 3.16 – Пример тестирования стандартного ГА

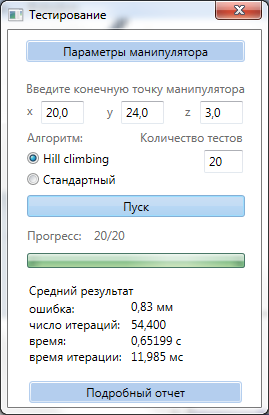


Рисунок 3.17 – Пример тестирования ГА Hill Climbing

Как видно из полученных результатов (рисунок 3.16, 3.17), средняя ошибка по тестам каждого ГА не превышает заданной ошибки, что говорит о том, что тесты прошли успешно.

Полученные результаты сравним с результатами других алгоритмов (таблица 3.1), взятыми из статьи [1] (таблица 3.1). Все алгоритмы, представленные в данной статье, выполнялись в среде Matlab. Они запускались на ПК с Pentium 2 Duo 2.2 GHz, как и алгоритмы, представленные в этом проекте.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Число итераций | Время выполнения, сек | Время на одну итерацию, мсек | Итераций в секунду |
| FABRIK | 15,461 | 0,01328 | 0,86 | 1164 |
| CCD | 26,308 | 0,12356 | 4,69 | 213 |
| Стандартный ГА | 38,850 | 0,16514 | 4,251 | 235 |
| Hill Climbing | 54,400 | 0,65199 | 11,985 | 83 |
| Jacobian Transpose | 1311,190 | 12,98947 | 9,90 | 101 |
| Jacobian DLS | 998,648 | 10,48051 | 10,49 | 95 |
| Jacobian SVD-DLS | 808,797 | 9,29652 | 11,50 | 87 |
| FTL | 21,125 | 0,02045 | 0,97 | 1033 |
| Triangulation | 1,000 | 0,05747 | 57,47 | 21 |

Таблица 3.1. Сравнение результатов тестирования алгоритмов.

Как видно из таблицы, наиболее быстрыми являются алгоритмы FABRIK и CCD. ГА следуют сразу за ними. Они явно обходят по скорости алгоритмы, использующие якобианы, в особенности стандартный ГА.

Алгоритмы FTL и Triangulation находятся в конце таблицы, т.к. они не способны решать ОКЗ при наличии ограничений для звеньев манипулятора. Однако в отсутствии этих ограничений показывают хороший результат.

Также можно сделать вывод, что алгоритм Hill Climbing не оптимально применять для решения ОКЗ, т.к. тратится наибольшее время на выполнение одной итерации (исключение составляет алгоритм Triangulation, но у этого алгоритма всего одна итерация).

Определим зависимость времени выполнения ГА от заданной точности решения (рисунок 3.18, 3.19).

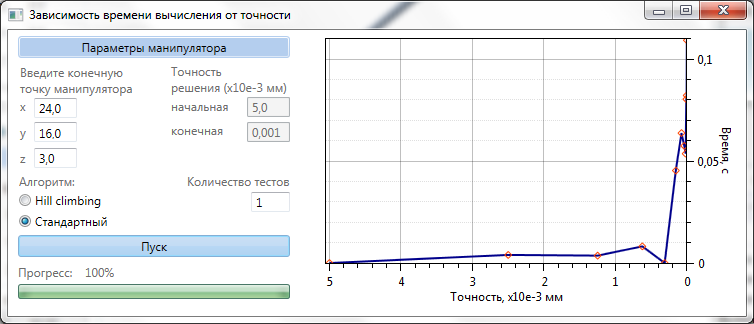


Рисунок 3.18– Пример запуска тестирования зависимости времени вычисления от точности для стандартного ГА

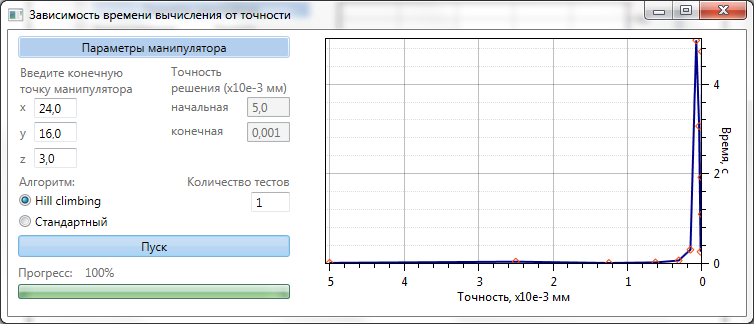


Рисунок 3.19 – Пример запуска тестирования зависимости времени вычисления от точности для ГА Hill Climbing

Как видно из рисунков 3.18 и 3.19, время, необходимое на поиск решения, «в основном» растет при увеличении заданной точности (уменьшении значения ошибки вычислений). «В основном» потому, что ГА – алгоритмы, основанные на манипулировании случайными числами, поэтому угадать, сколько точно времени потребуется на их выполнение практически невозможно.

Далее покажем пример работы программы для перемещения манипулятора по траектории. На рисунке 3.20 показан пример настройки параметров движения для перемещения по прямой.

На рисунке 3.21 изображен результат расчета ОКЗ для перемещения по прямой. Как видно из статистики, было выполнено 5 пересчетов, в результате чего положения манипулятора были улучшены. Глядя на окно графиков изменения наклона звеньев от итерации, можно заметить, что сильных выбросов не присутствует, что также видно из промежуточных положений манипулятора.

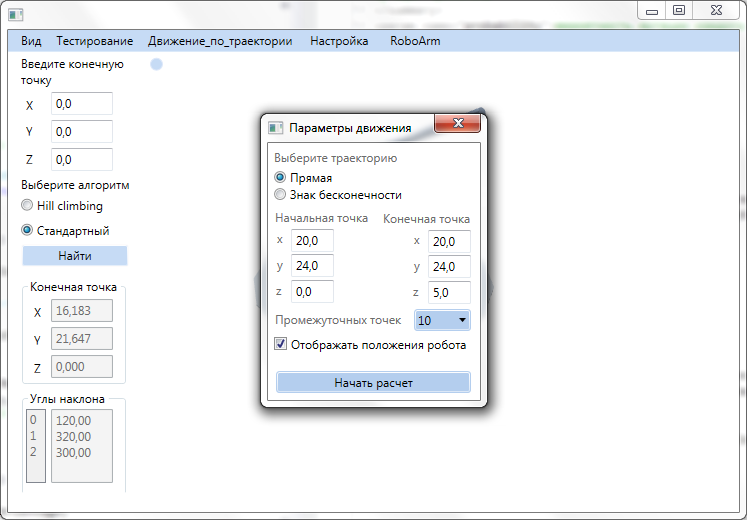


Рисунок 3.20 – Пример настройки параметров движения по прямой

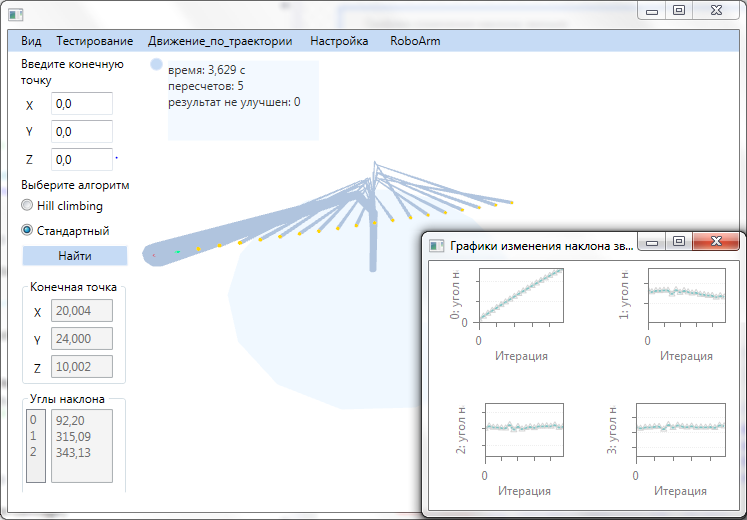


Рисунок 3.21– Результаты расчета примера перемещения по прямой

На рисунке 3.22 показан пример настройки параметров движения по траектории знака бесконечности. Для примера в этот раз не будем отображать промежуточные положения звеньев. Результат расчета для данного случая показан на рисунке 3.23.

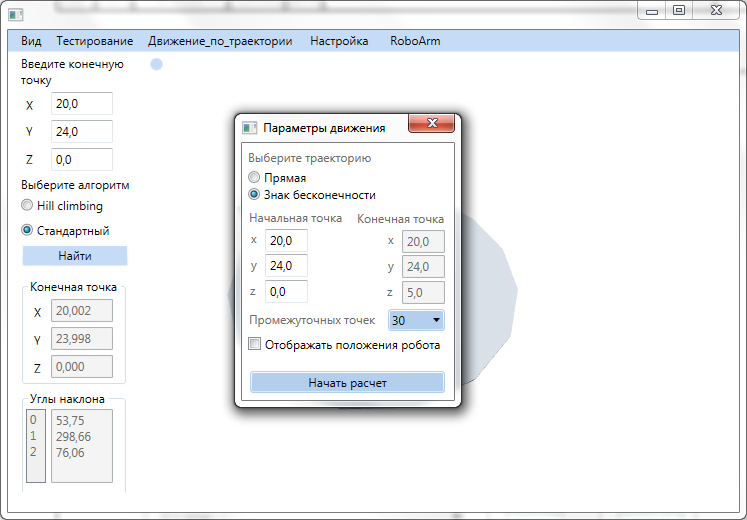


Рисунок 3.22 – Пример настройки параметров движения по траектории знака бесконечности

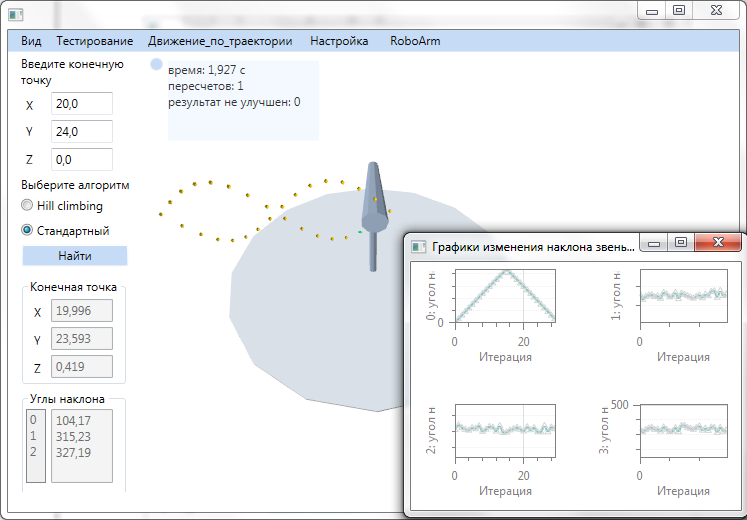


Рисунок 3.23 – Результат расчета примера движения по траектории знака бесконечности

1. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ
   1. Исходные данные для расчета экономического эффекта

Наименование проекта – «Разработка программной системы контроля досту-па с поддержкой измерения температуры на основе платформы Cantonk S02T».

Среда разработки ПО – PyCharm Community Edition (Python). ПО функционального назначения.

Разработка программного средства предусматривает проведение всех стадий проектирования (техническое задание, эскизный проект, технический проект, рабо-чий проект), относится ко второй группе сложности.

Последовательность расчетов:

1. Расчет объема функций программного модуля

2. Расчет заработной платы разработчиков и полной себестоимости

3. Расчет отпускной цены и чистой прибыли

* 1. Расчет объема функций программного обеспечения

Общий объем ПО (*Vo*) определяется исходя из количества и объема функций, реализуемых программой:

(4.1)

где – общий объем ПС;

– объем функций ПС;

– общее число функций.

Расчет общего объема ПО (количества строк исходного кода (LOC)) предполагает определение объема по каждой функции. В том случае, когда на стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций, данный объем может быть получен на основании ориентировочной (прогнозной) оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения нормативов по каталогу функций.

На основании информации о функциях разрабатываемого ПО по каталогу функций определяется общий объем ПО. В зависимости от организационных и технологических условий, в которых разрабатывается ПО, корректируется объем на основе экспертных оценок.

Уточненный объем ПО (*Vу*) определяется по формуле:

(4.2)

где – уточненный объем отдельной функции в строках исходного кода (см. таблицу 4.1).

Таблица 4.1 – Перечень и объем функций программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер функции** | **Наименование (содержание) функции** | **Объем функции строк исходного кода (LOC)** | |
| **По каталогу** | **Уточненный** |
| **1. Ввод, анализ входной информации, генерация кодов и процессор входного языка** | | | |
| 107 | Организация ввода/вывода информации в интерактивном режиме | 280 | 222 |
| **2. Формирование, введение и обслуживание баз данных** | | | |
| 203 | Обработка наборов и записей базы данных | 2370 | 663 |
| 205 | Обслуживание базы данных в интерактивном режиме | 4840 | 541 |
| 206 | Манипулирование данными | 7860 | 940 |
| **3. Формирование и обработка файлов** | | | |
| 301 | Формирование последовательного файла | 590 | 218 |
| 303 | Обработка файлов | 1050 | 97 |
| **5. Управление ПО, компонентами ПО и внешними устройствами** | | | |
| 503 | Управление внешними устройствами и объектами | 4730 | 554 |
| 506 | Обработка ошибочных и сбойных ситуаций | 1540 | 134 |
| 507 | Обеспечение интерфейса между компонентами | 1680 | 5587 |
| **6. Тестирование, проведение тестовых испытаний прикладных программ, вспомогательные программные функции** | | | |
| 601 | Проведение тестовых испытаний прикладных программ в интерактивном режиме | 3780 | 227 |
| **7. Расчетные задачи, формирование и вывод на внешние носители документов сложной формы и файлов** | | | |
| 703 | Расчет показателей | 420 | 34 |
| 707 | Графический вывод результатов | 420 | 256 |

Учитывая информацию, указанную в таблице 4.1, о функциях разрабатываемого программного обеспечения, уточненный объем ПО () составил 9 473 строк исходного кода (LOC) вместо предполагаемого количества строк 22 810.

* 1. Расчет заработной платы разработчиков и полной себестоимости

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

* заработную плату исполнителей ()
* отчисления на социальные нужды ();
* материалы и комплектующие изделия (Рм);
* спецоборудование (Рс);
* машинное время (Рмв);
* расходы на научные командировки (Рнк):
* прочие прямые расходы (Рпр);
* накладные расходы (Рнр);
* затраты на освоение и сопровождение программного средства(Ро и Рсо).

Полная себестоимость (Сп) разработки программного продукта (ПП) рассчитывается как сумма расходов по всем статьям с учетом рыночной стоимости аналогичных продуктов. Определяется по формуле 4.3:

(4.3)

Рассчитаем значения по каждой из статей:

1. Заработная плата исполнителей

Расчёт заработной платы каждого из разработчиков ПП начинается с определения:

* продолжительности времени разработки ФРВ, которое устанавливается экспертным путём с учётом сложности, новизны ПП и фактически затраченного времени
* количества разработчиков ПП, количество которых составляет 2 человека.

Для данного дипломного проекта один разработчик (инженер-программист) и научный руководитель.

Предприятие-заказчик (БрГТУ) является бюджетной организацией, следовательно для расчета заработной платы разработчиков принимается тарифная система (ТС) «Об оплате труда работников бюджетных организаций».

Месячная заработная плата будет состоять из оклада и стимулирующих и компенсирующих выплат (см. формулу 4.4):

(4.4)

Тарифный оклад рассчитывается по формуле 4.5:

(4.5)

где Базовая ставка на 1 января 2021 года составляет 195 бел.р;

Тарифный коэффициент для инженера программиста 5 разряда равен 1.29, тарифный коэффициент для научного руководителя (должность – старший преподаватель) 14 разряда равен 2.31.

После того, как определена месячная заработная плата специалиста, необходимо пересчитать её на фактическое число дней работы конкретного специалиста над проектом по формуле 4.6:

, (4.6)

где 21.4 – среднемесячное количество рабочих дней в 2021 году при 5-дневной рабочей неделе

дней для инженера-программиста и для научного руководителя

Результаты вычислений внесём в таблицу 4.2.

Таблица 4.2 – Расчет заработной платы

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Категории работников** | **Тарифный разряд** | **Тарифный коэффициен** | **Фонд рабочего времени, дни** | **Премия, %** | **Надбавка за работу по контракту, %** | **Надбавка за стаж, %** | **Надбавка за характер**  **труда, %** | **Надбавка за высокие достижения в труде, %** | **Заработная плата,**  **бел. руб.** | |
| **Оклад** | **Всего** |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Научный руководитель | 14 | 2,31 | 10 | 5 | - | 20 | - | - | 450,45 | 239,24 |
| Инженер-программист | 5 | 1,29 | 50 | 5 | - | 10 | - | - | 251,55 | 662,69 |

Заработная плата научного руководителя:

Заработная плата инженера-программиста:

Таким образом, заработная плата инженера-программиста за 50 дней составляет 662,69 бел. руб., а заработная плата научного руководителя за 10 дней составила 239,24 бел. руб. Общие расходы на заработную плату составили 901,93 бел. руб.

1. Отчисления на социальные нужды

Отчисления на социальные нужды включают предусмотренные законодательством отчисления в фонд социальной защиты (34 %) и фонд обязательного страхования (0,6 %) в процентах от заработной платы рассчитаем по формуле 4.7:

(4.7)

(бел.руб)

(бел.руб.)

(бел.руб.)

1. Материалы и комплектующие изделия

По статье «Материалы и комплектующие изделия» () отражаются расходы на бумагу, картридж и красящие ленты для лазерного принтера HP LaserJet 1020, необходимые для разработки ПП. Норма расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода. В данном дипломном проекте не рассчитывается.

1. Спецоборудование

Расходы по статье «Спецоборудование» () включает затраты на приобретение технических и программных средств специального назначения, необходимых для разработки ПП, включая расходы на проектирование, изготовление, отладку и др. В данном дипломном проекте для разработки ПП приобретением является Устройство распознавание лиц и контроля температуры Cantonk S02T, стоимость которого 1900 бел. рублей.

1. Машинное время

Расходы по статье «Машинное время» () включают оплату машинного времени, необходимого для разработки и отладки ПП. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени. Машинное время рассчитывается по формуле 4.8:

, (4.8)

где – цена одного машино-часа, тыс. бел. руб.; примем равным 0,8.

– уточнённый общий объём функций строк исходного кода;

– норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,7.

Подставим значения в формулу 4.8:

(бел.руб.)

1. Расходы на научные командировки

Расходы на научные командировки () берутся либо по смете научных командировок, разрабатываемой на предприятии, либо в процентах от основной заработной платы исполнителей (10-15%). Так как научные командировки не предусмотрены, данная статья не рассчитывается.

1. Прочие прямые расходы

Расходы по статье «Прочие затраты» () включают затраты на приобретение специальной научно-технической информации и специальной литературы. Определяются в процентах к основной заработной плате исполнителей (10-15 %) и рассчитываются по формуле (4.9):

(4.9)

где – норматив прочих затрат

Примем Подставим значения в формулу 4.9:

(бел. руб.)

1. Накладные расходы

Затраты по статье «Накладные расходы» () связаны с содержанием вспомогательных хозяйств, опытных производств, а также с расходами на общехозяйственные нужды. Определяются по нормативу в процентах к основной заработной плате исполнителей (для бюджетных организаций норматив устанавливается в пределах 50-100%) и рассчитываются по формуле 4.10:

(4.10)

где - норматив накладных расходов, который измеряется в процентах

Примем Подставим значения в формулу 4.10:

(бел.руб.)

Сумма вышеперечисленных расходов по статьям (пп. 1 - 8) на ПП служит исходной базой для расчёта затрат на освоение и сопровождение ПП (см. формулу 4.11):

(4.11)

Подставим все найденные значения в формулу 4.11:

(бел.руб.)

1. Затраты на освоение и сопровождение программного средства ( и ).

Организация-разработчик участвует в освоении ПП и несёт соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Для упрощения расчётов затраты на освоение () определяются по установленному нормативу () от суммы затрат по пунктам 1 – 8 и рассчитываются по формуле 4.12:

(4.12)

где - норматив отчислений на освоение

Примем Подставим значения в формулу 4.12:

(бел.руб.)

Организация-разработчик осуществляет сопровождение ПП и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Для упрощения расчётов затраты на сопровождение () определяются по установленному нормативу ( = 5-10 %) от суммы затрат по пунктам 1 – 8 и рассчитываются по формуле 4.13:

(4.13)

где – норматив отчислений на сопровождение

Примем Подставим значения в формулу 4.13:

(бел.руб.)

Полная себестоимость найдем по формулу 4.14:

(4.14)

Подставим все найденные значения в формулу 4.12:

(бел.руб.)

Таблица 4.3 – Расчет себестоимости ПП

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **пп** | **Наименование статей затрат** | **Норматив** | **Расчет, бел. руб.** |
| 1 | Зарплата, всего | - | 901,93 |
| 2 | Отчисления на социальные нужды | 34,6 | 312,07 |
| 3 | Спецоборудование | - | 1900 |
| 4 | Материалы | Не применялись | - |
| 5 | Машинное время | - | 53,05 |
| 6 | Научные командировки | Не планировались | - |
| 7 | Прочие затраты | 10 | 90,19 |
| 8 | Накладные расходы | 50 | 450,97 |
| 9 | Сумма затрат | - | 3708,21 |
| 10 | Затраты на освоение ПП | 7 | 259,57 |
| 11 | Затраты на сопровождение | 10 | 370,82 |
| 12 | Полная себестоимость | - | 4338,6 |

* 1. Расчет отпускной цены и чистой прибыли

Отпускная цена предприятия – это цена, при которой обеспечивается возмещение текущих затрат производства и получение прибыли. Необходимо рассчитать отпускную цену для разработанной системы.

Стоимостная оценка системы включает в себя следующие статьи:

* + полную себестоимость разрабатываемого устройства ()
  + плановую прибыль от реализации на единицу ()
  + налог на добавленную стоимость ()

Полная себестоимость составила 4338,6 бел.руб.

1. плановая прибыль

Рентабельность и прибыль создаваемого ПП определяется исходя из результатов анализа рыночных условий, переговоров с заказчиком и согласования с ним отпускной цены. Плановая прибыль () рассчитывается по формуле 4.15:

(4.15)

где – уровень рентабельности ПП, % (можно принять в размере 10 - 30%)

Примем . Подставим значения в формулу 4.15:

(бел.руб.)

1. прогнозируемая цена ПП без налогов

После расчета прибыли от реализации определяется прогнозируемая цена ПП () без налогов по формуле 4.16:

(4.16)

Подставим значения в формулу 4.16:

(бел.руб.)

1. Отпускная цена (цена реализации)

Отпускная цена (цена реализации) ПП () включает налог на добавленную стоимость (в настоящее время НДС- 20 %) и рассчитывается по формуле 4.17:

(4.17)

Налог на добавленную стоимость () – косвенный налог, форма изъятия в бюджет государства части стоимости устройства, который рассчитывается по формуле 4.18:

(4.18)

Подставим значения в формулы 4.17 и 4.18:

(бел.руб.)

(бел.руб.)

1. Прибыль от реализации ПП за вычетом налога на прибыль

Прибыль от реализации ПП за вычетом налога на прибыль () является чистой прибылью, остается организации разработчику и представляет собой экономический эффект от создания нового программного продукта. Рассчитывается по формуле 4.19:

(4.19)

где – ставка налога на прибыль (в настоящее время = 18%).

(бел.руб.)

Подставим значения в формулу 4.19:

(бел.руб.)

Таблица 4.4– Отпускная цена ПП и чистая прибыль

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Наименование** | **Норматив, %** | **Расчет, бел. руб.** |
| 1 | Полная себестоимость | - | 4338,6 |
| 2 | Прибыль | 20 | 867,72 |
| 3 | Прогнозируемая цена без налогов | - | 5206,32 |
| 4 | НДС | 20 | 1041,26 |
| 5 | Отпускная цена | - | 6247,58 |
| 6 | Чистая прибыль | 18 | 711,53 |

Итак, определены основные экономические показатели:

• полная себестоимость - 4338,6 бел. руб;

• прогнозируемая цена - 6247,58 бел. руб;

• чистая прибыль - 711,53 бел. руб.

Данное ПО позволяет контролировать доступ лиц, а также получать информацию о температуре посетителя, тем самым понижая шансы распространению заболеваний при том, что посетитель не тратит больше времени, чем при обычном прохождении контроля доступа, когда как проверка температуры классическими средствами занимает значительно больший промежуток времени. К тому же данное устройство дешевле относительно других платформ со схожим функционалом. Совокупность этих фактором делает экономический эффект разработки и внедрения данной системы положительным. Также данную ПС можно продавать в качестве готового решения в другие организации и компании отдельно от самих устройств Cantonk S02T, которые будут закупаться организациями в том количестве, в каком им будет необходимо.

1. ЭНЕРГО- И РЕСУРСОСБЕРЕЖЕНИЕ

Энергосбережение – комплекс научных, технических, организационных мероприятий, направленных на уменьшение потерь энергии у потребителя и уменьшении затрат при производстве энергии у производителя. Основным направлением энергосбережения является получение максимального эффекта при минимальных затратах.

Основной причиной повышения энергоэффективности и целью энергосбережения, является истощаемость энергетических ресурсов и их высокая стоимость. Таким образом, повышение энергетической эффективности в сфере производства и применения компьютерной техники должно быть направлено на:

* разработку и внедрение энергосберегающих технологий в производство;
* оптимизацию характеристик энергопотребления различных классов компьютерной техники;
* использование энергосберегающих технологий утилизации отслуживших устройств.

Причины, вызывающие необходимость политики энергосбережения в Республике Беларусь:

* + - низкая обеспеченность собственными ресурсами. За счёт собственных топливно-энергетических ресурсов республика обеспечивает потребности энергии лишь на 15%
    - высокая доля природного газа. Доля природного газа не должна превышать 65%, т.к. электростанции работают в режиме постоянной подачи топлива, а отсутствие альтернативы требует больших запасов резервного топлива либо строительства газовых хранилищ. Доля импортного природного газа составляет 78%
    - высокая степень износа основных фондов энергетического комплекса страны. Более 51% основного генерирующего оборудования страны выработало свой ресурс. Протяжённость воздушных линий напряжением от 220 до 330 кВ со сроком эксплуатации более тридцати лет в целом по республике составляет более 80%. Средний коэффициент износа воздушных линий составляет 52%
    - физический износ оборудования подстанций, находящихся в эксплуатации от пятнадцати до двадцати лет и более превышает 60%
    - ежегодно требуется замена порядка 100 км изношенных участков тепловых сетей, протяжённость которых составляет 4800 км
    - импорт топливно-энергетических ресурсов преимущественно из одной страны
    - большие затраты на импортируемые энергоресурсы
    - недостаточные инвестиции в топливно-энергетический комплекс страны
    - высокая энергоёмкость энергетики Республики Беларусь
    - нехватка электроэнергии

Важным компонентом достижения энергоэффективности является программное обеспечение, в том числе программная реализация системы управления питанием и ПО для исследования возможностей энергосбережения при решении прикладных задач. Для многоядерных платформ значение ПО становится еще более важным, так как учет особенностей многоядерной и многопоточной организаций вычислительной системы влияет не только на многозадачную производительность таких платформ, но и на эффективность использования электроэнергии.

Ресурсосбережение – совокупность мер по бережливому и эффективному использованию фактов производства, которые обеспечиваются за счёт:

* + - использования ресурсосберегающих и энергосберегающих технологий;
    - снижения фондоёмкости и материалоёмкости продукции;
    - повышения производительности труда;
    - повышения качества продукции;
    - рационального применения труда;
    - использования выгод международного разделения труда.

Выделяют следующие основные требования ресурсосбережения:

* + - требования к содержанию ресурсов, определяющие совершенство процес-сов, продукции и услуг, в ходе которых задействованы те или иные ресурсы предприятия;
    - требования ресурсоемкости (по технологичности), определяющие воз-можность и условия для достижения оптимальных затрат ресурсов при изготовлении, ремонте и утилизации продукции.

Соблюдение ресурсосбережения – важная характеристика качества техники и применяемых технологических процессов. Необходимость ресурсосбережения вызвана дефицитом многих видов ресурсов, истощением их запасов в природе, значительными стоимостными затратами их добычи и другими факторами. Ресурсосбережение способствует росту эффективности экономики, а также повышению её конкурентоспособности на мировом рынке. Ресурсосберегающая технология предполагает, что производство и реализация конечных продуктов выполняется с минимальным расходованием вещества и энергии на всех стадиях ресурсного цикла. При этом воздействие на природные системы и человека должно быть наименьшим.

Робототехника участвует в процессе энерго- и ресурсосбережения. Она, с одной стороны, избавляет человека от тяжелого и опасного труда, с другой стороны освобождает производство от непосредственного участия в нем людей и тем самым снимает связанные с их участием ограничения, препятствующие дальнейшей интенсификации производственных процессов и внедрения новых технологий, принципиально не допускающих присутствия людей.

В любой области применения роботов их необходимо рассматривать как компонент всего производственного комплекса, включающего другое основное и вспомогательное технологическое оборудование, объединенное общей системой управления. Использование роботов оказывает существенное влияние на такие важные экономические характеристики, как производительно труда, объем производства продукции, себестоимость, рентабельность, фондоотдача.

В условиях роботизации происходит абсолютное и относительное сокращение численности производственных рабочих. Под относительным сокращением численности понимается возможность повысить объем производства при той же численности производственных рабочих благодаря увеличению годового эффективного фонда времени работы оборудования в результате использования роботов.

Социальные аспекты роботизации, связанные с улучшением условий труда, ликвидацией тяжелых, опасных и вредных для здоровья видов работ, с повышением общей культуры производства, должны быть количественно оценены и учтены в расчетах экономической эффективности применения роботов. Роботизация позволяет уменьшить количество профессиональных заболеваний, снизить травматизм, сократить затраты на лечение и мероприятия по охране труда и технике безопасности.

ЗАКЛЮЧЕНИЕ

В результате дипломного проектирования разработана система для решения ОКЗ на основе ГА.

Система обеспечивает решение следующих задач: трехмерное моделирование и визуализацию модели робота-манипулятора; ввод, редактирование параметров модели робота-манипулятора; вычисление прямой задачи кинематики; решение ОКЗ с помощью ГА; визуализация результатов решения ОКЗ; построение траектории движения манипулятора на основе полученных решений ОКЗ (движение по прямой и по траектории символа бесконечности); взаимодействие с реальным роботом-манипулятором RoboArm.

Для решения ОКЗ выбрано два ГА (стандартный ГА, использующий дискретную рекомбинацию для получения новой особи; Hill Climbing, основанный на постоянной мутации одной особи). В результате тестирования данных алгоритмов выяснилось, что они по скорости обходят алгоритмы, использующие якобианы (таблица 3.1). Также можно сделать вывод, что алгоритм Hill Climbing не оптимально применять для решения ОКЗ, т. к. тратится наибольшее время на выполнение одной итерации (из рассмотренных алгоритмов исключение составляет алгоритм Triangulation, но у этого алгоритма всего одна итерация).

Для реализации приложения использован API-интерфейс WPF. В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. Построение и визуализация модели робота-манипулятора осуществлены с помощью встроенные средства WPF-приложений – классов, содержащиеся в пространстве имен System.Windows.Media.Media3D.

Произведен расчет технико-экономических показателей: общая трудоемкость разработки программного обеспечения, основная и дополнительная заработная плата разработчиков программного обеспечения, себестоимость и отпускная цена программного обеспечения.

СПИСОК СОКРАЩЕНИЙ

ГА – генетический алгоритм

ОКЗ – обратная кинематическая задача

MVS – Microsoft Visual Studio

WPF – Windows Presentation Foundation

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Andreas Aristidou, Joan Lasenby. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. 2010. 18 c.
2. Боголюбов А.Н., Никитин Д.А. Популярно о робототехнике/ Отв. ред. В.Д. Новиков. Киев: Наук, думка, 1989.200 с.
3. Вукобратович М., Стокич Д. Управление манипуляционными роботами: теория и приложения. М.: Наука, 1985. 384 с.
4. Панченко Т.В. Генетические алгоритмы/Отв. ред. Тарасевич Ю.Ю. Астраханский университет, 2007. 88 с.
5. Программирование на языке C# [Электронный ресурс]/Виртуальный справочник – Режим доступа: https://msdn.microsoft.com.

БЛОК-СХЕМЫ АЛГОРИТМОВ



Рисунок А.1 – Блок-схема алгоритма Hill Climbing



Рисунок А.2 – Блок-схема алгоритма решения задачи движения по траектории